Wireless Distributed Computing in Cloud Computing Networks

Dinesh Datla

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Jeffrey H. Reed, Co-Chair
Tamal Bose, Co-Chair
Allen B. MacKenzie
Jung-Min Park
Madhav V. Marathe

April 22, 2013
Blacksburg, Virginia

Keywords: distributed computing, wireless cloud computing, wireless networks
Copyright 2013, Dinesh Datla

Wireless Distributed Computing in Cloud Computing Networks

Dinesh Datla

(ABSTRACT)

The explosion in growth of smart wireless devices has increased the ubiquitous presence of computational resources and location-based data. This new reality of numerous wireless devices capable of collecting, sharing, and processing information, makes possible an avenue for new enhanced applications. Multiple radio nodes with diverse functionalities can form a wireless cloud computing network (WCCN) and collaborate on executing complex applications using wireless distributed computing (WDC). Such a dynamically composed virtual cloud environment can offer services and resources hosted by individual nodes for consumption by user applications. This dissertation proposes an architectural framework for WCCNs and presents the different phases of its development, namely, development of a mathematical system model of WCCNs, simulation analysis of the performance benefits offered by WCCNs, design of decision-making mechanisms in the architecture, and development of a prototype to validate the proposed architecture.

The dissertation presents a system model that captures power consumption, energy consumption, and latency experienced by computational and communication activities in a typical WCCN. In addition, it derives a stochastic model of the response time experienced by a user application when executed in a WCCN. Decision-making and resource allocation play a critical role in the proposed architecture. Two adaptive algorithms are presented, namely, a workload allocation algorithm and a task allocation - scheduling algorithm. The proposed algorithms are analyzed for power efficiency, energy efficiency, and improvement in the execution time of user applications that are achieved by workload distribution. Experimental results gathered from a software-defined radio network prototype of the proposed architecture validate the theoretical analysis and show that it is possible to achieve 80% improvement in execution time with the help of just three nodes in the network.

This work has received support from the Defense Advanced Research Projects Agency (NSWCDD Contract number: N00178-09-D-3017), the Office of Naval Research (Grant Number N300014-07-01-0536), the National Science Foundation sponsored Wireless Internet Center for Advanced Technology, and Virginia Tech's Institute for Critical Technology and Applied Science.

To my wife Divya, my family, and my advisors ....

Acknowledgments

It has been an excellent learning experience and privilege pursuing my doctoral research under the guidance of Dr. Jeffrey H. Reed and Dr. Tamal Bose. I am very fortunate to have the opportunity to work on cutting-edge wireless technologies, namely software radios, cognitive radios, dynamic spectrum access, green communications, wireless distributed computing, and wireless cloud computing, under their guidance. In addition to teaching me the technical skills that are necessary to be a successful researcher, they have given me ample opportunities to develop my soft skills, such as proposal writing, customer interaction, and general project management.

I am greatly privileged to have Dr. Madhav V. Marathe, Dr. Allen B. MacKenzie, and Dr. Jung-Min Park in my dissertation committee. I have received invaluable feedback from each one of them at different stages of my dissertation research. I thank Dr. Bruce Fette of DARPA for his interest and feedback on our work on wireless distributed computing.

I thank Dr. S. M. Shajedul Hasan and Dr. Haris Volos for supporting my research while working on various funded projects related to wireless distributed computing. They have also provided invaluable feedback on the architectural design proposed in this dissertation. I also thank my lab-mates Benjamin Hilburn, Jeffery Wilson, and Sahana Raghunandan for their help while developing the wireless distributed computing prototype. Benjamin Hilburn assisted me with the implementation of the prototype. He developed the skeletal software that enabled me to implement the individual blocks in the architecture. Jeffery Wilson assisted me in implementing a SQL database and image processing software for the

prototype. The image processing software used in the prototype was initially developed using OpenCV by Sahana Raghunandan. It has been a memorable experience working with Xuetao Chen, Tom Tsao, and Somava Bera on wireless distributed computing. I thank Dr. Carl Dietrich for his invaluable feedback on my manuscript on wireless distributed computing which was submitted for publication in the IEEE Communications Magazine. I was also closely associated with Dr. Kyung K. Bae and Dr. Timothy Newman during my initial years in Wireless@Virginia Tech. I also thank my research sponsors, namely the Defense Advanced Research Projects Agency, the Office of Naval Research, the National Science Foundation sponsored Wireless Internet Center for Advanced Technology, and Virginia Tech's Institute for Critical Technology and Applied Science, for all their support without which this work would not have been possible.

I also thank my labmates in Wireless@Virginia Tech, including Gautham Chavali, Shravan Garlapati, Amaar Ahmed, Harpreet Dhillon, Chris Phelps, Joe Gaeddert, Chris Headley, Carlos Aguayo Gonzalez, Abid Ullah, Nitin Arora, and Ashwin Amanna, for the fun times we had together.

I thank Nancy Goad, Jenny Frank, Hilda Reynolds, Cyndy Graham, James Dunson, John Haris, and Cynthia Hopkins, who have been of great help to me during my stay in Virginia Tech. I also thank Virginia Tech, the Bradley Department of Electrical and Computer Engineering at Virginia Tech, and the Newman Library at Virginia Tech for providing me a very supportive and resourceful academic environment.

None of this would have been possible without the unconditional support and love shown by my wife Divya and my dear family. My family, mentors, and friends have influenced me in many positive ways and have made me the person I am today.

Contents

1 Introduction **1**

1.1 Summary of Related Work 5

1.2 Research Overview 6

1.2.1 Research Objective and Goals 6

1.2.2 Dissertation Scope and Contributions 8

1.2.3 List of Publications 11

1.3 Report Organization 12

2 Background on Wireless Cloud Computing Networks and Wireless Distributed Computing **13**

2.1 Existing Distributed Computing Paradigms 14

2.1.1 Traditional Distributed Computing 14

2.1.2 Distributed Computing in Wireless Sensor Networks 17

2.1.3 Wireless Grid Computing 18

2.1.4 Opportunistic computing 20

2.2 Wireless Cloud Computing Networks and Wireless Distributed Computing . 20

2.2.1 Wireless Distributed Computing and Other Enabling Technologies . . 24
2.2.2 Potential Benefits 25
2.2.3 Typical WCCN Application Scenarios and Use Cases 27
2.3 Survey of WCCN and WDC Design Challenges 30
2.3.1 Novel Modeling, Performance Evaluation, and Design Approaches . . 33
2.3.2 Requirements and Challenges Imposed on Communications Design . . 36
2.3.3 Synchronization 40
2.3.4 Control and Management in WCCNs 41
2.3.5 Service Management and Execution 51
2.4 Architecture Design Factors and Implementation Issues 52
2.5 Chapter Summary 55

3 System Model and Notation **56**

3.1 Overview 57
3.2 Computation Model 58
3.2.1 Task Graph Model of User Applications 58
3.2.2 Computational Power Consumption, Energy Consumption, and Execution Time 61
3.3 Communication Model 62
3.3.1 Network Topology 62
3.3.2 Communication Power Consumption 64
3.3.3 Communication Latency 68

3.3.4 Communication Energy Consumption 72
3.4 Power Supply Resource Model 75
3.4.1 Power Availability for Computational Activities 76
3.5 WDC Workload Distribution Model 81
3.6 Total Power and Energy Consumption of WDC Workload Distribution . . . 83
3.7 WDC Power and Energy Savings Metrics 86
3.8 Simulation Parameters 87
3.9 Example WCCN Scenario: Collaborative UAVs in a Tactical Environment . 91
3.9.1 Simulation Results 99
3.10 Chapter Summary 100

4 An Architecture And Methodology for WCCNs 102

4.1 System Requirements for Architecture Design 104
4.2 System Level WCCN Framework 105
4.3 Cross-Layer Decision-Making, Control and Environment Awareness 109
4.3.1 Metrics and Data Structures 112
4.4 Communication Interface and Communication Layer 116
4.5 Wireless Distributed Computing Decision-Making and Resource Management Layer 119
4.6 Service Management Layer 127
4.7 Application and User Interface Layer 129
4.8 Use Case and Demonstration of Workload Distribution 130

4.9 Related Work and Novelty of the Proposed WCCN Architectural Framework 138

4.10 Chapter Summary 143

5 Workload Allocation And Scheduling for Wireless Distributed Computing in WCCNs 146

5.1 Introduction 146

5.2 Summary of Contributions 148

5.3 Power Efficiency of Uniform Workload Allocation in Homogeneous WCCN Environments 151

5.4 WDC Workload Allocation in Heterogeneous WCCN Environments 154

5.4.1 Problem Formulation 155

5.4.2 Algorithm Description 159

5.4.3 Simulation Results 165

5.5 Proposed Task Allocation - Scheduling Algorithm 178

5.5.1 Problem Formulation 179

5.5.2 Algorithm Description 180

5.5.3 Simulation Results 190

5.5.4 Experimental Results 211

5.6 Application of WDC Workload Allocation in Cognitive Radio Networks . . . 217

5.7 Chapter Summary 221

6 A Response Time Model for WCCNs 227

6.1 WCCN Queuing Network Model 229

6.2 Delay Distribution in Finite Sized Discrete-Time Queues 232
6.3 Communication Latency Model 244
6.3.1 Distance Distributions in a Binomial Point Process 245
6.3.2 Link Connectivity in Finite Stochastic Networks in the Presence of Path Loss, Fading, and Noise 248
6.3.3 Delay Distribution in TDMA Networks in the Presence of Packet Buffering, Path Loss, Fading, and Noise 253
6.4 WCCN Response Time Model 270
6.5 Chapter Summary 279

7 Conclusions **281**

Bibliography **285**

Appendices **294**

A Power and Energy Consumption Models **295**

A.1 Computational Power Consumption 296
A.2 Transmitter Power and Energy Consumption 297
A.3 Receiver Power and Energy Consumption 302

B Notation **304**

List of Figures

1.1 An example image processing application which is executed using wireless distributed computing in a WCCN. 4

2.1 A typical wireless cloud computing network environment. 21

2.2 A fundamental synchronization problem in distributed computing. 40

3.1 Interaction among the computation, communication, and power subsystems in a WCCN node. 57

3.2 Task graph models considered in this dissertation. 59

3.3 An example network communication graph model. 63

3.4 A tradeoff between communication range and maximum available power supply for computing under different channel conditions. 76

3.5 A tradeoff between computational capability and communication range under different channel conditions. 77

3.6 Comparison of the theoretical and empirical cumulative density functions of power availability for computing purposes. 79

3.7 Allocation of tasks in a task graph to nodes in a communication graph. . . . 81

3.8 The power consumption - clock rate characteristics for processor models 1, 2, and 3. 88

3.9 An example tactical communication scenario involving UAV nodes and a remote super node. 92

3.10 Energy savings achieved by the example UAV network, via in-network processing and workload distribution, for different backhaul communication ranges and WDC network ranges. 95

3.11 Lifetime improvement achieved by the example UAV network, via in-network processing and workload distribution, for different backhaul communication ranges and WDC network ranges. 96

3.12 Percentage lifetime improvement achieved by the example UAV network, via in-network processing and workload distribution, for different UAV network sizes and backhaul communication ranges. 97

3.13 Lifetime improvement (expressed in hours) achieved by the example UAV network, via in-network processing and workload distribution, for different UAV network sizes and backhaul communication ranges. 98

4.1 Horizontal and vertical layers in the WCCN framework. 106

4.2 A UML interface diagram of Communication Layer components. 116

4.3 The WDC Layer in the WCCN architectural framework. 120

4.4 A UML interface diagram of core WDC Layer components. 121

4.5 A UML representation of the interface provided by the WDC Layer. 126

4.6 A UML interface diagram of the interface provided by the Service Management Layer. 128

4.7 A methodology for achieving WDC workload distribution using the WCCN architecture. 131

4.8 Sequence diagram showing the interaction between layers in the WCCN architecture at startup. 133

4.9 Sequence diagram showing the interaction between layers in the WCCN architecture when optimizing WDC workload distribution at the master node. 134

4.10 Sequence diagram showing the interaction between WDC Layer components when optimizing WDC workload distribution at the master node. 135

4.11 Sequence diagram showing the interaction between layers in the WCCN architecture when executing WDC workload distribution. 136

5.1 Network power savings achieved by WDC workload distribution for different processor models and network sizes. 152

5.2 Network power savings achieved by WDC workload distribution for different processor models, network ranges, and network sizes. 152

5.3 A decision-tree formulation for solving the WDC workload allocation problem. 158

5.4 Illustration of decision-tree search for cost minimization. 164

5.5 Master node's energy savings achieved by workload distribution for various computational power consumption values and application workloads when there is no computational diversity. 166

5.6 Master node's energy savings achieved by workload distribution for various network sizes and network ranges when there is no computational diversity. . 167

5.7 Master node's energy savings achieved by workload distribution for various network sizes, computational workloads, and power supply values when the application comprises independent tasks. 168

5.8 Master node's energy savings achieved by workload distribution for various network sizes, computational workloads, power supply values, and application task graph models. 169

5.9 Master node's energy savings achieved by workload distribution for different network ranges and network sizes. 173

5.10 Master node's energy savings achieved by workload distribution for various ranges of power supply values, computational workloads, and network sizes. . 175

5.11 Master node's energy savings achieved by workload distribution for different network sizes and ranges of task execution time. 177

5.12 Variation in an application's makespan with network range. 193

5.13 Percentage of total workload delegated to slave nodes for different network ranges. 194

5.14 Improvement in an application's makespan achieved by WDC workload distribution for different network ranges. 195

5.15 Scalability of an application's makespan with task execution time range. . . 197

5.16 Improvement in an application's makespan achieved by WDC workload distribution for different task execution time ranges. 198

5.17 Scalability of an application's makespan with task execution time range. . . 199

5.18 Scalability of percentage makespan improvement with task execution time range, for applications comprising concurrent and non-concurrent software structure. 200

5.19 Scalability of an application's makespan with the application's workload. . . 202

5.20 Variation in workload delegation for different application workloads. 203

5.21 Improvement in an application's makespan achieved by workload distribution for different application workloads. 204

5.22 Variation in an application's makespan with network size for applications comprising independent and dependent tasks. 205

5.23 Variation in workload delegation with network size for applications comprising independent and dependent tasks. 206

5.24 Scalability of percentage makespan improvement with network size for applications comprising independent and dependent tasks. 207

5.25 Variation in an application's makespan with network size for applications comprising concurrent and non-concurrent software structures. 208

5.26 Variation in workload delegation with network size for applications comprising concurrent and non-concurrent software structures. 209

5.27 Scalability of percentage makespan improvement with network size for applications comprising concurrent and non-concurrent software structures. 210

5.28 Variation in an application's makespan with transmitted SNR. 212

5.29 Variation in workload delegation with transmitted SNR. 213

5.30 Scalability of percentage makespan improvement with transmitted SNR. . . 214

5.31 Network state information gathered by network probing during an experimental run of the WCCN prototype. 215

5.32 Workload allocation scheme employed during an experimental run of the WCCN prototype. 216

5.33 Application makespan achieved by WDC workload distribution for different network sizes of the WCCN prototype. 217

5.34 Flow diagram and task graph of an example cognitive engine application. . . 220

6.1 The proposed WCCN queuing network model. 229

6.2 A fundamental resource contention scenario in a WCCN. 230

6.3 Simplified WCCN queuing network model used to model response time. . . . 231

6.4 Validation of the theoretical probability mass function of queuing delay. . . . 238

6.5 Validation of mean delay computed using the proposed queuing delay model. 238

6.6 Validation of standard deviation of delay computed using the proposed queuing delay model. 239

6.7 Validation of queue departure rate computed using the proposed queuing delay model. 239

6.8 Comparison of skewness of simulation and theoretical queuing delay PMFs. . 240

6.9 Comparison of kurtosis of simulation and theoretical queuing delay PMFs. . 240

6.10 Validation of mean delay computed using the proposed queuing delay model for short mean service times. 242

6.11 Validation of standard deviation of delay computed using the proposed queuing delay model for short mean service times. 242

6.12 Comparison of skewness of simulation and theoretical queuing delay PMFs for short mean service times. 243

6.13 Comparison of kurtosis of simulation and theoretical queuing delay PMFs for short mean service times. 243

6.14 Distance distribution of n^{th} closest neighbor in a binomial point process comprising 5 points located within a radius of 50 meters. 248

6.15 Validation of the theoretical link connectivity model. 251

6.16 Link connectivity for different network sizes. 252

6.17 Link connectivity achieved under different channel path exponents. 253

6.18 Link connectivity achieved with different transmitter SNRs. 254

6.19 An example round-robin transmission schedule of a TDMA network. 256

6.20 Comparison of the theoretical and simulation PMFs of forward link communication latency experienced by node N_4 in a TDMA network comprising 10 nodes, where the PMFs have a resolution of 1 time slot. 262

6.21 Comparison of the theoretical and simulation CDFs of forward link communication latency experienced by nodes in a TDMA network comprising 10 nodes, where the CDFs have a resolution of 1 time slot. 262

6.22 Comparison of the theoretical and simulation PMFs of forward link communication latency experienced by nodes in a TDMA network, where the PMFs have a resolution of 10 time slots. 263

6.23 Comparison of the theoretical and simulation CDFs of forward link communication latency experienced by nodes in a TDMA network, where the CDFs have a resolution of 10 time slots. 263

6.24 Comparison of the theoretical and simulation mean of TDMA forward link communication latency for different network sizes. 264

6.25 Comparison of the theoretical and simulation standard deviation of TDMA forward link communication latency for different network sizes. 264

6.26 Comparison of the skewness of theoretical and simulation PMFs of TDMA forward link communication latency for different network sizes. 265

6.27 Comparison of the kurtosis of theoretical and simulation PMFs of TDMA forward link communication latency for different network sizes. 265

6.28 Mean communication latency experienced by nodes in a finite TDMA network in the presence of packet buffering, path loss, fading, and noise, for different path loss exponents. 267

6.29 Standard deviation of communication latency experienced by nodes in finite TDMA networks in the presence of packet buffering, path loss, fading, and noise, for different path loss exponents. 268

6.30 Throughput achieved by nodes in finite TDMA networks with packet buffering, fading, and noise, for different path loss exponents. 269

6.31 Network throughput achieved in finite TDMA networks with packet buffering, fading, and noise, for different network sizes. 270

6.32 Network throughput achieved in finite TDMA networks with packet buffering, fading, and noise, for different path loss exponents. 271

6.33 Comparison of simulation and theoretical PMFs of node N_1's response time for different channel conditions. 276

6.34 Comparison of simulation and theoretical PMFs of node N_{10}'s response time for different channel conditions. 276

6.35 Mean response time experienced by a service request in a finite TDMA-based WCCN for different channel conditions. 277

6.36 Mean distance of n^{th} nearest neighbor in a network that spans a radius of 200 meters. 277

6.37 Standard deviation of response time experienced by a service request in a finite TDMA-based WCCN for different channel conditions. 278

List of Tables

2.1 WDC Resource Allocation in WSNs and WCCNs. 48

2.2 WDC Resource Allocation in WSNs and WCCNs (contd.). 49

3.1 System parameters used in this dissertation. 90

3.2 Parameters used in the simulation of a tactical UAV WCCN scenario. 93

4.1 Metrics and decision-making in the WCCN framework. 109

4.2 Example state metrics employed in the WCCN framework. 112

4.3 Example performance metrics employed in the WCCN framework. 113

4.4 WDC performance metrics for an example application service request. 114

5.1 Simulation parameters employed for the analysis of scalability of energy savings with network range. 172

5.2 Simulation parameters employed for the analysis of scalability of an application's makespan with network range. 192

5.3 Simulation parameters employed for the analysis of scalability of an application's makespan with task execution time range. 196

5.4 Presence of diversity and concurrency in different simulation scenarios. . . . 211

B.1 Notation employed in this dissertation. 304

Chapter 1

Introduction

In the commercial cellular market, mobile phone and device connections now exceed 4 billion [1], thereby opening potential avenues for increased connectivity and information availability through collaboration among radios. In military markets, networks of information systems and radios are becoming increasingly important for gathering and sharing critical situational awareness in tactical missions operating in harsh environments. The explosion in the growth of smart wireless devices has increased the ubiquitous presence of computational resources and location-based data. This new reality of numerous wireless devices capable of collecting, sharing, storing, and processing information, as well as executing user applications, makes possible an avenue for new enhanced capabilities and applications.

User applications, such as those that involve information gathering and processing, are expected to play an increasingly important role in daily activities that rely on information, such as surveying product pricing in large shopping malls, gathering situational awareness in public safety and military missions, and gathering up-to-date analytical information on stock trading from individual brokers in a stock exchange. The applications may require abundant resources, including communication, computational, and power resources, in order to process complex computational tasks; appropriate software and data to generate user requested information; and specialized hardware functionality to collect data required by

the user application, execute application software, and present processing results to the user in an appropriate manner. In addition to these requirements, user applications can impose stringent constraints on quality-of-service, security, power consumption, energy consumption, latency, and network range. When these applications are hosted on low complexity and small form-factor portable devices, such as commercial smart phones, hand-held military radios, and tactical tablet computers, it may be hard for a single device to fulfill the application requirements, mainly due to resource constraints (i.e., limited computing power and low power resources); or non-availability of specialized data, software, and hardware that are required for executing the user application.

Traditionally, user applications are either processed locally in a single node or offloaded to a remote cloud network via a cellular link or a WLAN link. Cloud infrastructures may not always be available for access or the connectivity to cloud resources may not be reliable, both in terms of link performance and link security. As an alternative to traditional approaches to processing user applications, multiple resource-constrained radio nodes with diverse functionalities can dynamically form a virtual wireless cloud computing network (WCCN) and collaborate on executing the user application's complex computational tasks using wireless distributed computing (WDC). Data, software, and computational resources from numerous devices can be pooled together and be made available on-demand to user applications. Mobile users can now execute complex computational applications from their smart wireless devices while being on the move by leveraging the computational resources which are accessible in such collaborative environments. In addition to consuming resources made accessible in cloud environments, users of smart wireless devices may offer services via such cloud environments. Users can trade computational resources hosted by their personal devices, including software, specialized hardware, and customized data, in the form of customizable services that can be accessed by other users' applications. In addition to trading services, users can permit opportunistic access to idle resources which, in turn, reduces wastage of resources that may be otherwise left unused. For instance, in network-centric systems where software defined radios (SDRs) collaborate to achieve a common objective,

idle computational resources of SDRs can be harnessed in an opportunistic manner to execute secondary user application software in addition to the primary communications signal processing software. This new approach requires a paradigm shift in the way information services are executed in both military and consumer arenas. This notion of wireless cloud computing can be extended to handle complex applications in modern radio networks such as tactical radio networks, personal communication networks, and wireless sensor networks. Furthermore, future wireless standards are expected to adopt computationally complex ad-hoc networking approaches involving complex communication and computation tasks which operate under stringent quality of service (QoS) constraints. The notion of collaborative computing using WCCN configurations can be applied to such application scenarios in order to allow practical realization of future wireless standards.

Wireless distributed computing is a technology that enables the distributed execution of user applications among nodes in a WCCN, where the applications' distributed computational tasks coordinate with each other via wireless communication between their respective host nodes. WDC can be employed for the distribution of computational workloads imposed by a user application on one or more nodes in order to meet the specified application goals, reduce the resource burden on the user's node, and meet the network's resource constraints. In addition, a user application can be composed in terms of services hosted by different nodes in a WCCN, such that the data requirements and other application needs are satisfied, and executed in a distributed manner on the physically separated host nodes. Furthermore, WDC can be applied to security applications, as will be discussed in Section 2.2.3. WDC technology provides the core mechanisms necessary for the distributed execution of user applications in WCCNs, including resource allocation, fault management, and synchronization and communication between individual distributed components of the user applications.

The following example illustrates a WCCN which is formed among smart phones. A smart phone user wishes to capture an image of a group of persons and gather information about the persons in the image. This user application requires access to specialized hardware, such as a high definition camera; an image segmentation software to recognize faces in the group

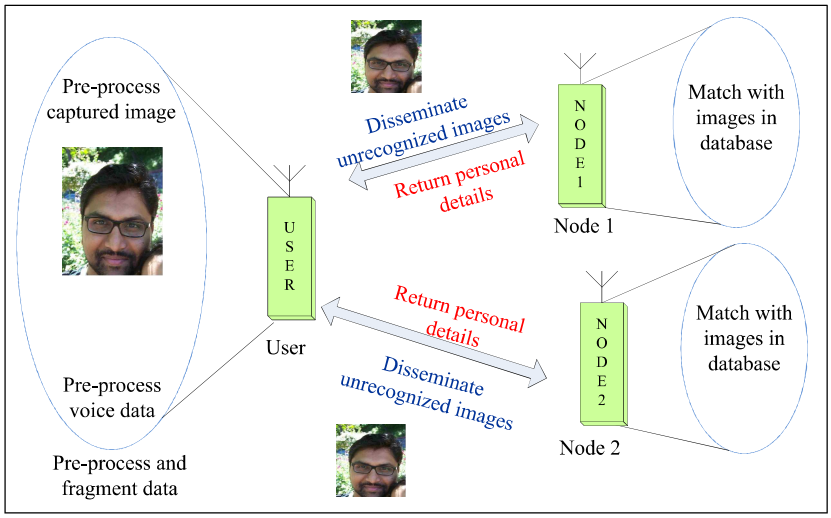


Figure 1.1: An example image processing application which is executed using wireless distributed computing in a WCCN.

photo image; a database of known persons with their respective face templates mapped to personal information; and a face matching software which can match the faces identified in the image with face templates in the database. While the user's smart phone can use its camera to capture the image and use its processor to execute the software, the database may be distributed among several users. Moreover, local execution of the user application on the user's device may place a burden on the limited resources of the user's device and, in turn, affect the application's execution time. As an alternative, the user's smart phone can form a WCCN with two neighboring smart phones, namely, nodes 1 and 2, and execute the user application as shown in Figure 1.1. The user's smart phone runs the image segmentation software on the group photo image and distributes the resulting face images to nodes 1 and 2. Upon receiving their respective share of the computational workload, nodes 1 and 2 match the face images with their respective databases and return the results back to the user's smart phone.

1.1 Summary of Related Work

Wireless cloud computing is a next-generation collaborative computing paradigm which builds upon contemporary distributed computing paradigms such as wireless grid computing, cloud computing, and opportunistic computing. As in a typical wireless grid and cloud computing environment, diverse resources can be pooled together dynamically and shared in a WCCN. This virtual pool of resources can be provided as services to user applications via the network, which is similar to the notion of offering resource-as-a-service in a traditional cloud environment. This dissertation follows the definition of cloud computing as set forth by the National Institute of Standards and Technology (NTIA) [2]. In this regard, the concept of WCCNs builds upon NTIA's notion of cloud computing by handling service requests in dynamic and uncertain operating environments.

A WCCN fundamentally differs from grids and clouds in its use of wireless distributed computing as an enabling technology for dynamic execution and robust management of distributed applications in heterogeneous, dynamic, unreliable, insecure, and uncertain wireless environments. A WCCN is dynamically formed as a virtual conglomeration of autonomous but cooperating mobile radio nodes where the connectivity between the entities may be unreliable and probabilistic. In addition, radio mobility induces uncertainty to the availability of "cloud computing resources" unlike a traditional cloud computing infrastructure that is stable, secure, and reliable. A traditional cloud comprises individual nodes which are stationary and are connected to the cloud via reliable links. Unlike the NTIA's notion of cloud computing where the resources appear unlimited to the user application, the WCCN environment is characterized by resource constraints that are taken into consideration while provisioning resources for the distributed execution of user applications. In addition, WCCNs allows a user application to be intelligent and be able to make decisions, which impact its execution, based on its awareness of the WCCN environment. These differentiating factors are evident in the architectural design presented in Chapter 4. In addition to architecture-level novelties, there is a stress on novel modeling and design approaches, performance metrics, and

application scenarios, as will be discussed in Chapter 2.

Distributed computing has been previously studied in the context of traditional wired networks [3–5] and WSNs. Sang et al. study the feasibility of distributed computing on different wireless local area network topologies [6]. Xu et al. [7] present a distributed computing paradigm that enables collaborative signal processing and addresses challenges such as energy efficiency, reliability, data fusion, and scalability of distributed signal processing algorithms. Resource allocation for distributed computing has been extensively studied in the context of networked embedded systems and WSNs [8–13]. Some of the existing distributed computing resource allocation concepts can be used as a basis for WCCN research, however, they need to be significantly adapted before being applied to WCCNs which operate in unique and challenging environments. In traditional WSNs, for instance, the sensor nodes are fixed in position thereby simplifying several aspects of distributed computing system design. In addition, most of the network state is known a priori to the designer in most WSN situations. WCCNs, however, do not restrict themselves to WSN-like scenarios and operate with limited a priori knowledge of their dynamic environments. Chapter 2 presents an elaborate discussion on the comparison between challenges faced by WCCNs and traditional distributed computing networks.

1.2 Research Overview

1.2.1 Research Objective and Goals

The objective of this dissertation is stated as follows: "***To propose a high-level architectural framework and methodology which enables collaborative computing in a dynamic, uncertain, and heterogeneous wireless cloud computing network environment; and to design, prototype, and evaluate the performance of WDC resource allocation mechanisms in the proposed architecture.*** "

The research objective is realized in terms of the following goals.

Research goal 1: Development of an architecture and methodology for collaborative computing in WCCNs The primary objective of this dissertation is to present a high-level architecture for WCCNs and present a methodology for the execution of user applications using the proposed WCCN architecture. The following are research questions which are addressed as part of this research goal: what core architectural functionalities and components are necessary to enable collaborative computing in a WCCN environment; how do the core architectural components interact with each other in order to enable WCCN activities; and what application programming interface (API) is needed in order to plug-and-play a user application in a WCCN environment?

Research goal 2: Development of fundamental models and metrics for performance analysis of WDC workload distribution in WCCNs The goal is to develop models that capture power consumption, energy consumption, and latency costs in a WCCN environment, and use the models to analyze feasibility, design tradeoffs, and limitations in performance of WCCNs. The models are also intended to be used in the design and performance analysis of decision making and resource allocation mechanisms in the WCCN architecture. Specifically, the following research questions are addressed: what are the quantitative benefits of WDC workload distribution over traditional local computing in terms of power, energy and execution time; under what conditions is WDC beneficial over local computing; what metrics can be used to quantify the worthiness of assigning a computational task to a remote wireless node; and how does WDC performance in WCCNs scale with system parameters such as communication range, link quality, network size, computational workload (application complexity), and finite buffering of communication packets and service requests?

Research goal 3: Development of resource allocation algorithms for WDC workload distribution in WCCNs One of the goals of this dissertation is to develop WDC resource allocation algorithms that can be employed in the proposed architecture. The following is a research question that is addressed as part of this goal: with knowledge of the quality-of-service and resource requirements of a user application and knowledge of the characteristics of the underlying stochastic wireless environment, how can the application's computational workload be distributed and scheduled among the nodes in a WCCN in a manner that meets specified operational goals?

1.2.2 Dissertation Scope and Contributions

The system under consideration is a single-hop wireless network that is setup in a master-slave configuration. It comprises a single cluster with a single master node and several slave nodes. All nodes communicate with each other over wireless links using a centralized and scheduled medium access control protocol, such as time division multiple access, which allows re-transmissions of erroneously received packets. The wireless channel model accounts for large scale fading, which includes path loss and shadowing; small scale fading, specifically Rayleigh fading; and additive white Gaussian noise. All wireless links are assumed to be free of any interference. Every node in the network comprises a computational platform with a single core processor. In other words, there is no concurrency in computational processing within a node. The system model considers both homogeneous and heterogeneous operating environments. In heterogeneous WCCNs environments, the nodes may possess varying processing capabilities and varying battery capacities. The system model and assumptions are further elaborated in Chapter 3.

The architecture development is presented in three phases, namely, framework design, mathematical modeling and analysis, and algorithm design. The dissertation proposes an architectural framework that identifies the core components, as well as their respective functionalities and interfaces, required for the collaborative execution of user applications in

WCCN environments. The dissertation presents a basic methodology for the execution of user applications in a WCCN environment using the proposed architecture. The dissertation also presents a software defined radio based prototype of the proposed architecture, which not only serves as a proof-of-concept of the architecture but also serves as an experimental test-bed for WCCNs. While the architecture takes into consideration several aspects of a typical WCCN environment, the prototype implementation focuses specifically on the WDC aspects of the architecture and, in fact, is used for experimental analysis of the proposed WDC resource allocation algorithms. The prototype operates for a fundamental scenario involving one master node located in a single hop network and handles only a single application request at a time. Experimental results captured using this prototype demonstrate significant improvement in the execution time of an example image processing user application, achieved by employing the proposed adaptive computational workload allocation - scheduling algorithm in the prototype. Other aspects of WCCN design, such as service discovery, synchronization, fault tolerance, security, and mobility management, are not within the scope of this dissertation. The architecture can be extended to support a broad range of WCCN configurations, such as single-hop and multi-hop networks; a variety of operating scenarios, including those involving multiple service requesters and multiple service providers; and a broad range of applications.

In the modeling and analysis phase, the dissertation presents a mathematical model of a typical WCCN environment, and defines performance metrics that quantify the power consumption, energy consumption, and execution time benefits achieved by WDC workload distribution in WCCNs. The system model captures power consumption, energy consumption, and latency incurred by communication and computation activities involved in workload distribution in a typical WCCN environment. In addition, it characterizes the statistical availability of power resources for consumption by computational activities. The dissertation presents the analysis of workload distribution in both simple homogeneous environments and the more complex heterogeneous environments.

In the algorithm design phase, the dissertation focuses on WDC resource allocation in WC-

CNs. The resource allocation problem in this dissertation is focused on the allocation of computational tasks to available nodes in a WCCN, and scheduling of computational and communication activities involved in the execution of user applications, such that the resource allocation objective is optimized. Workload allocation and scheduling algorithms are designed with the following assumptions: (a) every node can execute at most one computational task at a time, i.e., every node comprises a single core processor which does not process multiple tasks concurrently, (b) every task is executed on one and only one node, i.e., redundancy in task executions which aids fault tolerance is not considered, (c) scheduling is non-preemptive, where a particular task continues execution until its completion without any interruptions ever since it begins execution, and (d) precedence constraints between the computational tasks in a given user application are satisfied. The dissertation proposes a stochastic workload allocation algorithm which determines the optimal allocation of an application's computational tasks to nodes in a WCCN with a goal of minimizing the total energy consumption incurred during the execution of the application. In addition, the dissertation proposes a heuristic algorithm that handles both task allocation and scheduling. Both algorithms account for the energy constraints in each node in the network, heterogeneity in computational costs, and communication costs incurred due to path loss and fading. In addition, the task allocation - scheduling algorithm takes into consideration the scheduling constraints of computational and communication resources of individual nodes in the network. Both algorithms are designed based on the following proposed design principles: (a) joint allocation - scheduling of computational and communication resources; and (b) inter-layer design which takes into consideration the interaction between the WDC layer and the Communication layer in the proposed architecture.

All work in this dissertation, with the exception of the proposed response time model, is scoped to systems that handle a single application request at a time. The dissertation proposes a response time model which aims at characterizing the overall delay experienced by application requests in a single-hop WCCN environment, where there are multiple service requesters and each service requester generates multiple service requests. The model is based

on stochastic queuing networks and considers the effect of queuing delays introduced by multiple requests. It uses finite sized queues to represent buffering of packets at wireless transceivers of individual nodes and to represent contention of service requests for computational resources hosted by service providers. The proposed model can be easily extended to include the effects of relaying nodes in a multi-hop scenario.

1.2.3 List of Publications

- D. Datla et al., "An architecture for wireless cloud computing networks," manuscript under preparation, Apr. 2013.
- D. Datla et al., "A response time model for wireless cloud computing networks," manuscript under preparation, Apr. 2013.
- D. Datla et al., "Wireless distributed computing in cognitive radio networks, Ad Hoc Networks, Special Issue on Cognitive Radio Ad Hoc Networks, vol. 10, no. 5, pp. 845-857, Jul. 2012. (Invited)
- D. Datla et al., "Wireless distributed computing: a survey of research challenges," IEEE Commun. Mag., vol. 50, no. 1, pp. 144-152, Jan. 2012. [Appeared in top 10 most downloaded articles in IEEE Communications Society Magazines and Journals in February 2012.]
- D. Datla et al., "Task allocation and scheduling in wireless distributed computing networks," Analog Integrated Circuits and Signal Processing, vol. 69, no. 2-3, pp. 341-353, Dec. 2011. (Invited)
- D. Datla et al., "Fundamental issues of wireless distributed computing in SDR networks," in SDR Forum Technical Conference, Washington DC, Dec. 2010.
- D. Datla et al., "Waveform Level Computational Energy Management in Software Defined Radios," SDR Forum Technical Conference, Washington DC, Dec. 2009. [Won

best paper award]

- D. Datla et al., "Power efficiency in wireless network distributed computing," in Proc. Vehicular Technology Conference, Anchorage, AK, Sept. 2009, pp. 1-5.

1.3 Report Organization

The rest of the dissertation is organized as follows. Chapter 2 presents a survey of some prominent distributed computing paradigms and introduces the notion of WCCNs. The chapter also presents a survey of potential challenges faced while designing and deploying WCCNs. The system model and assumptions for all work presented in this dissertation are described in Chapter 3. The proposed WCCN architecture and methodology are described in Chapter 4. A prototype of the proposed WCCN architecture is also presented in Chapter 4. Chapter 5 presents the analysis of power and energy efficiency achieved by WDC load distribution in WCCNs. The proposed workload allocation and scheduling algorithms are described in Chapter 5 along with their performance analysis. Chapter 5 also presents experimental results gathered from the WCCN prototype. Chapter 6 presents the proposed response time model and uses the model to analyze the response time of application requests under different network conditions. Finally, the conclusions and future directions are discussed in Chapter 7. Appendix A describes the power and energy consumption models that form the basis for the models presented in Chapter 3. Appendix B presents the notation used throughout this dissertation.

Chapter 2

Background on Wireless Cloud Computing Networks and Wireless Distributed Computing

The notion of WCCNs builds upon traditional distributed computing concepts and recently introduced collaborative computing paradigms such as wireless grid computing, cloud computing, and opportunistic computing. While these paradigms serve as a starting point for research on WCCNs, novel design approaches are needed in order to meet the unique requirements and challenges imposed by WCCNs. In this chapter, WDC is introduced as a novel enabling technology for WCCNs. The chapter begins with a survey of existing distributed computing paradigms as presented in Section 2.1. The envisioned WCCN environment is described next in Section 2.2 along with its enabling technologies, benefits, and potential application scenarios. Section 2.3 provides an overview of the unique challenges faced by WCCNs and WDC. This section begins with a discussion on how WCCN environments differ from traditional distributed computing environments. Next, some architectural and implementation issues are discussed in Section 2.4. Section 2.5 presents a brief summary of the chapter.

2.1 Existing Distributed Computing Paradigms

This section begins by providing background on traditional distributed computing with particular focus on simple modeling and performance evaluation approaches. Next, the section discusses on the application of distributed computing to wireless sensor networks. This is followed by a survey of recently introduced distributed computing paradigms, namely, wireless grid computing and opportunistic computing.

2.1.1 Traditional Distributed Computing

There are two broad thrust areas involved in the deployment of distributed computing environments, namely, distributed applications and distributed systems. Distributed systems research is concerned with system-level issues that arise when executing distributed applications on a network of computing devices, such as communications, optimal management of computational resources, and synchronization. System design has to meet underlying resource constraints and satisfy the requirements of the distributed algorithm, including inter-node communication quality-of-service (QoS), synchronization requirements between the distributed components executed on different nodes, and robustness. Distributed applications research involves the development of distributed algorithms. Distributed algorithms are designed to run in environments comprising many interconnected processors with the objective of solving specific problems in a distributed manner [3]. The individual pieces of a distributed algorithm run concurrently and independently, each with only a limited amount of information [3]. The algorithms are supposed to work correctly even if the individual processors and communication channels operate with different characteristics and even if some of the components fail. Traditionally, system issues have been abstracted in the development of distributed applications. However, the use of a cross-layer approach in designing applications that are targeted towards dynamic wireless network environments is becoming commonplace nowadays. Abdelhak [8], for instance, presents a methodology for developing distributed

QR decomposition algorithms and executing them in a distributed manner on collaborative wireless sensor networks (WSNs). The following are some research questions that one encounters when pursuing research at the intersection of distributed applications and distributed systems: under what conditions are distributed applications significantly impacted by behavior of the underlying system; does consideration of system issues in distributed applications development provide an optimized approach to distributed computing and improve performance of distributed computing in uncertain environments; and can problems arising from distributed applications and distributed systems research be solved separately with specified abstraction between the two aspects of a distributed computing environment? Some modeling and performance evaluation approaches employed in traditional distributed computing research are discussed next.

Modeling communication and intra-node processes Distributed algorithms and systems are modeled using state machines, compositions of state-machines, queues, and set theory [3]. The core elements of a distributed application include the processes that run inside each node in the network and the message passing between them.

Communication channel models for traditional distributed computing networks (DCNs) can be one of a combination of point-to-point, broadcast, and multicast models. Channel models account for the following faults: (i) message errors, (ii) message losses, (iii) message duplication, and (iv) re-ordering in message delivery. Accordingly, point-to-point channels are modeled as reliable first-in-first-out (FIFO) channels, reliable reordering channels, and erroneous channels [3]. A FIFO channel is one in which the channel state is represented by a message queue and the channel does not exhibit any of the faults listed above. Reliable reordering channels do not guarantee message ordering. Erroneous channels exhibit faults such as message errors, message losses, and message duplication.

A state space representation can be used to model the dynamics of intra-node processes [3], which include computing processes and communication processes. The computing process is

modeled in terms of its state at round r of the application's execution. The representation of a state encodes the inputs and outputs to that state. In a synchronous execution of a distributed application, the state transition inside a node at a particular round is a function of the current state, message(s) generated and transmitted upon completion of the current round, and message(s) received before the beginning of the next round. Message generation, in turn, can be modeled as a function of the current state of the computing process. Accordingly, synchronous executions are described as a sequence of events termed as an execution trace. An example execution trace is as follows: $C_0, M_1, N_1, C_1, M_2, N_2, C_2 \ldots$, where C_r^i is the state of the computing process inside node i at round r, and M_r^i and N_r^i are messages transmitted and received by node i at the end of round r [3]. Initial states and state transitions can play an important role in the correctness and stability of distributed applications that are executed in a DCN.

The performance of a DCN in complex environments can be analyzed by making simplifying assumptions on the fault detection capabilities of the DCN, such as the capability to detect fault occurrence in a certain link or a node [14]. The various types of faults include communication faults (discussed previously under channel models), partial reliability and total reliability. Under the assumption of partial reliability, faults will not occur under specified conditions. Total reliability assumes that faults do not occur at all. The assumptions, especially the latter ones, make a DCN model restricted in its applicability in fault prone scenarios.

Performance evaluation The performance of traditional DCNs can be analyzed in terms of their reliability in executing a distributed computing process; QoS metrics such as time-to-completion and efficiency; and system cost metrics such as communication overhead. Time-to-completion is the time that elapses between the start of computation by the first node in the DSN and the end of computation by the last node that completes execution of the distributed application. Communication delay includes delays incurred by message transmission, queuing, and processing. Each of the delays can be a random variable dependent on

several factors such as resource availability and network traffic conditions. Traditional DCN models assume that delays are finite and bounded with a deterministic upper limit under ideal conditions of synchronization and fault occurrence [14].

Communication overhead or communication cost takes into consideration the amount of communication activity, including message transmission, reception and relaying, that results while executing a distributed application. It can be measured in terms of messages for higher layer performance analysis, and in terms of packets and bits for link layer performance evaluation. Example cost metrics include mean cost per node and mean transmission load per link [14]. The mean cost per node is expressed as the ratio of the total communication load and the number of nodes in the network.

2.1.2 Distributed Computing in Wireless Sensor Networks

Distributed computing has been applied in WSN application scenarios, mainly for in-network processing of sensor data. In-network processing involves pre-processing of sensor data, which reduces the amount of sensor data that is transmitted to the WSN's base-station. This results in a reduction in communication energy consumption at the cost of some computational energy consumption. In-network processing is beneficial under the following conditions. First, in-network processing is particularly energy efficient when the computational energy required for pre-processing is significantly lower than the communication energy. Second, the probability of packet loss/error and communication latency increase with data size and communication range. A reduction in the amount of sensor data , as achieved by in-network processing, is beneficial particularly in the presence of an insecure and unreliable long range backhaul between the sensor nodes and the base-station. Third, when the sensor nodes and the base-station are resource-constrained, distributing the data processing operation among the sensor nodes can result in a processing speed-up. Fourth, when the sensor data generation rate is greater than the maximum channel data rate that is supported in the network, data can be pre-processed to fit the communication bandwidth and meet the real-time needs of the

application that runs at the base-station. When applying distributed computing workload balancing algorithms to in-network data processing, the following factors should be taken into consideration while iteratively processing the sensor data as it is being generated: incoming sensor data generation rate, resource availability at individual nodes, sensor data storage limit, limits on data processing rate, communication queue length limit, and communication capacity limit.

Networked embedded systems (EmNets) and WSN designs are motivated by the need for energy-efficiency and energy-awareness at all layers in the protocol stack. Distributed computing resource allocation has been extensively studied in the context of EmNets and WSNs [8–13]. Techniques proposed in the past address one or all of the following aspects of resource allocation: (a) workload allocation; (b) configuration of computational and communication parameters, such as clock frequency and modulation; and (c) scheduling of communication and computational processes.

2.1.3 Wireless Grid Computing

Foster [15, 16] presents some of the earliest ideas on grid computing. Grid constituents include hardware and software resources, communication interfaces, and different types of platforms and operating systems. *The grid is envisioned to be an adaptive network that allows inexpensive, flexible, secure, and coordinated resource sharing among dynamic heterogeneous collections of individuals, institutions and resources which span across uncontrolled and dynamically varying geographic, political, and cultural boundaries; and which provides mechanisms for real-time access to that pool of resources in a manner that is simplified, stable, transparent, scalable, and flexible to the end user's application* [16,17]. Recent works [17,18] introduce wireless grids as a paradigm that provides a seamless extension of the resource sharing functionality of stable infrastructure grids to ubiquitous mobile and nomadic devices, thereby expanding the scope of the grid and opening up newer avenues for data collection and resource sharing. According to McKnight [18], "*wireless grid computing is*

a distributed computing paradigm which allows dynamic and coordinated sharing of ad hoc resources among fixed, nomadic, and mobile nodes which may be affiliated to different types of networks such as the internet-based grids, ad-hoc networks, and peer-to-peer networks". McKnight [18] also presents some interesting introductory remarks on economic and policy aspects of such resource sharing.

Agarwal et al. [17] classifies wireless grids based on application and architecture type. Computational grids, such as WSNs, specialize in cooperative or parasitic sharing of computational resources. A data grid enables coordinated and secure access to distributed data which may be located all in one node if not for the node's limited storage space. Utility grids provide users paid access to specialized hardware, software, or any resource type. Grid architectures may either be ad-hoc or infrastructure-based. Based on the homogeneity of grid constituents and the level of administrative control, grid infrastructures can be layered and have the following hierarchy: (a) homogeneous wireless grid, (b) wireless intra-grids, and (c) wireless inter-grids. Homogeneous grids comprise homogeneous devices that are located in a local area and controlled by a single authority. Various aspects of resource sharing is made easier by employing a common set of policies and protocols for cooperation and general interaction among devices, including the use of a common language to characterize resource availability. Each division of a large organization, for instance, can control and manage its own homogeneous wireless grid. An intra-grid may be an integration of all divisional homogeneous grids in the organization, where the constituent homogeneous grids may be virtually local to each other while they may be physically located in different geographical regions. Although the constituent homogeneous grids are governed by different policies, all the homogeneous grids can share a common level of trust and readily authenticate and identify each other under a common higher level administrative body. An inter-grid comprises intra-grids of several organizations, where interaction between the constituent intra-grids may be strongly regulated based on content confidentiality and level of trust between the grids. Agarwal et al. [17] presents some example business cases for this hierarchical structure. An example scenario may involve three homogeneous wireless grids, one each for fire personnel,

police and rescue squads. At the time of national calamity, the three homogeneous grids are rapidly deployed and integrated into a wireless intra-grid of various public safety grids that follow the P25 standard. This grid may be integrated with the cellular network, which is by itself an intra-grid comprising 2.5G, 3G and 4G networks. Together, this massive rapidly deployed grid forms an inter-grid that can be used for sharing situational awareness over a large scale amongst the first responders as well as with the general public.

2.1.4 Opportunistic computing

Conti [19] is one of the popular works on opportunistic computing. Opportunistic computing is a resource sharing computing paradigm which enables opportunistic utilization of idle computational resources. In a network where opportunistic access to computational resources is permitted, the idle computing resources of a node, namely the primary node, can be utilized by other nodes, namely the secondary nodes, which are in need of computing resources for their own purpose. When the primary node requires access to its computational resources, all secondary processes utilizing its resources may be preempted.

2.2 Wireless Cloud Computing Networks and Wireless Distributed Computing

There are several definitions of cloud computing in industry. This dissertation follows the definition set forth by NTIA [2]. According to the NTIA, "*cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*" [2]. Cloud computing drives a network-based approach to computing wherein services are provisioned via the network. Access to network computing resources and services in a

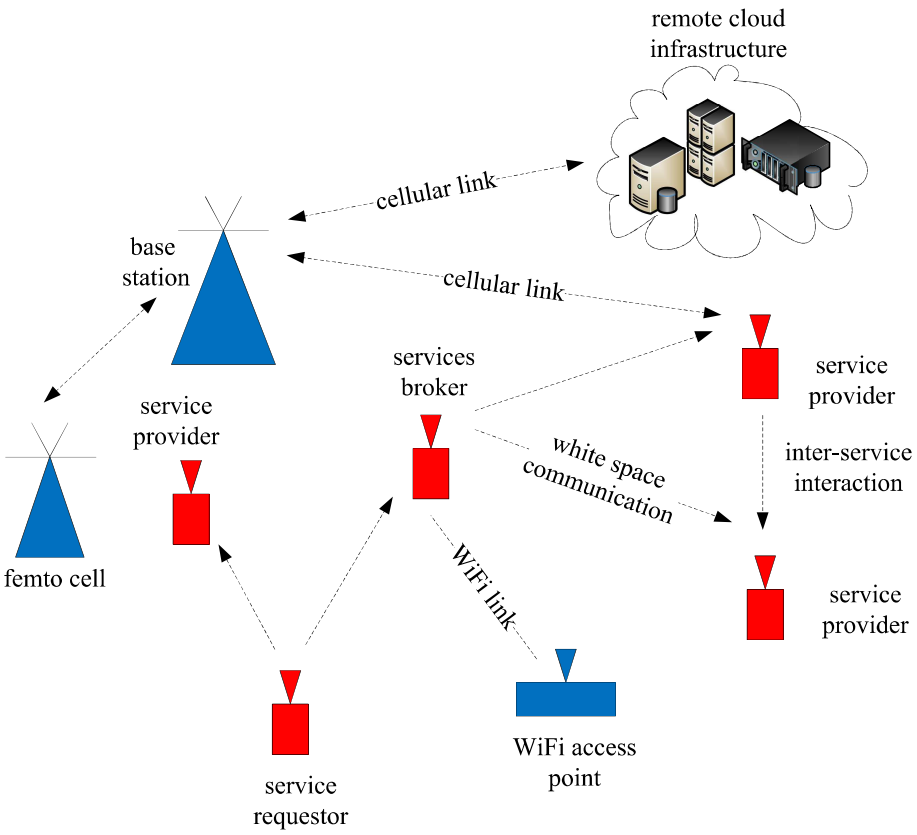


Figure 2.1: A typical wireless cloud computing network environment.

typical cloud infrastructure is provided through simple user application friendly interfaces. The cloud appears to the user application as an abstraction of the computing resources and complex mechanisms that manage access to them.

A Wireless Cloud Computing Network provides a virtual, reliable collaborative computing environment which comprises a network of autonomous nodes that host computing resources and nodes that consume them. Resources are provisioned in the form of services that can be identified, requested, and consumed by user applications that run in the individual nodes. A WCCN environment provides coordination and management mechanisms that enable operation in dynamic and uncertain cloud computing environments. Management activities include the exchange of information on cloud environment awareness, and the coordination of interactions between services and user applications. A Mobile Ad-Hoc Cloud Computing

Network (MACNET) is similar to a WCCN with the exception that it is designed to support ad-hoc network configurations and node mobility.

A typical WCCN environment is shown in Figure 2.1. Resources in a WCCN include computational resources such as processor cycles, cache memory, and permanent storage memory; software resources such as data and computer code which may be available either in source code or machine readable format; special purpose hardware such as sensors and multimedia capture hardware; communication resources such as transceiver duty cycles and communication bandwidth/spectrum; and energy resources such as battery energy and environmentally harvested energy. These cloud resources can be provisioned to user applications in the form of services via the WCCN. In other words, services regulate access to resources that they abstract. The main players in a WCCN environment are service providers, service requesters, service brokers, and managers. The players may be stationary in the case of a WCCN or mobile in the case of a MACNET. They may be present locally or be present remotely with access via the web. A service provider hosts services which can be consumed by user applications that run in service requesters. A service broker mediates between the service requester and the service providers, and facilitates access to individual services or a customized composition of services. A service requester may send out a request for services either directly to the individual service providers which host the requested services, or to a service broker. In the latter case, the service broker monitors execution of the requested services in order to ensure that they meet the request requirements, coordinates interaction between services, and returns the requested results to the service requester. In this manner, service brokers assume increasing levels of responsibilities thereby reducing burden on the service requester. The additional layer of abstraction between service requesters and service providers introduced by service brokers incorporates hierarchy in the access to services. Managers monitor various aspects of the environment and play a role in managing the cloud environment.

The three main entities of a user application include data, code to process data, and a user interface to gather user requirements and display results to the user. At the time when a user makes a request to an application, the application is dynamically composed from an identified

set of available services such that the application's resource and QoS requirements are met. The application can choose from a diverse set of service combinations and execution options to optimize the application's objectives. Execution of the application may require access to a combination of cloud services and may necessitate interaction between the services. The required services may not all be co-located within a single node, but may be distributed throughout the WCCN. As a result, the execution of an application may involve wireless communication between nodes hosting the required cloud resources. For instance, node X may be rich in data, but may not have sufficient computational or energy resources required to process the data. A second node, namely node Y, offers a utility service to process the data received from the first node. Data is transmitted from node X to node Y, which in turn processes the data and returns the results to the node which hosts the user application. The link connecting the service requester to a service provider, over which a service request is transmitted, is termed as the forward link. In this dissertation, the link from a service provider back to the service requester, over which the provider transmits results to the provider, is termed as the reverse link.

WCCN environments can be highly dynamic and heterogeneous. The cloud entities may be stationary (or mobile in the case of a MACNET). Nodes may enter and leave the cloud environment at random thereby directly impacting the availability of cloud services hosted by them. The availability of cloud services can vary dynamically due to probabilistic contention for shared resources, varying resource utilization, the dynamic nature of radio environments, and user-induced dynamics including intentional termination of services hosted by the user. In MACNETs, the mobility of cloud entities can further impact resource availability. The utilization of energy resources may vary due to variations in communication costs, which in turn are caused by dynamics in radio operating environments, and variations in computational workloads. The characteristics of the radio environment can greatly impact the accessibility of hosted services. Data hosted by a node, for instance, will be accessible only to other nodes which are located within radio reach of the service provider node.

A WCCN environment is characterized by heterogeneity in various aspects such as link

quality; wireless communication signaling formats, protocols and standards; service ontology; node computational and communication capabilities; software and hardware platforms with varying hardware and software capabilities; and availability of cloud services. As shown in Figure 2.1, a node in the cloud may be one of different types which differ in form factor, capabilities, and accessibility options. For instance, a user's smart phone can be accessed either via bluetooth, wifi, or cellular network; a stationary server or cluster of servers can be accessed either wirelessly or via a cellular infrastructure; a remote server cluster can be accessed via the web; and a laptop can be reached via a hotspot. In addition, nodes may be remote or local. Furthermore, user requirements and preferences can vary drastically in terms of QoS expectations and user interfaces. Some of the enabling technologies, potential benefits, and applications of WCCNs are presented next.

2.2.1 Wireless Distributed Computing and Other Enabling Technologies

Earlier, wireless devices did not generally have access to computational resources, such as unutilized processing power and information, on peer nodes. The advent of self-organizing distributed ad-hoc networks has improved interaction and collaboration among nodes. Recently, wireless devices, such as video capable smart phones and multi-sensor equipped soldiers, are seen not simply as objects for communication, but as devices able to collect and process information. In addition, the ability to connect to other devices and share information wirelessly and seamlessly has revolutionized the way devices interact with one another. Processed information and computational resources can then be shared with other nodes in a collaborative network, thereby creating an avenue for new, enhanced capabilities, and applications.

Some of the key enabling technologies of WCCNs include wireless distributed computing; dynamic service discovery to search for services required by user applications; cloud resource management and service provisioning, which manage dynamic resource sharing and integra-

tion of resources into services with QoS guarantees in a rapidly changing cloud environment; and intelligent decision-making in user application. While service management technologies have prominence at a high level in WCCNs, *wireless distributed computing is a core technology for WCCNs that exploits wireless connectivity to share processing-intensive tasks among multiple devices. It involves the execution and management of user applications as a distributed set of sequential and/or concurrent computational tasks, all executing their part of the application independently on physically separate hardware, but coordinating their activity through wireless communication.* The distribution of computational workload in WDC aims to reduce per-node and network power, energy, and processing resource requirements; improve security through distribution of code and data; boost resource utilization by utilization of idle resources; and enable complex applications not otherwise possible. With the help of an example application scenario, Sections 3.9 and 5.3 demonstrate how some of these goals, particularly energy efficiency, are achieved by applying WDC to complex data processing applications.

WDC in WCCNs is possible today due to the availability of key technologies such as fault tolerant computing, distributed computing, software defined radio (SDR), and cognitive radio (CR). These technologies can offer capabilities such as collaboration among wirelessly connected computational devices; flexible link design; and adaptation and autonomous operation in dynamic WDC environments. These capabilities enable flexible and optimized use of resources, reliable wireless communications, joint optimization of computation and communication processes, and customization for the QoS needs of WDC.

2.2.2 Potential Benefits

WCCNs can enable the exploitation of diversity in a cloud environment; opportunistic utilization of idle resources; sharing or delegation of computational workloads; and extended capabilities of a network of resource-constrained nodes.

When heterogeneous cloud resources with varying utilization levels are pooled together in

a WCCN, the resulting diversity in resource availability and quality can be exploited to benefit user applications. For instance, when data or specialized hardware required by a user application is not locally available within the user's node, there may be other nodes that are able to share that data. Such diversity can also benefit applications that rely on location based processing. The opportunistic usage of idle computing resources, which would otherwise remain unused, can improve resource utilization and benefit user applications by providing performance enhancements.

Local processing of user applications requires a high resource requirement per node and there can be no QoS guarantee from individual fault prone radio nodes. Hence, even if all nodes are equally well equipped, which makes them self-sufficient in meeting application needs, the sharing of resources and computational workloads can potentially offer several benefits over local computing. It enables: (a) lower energy and power consumption per node as well as for the network as a whole, resulting in improved power and energy efficiency; (b) robust, secure, and fault tolerant execution of user applications; and (c) simplified small form factor node designs with lower resources per node made possible by the reduced burden on resources of individual nodes. The security benefits of this technology will be discussed in the next section.

In application scenarios where individual nodes have to operate under stringent resource constraints, the enhanced computational capability of a network of resource-constrained radio nodes, which is made possible by leveraging network computational resources, can improve the performance of complex applications in terms of metrics such as the application's time-to-completion. These benefits also project WDC as an excellent candidate enabling technology for green radio networks which are composed of radios that are expected to operate from eco-friendly constrained power sources such as solar power.

The load-shedding benefits of WDC are negated by the cost of communication between the distributed processes on collaborating radio nodes, and the uncertainty induced by the wireless channel. These factors are a major concern in WDC design and workload allocation.

Chapter 5 presents results which show that WDC is beneficial over local processing under certain conditions, particularly when the computational cost dominates over the communication overhead. It is relatively easier to realize these conditions, for instance, when complex computational tasks are distributed in short-range networks characterized by good quality wireless links.

2.2.3 Typical WCCN Application Scenarios and Use Cases

Various categories of WCCN applications are presented in this section along with some example applications. WCCN applications can be categorized based on the purpose of distributed computing, based on the application's operational objectives, and based on the layer at which the application resides in the wireless network protocol stack.

Based on the purpose of distribution, WCCN applications can be broadly classified as load distribution applications and distributed applications. A load distribution application can function entirely in one node, however it may benefit from the distribution of the application's workload among several nodes. In a load distribution application, a node, designated as the master node, delegates its computational workload to other nodes, designated as slave nodes, in the collaborative network. Section 3.9 describes an example tactical UAV scenario which benefits from workload distribution. In the class of distributed applications, an application is inherently distributed. It cannot function entirely in one node and requires several nodes for its complete execution. For instance, distributed MIMO, which requires coordinated physical layer actions of two or more nodes, can be viewed as a distributed application.

Applications can further be categorized based on their objectives. Network centric applications, for instance, operate to serve a common network mission. On the contrary, node-centric applications have individual goals and, accordingly, the individual nodes may act in a greedy manner with limited emphasis on network-wide objectives.

WCCN applications can also be classified according to the layer in a wireless network at

which distributed computing operates. Distributed waveform applications, such as distributed MIMO, cooperative relaying, and distributed signal detection & classification, are closely integrated with the physical layer communications system. On the other hand, distributed user applications, such as video processing and application-layer decryption, apply distributed computing to higher layers of a protocol stack.

Other example classes of applications which can potentially benefit from WCCNs are listed as follows:

- Real-time data capture, processing and dissemination - These applications involve real-time data collection and processing in order to provide time-critical information to a remote node.
- Complex communication waveform generation - Depending on QoS requirements and channel conditions, robust communication signal processing algorithms often involve complex computations which can be executed in a distributed manner.
- Information sharing - Real-time sharing of video, images, and voice data may require complex information coding and ad-hoc network capabilities.
- Robust and real-time distributed process control - Control applications require time-critical coordination among wireless sensors and actuators, as well as workload sharing to meet application's real-time requirements.

Example applications of WCCNs are: tactical wireless clouds; social networking applications; portable mobile scientific computing, including sensor data processing in WSNs; position location, target tracking, infrastructure monitoring, and surveillance; multi-media processing, including image processing, speech recognition, face matching, pattern recognition, and complex video decoding; distributed data storage and database search; Synthetic Aperture Radar (SAR) processing; collaborative jamming; distributed signal detection and classification; gathering data and mapping situational awareness; and collaborative gaming. A large

number of such nodes, which are capable of individually collecting and analyzing location-based data, can form an information highway that allows ubiquitous information access. In a military scenario where an expeditionary warfare unit is exploring an area, complex mapping data that is collected by nodes carried by individual foot soldiers can be segmented into pieces and stored locally amongst the nodes in a distributed database, instead of sending the data to a database server located at a remote base. This approach replaces communication with a remote server over insecure, unreliable links with communication between the WCCN nodes which is far more reliable, faster, and secure. The soldiers make updates to the map as they inspect the area. This user-generated data can be indexed and searched by user applications.

The security applications of this technology include distributed deciphering of complex security codes; distributed monitoring and assessment of networks, systems and building facilities; and multi-layer coordinated security attacks. WDC workload distribution can be applied to security applications which involve intense processing to decipher encrypted information. In distributed security monitoring applications, every node in the WCCN collects data pertaining to security of a facility or security of a network. This data can be made available to a remote central node which assesses security of the system being monitored in real-time and also predicts future attacks. Multiple nodes can coordinate in order to generate distributed security attacks. Collaborative jamming and distributed denial-of-service attacks are some examples. In addition, the technology can also be used to safeguard applications, and improve their fault-tolerance and security. Different parts of the code and data pertaining to an application can reside in different nodes, instead of having them all located in a single node. If processing over several nodes, a significant number of nodes have to fail or have to be compromised in order to adversely affect overall operation or to have access to all software, hardware, data, and codes. In a similar manner, WDC benefits highly secure tasks that require coordination among multiple units, where the tasks cannot be executed by a small number of compromised units.

WCCNs can operate as an overlay over a wide variety of networks including personal commu-

nication networks composed of smart phones and multi-media terminals; WSNs which collect and process large sets of data, sometimes in a real-time manner; cognitive radio networks which involve complex decision making; and tactical information networks which involve complex data and signal processing in order to gather military intelligence and enable highly robust communication among military units. In WSNs and CRNs, for instance, the various operations such as sensing, data processing and decision making can be delegated to different nodes in a WCCN environment. In addition to WCCNs, WDC can separately be employed in other systems such as wireless grids. Chapter 5 presents details on the application of WCCNs to Cognitive Radio Networks.

2.3 Survey of WCCN and WDC Design Challenges

WCCN research aims to extend the capabilities of traditional wired distributed computing to allow operation in heterogeneous, dynamic, uncertain, and resource-constrained wireless network environments, as well as meet challenges unique to WCCNs with the help of recently available enabling technologies. Although some of the distributed computing concepts developed previously can be used as a basis for research on WDC in WCCNs, they cannot be directly applied for WDC due to the unique challenges faced by it as compared to traditional distributed computing. In traditional networks, each node has been considered to be a powerful machine and, typically, the time required for local computation is ignored while analyzing distributed algorithms. In addition, node locations are static. Furthermore, communication latency observed in traditional short range and reliable WLAN-based distributed computing systems approaches that in wired distributed computing systems [6]. WDC, however, does not restrict itself to such network environments or WSN-like scenarios. It, instead, considers scenarios where the network comprises heterogeneous, and resource-constrained radio nodes (or cloud resources) which possess limited a priori knowledge of the uncertain and dynamic wireless environment in which they operate. Unique issues that arise when supporting distributed computing in such environments have not been addressed in a

holistic manner in the past.

Wireless environments possess the following characteristics: (a) stochastic variation in message transit times [6] and link quality due to dynamic and random spatio-temporal variation of wireless environment behavior, including uncertainty in channel data rates, retransmissions, and network processing and routing delays, (b) limited channel data rates caused by high probability of error, interference, and limited RF spectrum, (c) random link outages, and random node availability that impact network connectivity, and (d) dynamic variation in spectrum availability. In addition to the challenges imposed by the underlying wireless network, WCCN environments are characterized by limitations as well as stochastic availability of individual nodes' power resources (small form factor battery supply or harvested energy), computational power, and communication capacity; random node failures that can occur either due to calamity, software crashes or hardware malfunctioning, or dieing of power supply; and limited or no a priori knowledge of the availability of cloud services which can be accessed by a node at a certain time. In the case of a mobile environment, this situation is further complicated by the random mobility of cloud resources which incorporates dynamics into several aspects of the environment, including resource availability, connectivity between cloud entities, and accessibility to cloud resources. The intermittent connectivity in typical opportunistic networking scenarios [19] introduced by mobility and channel variations makes it hard to guarantee application QoS requirements and have stable knowledge of the MACNET environment state. While uncertainty in channel characteristics and availability of resources contribute to the negative effects of mobility on MACNET performance, mobility can assist with improvement in information diversity, particularly for social networking applications.

The unique characteristics of wireless environments and WCCNs directly impact the design and performance of WCCNs in the following ways: (a) uncertainty in meeting application QoS in harsh environments; (b) high power and energy consumption costs involved in communication between distributed computational tasks over wireless links compared to wired counterparts, which negates the benefits achieved by load distribution in WCCNs;

(c) non-deterministic design tradeoffs and scalability of WCCN system parameters, such as degradation of transceiver power consumption with range; (d) limitations in WCCN performance metrics, such as throughput and application time-to-completion, due to resource constraints and uncertainty in link qualities, particularly in scenarios where the wireless network acts as a bottleneck to user applications that involve high speed computations; and (e) challenge in design, execution, and evaluation (including correctness, efficiency, and promptness) of distributed computing algorithms in uncertain environments in addition to complexities involved with link layer design and operation in interference-prone channels. Resource constraints not only serve as an important motivation for WCCNs but also as a challenge in reaping benefits of WCCNs. When attempting to integrate wireless devices into a wired cloud/grid environment, for instance, resource-constrained wireless devices are limited in their resource contributions to the cloud environment.

It is challenging to port distributed computing concepts that have been designed for the traditional wired environments, such as cloud computing and grid computing, where the above listed challenges are greatly relaxed or absent. First, user interfaces of applications which have been designed for traditional distributed computing platforms will have to be adapted to fit within the form factor of the mobile device I/Os, such as small screens and keyboards with limited alphabets, as well as handle the heterogeneity in user interfaces [17]. In addition, the application may have to accommodate additional user modalities such as voice in order to allow specialized features such as voice invoked user-application interaction. Second, remote procedural calls in a traditional distributed computing environment keep an application transparent to the underlying network configuration as well as transparent to the location of the interacting software routines. However, dynamic network environment (and node mobility in a MACNET) would require the application to be aware of the dynamic configurations [20]. Third, security mechanisms developed for traditional distributed computing environments are both computationally and communication intensive. Security algorithms involve complex computations while security protocols involve back-and-forth interaction between the transmitter and receiver nodes. Such activities can cut into the

limited computational, communication and energy resources of WCCN nodes, and impose significant overhead on the collaborative computing process.

Issues which are critical to the practical realization of WCCNs include: synchronization; efficient and robust communication protocols; WCCN control and management; willingness among participating nodes to collaborate and share workload; robust resource allocation and fairness in load distribution in terms of energy and workload; security and trust issues pertaining to collaborative computing and data communication over the air; and reliability, in terms of computational accuracy and failure of nodes/links during application execution, in executing user applications in a distributed manner [12, 21]. Keeping in mind the drastic differences between wired and wireless distributed computing brought about by the aforementioned challenges, this section provides an overview on WCCN modeling approaches and focuses on five core design issues, namely, communications, synchronization, WDC control and management, and service management.

2.3.1 Novel Modeling, Performance Evaluation, and Design Approaches

WCCN models, design tradeoffs, fundamental performance limitations, and scalability in terms of network range, node density, and computational workload are greatly influenced by the underlying wireless network and social environment. This motivates the unique modeling and design approaches presented in this section.

Modeling and performance evaluation of WCCNs Traditional distributed computing models are restricted to consideration of inter-node communication overhead and computational service time distribution. An important research challenge is to develop holistic models and performance evaluation metrics which are unique to the dynamic WCCN environment and can differ vastly from those used in traditional distributed computing research. Example WCCN performance metrics (or QoS metrics) include power and energy efficiency;

application makespan or response-time distribution; cloud resource outage probability and blocking probability; WCCN throughput (in terms of jobs per unit time) distribution; and reliability. In order to analyze such metrics, a WCCN model must take a more holistic view of the WCCN environment and, hence, capture behavior of the service environment including uncertainty in knowledge on service availability; the distributed computing environment, including the dynamics of resource availability both at the individual node-level as well as the WCCN system level; and the underlying wireless network environment. In addition, the model should capture energy consumption, stochastic properties, and heterogeneity at various levels. Heterogeneity, for instance, may be observed in link characteristics, resource availability in different nodes, and node functionality. At a low level, a WCCN node is a system by itself which comprises three subsystems, namely, power, computation and communication subsystems, where each subsystem has different characteristics that can be evaluated independently.

While there has been extensive research on communication overhead [22], traditional distributed computing network models abstract out the low level details of channel characteristics in order to capture the perspective that is relevant for the distributed computing process. Traditional works do not take into consideration asymmetry in both link cost and resource availability, and broadcast and interference-prone nature of wireless links. Channel models in a WCCN model, on the other hand, must be extended to include realistic wireless network physical and link layer parameters such as randomly distributed SINR and capacity. Traditional failure analysis of algorithms in noisy channels [23], for instance, can be extended to account for space-time varying link quality. In this regard, it is expected that WCCN modeling must involve a combination of approaches used in traditional distributed computing and wireless networking.

System-level WCCN cost metrics take into consideration communication costs and computational costs, both in terms of time consumption and energy consumption. Consider a hypothetical scenario where a raw data bit is transmitted by a service requester node to a service provider node and processed at a certain rate by the service provider upon receiving

that bit. Accordingly, the following metrics can be defined. Energy per raw data bit accounts for both energy consumption for communication and computation purposes that is expended in processing one raw data bit. Throughput (bits processed per second) and makespan account for processing and communication delays experienced by each bit. Energy and power efficiency is another important WCCN performance metric. Network energy (power) savings [24] quantifies the energy (power) savings achieved by executing a computational task in a distributed manner as against performing the task locally in a single node. Fundamental analysis of load distribution efficiency using power consumption models has been presented in [24, 25], along with results on limitations and associated tradeoffs of load distribution. It is important to understand this tradeoff between local and distributed processing [12, 24] along with scalability and fundamental limitations imposed by the underlying communication subsystem before pursuing further research on developing architectures, protocols, and algorithms for WCCNs. In a load distribution scenario, it is important for a node to decide between the two forms of processing with regard to specific benefits such as energy, power, time and reliability in processing the computational tasks of an application.

Reliability may be expressed in terms of packet loss and uncorrected packet errors. Uncorrected packet errors can impact the distributed computing process by supplying a computational task with erroneous data. Reliability in executing user applications in a WCCN environment is impacted by various factors including: availability of RF spectrum and channels to communicate; link outages experienced while sending requests as well as while receiving results; outages in availability of computational resources due to contention from other service requests; and the finite buffering capabilities of both the communication queues and the resource/service queues. After a request successfully reaches a service provider, the request will have to be queued. In the practical case of a finite queue, if there is no remaining queue location to store the request, the request will have to be rejected. This is an outage of the computational resources. Furthermore, outages in power resources can impact while a request is being serviced. When the battery in the service provider goes low in energy, it can impact the processing rate of the service before it eventually dies out.

Design approach WCCN design has to take into consideration the aforementioned challenges in meeting application QoS requirements in such heterogeneous, dynamic, and resource constrained operating environments. It may be perceived it as a convergence of service management, distributed computing and wireless networking, along with consideration of the interdependence between power, computational and communication systems of a WCCN node, and interaction between the user application and underlying layers. Hence, WCCN research takes a cross-layer and cross-subsystem approach to system design. This approach, for instance, is evident from the design of WDC scheduling algorithms presented in Section 2.3.4. Traditional distributed computing systems focus on the minimization of computational and communication latencies and maximization of throughput. Other example objectives include makespan minimization of makespan and minimization of total completion time (which is the weighted sum of the completion times of all tasks in a task graph where the non-negative weights are indications of the importance of the task) [26]. In addition to these objectives, WCCNs operate with unique objectives such as minimization of power consumption, maximization of robustness, maximization of energy/power efficiency, and maximization of energy balancing, while also meeting resource constraints and application requirements. In WDC, for instance, there is a need for dynamic resource management mechanisms that optimize resource allocation over a multi-dimensional design space, and adapt to the dynamics of the operating environment and user needs.

2.3.2 Requirements and Challenges Imposed on Communications Design

A user application may be dynamically composed of services which have to interact with each other and are not hosted on the same node. In this case, the execution of the user application may involve interaction between services which reside in different nodes via message passing over wireless links. Communication between nodes is critical to WCCN operation in order to enable such inter-service interaction, as well as enable WCCN management and

control activities such as synchronization, remote process monitoring, and scheduling. Due to the critical importance of inter-node communication, communication subsystem robustness and reliability is key to the successful execution of a user application in a WCCN. Weinman [27] summarizes the impact of traditional infrastructure-based cloud applications on the network. Cloud computing applications can be highly demanding, being both data and computationally intensive. Interactive applications, for instance, involve exchange of real-time multimedia, including voice, high definition video, images, and text, and data intensive content such as 3D high definition video. In addition, the "pipes" connecting the outside world with the cloud possess diverse characteristics.

Some of the requirements and design consideration of the communication system in a WCCN are discussed below. The communication subsystem design should: (a) possess total independence in ownership of hosted communication services and resources along with availability of free unlicensed spectrum [18]; (b) support interoperability among diverse air interfaces; (c) provide secure links with tolerance to intentional jamming, eavesdropping, and other security attacks on the distributed computing process which originate from the wireless environment; (d) support security mechanisms and control over transportation of content; (e) provide reliable and robust links with grade-of-service and QoS guarantees to support real-time and critical distributed computing activities; (f) allow fine and direct control over communication parameters and resources in order to enable dynamic and opportunistic resource allocation; (g) provide high data rates with low jitter to support interactive applications and applications that exchange real-time multimedia data; (h) be autonomous, adaptive and cognitive with ability to perform self-healing and self-administering [17] using environment awareness; and (i) enable network configuration with focus on energy, power and time efficiency in order to meet resource constraints of portable mobile terminals, ensure long battery lifetimes, and, in turn, ensure uninterrupted operational readiness.

GoS guarantees and optimized communication performance in harsh, dynamic, and uncertain wireless network environments can be provided only by networks with cognitive capabilities and fine grained control over communication parameters. Cognitive radio networks can

dynamically update themselves with knowledge of the wireless environment state and use that knowledge in performing network-wide adaptations and re-configurations at various levels of the protocol stack in order to handle dynamic variations in network environment and meet the QoS requirements. Furthermore, interoperability between nodes with diverse air-interfaces can be practically feasible by employing cognitive radio techniques that enable identification of characteristics of an incoming signal and software radio techniques that enable dynamic re-configuration of the physical layer such that the recipient node can respond back to the transmitting node using the same signal format as the incoming signal.

Communication network architecture is an important design consideration. When analyzing wireless environments, three broad categories of network models can be identified, namely, infrastructure-based wireless networks (e.g., WLANs, cellular networks), infrastructure-less wireless networks (e.g., pure ad hoc networks for tactical applications,, vehicular networks, flat sensor networks, opportunistic networks), and wireless networks with partial infrastructure (e.g., mesh networks, hierarchical sensor networks,WSNs, infostations). Each one of these models presents different characteristics and issues. In other words, distributing the computational tasks in a WLAN where there is a base station that is an intrinsic coordinator for the wireless clients associated to it, is different with respect to the same task performed in a multi-hop ad hoc network where there are no centralized entities.

Energy efficiency is yet another important design consideration. The challenge is to keep energy consumption at a minimum in the presence of communication overhead imposed by WCCN operations such as information exchange, service discovery, and network probing. Communication overhead can drastically increase with random channel losses and high power consumption by transceiver components such as channel decoders and power amplifiers. Some of the basic functionality expected from the underlying communication system are discussed next.

Scatter and Gather: In a fundamental load distribution scenario, the master node "scatters" unprocessed data to the slave nodes. When the slave nodes finish processing the data assigned to them, the master node "gathers" the results back from the slave nodes. The

scatter and gather operations differ fundamentally between WCCNs and traditional wired distributed computing systems. Unlike the point-to-point bi-directional nature of wired links, the wireless link is inherently broadcast and prone to interference. In order to minimize interference during gathering, WCCNs require an efficient MAC that schedules message transmissions in synchronization with the computational process execution.

Routing and Relaying: Multi-hop relaying is beneficial when the communication burden of the WCCN nodes can be shared by the relaying nodes. Traditional routing and relaying methods take into consideration the communication cost associated with different links and the availability of communication resources. However, in WCCNs that comprise SDRs where the communication functionality is implemented in software, computational burden and the availability of computational resources have to be taken into consideration [25]. This is particularly true for relaying methods that require significant amount of computational resources in order to execute channel decoding algorithms. Hence, several communication and computational factors influence WCCN routing decisions, such as connectivity (relaying between two disconnected processes), interference, joint scheduling of computational and communication processes, channel selection, and allocation of both communication and computational resources. This problem is more challenging in heterogeneous networks where the computational resource availability varies between nodes.

Buffering: Consider a fundamental distributed computing scenario wherein an application generates data which has to be transmitted to another node over a wireless link. Limited and variable data rates over wireless channels can pose a bottleneck to applications which involve fast computations and generate data at a high rate. In such cases, buffering between communication and computational subsystems can control message losses and delays. However, the limited buffering capacity of portable WCCN nodes makes it challenging to address this problem.

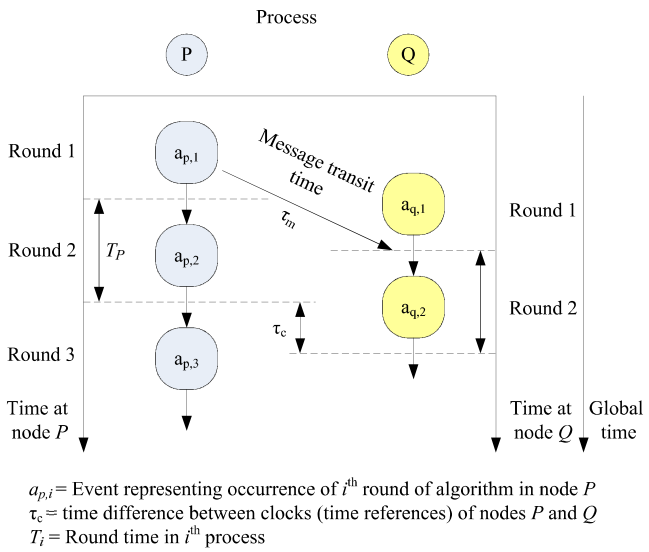


Figure 2.2: A fundamental synchronization problem in distributed computing.

2.3.3 Synchronization

Some distributed applications require the computational activities on different nodes in WCCN to be synchronized with each other. Synchronization is particularly important when the distributed computing activities have to occur in a certain order and there is dependency of one activity on another while executing the application. Synchronization involves the establishment of a temporal relationship between these activities [28]. In WCCNs, synchronization is a challenge when the computational processes which have to be synchronized are heterogeneous in terms of their execution times, and execution times and message transit time are uncertain. Execution times are particularly random when there are several computational activities contending for limited computational resources in a node. The two main aspects of synchronization, namely, process synchronization and clock synchronization, are discussed next.

Process synchronization Different computational events can occur during each execution round of a computational process. If event $a_{q,r+1}$ in round $r+1$ of process Q has to occur after event $a_{p,r}$ in process P, then there should be a way for the events to be synchronized.

Upon completion of event $a_{p,r}$, process P sends a message to process Q thereby triggering event $a_{q,r+1}$, as shown in Figure 2.2. The sequence of events in process P has to be executed such that the message sent after completion of event $a_{p,r}$ reaches process Q before event $a_{q,r+1}$ occurs. This is the fundamental synchronization problem [28] which is prevalent in both traditional DC and WDC. The relative time difference, denoted by τ, between processes P and Q, can be a random variable. Channel uncertainty in WCCNs poses a challenge to the determination of message transit time bounds that are critical for synchronization. In addition to synchronization of the individual computational and communication processes on all the nodes, it is a challenge to synchronize the two intra-node processes within individual nodes in order to optimize buffering between them.

Clock synchronization The local clock on each node needs to be synchronized to the network's global clock for various purposes such as time stamping of events, network re-synchronization, and fault tolerance [28]. Unbounded message transit times and differences between local clocks in WCCNs can pose a problem when clock synchronization is based on message time stamps. In addition, incorrect clock readings provided by erroneous or malicious nodes can affect synchronization.

2.3.4 Control and Management in WCCNs

Some of the control activities involved during distributed application execution in WCCNs include enabling cooperation between the distributed computational processes, segmentation of an application into tasks, and optimal WDC resource allocation. Control can be either centralized, distributed (possibly concurrent), or a combination of both. In the case of distributed control, the consensus among nodes is made through agreement protocols. Computational burden, communication overhead, and reliability of control mechanisms in dynamic environments are of concern in WCCN design.

Leader election Many distributed computing applications require a leader process that can perform centralized control functions. In a general WDC load distribution scenario involving a cluster of slave nodes and a master node, the leader process may reside either in the master node, a slave node or any node outside the cluster. In addition to existing challenges to leader election, WDC leader election must consider both computational and communication costs incurred by the potential leader. In distributed control, the control functions can be allocated to multiple leader nodes based on their ability to execute those functions.

Topology control and mobility management Clustering and topology construction involve the selection of a subset of collaborative nodes based on link quality and resource availability in order to construct a topology (such as a spanning tree) that optimizes objectives such as power consumption, connectivity, network reliability, and resource availability.

Mobility management is of concern to MACNETs. The impact of mobility on reliability and performance of MACNETs is multifold, as it impacts both forward and reverse link communication, and computational workloads at the provider. First, mobility impacts quality and reliability of the wireless link between the service provider and service requester. Consider a simulation model of user application execution in a MACNET environment which has knowledge of the sequence of communication and computational activities that need to be carried out while executing a user application, as well as the allocation of these activities to specific nodes and wireless links. Reliability of the application execution can be analyzed assuming a certain node mobility model by determining the link outage probability experienced by the application at the time when a communication activity needs to be executed. Reverse link communication reliability is an important consideration in addition to the performance of forward link communication. In the case of a single hop network where there has to be a direct link between the service provider and service requester without using any intermediate relays, link failure at the time of reverse link communication can completely fail the entire user application execution even though the service provider has successfully completed pro-

cessing the request after having successfully received the request from the requester. In the case of a multi-hop network, the provider has the option of sending the service request results back to the requester via reliable relays. Second, mobility causes the provider to move into a region which may have a high density of service request arrivals where the arriving requests are of higher priority than the request being currently serviced. In such as scenario, the currently serviced request may be preempted by the higher priority requests thereby causing the wait time of the current request to increase beyond acceptable levels. Mobility across subnets and domains can also have an impact on several aspects of network protocol stack such as IP addressing.

Resource management: mutual exclusion and resource access The mutual exclusion problem involves the "allocation of a single, indivisible, and non-shareable resource that can support only one user at a time", among n concurrent entities [3]. This problem is prominent in WCCN environments where resources must be shared in an opportunistic manner. In general, a user process requests the resource's manager process for access to the resource. Interactions between these processes, as well as interactions among various manager processes, are of equal importance in the exclusion problem. Each resource maintains a queue of indices of user processes that require access to that resource [3]. It is important to access the current queue state including resource occupancy and user position in the queue.

Resource access in WCCNs faces two challenges, namely, queue access fairness and stacking of resource queues. Traditionally, a process can instantaneously enter a resource queue. In WCCNs, however, random variation in message transit delays between different links may not allow fair queue access. In addition, a user process would have to contend for intermediate network resources (such as spectrum access and relay node computing resources) before gaining access to the required resource's queue. Moreover, any interaction between the resource process and the user process has to surpass the communication network resource queues.

Resource allocation: task scheduling and workload allocation WDC resource allocation in WCCNs involves the allocation of power, computational and communication resources in a WCCN for processing user service requests. The two main aspects of WDC resource allocation in WCCNs are workload allocation/distribution (or spatial task allocation) and task scheduling (or temporal task allocation). Workload allocation involves allocation/delegation of computational tasks of a user application to nodes in the WCCN while scheduling involves temporal ordering of task executions within the WDC nodes and communication between task in different nodes. This results in load shedding or load balancing, setting optimal parameters for communication and computational parameters; and scheduling the communication and computational activities on the resources, with the intent that it optimizes WDC operational objectives.

In traditional homogeneous distributed computing networks with deterministic channels, efficient resource utilization can be achieved by simple uniform workload allocation. However, WDC resource allocation in WCCNs must consider several factors with cross layer and cross subsystem influences in a dynamic environment, resulting in the search for an optimum scheduling strategy over a multi-dimensional design space. WDC task allocation can be viewed as a problem of mapping the application's task graph to the network's communication graph. The task graph and communication graph are introduced in Chapter 3. WDC resource allocation in WCCNs can be formulated as an optimization problem which takes into consideration the following factors: (a) application QoS requirements; (b) resource availability, current network state and resource constraints specified by the communication graph, which include channel conditions, current processor load, and availability of computing and communication resources; (c) cost functions that quantify both communication and computation cost; and (d) WDC operational objectives. such as throughput, fairness, reliability, concurrency, and overhead (energy/power, communication, and time). Application QoS requirements can be expressed in terms of makespan (or time-to-completion); power/energy consumption (including network energy consumption and energy consumption per node); reliability; energy balancing and fairness; and resource utilization. Time-to-completion or

makespan of an application is the total time it takes to complete execution of all computational tasks in the application from start to finish as per its execution schedule. Power consumption minimization is yet another important design consideration due to two reasons. First, power consumption directly impacts energy consumption. Second, power reduction is critical in the case of systems operating from green energy sources, such as solar power, which are power limited and not as much energy constrained. In addition to high level objectives, sometimes low level objectives, such as maximization of concurrency and minimization of communication overhead in energy/power and time, may be considered. Some important design aspects of a task allocation and scheduling algorithm include: (a) problem formulation, (b) system model, (c) objective or selection criteria, (d) type of scheduling algorithm, and (e) metric for evaluating quality of the schedule. The system model takes into consideration the costs associated with executing tasks as per the task allocation and scheduling scheme determined by the resource allocation algorithm, the availability of resources, and assumptions on the characteristics of parameters in the scheduling problem, such as the probability distribution of the job processing times, job arrival times, and channel communication time. One of the main considerations of resource allocation is the location of data and code which are required for the execution of the application. The required data and code will have to be virtually co-located to enable the source code to act on the data. If, for instance, data is located in node A which has limited computational capabilities and the source code is located in node B which has ample computing resources, the data may have to be transmitted to node B where it can be processed. In scenarios where the service requester uses the computational resources of a service provider in order to execute its application, the requester may have to transmit the source code to the provider for the execution. The source code may be supplied either in a pre-compiled form or in its original form as a text file. In the former case, the resulting object or library can be dynamically linked with other objects hosted by the provider in order to produce an executable of the desired application. In the latter case, the overall execution time may be significantly increased since the service provider will have to build the source code supplied by the requester. Secure transmission

of source code across nodes is a critical requirement of the system and security mechanisms can significantly increase the overhead.

The resource allocation factors listed above can span across the application and communication layers, as well as across the communication, computational and power subsystems. The choice of the workload measure to formulate the problem can sometimes significantly impact the performance improvement [29]. It is to be noted that power and energy optimality are clearly distinguished in WDC design [24, 30]. Task allocation may also consider synchronization, fault tolerance, and the security of individual nodes. The presence of stochastic environmentally-harvested power sources, such as solar power cells, adds an additional challenge to the already complex resource allocation problem.

Some WDC resource allocation problems are pointed out. Processing rate can be assigned based on task concurrency and execution time of tasks in each node, in order to minimize time-to-completion or power consumption. Workload allocation can minimize energy consumption by assigning message traffic onto a link based on its channel quality and by allocating computational workload based on cost of executing a task on the node under consideration. Adaptive resource allocation to achieve equal energy consumption across all nodes, particularly when channel conditions vary between links and processor loads differ between nodes, can ensure graceful degradation of battery life on all network nodes which, in turn, maximizes network lifetime.

Scheduling should also take into consideration the dynamics of network connectivity and resource availability, and the resulting time-variant node availability. If a participating node gets disconnected from the cluster while the application execution is in progress after workload allocation, there are two options: (a) the node's completed task results can be relayed back to the cluster, or (b) "service-handoff" can occur from the disconnected node to appropriate nodes within the cluster. Task migration enables a node that experiences node and/or link failures to handoff its computing process to another node that is available to pursue the computation. In a similar manner, the WDC control mechanisms have to take

into consideration the addition of computational resources by nodes that are newly admitted into the WCCN.

Robustness, in terms of application execution time variance, of different workload allocation schemes, such as uniform allocation, allocation considering mean channel condition, and allocation considering both channel mean and variance, has been analyzed for WDC networks [31]. It has been reported that the latter scheme can reduce execution time variance, improve robustness dramatically [31], and improve power availability for computing purposes by 14-22% [32]. The scheme simultaneously minimizes communication power consumption and maximizes computing rate. In this manner, nodes located in favorable channel conditions (smaller distances and channel variance) are allocated a higher computing rate. Due to the smaller communications cost in these nodes, the ratio between computing and total power consumption can be improved for the WDC network.

Comparison of WDC resource allocation in WCCNs and WSNs In WSNs, distributed computing resource allocation issues unique to uncertain wireless environments have not been completely addressed. For instance, Olsen et al. [12] identify the need for energy management both in the individual nodes' processors and in the wireless network. However, the energy resources are managed separately for the computation and communication aspects. Contrary to traditional design approaches, WCCN resource allocation intends to take into account cross-subsystem and cross-layer interactions in order to achieve a globally optimal WDC performance. A comparison between distributed computing resource allocation concepts developed for WSNs and the concepts employed for WCCN design is summarized in Table 2.1.

Deterministic optimization tools have been employed in distributed resource allocation in WSNs [8, 9]. Techniques have been classified into static and dynamic task allocation [10]. On the other hand, stochastic optimization tools will be employed for WDC resource allocation. In addition, WDC allocation techniques for WCCNs need to dynamically adapt to

Table 2.1: WDC Resource Allocation in WSNs and WCCNs.

Design Principle	WSNs	WCCNs
Optimization techniques for resource allocation	Deterministic optimization tools have been employed [8,9]. Techniques have been classified into static and dynamic task allocation [10].	Stochastic optimization tools will be employed. The allocation techniques need to dynamically adapt to the varying environment conditions.
Goals of resource allocation	Energy minimization and balancing are some of the key goals. Distributed computing is not designed to operate under outages.	Both robustness and energy goals are important. WDC system outage statistics will be taken into consideration in the resource allocation.
Cross layer optimization with link adaptation	In general, sensor nodes do not have adaptive radios because the network topology is static and sometimes known a priori. For instance, the transmit power has been assumed to be fixed in [9].	Link adaptation and cross layer optimization will be employed for dynamic performance. The focus will be on development of cross-layer metrics/cost functions for optimization.
Joint resource allocation of communication and computation resources	Resources have been managed separately for communication and computation processes [9]. The impact of network topology and channel conditions has not been explored in [9].	Joint resource allocation of computing and communication resources will be performed at all protocol layers taking into account dependencies between the computation and communication aspects of WDC.
Adaptation of computational parameters	Computational parameters remain fixed after the resource allocation is performed [9].	Temporal changes in channel behavior impacts communication energy. Since the total energy resources are shared between communication and computation processes, computational parameters have to change too for optimal resource usage.

Table 2.2: WDC Resource Allocation in WSNs and WCCNs (contd.).

Design Principle	WSNs	WCCNs
Network homogeneity	Network is assumed to comprise nodes with same computational capability and radio features [8, 9].	Heterogeneity is important to explore. It poses an additional challenge to resource allocation.
Energy versus power optimality	Power has been considered [10] only in the context of computational rate. Since the communication aspects have not been dealt with, the true difference between power and energy optimality is not demonstrated.	Power and energy optimality have been distinguished from each other [24, 30]. Power optimality is particularly important when considering solar-powered radios where power is limited while energy is abundant. Energy optimality is critical in battery-powered radios which can supply relatively higher power but provide limited energy.
Trade-off between local and distributed computing	This tradeoff has been addressed in a limited sense in [12]. However, the impact of wireless network parameters on this tradeoff has not been addressed.	The tradeoff analysis jointly considers several communication and computation parameters [24, 30] and takes uncertainty in communication overhead into consideration.
Adaptive workload allocation	A uniform task allocation scheme has been proposed in [8]. An adaptive workload allocation scheme has been proposed in [9].	The proposed algorithms will adapt the workload allocation based on the resource constraints and channel conditions.

the varying environment conditions. WDC resource allocation goals vary between WCCNs and WSNs. In WSNs, energy minimization and balancing are some of the key goals. It is not designed to operate under outages. In WCCNs, on the other hand, both robustness and energy goals are important. In past WSN distributed computing work, resources are managed separately for communication and computational processes with limited consideration of network topology and channel conditions [9]. On the contrary, joint WDC resource allocation of computing and communication resources will be performed at all protocol layers in WCCNs taking into account dependencies between the computation and communication aspects of WDC. In past WSN-DC research, the network and tasks are assumed homogeneous. Network is assumed to comprise nodes with same computational capability and radio features [8, 9]. Network and task heterogeneity is both important and challenging to explore for WDC in WCCNs due to the broad variety of WCCN applications. WSNs and WCCNS also differ in the treatment of energy optimality and power optimality. In WDC research for WCCNs, power and energy optimality have been distinguished from each other [24, 30]. Power optimality is particularly important when considering solar-powered radios where power is limited while energy is abundant. Energy optimality is critical in battery-powered radios which can supply relatively higher power but provide limited energy. An important aspect of WDC research is analyzing the trade-off between local and distributed computing. This tradeoff has been addressed in a limited sense in WSN distributed computing research without consideration of the impact of wireless network parameters on this tradeoff [12]. In WCCN research, however, the tradeoff analysis jointly considers several communication and computation parameters and takes into consideration uncertainty in communication overhead [24, 30].

Network information management An important WCCN research challenge is to determine the amount of information that each node should have about the operating environment in order to reliably perform control tasks. Network information is used by the leader node to perform critical operations such as stability analysis, failure recovery, and analysis of

algorithm correctness [28]. The state of a WCCN environment is multi-dimensional as it has to account for the time-varying states of individual entities in the environment, including intra-node processes, node resources, and links. Keeping track of the dynamic variations in the multi-dimensional environment state can be both time-consuming as well as resource intensive. Hence, efficient and smart approaches to environment information gathering and storage are necessary for efficient and adaptive operation of the WCCN. When taking a distributed approach to information gathering in a WCCN environment, for instance, the dynamic global WCCN environment state can be sampled and quantized in a distributed manner before it is disseminated to leader node(s) or shared among the nodes.

2.3.5 Service Management and Execution

The discovery and publication of cloud services is an important aspect of service management. Service discovery is a proactive approach to gathering knowledge about available cloud services in a WCCN environment while service publication is initiated by the owner of the service. As other information gathering mechanisms, service discovery and publication techniques have to be dynamic and efficient in keeping track of changes in the dynamic service environment where service providers enter and leave the environment.

Universal interoperability is an important requirement of WCCNs. WCCNs can comprise entities that are equipped with diverse software and hardware platforms as well as diverse air interfaces. Such diversity makes it nearly impossible for services, which are hosted by nodes with different air interfaces, to interact directly without a common mode of interaction. There needs to be a universal ontology that can be subscribed by all nodes in the network. The ontology can be used by nodes to specify services, and share information on the availability, capabilities and parameters of services. In addition, heterogeneity in computational platforms can directly impact portability of services. Code that may be machine readable and optimized for one platform may not be portable to another platform. It is a challenge to ensure portability of cloud applications on various platforms hosted by the

cloud entities, whether resource-constraint embedded platforms or server-based platforms that are abundant in resources. Different platforms impose different limitations, such as user interface display size, memory and processor speed.

2.4 Architecture Design Factors and Implementation Issues

The realization and adoption of the capabilities expressed by theoretical WCCN models on actual hardware and in operational systems involves overcoming significant obstacles. This section discusses some architectural and implementation issues that may be faced when executing distributed applications in a general WCCN environment.

Software architecture and operating system challenges To realize WDC benefits, an underlying architecture [33] is necessary to support both SDR operations for physical layer connectivity and more general processing for distributed computing and user application tasks. Existing software architectures typically assume either role, but not both. Physical layer activities may be characterized by strict requirements on timing or bandwidth. On the other hand, application tasks can vary from long-running background activities to real-time applications that may resemble physical layer communications. In a SDR implementation, this spectrum of activities must be managed on a single platform with one or more shared processors, which creates conflicting interests between simultaneously operating computing tasks. Such conflicts over shared resources have not been accounted for in traditional physical layer models which assume a dedicated implementation composed of discrete processing components.

In addition, opportunistic computing in SDR based WDC is premised on the availability of underutilized processing capability. In practice, processor overhead or memory barriers may affect the ability to remotely target spare processor cycles over a wireless channel in an effi-

cient manner. Furthermore, if remote processing is readily accessible, distributed computing tasks must not preempt primary physical layer operations. In other words, software implementations of various components must provide the behavior of a discrete design, although the underlying structure may be composed of shared and limited processing resources.

Power consumption on real devices Real-world energy utilization may include effects not included in theoretical network power/energy consumption models. These models are understandably generic due to the multitude of differences that exist between different platform implementations. However, to achieve the desired management for prolonged functionality, whether at the individual node or network level, effects of implementation related critical factors on system design must be taken into account.

For tractability, theoretical computational power models often focus on processor performance characteristics. For example, fundamental physical relationships between processor voltage and switching frequency are easily abstracted for mathematical analysis [24]. However, physical properties on modern processors are not subject to direct manipulation by software applications. Furthermore, basic physical processor parameters do not account for all the computational energy draw in a node.

Modern processor management schemes include voltage scaling, frequency scaling, and, more recently, power gating at the processor level. These parameters are not directly manipulable but are accessed through processor interfaces which provide selectable discrete power states. In addition, power control mechanisms at a higher operating system level allow for larger scale manipulation of overall platform properties. For example, sleep and hibernation states may power down individual portions of the platform or the entire platform. Thus, static loads can be manipulated and enabled or shut down completely. Consequently, computational power management in real, modern processors is more appropriately viewed as state based rather than a combination of continuously variable parameters, which leads to a divergence from the approach of many theoretical models.

These features of power management, while difficult to abstract, are critical in computational energy determination on real systems. Application of these features to existing models may be feasible, depending on specific application requirements and this will need to be resolved for actual implementations.

Synchronization, determinism, and power constraints WDC implementations must address synchronization issues often implicitly assumed to be solved in theoretical models. The introduction of general purpose processors and software based communication stacks further complicates these issues as they add non-deterministic properties through contention over consolidated processor resources. A challenge in embedded real-time computing systems is the determinism required for synchronization and efficient usage of power management schemes are intertwined on multiple levels with numerous contradicting characteristics [33]. For example, the most power efficient processor states generally have the longest wake up and response times. In addition, within the operating system of a single node, the most deterministic behavior typically requires a higher degree of headroom, or unutilized excess processor capacity, in order not to jeopardize timing deadlines.

Topologies for wireless distributed computing Topologies and configurations can mainly be of the following three types [34]: (a) ad-hoc, (b) centralized, and (c) hybrid. Networks configured in an ad-hoc manner do not rely on any centralized infrastructure for command and control. The nodes in such networks act in an autonomous manner. In a centralized layout, on the other hand, a base-station node plays a central role in control and operations. A hybrid topology includes a mix of both ad-hoc and centralized infrastructure-based systems interoperating with each other. An example of such a topology is the commercial wireless grid which comprises a MANET that is connected to a wired grid infrastructure via one or more access points [17].

2.5 Chapter Summary

WCCNs enable powerful computing and extended service execution capabilities with the help of a collaborative network of small form factor radio nodes that operate with reasonably low capacity batteries. This chapter discusses some ways in which traditional distributed computing approaches may fail or not operate optimally in dynamic wireless environments. In this regard, the chapter introduces the potential challenges of WDC and WCCNs, which provide the motivation to bring about design changes that can address the challenges. On the implementation front, considerable challenges remain in order for practical WCCNs to become a reality. SDR-based implementations need to be developed with an expanded view of software architecture design that takes into account the characteristics of both physical layer communications and network computing. In addition to the technological factors discussed in this dissertation, the performance of this cloud computing paradigm is influenced by other factors, namely, societal and human factors, regulatory factors, policy factors, and economic factors.

Chapter 3

System Model and Notation

This chapter presents the system model of a simple WCCN operating environment and states assumptions for the analysis of the power and energy efficiency of WDC workload distribution. The main objective of the system model is to capture the various costs encountered while pursuing WDC in a simple WCCN environment, namely, power consumption, energy consumption, and latency. In addition, performance evaluation metrics that quantify the power and energy efficiency of WDC load distribution are presented in this chapter. While the models presented in this chapter capture the power and energy consumption characteristics at a macroscopic scale, details of the derivation of these models from low level models can found in Appendix A.

The system model presented in this chapter is employed in the analysis of the design tradeoffs and feasibility of workload distribution in WCCNs, and in the design and performance evaluation of WDC workload allocation and scheduling algorithms, as presented in Chapter 5.

This chapter is organized as follows. Section 3.1 presents an overview of the system model. The computation model, communication model, and a model of the power supply resources are presented, respectively, in Sections 3.2, 3.3, and 3.4. The WDC workload allocation scenario used in this dissertation is described in Section 3.5. Based on the workload alloca-

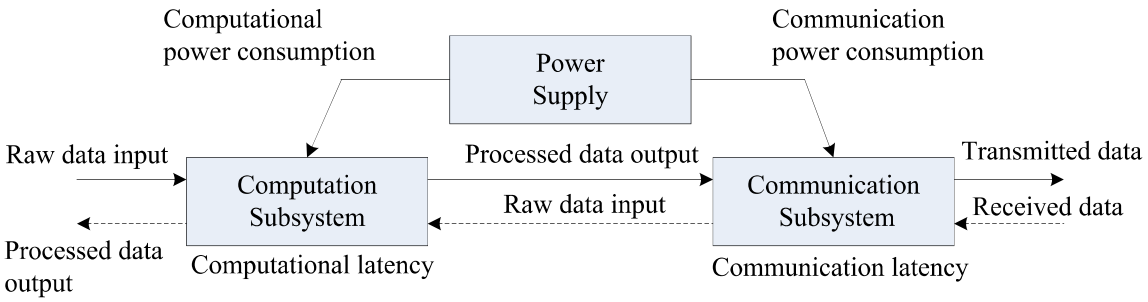


Figure 3.1: Interaction among the computation, communication, and power subsystems in a WCCN node.

tion model, the total power and energy consumption of WDC is presented in Section 3.6. The metrics used to quantify the efficiency provided by WDC are defined in Section 3.7. Section 3.8 presents the simulation parameters and their values used throughout this dissertation. A simulation analysis of an example WCCN application scenario is presented in Section 3.9. Finally, the chapter concludes with a summary presented in Section 3.10.

3.1 Overview

The WCCN environment under consideration primarily consists of N_{nodes} wirelessly connected radio nodes with computational capabilities. Each node comprises of communication, computational, and power subsystems that interact with each other as shown in Figure 3.1. Broadly speaking, the execution of a user application in a distributed manner involves computational activities and communication activities. A computational activity involves the execution of a computational task inside a node which has been assigned to execute the task. The communication subsystem connects the computational subsystems of respective nodes through wireless links and serves the WDC process by enabling message passing between the user application's computational tasks that are executed in different nodes and handling the communication aspects of WCCN management mechanisms.

Both homogeneous and heterogeneous WCCN environments are considered in this disserta-

tion. In a homogeneous operating environment, all nodes possess identical communication and computational subsystem characteristics. In addition, all constituent tasks of a user application have identical execution characteristics. In a heterogeneous environment, there can be three forms of heterogeneity in the environment. Heterogeneity can be observed in communication resources, computational resources, and the constituent computational tasks of the application. Computational capabilities and battery capacity may vary from node to node. Heterogeneity in link conditions is inherent to wireless environments where channel conditions can vary spatially, temporally, and across frequencies. The execution time characteristics, particularly the number of processor cycles consumed for execution, vary with the processor and the task. Heterogeneity in computational resources is inherent in some networks and is necessary in others. It is a necessity particularly when different computational operations execute with different efficiencies in different computing platforms. For instance, signal processing operations are efficiently executed in digital signal processors, control operations are performed using general purpose processors, and massively parallel operations can be executed in graphical processing units. Heterogeneity also exists among the computational tasks of an application, where the tasks differ from each other in terms of execution time, power consumption, and functional requirements.

3.2 Computation Model

The computation model employed as part of the overall system model represents the characteristics of tasks in the user application and the interaction between them, and captures the costs associated with execution of the application.

3.2.1 Task Graph Model of User Applications

The user application is modeled in terms of a task graph where the tasks may be either independent or dependent on each other. As shown in Figure 3.2, a user application is

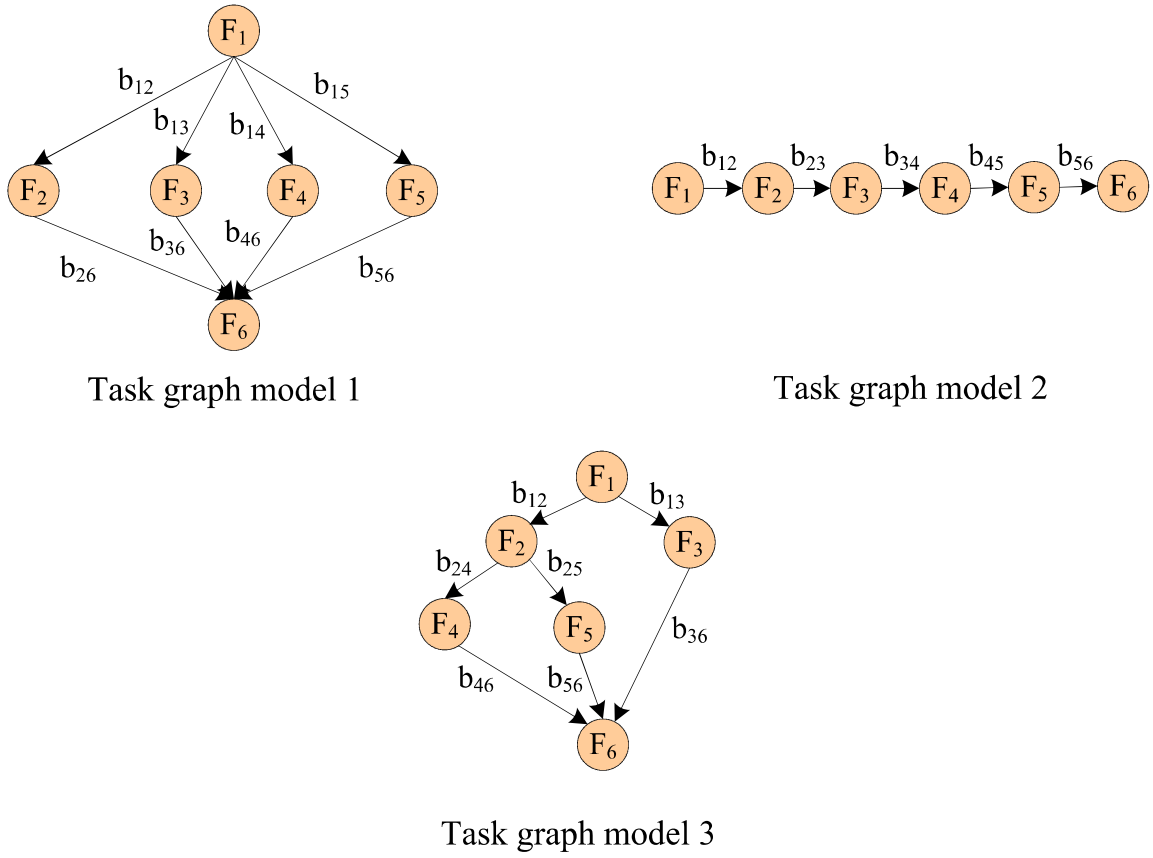


Figure 3.2: Task graph models considered in this dissertation.

segmented into N_{tasks} computational tasks. The computational tasks, the communication between them, and their precedence constraints can be represented in the form of a directional acyclic graph (DAG), denoted by $G_{cp} = (F, E)$. The set of tasks is denoted by $F = \{F_1, F_2, \ldots, F_m\}$, where $m = N_{tasks}$, and E represents the set of directed edges. Vertex F_i in the task graph represents an abstraction of the i^{th} task, which can include resource requirements and computational cost associated with that task. When the computational subsystem in a node processes a task, it accepts N_{bits}^{in} bits at the input and generates N_{bits}^{out} bits at the output such that $N_{bits}^{out} = R_{comp}\, N_{bits}^{in}$, where R_{comp} is a positive scaling factor similar to the data compression factor of a compression algorithm. In the case of a data compression task, for instance, $R_{comp} < 1$. A communication activity involves data passing between tasks that have been allocated to different nodes, specifically, those tasks that have

an edge between them in the task graph. According to the precedence constraint, a task cannot execute until its predecessor tasks finish execution and it receives messages from all its predecessor tasks. A precedence constraint is represented by an edge $(i, j) \in E$ in the task graph where F_j cannot begin processing before F_i is completed. In this manner, the edges between vertices in a task graph represent the dependencies (and precedence constraints) between the respective tasks and the communication activities between them. In a similar manner, the communication activity between F_i and F_j precedes over the communication activity between F_x and F_y if there is an edge of non-zero weight between F_j and F_x. Interaction between F_i and F_j is represented by the edge metric denoted by b_{ij}. The edge metric represents the number of information bits transferred between the tasks while the user application is being executed. In addition, the metric may also comprise of a set of communication QoS requirements such as tolerable latency. Data passing between tasks executed in different nodes occurs over wireless links while communication between tasks executed in the same node occurs internally. In the latter case, the data passing cost is assumed to be negligible. It is assumed that the computational costs and amount of inter-task data-passing are independent of the mode of processing, whether a pair of dependent tasks is executed in a single node or executed in separate nodes. There is, however, an additional overhead of channel coding that gets applied to the data in the case of communication over wireless links.

Application task graphs used in this dissertation comprise a single source task and a single sink task. The following task graph models, shown in Figure 3.2, are employed for the performance evaluation of workload allocation and scheduling in Chapter 5. In task graph model 1, all tasks other than the sink task operate independent of each other except for their dependency on the source task. N_{bits} bits of information is transferred over all non-zero edges. Task graph model 2 considers a task graph with no concurrency i.e., the tasks are connected along a line and N_{bits} bits of information is received by a task from its predecessor. In task graph model 3, the dependency and amount of information passed between tasks is randomly generated for a specified number of edges.

3.2.2 Computational Power Consumption, Energy Consumption, and Execution Time

Let E_{cp}^{ij}, P_{cp}^{ij}, T_{cp}^{ij}, and N_{cycles}^{ij} denote, respectively, the energy consumption, power consumption, execution time, and number of processor cycles consumed in order to process task F_i in node N_j. Let f_{cp} represent the processor clock frequency employed in the nodes. The computational energy and power consumption in a heterogeneous environment is given by

$$E_{cp}^{ij}(f_{cp}) = P_{cp}^{ij}(f_{cp}) \times T_{cp}^{ij}(f_{cp}), \tag{3.1}$$

$$P_{cp}^{ij}(f_{cp}) = a_{cp}^{ij}\, f_{cp}^{3} + b_{cp}^{ij}\, f_{cp}^{2} + c_{cp}^{ij}\, f_{cp} + d_{cp}^{ij}, \tag{3.2}$$

$$\text{where } T_{cp}^{ij}(f_{cp}) = \frac{N_{cycles}^{ij}}{f_{cp}} \tag{3.3}$$

$$\text{and } 0 < f_{cp} \leq f_{cpmax}.$$

f_{cpmax} represents the maximum processor clock frequency supported by the node's computational subsystem. Equation 3.2 expresses the non-linear polynomial relationship between P_{cp}^{ij} and f_{cp}, where the coefficients, namely, a_{cp}^{ij}, b_{cp}^{ij}, c_{cp}^{ij}, and d_{cp}^{ij}, are all positive and empirically related to the processor model and the computational task being executed. The coefficients a_{cp}^{ij} and b_{cp}^{ij} capture the non-linearity in the processor power-clock frequency characteristics. The computational power consumption model presented in Equation 3.2 is a high level abstraction of an existing CMOS-based processor power consumption model as shown in Section A.1 in Appendix A.

In the case of an application that uses the task graph model 3 comprising independent and homogeneous tasks, the computational sub-system parameters are no longer indexed by node and task. In this case, the total energy consumed in executing the application, denoted by

E_{cp}^{total}, is given by

$$E_{cp}^{total} = E_{cp} \times N_{tasks} = P_{cp} \times T_{cp} \times N_{tasks} = P_{cp} \times T_{cp}^{total}, \quad (3.4)$$

$$\text{where } T_{cp}^{total} = \frac{N_{cycles} \times N_{tasks}}{f_{cp}}. \quad (3.5)$$

In Equation 3.4, T_{cp}^{total} denotes the total execution time of the application when all the tasks are executed in a serial manner on a single core processor.

The following assumptions are made for analysis purposes throughout this dissertation. The computing platform in every node comprises a single core processor and there is no concurrency in task execution within a node. The computational power consumption of each task is assumed fixed and is the same for all nodes. Execution time is simulated as a uniform random variable in the interval $[1, T_{cpmax}]$, where T_{cpmax} denotes the range of the execution time. In this manner, the execution time differs between tasks and also differs for a task when executed on different nodes. This introduces diversity in the computational characteristics of tasks and nodes. The source and sink tasks, however, have a fixed execution time of 1 second.

3.3 Communication Model

This section describes the network topology and presents models of the communication power consumption, communication latency, and communication energy consumption.

3.3.1 Network Topology

The network is setup in a master-slave configuration and comprises only a single cluster with a single master node and several slave nodes. The master node and slave nodes communicate using a centralized, scheduled medium access control protocol that allows re-transmission of erroneously received packets. Communication occurs over a single-hop and the system

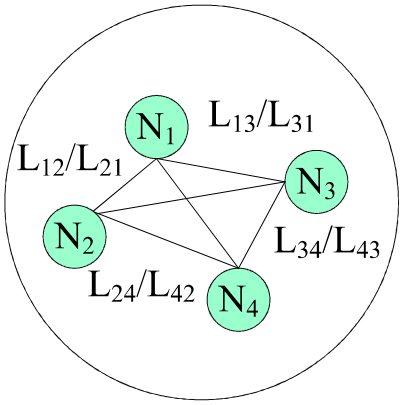


Figure 3.3: An example network communication graph model.

model does not consider the effects of routing on the system performance. The model also does not consider any specific transport layer protocol and, hence, does not take into consideration connection establishment and end-to-end flow control. The physical layer wireless channel model accounts for large scale fading that includes path loss and shadowing; small scale fading, specifically Rayleigh fading; and additive white Gaussian noise without any interference.

A network communication graph, denoted by $G_{cm} = (N, L)$ and shown in Figure 3.3, represents a network of nodes and wireless links between them. The set of nodes (or graph vertices) is denoted by $N = \{N_1, N_2, \ldots, N_m\}$, where $m = N_{nodes}$ and L denotes the set of wireless links between the nodes. The wireless link (or graph edge) between nodes N_i and N_j is denoted by L_{ij}. The graph model captures the heterogeneous characteristics of the wireless network environment. Nodes and links are abstracted by their attributes such as capacity, resource availability, and link quality. The edge metric, for instance, may comprise a set of link characteristics such as the inter-node distance, channel variance, link throughput, and transmission time per information bit. Let P^i_{supply}, E^i_{supply}, P^i_{cpt} and E^i_{cpt} denote, respectively, the remaining power supply, remaining battery capacity, power and energy available for computational purposes in node N_i. Nodes communicate with each other by passing messages over one of $N_{channels}$ communication channels (or RF frequencies), where each channel has a bandwidth BW. The network possesses a single-hop mesh topology where the point-to-

point connectivity is directly influenced by several factors such as inter-node distance and link quality. The network cluster is located within a finite bounded circular region of radius R_c comprising a finite number of nodes where the location of the nodes is modeled using a binomial point process. The inter-node distances are, then, computed from the binomial point process. Throughout this dissertation, node N_1 whose index is $i = 1$ refers to the master node.

3.3.2 Communication Power Consumption

In order to calculate the transmitter power consumption, two transmit power models are considered, namely, fixed transmit power model and adaptive transmit power model. The adaptive transmit power model takes into consideration the variation in the output transmitted power to compensate for variation in inter-node distances and channel conditions. This section first presents a transmitter power consumption model which is valid when the adaptive transmit power model is employed and the operating environment is characterized by log-normal shadow fading, path loss, and noise. This model is derived as shown in Section A.2. This model is extended to derive a statistical model of the transmitter power consumption in wireless environments characterized by Rayleigh fading, shadow fading, path loss, and noise when the communication subsystem operates at minimum power without any rate constraints. BPSK is chosen for the analysis since it is a simple power optimal modulation scheme that is well suited when rate is of no concern. Next, the power consumption model is presented for a transmitter that employs fixed transmit power.

Adaptive transmit power model During a communication activity over link L_{ij} which is characterized by path loss, log-normal shadow fading, and noise, the power consumption at the transmitter and receiver when N_i is transmitting to N_j are denoted, respectively, by P_{tx}^{ij} and P_{rx}^{ij}. Let D_{ij} denote the distance between nodes N_i and N_j, BW denote the receiver bandwidth, SNR_{min} denote the minimum required received signal power-to-noise

power ratio, and η_{pl}^{ij} denote the channel path loss exponent for link L_{ij}. Then, the power consumption at the transmitter for a given radio system can be expressed as a function D_{ij}, BW, and SNR_{min}, as given by

$$P_{tx}^{ij} = \left(G_1 \times SNR_{min} \times D_{ij}^{\eta_{pl}} \times BW\right) + G_2. \tag{3.6}$$

In Equation 3.6, G_1 and G_2 are constants that depend on the transceiver and channel characteristics and are defined in Equation A.19 of Appendix A. Although the receiver power consumption can vary with receiver parameters, such as decoding complexity and channel conditions, it is assumed fixed and equal for all nodes in this system model.

Next, the objective is to derive the mean transmitter power consumption, denoted by $\overline{P}_{tx}^{ij}$, when the wireless operating environment is characterized by Rayleigh fading, shadow fading, path loss, and noise. In the presence of small scale fading, the wireless system employs a simple adaptive transmission scheme with power control and ON/OFF transmission, where data is transmitted only when the channel conditions are favorable for BPSK transmission with required QoS. The transmitter power consumption model presented in Equation 3.6 can be extended to this wireless system, as given by

$$P_{tx}(t) = (G_1 \times SNR_{min} \times D^{\eta_{pl}} \times BW \times \alpha_{pc}(t)) + G_2, \tag{3.7}$$

where $\alpha_{pc}(t)$ is the instantaneous power control scaling factor which, ideally, is varied according to the instantaneous channel conditions. Note that, subscripts and superscripts that index the link are temporarily dropped from the notation until the end of this derivation. Let $\gamma_{rx}(t)$ denote the instantaneous received SNR and the corresponding mean received SNR be represented by $\overline{\gamma}_{rx}$. Let $g(t)$ denote the instantaneous channel power fading gain with a probability distribution denoted by $f_g(g)$ and $\overline{P}_r$ denote the mean received signal power. Assuming that the signal envelope undergoes Rayleigh fading, $g(t)$ follows a gamma distribution with parameters 1 and $2\sigma_g^2$ (or exponential distribution with parameter $1/\left(2\,\sigma_g^2\right)$), where σ_g denotes the standard deviation of channel gain. The instantaneous received SNR

is given by [35]

$$\begin{aligned} \gamma_{rx}(t) &= g(t)\,\overline{\gamma}_{rx}\,\alpha_{pc}(t), && (3.8)\\ \text{where } \overline{\gamma}_{rx} &= \frac{\overline{P}_r}{P_{noise}}, && (3.9)\\ \text{and } f_g(g) &= \frac{1}{2\,\sigma_g^{\,2}}\,e^{-g/2\sigma_g^{\,2}}\,H(g),\ g \geq 0. && (3.10) \end{aligned}$$

$H(g)$ denotes the Heaviside step function. When Rayleigh fading is assumed, $\sigma_g = 1$. Consider a simple power optimal adaptive transmission scheme that operates under no rate constraints where the transmit power is varied such that $\gamma_{rx}(t)$ is just sufficient enough to meet the minimum required SNR constraint. This scheme can be expressed as given by

$$\begin{aligned} &\min_{\alpha_{pc}(t)} \left[\gamma_{rx}(t) - SNR_{min}\right] && (3.11)\\ \text{S.T} \quad &\gamma_{rx}(t) \geq SNR_{min}. \end{aligned}$$

By substituting for $\gamma_{rx}(t)$ in Equation 3.11 using Equation 3.8 and assuming that the transmitted signal power is set such that $\overline{\gamma}_{rx} = SNR_{min}$, the problem may be solved by varying the transmit power level as given by

$$\alpha_{pc}(t) = \left\{ \begin{array}{rcl} \frac{1}{g(t)} & ; & \gamma_{rx}(t) \geq SNR_{min} \\ 0 & ; & \gamma_{rx}(t) < SNR_{min} \end{array} \right\}. \qquad (3.12)$$

This results in an adaptive scheme where the signal is transmitted whenever the channel gain is sufficiently high and the transmitter is switched OFF when the channel undergoes a deep fade. The minimum channel gain that can be compensated by power control, denoted by g_{min}, is related to SNR_{min} such that g_{min} becomes 1 when $SNR_{min} = \overline{\gamma}_{rx}$. Let P_{supply} denote the remaining power supply in a node. The resulting transmitter power consumption

can be expressed as given by

$$P_{tx}(g) = \left\{ \begin{array}{rcl} \left(\frac{G_3}{g}\right) + G_2 & ; & g \geq g_{min} \\ G_2 & ; & 0 \leq g < g_{min} \end{array} \right\}, \tag{3.13}$$

$$\text{where } G_3 = G_1\, SNR_{min}\, D^{\eta_{pl}}\, BW, \tag{3.14}$$

$$\text{and } g_{min} = \frac{G_3}{(P_{supply} - G_2)}. \tag{3.15}$$

Being a function of random variable g, P_{tx} is a random variable whose mean can be derived as given by

$$\begin{aligned} \overline{P}_{tx} &= E\{P_{tx}\} = \int_g P_{tx}(g) \cdot f_g(g) \cdot dg \\ &= \left(\lim_{\delta \to 0} \int_0^{g_{min}-\delta} G_2 \cdot f_g(g) \cdot dg\right) + \int_{g_{min}}^{\infty} \left(\frac{G_3}{g} + G_2\right) \cdot f_g(g) \cdot dg, \end{aligned} \tag{3.16}$$

where δ is an infinitesimally small interval that converges from the left. The integrals in Equation 3.16 are evaluated as follows:

$$\begin{aligned} I_1 &= G_2 \lim_{\delta \to 0} \int_0^{g_{min}-\delta} f_g(g) \cdot dg \\ &= G_2 \cdot \lim_{\delta \to 0} F_g(g_{min} - \delta) = G_2 \left(1 - e^{-g_{min}/2{\sigma_g}^2}\right), \\ I_2 &= \int_{g_{min}}^{\infty} G_2\, f_g(g)\, dg \\ &= G_2 \left[\int_0^{\infty} f_g(g)\, dg - \int_0^{g_{min}} f_g(g)\, dg\right] = G_2\, [1 - F_g(g_{min})] = G_2\, e^{-\frac{g_{min}}{2{\sigma_g}^2}}, \\ I_3 &= G_3 \int_{g_{min}}^{\infty} \frac{f_g(g)}{g}\, dg \\ &= \frac{G_3}{2{\sigma_g}^2} \int_{g_{min}}^{\infty} \frac{e^{-g/2{\sigma_g}^2}}{g}\, dg = \frac{G_3}{2{\sigma_g}^2} \int_{\frac{g_{min}}{2{\sigma_g}^2}}^{\infty} \frac{e^{-t}}{t} dt = \frac{G_3}{2{\sigma_g}^2}\, E_1\left(\frac{g_{min}}{2{\sigma_g}^2}\right);\ \frac{g_{min}}{2{\sigma_g}^2} > 0, \\ \text{where} \quad & \sigma_g \geq 0. \end{aligned}$$

$E_1(z)$ denotes the exponential integral which is defined as follows: $E_1(z) = \int_z^{\infty} \frac{e^{-t}}{t}\, dt$ for $|arg.z| < \pi$. The condition $|arg.z| < \pi$ is, anyways, met when the imaginary part of z is

zero. Thus,

$$\overline{P}_{tx} = \frac{G_3}{2\sigma_g^2} E_1\left(\frac{g_{min}}{2\sigma_g^2}\right) + G_2. \tag{3.17}$$

Fixed transmit power model Let P_t denote the transmitted signal power measured at the output of the power amplifier before it is input to the transmitter antenna, P_{noise} denote the noise power, η_{amp} denote the linear PA inefficiency (see Equation A.7), and γ_{tx} denote the transmitted signal power-to-noise power ratio. γ_{tx} is defined as given by

$$\gamma_{tx} = \frac{P_t}{P_{noise}}. \tag{3.18}$$

In the fixed transmit power model, γ_{tx} is fixed and independent of all channel characteristics except noise. Using Equation A.6, the transmitter power consumption can be expressed as given by

$$P_{tx} = (\eta_{amp} \times P_t) + G_2, \tag{3.19}$$

$$\text{where } P_t = \gamma_{tx} \times P_{noise}. \tag{3.20}$$

3.3.3 Communication Latency

In this section, communication latency models are presented for systems employing the adaptive transmit power model and the fixed transmit power model. In a single-hop wireless link, the main factors that impact the communication latency experienced by a packet include the propagation delay experienced during every transmission trial of the packet and the delay incurred due to re-transmissions. The number of re-transmissions is a random variable whose distribution depends on the probability of success (or link connectivity) experienced by a single packet transmission trial. The probability of success experienced over link L_{ij} is denoted by $P^{ij}_{success}$. A model of the probability of success is derived as the probability

that the received SNR meets the minimum required SNR constraint of the receiver. The following assumptions are made while deriving models of link connectivity and communication latency: (a) channel power fading gain does not vary during a packet duration so that the probability of success model presented in this section corresponds to a packet success probability, (b) the channel is decorrelated between packet transmission intervals, (c) the channel is stationary resulting in the fact that the stochastic properties of latency, including the probability distribution and mean latency, are fixed across all bits and all packets that constitute a user application data file, and (d) buffering delays at the transceiver are not taken into consideration while deriving communication latency models in this section. A latency model that incorporates queuing delays is derived in Chapter 6.

In a system that uses the adaptive transmit power model, the received SNR meets the minimum required SNR constraint always except during the times when the channel undergoes a deep fade. During a deep fade, the transmitter does not transmit any signals and, hence, the duration of the deep fade adds to the total delay experienced in transmitting data. Accordingly, the probability of success experienced over link L_{ij} is given by

$$\begin{aligned} P_{success}^{ij} &= Pr\left\{g_{ij} \geq g_{min}^{ij}\right\} \\ &= exp\left(-\frac{g_{min}^{ij}}{2\sigma_{g_{ij}}^2}\right). \end{aligned} \tag{3.21}$$

In the case of a channel model that accounts for shadowing fading and noise with no consideration of small scale fading, $P_{success}$ is simply the coverage probability $P_{coverage}$ (see Equation A.10).

When the fixed transmit power model is used, the received SNR can be computed using Equations A.15 and A.16. Accordingly, the probability of success in a system that employs

the fixed transmit power model is given by

$$\begin{aligned} P_{success}^{ij} &= Pr\left\{\gamma_{tx}\ g_{ij}\ D_{ij}^{-\eta_{pl}}\ \left(\frac{\eta_{amp}\ P_{noise}}{G_1\ BW}\right) > SNR_{min}\right\} \\ &= exp\left(-\frac{G_1 \times BW \times SNR_{min} \times D_{ij}^{\eta_{pl}}}{2 \times \sigma_{g_{ij}}^2 \times \gamma_{tx} \times \eta_{amp} \times P_{noise}}\right). \end{aligned} \tag{3.22}$$

A user application data file is fragmented into N_{pkts} packets and each packet is transmitted only after the previous packet has been successfully received at the receiver. T_{pktx}^{ij} and T_{bit}^{ij} are random variables that represent, respectively, the time taken to successfully transmit a packet and a channel bit over link L_{ij}; and $\overline{T}_{pktx}^{ij}$ and $\overline{T}_{bit}^{ij}$ denote their respective means. An additional superscript k is sometimes used as in T_{pktx}^{ijk} to represent the packet transmission time corresponding to the k^{th} packet in a data file. T_{tx}^{ij} is a random variable that represents the total time taken to successfully transmit a user application data file comprising of N_{data} uncoded bits which include the user application data bits and protocol overhead, and $\overline{T}_{tx}^{ij}$ denotes its mean. X_{pktx}^{ij} and X_{tx}^{ij} are random variables that represent, respectively, the number of packet transmission attempts that correspond to a single packet and total number of packet transmission attempts while transmitting a data file; and $\overline{X}_{pktx}^{ij}$ and $\overline{X}_{tx}^{ij}$ represent their respective means. X_{max} denotes the maximum number of transmission attempts permitted per packet which includes the first attempt and consequent re-transmissions. T_{pktp} and T_{bitp} denote, respectively, a single packet transmission time interval (or propagation time experienced per packet transmission attempt) and propagation time per channel bit. The number of uncoded bits in a single packet and the number of bits in a data file, where the bits include the user application data bits and protocol overhead, are denoted, respectively, by N_{pdata} and N_{data}. R_{code} denotes the channel coding rate. R_{txout} denotes the radio transmission rate expressed in bits per second and is given by

$$R_{txout} = R_{baud} \times log_2 K_m, \tag{3.23}$$

where $log_2 K_m$ represents the bits per symbol mapped at the modulator, K_m is the modulation

index, and R_{baud} is the radio transmission symbol rate (or baud rate) expressed in symbols per second. In order to comply with Nyquist's theoretical minimum bandwidth for baseband transmission as well as account for spectral expansion due to the constraints of signal filtering, the baud rate may be expressed as given by $R_{baud} = 1.4 \times BW$, where it is assumed that the baseband digital communication throughput is 1.4 symbols/sec/Hz [36].

The communication latency per channel bit and latency per packet over link L_{ij} follow a geometric distribution and are, respectively, given by

$$T_{bit}^{ij} = X_{pktx}^{ij} \times T_{bitp}, \tag{3.24}$$

$$T_{pktx}^{ij} = \sum_{l=1}^{\frac{N_{pdata}}{R_{code}}} T_{bit}^{ij} = X_{pktx}^{ij} \times T_{pktp}, \tag{3.25}$$

$$\text{where } T_{pktp} = \frac{1}{R_{txout}} \times \left(\frac{N_{pdata}}{R_{code}} \right), \tag{3.26}$$

$$T_{bitp} = \frac{1}{R_{txout}}, \tag{3.27}$$

$$Pr\left(X_{pktx}^{ij} = x\right) = \left(1 - P_{success}^{ij}\right)^{x-1} \times P_{success}^{ij},\ x \in \{1,\ 2, \ldots, X_{max}\}. \tag{3.28}$$

Equation 3.25 relies on the assumption that all bits in a packet undergo the same fading and possess the same stochastic properties. Equation 3.26 presents the deterministic duration of a single packet transmission time interval which involves the transmission of $\frac{N_{pdata}}{R_{code}}$ channel bits in a arbitrary packet transmission attempt. The mean latency per channel bit and per packet are, respectively, given by

$$\overline{T}_{bit}^{ij} = \overline{X}_{pktx}^{ij} \times T_{bitp}, \tag{3.29}$$

$$\overline{T}_{pktx}^{ij} = \overline{X}_{pktx}^{ij} \times T_{pktp} \tag{3.30}$$

$$\text{where } \overline{X}_{pktx}^{ij} = \frac{1}{P_{success}^{ij}}. \tag{3.31}$$

The packet latency model presented in Equation 3.25 can be extended to compute the total time taken to transfer a user application data file over link L_{ij}. T_{tx}^{ij} is characterized by a

negative binomial distribution and is given by

$$T_{tx}^{ij} = \sum_{k=1}^{N_{pkts}} T_{pktx}^{ijk} = X_{tx}^{ij} \times T_{pktp} \tag{3.32}$$

$$\text{where } Pr\left(X_{tx}^{ij} = y\right) = \begin{pmatrix} y-1 \\ N_{pkts}-1 \end{pmatrix} \times \left(P_{success}^{ij}\right)^{N_{pkts}} \times \left(1 - P_{success}^{ij}\right)^{y-N_{pkts}}, \tag{3.33}$$

$$y \in \left\{N_{pkts},\ N_{pkts}+1,\ N_{pkts}+2, \ldots, (N_{pkts} \times X_{max})\right\}.$$

Then, the mean latency per data file is given by

$$\overline{T}_{tx}^{ij} = \overline{X}_{tx}^{ij} \times T_{pktp} = \overline{T}_{bit}^{ij} \times \left(\frac{N_{data}}{R_{code}}\right), \tag{3.34}$$

$$\text{where } \overline{X}_{tx}^{ij} = \frac{N_{pkts}}{P_{success}^{ij}}. \tag{3.35}$$

In Equation 3.34, $\frac{N_{data}}{R_{code}}$ represents the total number of bits transmitted during a single data file communication activity.

3.3.4 Communication Energy Consumption

The energy consumption at the transmitter and the receiver, denoted respectively by E_{tx}^{ij} and E_{rx}^{ij}, for the transmission and reception of a single user application data file over link L_{ij} can be expressed, respectively, as given by

$$E_{tx}^{ij} = E_{tx1} + \left(\left(P_{tx}^{ij} - P_{trs}\right) \times T_{tx}^{ij}\right), \tag{3.36}$$

$$E_{rx}^{ij} = E_{rx1} + T_{rx}^{ij}\, E_{rx2} + E_{bdec}\, N_{data}. \tag{3.37}$$

E_{tx1}, E_{rx1}, and E_{rx2} are constants that are defined, respectively, by Equations A.26, A.29, A.30 in Appendix A. P_{trs} denotes the transient power consumption in the RF circuits of a transceiver. T_{rx}^{ij} denotes the reception time. E_{bdec} is the energy consumption to decode an information bit at the receiver and is treated as a constant. This general model is used as

a basis to derive energy consumption models for systems using the adaptive transmit power model and the fixed transmit power model.

Adaptive transmit power model Let E_{pktx}^{ij} denote the total energy consumption for successfully transmitting an arbitrary packet over link L_{ij} and $\overline{E}_{pktx}^{ij}$ denote its mean. A superscript, say k, is used as in E_{pktx}^{ijk} to denote the energy consumption associated with the k^{th} packet of a data file. E_{txb}^{ij} and $\overline{E}_{txb}^{ij}$ denote, respectively, the energy consumption for the transmission of an arbitrary channel bit over link L_{ij} and its mean. The corresponding mean energy consumption per channel bit at the receiver is denoted by $\overline{E}_{rxb}^{ij}$. E_{pktrl}^{ijk} denotes the total energy consumption for the first k transmission trials an arbitrary packet over link L_{ij}. Let P_{tx}^{ijk} denote the k^{th} instance of random variable P_{tx}^{ij} which has been defined in Equation 3.7.

If a packet transmission undergoes X_{pktx}^{ij} transmission attempts, the total energy consumption for transmitting an arbitrary channel bit and packet over link L_{ij} is given, respectively, by

$$E_{txb}^{ij} = \sum_{k=1}^{X_{pktx}^{ij}} \left(P_{tx}^{ijk} - P_{trs} \right) \times T_{bitp} \tag{3.38}$$

$$\text{and } E_{pktx}^{ij} = \sum_{k=1}^{X_{pktx}^{ij}} \left(P_{tx}^{ijk} - P_{trs} \right) \times T_{pktp}. \tag{3.39}$$

In Equation 3.38, it should be noted that the random transmitter power consumption is fixed for all bits within the same packet since the channel is assumed to be fixed over the entire duration of a packet. E_{pktx}^{ij} is a random variable that is a function of two random variables, namely, X_{pktx}^{ij} and P_{tx}^{ijk}. Hence, its probability distribution can be expressed as a mixture

distribution as given by

$$f_{E^{ij}_{pktx}} = \sum_{m=1}^{X_{max}} Pr\left(X^{ij}_{pktx} = m\right) \times f_{E^{ijm}_{pktrl}}, \tag{3.40}$$

$$\text{where } E^{ijm}_{pktrl} = \sum_{l=1}^{m} \left(P^{ijl}_{tx} - P_{trs}\right) \times T_{pktp} \tag{3.41}$$

The distribution of E^{ij}_{txb} can be modeled in a similar manner. Then, the mean energy consumption per packet and per channel bit are computed, respectively, as given by

$$\begin{aligned}
\overline{E}^{ij}_{pktx} &= \sum_{m=1}^{X_{max}} Pr\left(X^{ij}_{pktx} = m\right) \times T_{pktp} \times \mathbb{E}\left\{\sum_{l=1}^{m}\left(P^{ijl}_{tx} - P_{trs}\right)\right\} \\
&= \left(\overline{P}^{ij}_{tx} - P_{trs}\right) \times T_{pktp} \times \sum_{m=1}^{X_{max}} m \times Pr\left(X^{ij}_{pktx} = m\right) \\
&= \left(\overline{P}^{ij}_{tx} - P_{trs}\right) \times T_{pktp} \times \overline{X}^{ij}_{pktx}, \qquad (3.42) \\
\overline{E}^{ij}_{txb} &= \left(\overline{P}^{ij}_{tx} - P_{trs}\right) \times T_{bitp} \times \overline{X}^{ij}_{pktx} = \left(\overline{P}^{ij}_{tx} - P_{trs}\right) \times \overline{T}^{ij}_{bit}, \qquad (3.43) \\
\text{and } \overline{E}^{ij}_{rxb} &= \left(E_{rx2} \times \overline{T}^{ij}_{bit}\right) + \left(\overline{X}^{ij}_{pktx} \times E_{bdec} \times R_{code}\right). \qquad (3.44)
\end{aligned}$$

Let E^{ij}_{tx} and $\overline{E}^{ij}_{tx}$ denote, respectively, the total energy consumption for the transmission of a data file over link L_{ij} and its mean. The model of transmitter energy consumption per packet can be extended to define the transmitter energy consumption per transmitted data file as given by

$$E^{ij}_{tx} = E_{tx1} + \sum_{k=1}^{N_{pkts}} E^{ijk}_{pktx} = E_{tx1} + \sum_{m=1}^{X^{ij}_{tx}} \left(P^{ijm}_{tx} - P_{trs}\right) \times T_{pktp} \tag{3.45}$$

$$\begin{aligned}
\text{and } \overline{E}^{ij}_{tx} &= E_{tx1} + \left(\left(\overline{P}^{ij}_{tx} - P_{trs}\right) \times T_{pktp} \times \sum_{m=N_{pkts}}^{X_{max}\, N_{pkts}} m \times Pr\left(X^{ij}_{tx} = m\right)\right) \\
&= E_{tx1} + \left(\left(\overline{P}^{ij}_{tx} - P_{trs}\right) \times T_{pktp} \times \frac{N_{pkts}}{P^{ij}_{success}}\right) \\
&= E_{tx1} + \left(\overline{E}^{ij}_{pktx} \times N_{pkts}\right) \qquad (3.46)
\end{aligned}$$

Fixed transmit power model In the fixed transmit power model, the transmitter power consumption is fixed leaving only the communication latency to be random. Hence, the communication energy consumption parameters for link L_{ij}, namely, the transmitter's communication energy consumption per arbitrary channel bit, total energy consumption per data file, mean transmitter energy consumption per arbitrary channel bit, and mean receiver energy consumption per arbitrary channel bit are given, respectively, by

$$E_{txb}^{ij} = (P_{tx} - P_{trs}) \times T_{bit}^{ij}, \tag{3.47}$$

$$E_{tx}^{ij} = E_{tx1} + \left((P_{tx} - P_{trs}) \times T_{tx}^{ij}\right), \tag{3.48}$$

$$\overline{E}_{txb}^{ij} = (P_{tx} - P_{trs}) \times \overline{T}_{bit}^{ij}, \tag{3.49}$$

$$\text{and } \overline{E}_{rxb}^{ij} = \left(E_{rx2} \times \overline{T}_{bit}^{ij}\right) + \left(\overline{X}_{pktx}^{ij} \times E_{bdec} \times R_{code}\right), \tag{3.50}$$

where P_{tx} is computed using Equation 3.19.

3.4 Power Supply Resource Model

This section first presents the power supply models employed for simulation analysis in this dissertation. The section also presents the limitations and scalability in the availability of power resources for computational purposes in WCCNs. Two power supply models are employed in the simulation analysis, namely, the fixed power supply model and the random power supply model. In the fixed power supply model, all nodes in the network possess a fixed and equal remaining battery capacity. In the latter model, the remaining power supply in each node in the network is generated as a uniform random variable in the interval $\left[P_{supply}^{min}, P_{supply}^{max}\right]$.

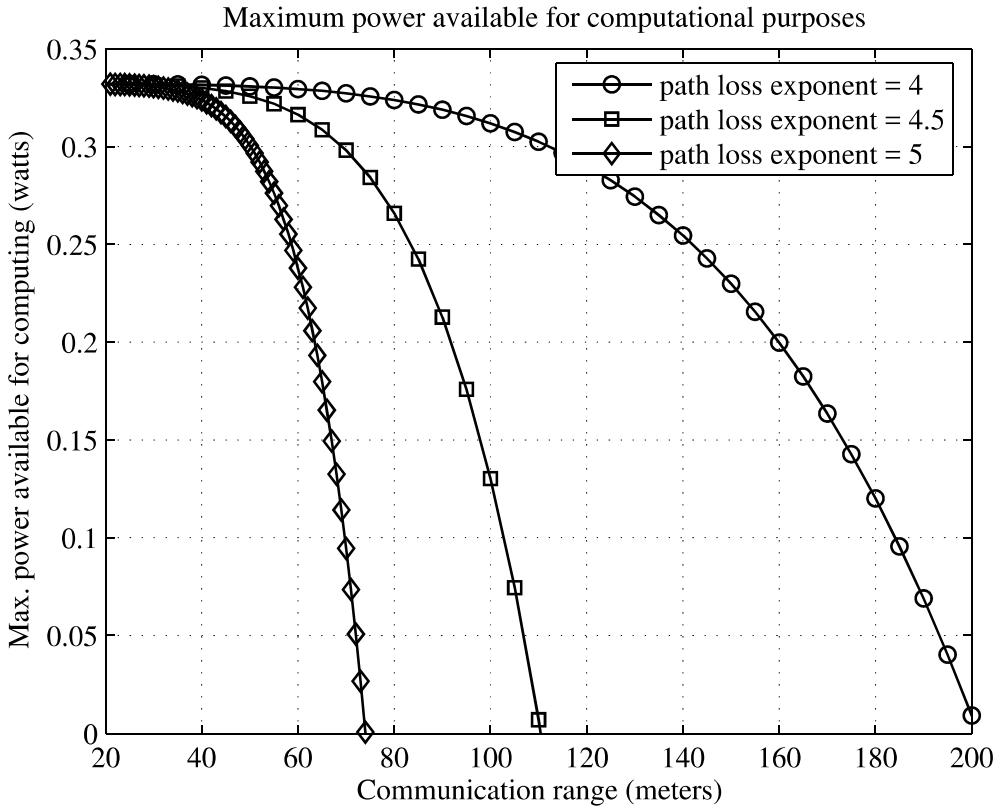


Figure 3.4: A tradeoff between communication range and maximum available power supply for computing under different channel conditions.

3.4.1 Power Availability for Computational Activities

A fundamental limitation of the computational capability of a WCCN node is encountered when the computational activities in the node run concurrently alongside the node's transmission activities, particularly under a power (or energy) supply constraint. This results in both the communication and computational activities drawing power from the same power source. This scenario is prevalent in software-defined radios where it is important to take into consideration the tradeoff between the communication and computational power consumption components. In this case, the maximum power available for a computational activity in

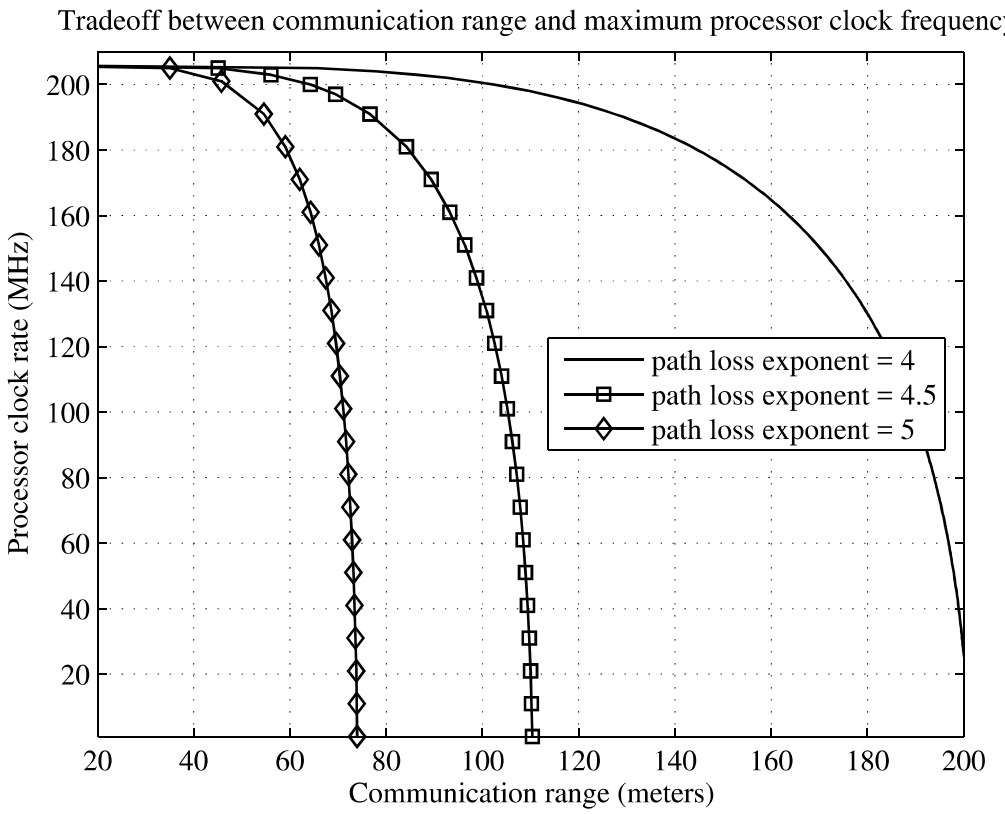


Figure 3.5: A tradeoff between computational capability and communication range under different channel conditions.

the node is given by

$$P^i_{cpt} = P^i_{supply} - (G_1\ BW\ SNR_{min}\ D_{ij}^{\eta_{pl}}) - G_2, \tag{3.51}$$

where D_{ij} corresponds to the distance between node N_i and its nearest neighbor N_j. Equation 3.51 can be manipulated to express the tradeoff between transmission range and processor clock rate as given by

$$D_{ij} \leq \left\{ \frac{P^i_{supply} - (a_{cp}\ f^3_{cp} + b_{cp}\ f^2_{cp} + c_{cp}\ f_{cp} + d_{cp} + G_2)}{G_1\ SNR_{min}\ BW} \right\}^{1/\eta_{pl}}, \tag{3.52}$$

$$\text{where } P^i_{supply} > P_{cp} + G_2. \tag{3.53}$$

The tradeoff can be derived in a similar manner in the presence of an energy constraint. Equations 3.51 and 3.52 portray the non-linear tradeoff between computational capability (expressed in terms of processor clock frequency) of the node and its communication range, as shown in Figures 3.4 and 3.5. The figures are plotted considering an example computational task executing on a StrongArm SA-1100 processor. It is assumed that the node has sufficient power supply available to operate at the processor's maximum clock frequency by setting P_{supply} to 0.66 watts in order to satisfy Equation 3.53. More details on the simulation values employed for this analysis are presented in Section 3.8. This fundamental limitation in the computational capability of a node can appear as a constraint in an optimization problem involving the design of WDC mechanisms such as resource allocation.

The power availability model presented in Equation 3.51 is extended to systems operating with the adaptive transmit power model in wireless channel conditions characterized by shadow fading, Rayleigh fading, noise, and path loss where the transmitter power consumption is a random variable. When a node engages in both communication and computational activities concurrently, the total power available for computational purposes can be defined as

$$P_{cpt}(g) = \left\{ \begin{array}{lll} P_{supply} - (G_3/g) - G_2 & ; & g \geq g_{min} \\ P_{supply} - G_2 & ; & 0 \leq g < g_{min} \end{array} \right\}. \tag{3.54}$$

The density function of P_{cpt} is derived next. The following preliminary observations can be made on the distribution of P_{cpt}:

$$F_{P_{cpt}}(p_{cpt}) = \left\{ \begin{array}{lll} 1 & ; & p_{cpt} > P_{supply} - G_2 \\ 0 & ; & 0 \leq p_{cpt} < P_{supply} - (G_3/g_{min}) - G_2 \end{array} \right\}. \tag{3.55}$$

Since $P_{cpt}(g)$ is constant in the interval $[0,\ g_{min})$, $F_{P_{cpt}}(p_{cpt})$ has a discontinuity at $P_{supply} - G_2$

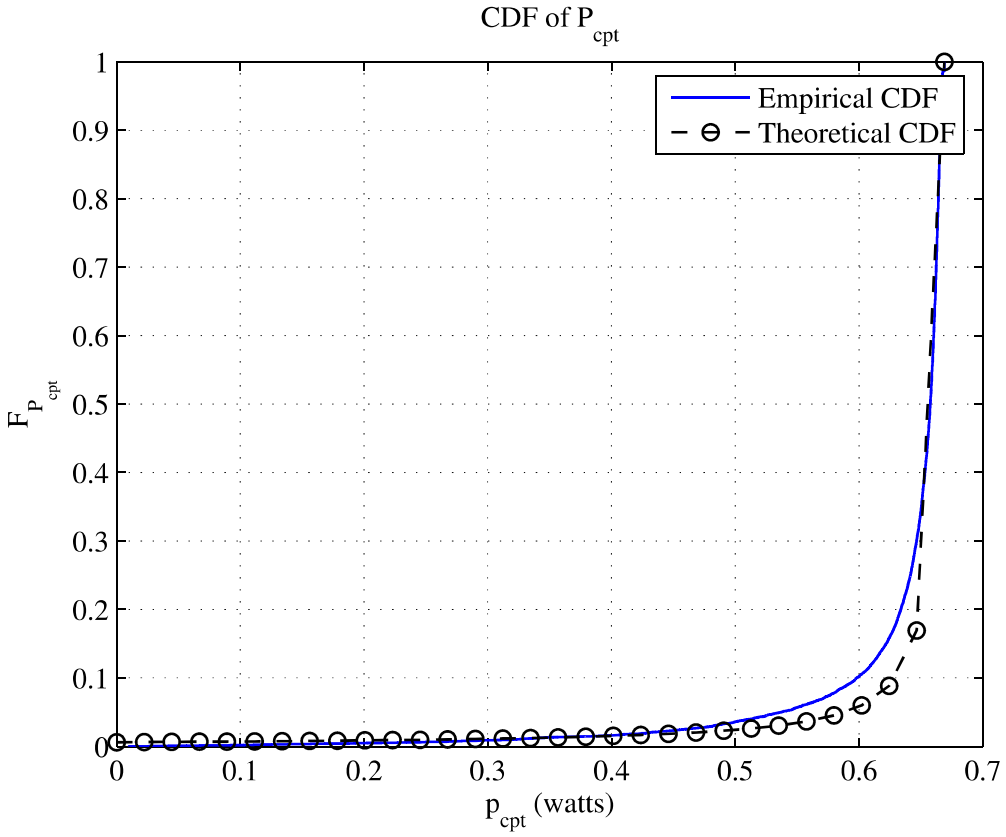


Figure 3.6: Comparison of the theoretical and empirical cumulative density functions of power availability for computing purposes.

as given by

$$\begin{aligned} P\{P_{cpt} = P_{supply} - G_2\} &= P\{0 \leq g < g_{min}\} \\ &= F_g(g_{min}). \end{aligned} \tag{3.56}$$

In the interval $P_{supply} - \frac{G_3}{g_{min}} - G_2 \leq p_{cpt} < P_{supply} - G_2$, the distribution function can be

computed as given by

$$\begin{aligned} F_{P_{cpt}}(p_{cpt}) &= P\{P_{cpt} \leq p_{cpt}\} \\ &= P\{P_{supply} - (G_3/g) - G_2 \leq p_{cpt}\} \\ &= P\left\{g \leq \frac{G_3}{P_{supply} - G_2 - p_{cpt}}\right\} \\ &= F_g\left(\frac{G_3}{P_{supply} - G_2 - p_{cpt}}\right). \end{aligned}$$

In summary, the distribution function can be expressed as given by

$$F_{P_{cpt}}(p_{cpt}) = \left\{ \begin{array}{ll} 1 & ;\ p_{cpt} > P_{supply} - G_2 \\ F_g(g_{min}) & ;\ p_{cpt} = P_{supply} - G_2 \\ F_g\left(\frac{G_3}{P_{supply} - G_2 - p_{cpt}}\right) & ;\ P_{supply} - \frac{G_3}{g_{min}} - G_2 \leq p_{cpt} < P_{supply} - G_2 \\ 0 & ;\ 0 \leq p_{cpt} < P_{supply} - \frac{G_3}{g_{min}} - G_2 \end{array} \right\} \quad (3.57)$$

where $F_g(g) = (1 - e^{-g/2\sigma_g^2}) \cdot H(g)$.

Figure 3.6 provides validation of the derived CDF by comparing it with the simulated CDF of P_{cpt}. The CDF is plotted for a communication range of 80 meters and power supply of 1 watt. The values of G_1, G_2, and other parameters used in the calculations are presented in Section 3.8.

Let P_{outage}^{cp} denote the application's tolerance of the probability of non-availability of power resources for computational purposes. Let r_{cpt}^i denote the power available for computational purposes in node N_i that meets outage constraint P_{outage}^{cp}. In the case of a system that employs the adaptive transmit power model, r_{cpt}^i is given by

$$P\left\{P_{cpt}^i \leq r_{cpt}^i\right\} = P_{outage}^{cp} \quad (3.58)$$

$$\Rightarrow r_{cpt}^i = F_{P_{cpt}^i}^{-1}\left\{P_{outage}^{cp}\right\}. \quad (3.59)$$

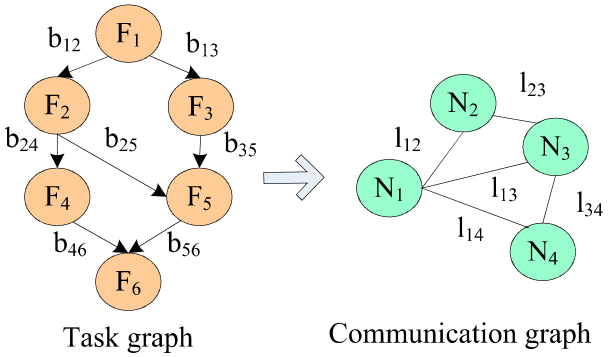


Figure 3.7: Allocation of tasks in a task graph to nodes in a communication graph.

Assuming a P^{cp}_{outage} that is neither 1, 0 or $F_g\left(g_{min}\right)$, r^i_{cpt} is given by

$$r^i_{cpt} = P^i_{supply} - G_2 - \left(\frac{G_3/2{\sigma_g}^2}{\log\left(1/\left(1-P^{cp}_{outage}\right)\right)}\right). \tag{3.60}$$

In the case of the fixed transmit power model, r^i_{cpt} is computed as given by

$$r^i_{cpt} = P^i_{supply} - P_{tx}. \tag{3.61}$$

3.5 WDC Workload Distribution Model

WDC workload allocation can be viewed as a problem of mapping the user application's task graph to the network communication graph, as shown in Figure 3.7. In a typical load distribution scenario, the master node receives a request for processing a user application with a total computational workload of N_{tasks} tasks. Then, the master node makes a decision on whether to process all the computational tasks locally, or delegate some or all of the workload to the slave nodes. It makes this decision based on evaluating the proposed WDC power (or energy) savings metric which indicates the power (or energy) consumption savings achieved by executing the application in a distributed manner in comparison to local on-board processing of the same application. If it decides to pursue distributed processing, the

master node and the slave nodes collaborate in processing the application's computational tasks in the following manner. Let N^i_{cpw} denote the computational workload assigned to the i^{th} node in the network. The master node internally processes N^{mr}_{cpw} $(= N^1_{cpw})$ tasks and delegates the remaining N^{sl}_{cpw} tasks to the slave nodes, i.e.,

$$N^{sl}_{cpw} = \sum_{i=2}^{N_{nodes}} N^i_{cpw}, \tag{3.62}$$

$$\text{and } N^{mr}_{cpw} = N_{tasks} - N^{sl}_{cpw}. \tag{3.63}$$

The master node transmits data pertaining to the N^{sl}_{cpw} tasks to the respective slave nodes. Upon processing the data, the slave nodes return the processing results back to the master node.

When uniform workload allocation is employed, the computational workload is uniformly allocated to all nodes in the network, including the master node and slave nodes, without any consideration for communication costs, computational costs, and resource constraints. The number of tasks assigned to the i^{th} node in the network is given by

$$N^i_{cpw} = \left\{ \begin{array}{ll} \lfloor N_{tasks}/N_{nodes} \rfloor & ; \quad i = 2 \ldots N_{nodes}, \\ N_{tasks} - ((N_{nodes} - 1) \times \lfloor N_{tasks}/N_{nodes} \rfloor) & ; \quad i = 1 \end{array} \right\}. \tag{3.64}$$

When an adaptive workload allocation algorithm is employed, the computational workload is adaptively allocated to nodes in the network.

When applying WDC workload distribution to an application that is structured based on task graph model 1, the network operates either in the power reduction mode or the computational latency reduction mode. The power reduction mode is appropriate when the application's latency requirement, given by N_{cycles}/f_{cp}, can be met by the master node itself, however the master node does not possess sufficient power supply to support the latency requirement. Assuming uniform workload allocation and a homogeneous network, all nodes in the network process the application's tasks in parallel, where each node operates its pro-

cessor at a lower clock rate given by $f_{cpd} = f_{cp}/N_{nodes}$. In the latency reduction mode, the individual nodes operate at the maximum processor clock rate and the network achieves a N_{nodes} fold improvement in computational latency. In both cases, it is assumed that the communication latency is negligible compared to the computational latency. The power reduction mode is not considered in the analysis of heterogeneous environments since it involves elaborate stochastic modeling of the relationship between computational and communication latency, which is outside the scope of this dissertation. Using the workload allocation model presented in this section, the total power and energy consumption of WDC is computed as described in the next section.

3.6 Total Power and Energy Consumption of WDC Workload Distribution

When executing a user application in a distributed manner, the collaborating nodes in a WCCN consume power and energy in order to run communication and computational activities involved in the execution of the application. There can be two metrics for network power consumption, namely, instantaneous network power consumption and peak network power consumption. Instantaneous network power consumption is a time-varying parameter which depends on the number of nodes that are actively transmitting at a particular instant of time. The system model in this dissertation uses the latter metric in order to take into consideration the worst case scenario when all collaborating nodes are transmitting concurrently. P_{cp}, P_{tx}, and P_{rx} are some of the power consumption components which are associated with computational and communication activities in a node. Each of these components have to individually meet a node's power supply constraint. The net power consumption in a node depends on the concurrency of these activities. In the ideal scenario where a node can process computational tasks, transmit data, and receive data simultaneously, the peak

power consumption in node N_i is given by

$$P^i_{node} = P_{cp} + \max_{\forall j}\left\{P^{ij}_{tx}\right\} + \max_{\forall j}\left\{P^{ji}_{rx}\right\}, \text{ where } i \neq j,\ j = 1 \ldots N_{nodes}. \tag{3.65}$$

In Equation 3.65, N_{nodes} represents the number of nodes in a WCCN that collaborate in executing a user application. The power consumption term $\max_{\forall j}\left\{P^{ij}_{tx}\right\}$ represents the worst case transmitter power consumption which is incurred when the node transmits to another node which is located farthest away it. If all activities in a node occur over non-overlapping time intervals, the peak power consumption of the node is the maximum of all the power consumption components. The communication and computational activities inside a node can be scheduled to run concurrently in order to comply with latency and buffering constraints, particularly to avoid buffering of large amounts of data that flows between the computational and communication subsystems inside a node. Alternatively, the activities can occur over non-overlapping time intervals in order to meet power constraints at the cost of latency. In this dissertation, the former scenario is assumed while performing power consumption calculations for simulation analysis. In the WCCN prototype presented in Chapter 4, the communication and computational activities are scheduled such that they never overlap in time. This is done mainly to avoid sharing of processor resources between the communication and computational activities in a node.

A general expression of network power and energy consumption corresponding to the distributed execution of a user application is presented with the following assumptions: (a) all tasks in the application are homogeneous and are executed independently as in task graph model 1; (b) all participating nodes in the network operate concurrently; (c) a node can only transmit or receive at a time and not do both concurrently; (d) the communication and computational activities inside each node occur concurrently. As a consequence of assumptions (c) and (d), the calculation of a node's peak power consumption takes into consideration only the node's computational power consumption and its transmitter power consumption. It does not include the receiver power consumption since $P_{tx} = \max\{P_{tx},\ P_{rx}\}$. The total

WDC network power consumption is given by

$$P_{nw} = \sum_{i=1}^{N_{nodes}} P^i_{node},\ P^i_{node} \leq P^i_{supply}, \tag{3.66}$$

$$\text{where } P^i_{node} = \left\{ \begin{array}{lll} P_{cp}(f^i_{cpd}) + \max\limits_{\forall j} \left\{ P^{ij}_{tx} \right\} & ; & i = 1,\ j = 2 \ldots N_{nodes}, \\ P_{cp}(f^i_{cpd}) + P^{i1}_{tx} & ; & i = 2 \ldots N_{nodes} \end{array} \right\}. \tag{3.67}$$

Let T^i_{cpd} denote the time taken to execute a task in node N_i when it operates its processor at f^i_{cpd} Hz. The master node consumes energy in order to process N^{mr}_{cpw} tasks, and transmit (receive) data corresponding to N^{sl}_{cpw} tasks to (from) slave nodes. For every task assigned, the slave node consumes energy to process the task, receive N^{in}_{bits} bits of data from the master node as input to the task, and transmit N^{out}_{bits} bits of processed information, which are generated at the output of the task, to the master node. The computational workload assigned to every node in the network is determined based on the WDC workload allocation strategy employed by the master node. Then, the total WDC network energy consumption is given by

$$E_{nw} = \sum_{i=1}^{N_{nodes}} E^i_{node}, \tag{3.68}$$

$$\text{where } E^i_{node} = \left\{ \begin{array}{lll} P_{cp}(f^i_{cpd})\ T^i_{cpd}\ N^i_{cpw} & & \\ + \sum\limits_{j=2}^{N_{nodes}} N^j_{cpw} \times \left(E_{tx1} + \left(\overline{E}^{1j}_{txb} \times \frac{N^{in}_{bits}}{R_{code}} \right) \right) & & \\ + \sum\limits_{j=2}^{N_{nodes}} N^j_{cpw} \times \left(E_{rx1} + \left(\overline{E}^{j1}_{rxb} \times \frac{N^{out}_{bits}}{R_{code}} \right) \right) & ; & i = 1 \\ P_{cp}(f^i_{cpd})\ T^i_{cpd}\ N^i_{cpw} & & \\ + N^i_{cpw} \times \left(E_{tx1} + \left(\overline{E}^{i1}_{txb} \times \frac{N^{out}_{bits}}{R_{code}} \right) \right) & & \\ + N^i_{cpw} \times \left(E_{rx1} + \left(\overline{E}^{1i}_{rxb} \times \frac{N^{in}_{bits}}{R_{code}} \right) \right) & ; & i = 2 \ldots N_{nodes} \end{array} \right\} \tag{3.69}$$

$$\text{and } T^i_{cpd} = \frac{N_{cycles}}{f^i_{cpd}}. \tag{3.70}$$

3.7 WDC Power and Energy Savings Metrics

Under certain conditions, power and energy consumption savings can be achieved by offloading the computational tasks of an application from the master node to the slave nodes in the network. Let P^m_{sav}, P^{nw}_{sav} and E^m_{sav} denote, respectively, the master node's power savings, network-wide power savings, and the master node's energy savings achieved by executing an application in a distributed manner in a WCCN instead of processing all the computational tasks of the application in the master node itself. Let f_{cpl} denote the processor clock frequency employed by the master node when all tasks in the application are processed locally and T^{ij}_{cpl} denote the time taken to process task F_i in node N_j while using a processor clock rate of f_{cpl}. The master node's power savings, network-wide power savings, and the master node's energy savings are defined, respectively, as given by

$$
\begin{aligned}
P^m_{sav} &= \max_{\forall i} \left\{ P^{ij}_{cp}(f_{cpl}) \right\} - P^j_{node}, && (3.71)\\
P^{nw}_{sav} &= \max_{\forall i} \left\{ P^{ij}_{cp}(f_{cpl}) \right\} - P_{nw}, && (3.72)\\
\text{and } E^m_{sav} &= \left(\sum_{i=1}^{N_{tasks}} P^{ij}_{cp}(f_{cpl}) \times T^{ij}_{cpl} \right) - E^j_{node}, && (3.73)\\
\text{where } T^{ij}_{cpl} &= \frac{N^{ij}_{cycles}}{f_{cpl}}, && (3.74)\\
i &= 1,\ 2,\ \ldots, N_{tasks},\\
j &= 1 \text{ (master node's index)},\\
\text{and } 0 < \ f_{cpl} &\leq f_{cpmax}.
\end{aligned}
$$

P^j_{node}, P_{nw}, and E^j_{node} are computed, respectively, using Equations 3.66, 3.67, and 3.69. The first sum term in Equations 3.71 and 3.72 represents the peak power consumption in the master node when executing all tasks locally. The power (and energy) savings are expressed as a percentage of the local processing peak power (and energy) consumption.

Negative power savings is an indication that it is favorable to opt for local processing over

distributed processing. Negative power savings are incurred when the network is operated in the latency reduction mode, i.e., when $f_{cpl} = f_{cpd}$. In addition, the savings achieved by WDC load distribution is negated by the communication overhead associated with communication that occurs between nodes while executing the application in a distributed manner. When the communication and computational activities in the master node are run during non-overlapping time intervals, the amount of power savings achieved by the master node is given by

$$\begin{aligned} P_{sav}^{m} &= \max_{\forall i}\left\{P_{cp}^{ij}(f_{cpl})\right\} - \max\left\{\max_{\forall i}\left\{P_{cp}^{ij}(f_{cpd})\right\},\ \max_{\forall k}\left\{P_{tx}^{jk}\right\},\ \max_{\forall k}\left\{P_{rx}^{kj}\right\}\right\}, \quad (3.75) \\ \text{where } j &= 1, \\ i &= 1,\ 2,\ \ldots,\ N_{tasks}, \\ \text{and } k &= 1,\ 2,\ \ldots,\ N_{nodes},\ k \neq j. \end{aligned}$$

3.8 Simulation Parameters

Three processor power consumption models are used in this system model, namely, processor models 1, 2, and 3. Processor model 1 captures the characteristics of a StrongArm SA-1100 processor that executes a 1024-point FFT operation. The coefficients of model 1 are derived from experimental data [37–39] [1] and are listed as follows: $a_{cp} = 1.7745 \times 10^{-26} W/Hz^3$, $b_{cp} = 3.1472 \times 10^{-18} W/Hz^2$, $c_{cp} = 2.039 \times 10^{-10} W/Hz$, and $d_{cp} = 1.36\ mW$. The number of processor cycles consumed in this processor in order to process one 1024-point FFT is given by $N_{cycles} = 43.67 \times 10^6$. Processor models 2 and 3 are hypothetical and they differ from Processor model only in the value of coefficient b_{cp}. The coefficient b_{cp} is chosen such that the network yields positive network power savings for WDC workload allocation. The condition under which computational platforms can achieve positive power savings is derived next.

For analytical purposes, the cubic coefficient a_{cp} in Equation 3.2 is dropped from consid-

[1] http://www.lartmaker.nl/278088.pdf

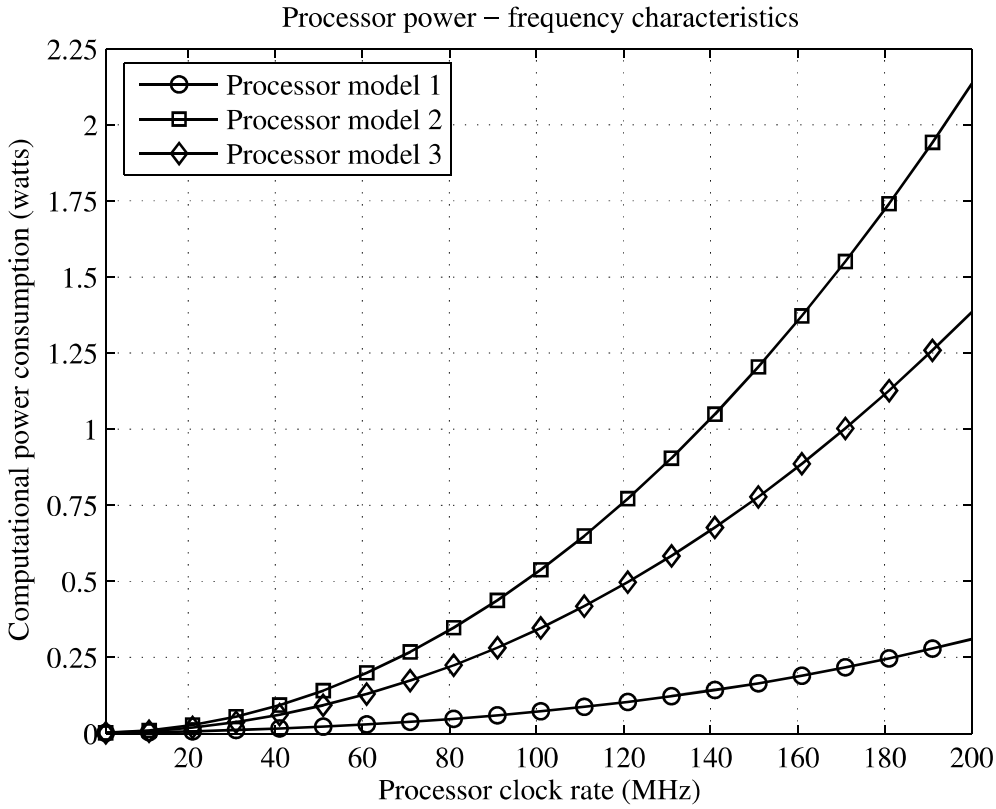


Figure 3.8: The power consumption - clock rate characteristics for processor models 1, 2, and 3.

eration due to its relatively insignificant value. In order to achieve positive network power savings, the expression on the right of Equation 3.72 is evaluated as given by

$$b_{cp}\ \left(f_{cpl}^2 - N_{nodes}\ f_{cpd}^2\right) + c_{cp}\ \left(f_{cpl} - N_{nodes}\ f_{cpd}\right) > N_{nodes}\ P_{tx}, \tag{3.76}$$

where the following assumptions are made: a. all tasks are homogeneous; b. the computational subsystems on all nodes are identical; c. all nodes operate their processors at the same clock rate; and d. the worst case transmission power consumption is taken into consideration where all slave nodes are located at the edge of the circular coverage area and the master node is located at the center. When operating in the power reduction mode, where $f_{cpd} = f_{cpl}/N_{nodes}$, the condition for positive network savings can be derived from

Equation 3.76 as given by

$$b_{cp} \quad > \quad \frac{N_{nodes}^2 \, P_{tx}}{(N_{nodes} - 1) \; f_{cpl}^{\,2}}. \tag{3.77}$$

The b_{cp} coefficient of processor model 2 is computed using Equation 3.77 with $f_{cpl} = 206$ MHz, $N_{nodes} = 5$, and $P_{tx} = 0.331$ watts and is given by $b_{cp} = 4.8741 \times 10^{-17} W/Hz^2$. The b_{cp} coefficient for processor model 3 is determined as an arbitrary value that lies between the b_{cp} coefficient values of processor models 1 and 2, and is given by $b_{cp} = 3 \times 10^{-17} W/Hz^2$. Overall, it is intended to employ processor models with varying degrees of non-linearity in their power-clock rate characteristics. The computational power consumption - clock frequency characteristics of the three processor models are plotted in Figure 3.8. The difference in the degree of non-linearity in the processor characteristics between the processor models is clearly evident from Figure 3.8. Practically speaking, computational subsystem power consumption characteristics have been found to exhibit non-linearity, particularly when the processor cooling fan gets activated at threshold temperatures resulting in a sharp increase in power consumption.

Table 3.1 presents the simulation values that are employed in all simulation analysis presented in this dissertation. In some cases, however, the values may be overridden by assumptions made individually for each simulation result. f_{cp} is set to 206 MHz for the receiver's signal processor. The empirical power consumption values of the processor and RF components have been cited from several references [37–41]. The close-in reference distance is set to 10 meters and is understood to be reasonable based on a survey of link budget analysis performed for short range networks, an example link budget analysis performed for IEEE WLANs is presented in reference [42]. The RF power consumption models and simulation values employed for simulation analysis in this dissertation hold good for narrow band systems and it is beyond the scope of this dissertation to comment on how the models would change for wide band systems. In all simulation analysis presented in this Chapter and Chapter 5, the Monte-Carlo approach has been employed to simulate the location of all

slave nodes as a binomial point process in a circular coverage area with the master node being located at the center of the circle. The inter-node distances are, then, computed from the simulated node locations. The size of the data file that is passed between tasks which interact with each other is set to 20 Kbits.

Table 3.1: System parameters used in this dissertation.

Parameter	Value	Parameter	Value
β_{amp}	174 mW	η_{amp}	5
η_{pl}	4	λ_{sig}	0.667 m
σ_s	8 dB	σ_g	1
BW	30 KHz	C_{prstc}	1 pF
C_{switch}	0.626 pF	D_{ant}	0.5 meters
D_{far}	0.75 meters	D_{ref}	10 meters
E_{bdec}	5.9416 μJ	E_{rx1}	27.6 μJ
E_{rx2}	0.1074 W	E_{tx1}	27.6 μJ
f_{corner}	1 MHz	f^i_{cpl}	206 MHz
f_{cpmax}	206 MHz	f_{sig}	450 MHz
G_1	-126.72 $dBm/m^4/Hz$	G_2	0.3309 W
G_r	2 dBi	G_t	2 dBi
I_{LSB}	10 μA	k_B	1.3806×10^{-23} J/K
K_{cl}	4	K_m	2 (BPSK)
L_{chlen}	0.5 μ meters	LM	10 dB
n_{adc}	10 bits	n_{dac}	10 bits
N_{psd}	-163.83 dBm/Hz	NF	10 dB
P_{ADC}	4.6 mW	$P_{coverage}$	99 %
P_{DAC}	15.4 mW	P^r_{filter}	2.5 mW
P^t_{filter}	2.5 mW	P_{LNA}	20 mW

Continued on next page

Table 3.1 – continued from previous page

Parameter	Value	Parameter	Value
P_{LO}	50 mW	P_{mixer}	30.3 mW
P_{noise}	-119 dBm	P_{outage}^{cp}	0.1
P_{static}	0.25 mW	P_{trs}	58.7 mW
R_{baud}	32 symbols per sec	R_{code}	2/3
R_{txout}	32 $kbps$	SNR_{min}	15 dB
T_{idle}	0	T_{noise}	300 K
T_{sprb}	17.5 μsec	T_{trs}	470 μ sec
V_{supply}	3 V		

3.9 Example WCCN Scenario: Collaborative UAVs in a Tactical Environment

This section describes the application of WDC to a tactical scenario involving a network of unmanned aerial vehicles (UAVs). The example scenario demonstrates how the notion of WCCNs can empower a network of resource-constraint nodes and minimize its dependency on remote super nodes for mission-critical complex computational processing. This example scenario can be extended to wireless sensor networks, networks comprising smart phones in a social setting, and collaborative networks comprising tactical hand-held radios. The section also presents a simulation analysis of the energy benefits which are achieved by WDC workload distribution is this scenario. The scenario is illustrated in Figure 3.9. The master node in the UAV network is tasked with capturing images (or any other data) and conveying information gathered from the images to a remote base. The captured data can be processed in one of the following ways: (a) by employing remote data processing, where unprocessed data is transmitted over a long range backhaul to a node abundant in computing resources located in the remote base and the data is processed by the remote node; or (b)

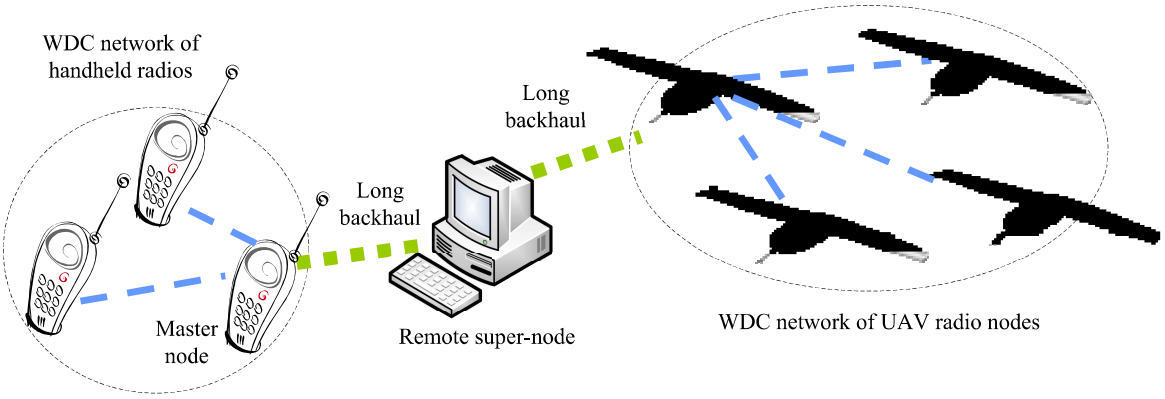


Figure 3.9: An example tactical communication scenario involving UAV nodes and a remote super node.

by employing in-network data processing, where data is first processed within the UAV network before transmitting processed information to the remote base. The latter option, when performed in a distributed manner, also allows the use of a large number of relatively small, simple, and inexpensive radio nodes in place of a single complex and expensive one. A large number of UAV nodes form a WCCN, where the master UAV node broadcasts data to the slave UAV nodes which return processed data to the master node. The master node, then, aggregates all the processed data and transmits it to the remote base. Processed data accounts for relatively smaller communication payload compared to raw data. The savings in energy consumption of the UAV nodes is achieved in several ways. First, WDC enables in-network processing, which significantly reduces the number of data bits transmitted over the long range backhaul at the cost of some computational energy consumption. Second, power-hungry processor cooling mechanisms activate when the processor operates at a certain high clock frequency. Using workload distribution, the required computational latency can be met by concurrently processing data in all the UAV nodes at a fraction of the required rate.

Energy calculations and assumptions Simulation analysis is performed using the computational and communication models presented earlier in this chapter to determine the UAV network's energy savings and improvement in battery lifetime achieved by distributed

Table 3.2: Parameters used in the simulation of a tactical UAV WCCN scenario.

Parameter	Value	Parameter	Value
R_c	50 meters	η_{pl}	3
N_{nodes}	5	N_{tasks}	3002
N_{pts}	1024	N_{bpp}	10 bits
f_{cpl}	200 MHz	R_{comp}	0.1
N_{cycles}	43.67 $\times 10^6$ cycles	E_{supply}	10 kJ

in-network processing in comparison to remote processing. The lifetime of the UAV network, which is essentially the lifetime of the master node's battery, achieved with the remote processing and in-network processing options is denoted, respectively, by T^r_{life} and T^w_{life}. Simulation parameters which are presented in Table 3.2 are used for the simulation in addition to the parameters presented in Table 3.1. The user application processes images captured by the master UAV node and outputs information gathered from the processed images, such as the presence of objects of interest. Processed data size sent to the remote node is assumed to constitute 10 % of the raw data size. The application comprises a source task, one sink task, and $(N_{tasks} - 2)$ homogeneous tasks, each representing a single 1024-point FFT operation. It is structured based on task graph model 1. It is assumed that $N^{in}_{bits} = N^{out}_{bits} = N_{bits}$ for all FFT tasks, where N_{bits} denotes the number of bits that appear either at the input or output of a FFT task and is given by

$$N_{bits} = N_{pts} \times N_{bpp}. \tag{3.78}$$

In Equation 3.78, N_{pts} denotes the number of data points per FFT and N_{bpp} denotes the number of bits used to represent a single FFT point. The source and sink tasks are always executed by the master node and are assumed to impose negligible costs on the master node. The network comprises homogeneous nodes, i.e., every FFT task consumes the same number of processor cycles in all nodes in the UAV network and all nodes have the same remaining energy supply. Uniform workload allocation is employed in the UAV network and the network operates in the power reduction mode, where the net required processor clock

rate is 200 MHz. In simulations where the network size is fixed, the UAV network size is set to 5 nodes. The wireless channel is characterized by log-normal shadow fading, path loss, and noise.

The total energy consumption when adopting remote data processing and WDC processing is denoted, respectively, by E_{remote} and E_{wdc}. E_{remote} comprises of only one energy consumption component, namely, the energy consumed by the master node to transmit unprocessed data to the remote node. The mean transmitter energy consumption in this case is computed using Equation 3.43, where the mean transmitter power consumption is computed using Equation 3.6 by taking into consideration the communication range of the long backhaul between the master node and the remote node; the probability of success is given by $P_{coverage}$; and the mean latency per channel bit is computed using Equation 3.29. Let N_k represent the remote node, where k denotes its node index. The total mean transmission time for the remote processing option, denoted by $\overline{T}_{tx}^{1k}$, is computed using Equation 3.34, where the total number of information bits transmitted to the remote base is given by

$$N_{data} = (N_{tasks} - 2) \times N_{bits}. \tag{3.79}$$

The lifetime of the master node is computed by determining the time for which the master node's battery life lasts when the master node pursues the remote processing option several times until its battery is completely discharged. The network's lifetime achieved when the UAV network chooses to pursue the remote processing option is given by

$$T_{life}^{r} = \left(\frac{E_{supply}}{E_{remote}}\right) \times \overline{T}_{tx}^{1k}. \tag{3.80}$$

When the UAV network pursues the in-network processing option, the network consumes energy in order to execute the application using WDC workload distribution and consumes energy to transmit processed data to the remote node. The latter energy component is denoted by $E_{txRemote}$. Then, the UAV network's total energy consumption when adopting

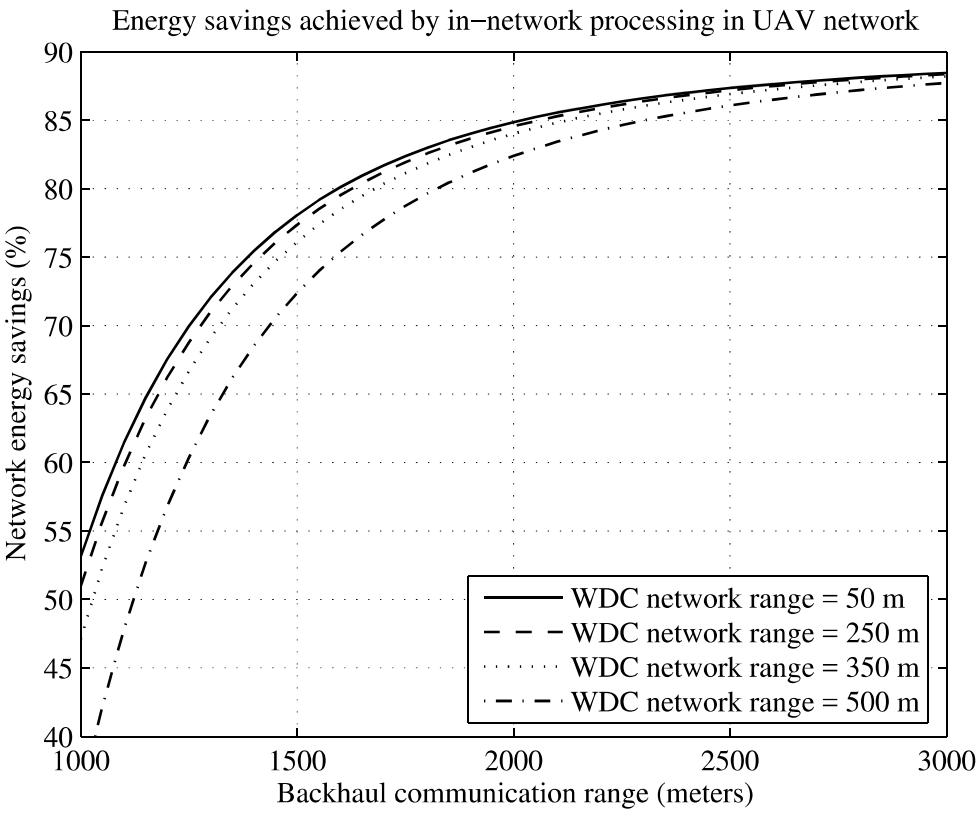


Figure 3.10: Energy savings achieved by the example UAV network, via in-network processing and workload distribution, for different backhaul communication ranges and WDC network ranges.

in-network processing is computed as given by

$$E_{wdc} = E_{nw} + E_{txRemote}. \tag{3.81}$$

E_{nw} is computed using Equation 3.68. In Equation 3.68, the mean transmitter energy consumption per channel bit is computed by taking into consideration the UAV network range, which is given by R_c. The mean receiver energy consumption per channel bit is computed using Equation 3.44. The computational power consumption is computed using Equation 3.2, where the processor model coefficients are set with the values presented in Section 3.8 with the assumption that the StrongArm SA-1100 processor is used. The computational execution time is computed using Equation 3.70, where the processor clock frequency is set as given

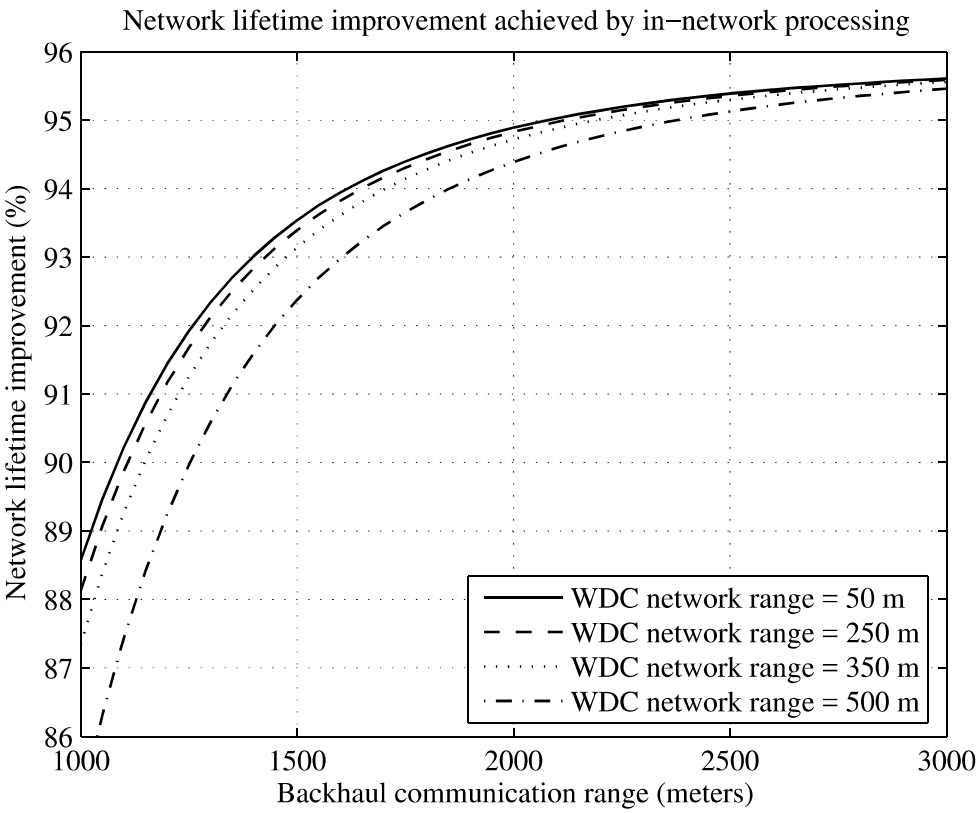


Figure 3.11: Lifetime improvement achieved by the example UAV network, via in-network processing and workload distribution, for different backhaul communication ranges and WDC network ranges.

by $f_{cpd} = \lceil f_{cpl}/N_{nodes} \rceil$. $E_{txRemote}$ is computed by following the steps to compute E_{remote}, with the exception that the total number of information bits transmitted to the remote base is given by

$$N_{data} = R_{comp} \times (N_{tasks} - 2) \times N_{bits}. \tag{3.82}$$

The network's lifetime (or the master node's lifetime) achieved when the UAV network

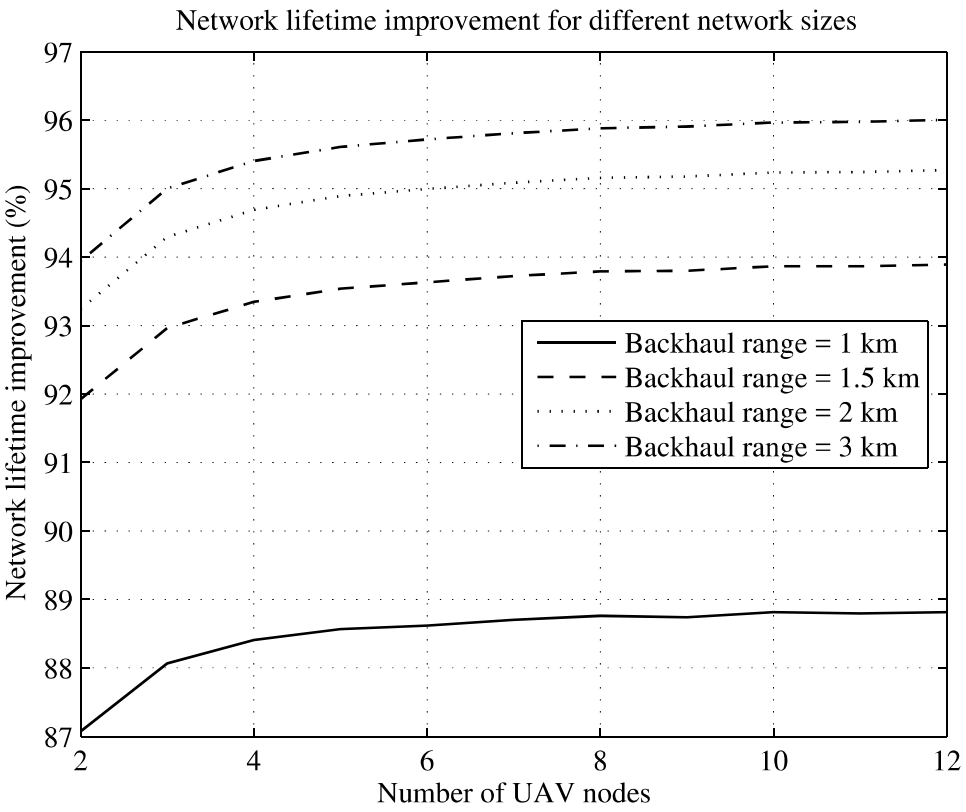


Figure 3.12: Percentage lifetime improvement achieved by the example UAV network, via in-network processing and workload distribution, for different UAV network sizes and backhaul communication ranges.

chooses to pursue the in-network processing option is given by

$$
\begin{aligned}
T^{w}_{life} \;=\; & \left(\frac{E_{supply}}{(E^{1}_{node} + E_{txRemote})} \right) \\
& \times \left\{ \left(T^{1}_{cpd} \times N^{1}_{cpw} \right) + \left(\sum_{j=2}^{N_{nodes}} T_{trs} + \left(\overline{T}^{1j}_{bit} \times \frac{N^{in}_{bits}}{R_{code}} \times N^{j}_{cpw} \right) \right) + \right. \\
& \left(\sum_{j=2}^{N_{nodes}} T_{trs} + \left(\overline{T}^{j1}_{bit} \times \frac{N^{in}_{bits}}{R_{code}} \times N^{j}_{cpw} \right) \right) + \left(\sum_{j=2}^{N_{nodes}} \overline{X}^{j1}_{pktx} \times T_{sprb} \times N_{bits} \times N^{j}_{cpw} \right) \\
& \left. + \left(T_{trs} + \overline{T}^{1k}_{bit} \times \frac{N_{data}}{R_{code}} \right) \right\}. \qquad (3.83)
\end{aligned}
$$

The first product term, which is shown in parenthesis in Equation 3.83, represents the number

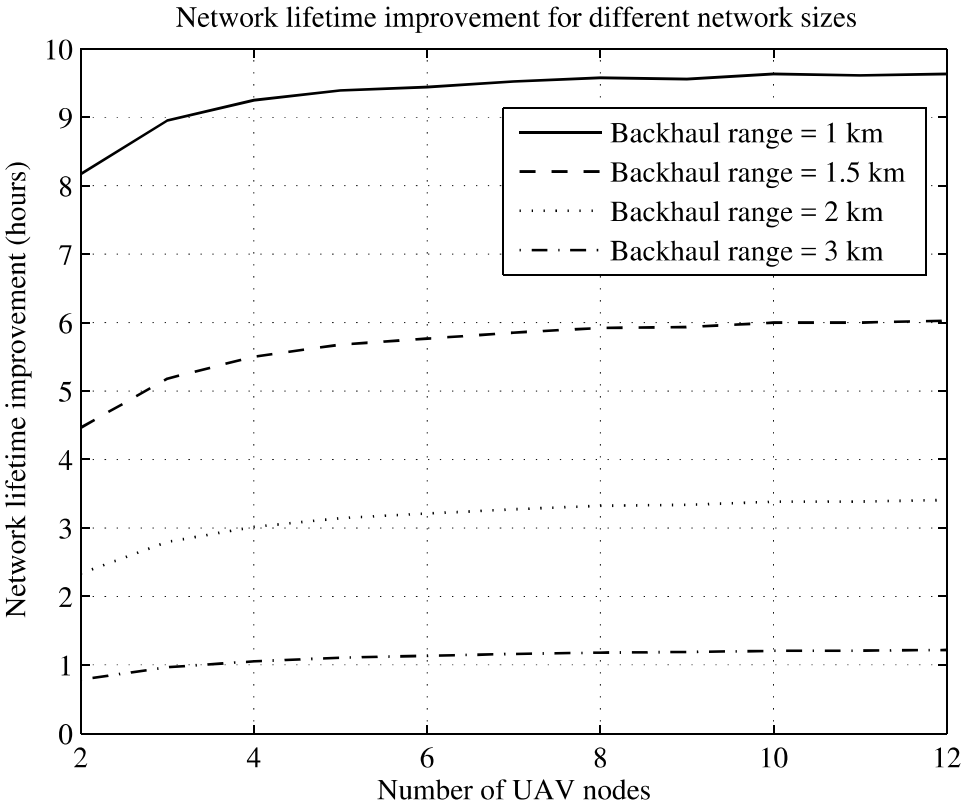


Figure 3.13: Lifetime improvement (expressed in hours) achieved by the example UAV network, via in-network processing and workload distribution, for different UAV network sizes and backhaul communication ranges.

of times the network can execute the same application repetitively using the in-network processing option, where the process is repeated until the master node's battery discharges completely. The second product term, which is shown within curly parenthesis, represents the total time taken to execute the user application using the in-network processing option, where it is assumed that the master node's computational and communication activities occur over non-overlapping time intervals. Note that, the assumption is made only for the simulation analysis in this section in order to simplify scheduling constraints. The terms represent, respectively, the time taken by the master node to process its share of the workload, total time taken by the master node to disseminate unprocessed data to the slave nodes, total time taken by the master node to gather processed data from the slave nodes, total time

taken by the master node to decode channel bits received from the slave nodes, and total time taken by the master node to transmit results to the remote node.

The percentage energy savings and the percentage improvement in lifetime achieved by the UAV network are defined, respectively, as given by

$$E_{sav}^{uav} = \frac{E_{remote} - E_{wdc}}{E_{remote}} \times 100 \tag{3.84}$$

$$\text{and } T_{lifesv} = \frac{T_{life}^{w} - T_{life}^{r}}{T_{life}^{w}} \times 100. \tag{3.85}$$

3.9.1 Simulation Results

The UAV network's energy savings achieved by distributed in-network processing is plotted in Figure 3.10 for different backhaul and WCCN ranges. The corresponding improvement in the network's lifetime is plotted in Figure 3.11. The savings increase with an increase in the backhaul range. However, the savings curves flatten when the transmitter power consumption incurred during communication with the remote node begins to dominates the savings. It is shown that energy savings of nearly 88% can be achieved for a backhaul range of 2 km and a WDC network range that falls in the range of 50-350 m. The savings varies with the UAV network range, particularly at short backhaul ranges, because of the communication overhead involved in distributing the workload among the UAV nodes and gathering processing results at the master node.

Figures 3.12 and 3.13 show the improvement in the network's lifetime for a UAV network that is located within a range of 50 meters. It is shown that in-network processing using WDC provides a lifetime improvement of 87% (or 8 hours) with a UAV network as small as two nodes when the backhaul range is 1 km. Greater benefits are achieved with long range backhauls, which is evident by comparing the percentage network lifetime improvement for backhaul ranges of 1 km and 3 kms. As the network size increases and more tasks are delegated to slave nodes under the uniform workload allocation scheme, the master node's

computational energy consumption decreases, both due to a reduction in the master node's share of the computational workload and a reduction in the processor clock frequency per node. However, the reduction in computational energy consumption is accompanied with an increase in the master node's communication energy consumption, since the master node has to broadcast (gather) more data to (from) the slave nodes. Thus, the network's lifetime improves with network size, and there is no significant improvement in network lifetime beyond a certain network size of about 8 nodes.

3.10 Chapter Summary

This chapter presents a system model that captures the energy consumption, power consumption, and latency incurred by the computational and communication activities involved in the execution of a user application's task graph. This chapter also defines metrics to measure, respectively, the power and energy efficiency achieved by WDC workload allocation. The system model is utilized in the analysis of power and energy efficiency achieved by WDC workload distribution in Chapter 5. In addition, it is used in the performance analysis of the proposed task allocation - scheduling algorithm, particularly the improvement in the application's makespan achieved by workload distribution. In this manner, the system model helps in the development of resource management and decision-making mechanisms in the proposed WCCN architecture. An example WDC workload distribution scenario is analyzed using the system model in this chapter.

While this dissertation employs fundamental models for the analysis, there is tremendous scope in extending the system model to incorporate sophisticated wireless systems. The adaptive transmission model, for instance, presented in this chapter is greatly simplified. In the case of adaptive modulation schemes, T_{bitp} can be modeled as a random variable which varies according to the modulation scheme and channel outage probability. However, T_{bitp} is assumed constant in the system model which is a reasonable assumption considering the fact

that only BPSK modulation is employed with no rate control in the system under consideration. In addition, the power consumption, latency and energy consumption models can be extended to take into consideration various factors such as channel estimation accuracy, delays incurred by channel estimation mechanisms, and advanced transmission schemes such as MIMO-OFDM.

The system model in this dissertation considers Rayleigh fading. While Rayleigh fading is appropriate for crowded urban environments, Rician fading may be used in scenarios where there is a dominant line-of-sight component which may be common in open areas. However, Rayleigh fading is preferred for modeling purposes because of its analytical simplicity thereby enabling one to focus on the WDC aspects of the model. The performance of Rician fading channels over Rayleigh fading channels depends on the Rice factor. A strong value of the Rice factor can improve the performance in Rician fading channels over Rayleigh fading channels thereby giving better WDC workload allocation energy savings. This behavior is further understood by considering the approximate relationship between Rician fading channel power gain and Nakagami fading channel power gain, where it is observed that a high value of the Rice factor increases the value of the Nakagami factor m which, in turn, reduces the fading in signal strength [43] and reduces the communication cost in WDC.

Chapter 4

An Architecture And Methodology for WCCNs

There arises a strong need for a formal architectural framework to conceptualize, model, implement, test and verify the behavior of complex distributed systems such as a WCCN. A framework provides an overview of both the logical organization as well as the actual implementation of the software and hardware components that comprise a distributed system. This chapter presents the proposed architectural framework for WCCNs and provides details of the high level functionality of the components in the architecture as well as a fundamental use-case for the proposed framework.

The proposed WCCN framework is organized into layers where each layer further comprises components and services. The layers interact with each other via clearly defined interfaces. As per the component-based approach, the logical software components are modeled in terms of their interfaces (including externally configurable parameters), functionality and attributes. Interactions between components occurs via the component interfaces and can be specified in terms of functional parameters or messages. In this manner, the architecture formalizes the interaction between the various components as well as the functional contribution of each component to the overall operation of the distributed system. The framework

provides application programming interfaces (APIs) that can be used by end users to port user customized applications which can utilize services that are hosted in a WCCN environment. The framework is capable of operating in different scenarios and topologies, including single-hop/multi-hop and static/dynamic environments. It can be employed for both load distribution and distributed application scenarios. In this dissertation, however, the discussion of the architecture design is limited to load distribution scenarios.

Along with a description of the architecture, this chapter also describes the prototype implementation of the proposed architecture. The framework has been prototyped on a software defined radio platform for a simple single-hop WDC load distribution scenario involving a single master node and several slave nodes. Each node comprises a Universal Software Radio Peripheral ($http://www.ni.com/usrp/$) device connected to a laptop that executes the software. The software design follows an object-oriented component-based approach. The prototype serves as a validation of the WDC aspects of the proposed WCCN framework. Although the service management mechanisms of the architecture are briefly described in this chapter, the implementation and demonstration of the service management aspects of the architecture are not within the scope of this dissertation. Adaptive inter-layer resource allocation, which is an important example of the cross-layer approach presented in section 4.3, is the main focus of the architecture prototype implementation.

The rest of the chapter is organized as follows. Section 4.1 discusses the various driving factors for the architecture design, namely, the core system requirements and challenges that have to be met by the architecture design. The proposed architecture is presented with an overview in Section 4.2. Section 4.3 describes one of the main novelties of the proposed framework, namely, the multi-layer decision making, control and environment awareness. Next, the individual layers in the proposed architecture are described in order starting from the lowest layer and going upwards. Sections 4.4, 4.5, 4.6, and 4.7 describe, respectively, the Communication Layer and its interface; WDC Layer and its interface; Service Management Layer; and WCCN Application Layer. Section 4.8 presents a simple use case of the proposed architecture that is used in implementing the prototype. The novelties in the proposed

architecture design are discussed along with comparison with existing architectural designs in Section 4.9. The chapter ends with a summary as discussed in Section 4.10.

4.1 System Requirements for Architecture Design

The WCCN framework is expected to handle the challenges imposed by dynamic WCCN environments as discussed in Chapter 2. *The challenge is to optimally meet dynamic user application requirements with the available resources and services while operating in a diverse (or heterogeneous) wireless cloud computing environment characterized by dynamic variations and uncertainty.* For instance, the availability of services varies dynamically either due to the mobility of service providers or the deployment and termination of services by mobile users. Based on the vision presented in Section 2.2, the core system requirements are highlighted as follows.

Provide mechanisms for dynamically hosting and searching services The architecture should provide robust mechanisms that can be used by service requesters to dynamically search for available services, and mechanisms that can be used by service providers to dynamically host and publish services.

Enable the execution of user applications by dynamically integrating and provisioning available services (or resources) User applications may be dynamically composed of services which are hosted by more than one service provider. Due to the dynamic nature of the service availability, there is a need for robust mechanisms that identify, provision, and integrate the required services in order to execute the user application.

Enable robust inter-service interaction over uncertain wireless links In order to enable distributed execution of user applications in terms of services hosted in the WCCN environment, the architecture should support the interaction between services that may be hosted on different mobile nodes. The interaction between services will have to be transparent to the mode of communication and the physical proximity of the services. For instance,

services hosted by the same node may interact with each other via shared memory while services located in different nodes communicate over uncertain wireless links using wireless protocols.

Meet dynamic user application requirements and objectives in an uncertain environment QoS requirements and operational objectives specified by user applications can vary from one service request to another. Meeting the custom application requirements is critical, for instance, in the case of applications that are time-sensitive and expect high reliability. The system should also be able to optimize WCCN performance based on the specified objectives. In addition to meeting the requirements, a WCCN is expected to handle the dynamics and uncertainty in the operating environment and provide a computing environment that is reliable and stable from the perspective of the user application.

Handle multiple access to resources and services The architecture design should be able to handle multiple competing service requests and coordinate multiple access to cloud resources.

4.2 System Level WCCN Framework

This section presents an overview of the proposed WCCN architectural framework. The framework is organized in terms of horizontal and vertical layers, as shown in Figure 4.1. The vertical layers operate across horizontal layers and facilitate inter-layer interaction between the horizontal layers. The layered approach is intended to delineate each layer while allowing interaction and sharing of environment awareness between layers. Each layer provides an interface which defines APIs that can be used by any other layer to interact with it. The interfaces provided by the horizontal layers, for instance, are represented by colored double headed arrows in Figure 4.1. The layered approach allows system developers to switch between different implementations of the individual layers and components without significantly impacting the rest of the system. The APIs enable any user application to

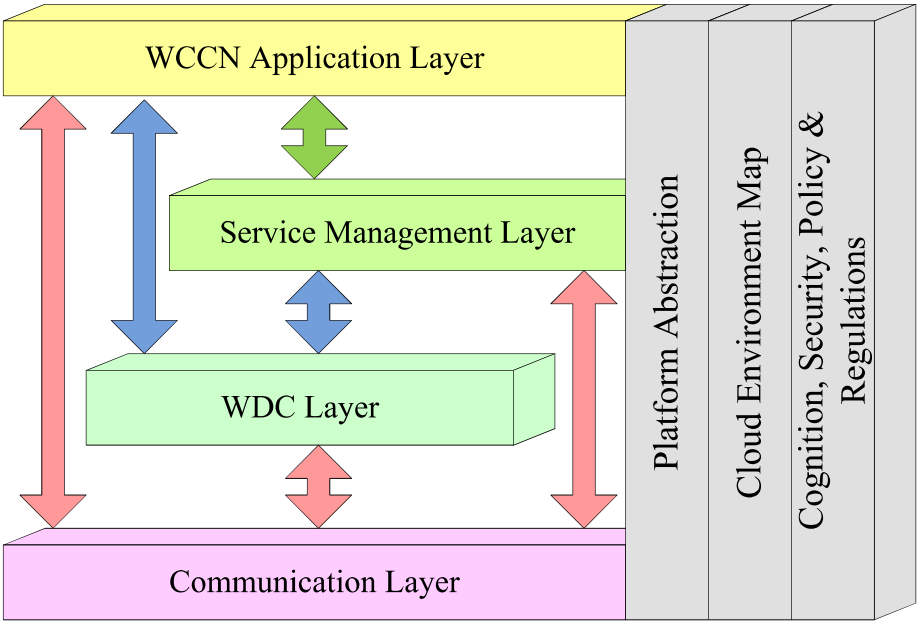


Figure 4.1: Horizontal and vertical layers in the WCCN framework.

harness the various functionality and information provided by the framework without being bogged down by the complexities of the underlying mechanisms. In addition, the framework can operate as an overlay network on top of an arbitrary wireless network as long as the interface specifications set by the framework are met.

The architecture comprises the following horizontal layers: (a) WCCN Application and User Interface Layer (or WCCN Application Layer in short), (b) Service Management Layer, (c) Wireless Distributed Computing Decision Making and Resource Management Layer (or WDC Layer in short), and (d) Communication Layer. Data messages, control messages, and state information messages are the three main types of messages that are passed between any components in the WCCN framework regardless of the physical location of the components. Data messages carry processed or unprocessed data between the tasks of a user application; control messages can be passed to a component in order to invoke its functionality, configure its parameters, or interrupt its operation; and state information messages convey environment state information that includes the state of the communication network, status of a distributed computing process, and the availability of computing resources.

Horizontal layers in the WCCN framework The WCCN Application Layer performs application-dependent function required for cloud computing in a wireless environment while the other horizontal layers performs application-independent functions for the same. For a given application service request, the Application Layer gathers the user requirements and displays results to the user. The Service Management Layer handles the service request in terms of services which, in turn, may involve WDC activities. The WDC Layer handles the request in terms of computational tasks represented in the form of a task graph. The horizontal layers are discussed in detail after this section.

Vertical layers in the WCCN framework The Platform Abstraction Layer abstracts the underlying hardware and software execution platform from the rest of the framework. It maintains abstractions of hardware pertaining to the three subsystems of a WCCN node, namely, the power, communication and computational subsystems. The underlying platform includes the operating system, hardware device drivers, computing hardware, radio hardware, sensors, and power supply hardware. This layer provides interfaces for accessing, configuring, and controlling the underlying platform. Resource access control is particularly important for embedded systems with limited computing and energy resources. All cross-layer security mechanisms are located in the Security Layer. The Cognition Layer handles the cognitive decision making and adaptation mechanisms that operate in conjunction with the decision making mechanisms of the horizontal layers. A case-based reasoning cognitive engine, for instance, makes decisions based on past experiences in processing application requests. The Cloud Environment Map stores and organizes all information pertaining to the WCCN environment gathered by each of the horizontal layers. The information may indexed in various dimensions including time, GPS coordinates, and cloud entity ID. The Cloud Environment Map also houses the knowledge-base which comprises past knowledge that is acquired by the cognitive engine. It includes records of past reputation of service provider nodes; and computational and communication latencies encountered while executing distributed applications in the past. The Policy and Regulation Layer imposes regulatory

and economic policies in different aspects of the framework. Policies are required for various purposes including billing for services consumed by service requesters; governing the access to resources; specifying quality of service requirements; regulating interaction between cloud entities and services; and resolving contradictory policies [17].

The framework can be employed for different WCCN topologies. The core functionalities and interfaces of components in the framework remain the same for different topologies, while the implementation of the components can differ. WCCNs can be configured in different topologies, namely, centralized, ad-hoc, and hybrid topologies. In a centralized topology, a leader node handles the complex functionalities involved in the distributed execution of user applications in the cloud, which include monitoring of the operating environment; resource allocation and other core decision-making processes that ensure successful execution of applications; and coordination of access to resources and interaction between services. Challenges imposed by the dynamic operating environment are handled by the leader node, thereby greatly simplifying the implementation of the framework components in service requesters. Service requesters may simply focus on application specific functionalities and be concerned only with their interaction with the leader node. The leader designation may be held by a dedicated node or it may dynamically move from one node to another based on changing network conditions. The leader role may also be shared between two or more nodes in the network. Large networks can be segmented into clusters, where each cluster has a cluster leader node (or a broker node). A hybrid topology is one where certain operations are centralized while other operations are managed by individual nodes autonomously. In order to execute an application in a hybrid topology, a service requester can choose between accessing a broker node and accessing service providers directly. The rest of the chapter is focused on the horizontal layers and their implementation for a master-slave configuration.

Table 4.1: Metrics and decision-making in the WCCN framework.

Layer	Inputs	Decision-making & control
Application Layer	WDC performance metrics, Service Layer metrics	Deciding on local versus distributed , allowing network access to an application.
Service Management Layer	Service state metrics	Choosing the right services to compose the application
WDC Layer	WDC state metrics, Communication Layer state metrics, Application Layer performance metrics	Task allocation and scheduling of communication and computational activities
Communication Layer	Communication state metrics	Waveform, transmit power, route

4.3 Cross-Layer Decision-Making, Control and Environment Awareness

In order to meet the unique challenges imposed by WCCN environments from a wireless systems perspective, the architecture provides the following: mechanisms to monitor the state of the cloud environment and maintain real-time awareness of the environment; decision-making mechanisms that determine optimal strategies for the execution of user applications based on the observed state of the environment; and interfaces that allow the architecture to exercise control over the system in order to implement the strategies. Table 4.1 summarizes the decision-making mechanisms in the architecture and the factors that influence them. Decision-making in the architecture includes choosing the correct services to compose the application; choosing service providers that can offer the required services and be able to execute the user application such that the application requirements are met; scheduling the computational and communication activities involved in the distributed execution of user applications; and choosing between local and distributed processing. The metrics used for decision-making (as shown in Table 4.1) are discussed in the later part of this section.

The mechanisms for decision-making, monitoring and gathering state information, and control are distributed among the horizontal layers of the framework such that different aspects of the cloud environment are controlled and monitored by different horizontal layers in the framework. This is done so for a couple of reasons. First, each layer operates at different time scales. The dynamics in the wireless network, for instance, can occur over smaller time scales compared to the dynamics in service availability. Accordingly, the wireless network state monitoring component in the Communication Layer operates at a smaller time scale compared to the task allocation - scheduling component in the WDC Layer. Second, different layers operate on different levels of abstraction of the cloud environment. Due to the multi-dimensional nature of the wireless cloud environment, the environment state information may be represented in terms of several metrics that can be organized into different levels of abstraction. For instance, low level state information such as that of the underlying wireless operating environment is of importance to the Communication Layer and may not be of direct concern to the layers above. The Communication Layer monitors the state of the wireless network and handles decision-making to control communication parameters such as modulation-coding and operating frequency. The WDC Layer is concerned with task allocation - scheduling while it is not directly concerned with the control of low level communication parameters. The Application Layer acts on high level information provided by the WDC Layer in the form of performance metrics and does not need to concern itself with the state of the underlying wireless network.

Although the mechanisms are distributed among the layers, the architecture allows the respective mechanisms at different layers to interact with each other in order to exchange state information and coordinate their operation. First, a primitive functionality of gathering awareness information is environment probing, specifically, probing network entities for their state information. The probing functionality may be similar across the layers, differing only in the timing of the probing process, the probing protocol, and the probed state variable. Thus, the environment awareness components in the various layers, such as network discovery and service discovery, can cooperate with each other and schedule their respective probing

activities onto a common probing process. Second, due to the dependency between metrics of different layers, the state information gathered by lower layers is made available to the higher layers thereby enabling them to make high level decisions. For instance, the decision on choosing a remote service is dependent on both its availability (Service Management Layer metric) and the reliability of the channel (Communication Layer metric) to access the service provider that hosts it. Adaptive inter-layer resource allocation is an important part of cross-layer decision-making that involves the WDC Layer and the Communication Layer and is the main focus of the architecture prototype implementation. The impact of wireless network conditions on the scalability and performance of WDC workload distribution necessitates an adaptive inter-layer approach where WDC resource allocation takes into consideration the randomly varying costs imposed by the underlying radio environment and the availability of resources in nodes. In addition to the inter-layer approach, WDC resource allocation is expected to operate with joint consideration of computational, communication and energy factors.

The decision-making capability of the Application Layer provides WCCN applications with greater transparency and control on the underlying system compared to traditional user applications. In addition, it allows user involvement and intervention in the cross-layer decision-making process. The decisions made by the Application Layer include: (a) choosing between local on-board processing and distributed processing of a user application based on the recommendations provided by the WDC Layer via WDC performance metrics, (b) making decisions on allowing network access to a user application and other interactive mechanisms, such as network probing, based on user security preferences and situational awareness; and (d) configuring the application objectives, including assigning weights to multiple objectives based on situational awareness and user preferences. The Application Layer can also get involved in making decisions on task migration.

The various metrics and data structures that are used in the decision-making process are described next.

Table 4.2: Example state metrics employed in the WCCN framework.

Layer	State Metrics
Comm. Layer	Link characteristics such as SNR, path loss, link reliability, outage probability, channel variance, re-transmission probabilities, communication latency, and link goodput; inter-node distances; communication capabilities of a node including supported waveforms, data rates, and transmit power levels; power consumption characteristics of communication hardware; communication resource availability expressed in terms of outage probability, communication buffer size and wait time characteristics, and spectrum occupancy; number of nodes in the network; and energy resource availability expressed in terms of remaining energy supply, outage probability, and battery characteristics
WDC Layer	Computational capabilities of a node including memory capacity; node reliability; processor clock frequency-power consumption characteristics; computing resource availability expressed in terms of processor queue size and waiting time characteristics; processor execution time characteristics.
Service Management Layer	Services offered in the network and service options; availability of services expressed in terms of wait time characteristics; billing characteristics of services; and status of service executions.

4.3.1 Metrics and Data Structures

Broadly speaking, two categories of metrics are used in the architecture, namely, state metrics and performance metrics. Metrics that represent random parameters are defined in terms of their respective statistical parameters, such as mean, standard deviation, and range of values that the metric is expected to take with a certain confidence.

State metrics are used to quantify the state of the wireless cloud environment, as shown in Table 4.2. State metrics may be either fixed or variable. State variables are monitored based on their frequency of variation. Examples of fixed metrics are computing and communication hardware capabilities and characteristics. Variable metrics include the current processor load and remaining energy supply. State metrics may either be basic or derived. Basic metrics are directly measured from the environment while derived metrics are computed as functions of one or more basic metrics. Link goodput can be either be a state metric or a performance

Table 4.3: Example performance metrics employed in the WCCN framework.

Layer	State Metrics
Comm. Layer	Communication latency; communication energy consumption; communication reliability expressed in terms of packet error rate, packet loss
WDC Layer	WDC energy and power consumption; WDC energy and power efficiency; speed-up in execution time achieved by WDC; application makespan; probability of successful execution
Service Management Layer	Total cost and reliability of executing an application; and energy consumption and total time consumed for service discovery
Application Layer	QoS requirements and constraints on metrics listed under WDC Layer and Service Management Layer; operational goals such as minimization of makespan, minimization of energy consumption, maximization of reliability and security; optimization weights on individual goals

metric. It is treated as a state metric in the prototype implementation where the communication waveform is fixed and the link goodput cannot be varied by the Communication Layer. It may be treated as a performance metric in the case of a Communication Layer that is equipped with a cognitive engine that can try different waveforms to maximize goodput. Communication latency can be expressed in terms of latency experienced per information bit at individual links or be expressed in terms of end-to-end latency per information bit that takes into consideration the delay incurred in transferring a information bit from a source node to a destination node over several hops. Communication latency can either be a basic metric or a derived metric. In the prototype implementation, communication latency is measured directly by the Communication Layer and, hence, is treated as a basic metric. On the other hand, communication latency can be derived as a function of link SNR, as discussed in Section 3.3.3. In the architecture implementation, the *State_Metrics* data structure bundles together state information that characterizes the network graph, including the number of nodes, mean goodput of the individual links, and current processor load of individual nodes in the network.

Some examples of performance metrics are listed in Table 4.3. Performance metrics are used

Table 4.4: WDC performance metrics for an example application service request.

Metric	Local Processing	Distributed Processing
Makespan	$\left[\hat{M}_{11min},\ \hat{M}_{11max}\right]$ (21/22 secs)	$\left[\hat{M}_{21min},\ \hat{M}_{21max}\right]$ (10/13 secs)
Power Consumption	$[M_{12min},\ M_{12max}]$ (0.78/0.80 W)	$[M_{22min},\ M_{22max}]$ (1.00/1.25 W)
Energy per bit	$[M_{13min},\ M_{13max}]$ (0.035/0.04 J/b)	$[M_{23min},\ M_{23max}]$ (0.08/0.11 J/b)

by the Application Layer and all other layers differently. The Application Layer uses performance metrics to express the following: (a)QoS and resource requirements, (b) goals, and (c) constraints. In the case of multiple goals, the application specifies weights on individual goals. The *Application_Specifications* data structure holds information on the application's task graph, the application's requirements and constraints, and the application's goals. This includes the number of tasks in the application's task graph, the amount of communication between tasks, the estimated execution time and computational energy consumption of each task when executed on a certain computing platform, and the weights associated with energy minimization and makespan minimization.

In all horizontal layers other than the Application Layer, performance metrics are estimates that are computed by the respective decision-making components as a function of state metrics and application specifications. The decision-making mechanisms are expected to ensure that their respective estimated performance metrics match the application specifications. In the WDC Layer, for instance, the Optimizer component takes communication latency and application specifications as its inputs and computes the application's makespan, WDC energy consumption and WDC energy efficiency. If the user application specifies makespan minimization as its goal, then the Optimizer returns the sets of WDC performance metrics as shown in Table 4.4. In Table 4.4, example metric values are shown in parenthesis. The metrics for makespan, denoted by $\hat{M}_{11}$ and $\hat{M}_{21}$, are optimized, while the other metrics are not necessarily optimized. Each metric X is expected to be in the range of M_{Xmin} and

M_{Xmax} with a certain confidence. The application makespan metric specifies the expected time taken to execute all tasks in the user application. In the case of distributed processing, this includes the computational and communication latencies experienced from the start of execution of the application till its completion. The WDC power (energy) consumption indicates the expected power (energy) consumed while executing the user application. In the case of distributed processing, this includes power (energy) consumption by both communication and computational activities. The energy consumption metric can be expressed in units of joules per raw bit which indicates the average amount of energy needed to process a single bit of unprocessed data, and to transfer it over a wireless link to a destination node.

The performance metrics computed by a layer are made available to layers above it and provide a high level abstraction of the estimated costs, risks, and resource consumption associated with an application service request to layers above it. For instance, the Application Layer receives performance metrics from the WDC Layer in the format specified by Table 4.4 and employs them for its decision-making without being concerned with low level performance metrics such as communication latency.

The *WDC_Performance_Metrics* data structure bundles together predicted WDC performance metrics that are important to the application, namely, the application's estimated makespan and total energy consumption when the application is executed using WDC and single-node processing, respectively. *Resource_Alloc_Parameters* bundles together resource allocation parameters that are used by the WDC Layer to achieve optimal execution of the user application. The parameters include computational start and stop times of each task in the given task graph, forward link and reverse link communication start and stop times of each communication activity, number of tasks allocated to each slave node, and list of tasks allocated to each slave node.

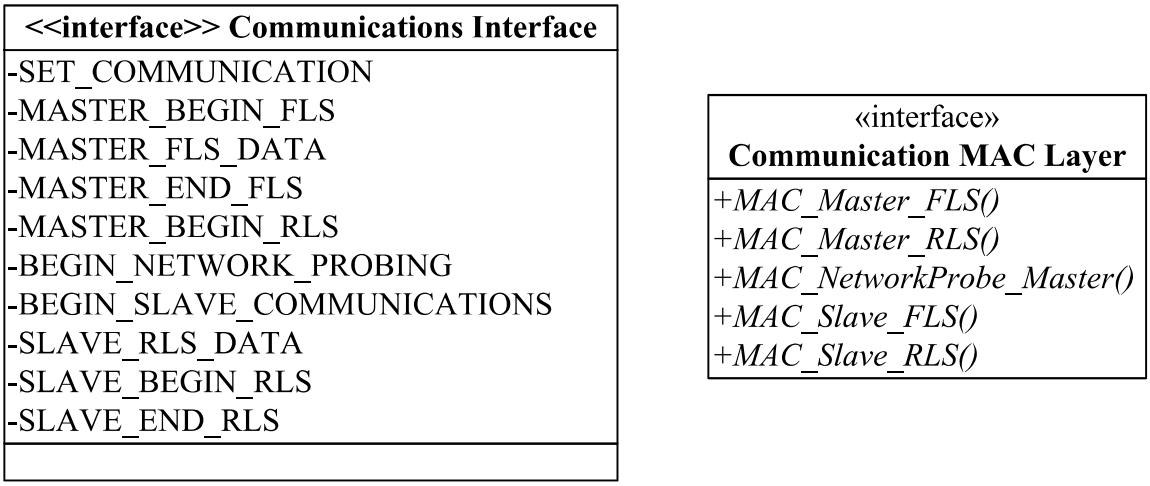


Figure 4.2: A UML interface diagram of Communication Layer components.

4.4 Communication Interface and Communication Layer

The Communication Layer provides communication services to all components in the framework. It delivers data and control messages between components on different nodes; and enables interaction between computational tasks that execute on different nodes. From the perspective of the OSI model, the WCCN Application Layer, Service Management Layer and the WDC Layer constitute the Application Layer in the OSI model. All layers below the OSI Application Layer starting from the transport Layer are part of the Communication Layer.

In the prototype implementation, the Communication Layer uses Liquid Radio for its physical layer. Liquid radio [44] is a Virginia Tech in-house developed open source library of signal processing functions for transceiver operation. A centralized medium access control layer operates over the Liquid Radio physical layer. In the Communication Layer, a data file or control message that is passed on from the layers above is packetized and the packets are transmitted in a serial fashion. A erroneously received packet is re-transmitted several times until it is successfully received or the maximum number of transmission attempts is reached.

In the latter case, the file or message transmission is considered as failed. In a WDC workload distribution scenario, the master node transmits data files to the slave nodes during the forward link session (FLS) and gathers processing results from the slave nodes during the reverse link session (RLS).

As shown in Figure 4.2, the current implementation of the Communication Layer provides communication services for network probing, dissemination of unprocessed data to slave nodes in a forward link session, and gathering processing results from slave nodes in a reverse link session. *MAC_Master_FLS* and *MAC_Master_RLS* execute the master node's FLS and RLS communications. *MAC_Slave_FLS* and *MAC_ Slave_RLS* handle the same at the slave end. *MAC_NetworkProbe_Master* and *MAC_NetworkProbe_Slave* execute network probing at the master node and slave node ends, respectively. Network probing involves gathering the state information of relevant wireless links and nodes in the network. The link connecting a transmitter and a receiver node is probed by transferring a message of specified size over the link to the intended receiver and measuring its transfer characteristics. The transmitter records the total time taken to successfully transmit the probe message to the intended receiver. The measured time takes into consideration the delay incurred due to re-transmissions, propagation delay of each packet, and the receiver's packet processing limitations. Hence, it directly provides an estimate of the link's goodput which is computed as the ratio of the total probe message size and the total time taken to successfully transfer the message. The communication controller provides interfaces to configure waveform and protocol parameters. In cognitive radio networks, the communication controller implements the cross-layer cognitive engine.

The Communication Interface component provides a message interface to the suite of communication services provided by the Communication Layer and handles messages passed between the Communication Layer and layers above it. It receives control messages from the higher layers and invokes the respective communication services as conveyed by the control messages. At the master node, for instance, it handles messages carrying unprocessed data pertaining to a forward link session received from the WDC Layer and passes them

to the *MAC_Master_FLS* service. It sends back messages reporting the status of a communication service to the component that is currently utilizing the service. In addition, it handles control and data messages received by the Communication Layer from other nodes and passes them to the specified destination components.

The interaction between the Communication Interface, WDC Layer components within a node, and WDC Layer components across different nodes occurs via message passing. A common message structure is defined for this purpose. The message header holds information on the message type, payload size, message priority, timestamp, message source and destination (including the component names and node addresses), and a unique message identifier (UMID). The address includes the component name and the node address. The UMID includes the source node address, user application ID, data file ID, and message sequence number (in case a data file spans across multiple messages). The unique identifier is a way to distinguish between messages received from multiple master nodes pertaining to multiple user applications and multiple data files. There is a pre-defined list of message types that are supported by the prototype and the format of message payload contents is defined for each message type. Thus, a message payload is processed by its target component based on the type of the received message.

Figure 4.2 shows the message interface defined by the Communication Interface. The *SET_COMMUNICATION* message configures the Communication Interface for master-slave communications. It contains the following information: the host node's address, host node's designation (whether it is a master node or slave node), and the network's master node address. The Communication Interface sends back a *COMMUNICATION_CONFIGURED* message that conveys the status of the configuration operation. *MASTER_BEGIN_FLS* marks the beginning of a FLS burst and carries information that is used by the Communication Interface to configure itself for FLS communication. A *MASTER_END_FLS* message marks the end of a FLS burst. *MASTER_FLS_DATA* messages carry data files for a forward link session. The Communication Interface conveys the transmission status of all data files pertaining to a FLS session by sending a *MASTER_FLS_STATUS* mes-

sage back to the component that is using the FLS service of the Communication Layer. A *MASTER_BEGIN_RLS* message conveys information on the number of files to receive, the number of participating slaves, and the addresses of the slave nodes. This information enables the Communication Layer to execute the *MAC_Master_RLS* service. The Communication Interface conveys the reception status of all specified data files pertaining to a RLS session by sending a *MASTER_RLS_STATUS* message back to the component that is using the RLS service of the Communication Layer. *BEGIN_NETWORK_PROBING* is a control message that drives the Communication Interface to invoke the network probing service. The message payload carries configuration information, including the transmitter-receiver pair node addresses and the size of the probe message for each individual link, which enables the Communication Layer to execute network probing on all relevant links in both the forward and reverse directions. The Communication Interface conveys the status of a network probing session and the state information of all specified links and nodes gathered during the session by sending a *NETWORK_PROBING_COMPLETED* message back to the component that is currently using the network probing service of the Communication Layer. *BEGIN_SLAVE_COMMUNICATIONS* drives the Communication Interface to configure itself for slave node communications. In a slave node, the *SLAVE_BEGIN_RLS* message drives the Communication Interface to prepare itself for a reverse link session with the master node; the *SLAVE_RLS_DATA* message is used to carry processed results to the master node via the Communication Layer; and the *SLAVE_END_RLS* message indicates the end of a reverse link communication session.

4.5 Wireless Distributed Computing Decision-Making and Resource Management Layer

The WDC Layer is primarily concerned with handling application requests in terms of computational tasks and optimally executing them in the WCCN environment. Specifically, it of-

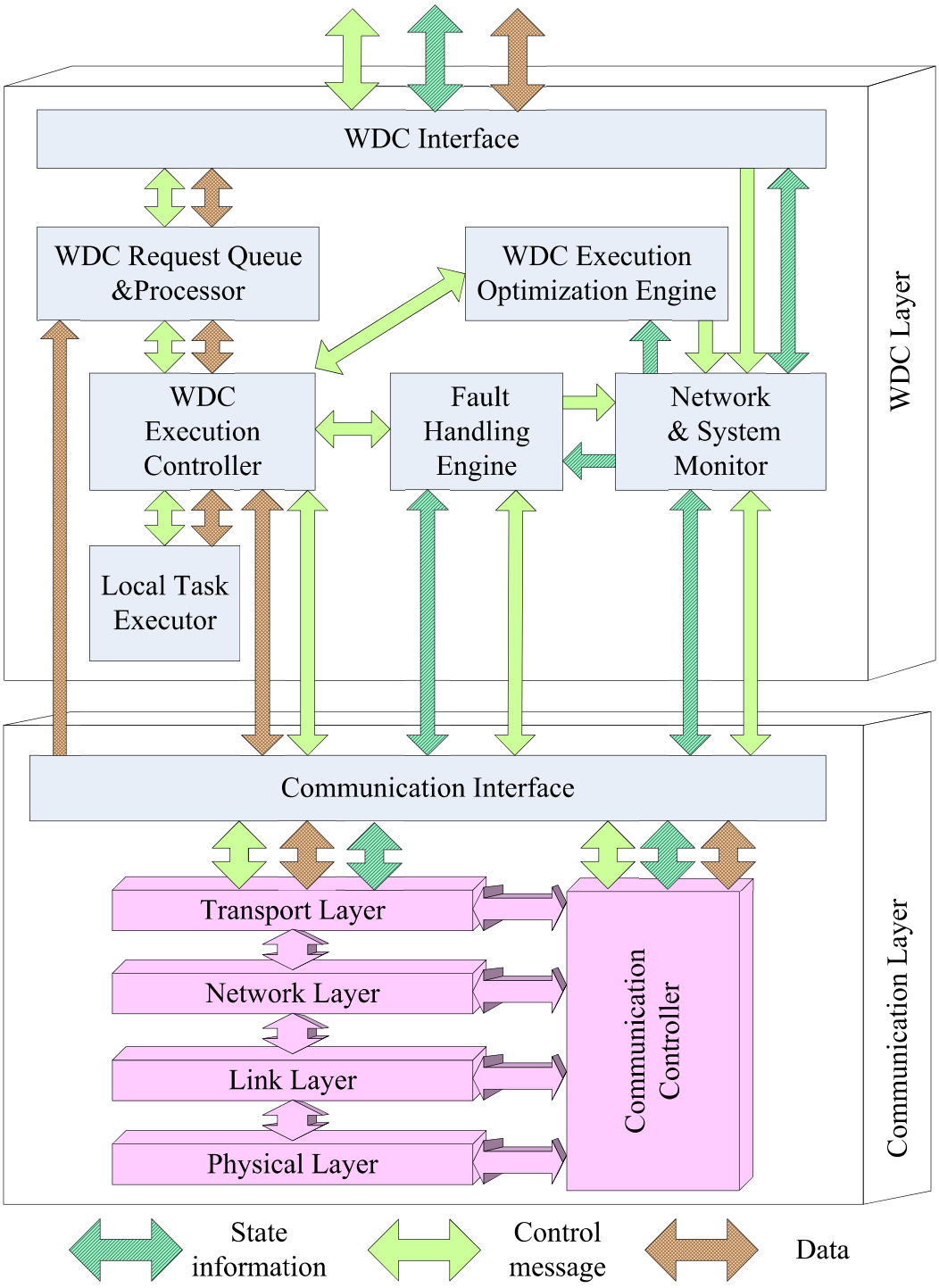


Figure 4.3: The WDC Layer in the WCCN architectural framework.

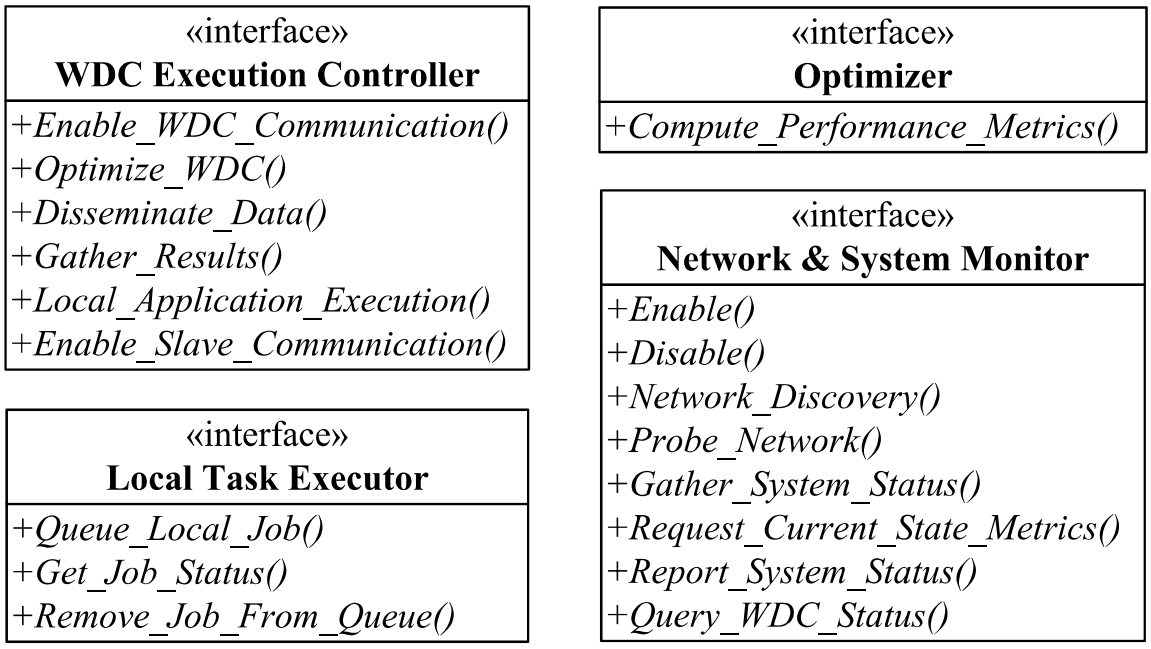


Figure 4.4: A UML interface diagram of core WDC Layer components.

fers the following services: (a) provide WDC execution options and estimates of performance metrics for an application request; (b) provide optimal WDC resource allocation options for an application request; (c) control and management of the execution of distributed tasks, which includes monitoring the remote execution of tasks, monitoring the interaction between tasks, and initiating task migrations; and (d) provide real-time updates on execution status and faults. The WDC Layer may be either centralized or decentralized. In a decentralized WDC Layer, for instance, every node individually makes decisions on resource allocation. In the case of a centralized one, a broker service may decide the resource allocation for a cluster of nodes. The WDC Layer in the prototype is a centralized one.

The WDC Layer comprises the following components: WDC Execution Controller and Coordinator (or WDC Controller in short); WDC Optimizer; Local Computing Resource Manager and Task Executor (or Local Task Executor); Network and System State Monitor; WDC Fault Manager; and Service Request Queue and Processor. The latter two component are not implemented in the current prototype. The WDC Fault Manager monitors the status

of distributed tasks and reacts to events in the operating environment that can potentially impact their successful execution. The Service Request Queue and Processor component is responsible for queuing application requests, which are received either from within the node or from remote service requesters, before they are processed. The role of this component is prominent when handling multiple simultaneously occurring service requests which originate from multiple applications. Figure 4.4 presents a UML interface diagram of the WDC Layer components that are implemented in the prototype. The rest of the section describes the implementation of these components.

WDC Execution Controller and Coordinator The WDC Controller processes an application request and coordinates with various components in the WDC Layer to control and manage the distributed execution of an application.

Optimize_WDC returns *WDC_Performance_Metrics* and *Resource_Alloc_Parameters* for a given set of *Application_Specifications*. It invokes the Optimizer to compute the WDC performance metrics and resource allocation parameters.

Enable_WDC_Communication sends a *SET_COMMUNICATION* message in order to drive the Communication Interface to configure itself for master-slave communications. The function returns after it receives a *COMMUNICATION_CONFIGURED* status message from the Communication Interface.

Disseminate_Data is used to disseminate unprocessed data to slave nodes based on the *Resource_Alloc_Parameters* passed as input to the function, as shown in Figure 4.11. For a single FLS, it sends a burst of messages to the Communication Interface in the following order: one *MASTER_BEGIN_FLS* message, one or more *MASTER_FLS_DATA* messages, and one *MASTER_END_FLS* message. The data messages are disseminated at their respective communication start times as specified by the input communication schedule. A *MASTER_FLS_STATUS* message is then received by the function from the Communication Interface. The transmission status of the data messages conveyed by the *MASTER_FLS_STATUS*

message is used by the WDC Controller to identify failed transmissions and re-schedule the tasks associated with the failed transmissions to other slave nodes or the master node itself.

Gather_Results sends a *MASTER_BEGIN_RLS* message to the Communication Interface which, in turn, drives the Communication Layer to query specified slave nodes for processing results. Upon gathering the results, it stores them at a pre-defined memory location. The function returns after it receives a *MAC_RLS_STATUS* message from the Communication Interface.

Enable_Slave_Communication performs the WDC Controller's functionality in slave nodes. The function runs in an infinite loop and its operation is dictated by a simple state machine comprising the following three states: *SLAVE_COMMUNICATION_INITIALIZATION*, *WAIT_UNTIL_FLS_ENDS*, and *WAIT_UNTIL_RLS_ENDS*. The function initially operates in the *SLAVE_COMMUNICATION_INITIALIZATION* state and transitions to the *WAIT_UNTIL_FLS_ENDS* state after sending the *BEGIN_SLAVE_COMMUNICATIONS* message to the Communication Interface. In the *WAIT_UNTIL_FLS_ENDS* state, the function is prepared to receive computational tasks (and their respective input data) allocated to the slave node. For every task received, the Controller creates a job and queues the job to the Local Task Executor. After receiving all the tasks pertaining to a FLS, the function receives a *MASTER_END_FLS* message which indicates the end of the FLS. Upon completion of processing of all tasks assigned to it, the function sends messages in the following order to the Communication Interface and transitions to the *WAIT_UNTIL_RLS_ENDS* state: one *SLAVE_BEGIN_RLS* message, multiple *SLAVE_RLS_DATA* messages, and one *SLAVE_END_RLS* message. Each *SLAVE_RLS_DATA* message carries results obtained from processing a particular task. In the *WAIT_UNTIL_RLS_ENDS* state, the function does not accept any tasks from the master node. It waits until it receives a *SLAVE_END_RLS* message from the Communication Interface before transitioning back to the *WAIT_UNTIL_FLS_ENDS* state.

Local_Application_Execution manages the execution of tasks assigned for local processing to

the master node based on a specified time schedule comprising task execution start and stop times. A job object is created for each task and is queued to the Local Task Executor. Upon completion of a task's execution, its results are stored in a file at a pre-defined memory location.

Local Task Executor The Local Task Executor handles the execution of tasks assigned to a node and, in turn, manages computational resources in the node. The implementation of this component is platform-dependent while its interface is platform independent. *Queue_Local_Job* is used to queue a new job, process the job, and store its results until they are retrieved. Every job is represented by its XML description which includes the directory path locations of the following entities associated with the job: executable code, input data file, and output results file. In addition, each job is associated with a job ID and current job status. A job ID can be uniquely mapped to a node, user application, and a particular task in the application. The component also provides the *Get_Job_Status* and *Remove_Job_From_Queue* functions to query the status of a currently queued job and remove a job from the job queue, respectively.

Network and System State Monitor The Network and System State Monitor (NSM) is responsible for maintaining WDC environment awareness including self-awareness. It gathers up-to-date information on the network state and local system state; maintains the information in a *Network_State_Information* database; computes state metrics and evolves models of the network and local system based on the state information gathered by it; and processes requests for information received from other components in the local node as well as from remote nodes. Every record in the *Network_State_Information* database holds the following information for a particular link: transmitter node address, receiver node address, probe message size, timestamp of the network probing activity, measured communication latency of the link, mean goodput achieved with the probe message, link reliability, energy remaining in the receiver node, and CPU load information of the receiver

node. Link reliability is a statistical measure of the successful transmission of probe messages on a link. A link is said to have failed under one or a combination of the following conditions: transmitter node failure, poor channel conditions, and receiver node failure. Different records are maintained for the forward and reverse directions of a particular link because a wireless link can be asymmetric with significant difference in link characteristics between the forward and reverse directions. The database module provides functions to recreate the database, append network state information to the database, and retrieve records for a specified link. The database can be accessed only from inside NSM.

Network_Discovery is responsible for gathering awareness of the network configuration. In the current prototype version, all nodes in the network are programmed with fixed addresses which are known to each other a priori thereby enabling them to begin communication with each other upon startup. NSM internally stores the following information on the network configuration: host node address; designation of the host node, i.e., whether it is a master or slave node; address of the master node in the network; number of slave nodes in the network; and addresses of the slave nodes. This information is used during network probing in order to know which nodes (and links connecting them) in the network have to be probed.

Probe_Network executes a single session of network probing. It sends a *BEGIN_NETWORK_* PROBING control message to Communication Interface. It returns after it receives a *NETWORK_PROBING_COMPLETED* message back from the Communication Interface and updates the *Network_State_Information* database with network state information conveyed by the message. In network probing, the probe message size may be increased in order to collect more measurement samples and improve the statistical significance of link measurements.

Gather_System_Status gathers status information of the local host system, including remaining battery energy and CPU load. A component can request network state information from NSM by invoking *Request_ Current_State_Metrics*. The function internally queries the *Network_State_Information* database and returns a *State_Metrics* object. *Report_System_Status*

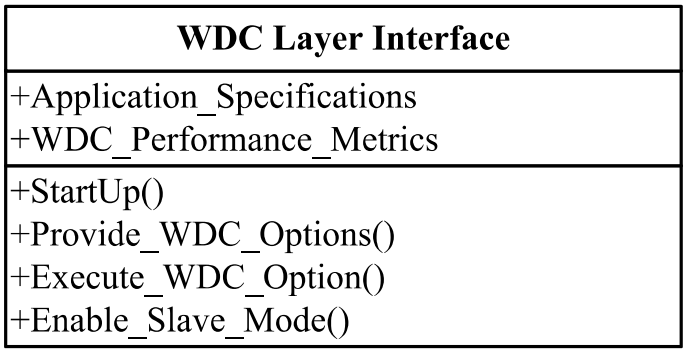


Figure 4.5: A UML representation of the interface provided by the WDC Layer.

and *Query_WDC_Status* provide interfaces for components to report their operational status and query the current status of a load distribution operation, respectively.

WDC Optimizer The WDC Optimizer is the decision-making engine of the WDC Layer and implements adaptive inter-layer optimized resource allocation which is discussed in Section 4.3. *Compute_Performance_Metrics* directs the Optimizer to compute and return *WDC_Performance_Metrics* and *Resource_Alloc_Parameters* for given input *Application_ Specifications*. The metrics are computed by a task allocation - scheduling algorithm based on the state metrics queried from NSM and the input application requirements. The algorithm is a simplified version of the task allocation-scheduling algorithm presented in Chapter 5 and differs from it in two main aspects. First, the simplified version is customized for applications which have a software structure that can be modeled using the task graph model 1, while the original one can handle applications with dependent tasks. Second, the simplified version schedules the computational and communication activities in a node such that they do not overlap in time. This is desired when the communication and computational activities share common computational resources and it is essential that real-time communication activities have untethered access to computational resources. This constraint can be relaxed for platforms that have dedicated resources for application processing and communication signal processing.

WDC Layer Interface The WDC Layer Interface provides functions which can be leveraged, for instance, by user applications to interface with the distributed computing environment in a WCCN. Figure 4.5 shows the WDC Layer interface for load distribution applications. As shown in Figure 4.8, the *StartUp* function invokes various routines that initialize WDC components and configure the Communication Layer for master-slave communications before the node becomes ready for WDC. *Execute_Slave_Mode* directs the WDC and Communication Layers to configure themselves in order to enable a node's operation in slave mode. It is implemented as a wrapper to the WDC Controller's *Enable_Slave_Communication* function. *Provide_WDC_Options* is implemented as a wrapper to the WDC Controller's *Optimize_WDC* function. *Execute_WDC_Option* passes application data and software pertaining to the application's computational tasks to the WDC Layer and directs the WDC Layer to execute the user application in a distributed manner using WDC. Before the execution begins, timers are set to collect timing measurements of the application's time-to-completion. The function has fail-safe mechanisms which switch the WDC Layer to local processing in case remote processing of tasks fails.

4.6 Service Management Layer

The services provided by the Service Management Layer are as follows: (a) provide real-time awareness about services and service options currently being offered in the network; (b) translate a user application request into services currently offered in the network and provide execution options for a given application request; (c) provide real-time status of service executions, and (d) provide service discovery and publication services. There are three main components in the Service Management Layer, namely, Service Environment Monitor, Application Request Processor, and Service Hosting and Publication. The remaining part of this section describes the components. The Service Management Layer is not implemented in the current prototype. The interfaces provided by the Service Management Layer are presented in Figure 4.6. The *Publish_Service* interface in the Service Hosting and Publication

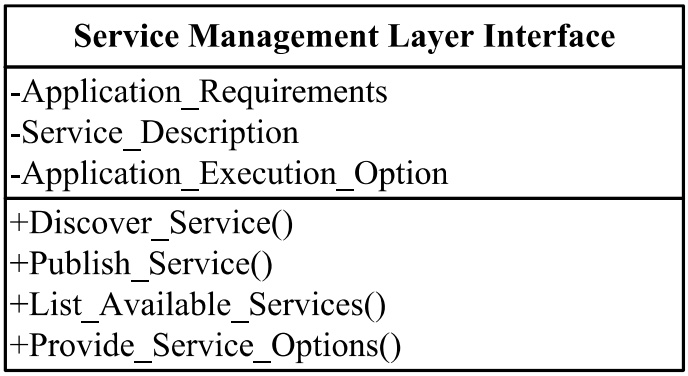


Figure 4.6: A UML interface diagram of the interface provided by the Service Management Layer.

component advertises a given service description to the WCCN environment.

Service Environment Monitor Service availability in the WCCN environment can be characterized in terms of the following: list of all possible services (or service types), services currently accessible to a node, and options provided by each service. A pre-defined list of all possible services aids an application developer to specify rules on composing a user application in terms of services that exist in the list. In order to be capable of maintaining awareness of services in the operating environment, every node should subscribe to a Universal Service Specification Language (USSL) dictionary. The SSL is used by nodes as a common language to describe services when querying and exchanging information on services, as well as during service discovery and service publication. It is used, for instance, by a service requester to specify service options when making a service request to a service provider. A Universal Services Repository (USR) can be used as a platform to publish the USSL dictionary and a list of all possible services including their respective interfaces and service configuration options. The USR can be made available to a node in three ways. It can be pre-loaded to all nodes that are expected to operate in a WCCN. As an alternative approach, the USR can be made available in the web and nodes can synchronize with the master web copy every time they have access to the internet. In the third alternative approach, a node can receive updates on the USR from neighbor nodes in the WCCN. A Local Service Availability

Repository (LSAR) maintains a list of services (which is a subset of the list published in the USR) that are accessible to a node and their service options.

The Service Environment Monitor (SEM) maintains a node's awareness of the service environment. It is responsible for synchronizing with the USR and keeping its LSAR up-to-date. The LSAR is updated by the node either after gathering service availability information via service discovery or whenever the node hears a service publication. The SEM components on different nodes can interact each other and share information on services. The interfaces provided by this component are discussed next. *Discover_Service* searches and determines the availability of a specified service in the WCCN. The service is specified using a *Service_Description* object. *List_Available_Services* returns a list of services in the node's LSAR and their service options.

Application Request Processor *Provide_Service_Options* returns an array of *Application_Execution_Option* objects for an input application request. Each execution option pertains to an approach to execute the application request in terms of services that are listed in the LSAR. A *Application_Execution_Option* object describes an execution option in terms of a task graph and a description of the services that constitute the task graph. The translation of the request into execution options is performed either based on a pre-defined set of rules or may be dynamically determined based on past experience. The translation may or may not be oblivious to the network graph.

4.7 Application and User Interface Layer

This section describes the role of the WCCN Application Layer in the WCCN architecture. The Application Layer hosts a library of user-defined cloud applications; databases details of every application in the library including the application's functionality, format of results presented to the user, QoS requirements, configuration details, and other user preferences;

and performs application-dependent functions required for wireless cloud computing, such as fragmentation of application data into data blocks. In addition, it provides a smart user interface which handles the presentation of results to the user in the user preferred format with consideration of the user preferences in viewing data, user limitations, and platform-dependent display limitations. A user application can utilize the API and features provided by the WCCN framework in order to port itself to a WCCN environment. In this regard, the user interface in the prototype displays the status of a load distribution operation based on the status queried from the NSM component. The status information may, in addition, be used by the application for decision-making purposes. In the prototype, a GUI made using C++ wxWidgets is used to interface with the user application.

4.8 Use Case and Demonstration of Workload Distribution

This section presents a basic use case of WDC workload distribution using the proposed architecture. The example image processing user application used in the prototype demonstration is described first followed by a description of the use case implemented by the prototype.

Image processing application and overview of methodology A face recognition user application is considered for the demonstration. The application takes a set of images as its input and returns details of the persons in the images. An overview of the methodology for load distribution using the example image processing application is illustrated in Figure 4.7. The application comprises a database of known persons where each database record includes the template of a person's face and the person's biographical details. Image processing programs developed based on the Open Source Computer Vision Library [45] are used for image segmentation and image matching purposes in the user application. The source task

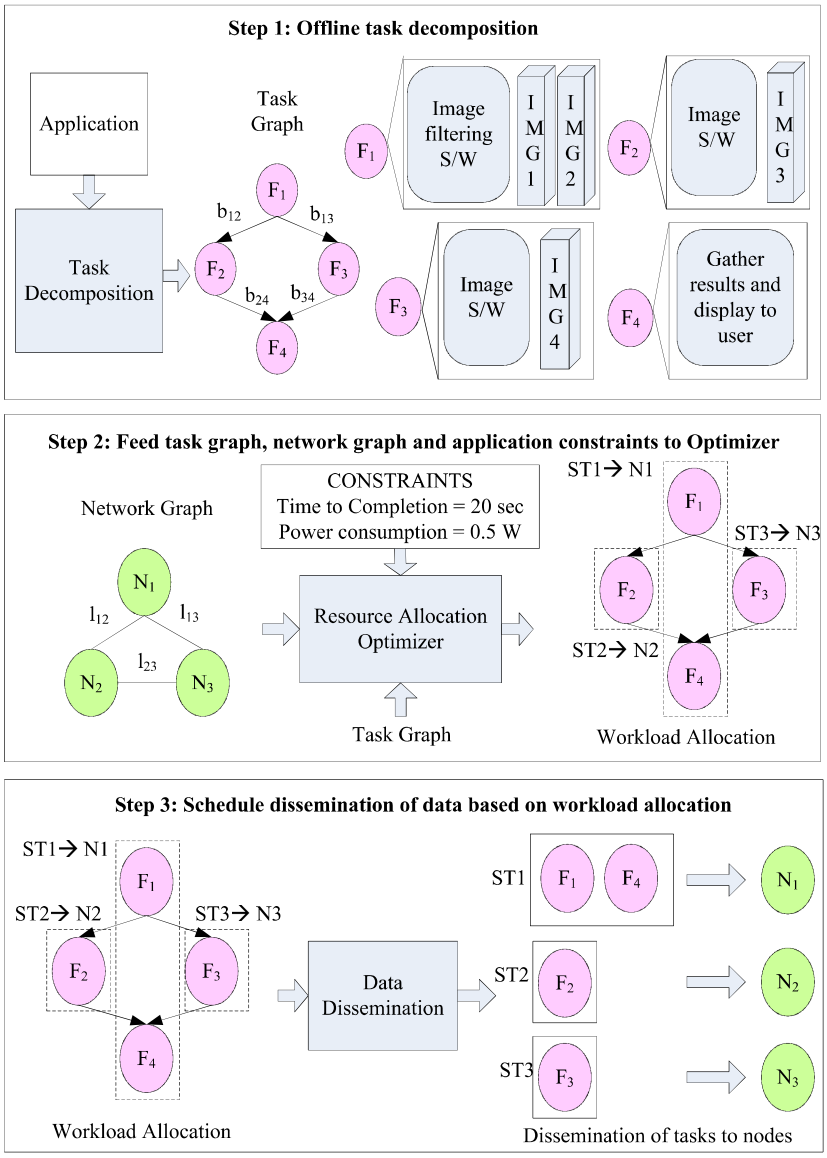


Figure 4.7: A methodology for achieving WDC workload distribution using the WCCN architecture.

in the application's task graph, denoted by F_1 in Figure 4.7, takes the input images, denoted by $IMG1$ and $IMG2$, and segments out the faces in the images. The output images of F_1, namely $IMG3$ and $IMG4$, are passed as inputs to tasks F_2 and F_3, respectively. Each one of the tasks F_2 and F_3 matches its input image against the database of face templates and the biographical information associated with the face template that closely matches with the input image is made available at the task's output in the form of a text file. The sink task, represented by F_4 in the example, gathers all the text result files and presents the information to the user in a user customized format. In this manner, the user application is represented by a task graph that resembles the task graph model 1 such that communication activities involved in the distributed execution of the user application occur only between the master node and the slave nodes with no communication amongst the slave nodes.

The user application is decomposed into the tasks: $\{F_1, F_2, F_3, F_4\}$ and represented in the form of a task graph, as illustrated by step 1 in Figure 4.7. This task graph is provided by the application developer and is stored either in the WCCN Application Layer or the Service Management Layer. Step 2 of Figure 4.7 illustrates workload allocation and scheduling, where ST_i denotes a grouping of tasks allocated to node N_i. In step 3, the WDC Controller disseminates the data to the respective nodes based on the workload allocation. As per an example schedule, the tasks are disseminated in order to nodes 1, 2, and 3.

Use case The interaction between the horizontal layers in the architecture while processing an application request is portrayed in the message sequence diagrams presented in Figures 4.8, 4.9, 4.10, and 4.11.

1. The Application Layer's GUI first prompts the user to configure the node as a master or slave node. Next, it prompts the user to issue permission to allow the node and the user application to connect to the network. If the user provides the permission, the application invokes *StartUp* and spans threads to display real-time status of the application execution to the user and to execute WDC workload distribution. The

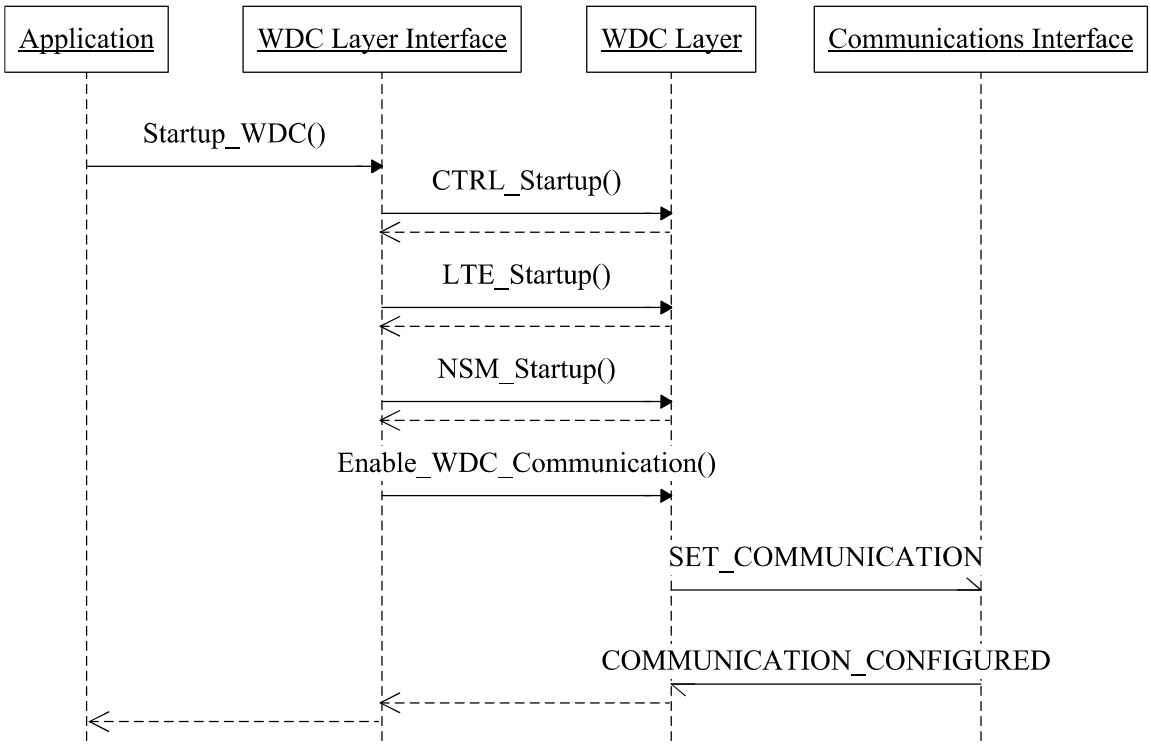


Figure 4.8: Sequence diagram showing the interaction between layers in the WCCN architecture at startup.

startup operation is illustrated in Figure 4.8. The user permission is also a cue for environment awareness processes such as network probing to begin operating in the background. At the slave nodes, the user's permission for network access prompts the Application Layer to invoke the *Enable_Slave_Communication* service of the WDC Layer which enables the node to accept application requests as well as participate in network probing in an automated manner without any further user involvement.

2. Generation and dispatch of application request - The user generates an application request and specifies the goals, requirements and constraints which are bundled into an *Application_Specifications* object by the Application Layer.

3. WDC Layer application request processing and WDC feasibility analysis: *Provide_WDC_Options* is executed as shown in Figures 4.9 and 4.10. It first invokes the *Probe_Network* interface of the NSM component. In the master node, the NSM com-

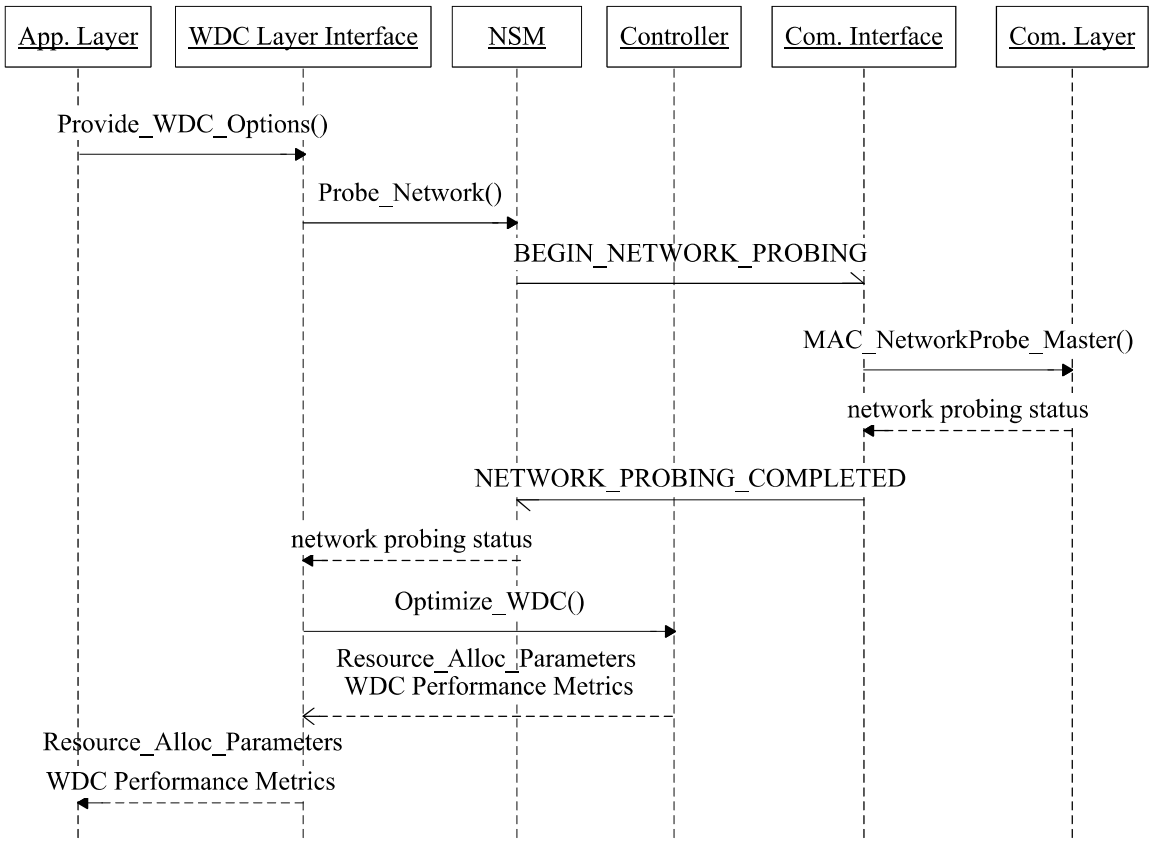


Figure 4.9: Sequence diagram showing the interaction between layers in the WCCN architecture when optimizing WDC workload distribution at the master node.

ponent drives the Communications Interface to invoke the *MAC_NetworkProbe_Master* service of the Communications Layer. When the Communication Layer in the slave node receives a request for network probing from the master node, it runs its *MAC_NetworkProbe_Slave* service which implements the slave end of the network probing protocol. Upon successful return of *Probe_Network*, *Provide_WDC_Options* invokes the WDC Controller's *Optimize_WDC* interface to perform a feasibility analysis and compute WDC performance metrics for local and distributed processing for the given application specifications. The Optimizer returns the metrics and the resource allocation parameters to the WDC Controller. The metrics and parameters are passed to the WDC Layer Interface and, in turn, to the Application Layer.

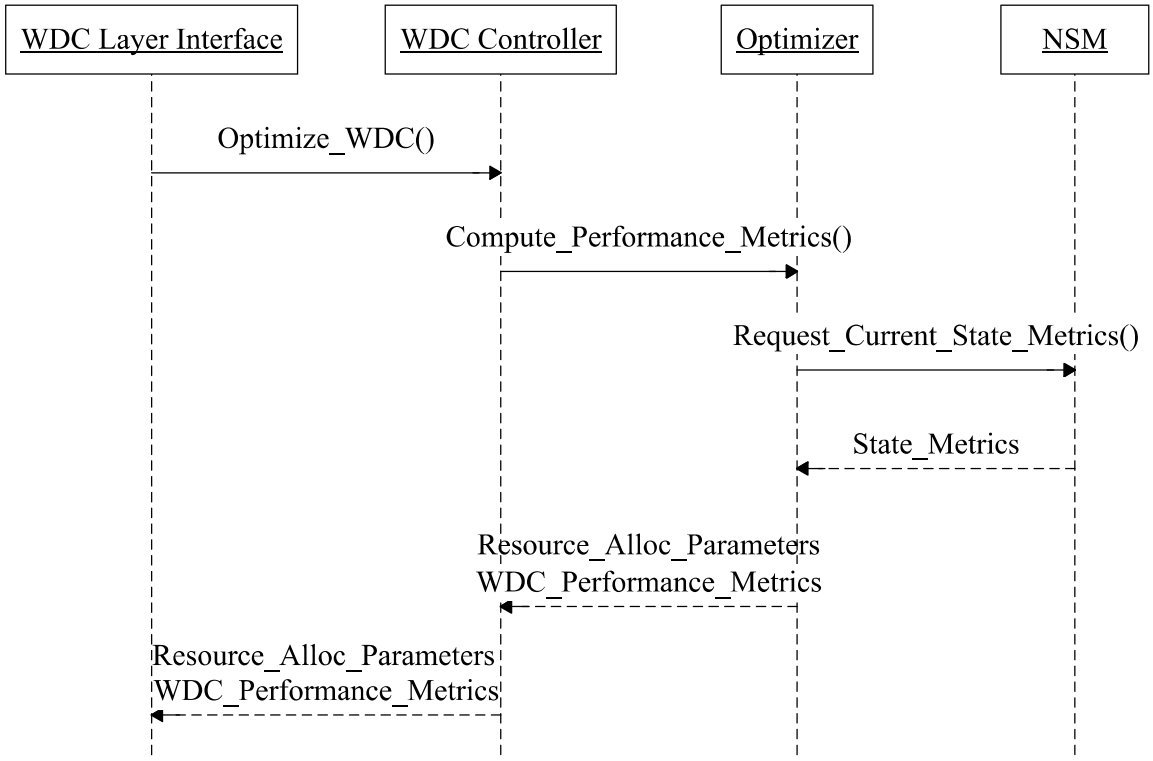


Figure 4.10: Sequence diagram showing the interaction between WDC Layer components when optimizing WDC workload distribution at the master node.

4. Application Layer decision-making and WDC initiation - The WDC performance metrics and resource allocation parameters are displayed by the GUI to the user. This information enables the user to decide between local and distributed processing of the application request. If the user chooses to pursue WDC, *Execute_WDC_Option* is invoked. As an alternative, the application can automate decision-making without user involvement and possibly manipulate the resource allocation parameters. The memory location of the unprocessed data is provided by the application to the WDC Layer.

5. Execution of WDC Workload Distribution - As shown in Figure 4.11, *Execute_WDC_Option* invokes the following functions provided by the WDC Controller in order: *Disseminate_Data*, *Local_Application_Execution*, and *Gather_Results*. The resource allocation parameters returned by the Optimizer are used by *Disseminate_Data* to set the destination node address fields of data messages to the address of appropriate slave

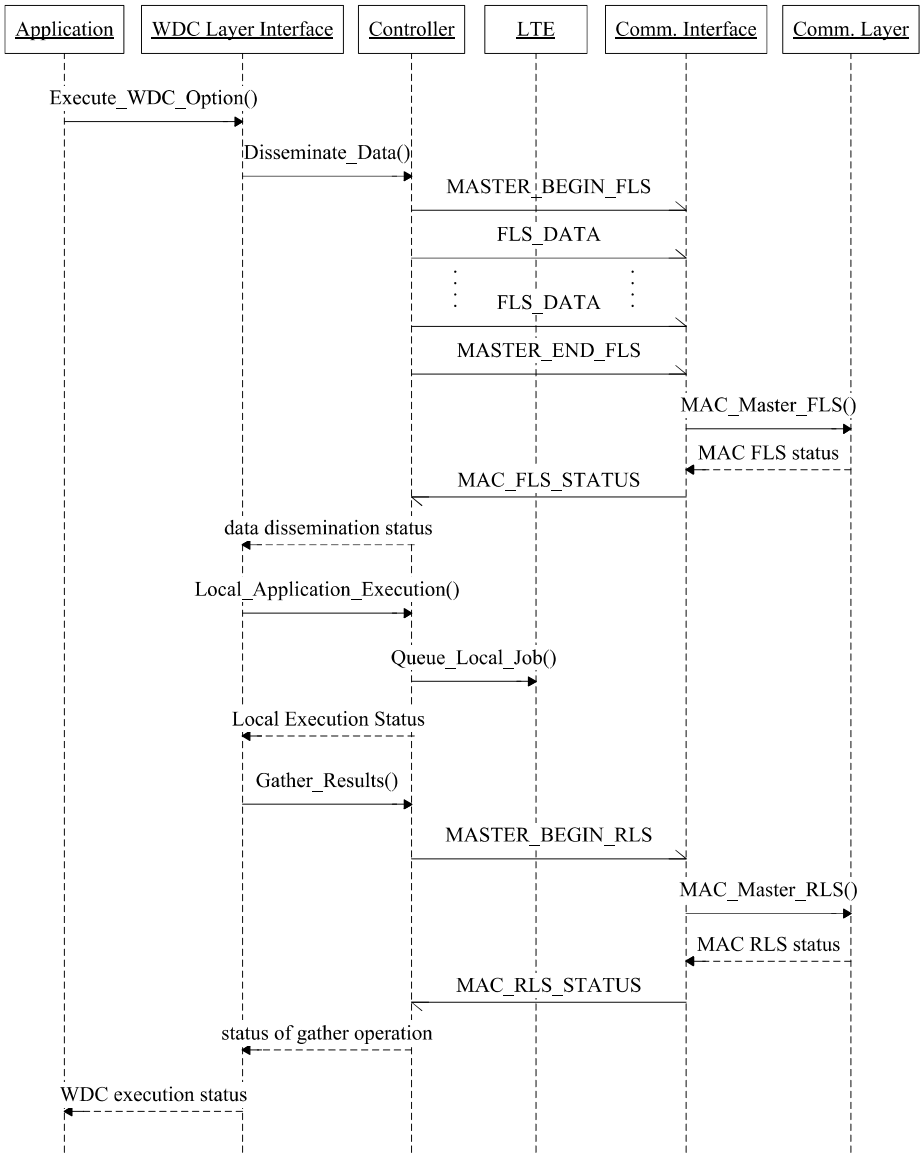


Figure 4.11: Sequence diagram showing the interaction between layers in the WCCN architecture when executing WDC workload distribution.

nodes and schedule the dissemination of data to slave nodes. If any data transmissions fail, as conveyed by the data dissemination status returned by *Disseminate_Data*, the tasks associated with the failed transmissions are re-assigned to the master node.

Local_Application_Execution queues all tasks that are assigned to the master node, as specified by the workload allocation parameters, to the Local Task Executor, including the tasks re-assigned to the master node. The execution start times of these tasks are specified by the Optimizer's resource allocation parameters.

Upon receiving messages from the master node, the Communication Layer at a slave node propagates the messages to the WDC Controller via the Communication Interface. The WDC Controller assigns the received tasks to the Local Task Executor at the time specified by the task scheduling parameters. In this manner, all nodes process tasks as per temporal schedule determined by the master node's Optimizer.

6. Completion of application execution: Upon completion of the task executions, *Local_Application_Execution* stores the results in files located at specified memory locations and returns with the status of the executions. At the slave nodes, the WDC Controller pushes messages carrying the results to the Communication Interface and invokes the Communication Interface to be ready for the RLS. The resource allocation parameters are used by *Gather_Results* to schedule the gathering of results from slave nodes. The WDC Controller at the master node receives the messages that convey the processing results from the slave nodes via the Communication Interface.

   Tasks whose results are not received from the slave nodes due to failure in RLS communication, as specified by the status of the gather operation, are executed locally using *Local_Application_Execution*. *Execute_WDC_Option* returns with the status of all task executions. The Application Layer assimilates the received results and prepares the results in the format which will be presented to the user. The GUI displays the results to the user along with the performance metrics of the WDC operation, such as the measured time-to-completion of the application execution.

Three enhancements can be made to the use case. In the case of applications comprising dependent tasks, any communication messages pertaining to inter-task data passing received by a node can be propagated by the receiver node's Communication Layer to the WDC Controller. Second, the use case can be enhanced to accommodate activities related to the management of the WDC process. The WDC Controller and the WDC Fault Manager can periodically interact with their counterparts in other nodes in order to monitor the state of the remote and local tasks in order to ensure their successful and timely execution. The components can handle any disruptions to task executions due to the occurrence of unanticipated or undesired events and make decisions on task migrations. The role of the Service Management Layer in the use-case described above is greatly simplified. As a third enhancement to the use case, the Application Layer can invoke the Service Management Layer's *Provide_Service_Options* function and pass all the service execution options returned by the function to the WDC Layer. Instead of considering one task graph per application request as done so in the use case, the WDC Layer can choose the optimal task graph in addition to computing the WDC metrics and resource allocation parameters.

4.9 Related Work and Novelty of the Proposed WCCN Architectural Framework

This section first discusses some traditional works on mobile distributed computing. Next, the section discusses some existing works on architectures for mobile cloud computing. Finally, the section points out the contributions of the proposed framework in comparison with existing works.

Past works on software architectures for mobile distributed computing environments primarily focus on the management of computing resources, services management, and higher layer communication protocols. Pullela [46] proposes an architecture for mobile information accessibility that addresses two challenges, namely, the need for a distributed computation

model that accounts for heterogeneous computing capabilities of nodes and the need for dynamic localization of services without placing limitations on the ability to specify services. Pullela [46] also points out that the web may not serve the purpose of mobile information access simply because the web does not have the notion of variable computational capability. The architecture extends the capabilities of Jini and approaches service description in a semantic manner. The architecture is applicable to scenarios where data is not readily available and may have to be computed dynamically with the help of services that are hosted by multiple nodes. An open wireless distributed computing environment is presented by Shpantzer et al. [47] wherein services form the basic building blocks. The architecture has four layers, namely, services, agents, distributed computing paradigms, and application API. An agent may manage several services while a paradigm may interact with several different agents. At the top most layer, the architecture provides API for applications to be developed on this framework and for the development of network management services. Schill et al. [20] present on architectural issues that arise with the integration of mobile computing and communications infrastructures, such as impact of mobility on network configuration, resource access and QoS of communications. It presents a architecture which is intended to operate in mobile nodes and comprises components which make decisions on access to computing resources, execute resource access decisions, provide interfaces for communications, and database information on the distributed computing environment. Pavan et al. [48] discuss on the design requirements of middleware for resource management in mission critical environments. Their middleware comprises low level resource managers which directly manage resources, and high level managers which operate on top of low level managers and function to map service requests received from applications to a set of resources. The middleware is expected to handle applications which compete for resources and impose diverse QoS requirements with strict expectations of guarantee on QoS metrics. Due to the diverse needs of various applications, the middleware is expected to arbitrate between applications and the underlying distributed system platform in an adaptive manner. In the case of limited resource availability, the middleware should be able to provide the best possible

options for the applications. Some architectures for mobile cloud computing are discussed next.

Zhang et al. [49] present an adaptive QoS management framework for mobile computing. The framework is designed for a centralized topology, where the mobile computing environment comprises mobile nodes and a cloud infrastructure which is accessed by mobile nodes through the web. While the resource management aspect of the framework is centrally controlled and performed by the QoS management component in the cloud, QoS environment monitoring is decentralized where mobile nodes provide status information to the QoS management component. Resource management is performed in an iterative manner, where the system first observes the environment, chooses the optimal mode of cloud operation based on the environmental conditions, applies the chosen mode of operation, and observes the performance of the new mode. If observations show that a chosen mode does not meet the QoS requirements, the system changes the mode dynamically. The framework is demonstrated for an internet-based video conference service. There are a couple of limitations in using this framework for WCCNs. First, the framework is not truly targeted towards mobile cloud computing. It is, instead, suited for mobile computing, where the computational services are provided by a remote cloud infrastructure and mobile nodes communicate with the infrastructure for access to its services. Second, the QoS requirements are restricted to communications and do not include computational cost metrics. Third, the framework is verified only through simulation. A real-world prototype is needed to address potential challenges encountered with QoS monitoring.

Fernando et al. [50] present an architecture for mobile cloud computing environments. The architecture resides in all mobile nodes and the nodes collaborate based on a master-slave configuration. The architecture comprises three main components, namely, a cost handler, a resource handler, and a job handler. The components resemble those of the WDC Layer in the proposed WCCN architecture. The cost handler comprises a cost estimator, a price negotiator, and a micropayment module. The cost estimator, which resembles the Optimizer in the proposed architecture, estimates performance metrics and chooses slave nodes based

on the perceived network state and application requirements. The resource handler is similar to the NSM component and the Cloud Environment Map in the proposed architecture. The job handler partitions the application into jobs, creates a job pool, and handles workload distribution of the jobs. The experimental prototype of the architecture uses bluetooth for communications. It is implemented in a heterogeneous computing environment comprising smartphones and a PC. In comparison with this architecture, the partitioning of the core functionalities among components is different in the proposed WCCN framework. In addition to the WDC Layers resource monitoring mechanisms which compare with the resource handler, the Service Management Layer has its own environment-monitoring mechanisms that focus on the service environment in the case of a service - oriented system. A drawback in the architecture design presented by Fernando et al. is that it employs uniform workload distribution. In addition, the workload distribution algorithm does not perform scheduling of computational and communication activities, which is a critical aspect of resource allocation. In addition, the algorithm is not geared towards complex applications that comprise dependent tasks.

Huerta-Canepa et al. [51] present an architecture for offloading computational workloads onto a virtual cloud computing environment. The virtual computing environment is created in the presence of mobile devices which are located in the vicinity of the user. The architecture comprises the following main components: an application manager, a resource manager, a context manager, a P2P component, and an offloading component. The main drawback in the architecture is that there is no documented interaction between the wireless communication component in the prototype and the rest of the architecture.

The proposed WCCN framework builds upon past works and contributes in the following aspects:

1. **Wireless systems approach to mobile cloud computing:** In addition to handling some higher layer aspects of mobile cloud computing, such as services management and application management, the framework employs WDC as an enabling technology

and supports cloud computing in dynamic, uncertain wireless environments. Current mobile cloud computing architectures do not take into consideration the challenges imposed by wireless environments. It may not be optimal to port existing cloud frameworks that are designed for stable and reliable environments to mobile devices in order to enable mobile cloud computing. In order to meet its purpose, the architecture employs the following design approaches.

- **Algorithms for adaptive and environment-aware management of heterogeneous resources in stochastic wireless environments:** The algorithms that implement resource allocation decision-making in the proposed WCCN architecture are designed to operate in stochastic wireless environment, as discussed in Chapter 5. They employ mathematical models, such as those presented in Chapter 3, that capture the dynamics and uncertainty imposed by wireless environments on cloud computing. Workload allocation and scheduling performed by the algorithms is adaptive in that all nodes may not get the same workload. In addition, they are aware of wireless link quality and availability of battery energy resources. Furthermore, the algorithms perform joint allocation of communication, computing, and energy resources in the WCCN. For instance, the algorithms perform workload allocation and scheduling of both communication and computational activities involved in the distributed execution of user applications.
- **Multi-layer approach to decision-making and environment awareness with cross-layer interaction:** The layers in the WCCN framework are each tasked with decision-making and environmental monitoring. In addition, the design takes a cross-layer approach to decision-making, as discussed in Section 4.3.

2. **Targeted topology:** The framework itself is oblivious to the topology of the network. However, the implementation of the individual components and the interaction between them can change with the configuration of the underlying network. The prototype, for instance, is implemented for a master-slave configuration. Unlike mobile

computing architectures, such as the one presented by Zhang et al. [49], the WCCN architecture does not require a cloud infrastructure. Resource allocation in the WCCN architecture, for instance, is performed by the master node itself based on information that it proactively gathers from slave nodes.

3. **Complete development cycle and experimental verification:** The WCCN architecture has been developed over multiple phases involving the development of mathematical model of typical WCCN environments, architectural framework design, design of resource allocation algorithms, and development of an experimental prototype of the architecture. The prototype verifies the real-world functioning of the proposed architecture.

4. **Use of software defined radios for communications:** The prototype employs software defined radios which add tremendous flexibility to radio resource management. In order to have better control over the communications mechanisms, the prototype employs open source radio hardware and signal processing software in the Communications Layer. It uses a custom OFDM transmission scheme with a scheduled MAC protocol that allows re-transmissions. This is seen as a step towards implementation in commonly used wireless standards which use OFDM.

5. **Comprehensive and flexible framework:** The WCCN framework takes into consideration issues related to the application, service environment, wireless distributed computing resource management, and wireless network communications. The framework can be implemented to be strictly layered or it can adopt a flatter structure where a layer can access any other layer.

4.10 Chapter Summary

This chapter presents the proposed architectural framework that enables WCCNs to overlay on a network of radio nodes. The chapter identifies the architecture's core requirements and

the core functionality that have to be provided by the components in the architecture in order to enable operation in challenging WCCN environments from a wireless systems perspective. The architecture is structured according to a component-based approach which allows the implementation of the individual blocks to be varied by the user without having to change the architecture itself. The framework is organized into 4 horizontal layers, namely, the WCCN Application Layer, Service Management Layer, WDC Layer, and Communication Layer. The framework also has provisions for vertical layers such as the Platform Abstraction Layer and Cognition Layer. The interfaces and functionality provided by the horizontal layers and their components, as well as their implementations are presented in this chapter. The application programming interfaces (APIs) exposed by the framework can be used by user applications to utilize the WCCN's hosted services. A simple use case is presented to demonstrate WDC workload distribution using a prototype implementation of the proposed architecture. The use case shows how the various components in the architecture interact with each other to enable WDC workload allocation.

The architecture is evaluated against NTIA's requirements of cloud computing [2] as follows:

- User convenience and standardized access to cloud for universal interoperability - The architecture's layered approach allows the application developer to choose the level of abstraction appropriate for the user application. In addition, the architecture provides API that can be used to port user applications to the WCCN environment. The API standardizes the access by user applications to resources in the cloud.

- Resource pooling - Inside each node, the NSM and SEM components in the architecture frequently monitor the availability of resources and services being offered in the operating environment. This is a distributed approach to resource pooling.

- On-demand autonomous and rapid provisioning and release of resources: In the architecture, resources that are permitted by the user for sharing within the cloud are not statically assigned to any task. Instead, they are provisioned only when an application request is received and provisioned to sufficiently match the application requirements.

Next, resource management and user access to resources is automated and rapid without the need for user intervention. User intervention is optional. In a similar manner, resources are released momentarily in an automated manner upon completion of the execution of an application.

- Optimized resource management - Resource provisioning is optimized such that the right resources are chosen to meet application requirements. The WDC Layer implements adaptive inter-layer resource allocation with prominent roles played by the Optimizer and the NSM components. This mechanism can be extended to include the role of the Service Management Layer.

Chapter 5

Workload Allocation And Scheduling for Wireless Distributed Computing in WCCNs

5.1 Introduction

An introduction to resource allocation in WCCNs is provided in Section 2.3. The efficiency and performance benefits of WDC in WCCNs are dependent on the resource allocation mechanisms employed by the architecture, particularly the workload allocation scheme employed to share the computational workload among the collaborating nodes and the scheduling algorithm employed to schedule the communication and computational activities involved in the distributed execution of the user application. This chapter presents an adaptive workload allocation algorithm and an adaptive task allocation - scheduling algorithm both of which are suited towards meeting the unique requirements and resource constraints in WDC. Both algorithms are designed to function in heterogeneous WCCN environments and provide significant benefits over round-robin workload allocation and scheduling especially

when operating in heterogeneous WCCN environments. A non-adaptive workload allocation scheme, such as one which allocates the workload uniformly among all the nodes in the WCCN without taking into consideration the current network state and resource constraints of individual nodes, would either fail completely or not operate optimally in heterogeneous environments where the availability of resources can vary drastically between nodes. The proposed workload allocation algorithm optimizes energy consumption and the proposed task allocation - scheduling algorithm optimizes both energy consumption and the application's makespan in a heuristic manner. The makespan of an application is the completion time of the last task in the application's task graph, i.e., it is computed as the maximum among the finish times of all tasks in the task graph. The algorithms assume non-preemptive execution and are designed for centralized resource allocation. The algorithms use a comprehensive model of power and energy consumption which is presented in Chapter 3. The proposed workload allocation algorithm aids in the theoretical analysis of WDC performance in resource-constraint and heterogeneous WCCN environments. The proposed task allocation - scheduling algorithm is employed for the practical implementation of the resource allocation component in the proposed WCCN architecture.

In addition to presenting novel resource allocation approaches, the chapter focuses on the analysis of power efficiency, energy efficiency, and makespan achieved by WDC load distribution and scheduling using the proposed algorithms. Simulation results presented in this chapter show the conditions under which WDC workload distribution is efficient (in terms of power, energy, and makespan) over local computing and illustrate the scalability of WDC performance with system parameters such as cluster radius, network size (i.e. number of nodes and cluster density), and computational workload. The system model and assumptions used in this analysis are presented in Chapter 3. The analysis presented in this chapter aids in the fundamental aspects of system design which involve the following: (a) feasibility study to assess if the user application requirements can be met when pursuing WDC under specified network conditions taking into consideration the limits on resource availability and other system constraints; (b) analysis of fundamental performance limitations of WDC work-

load distribution and the impact of WCCN architecture implementation, protocol design, and characteristics of the operating environment on WDC performance; and (c) study of design tradeoffs encountered while pursuing WDC in a WCCN environment.

The rest of the chapter is organized as follows. Section 5.2 discusses the novelty of the proposed workload allocation and scheduling algorithms, particularly in comparison with traditional distributed computing resource allocation algorithms. Section 5.3 presents the analysis of power efficiency achieved by uniform WDC workload distribution in homogeneous environments. Section 5.4 presents the analysis of energy efficiency achieved by WDC using the proposed workload allocation algorithm in heterogeneous environments. Section 5.5 presents the proposed task allocation - scheduling algorithm. Section 5.5.3 presents a simulation-based performance analysis of the proposed algorithm and Section 5.5.4 presents experimental results gathered from demonstrations of WDC using the WCCN prototype (see Chapter 4) and the task allocation - scheduling algorithm. It is shown that the experimental results match with the simulation results. The application of WDC workload allocation to cognitive radio networks (CRNs) is demonstrated with the help of a simple example cognitive engine application in Section 5.6. Section 5.7 presents the chapter summary.

5.2 Summary of Contributions

An elaborate comparison of resource allocation in WDC and WSNs is provided in Section 2.3.4. The following are some of the unique challenges encountered by WDC resource allocation mechanisms in comparison with their counterparts employed in traditional distributed computing: non-trivial cost modeling and optimization, heterogeneity in operating characteristics and resources, stringent resource constraints per node, and potential interference of simultaneous communication activities in wireless channels. In addition, WDC resource allocation algorithms should be able to operate with a master-slave configuration which is common in fundamental WDC load distribution scenarios. Among existing schedul-

ing and task allocation algorithms [11,29,52,53], some schedule both the communication and computational activities of an application's task graph and others are capable of operating in heterogeneous environments. While existing algorithms address some of the design issues unique to WDC, none of the current algorithms are fully ready to operate in WCCN environments. The algorithm proposed by Yu [11], for instance, addresses the need to schedule both communication and computational activities, and uses multiple channels for communication between nodes. It, however, does not apply to heterogeneous environments. The proposed algorithms improve over existing algorithms by fully addressing the unique design issues encountered in WDC. Some of the important design issues are presented next.

Communication cost models The communication models employed by resource allocation algorithms used for distributed computing in WSNs construct a high level notion of the communication cost, which makes them no different from traditional wired distributed computing communication models. The proposed algorithms use a more comprehensive model for communication energy consumption. In addition, the proposed algorithms take into consideration the following practical aspects of WDC resource allocation which are ignored in existing algorithms: (a) separation of the transmitter and receiver energy consumption components when characterizing the communication cost; and (b) energy resource constraints of individual nodes. Typically, the transmitter energy cost is significantly higher than the receiver energy cost in a communication activity. In addition, the resource allocation algorithm should be able to check the individual energy cost incurred at a node against the node's individual energy constraint when evaluating the allocation of tasks to potential nodes in the network. Hence, it is important to decompose the link cost of wireless systems into transmitter and receiver costs. Existing algorithms, on the other hand, consider a single link cost and check the link cost against a link constraint without any consideration of the individual constraints of the transmitter and receiver nodes. The impact of this approach on workload allocation can be explained with the help of an example. Consider a user application that comprises two dependent tasks, where task 2 depends on task 1. Assume that the network

comprises 2 nodes where the nodes can each process only 1 task. A node that is assigned to process task 1 occurs at the transmitting end of a communication activity between task 1 and task 2, while a node that is assigned to process task 2 operates at the receiving end. If node 2 has higher energy resource constraints, it may be preferred to operate node 2 at the receiving end of a communication activity. Hence, task 1 is allocated to node 1 while task 2 is allocated to node 2. Going by the traditional approach of considering a link cost where there is no way to consider the individual constraints of the nodes, node 2 may be assigned to process task 1 and node 1 may be assigned to process task 2 simply because node 2 happens to be more computationally efficient than node 1 in processing task 1.

Exclusive access constraint and orthogonalization of concurrent communication activities The exclusive access constraint is important in WDC where a wireless channel is used for inter-task communications. In traditional systems, communication between different pairs of nodes can occur concurrently due to the availability of point-to-point dedicated wired links between all nodes. Hence, the exclusive access constraint has not been widely applied in traditional distributed computing resource allocation problems [29]. In WDC, however, communication resources are constrained due to limited spectrum availability and the inherent broadcast (and interference) nature of the wireless channel which causes interference between concurrent communication activities that occur at the same RF frequency. Thus, in addition to scheduling the computational activities in a task graph, it is important to schedule communication activities such that the communication activities on a particular channel are orthogonalized, say serialized in time, in order to avoid interference. This shift in the design paradigm results in a notable difference between an application's makespan achieved in traditional distributed computing system and the same achieved in a WDC system.

Resource constraints Traditional task allocation and scheduling algorithms do not perform checks against individual node constraints. In WDC resource allocation, it is important

to check if an allocation - scheduling scheme meets the power, energy, and computational resource constraints of individual nodes, particularly for portable nodes that face stringent resource constraints. In a different perspective, constraints can be viewed as driving factors for offloading the workload. In a cost minimization problem, for instance, all tasks of a user application could potentially be allocated to just one node in order to save on communication costs in the absence of constraints. However, constraints do not let this happen and drive the delegation of tasks to other nodes.

Master-slave configuration The proposed algorithms cater to a master-slave configuration which is common in load distribution applications in WCCNs. Constraints are applied to the workload allocation problem in order to ensure that all source and sink tasks in the task graph under consideration are always allocated to the master node.

5.3 Power Efficiency of Uniform Workload Allocation in Homogeneous WCCN Environments

This section presents the analysis of power savings achieved by the network when employing WDC load distribution in homogeneous environments where there is no computational diversity. In this analysis, it is assumed that the network is operated in power reduction mode and the adaptive transmit power model is employed. The network power consumption is computed using Equation 3.66 and the network power savings is then computed using Equation 3.72.

Figure 5.1 shows a plot of the network power savings achieved for different network sizes and processor models when the network cluster radius is 50 meters. The savings are not plotted for processor model 1 since it exhibits negative power savings as expected due to its relatively linear processor characteristics. An increase in the network size results in a reduction in the total computational power consumption in the network due to the lowering

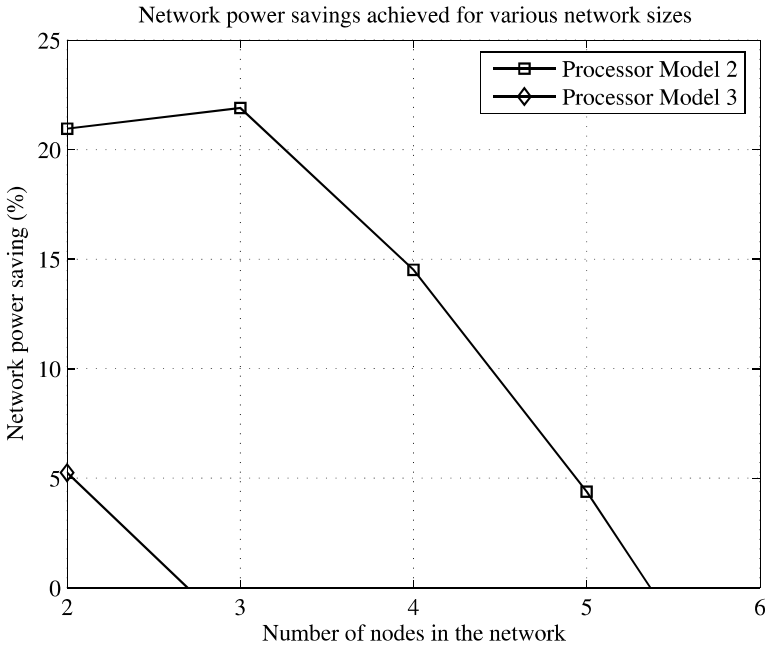


Figure 5.1: Network power savings achieved by WDC workload distribution for different processor models and network sizes.

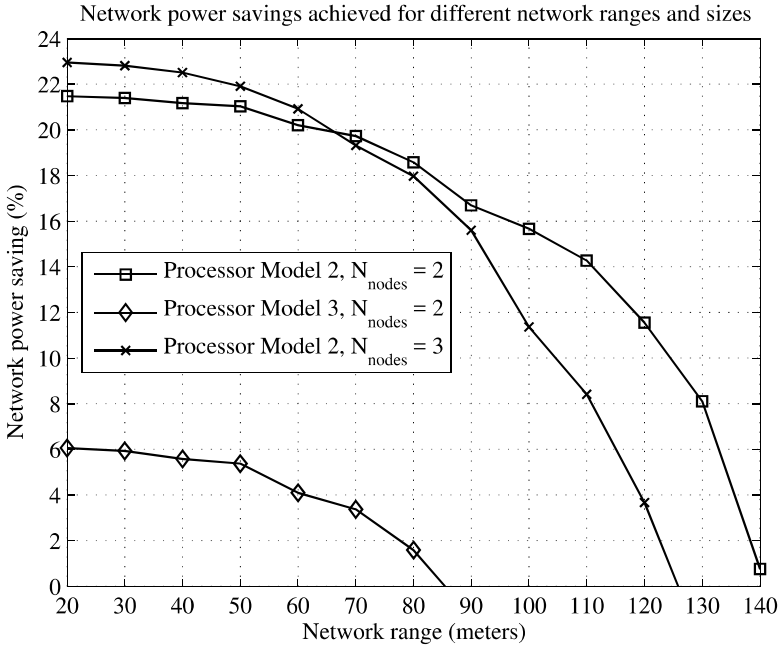


Figure 5.2: Network power savings achieved by WDC workload distribution for different processor models, network ranges, and network sizes.

of the processor clock rate per node and the non-linear processor characteristics. However, this reduction in power consumption is negated by an increase in the total network communication power consumption as more nodes are added to the network. This explains the trend observed in Figure 5.1, wherein network power savings increase until a certain network size (which is $N_{nodes} = 3$ in the case of processor model 2) and decrease thereafter. No power savings is achieved when the communication overhead dominates over the computational power consumption, as observed in the case of processor model 2 for a network size of 6 nodes and processor model 3 for a network with 3 nodes. Furthermore, by comparing the savings achieved for processor models 2 and 3 it can be inferred that high power computational tasks with non-linear characteristics (as represented by processor model 2) yield more power consumption benefits from WDC load distribution when compared to low power tasks (represented by processor models 1 and 3).

Figure 5.2 shows a plot of the network power savings achieved for different network ranges, sizes, and processor models. The savings are plotted for a network comprising of 2 nodes (which represents a limiting scenario) and 3 nodes. Plots that show negative power savings are not shown in the figure. As shown in Figure 5.2, the network power savings scale non-linearly with network range mainly due to presence of the path loss exponent. The drop in power savings with range is attributed to the increase in communication overhead. In the case of processor model 3, the savings are negative for a network size of 3 nodes as validated by the results for processor model 3 shown in Figure 5.1. By comparing the savings achieved with processor model 2 for different network sizes, it is observed that higher savings are achieved at short network ranges (network range of less than 60 meters) with a network size of 3 nodes compared to a network size of 2 nodes. Thereafter, however, the savings drop faster for a large network size due to the presence of the path loss exponent, and also due to the fact that a larger number of nodes and their respective transmit power consumption get impacted by the increase in network range.

5.4 WDC Workload Allocation in Heterogeneous WCCN Environments

The proposed workload allocation algorithm allocates the computational tasks of a user application to qualifying nodes in a WCCN such that energy consumption is minimized. In addition to energy minimization, the algorithm takes into consideration the availability of energy resources in each node required for executing computational tasks allocated to that node. The proposed algorithm uses a combination of stochastic optimization and decision-tree approaches to solve the WDC workload allocation problem. The use of the decision-tree is inspired from Perng-Yi's [54] workload allocation algorithm which is intended for traditional distributed computing networks. Perng-Yi's algorithm performs the minimization of a generic distributed computing cost function under deterministic channel conditions. In addition to the improvements provided by the proposed algorithm as discussed in Section 5.2, the algorithm builds upon the capabilities provided by the Perng-Yi's algorithm in the following ways. First, the algorithm uses stochastic models of transmitter power consumption and power availability which are presented, respectively, in Sections 3.3 and 3.4.1. This enables the algorithm to handle stochastic costs and constraints and, hence, be capable of operating in uncertain wireless environments. Second, it is designed to optimize two objective functions, namely, minimization of the master node's energy consumption and minimization of the network energy consumption.

The rest of this section is organized as follows. The workload allocation problem formulation is presented in Section 5.4.1 followed by a description of the workload allocation algorithm presented in Section 5.4.2. Simulation results of the proposed algorithm are presented in Section 5.4.3.

5.4.1 Problem Formulation

Workload allocation is optimized with respect to one of two objectives, namely, minimization of the network's energy consumption and minimization of the master node's energy consumption. The latter objective maximizes the master node's energy savings which is defined in Equation 3.73. Let x_{ij} denote the binary-valued decision variable which represents the allocation of F_i to N_j. The WDC workload allocation problem can be formulated as given by

$$\text{Obj 1: min } E_{nw} = \sum_{\forall F_i \in F} \sum_{\forall N_j \in N} x_{ij} \left(E_{cp}^{ij} + \sum_{\substack{\forall F_k \in F, \\ k \neq i}} \sum_{\forall N_l \in N} x_{kl} \frac{b_{ik}}{R_{code}} \left(E_{txb}^{jl} + E_{rxb}^{jl} \right) \right) \quad (5.1)$$

$$\text{Obj 2: min } E_{node}^{1} = \sum_{\substack{\forall F_i \in F, \\ j=1}} x_{ij} \left(E_{cp}^{ij} + \sum_{\substack{\forall F_k \in F \\ k \neq i}} \sum_{\forall N_l \in N} x_{kl} \left(E_{txb}^{jl} \frac{b_{ik}}{R_{code}} + E_{rxb}^{lj} \frac{b_{ki}}{R_{code}} \right) \right) \quad (5.2)$$

$$\text{S.T.} \quad x_{ij} \in \{0, 1\}; \sum_{\forall N_j \in N} x_{ij} \leq 1, \ \forall F_i \in F; \quad (5.3)$$

$$x_{ij} = 1 \text{ for } i \in \{1, \ N_{tasks}\}, \ j = 1; \quad (5.4)$$

$$x_{ij} \cdot P_{cp}^{ij} \leq P_{cpt}^{j}, \ \forall F_i \in F, \ \forall N_j \in N; \quad (5.5)$$

$$\text{and} \quad \left(\sum_{\forall F_i \in F} x_{ij} \cdot E_{cp}^{ij} \right) \leq E_{cpt}^{j}, \ \forall N_j \in N. \quad (5.6)$$

Note that, the constants E_{tx1} and E_{rx1} are not shown in Equations 5.1 and 5.2 in order to keep them simple. These constants are added to their respective energy consumption components for every data file transfer between two tasks. Equation 5.1 computes the total energy consumption in the network while executing the application after allocating all the tasks in the task graph to nodes in the network. The first term in Equation 5.1 computes the computational energy consumption associated with the execution of F_i in N_j given that F_i is allocated to N_j (i.e., $x_{ij} = 1$). The second term in the summation computes the cost of communication between F_i and all other tasks that are dependent on F_i. The communication

cost accounts for the amount of communication between F_i and its dependent task F_k, and the cost of communication over the wireless link that connects nodes N_j and N_l which are assigned, respectively, to execute F_i and F_k. The total energy consumption of the master node is computed using Equation 5.2. Equation 5.2 is similar to Equation 5.1 except that it takes into consideration only the costs incurred by the master node for the execution of a user application using WDC. In Equation 5.2, the communication cost is split up in order to account for the cost incurred by the master node when it operates as a transmitter for a particular communication activity and the cost incurred when it is a receiver. Consider a scenario where task F_j is dependent on task F_i such that $b_{ij} > 0$ and the tasks are assigned to different nodes with the master node being one of them. The master node operates as a transmitter when it is assigned task F_i and operates as a receiver otherwise. Note that, the task graph is assumed to be directed and acyclic which means that the interaction between two dependent tasks is uni-directional.

Equation 5.3 expresses the constraint that the decision variable x_{ij} is binary valued. In addition, it expresses the constraint that a task can be allocated to at most one node. This, however, may not be case for applications that allocate a task to multiple nodes to incorporate fault tolerance through redundancy. Equation 5.4 expresses the constraint that the source and sink tasks in the task graph are always allocated to the master node which is represented by N_1. This constraint is important in order to operate with a master-slave configuration. This is also relevant for most WCCN applications besides load distribution applications where the service requester is typically the origin of the request as well as the recipient of the final results of the request after it is processed. The constraint expressed in Equation 5.5 ensures that the power requirements of execution of task F_i in N_j can be met by N_j. In order to impose this constraint, the algorithm takes into consideration the stochastic properties of power availability in N_j. This constraint indirectly checks whether a node has sufficient power and energy resources required to support the communication activities involved in the execution of tasks assigned to it. Equation 5.6 is similar to Equation 5.5 except that it imposes energy constraints of individual nodes.

The formulations presented in Equations 5.1 and 5.2 can be solved by enumerating the various possible allocations of tasks to nodes in the form of a tree structure, such as the one illustrated in Figure 5.3. In Figure 5.3, E_{cmb}^{ij} represents the sum of the per bit transmitter and receiver power consumption components pertaining to communication on link L_{ij}. The problem can then be solved by using tree search algorithms [54]. In Figure 5.3, the i^{th} level of the tree corresponds to the i^{th} task in the task graph, where the tasks may be topologically sorted. Vertices in the tree are shaded with a color code, where each color corresponds to a particular node in the communication graph. Assuming that red corresponds to node N_j, a red vertex at level i in the tree represents the allocation $F_i \rightarrow N_j$. Every branch in the tree is labeled with a branch metric. Considering a branch that leads to a particular vertex, the branch metric accounts for the cost of the allocation represented by that vertex. The tree structure also captures the dependencies between decisions made on the allocation of different tasks. When traversing a particular path in the tree, the decision on allocating the $j + 1^{th}$ task to a particular node is made with knowledge of the decisions made on the allocations of all tasks $F_1,\ F_2, \ldots,\ F_j$. It is important to condition the decision made on one task on decisions made on other tasks, particularly when the tasks are dependent and there is interaction between tasks. Consider a simple workload allocation problem involving two dependent tasks, namely, F_i and F_k. The following question arises after allocating F_i to N_j: is the allocation of F_k to N_l optimal and feasible given the cost of execution of F_k in N_l, cost of communication between N_j and N_l, and the resource constraints of N_l. When considering the allocation of a particular task, the workload allocation algorithm evaluates its allocation to all possible nodes and chooses the node allocation that optimizes the objective. Figure 5.3 is constructed for a workload allocation problem involving a task graph comprising 3 tasks and a network comprising 3 nodes including the master node. The branch metrics shown in the tree are formulated taking into consideration the following precedence constraints in the task graph: task 1 is the source task, task 2 depends on task 1, and task 3 is the sink task and it depends on both task 1 and task 2.

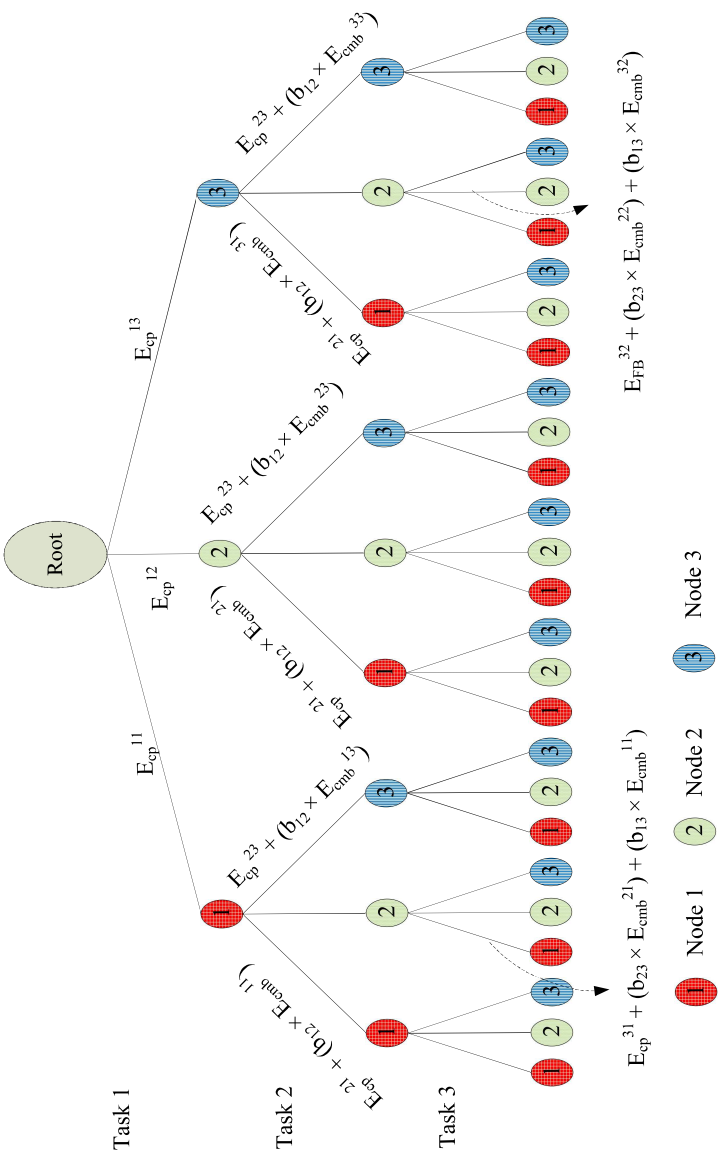


Figure 5.3: A decision-tree formulation for solving the WDC workload allocation problem.

5.4.2 Algorithm Description

Before running the algorithm on the decision tree, the power supply constraints specified by Equation 5.5 are applied to the solution space which is represented by the tree. The tree is pruned to eliminate nodes that cannot process the least power-intensive tasks due to their respective power supply constraints. The power supply constraint of node N_i is computed using Equation 3.60 by taking into consideration the maximum distance from the node to a potential receiver node in the network. This is done in order to take into consideration the worst case power availability in the node. After applying the power constraints, the algorithm does not execute if the master node does not possess sufficient power to support the execution of the source and sink tasks. In addition, the algorithm cannot run if one or more tasks in the task graph cannot be executed by any of the nodes in the network. The subset of candidate nodes for the execution of task F_i that remains after applying the elimination rules is denoted by N_G^i, where $N_G^i \subset N$.

The proposed workload allocation algorithm is presented in Algorithm 1. The algorithm operates in an iterative manner. The first step of the algorithm involves the initialization of algorithm variables. ρ denotes the current iteration of the algorithm and $C_{min}(\rho)$ denotes the optimized value of the objective (or minimized cost) at the end of the ρ^{th} iteration of the algorithm. During each iteration, the algorithm traverses one or more unexplored paths in the tree with the goal of finding a workload allocation solution that yields an objective value, namely $C_{min}(\rho)$, that is improved compared to the objective value achieved in the previous iteration of the algorithm, namely, $C_{min}(\rho - 1)$. N_L^i represents a set of unexplored branches emanating from a particular vertex in the tree. E_{ap} represents the upper limit imposed by the user application on the master node's total energy consumption or the total energy consumption of the network. When operating with objective 1, E_{ap} may be set to the total energy consumed by the master node for executing the user application locally. By doing so, the algorithm seeks for a workload allocation scheme which, at a minimum, achieves lower energy consumption than local processing. BM_{ij} denotes the metric of a branch associated

Algorithm 1 Decision-tree search algorithm.

Initialize algorithm variables as follows: Set $\rho = 1$ and $C_{min}(\rho) = E_{ap}$ for $\rho = 0$.
Let i denote the current level of the tree being traversed by the algorithm.
while (tree has not been completely explored) **OR** ($\delta < R_\delta$) **do**
  if (visiting vertex for the first time) **then**
    Initialize $N_L^i = N_G^i$
    Compute branch metrics for all branches emanating from the vertex:
    BM_{ij}; $\forall N_j \in N_L^i$ (use Equations 5.7 - 5.9)
  end if
  Choose branch j with the minimum branch metric, where
  $j : \left\{ BM_{ij} = \min_{\forall k \in N_L^i} \{BM_{ik}\}\,;\ \text{where } N_j \in N_L^i \right\}$
  Compute objective C_{net} using BM_{ij} (see Equations 5.10 and 5.11)
  if (constraints are satisfied (Equation 5.12)) **AND** ($C_{net} < C_{min}(\rho - 1)$) **then**
    Update cost variables and propagate down the tree by one level (see Figure 1a)
  else
    while ($i > 1$) **AND** ($N_L^i = \{\phi\}$) **do**
      Traceback one level and update cost metrics (see Figure 1b)
    end while
  end if
  if ($i = N_{tasks} + 1$) **then**
    Compute C_{min} and δ using Equations 5.13 and 5.14. {End of iteration}
    Set $N_L^{i-1} = \{\phi\}$
    Traceback at least 2 levels and update cost metrics (see Figure 1b)
    while ($i > 1$) **AND** ($N_L^i = \{\phi\}$) **do**
      Traceback one level and update cost metrics (see Figure 1b)
    end while
    Initialize $X_{\rho+1,i} = X_{\rho,i}$; $1 \leq i \leq N_{tasks}$
    Increment ρ by 1
  end if
end while

with the allocation of F_i to N_j. The branch metrics quantify the impact of allocating F_i to N_j on the overall objective. $X_{\rho,i}$ is a variable that stores the index of the node that is assigned to process F_i in the ρ^{th} iteration of the algorithm.

As the algorithm explores the tree, whenever the algorithm visits a vertex in the tree for the first time branch metrics are computed for all branches directly branching out from the vertex. This step is performed, for instance, when the algorithm traverses from the root of the tree for the first time and evaluates all candidate nodes for the allocation for a source task. In this dissertation, WDC workload allocation in master-slave configurations is analyzed for applications comprising a single source task and a single sink task where the source and sink tasks are allocated only to the master node. For the sake of maintaining generality, however, the algorithm is described in this section without taking into consideration these special conditions. In the simulation analysis presented in Section 5.4.3, these conditions are applied to the decision tree thereby causing the tree to have only one branch emanating from the root vertex and have only one branch connected directly to a leaf. The sole branch in both cases represents, respectively, the allocation of the source and sink tasks to the master node. The metrics on all branches emanating from the root of the tree are computed as given by

$$BM_{ij} = E_{cp}^{ij};\ \forall N_j \in N_L^i,\ \text{where } i = 1. \tag{5.7}$$

When the constraint of Equation 5.4 is applied, N_L^i comprises only the master node when $i = 1$. In all other levels of the tree, i.e., for tasks other than the source and sink tasks, the

branch metrics for the two objectives are computed as given by

$$\begin{aligned}
\text{Obj 1: } BM_{ij} &= E_{cp}^{ij} + \sum_{k=1}^{i-1} 2 \times \left(E_{tx1} + E_{rx1} \right) \\
&+ \sum_{k=1}^{i-1} \left(\frac{b_{ki}}{R_{code}} \cdot \left(\overline{E}_{txb}^{lj} + \overline{E}_{rxb}^{lj} \right) + \frac{b_{ik}}{R_{code}} \cdot \left(\overline{E}_{txb}^{jl} + \overline{E}_{rxb}^{jl} \right) \right), \ \forall N_j \in N_L^i; \quad (5.8) \\
\text{Obj 2: } BM_{ij} &= \left\{ \begin{array}{lll} E_{cp}^{ij} + \sum\limits_{k=1}^{i-1} \left(E_{rx1} + \left(\frac{b_{ki}}{R_{code}} \cdot E_{rxb}^{lj} \right) \right) & ; & j = 1 \\ \sum\limits_{\substack{k=1 \\ X_{\rho,k}=1}}^{i-1} \left(E_{tx1} + \left(\frac{b_{ki}}{R_{code}} \cdot \overline{E}_{txb}^{1j} \right) \right) & ; & \forall N_j \in N_L^i - \{1\} \end{array} \right\}, \quad (5.9) \\
\text{where} \quad & l = X_{\rho,k}, \ F_i \in F - \{F_1, \ F_{N_{tasks}}\}.
\end{aligned}$$

A partial allocation scheme at level k of the tree refers to the allocation mapping: $F_i \rightarrow N_j$, where $j = X_{\rho,i}$ and $i = 1$ to k, and is associated with the path traversed by the algorithm to reach the vertex under consideration. In this regard, a full allocation scheme is determined after the algorithm reaches a leaf of the tree. C_{cpe}^i and C_T^i are cost variables used by the algorithm to store, respectively, the computational energy consumption and the total energy consumption of node N_i, and are evaluated for a partial allocation scheme. They are initialized to zero at the beginning of the algorithm. C_{net} denotes the objective computed for a partial allocation. With knowledge of the branch metrics of all branches emanating from a vertex, the next step is to traverse the optimal branch from the vertex. Note that, a branch metric is a function of the respective node's transmitter energy consumption which is a random variable as discussed in Section 3.3.2. Hence, a comparison of branch metrics involves a comparison of their mean values. The mean value of a branch metric is computed using Equations 5.8 and 5.9. The branch with the minimum metric is chosen and the objective is evaluated for the partial allocation, as given by

$$\begin{aligned}
\text{Obj 1: } C_{net} &= \left(\sum_{\forall j \in N} C_T^j \right) + BM_{ij}, \quad (5.10) \\
\text{Obj 2: } C_{net} &= C_T^1 + BM_{ij}, \quad (5.11)
\end{aligned}$$

Algorithm 1a Pseudo code for updating costs while traversing down tree one level.

i and j, respectively, denote the index of the task (or tree level) and index of the node (or the chosen branch) under consideration, in the allocation $F_i \rightarrow N_j$.

1: Update the set of unexplored candidate nodes: $N_L^i = N_L^i - \{j\}$
2: Update the computational energy cost variable: $C_{cpe}^j = C_{cpe}^j + E_{cp}^{ij}$
3: Update the total energy cost variable: $C_T^q = C_T^q + BM_{ij}$; where $q = j$ for objective 1 and $q = 1$ for objective 2
4: Set $x_{ij} = 1$ and $X_{\rho,i} = j$
5: Increment the current tree level: $i = i + 1$

Algorithm 1b Pseudo code for updating costs while traversing up tree until an unexplored tree node is encountered.

Variable i denotes the index of the task (or tree level) under consideration.

1: Decrement the current tree level: $i = i - 1$
2: Update the computational energy cost variable: $C_{cpe}^j = C_{cpe}^j - E_{cp}^{ij}$; where $j = X_{\rho,i}$
3: Update the total energy cost variable: $C_T^q = C_T^q - BM_{ij}$; where $j = X_{\rho,i}$; and $q = j$ for objective 1 and $q = 1$ for objective 2

where i and j, respectively, denote the current tree level and the chosen branch.

Next, the workload allocation problem constraint specified by Equation 5.6 is applied. Since the constraint represented by E_{cpt}^j is a stochastic random variable, a statistic of the random variable is used for the constraint check as given by

$$C_{cpe}^j + E_{cp}^{ij} \leq \left(r_{cpt}^j \times T_{supply}^j\right), \tag{5.12}$$

where T_{supply}^j denotes the lifetime of the power supply in node N_j and N_j is the node associated with the branch chosen by the algorithm. If the constraint is satisfied, the algorithm proceeds down the tree to the next level via the chosen branch and the cost variables are updated for each node as shown in Algorithm 1a. In addition, the chosen branch is marked as explored by removing it from the subset N_L^i. If the constraint is not satisfied, the algorithm avoids the current tree vertex and seeks a different path through the tree. In that case, the algorithm traverses back one level up the tree, as shown in Algorithm 1b, and searches over

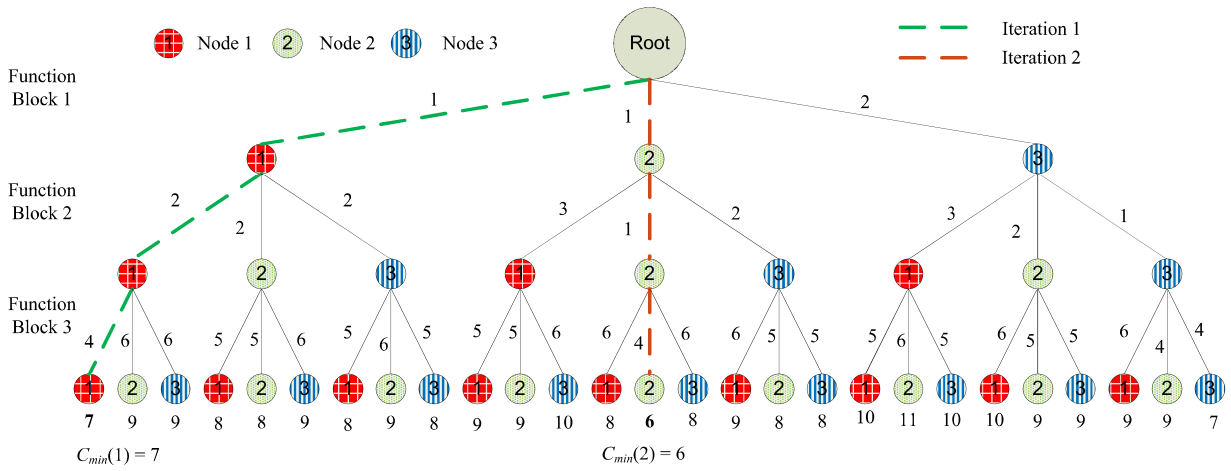


Figure 5.4: Illustration of decision-tree search for cost minimization.

an unexplored part of the tree. The procedure is repeated again for the next visited vertex. The algorithm completes an iteration upon reaching a leaf. At the end of an iteration, C_{min} and the improvement in objective, denoted by δ, are computed for that iteration, as given by

$$C_{min}(\rho) = \sum_{j} C_T^j; \text{ where } N_j \in N \text{ for objective 1 or } j = 1 \text{ for objective 2,} \quad (5.13)$$

$$\delta = C_{min}(\rho - 1) - C_{min}(\rho). \quad (5.14)$$

The algorithm functions iteratively until all of the tree is explored or if the minimum cost $C_{min}(\rho)$ has not improved by more than R_δ when compared with that achieved in the previous iteration, where R_δ denotes the minimum value of δ expected by user. After the algorithm terminates successfully, the energy savings metric is computed by treating the value of C_{min} achieved for the last iteration of the algorithm as the WDC energy consumption. Figure 5.4 illustrates the operation of the proposed algorithm for a simple scenario with example branch metrics. The constraints from Equations 5.4 and 5.5 are ignored for this example merely to portray the iterative operation of the algorithm.

Theoretical analysis of the complexity of the proposed algorithm is outside the scope of this

dissertation. However, some insight into the empirical execution time of the algorithm is provided. The execution time of the algorithm depends on the complexity of the decision tree. It not only depends on the number of nodes under consideration and the number of tasks in the application's task graph, it is also influenced by several other factors such as resource constraints in individual nodes and diversity in channel conditions between nodes. For instance, the decision tree may be pruned when most of the nodes do not have the minimum resources necessary to support even the smallest computational workload that may be delegated by the master node. In that case, the complexity of the decision tree is greatly reduced thereby reducing the execution time of the algorithm.

5.4.3 Simulation Results

This section presents a simulation study of workload allocation in homogeneous and heterogeneous environments using round-robin workload allocation and the proposed workload allocation algorithm. In addition to providing insight into the performance of the proposed workload allocation algorithm, the simulation results presented in this section show the scalability of WDC efficiency in stochastic heterogeneous environments w.r.t. the system parameters, namely, computational power consumption, application workload, network size, network range, and power supply availability. The system model and simulation assumptions used in this analysis are presented in Chapter 3. A power supply of 10 watts is used in the simulations in order to emulate relaxed power supply constraints. With the exception of one simulation scenario, all results presented in this section are simulated assuming a message size of 20 kbits for the interaction between the application's computational tasks when task graph models 1 and 2 are employed. In simulations which employ task graph model 3, the message sizes are randomly chosen from the range [1, 20000] bits.

Scalability with computational power consumption Figure 5.5 shows the scalability of energy savings with computational power consumption, total computational workload of

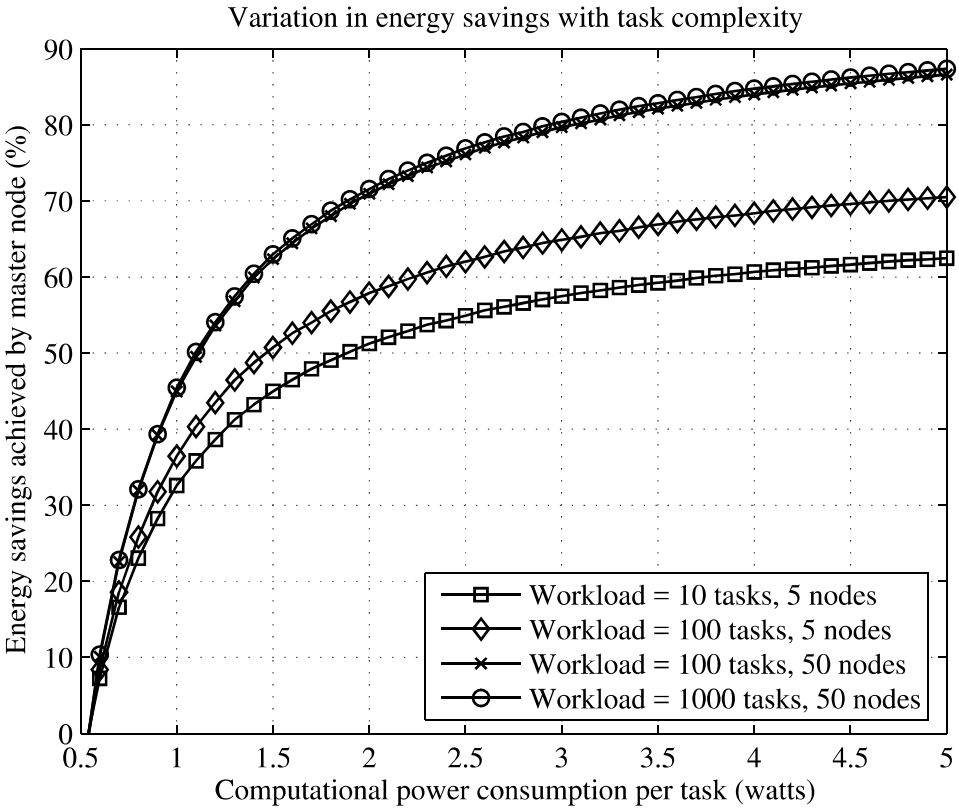


Figure 5.5: Master node's energy savings achieved by workload distribution for various computational power consumption values and application workloads when there is no computational diversity.

the user application, and network size when round-robin workload allocation is employed in an environment where there is no resource diversity. Task graph model 1 is employed in this analysis and the execution time is fixed to 1 second for all tasks regardless of which node executes the task. Power supply is fixed to a deterministic value of 10 watts for every node in the network. The network is located within a cluster radius of 50 meters and each node is assumed to operate per the adaptive transmit power model.

It is observed that energy savings do not scale linearly with the application workload. This is because higher workloads result in an increased delegation of the total workload to the slave nodes when a round-robin load balancing scheme is employed. An increase in workload delegation, in turn, results in lowering of the computational energy consumption at the

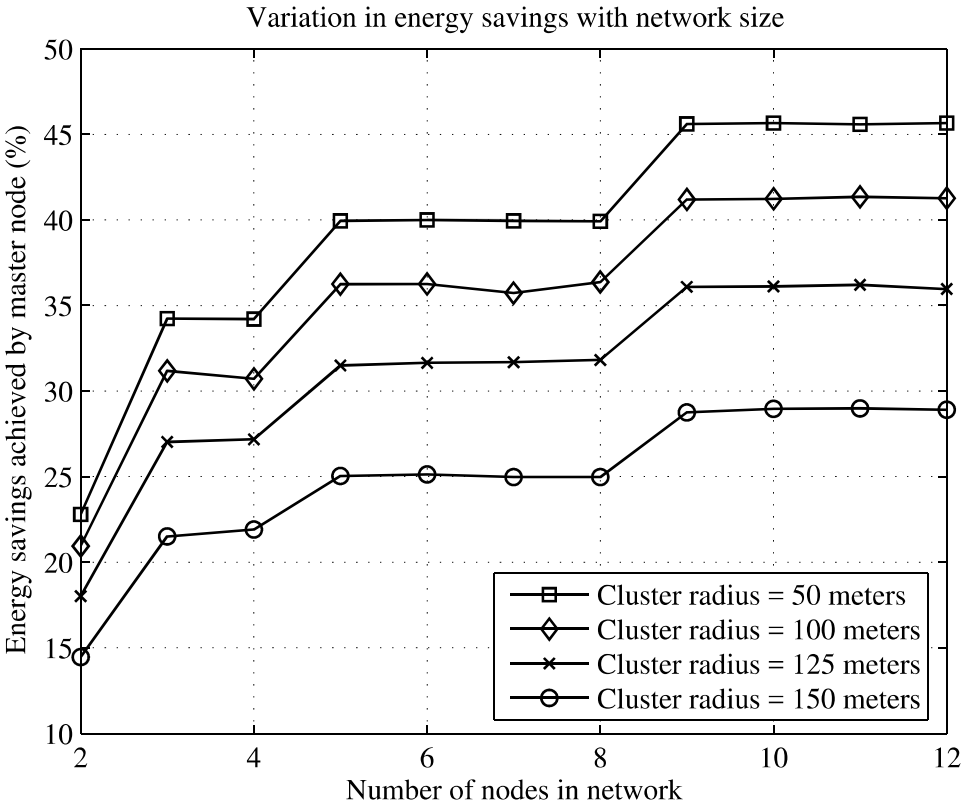


Figure 5.6: Master node's energy savings achieved by workload distribution for various network sizes and network ranges when there is no computational diversity.

master node, but this reduction in energy consumption is achieved at the cost of increased communication energy consumption for the master node.

Energy savings scale non-linearly with the computational power consumption per task. An increase in task complexity rapidly increases the savings until a certain point and, thereafter, the curve flattens. This behavior can be explained in two ways. First, the master node achieves savings in its total energy consumption whenever the computational energy consumption per task is greater than the communication energy consumption associated with a task. This explains the increase in savings with computational power consumption. Thereafter, however, the law of diminishing returns comes into effect. The scaling in energy savings with computational power consumption per task can also be reasoned by evaluating the expression for the master node's energy savings presented in Equation 3.73. When con-

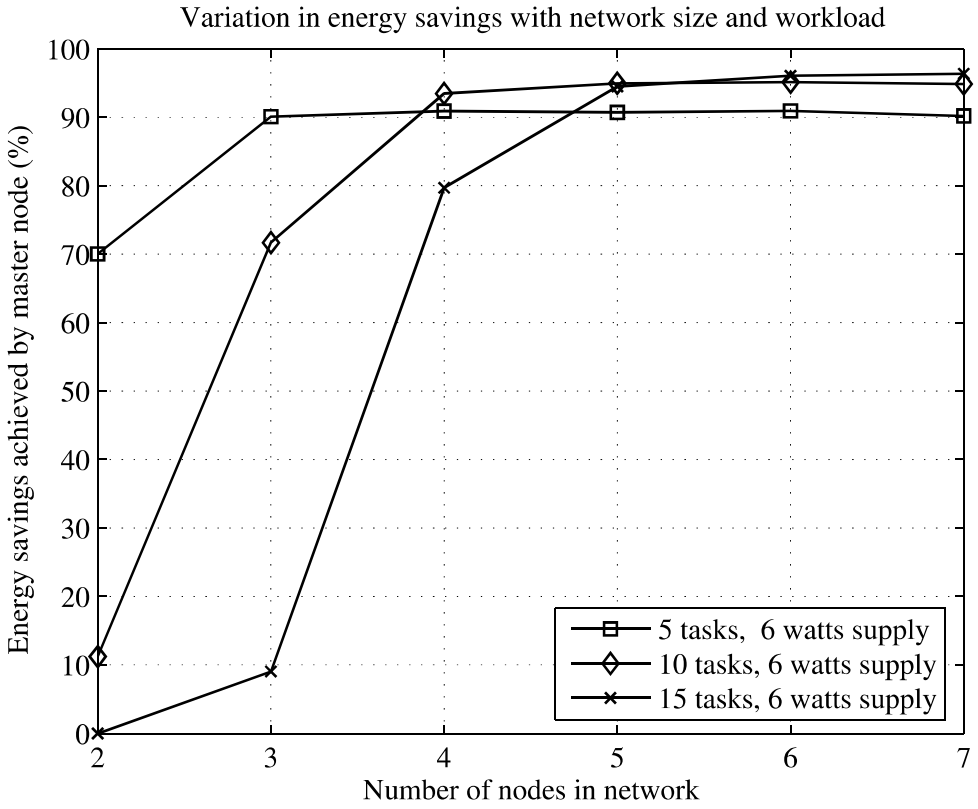


Figure 5.7: Master node's energy savings achieved by workload distribution for various network sizes, computational workloads, and power supply values when the application comprises independent tasks.

sidering P_{cp} as a variable, the expression for percentage savings evaluates to the form given by $1 - (constant/P_{cp})$.

Scalability with network size First, preliminary results are presented for simple round-robin workload allocation in an environment where there is no diversity in computational processing and available power resources. Power supply is fixed to a deterministic value of 10 watts across all nodes in the network. The power consumption per task is fixed to 0.5 watts for all tasks and the execution time of all tasks is fixed to 1 second regardless of which node processes the tasks indicating that there is no computational diversity in the operating environment. The application is structured based on task graph model 1 and comprises

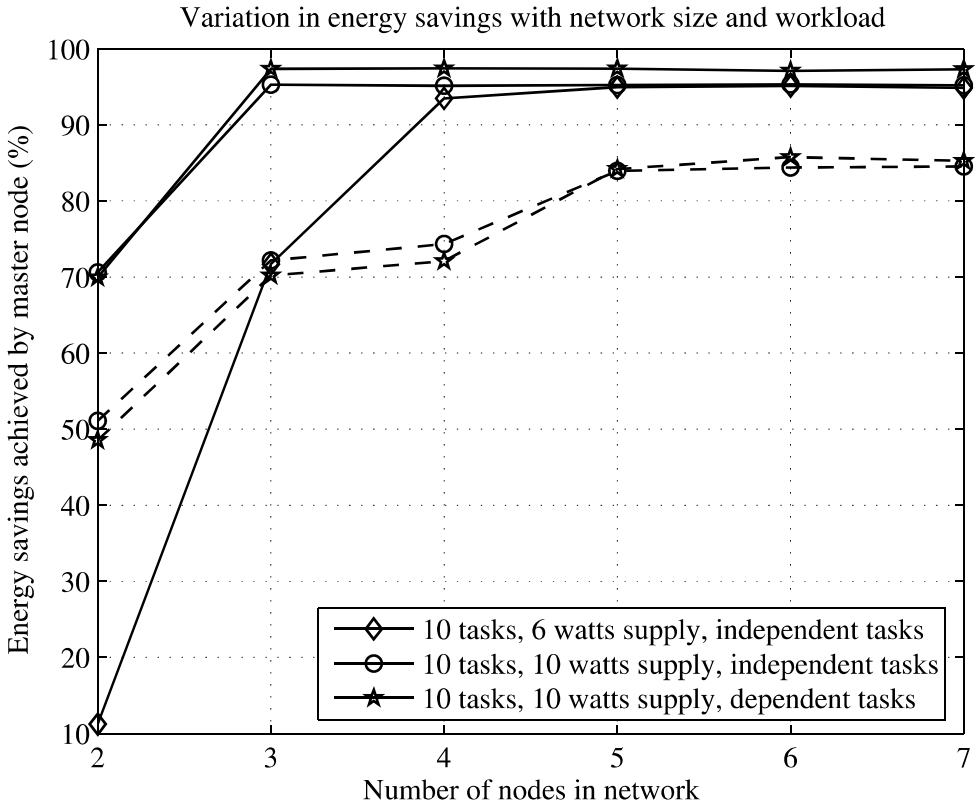


Figure 5.8: Master node's energy savings achieved by workload distribution for various network sizes, computational workloads, power supply values, and application task graph models. Plots associated with round-robin workload allocation are shown with dotted lines.

10 tasks. The nodes are assumed to operate per the adaptive transmit power model. The amount of application data transferred between dependent tasks in the task graph is set to 8000 bits before applying channel coding. The master node's energy savings is plotted for different network sizes and network ranges as shown in Figure 5.6. The savings improve with an increase in the number of collaborating nodes due to decreased computational workload at the master node as it sheds away more load to the slave nodes. However, it is observed that the savings do not increase significantly beyond a certain network size, i.e., the savings encounter the "law of diminishing returns". This occurs because of the corresponding increase in the communication overhead imposed by the transmission (and reception) of data (and processing results) to (and from) slave nodes. Furthermore, the decrease in energy savings

with distance is attributed to the increase in transmitter power consumption at the master node. The non-linear variation in savings with range is mainly attributed to the presence of the path loss exponent in the transmitter power consumption model. The step-wise trend in the curves is due to the inherent nature of the round-robin approach where an increase in the network size, while keeping the total workload fixed, does not always reduce the number of tasks processed by the master node.

The master node's energy savings metric is plotted in Figure 5.7 for different network sizes, different computational workloads, and different power supply values when the proposed workload allocation algorithm is employed in a heterogeneous environment. Power supply is fixed to a deterministic value across all nodes in the network. The power consumption per task is fixed to 2 watts for all tasks and the execution time range of all tasks except the source and sink tasks is set to 20 seconds. The network spans across a radius of 50 meters and the nodes operate per the adaptive transmit power model. Figure 5.7 shows the minimum number of nodes, denoted by N_{min}, required for minimizing the master node's energy consumption for different computational workloads and different power supply values. Both task graph model 1 and model 3 are employed in the simulation analysis. In the case of model 3, the number of edges between tasks other than the sink and source tasks is set to 25.

It is observed that the energy savings curves reach a peak savings value for network size N_{min} and flatten thereafter. The master node achieves maximum savings by workload distribution when $N_{min} - 1$ slave nodes are available to share its workload and the slave nodes possess sufficient resources to process the delegated computational tasks and transmit the results to the master node. As the number of nodes in the network is increased beyond the minimum required number of nodes, N_{min} nodes are utilized to process the tasks while the remaining nodes remain unused. Thus, increasing the number of nodes beyond N_{min} does not impact the master node's savings. While keeping the power supply fixed and varying the application's computational workload, it is noticed that the curves converge to a maximum for different network sizes. As observed from Figure 5.7, it requires a minimum of 5 nodes

to achieve the maximum energy savings for an application that comprises 15 tasks while it just takes 3 nodes to achieve the maximum savings for an application with 5 tasks. It is also observed from Figure 5.8 that round-robin allocation follows a similar trend but, as expected, it does not achieve high energy savings. Since round-robin allocation does not apply any power constraints, it is fair to compare its performance with the proposed workload allocation algorithm only when the power constraints applied to the proposed algorithm are relaxed. The round-robin approach performs poorly compared with the proposed algorithm as observed for an application comprising 10 tasks and a 10 watt power supply.

Figure 5.8 shows the scalability of the master node's energy savings with network size when the power supply constraints are relaxed. Increasing the power supply per node results in a relaxation of resource constraints in all nodes and, in turn, reduces the minimum required number of nodes needed to execute the application with maximum energy savings. This trend is observed by comparing the savings achieved for an application comprising 10 tasks when employing a power supply of 6 watts and 10 watts, respectively. When using a 10 watt supply, the curve converges to the maximum savings value for just 3 nodes while it takes a network of 4 nodes to achieve the same when using a 6 watt supply. It is also observed from Figure 5.8 that the efficiency achieved by workload distribution is not significantly impacted by switching between task graph models 1 and 3. In task graph model 3, the dependency between tasks is randomized, which in turn can potentially reduce the amount of interaction between the tasks in the task graph.

Scalability with network range In this analysis, all nodes employ the fixed transmit power model and the transmitter signal power-to-noise power ratio is set to 135 dBm (transmitted signal power of 15 dBm). The adaptive transmit power model is not used for this simulation analysis since it does not show clearly the scalability of savings with cluster radius. The number of edges is set to 25 when simulation scenario 3 is run using task graph model 3.

Table 5.1: Simulation parameters employed for the analysis of scalability of energy savings with network range.

Simulation scenario	Range of T_{cp} (secs)	P_{cp} (watts)	Range of P_{supply} (watts)	Workload	Network size
1	20	2	6	10	3
2	5	0.5	10	5	4
3	5	0.5	10	10	4
4	5	0.5	10	5	10

Figure 5.9 illustrates the scalability of the master node's energy savings with network range for 4 different simulation scenarios which are presented in Table 5.1. Simulation scenario 1 represents an extreme scenario involving an application with high complexity (i.e., high power requirements per task, long execution time range and high workload) and a small-sized resource-constrained network with low power supply per node. Simulation scenarios 2 and 3 represent scenarios involving an application with low complexity tasks and a network with abundant power supply per node. Scenarios 2 and 3 differ only in the application workload. Simulation scenario 4 has the same parameters as that of scenario 2 but differs only in the network size. The following trends are observed from the plots.

In a long range network comprising a small number of nodes, among the N_{min} slave nodes required for the optimal distribution of the workload, a significant number of slave nodes are located at a large distance from the master node. It takes high communication latency and communication power consumption to communicate with nodes that are located at a large distance from the transmitter. Hence, energy savings achieved by the master node degrade as the network range increases, particularly for small networks.

The following observation is made when comparing the simulation results obtained for simulation scenario 1 with those obtained for the other scenarios. The master node achieves higher energy savings in scenario 1 for short network ranges of up to 200 meters. This is because an application with a long task execution time range of 20 seconds achieves higher

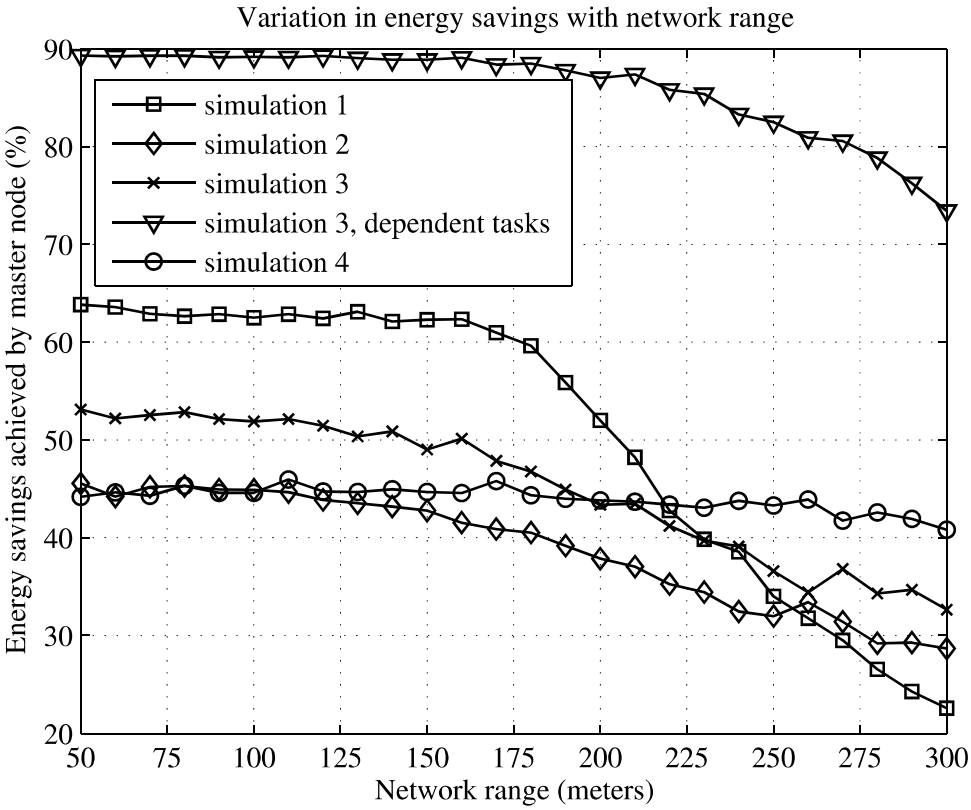


Figure 5.9: Master node's energy savings achieved by workload distribution for different network ranges and network sizes.

savings compared to an application with a shorter execution time range of 5 seconds. This trend is also evident from the results plotted in Figure 5.11 which are discussed later in this section. As the range increases, however, the savings degrade more rapidly in scenario 1 than in the case of the other simulation scenarios due to the following reason. Due to the severe network resource constraints in simulation scenario 1, the master node is discouraged from workload allocation for large network ranges where the communication cost of WDC workload distribution can be significantly expensive. In addition, the location of slave nodes in a small sized network can vary drastically as the cluster radius is increased, particularly when employing a binomial point process model.

Next, the variation in energy savings with application workload is analyzed by comparing the results obtained for simulation scenarios 2 and 3. When there are sufficient number

of nodes with sufficient resources in the network, an increase in the applications workload improves the master node's energy savings. This trend is explained later in this section when analyzing the variation of energy savings with task execution time range and workload in Figure 5.11.

The next observed trend pertains to the difference in the rate of degradation of energy savings with network range when the network size is increased from 4 to 10 nodes, as observed by comparing the results obtained for scenarios 2 and 4 in Figure 5.9. In large sized networks, the master node's energy savings are not significantly impacted by an expansion of the network range. A large network size provides greater diversity in several aspects, namely, inter-node distances, task execution time characteristics, and availability of power supply resources. It leads to increased network density, which in turn results in more number of nodes located within favorable distances from the master node, which directly impacts the master node's communication energy consumption. Hence, an increase in the network range does not significantly degrade the master node's savings. Simulation scenarios 2 and 4 yield the same savings for small network ranges since the master node's communication energy consumption does not differ significantly between the two scenarios when the network cluster radius is small.

When comparing the results achieved in simulation scenario 3 for task graph models 1 and 3, it is observed that task graph model 3 yields higher savings compared to task graph model 1. This is because of the inherent difference between task graph models 1 and 3. In task graph model 3, the amount of communication between tasks, and, in turn, the amount of communication between nodes is randomized. In task graph model 1, on the other hand, the amount of communication is fixed. Thus, the master node's communication energy consumption is lesser when task graph model 3 is employed.

Scalability with power supply availability The variation in the master node's energy savings with power supply range for different application workloads and network sizes is

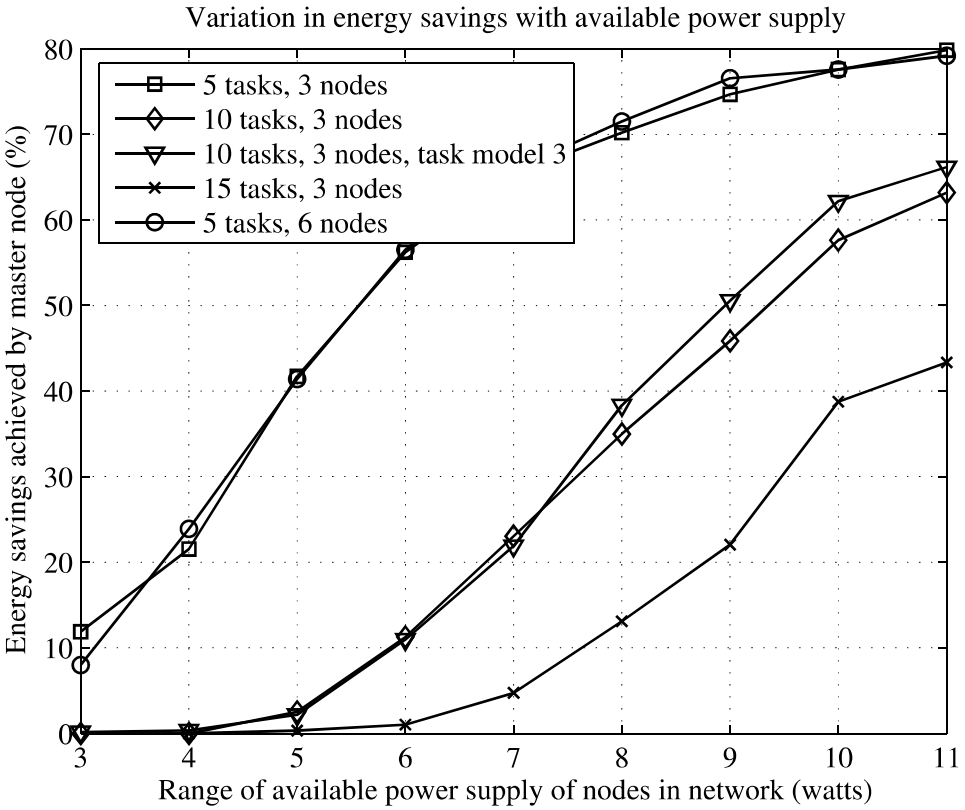


Figure 5.10: Master node's energy savings achieved by workload distribution for various ranges of power supply values, computational workloads, and network sizes.

shown in Figure 5.10. The simulation assumptions are as follows. The power consumption per task is fixed to 2 watts for all tasks. The execution time range of all tasks, except the source and sink tasks, is set to 20 seconds. The network spans across a radius of 90 meters and all nodes operate per the adaptive transmit power model.

The trends observed in Figure 5.10 infer non-linear scalability of energy savings with available power supply. The master node achieves zero savings for large application workloads when there is limited available power supply in the network. The savings curve plotted for a workload of 15 tasks and a network size of 3 nodes, for instance, exhibits this trend. In other words, the master node's energy efficiency degrades with an increase in application workload when there are tight power supply resource constraints throughout the network. This is because the master node cannot delegate its workload to the slave nodes when the power

resources in the network are limited. The efficiency, however, improves with an increase in the power supply range beyond a certain minimum value. The value of the minimum required power supply range is, for instance, 4 watts when the application comprises 10 tasks. Furthermore, the curves flatten beyond a certain maximum power supply range when the nodes have sufficient power resources to process the workload. Once the required power resources becomes available, any further increase in power supply does not help improve the master node's savings. The values of the minimum required power supply range and the maximum useful power supply range increase with an increase in the workload because of the increased power supply resource requirements of large workloads. It is also observed that the slope of a energy savings curve is larger for large application workloads. A 20% and 40% increase in savings is observed when the power supply range is varied from 6 watts to 10 watts for workloads of 5 and 15 tasks, respectively.

The maximum possible savings, which is achieved at the point where the curves begin to flatten, is lesser for large workloads (15 tasks) compared to smaller workloads (5 tasks) even when there is abundant power supply (a power supply range of 11 watts). This trend simply exhibits the inherent limitation in the maximum savings that can be achieved for a certain network size as exhibited by Figure 5.7. An increase in the number of nodes does not impact savings for an application with a relatively small workload. This trend, which is also backed by the analysis of Figure 5.7, is observed by comparing the curves pertaining to a workload of 5 tasks.

Scalability with task execution time range and application workload Figure 5.11 shows the scalability of the master node's energy savings with the task execution time range and application workload. The power supply is fixed to 10 watts for all nodes in the network. The power consumption per task is fixed to 2 watts for all tasks. The network spans across a radius of 50 meters and all nodes operate per the adaptive transmit power model. The task graph model 2 is employed in this analysis.

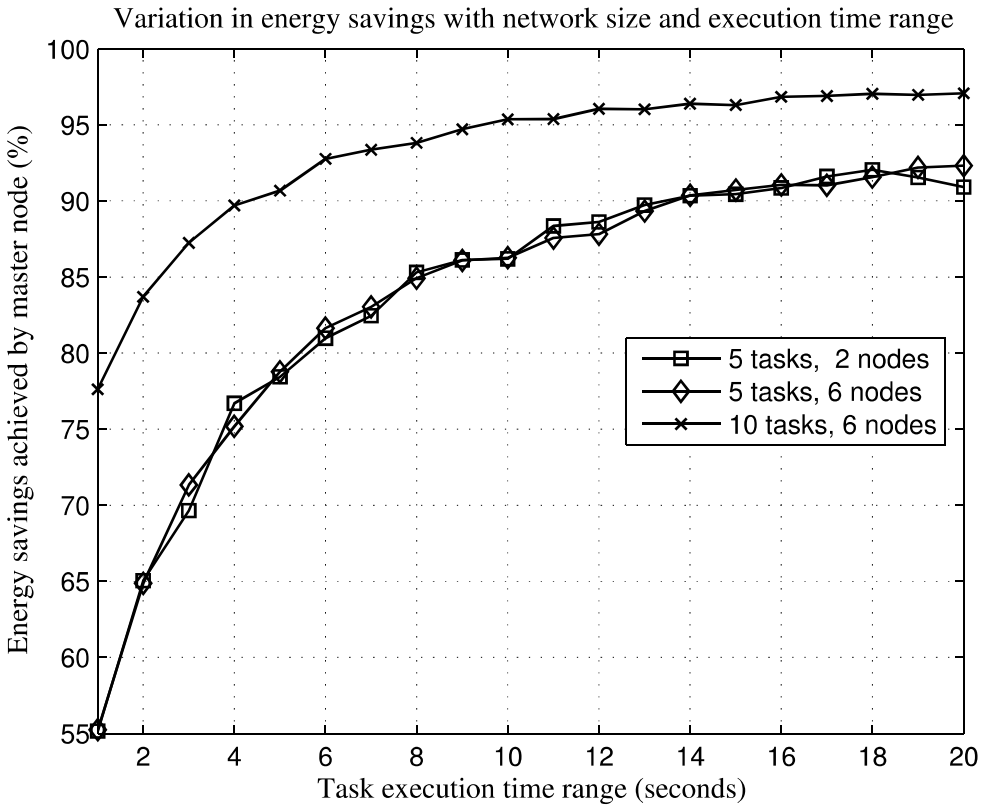


Figure 5.11: Master node's energy savings achieved by workload distribution for different network sizes and ranges of task execution time.

It is observed that the master node benefits more from workload distribution for large application workloads and long task execution time ranges. A large workload gives the master node larger scope of workload distribution, and hence, a larger percentage improvement in energy consumption. As observed in the analysis of scalability of energy savings with computational power consumption per task presented earlier in this section, the curve flattens beyond a certain value of task execution time range and the savings encounters a "law of diminishing returns".

5.5 Proposed Task Allocation - Scheduling Algorithm

The proposed task allocation - scheduling algorithm is presented in this section. The algorithm allocates tasks to participating nodes in the network and temporally schedules the communication and computational activities associated with the distributed execution of the application onto available computing and communication resources. The algorithm is designed based on list-scheduling. The original list scheduling algorithm, which is proposed by Hu [55], is a heuristic algorithm which iteratively allocates tasks that are retrieved from a priority queue to processors and updates the availability of the processors accordingly. Tasks are sorted in decreasing order of their priority in the priority queue. Task precedence constraints are taken into consideration in the case of applications comprising dependent tasks. Criteria are used to set priority to tasks and to choose the right processor for allocation of a task. The original list scheduling algorithm cannot be directly used for WDC workload distribution since it does not take into consideration the amount of communication between computational tasks of an application [55].

A related level-based list scheduling algorithm is presented in Alsalih [52], where a task's level reflects the precedence/dependency constraints on the execution of that task. The level of a task is important in resolving contention for resources, particularly when the resources are limited. In Alsalih [52], scheduling is performed in two stages: first, using an estimated measure of a task's level and, next, using an actual level. Accordingly, two metrics are used to order the scheduling of the tasks in a task graph, namely, an estimated level and an exact level. The estimated level of a task can be computed as the sum of the minimum execution times of all tasks that lie along the longest path from the task to a sink task in the task graph. The exact level is the total cost which includes the execution time and communication delays of all tasks that lie along the longest path from the task to a sink task, where the costs are computed with consideration of the task allocations. The communication delay between two tasks is computed based on the link delay between nodes to which the interacting tasks have been assigned. Based on the estimated level of a task, the algorithm tends to assign tasks

to nodes that provide maximum energy and time efficiency of execution, thereby minimizing the total cost of execution of the application. In scenarios where the communication cost is significantly smaller than the computational cost, the estimated level serves as a reasonable prediction of the actual level.

The level-based list scheduling algorithm improves over the original list scheduling algorithm for applications comprising dependent tasks operating in heterogeneous environments. It, however, does not schedule the communication activities of a task graph since it assumes dedicated links between nodes. The proposed algorithm, on the other hand, schedules communication activities in a task graph onto $N_{channels}$ communication channels in addition to scheduling the computational tasks. Further, the proposed algorithm comprises an offline portion and an online portion. The topological sort of the application's task graph and the actual task allocation - scheduling are performed separately. This separation of the two procedures allows the topological sort to be performed offline while the actual task allocation - scheduling is performed online based on current environmental conditions. The rest of this section is organized as follows. Section 5.5.1 presents the problem formulation. Section 5.5.2 describes the proposed algorithm. The performance of the algorithm is analyzed using simulation and experimental results presented in Sections 5.5.3 and 5.5.4, respectively.

5.5.1 Problem Formulation

Let F_p represent the sink task in the application's task graph and T^p_{cpstp} denote the scheduled finish time of its execution. The goal of the algorithm is to allocate tasks to nodes and schedule the computation and communication activities in the task graph such that it minimizes the application's makespan, i.e., the finish time of the sink task in the application's task graph, and the total energy consumed for the distributed execution of the application, as

given by

$$\begin{aligned} & \min \left\{ w_1 \, T^p_{cpstp} + w_2 \, E_{nw} \right\}, \qquad (5.15) \\ \text{where} \quad & w_1 + w_2 = 1; \; 0 \leq w_1, w_2 \leq 1. \end{aligned}$$

In Equation 5.15, w_1 and w_2 are weights assigned, respectively, to the individual goals of minimizing the application's makespan and total energy consumption in the network. In order to achieve this goal, the algorithm minimizes the objective for the allocation of every task in the task graph. Consider the allocation of task F_i to a node N_j. The resulting finish time of F_i and total energy consumption are denoted, respectively, by FT^{ij}_{cp} and E^{ij}_T. The total energy consumption takes into consideration the computational and communication energy consumption incurred as a result of the task allocation under consideration. The objective being optimized by the algorithm, denoted by O_{ij}, is given by

$$\begin{aligned} O_{ij} &= w_1 \, FT^{ij}_{cp} + w_2 \, E^{ij}_T, \qquad (5.16) \\ \text{where } w_1 + w_2 &= 1; \; 0 \leq w_1, w_2 \leq 1. \end{aligned}$$

The constraints of this problem are identical to the constraints presented in Section 5.4.1.

5.5.2 Algorithm Description

The proposed algorithm maintains timestamps on resources in order to impose the exclusive access constraint. The exclusive access constraint ensures exclusive utilization of the computational and communication resources in the network, wherein a resource can be held for only one activity at any time. Timestamps are an indication of the readiness of resources. By incrementing the timestamp of readiness of a resource after scheduling an activity onto the resource, the algorithm ensures that the computational (or communication) activities on a particular node (or channel) are serialized and operate in accordance with the exclusive access constraint.

Every activity has a ready time and an actual start time. The ready time of an activity indicates the earliest time when the activity is ready to be initiated regardless of the availability of resources required to execute that activity, while its actual start time is the time when the activity is started. The activity is started when both the task is ready and the resources required to execute the task are available. For instance, when the computational processor required to execute a task is not available at the ready time of the task, the task can be executed at the earliest time only when the processor becomes available. In other words, the earliest start time of a task is given by the maximum among its ready time and the ready time of its required resource.

Among the various resources available for a particular task, a resource is considered "best" for a task when it can most efficiently enable the execution of that task. A particular resource may be best for more than one task. As resources get scheduled with activities, those activities that are scheduled later are not left with many options as the resources become unavailable. Due to the exclusive access constraint, limited availability of resources, and the serial manner in which tasks are scheduled onto resources, there will be a question of what order should the tasks be assigned the "best" available resources. This issue is resolved by assigning priorities to tasks based on their estimated level. The estimated level is an indication of the order in which the tasks in a task graph have to be assigned available resources; i.e., it is a task's assignment priority. It directly influences allocation of tasks to nodes and scheduling of their activities by allowing high priority tasks to be allocated first to the "best" among available resources while the lower priority tasks have to wait for their turn and wait until their best resources become available.

The following assumptions are made by the proposed algorithm: (1) communication and computational activities inside a node do not share any resources, except the energy resources, so that the ready times of communication and computational resources inside the node are treated separately; (2) channels are of fixed bandwidth and data rates, hence, the communication latency is restricted by the channel bandwidth; (3) a communication activity can utilize at most one channel at a time; (4) all channels between a given transmitter-receiver

pair exhibit the same signal power-to-noise power ratio (SNR), although they would be different for different transmitter-receiver pairs as a result of the different inter-node distances, path loss, and channel characteristics; and (5) packet size is kept constant. The algorithm also relies on the fact that there will be at least one source task in any directed acyclic graph (DAG) which holds the maximum level among all tasks in the task graph. This claim can be proved by contradiction [56]. It is known that a DAG has a finite number of tasks. In order to contradict the statement, an arbitrary task in the task graph should have at least one predecessor, which in turn should have at least one predecessor. In this manner, there could be an unlimited number of predecessor tasks unless one of the predecessors eventually returns to a task that has been previously visited. This indicates that there would be a cycle, which contradicts the fundamental assumption that the task graph is acyclic. Hence, the contradiction is wrong and this proves the claim.

The scheduling parameters that are configured by the algorithm are as follows. Timestamps on computing and communication resources in node N_j are denoted, respectively, by TS^j_{cp} and TS^j_{cm}. The timestamp on channel r is denoted by TS^r_{ch}. The ready time of task F_i when executed in node N_j is denoted by RT^{ij}_{cp} and the ready time of a communication activity originating from task F_i is denoted by RT^i_{cm}. The start and stop times of task F_i are denoted, respectively, by T^i_{cpstr} and T^i_{cpstp}. The start and stop times of communication originating from task F_l to task F_i are denoted, respectively, by T^{li}_{cmstr} and T^{li}_{cmstp}. Decision variables x_{ij} and y_{lir} denote, respectively, the allocation of task F_i to node N_j, and the allocation of the communication activity originating from F_l to F_i to channel r. The following are scheduling variables that are used when evaluating the allocation of task F_i to a node N_j. The start and finish times of F_i are denoted, respectively, by ST^{ij}_{cp} and FT^{ij}_{cp}. The start and finish times of communication from a task F_k to the task F_i are denoted, respectively, by ST^{ijk}_{cm} and FT^{ijk}_{cm}. The timestamp on channel r and the communication resources of node N_s are stored, respectively, by variables TS^{ijr}_{cht} and TS^{ijs}_{cmt}. In addition to these variables, additional variables may be used in the algorithm to track the times when a particular resource is scheduled with an activity which eventually can be used to compute the utilization of that

Algorithm 2 Proposed Task Allocation - Scheduling Algorithm.

1: Initialize all scheduling parameters and variables to zero. Initialize $\acute{F} = F$.
2: Compute the estimated level of all tasks in the task graph using Algorithm 2c and assign priorities based on the estimated level.
3: Append source tasks to queue Q_{cp}.
4: **while** $\left(\acute{F} \neq \{\phi\}\right)$ **do**
5: Sort tasks in Q_{cp} based on their estimated levels.
6: Remove the first element of Q_{cp}, denoted by F_i, i.e., $\acute{F} = \acute{F} - \{F_i\}$.
7: Determine the set of predecessors of F_i.
8: **for** $(\forall N_j \in N_G^i)$ **do**
9: Consider the potential allocation of task F_i to node N_j.
10: Schedule the communication activities between F_i and its predecessors using Algorithm 2a.
11: Schedule the execution of F_i in N_j using Algorithm 2b and compute its finish time FT_{cp}^{ij}.
12: Compute the total energy consumption E_T^{ij} incurred by the allocation.
13: Compute the objective O_{ij} using Equation 5.16.
14: **end for**
15: Assign the task F_i to a node N_k which minimizes the objective and ensures that all resource constraints are met. Set $x_{ik} = 1$.
16: Update the remaining energy supply in every node which is affected by this allocation.
17: Using Equations 5.26 - 5.28, set the start and finish times of all communication activities destined to F_i and update the timestamps of all resources utilized by the activities.
18: Set the start and finish times of F_i and update the timestamp of the computing resources of node N_k, as given by Equations 5.29 - 5.31.
19: Set the ready time of communication activities originating from F_i.
20: Add tasks, which become ready as a result of the execution of F_i in N_k, to Q_{cp}.
21: **end while**

resource. The priority queues which hold computational and communication activities that are ready to be scheduled are represented, respectively, by sets Q_{cp} and Q_{cm}. $\acute{F}$ denotes the set of all unscheduled tasks in the task graph. The set of immediate predecessors of F_i is defined as given by

$$Pred\{F_i\} = \{F_l : b_{li} > 0;\ l = 1,\ 2, \ldots,\ N_{tasks},\ l \neq i\}\,.$$

The proposed algorithm is presented in Algorithm 2. All scheduling parameters and variables are initialized to zero and $\acute{F}$ is set equal to F. The estimated level of each task in the task graph is computed using Algorithm 2c which is described later in this section. A task's priority is assigned based on its estimated level. Queue Q_{cp} is initially populated with the source task of the task graph. Nodes that do not have sufficient energy supply resources to support the execution of at least one task in the task graph are eliminated from consideration by the algorithm and the remaining nodes serve as candidate nodes for task allocations. Let N_G^i represent the set of candidate nodes in the network which qualify to execute a task F_i, where $N_G^i \subset N$. The source and sink tasks are allocated to the master node which is assumed to possess sufficient resources to support them at a minimum.

The algorithm operates iteratively until all tasks in the task graph are allocated to nodes. During every iteration, task F_i with the highest level in Q_{cp} is removed from the queue and evaluated for task allocation and scheduling. The algorithm identifies the set of immediate predecessors of F_i and the nodes that are assigned to them. The algorithm uses this information when scheduling communication activities as shown in Algorithm 2a. Let F_l denote a predecessor of F_i and N_m denote the node that is assigned to execute F_l.

Next, the algorithm evaluates the objective for each candidate node in the network that is considered for the execution of task F_i. When considering a candidate node N_j, the objective O_{ij} is computed using Equation 5.16. In order to compute the objective, the finish time FT_{cp}^{ij} and total energy consumption E_T^{ij} associated with the allocation are computed as follows. The computation of the finish time requires knowledge of the time when F_i is

ready to execute in N_j. Task F_i is ready to execute only after it receives messages from all its immediate predecessors. In order to compute FT_{cp}^{ij}, hence, all communication activities destined to F_i are first scheduled using Algorithm 2a which is described later in this section. Algorithm 2a configures the communication schedule parameters, namely, ST_{cm}^{ijl} and FT_{cm}^{ijl}. With knowledge of the task's ready time, ST_{cp}^{ij} and FT_{cp}^{ij} are computed using Algorithm 2b which is described later in this section. C_T^q denotes the total energy consumption in node N_q in order to play its role in the distributed execution of the application. E_{nodeT}^{ijq} denotes the total energy consumption in node N_q incurred as a result of the allocation of task F_i to node N_j. E_{nodeT}^{ijq} and, in turn, E_T^{ij} are computed as given by

$$E_{nodeT}^{ijq} = \left\{ \begin{array}{lll} E_{tx1} + \overline{E}_{txb}^{mj} \times \frac{b_{li}}{R_{code}} & ; & q = m,\ j \neq m \\ E_{cp}^{ij} + E_{rx1} + \left(\overline{E}_{rxb}^{mj} \times \frac{b_{li}}{R_{code}} \right) & ; & q = j,\ j \neq m \\ E_{cp}^{ij} & ; & q = j = m \\ 0 & ; & \text{otherwise} \end{array} \right\},\ \forall N_q \in N; \quad (5.17)$$

$$E_T^{ij} = \sum_{q=1}^{N_{nodes}} E_{nodeT}^{ijq}; \quad (5.18)$$

where, as mentioned earlier, l and m refer to the indices of a predecessor task and its assigned node, respectively. It is known a priori that the sink task, which is denoted by F_e and has an index e, is always allocated to the master node. This knowledge can be exploited in the following manner to enhance the algorithm's performance. In the case of a task that directly interacts with the sink task, the objective is computed by taking into consideration the costs associated with the task itself, as well as the costs associated with the interaction between the task and the sink task, as given by

$$O_{ij} = w_1\ \acute{FT}_{cm}^{ije} + w_2\ \acute{E}_T^{ij}, \quad (5.19)$$

$$\text{where } \acute{FT}_{cm}^{ije} = \max\left\{ FT_{cp}^{ij},\ TS_{cht}^{ija},\ TS_{cmt}^{ij1},\ TS_{cmt}^{ijj} \right\} + \left(\overline{T}_{bit}^{j1} \times \frac{b_{ie}}{R_{code}} \right) \quad (5.20)$$

$$\text{and } \acute{E}_T^{ij} = E_T^{ij} + E_{tx1} + E_{rx1} + \left(\left(\overline{E}_{txb}^{j1} + \overline{E}_{rxb}^{j1} \right) \times \frac{b_{ie}}{R_{code}} \right). \quad (5.21)$$

In Equation 5.20, it is assumed that channel a has the earliest timestamp among all the channels in the system. $\acute{FT}_{cm}^{ije}$ is the estimated finish time of the communication activity originating from task F_i and destined to the sink task given that task F_i is allocated to node N_j. It is treated as an estimate since it does not take into consideration the scheduling of all messages destined to the sink task. $\acute{E}_T^{ij}$ is an energy consumption term that takes into consideration the energy consumption costs incurred as a result of the execution of task F_i in node N_j, as well as the energy consumption incurred by interaction between task F_i and the sink task.

The objective is evaluated in this manner for all nodes that can potentially execute F_i. The algorithm chooses a node, denoted by N_k, which minimizes the objective as well as ensures that all resource constraints are met, as given by

$$O_{ik} \leq O_{ij}, \text{ where } k \neq j, \forall N_j \in N_G^i, \tag{5.22}$$

$$C_T^q + E_{nodeT}^{ikq} \leq E_{supply}^q, \forall N_q \in N. \tag{5.23}$$

The algorithm sets x_{ik}= 1 and updates the remaining energy supply and per node total energy consumption in all nodes affected by this allocation as given by

$$E_{supply}^q = E_{supply}^q - E_{nodeT}^{ikq}, \forall N_q \in N, \tag{5.24}$$

$$C_T^q = C_T^q + E_{nodeT}^{ikq}, \forall N_q \in N. \tag{5.25}$$

Let channel a be allocated to support the communication from F_l to F_i. The start and stop times of every communication activity destined to task F_i and the related resource timestamps are set as given by

$$T_{cmstr}^{li} = ST_{cm}^{ikl}, \tag{5.26}$$

$$T_{cmstp}^{li} = FT_{cm}^{ikl}, \tag{5.27}$$

$$\text{and } TS_{cm}^k = TS_{ch}^a = TS_{cm}^m = FT_{cm}^{ikl}. \tag{5.28}$$

Equations 5.26 - 5.28 are repeated for all predecessors of F_i and the respective nodes that they are assigned for execution. Next, the start and stop times of the execution of task F_i and and the related resource timestamps are set as given by

$$T^i_{cpstr} = ST^{ik}_{cp}, \tag{5.29}$$

$$T^i_{cpstp} = FT^{ik}_{cp}, \tag{5.30}$$

$$\text{and } TS^k_{cp} = FT^{ik}_{cp}. \tag{5.31}$$

Next, the algorithm sets the ready time of all communication activities originating from F_i which become ready after F_i finishes execution, as given by

$$RT^i_{cm} = FT^{ik}_{cp}. \tag{5.32}$$

All unscheduled tasks, which become ready after the allocation of a task to a node, are appended to Q_{cp}. At all times, queue Q_{cp} contains tasks whose predecessors, if any, are scheduled; i.e.,

$$Q_{cp} = \left\{ F_r : \text{ if } F_s \in Pred\{F_r\}, \text{ then } F_s \notin \acute{F} \right\}$$

, and the tasks in the queue are sorted according to their level.

Scheduling of communication activities When considering the allocation of F_i to N_j, all communications received by F_i from its predecessors can be scheduled as shown in Algorithm 2a. First, the following timestamp variables are initialized every time the algorithm is executed: $TS^{ijj}_{cmt} = TS^j_{cm}$, $TS^{ija}_{cht} = TS^a_{ch}$, and $TS^{ijm}_{cmt} = TS^m_{cm}$. Note that, these variables temporarily store the respective timestamps when evaluating the allocation of F_i to different nodes in the network. The actual timestamps are set after choosing a node that optimizes the objective. An estimated finish time is computed for each communication activity received by F_i from one of its predecessors. The computation of the estimated finish time is

Algorithm 2a Scheduling of communication activities destined to task F_i from its predecessors when F_i is allocated to N_j.

F_l and N_m, respectively, denote a predecessor of task F_i and the node to which it is assigned for execution.

1: **if** (F_i is a source task) **then**
2: There is no communication activity to be scheduled.
3: **return**
4: **end if**
5: Initialize the following timestamp variables: $TS^{ijj}_{cmt} = TS^{j}_{cm}$, $TS^{ijm}_{cmt} = TS^{m}_{cm}$, and $TS^{ijq}_{cht} = TS^{q}_{ch}$, where $q = 1, 2, \ldots, N_{channels}$.
6: Compute an estimated finish time of every communication activity destined to task F_i without consideration of channel timestamps.
7: Arrange the communication activities in increasing order of the estimated finish time in Q_{cm}.
8: **while** ($Q_{cm} \neq \{\phi\}$) **do**
9: Remove the first element of Q_{cm}, which is a communication activity that originates from F_l and is destined to F_i.
10: Identify the node that is assigned to execute F_l.
11: Select a channel, say channel a, with the earliest timestamp.
12: Compute the start and stop time of the communication activity using Equations 5.33 and 5.34.
13: Update the timestamps on communication resources utilized by the communication activity using Equation 5.35.
14: **end while**

similar to that of the actual finish time (see Equations 5.33 and 5.34) except that channel timestamps are not taken into consideration. The communication activities are placed in a communication queue Q_{cm} and the elements of the queue are arranged in the increasing order of the estimated finish times. The communication activity with the earliest estimated finish time is removed from queue Q_{cm} and is assigned a channel, say channel a, with the earliest timestamp. The start and stop times of the communication activity and the timestamps of resources utilized by the activity are computed as given by

$$ST^{ijl}_{cm} = \max\left\{RT^{l}_{cm},\ TS^{ija}_{cht},\ TS^{ijm}_{cmt},\ TS^{ijj}_{cmt}\right\}, \tag{5.33}$$

$$FT^{ijl}_{cm} = ST^{ijl}_{cm} + \left(\overline{T}^{mj}_{bit} \times \frac{b_{li}}{R_{code}}\right), \tag{5.34}$$

$$TS^{ijj}_{cmt} = TS^{ija}_{cht} = TS^{ijm}_{cmt} = FT^{ijl}_{cm}. \tag{5.35}$$

Algorithm 2b Computation of finish time of task F_i if allocated to node N_j.

1: **if** (F_i is a source task) **then**
2: Compute the start time of task F_i using Equation 5.36.
3: **else**
4: Compute the finish time of all communications received by F_i using Algorithm 2a.
5: Set the ready time of F_i using Equation 5.37.
6: Compute the start time of task F_i using Equation 5.38.
7: **end if**
8: Compute the finish time of task F_i using Equation 5.39.

The procedure is iteratively repeated for all communication activities queued in Q_{cm}. Scheduling is performed such that the total waiting time of communication activities destined to F_i is minimized. When two or more channels have the same timestamp, the one with the smallest channel ID in the channel resource pool is chosen. By doing so, it is intended to improve the utilization of channels and avoid sparse utilization of channels with higher IDs.

Computation of finish time of a task The finish time of a task is computed as shown in Algorithm 2b. The finish time is dependent on the task's ready time and the availability of computing resources in N_j. If F_i is a source task, the finish time is dependent only on the timestamp of computing resources in node N_j and is set as given by

$$ST_{cp}^{ij} = TS_{cp}^{j}. \tag{5.36}$$

A task F_i is said to be ready after it receives messages from all its predecessors. Communication activities which originate from the task's predecessors and destined to task F_i are scheduled using Algorithm 2a. Then, the finish time of the latest communication activity becomes the ready time of F_i, as given by

$$RT_{cp}^{ij} = \max_{\forall l} \left\{ FT_{cm}^{ijl} \right\} \text{ where } F_l \in Pred\left\{F_i\right\}. \tag{5.37}$$

Schedule F_i to start execution in node N_j after all messages are received from its predecessors and at the earliest available time of the computational resources in N_j, as given by

$$ST_{cp}^{ij} = \max\left\{RT_{cp}^{ij}, TS_{cp}^{j}\right\}, \tag{5.38}$$

$$FT_{cp}^{ij} = ST_{cp}^{ij} + T_{cp}^{ij}. \tag{5.39}$$

Computation of estimated level of tasks The estimated level of each task in a given task graph is computed using Algorithm 2c. Let U represent the set of all possible paths from the source task to the sink task of a task graph. The j^{th} path is represented by U_j which is an ordered set of tasks and is an element of U. Let U_{ij} represent a subset of U_j comprising tasks along the j^{th} path starting from task F_i down to the sink task. The possible paths in the task graph are identified. A metric S_{ij} is computed as the sum of minimum computational costs of all tasks in the j^{th} path starting from the task F_i till the sink task as given by

$$S_{ij} = \sum_{\forall F_x \in U_{ij}} \min_{\forall N_y \in N} T_{cp}^{xy}. \tag{5.40}$$

In Equation 5.40, the minimum execution cost of a task is determined by considering the node which can execute the task with least computational cost. S_{ij} is set to zero if task F_i is not present in U_j. The estimated level of task F_i is, then, computed as given by

$$EL_i = \max_{j=1,\ldots,|U|} \{S_{ij}\}. \tag{5.41}$$

5.5.3 Simulation Results

In this section, the performance of the task allocation - scheduling algorithm is analyzed for different network cluster ranges, task execution time ranges, network sizes, and workloads. The performance of the proposed algorithm is measured in terms of the following met-

Algorithm 2c Computation of estimated level of task F_i.

U represents the set of all possible paths from the source task to the sink task of a task graph. U_j represents the j^{th} element of U.

1: Identify all possible paths in the task graph and populate U.
2: **while** $(U \neq \{\phi\})$ **do**
3: Remove an element of U, say U_j.
4: Compute the sum of the minimum execution times of all tasks along the j^{th} path in the task graph starting from task F_i down to the sink task, as given by Equation 5.40.
5: Compute the estimated level of task F_i using Equation 5.41.
6: **end while**

rics: application makespan, percentage of total workload delegated to the slave nodes, and percentage improvement in application makespan achieved by workload distribution when compared with that achieved by local processing. In addition to the scalability analysis, the performance of the proposed algorithm is compared against round-robin task allocation - scheduling. All simulation results presented in this section consistently show that the proposed algorithm significantly outperforms the round-robin approach.

The following apply to all simulation results presented in this section. The algorithm is operated for minimization of both makespan and energy consumption, i.e., the weights w_1 and w_2 are each set to 0.5. All results presented in this section are simulated assuming a message size of 4000 bits for the interaction between the application's computational tasks when task graph models 1 and 2 are employed. In simulations which employ task graph model 3, the message sizes are randomly chosen from the range [1, 4000] bits. All nodes in the network operate per the fixed transmit power model. The complexity of an application refers to the number of tasks it comprises (or workload) and the execution time of each task. For instance, a highly complex application comprises a large number of tasks and a long execution time per task. In all plots presented in this section, a curve in dotted lines corresponds to the performance achieved by the system when round-robin allocation - scheduling is employed with the same simulation parameters as those employed to simulate the corresponding curve in solid line with the same data plot marker.

Task execution time range is an indication of the heterogeneity of the tasks in an application.

Table 5.2: Simulation parameters employed for the analysis of scalability of an application's makespan with network range.

Simulation	Task execution time range	Number of tasks	Task graph model
Simulation 1	2 secs	5	Independent tasks (task graph model 1)
Simulation 2	2 secs	15	Independent tasks (task graph model 1)
Simulation 3	10 secs	15	Independent tasks (task graph model 1)
Simulation 4	10 secs	15	Dependent tasks (task graph model 3)

A large range indicates that the tasks in the application can have execution times that vary over a wide range of values. High values of the application's workload size, the task execution time range, and network size, can increase computational diversity in the environment. Increased diversity in the environment is a strong driving factor for workload distribution. It is observed in the results that an increase in these system parameters encourages workload delegation and increases the percentage improvement in the application's makespan. In addition, an increase in diversity, particularly by increasing the network size, keeps the application's makespan from varying much with variation in other system parameters.

Scalability with network range In this analysis, four simulation scenarios are employed. The scenarios are presented in Table 5.2. The simulations involves a network of 20 nodes where each node transmits at a SNR of 130 dB (transmit signal power of 11 dBm). The power consumption per task is assumed to be 0.5 watts. The number of edges is set to 20 in simulations where task graph model 3 is employed. Figures 5.12, 5.13 and 5.14 show, respectively, the scalability of application makespan, percentage workload delegation and percentage improvement in the application's makespan, with network range.

The following trends are observed from the simulation results. When executed in a distributed manner, the makespan of a complex application can increase rapidly with network range. On the other hand, the makespan of an application comprising 5 tasks with a task

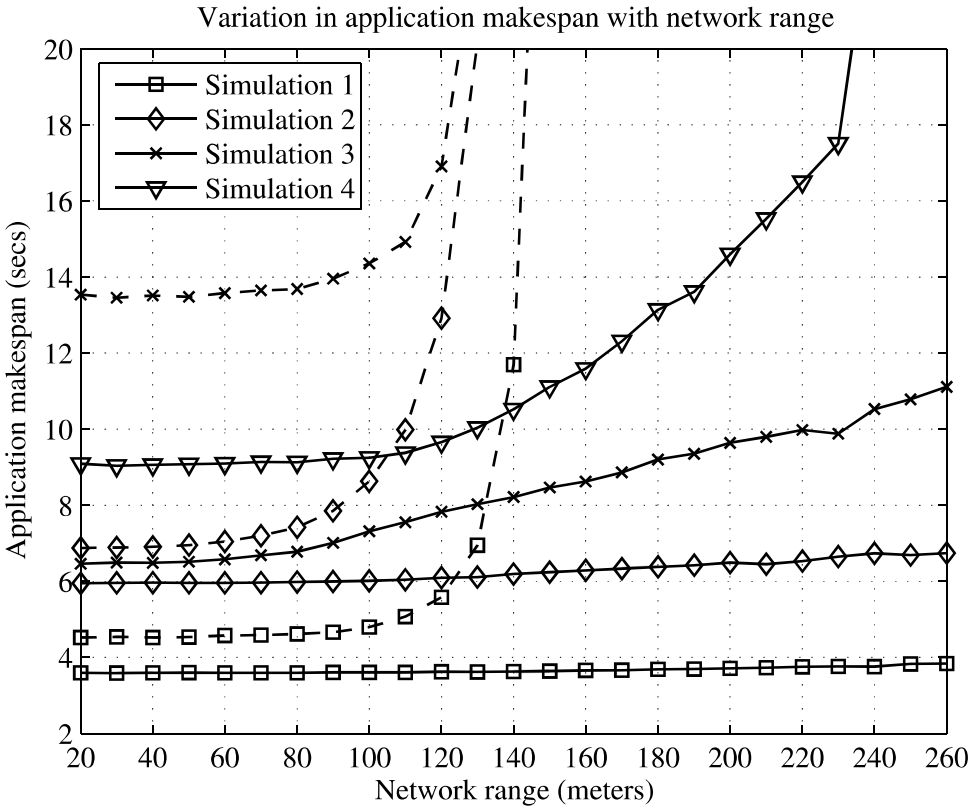


Figure 5.12: Variation in an application's makespan with network range. Curves in dotted lines correspond to the performance of round-robin allocation.

execution time range of 2 seconds remains nearly unaffected by variation in network range. The reason for this trend can be explained as follows. In the case of an application with a small workload of 5 tasks, the application can execute optimally with a small number of computational resources, hence the master node delegates a maximum of only 43 % of the total workload to the slave nodes as shown in Figure 5.13. In other words, there is less dependence on slave nodes and, in turn, less dependence on their locations. An application with a large workload, on the other hand, requires a larger number of nodes for its optimal execution. Due to the high dependence on the location of slave nodes and the increase in the inter-node distances with an increase in the network radius, the makespan of an application with a large workload is significantly impacted by variation in the network radius.

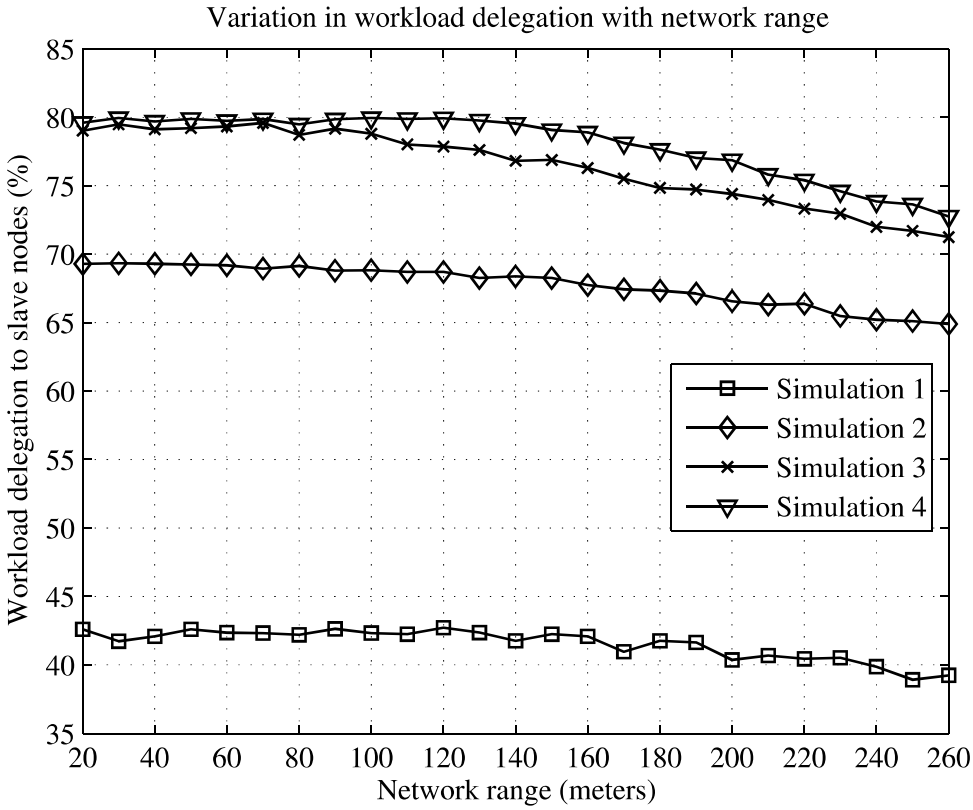


Figure 5.13: Percentage of total workload delegated to slave nodes for different network ranges.

An increase in the network range discourages workload delegation particularly for complex applications, as observed from Figure 5.13. In spite of the degradation of the makespan, applications with high complexity benefit more from workload distribution when compared to applications with lower complexity. This is observed by comparing the curves for simulation scenario 1 with those of scenarios 2 and 3 in Figure 5.14. The effect of task execution time range and workload on the performance of the algorithm is discussed later in this section.

In addition to the impact of application complexity, increased task dependency further degrades the application's makespan rapidly with an increase in the network range, which is evident by comparing the curves for simulation scenarios 3 and 4 in Figure 5.12. This is because of the increased communication latency incurred due to the interaction between tasks

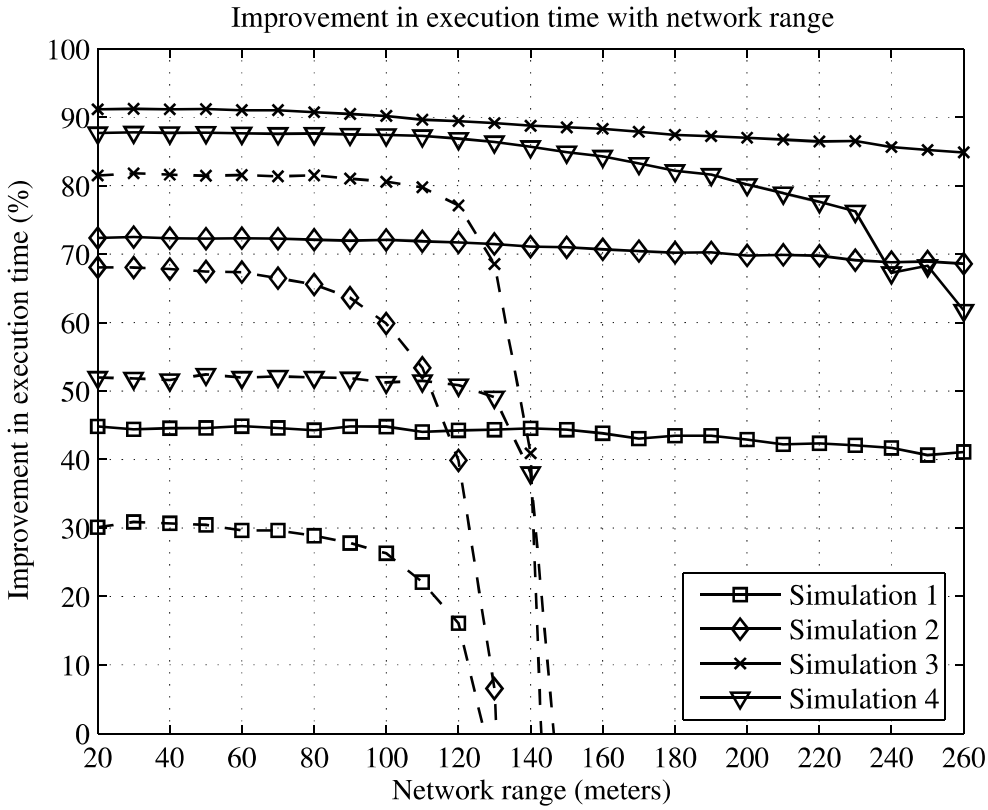


Figure 5.14: Improvement in an application's makespan achieved by WDC workload distribution for different network ranges.

allocated to different nodes and the increased wait time experienced by tasks which have to wait until all their respective predecessors have finished processing and have transmitted their results to the waiting tasks. The latter factor dominates when high transmit SNR is employed. Due to the potential increase in communication latency, workload delegation decreases rapidly with network range for simulation scenario 4, as observed from Figure 5.13. As a consequence, the percentage improvement in applications makespan degrades rapidly beyond a network range of 200 meters for simulation 4, as observed in Figure 5.14.

The proposed algorithm outperforms round-robin task allocation - scheduling both in terms of makespan and improvement in execution time. It achieves nearly linear scaling of application makespan with range while round-robin scheduling results in rapid and non-linear in-

Table 5.3: Simulation parameters employed for the analysis of scalability of an application's makespan with task execution time range.

Simulation	Network Size	Cluster Radius	Task Graph Model
Simulation 1	2 nodes	50 meters	Independent tasks (task graph model 1)
Simulation 2	5 nodes	50 meters	Independent tasks (task graph model 1)
Simulation 3	10 nodes	50 meters	Independent tasks (task graph model 1)
Simulation 4	10 nodes	200 meters	Independent tasks (task graph model 1)
Simulation 5	10 nodes	200 meters	Dependent tasks (task graph model 3)

crease in makespan with range, as observed from the curves with dotted lines in Figure 5.14. The reason behind this trend is explains as follows. In a round-robin task allocation approach, the application's workload is uniformly distributed among all nodes in the network, without consideration of their distance from the master node. In other words, the percentage workload delegation does not vary with network range when the round-robin algorithm is employed. On the other hand, the proposed algorithm only chooses nodes such that the objective is optimized.

Scalability with task execution time range The simulation analysis involves an application comprising 15 tasks, where the power consumption per task is 0.5 watts. The transmitted SNR is set to 135 dB (transmitted signal power of 15 dBm). 5 simulation scenarios are considered for the study and the parameters of the scenarios are presented in Table 5.3. Scenarios 1, 2, and 3 differ only in the network size (or network density). Simulation scenarios 3 and 4 differ in the cluster radius. Simulation 5 employs the same parameters as simulation 4 except that it uses task graph model 3. Figures 5.15 and 5.16 show, respectively, the scalability of application makespan and percentage improvement in the application's makespan, with task execution time range.

A task execution time range of 1 second indicates that all tasks have a fixed execution time of 1 second regardless of which node is used to execute them, i.e., the WCCN environment

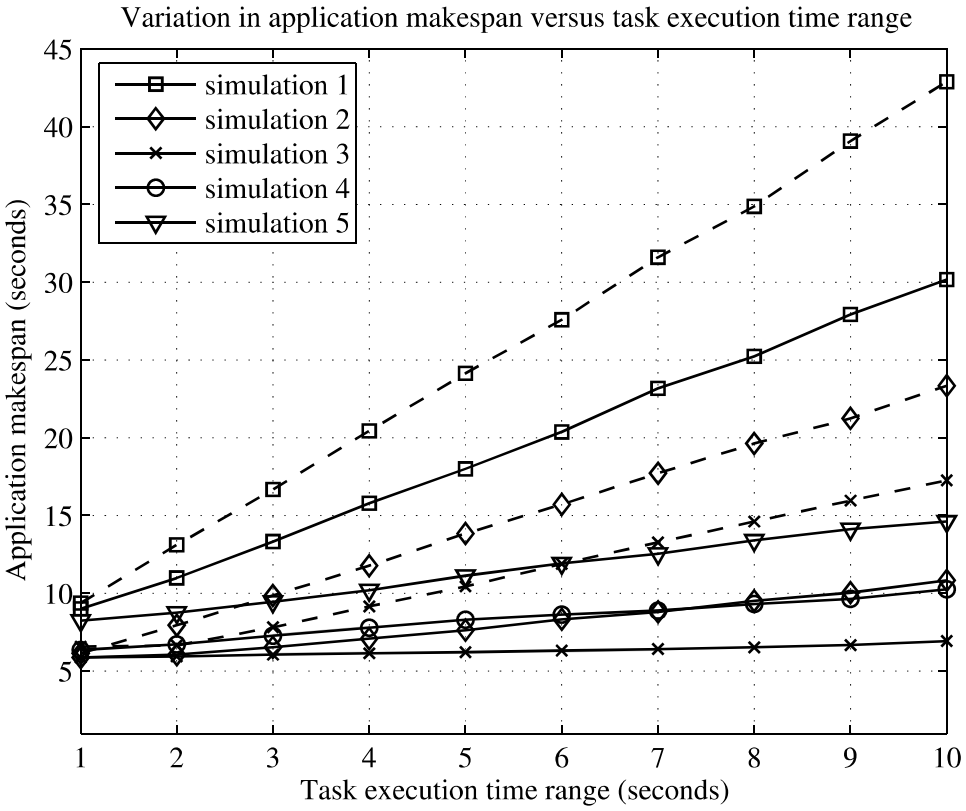


Figure 5.15: Scalability of an application's makespan with task execution time range. Curves in dotted lines correspond to the performance of round-robin allocation.

offers no computational diversity. Communication latency is short when the network radius is small and the transmitted signal power is set to a high value. When communication latency is short (scenario 1) and there is no computational diversity, the adaptive approach taken by the proposed algorithm does not provide any performance benefits over round-robin task allocation - scheduling. As computational diversity is increased by increasing the task execution time range in scenario 1, the performance curves corresponding to the two algorithms diverge from each other and the proposed algorithm provides better benefits. The makespan and percentage improvement curves are not shown for scenarios 4 and 5 when round-robin scheduling is employed because they result in very large values of makespan and no improvement in execution time.

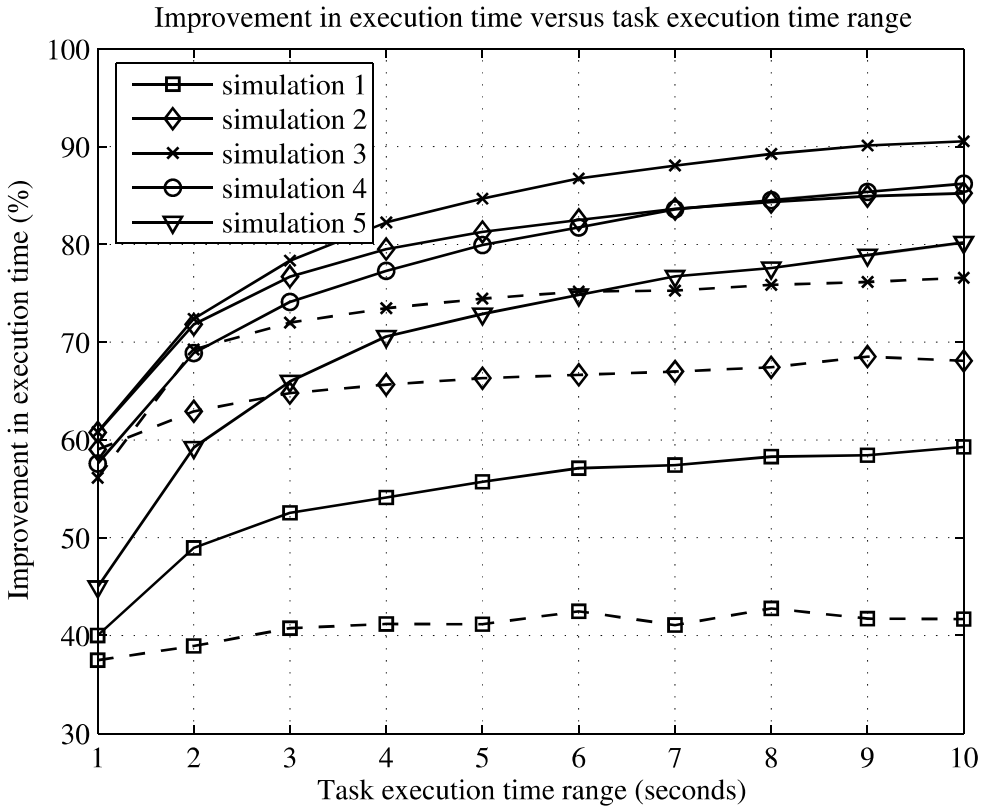


Figure 5.16: Improvement in an application's makespan achieved by WDC workload distribution for different task execution time ranges. Curves in dotted lines correspond to the performance of round-robin allocation.

As observed in Figure 5.15, there is a linear increase in application makespan with increase in task execution time range, particularly for a network with limited number of nodes. This occurs because an increase in the range of possible task execution times can potentially result in an increase in execution time of each task in the application and, in turn, an overall increase in the applications makespan. A higher improvement in the application's execution time is observed with an increase in the task execution time range, as shown in Figure 5.16, due to the following reasons. First, an increase in the task execution time range can potentially increase the task execution times. Applications comprising high complexity tasks tasks benefit more from the exploitation of concurrency by workload distribution compared to applications comprising low complexity tasks. Second, an increase in the range increases diversity in the

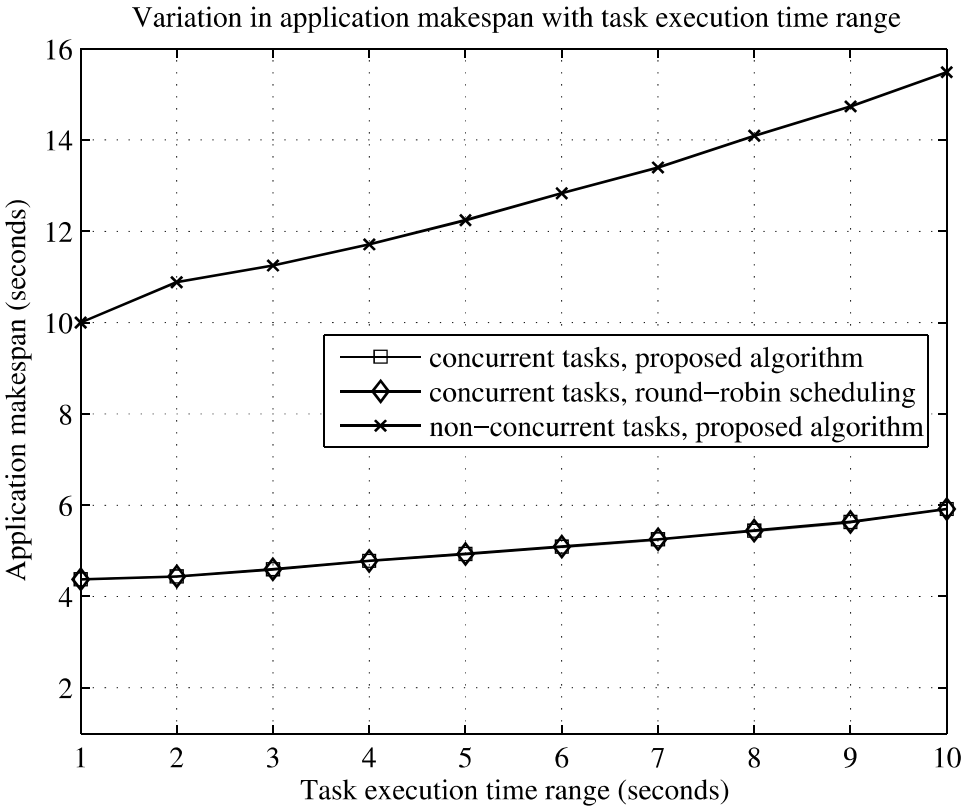


Figure 5.17: Scalability of an application's makespan with task execution time range.

execution characteristics of both the application's tasks and the computational platforms in nodes. WDC workload distribution exploits this diversity and provides greater gains in the application's execution time. The master node's computing platform may not always be efficient to execute all the application's tasks, while other nodes may be able to execute some of the tasks in significantly lesser time. The curves flatten beyond a task execution time range, which is expected as we take the percentage improvement over on-board processing.

The impact of increasing the network size along with an increase in the execution time range is analyzed by comparing the results obtained for simulations 1, 2, and 3. The slope of the makespan curve reduces as network size increases and, thereafter, the makespan becomes unaffected by the task execution time range for large network sizes. This trend results in greater execution time improvement, as observed from the results plotted in Figure 5.16.

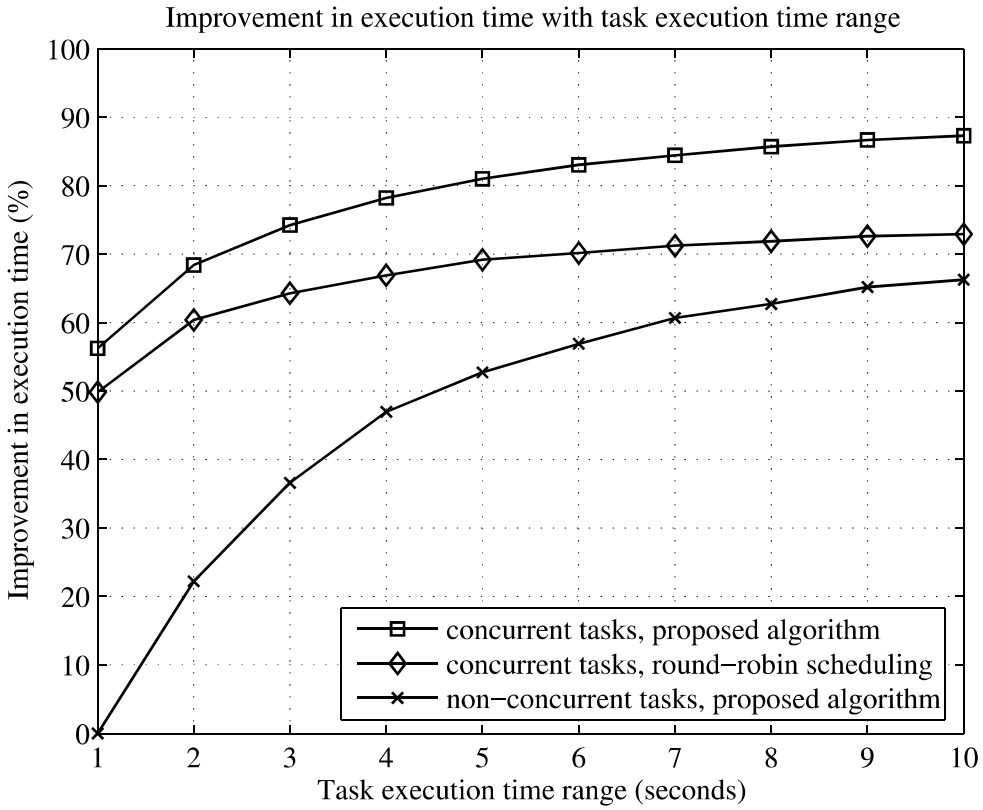


Figure 5.18: Scalability of percentage makespan improvement with task execution time range, for applications comprising concurrent and non-concurrent software structure.

The reason for this trend is discussed later in this section when analyzing the scalability of performance with network size.

When comparing the results for scenarios 3 and 4, it is observed that makespan increases with an increase in network radius and lower execution time improvement is achieved with a larger network radius. The increase in makespan, however, is small because of the large network size. Section 5.4.3 (see Figure 5.9) discusses the impact of large network sizes on the performance of workload distribution for different network ranges.

As expected, workload distribution of an application comprising dependent tasks (task graph model 3) results in a longer execution time compared to an application that is structured based on task graph model 1. This trend is evident by comparing the results from simulations

4 and 5. Next, Figures 5.17 and 5.18 compare the performance of an application which comprises concurrent tasks (based on task graph model 1) against that achieved by an application that exhibits no concurrency (based on task graph model 2). The following parameters are assumed for this simulation study. The network comprises 10 nodes located within a radius of 50 meters and the application comprises 10 tasks. The transmitted SNR is set to 135 dB (16 dB transmit signal power). An application that is structured based on task graph model 2 has no concurrency in its structure. In the absence of both concurrency and computational diversity, the proposed algorithm does not delegate any workload to the slave nodes, hence it provides no improvement over local processing. The application's makespan achieved by WDC does not increase much with an increase in the task execution time range (see Figure 5.17) while the makespan achieved by local processing increases rapidly due to lack of diversity and concurrency. This causes the percentage improvement in the application's makespan to increase rapidly as computational diversity increases. A round-robin approach to task allocation - scheduling does not provide any improvement in execution time when applied to an application which comprises non-concurrent tasks.

Scalability with application workload In this simulation analysis, the task execution time range is set to 20 seconds and the power consumption per task is set to 0.5 watts. The network is located within a radius of 100 meters and the transmitted SNR is set to 135 dB (16 dBm transmit signal power). Figures 5.19, 5.20 and 5.21 show, respectively, the scalability of application makespan, percentage workload delegation and percentage improvement in the application's makespan, with an application's workload.

In the case of an application comprising diverse tasks which are structured based on task graph model 1, as shown in Figure 5.21, an increase in the workload increases the percentage improvement in the application's makespan due to the following reasons. An increase in the workload can potentially increase computational diversity. In addition, it encourages workload delegation and, in turn, allows the exploitation of concurrency in the application's structure. It is also observed that the performance encounters the "law of diminishing

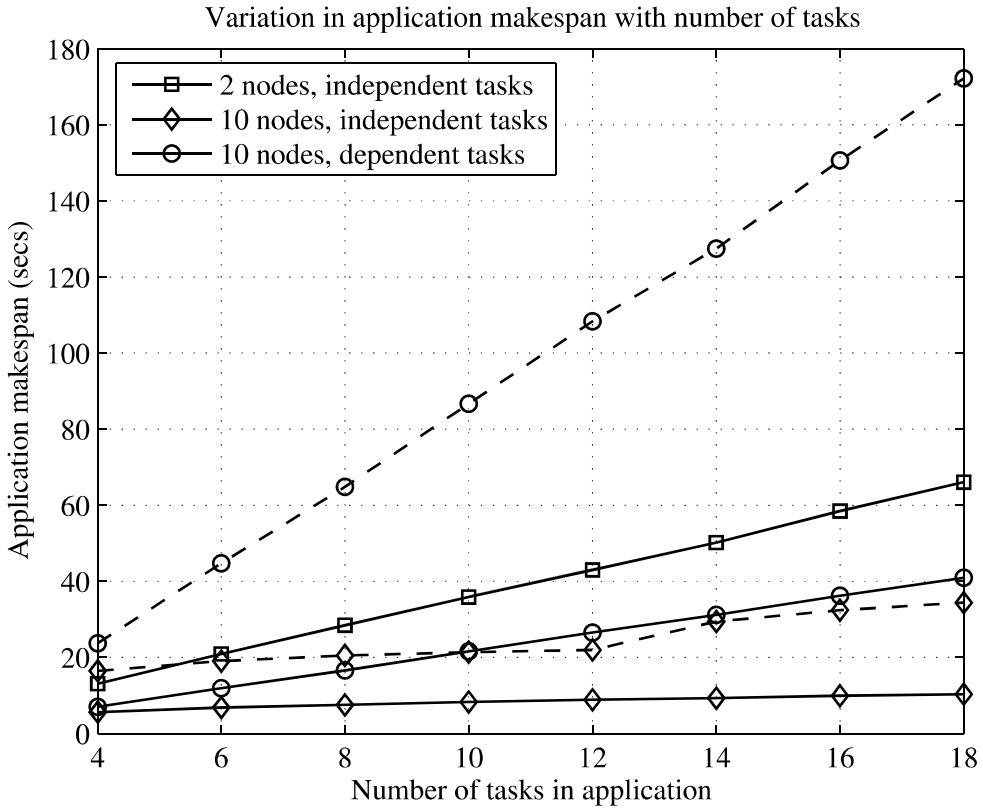


Figure 5.19: Scalability of an application's makespan with the application's workload.

returns" as the workload increases.

As observed in Figure 5.19, the impact of variation in an application's workload on the application's makespan is greatly reduced in the case of a larger network size of 10 nodes. A large network provides sufficient computational resources in order to handle a wide range of application workloads and provides greater computational diversity. Hence, large networks encourage workload delegation and, in turn, provide greater improvement in execution time. As observed from Figure 5.20, the workload delegation curve flattens for a network of 2 nodes as there are no sufficient slave node resources to handle the increasing workload.

When comparing the performance achieved by an application comprising dependent tasks (task graph model 3) with that achieved by an application comprising dependent tasks (task

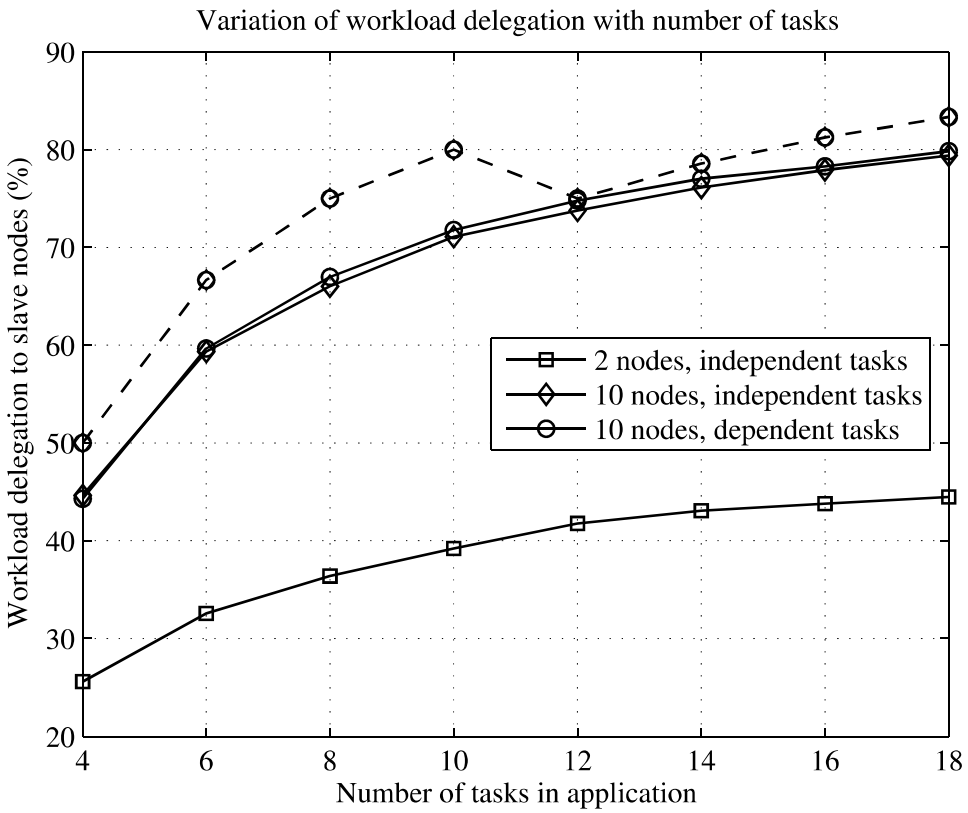


Figure 5.20: Variation in workload delegation for different application workloads.

graph model 1), it is observed that the former application experiences longer makespan while both applications achieve the same percentage workload delegation. This, in turn, results in a lower performance improvement for applications with dependent tasks.

As observed from Figure 5.20, workload delegation is lower for the proposed algorithm compared to round-robin task allocation - scheduling since the proposed algorithm delegates tasks adaptively and selectively, unlike the latter algorithm which delegates tasks to all slave nodes without any environmental awareness.

Scalability with network size In this simulation analysis, it is assumed that the application comprises 10 tasks and the power consumption per task is assumed to be 0.5 watts. The network radius is 100 meters and the transmitted SNR is set to 135 dB (16 dBm transmit

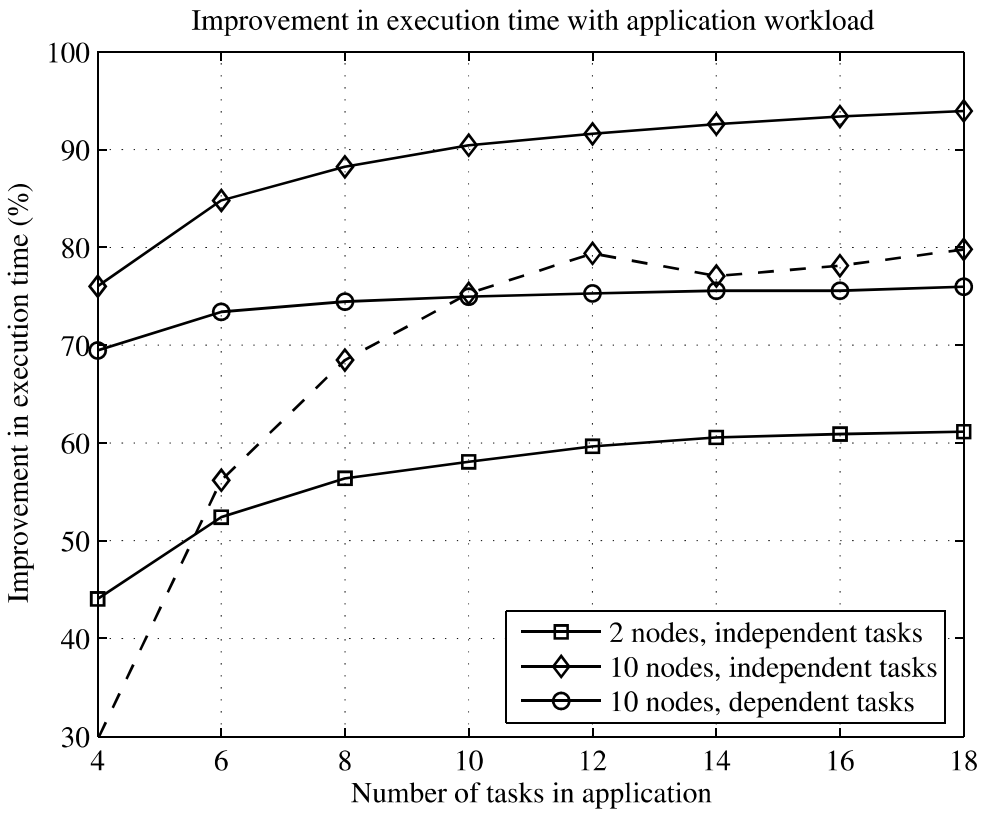


Figure 5.21: Improvement in an application's makespan achieved by workload distribution for different application workloads.

signal power). Figures 5.22, 5.23 and 5.24 show, respectively, the scalability of application makespan, percentage workload delegation and percentage improvement in the application's makespan, with network size.

An increase in the network size increases computational diversity, the number of nodes located close to the master node, and the amount of computational resources which can be utilized to exploit concurrency in the applications task graph. The proposed algorithm exploits these benefits of increasing the network size and, in turn, achieves a reduction in the application's makespan and an increase in the percentage improvement in the application's makespan.

The benefits of large sized networks are significant only when there is inherent computational

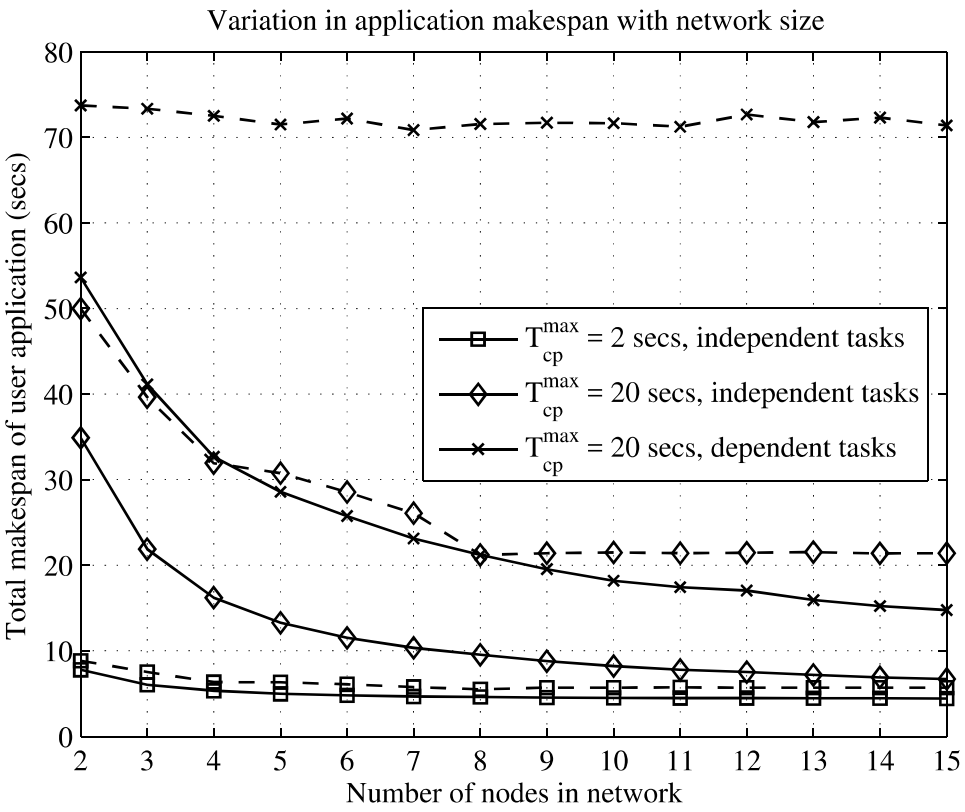


Figure 5.22: Variation in an application's makespan with network size for applications comprising independent and dependent tasks.

diversity in the environment, i.e., when the task execution time range is significantly large. This fact is evident by comparing the performance achieved for task execution time ranges of 2 seconds and 20 seconds, respectively, in Figures 5.22 and 5.24. In addition, it is observed from Figure 5.23 that the master node delegates only a maximum of 60 % of the total workload to slave nodes in the former case while it delegates up to 74 % in the latter case.

A user application with a certain workload and task complexity requires a certain minimum number of nodes in order to achieve efficient execution. When the network size increases beyond this minimum number, there is no significant decrease in the application's makespan. This explains the flattening of the performance curves in Figure 5.24. A similar trend is observed when analyzing the scalability of energy savings with network size, as discussed in

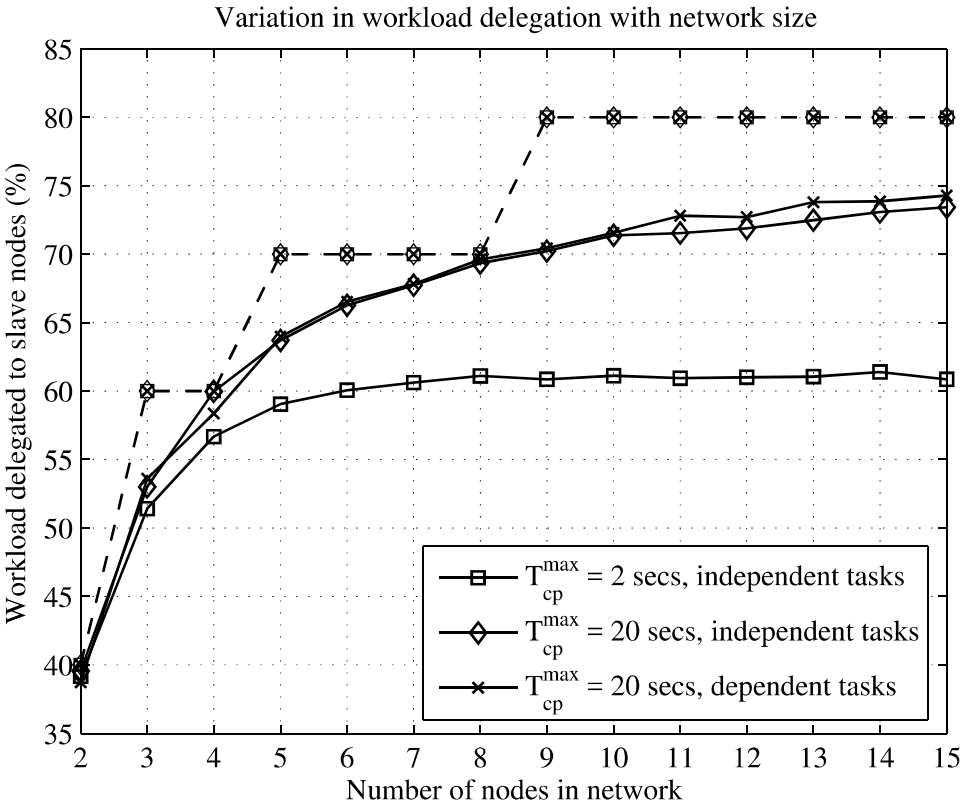


Figure 5.23: Variation in workload delegation with network size for applications comprising independent and dependent tasks.

Section 5.4.3.

In Figures 5.22 and 5.24, the performance curves corresponding to an application comprising dependent tasks follow the same trends as those exhibited by an application comprising independent tasks, except that the makespan and percentage improvement in execution time is, respectively, higher and lower in the former case. Next, the performance achieved with task graph model 1 is compared with that achieved with task graph model 2. The makespan, percentage workload delegation, and percentage makespan improvement for this analysis are shown, respectively, in Figures 5.25, 5.26, and 5.27. The main focus of this analysis is the study of the impact of diversity and concurrency on performance gains achieved by workload distribution. Table 5.4 tabulates the presence of diversity and concurrency in the different

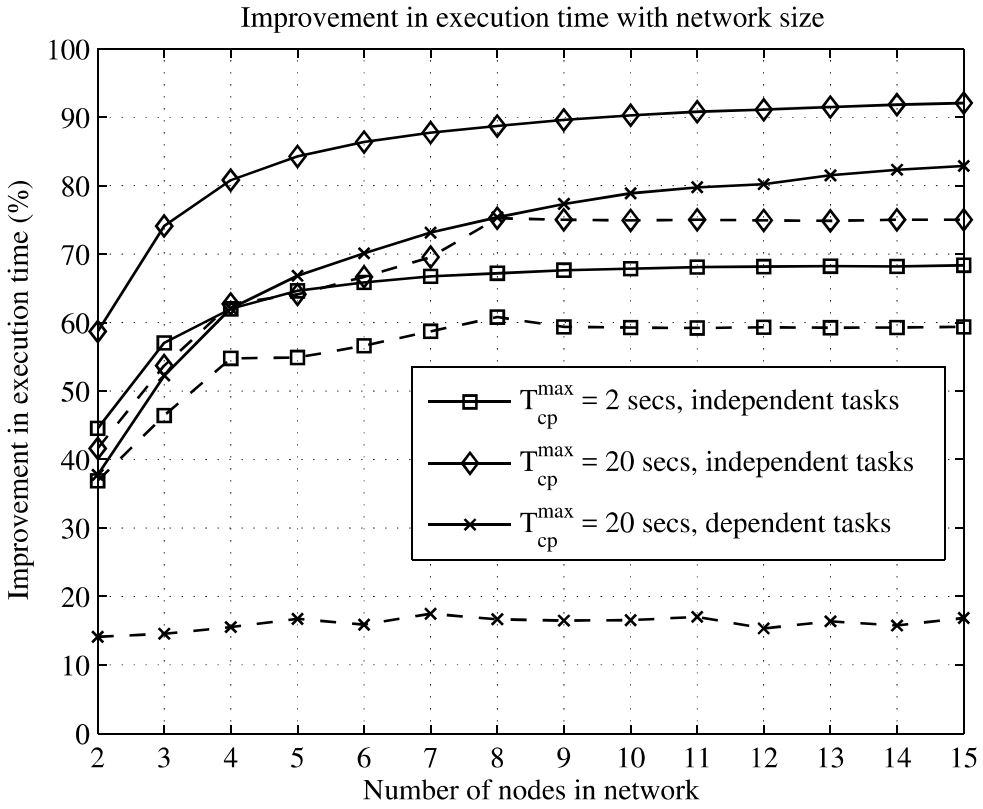


Figure 5.24: Scalability of percentage makespan improvement with network size for applications comprising independent and dependent tasks.

simulation scenarios which are employed for this analysis. Fixed execution time refers to a constant execution time of 1 second for all tasks, regardless of which node executes the tasks. Among the simulation scenarios, scenario 3 achieves the best performance benefits from workload distribution. In scenario 3, the presence of both concurrency and diversity in the WCCN environment encourages the master node to delegate a large percentage of the master node's workload to slave nodes (see Figure 5.26) as the network size increases, which in turn results in higher percentage improvement in the application's makespan. The worst case performance is achieved in scenario 2, where both concurrency and diversity are absent. No tasks are delegated to slave nodes in this scenario. As a result, no improvement in makespan is achievable and the application's makespan does not get impacted by network

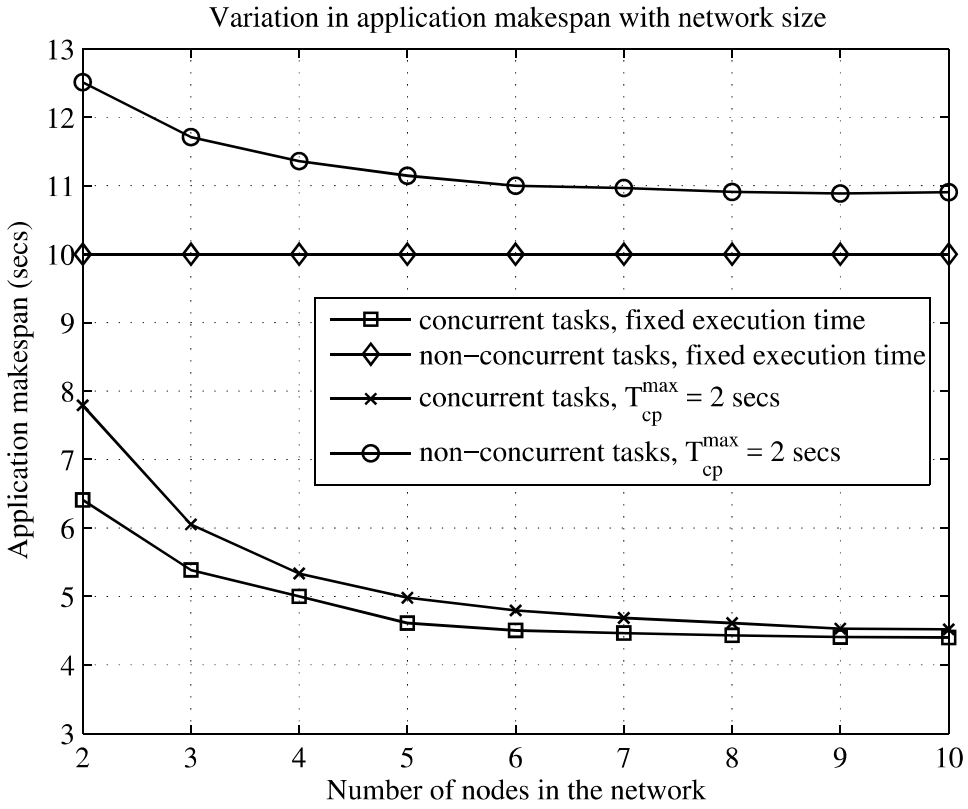


Figure 5.25: Variation in an application's makespan with network size for applications comprising concurrent and non-concurrent software structures.

size. By examining Figures 5.25 and 5.27, the gain in performance caused by the exploitation of concurrency in the application's software structure can be determined by comparing the respective performances achieved with scenarios 1 and 2. Similarly, "diversity gain" can be determined by comparing the respective performances achieved with scenarios 2 and 4. In addition, it is interesting to note that the respective makespan curves for scenarios 1 and 3 converge for large network sizes.

Scalability with transmit power In order to study the scalability of WDC performance with transmit power, simulations are run for an application comprising 10 tasks. The execution time range of all tasks is set to 10 seconds, except for the source and sink tasks which

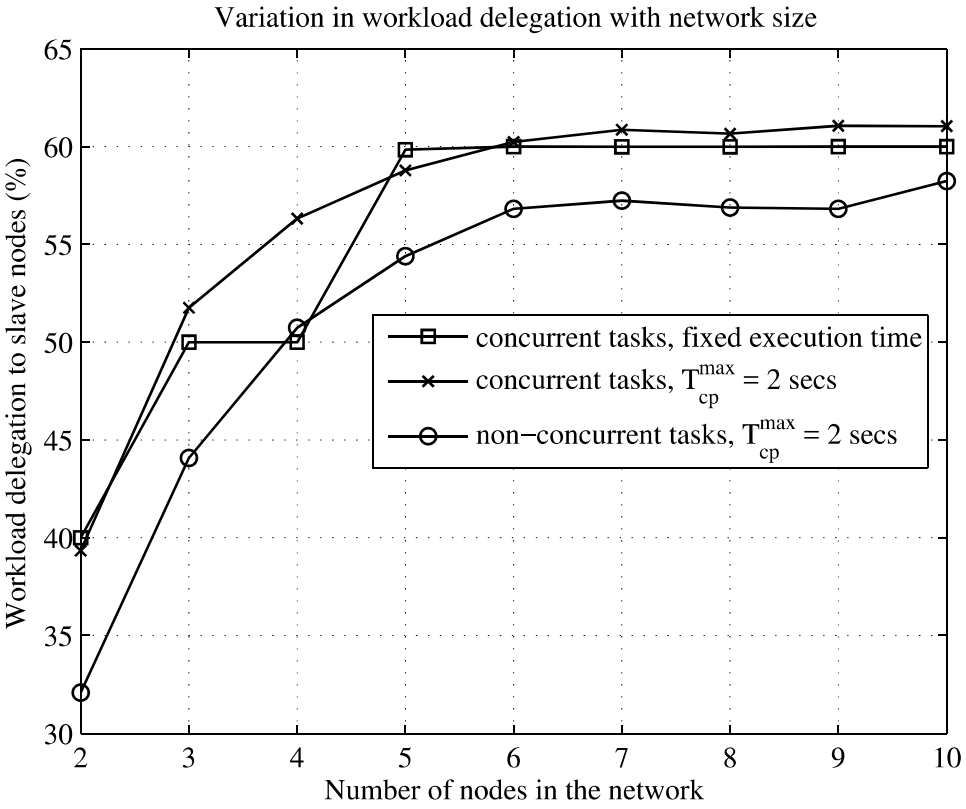


Figure 5.26: Variation in workload delegation with network size for applications comprising concurrent and non-concurrent software structures.

have a fixed execution time of 1 second. The network comprises 10 nodes located within a cluster radius of 50 meters. Figures 5.28, 5.29, and 5.30 plot the simulation results.

Communication latency is one of the key factors that impact an application's makespan when the application is executed using WDC, more so for applications comprising dependent tasks. For a given network radius, an increase in the transmitted SNR improves link connectivity, which in turn reduces communication latency. As the communication latency's dominance over computational latency reduces, an increase in transmitted power encourages workload delegation as observed in Figure 5.29. For a small cluster radius of 50 meters, however, a high transmitted power of 20 dBm is an overkill as it simply increases the communication energy consumption without achieving any further reduction in communication latency. Hence,

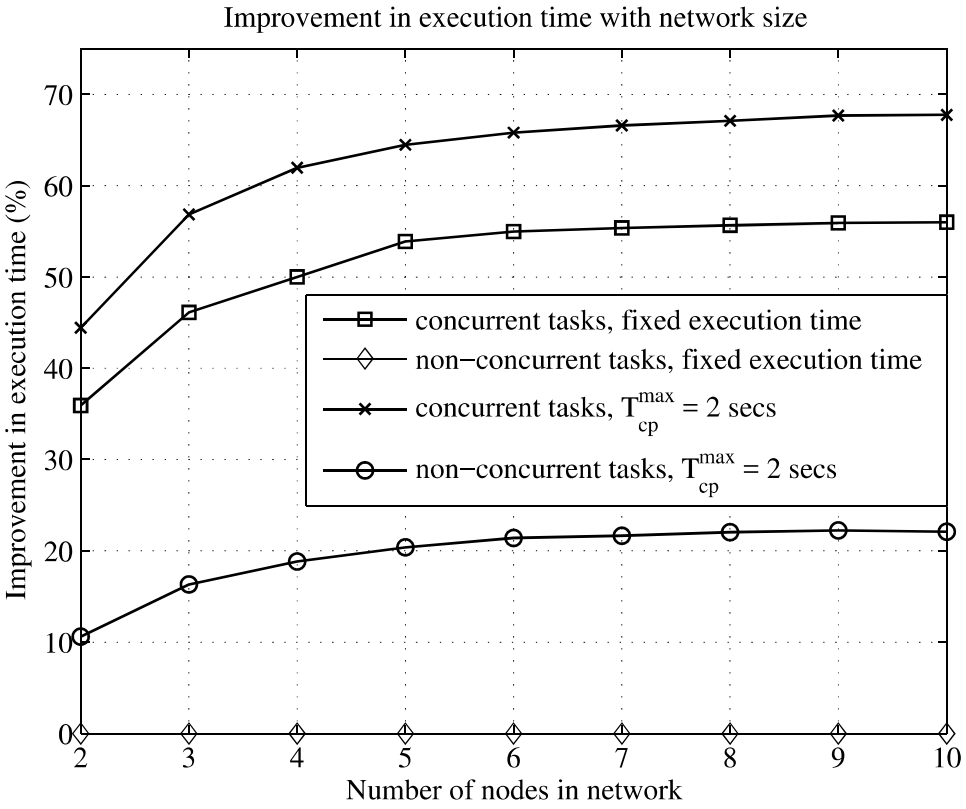


Figure 5.27: Scalability of percentage makespan improvement with network size for applications comprising concurrent and non-concurrent software structures.

a high transmitted signal power discourages workload delegation. The discussion so far explains the trends exhibited by the performance plots shown in Figures 5.28 and 5.30. The application makespan drastically reduces as the transmitted power increases up to 0 dBm and, thereafter, the makespan does not vary any more. For a low transmit signal power of -5 dBm, the communication latency is so high that it is more efficient to process the application locally in the master node itself. In a nutshell, it can be concluded that the transmitted signal power has to be set to an optimal value in order to achieve benefits from workload distribution in a given WCCN. When round-robin scheduling is employed, the workload delegation is 80% for all transmitted signal powers.

When considering an application with independent tasks (task graph model 1), the proposed

Table 5.4: Presence of diversity and concurrency in different simulation scenarios.

Scenario	Application Characteristics	Diversity	Concurrency
1	Concurrent tasks (task graph model 1), fixed execution time	X	✓
2	Non-concurrent tasks (task graph model 2), fixed execution time	X	X
3	Concurrent tasks (task graph model 1), random execution time	✓	✓
4	Non-concurrent tasks (task graph model 2), random execution time	✓	X

algorithm achieves reasonable makespan and percentage makespan improvement even under low transmit SNRs unlike the round-robin task allocation - scheduling algorithm. At high SNRs, the two algorithms do not differ in performance.

5.5.4 Experimental Results

This section presents experimental results gathered from the WCCN prototype which is described in Chapter 4. The prototype operates per the WCCN architecture use case described in Section 4.8. As described in Section 4.5, the Optimizer component in the prototype employs an adapted version of the proposed task allocation - scheduling algorithm which is presented in Section 5.5. As described in Section 4.8, the user application used for the experimental run of the prototype is modeled using task graph model 1. The battery energy supply status of all nodes in the network is hard-coded in the master node's NSM component with values which indicate that the master node has relatively lower energy remaining and the slave nodes have different amounts of remaining battery energy. In addition to the master node's energy resource limitation, this scenario emulates heterogeneity in the availability of energy resources in the network. The mean execution time per task has been measured experimentally to be about 18.67 seconds. It took considerable amount of time to perform

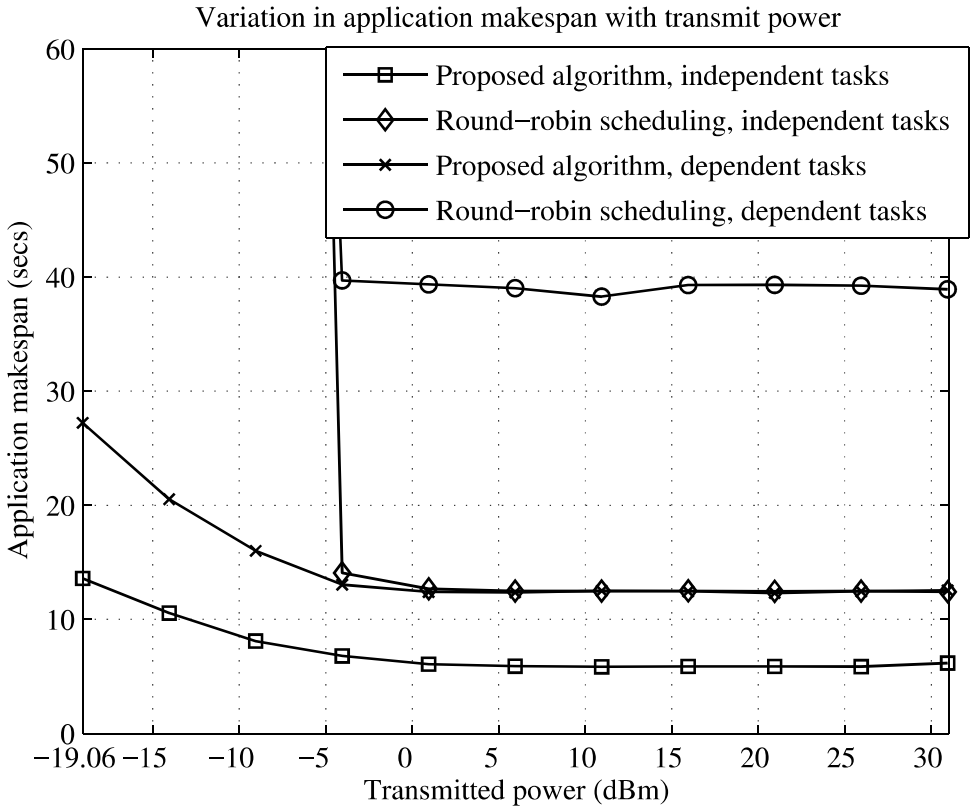


Figure 5.28: Variation in an application's makespan with transmitted SNR.

face recognition since the application's image database was loaded with around 50 high quality images. Since the network comprises nodes with identical computational platforms, the execution time and energy consumption per task are set to a fixed value for all nodes in the network. The task allocation - scheduling algorithm is operated with the objective of minimizing the application's makespan and the objective weights are set accordingly. Each node in the network is provided with knowledge of the addresses of all other nodes so that the experiment does not involve any network discovery. The master node in the network is labeled as node 1 and the slave nodes are labeled, respectively, as node 2 and node 3. Link 1 corresponds to the forward link connecting the master node with node 2 and link 2 represents the reverse link between the same pair of nodes. Links 3 and 4 represent the forward and reverse links connecting the master node with node 3, respectively. The master

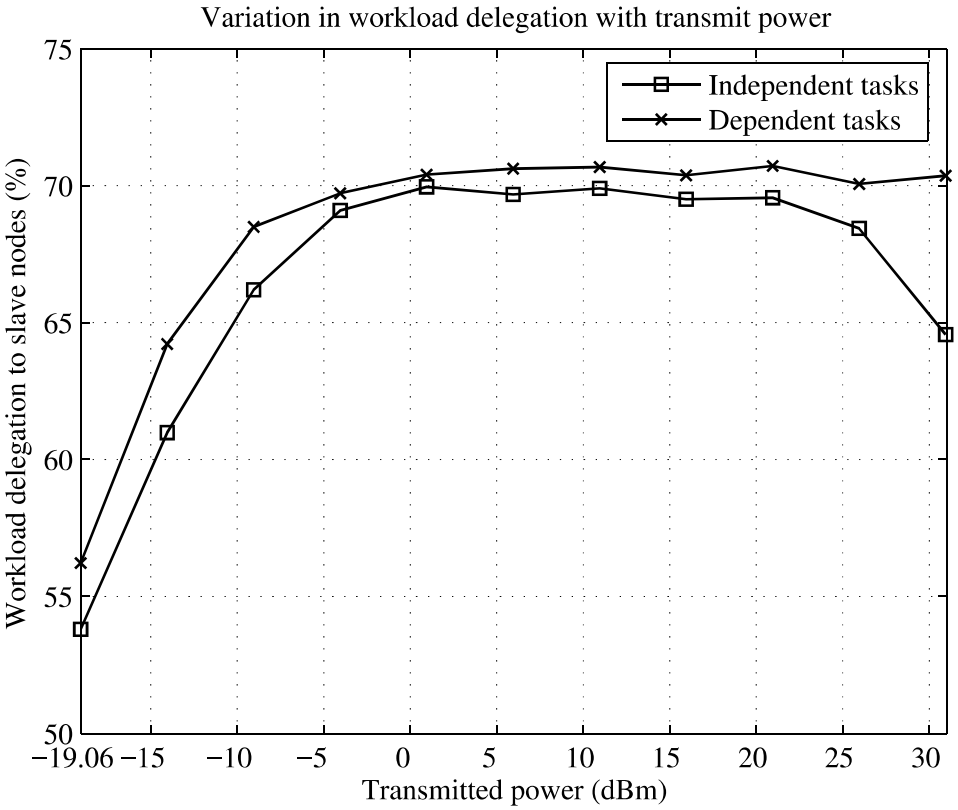


Figure 5.29: Variation in workload delegation with transmitted SNR.

node's NSM component gathers network state information, including link latency and node states, in real-time using network probing messages of size 100 kilobytes and computes the mean goodput of its links with the slave nodes.

Results that were gathered from an experimental run of the prototype are presented next. The results provide clear insight into the inner working of the Optimizer's task allocation - scheduling algorithm. The network state information that was gathered by network probing is displayed in Figure 5.31. Link data rates that were achieved in the experimental run are lower than those achieved with current state-of-the-art wireless technologies simply because the prototype's radio software is configured to operate at low transmission rates in order to avoid causing congestion to the software-defined radio receiver and have a reliable link for workload distribution. As observed from Figure 5.31, link 3 provided higher quality

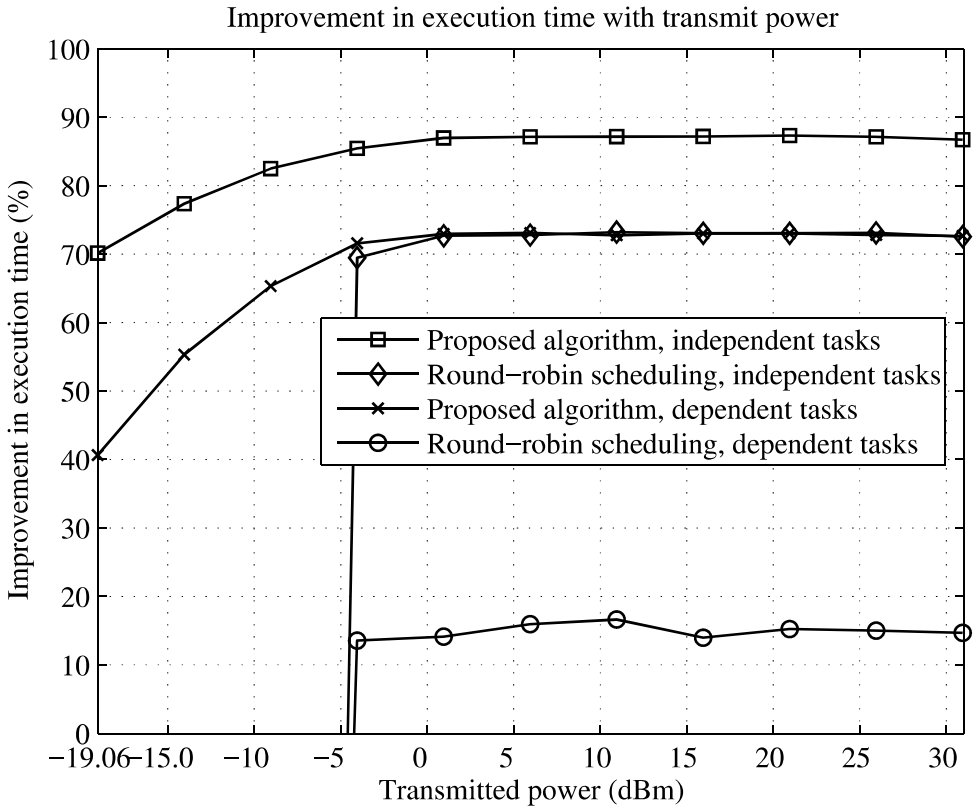


Figure 5.30: Scalability of percentage makespan improvement with transmitted SNR.

compared to link 1. When considering the reverse link qualities, however, link 2 performed better than link 4. The asymmetry in link qualities is attributed to inconsistencies in radio characteristics.

As shown in Figure 5.32, the measured link qualities influenced the workload allocation computed by the Optimizer. Due to the following reason, the task allocation - scheduling algorithm gives more weightage to forward link qualities over reverse link qualities when allocating tasks to the slave nodes. Each image processing task takes an image file as an input and generates a text file of considerably smaller size at the output. The master node uses a forward link to transmit the image file and uses a reverse link to receive a text file. Hence, the algorithm gives higher priority to a slave node that has the best forward link in spite of a poor reverse link. As a result, the algorithm was driven to allocate more workload

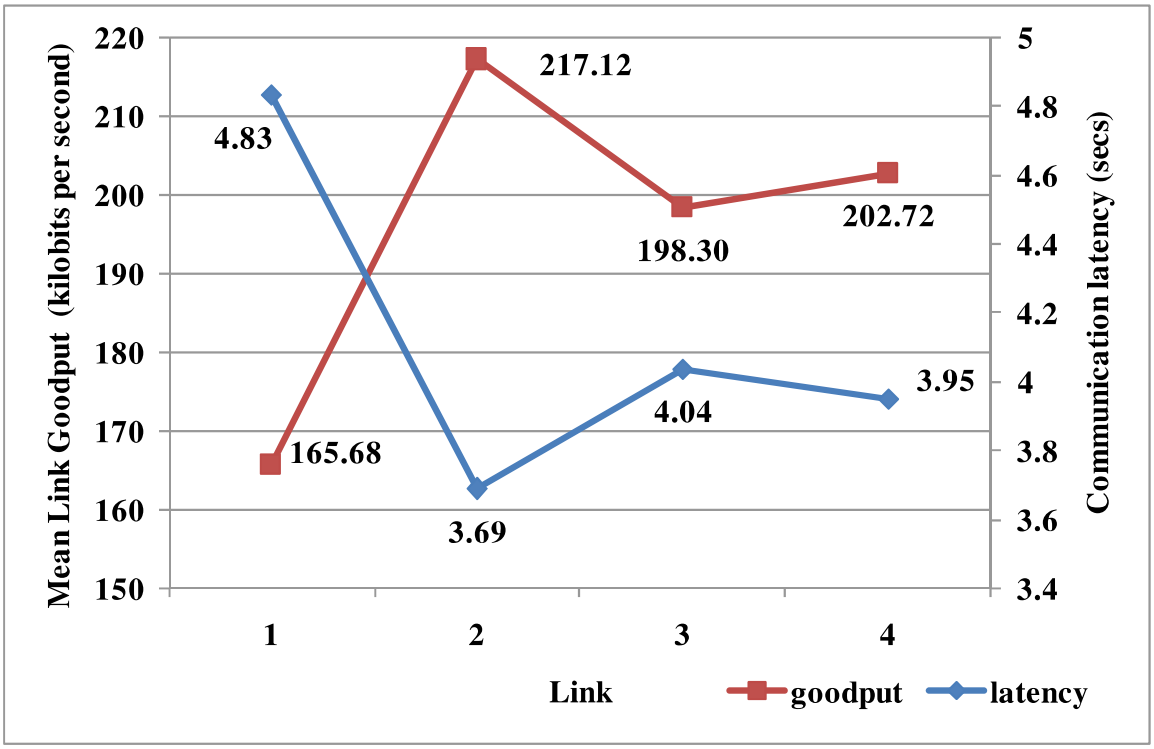


Figure 5.31: Network state information gathered by network probing during an experimental run of the WCCN prototype.

to node 3 as compared to node 2. The algorithm sorts the tasks based on their associated communication costs, which is proportional to the size of the image files. As observed from Figure 5.32, the algorithm began by assigning the tasks associated with the three largest image files, namely, tasks 4, 5 and 6, to the master node. The master node is preferred due to the zero communication overhead associated with the execution of the tasks. However, there occurs a point when the master node's Local Task Executor queue is loaded up so much that it is more efficient to transfer the image files to the slave nodes and have them processed nearly concurrently in the slave nodes. While delegating the remaining tasks to the slave nodes, the algorithm first chose node 3 and assigned it the task associated with the next largest image file, namely, task 3. As node 3's queue got loaded, the algorithm assigned task 2 to node 2. Thereafter, the last remaining task was assigned to node 3.

Experiments were run for different network sizes. The application's makespan was measured

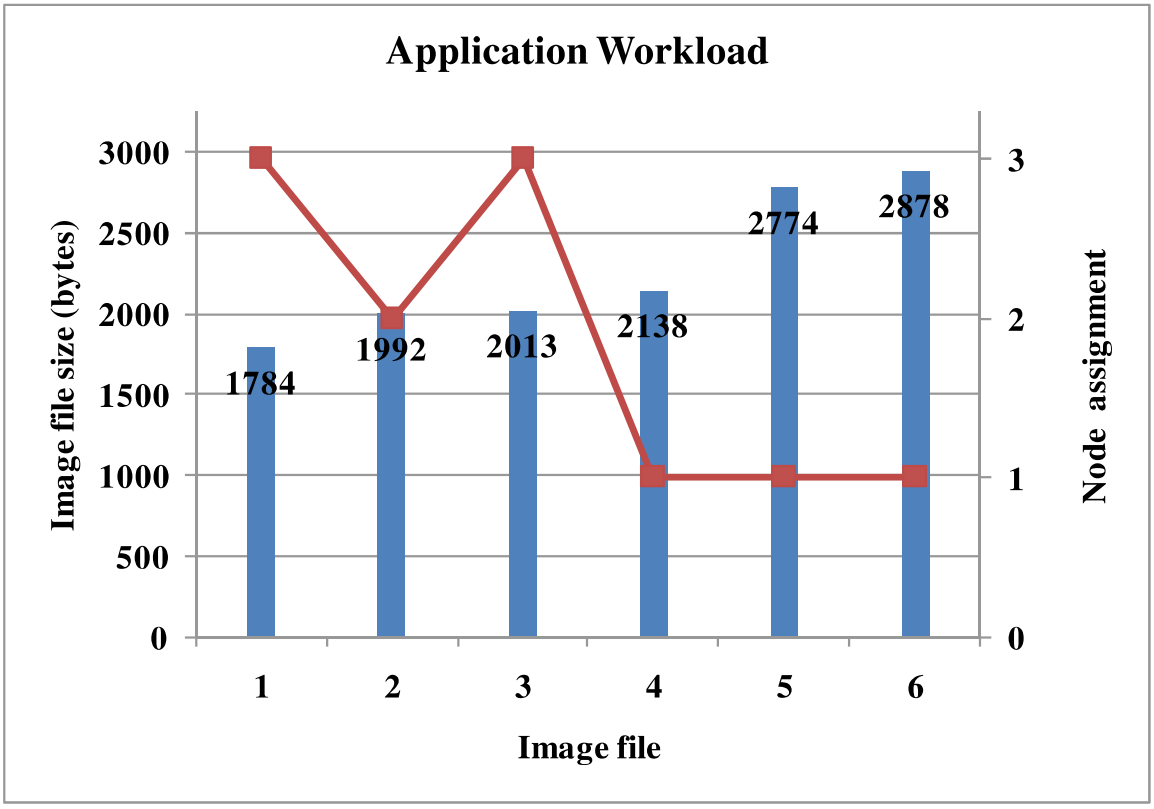


Figure 5.32: Workload allocation scheme employed during an experimental run of the WCCN prototype.

for different network sizes and is plotted in Figure 5.33 in order to demonstrate the scalability of performance with network size. In the case of a single node network, the network comprised only the master node and all processing was performed locally on-board the master node. A considerable improvement in the makespan was achieved after adding a slave node to the single node network. An increase in network size from 2 to 3 nodes did not achieve as much improvement in the application's makespan. It is important to note here that the scalability in the application's makespan with variations in network size as observed from the experimental results presented in Figure 5.33 closely matches with observations made from the simulation results presented in Figure 5.22 (see Section 5.5.3).

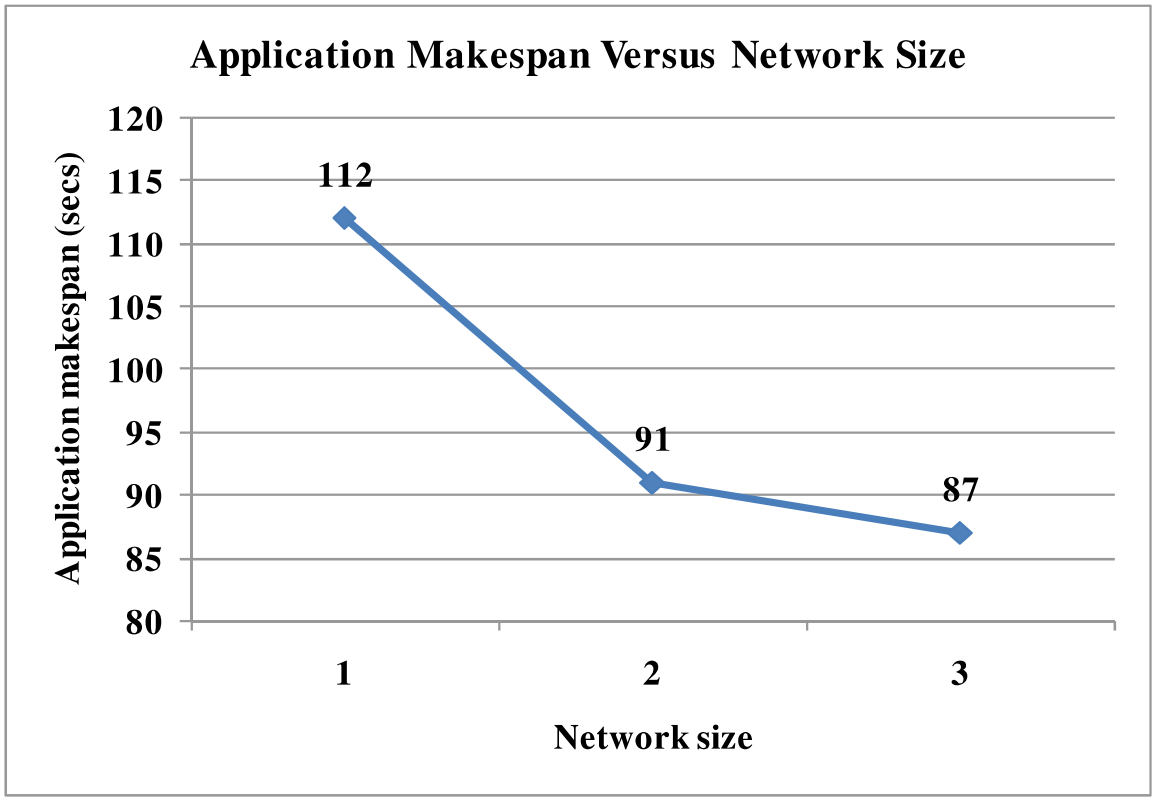


Figure 5.33: Application makespan achieved by WDC workload distribution for different network sizes of the WCCN prototype.

5.6 Application of WDC Workload Allocation in Cognitive Radio Networks

This section demonstrates the application of WDC to CRNs with the help of an example cognitive engine (CE) application proposed by Volos et al. [57]. Traditionally, collaboration among cognitive radio (CR) nodes is limited to sharing of communication resources and knowledge of the operating environment. WDC can serve as an enabling technology that can extend this collaboration to sharing of computational resources in order to collaboratively execute complex mission-critical computational tasks with minimum power and energy consumption. This notion can also be extended to form a WCCN comprised of CRs where information and services can be shared between nodes. Typically, CRNs are intended to

operate in an ad-hoc manner which means that there is no centralized node with abundant computing and power resources that can coordinate and control the network operation. Instead, the computational burden of network control and management is distributed among the CR nodes which are already burdened with execution of computationally-intense user applications such as face and voice recognition applications. Each CR has to perform complex data processing operations for several purposes, such as situational awareness for identification of the characteristics of signals and their sources; cognitive engine decision making which includes data gathering, learning, and reasoning; and execution of user applications. Data processing operations include spectrum sensing [58]; signal detection, classification [59] and feature extraction [60]; geo-location to identify location of malicious emitters; knowledge-base and case-base updates and searches; and natural language processing and computer vision for situational awareness and user interaction via natural user interfaces. In the context of a CE, the following tasks are common. First, the tasks involve the estimation of BER performance based on theoretical BER curves. CEs use the curves to evaluate hundreds of communication configurations and select a few to try over-the-air [61, 62]. During radio startup time, the BER curves can also be used to initialize an observations database which stores information about the observed effectiveness of various communication methods when employed under different channel conditions [57]. Second, estimating confidence intervals of observations in the database [57] requires estimation of the beta cumulative distribution function which is computationally demanding. Third, CEs that employ case-based reasoning frequently search through records in the knowledge-base and analyze past cases [61, 63]. Database management tasks in the CE can consume significant amount of computing resources. Such complex computational tasks, along with high resource demands and stringent real-time requirements, may not be well suited for handling by a single node. Furthermore, the availability of resources varies with time and varies between radio nodes, mainly due to the heterogeneity in the nature of computational tasks (as well as the associated computational burden) handled by different nodes. While some CRs may be engaged in signal classification, for instance, others may be involved in geolocation. These problems im-

pose a challenge to the real-world implementation of CRNs using low-complexity and small form factor CRs. As an alternative to the traditional approach of local computing, multiple resource-constrained CR nodes can form a WCCN and collaborate on executing the complex computational tasks in a distributed manner.

The CE under consideration functions based on an observations database that initially stores the number of successful and failed packets associated with a communication method and channel conditions pair. The CE uses the number of successful and failed packets to estimate a confidence interval for the packet success rate (PSR) of the associated communication method for a particular channel condition. For every packet received, the number of successful and failed packets for the particular communication method used under particular channel conditions is updated followed by update of the confidence intervals. The Gittins index is computed based on the confidence intervals and is used to optimally balance exploitation and exploration by selecting the method with the highest index [57, 64]. Each entry in the database, which is indexed by the associated communication method and channel condition (quantified by metrics such as SNR), consists of the number of successful packets, the number of unsuccessful packets, confidence interval of the PSR, and the Gittins index.

An overview of the database update process is presented as follows. Packet observation information is collected, where each observation consists of the channel condition, the communication method used, and an indication of success of the packet reception. The database entry for the corresponding channel condition and communication method is updated for each packet and the knowledge is propagated throughout the relevant portions of the database. For example, if a communication method A is found to drop a packet at 10 dB SNR, it is also expected to drop a packet in any SNR below 10 dB. Therefore, all the entries at 10 dB and below for communication method A are marked with an unsuccessful packet. In addition, if 32 QAM is observed to deliver a successful packet at a certain SNR, 16 QAM and 4 QAM are also expected to deliver a successful packet at that SNR, and for this reason the corresponding entries in the database are marketed with a successful packet too. Once the entire set of packet observations have been taken into consideration, the confidence in-

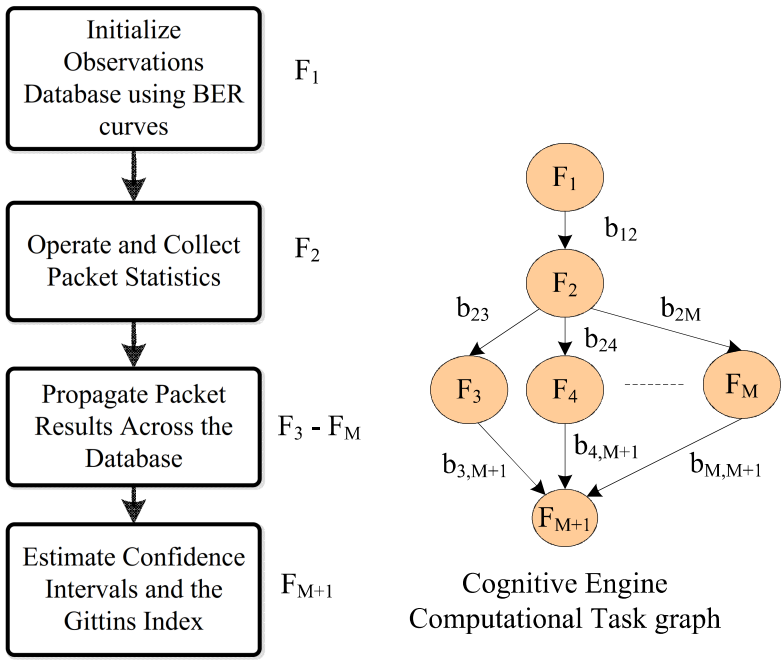


Figure 5.34: Flow diagram and task graph of an example cognitive engine application.

tervals and Gittins indices are calculated. Figure 5.34 illustrates the different computational tasks associated with the CE operation. The tasks $F_3 \ldots F_M$ can be processed by different nodes and aggregated back together. The various tasks that constitute the example cognitive engine application are explained as follow:

1. Task 1 (F_1) : Initialize the observations database by assigning uniform values for packet success and failures. Based on theoretical BER curves, populate the database (i.e. update the packet successes and failures) and compute the confidence intervals and Gittins indices.

2. Task 2 (F_2) : Try a particular communication method over-the-air and collect packet observation information. The packet information can either be collected by one node or by several nodes. The task graph under consideration assumes that the packet information is collected by only one instance of the "Operate and Collect Packet Statistics" operation, i.e., only one node collects the information.

3. Task 3 ($F_3 \ldots F_M$) : Use the packet information to update entries in the database. The collected packet information set can be fragmented into smaller mutually exclusive portions of data, where a node can process one or more subsets. Tasks $F_3 \ldots F_M$ represent different portions of the collected data set. Each portion can be used to update a different instance of the observations database, where each collaborating node maintains an instance of the entire database. For instance, if node 2 is allocated F_3 and F_4, node 2 uses the corresponding data to update its instance of the database. In the case when each portion of the data corresponds to one particular communication method, increasing the number of data sets allocated to a node also proportionally increases the amount of data passing of results to F_{M+1} (or the master node). In this manner, the amount of data passing between any one of the tasks $F_3 \ldots F_M$ and the sink task F_{M+1} does not depend on whether the tasks $F_3 \ldots F_M$ are co-located in one node or distributed among several nodes.

4. Task 4 (F_{M+1}): Assimilate the different instances of the observations database and compute the confidence intervals and Gittins index.

5.7 Chapter Summary

Resource allocation, which is one of the core design aspects of WCCNs, is discussed in this chapter with specific focus on workload allocation and scheduling. This chapter presents the proposed workload allocation algorithm and the proposed task allocation - scheduling algorithm. In addition to computing optimal resource allocation parameters, these algorithms help in making decisions between local and distributed processing of a user application based on perceived WCCN operating conditions. This chapter analyzes the impact of system design parameters on power savings, master node's energy savings, application's makespan, and percentage improvement in makespan, achieved by workload distribution using the proposed algorithms. The system design parameters employed in this analysis are network range, net-

work size, task execution time range, application's workload, and transmitted signal power. This chapter also introduces WDC as a technology that enables efficient execution of user applications and network management services in cognitive radio networks.

A general observation is that workload distribution is beneficial when the computational cost associated with processing a task is significantly greater than the communication cost incurred in offloading the task to a slave node. Non-linear processor power consumption characteristics, concurrency in user applications' software structure, and computational diversity inherent in a WCCN environment are some of the factors that provide performance gains from workload distribution. The results presented in this chapter are summarized as follows.

Power efficiency achieved by workload distribution: When analyzing the savings in power consumption achieved by WDC workload distribution, it is found that workload distribution can be effectively used to exploit non-linearity in the computational subsystem's processor characteristics. Power savings are high when workload distribution is applied to computational platforms with highly non-linear computational power consumption - frequency characteristics and applications comprising high power computational tasks. When simple round-robin workload distribution is employed in highly dense networks, network power savings improves with an increase in the network size when the network operates in the power reduction mode. However, network power savings begins to degrade beyond a certain network size when the costs incurred by the communication overhead begins to dominate over computational costs.

Are resource constraints really a driving factor for workload distribution: A node's resource constraints are important drivers for it to shed its computational workload to slave nodes. Workload distribution is, however, not beneficial when there is no sufficient number of nodes and no sufficient power resources in the network. A certain minimum network size and a certain minimum amount of power resources are required particularly when processing complex applications with large workloads. An increase in the network size along with the

availability of sufficient power resources per node results in an increase in the availability of resources that can be shared by the master node to process complex applications. On the other hand, having abundant resources does not linearly scale up the benefits of workload distribution. Whenever the application's complexity, amount of power resources or network size is increased beyond a certain value, the performance benefits of workload distribution encounter the "law of diminishing returns".

Energy savings achieved by workload distribution: The master node's energy savings scales non-linearly with the application's computational workload, computational power consumption per task, network size, power supply availability, and communication range.

Improvement in an application's makespan achieved by workload distribution: The chapter analyzes the impact of system parameters on an application's makespan achieved by workload distribution. The analysis is performed by first varying system parameters one at a time and, then, by varying pairs of system parameters. Individually increasing network size, task execution range, and application workload results in a non-linear increase in percentage workload delegation and, in turn, an increase in percentage improvement in makespan until the "law of diminishing returns" is encountered. Individually increasing network range, task execution time range, and workload increases an application's makespan. When increasing network size, the reduction in makespan is greater for long task execution time ranges. An increasing in transmit power encourages workload delegation and reduces the application's makespan. However, this occurs at the cost of increased power consumption. Hence, there exists an optimal range of transmit signal powers for workload distribution in a given wireless network environment.

When considering the joint impact of network size and one other system parameter, such as task execution time range, application workload, and network range, an increase in network size tends to reduce the variation in makespan caused by increasing the other system parameter. When jointly considering the impact of network range and application complexity, an application's makespan is not impacted significantly by an increase in network

range when the application's workload and task execution time range are small. In the case of an application comprising high workload and long task execution time range, however, the application's makespan increases when jointly increasing both network range and task execution time range.

Diversity in WCCN environments: The proposed algorithms exploit diversity in a WCCN environment in order to provide performance gains over local processing. Computational diversity, diversity in the availability of power-energy resources, and diversity in link quality and inter-node distances are some of the diversity factors that can exist in a WCCN environment. High values of workload size, task execution time range, and network size can improve computational diversity in a WCCN environment. An increase in network size improves diversity, as well as increases the number of nodes located close to the master node and the amount of computational resources that be utilized to exploit concurrency in the application's task graph. There is, however, significant benefit in large network sizes only when there is significantly high computational diversity in both the application and the environment.

An increase in diversity encourages workload delegation, which in turn decreases an application's makespan, increases the master node's energy savings, and increases percentage improvement in the application's makespan. In addition, an increase in diversity caused by increasing the network size reduces the impact of system parameters, such as network range and application's workload, on the performance metrics.

Comparison among task graph models: In the absence of both concurrency and computational diversity, the proposed task allocation - scheduling algorithm does not delegate any workload to the slave nodes, hence it provides no improvement over local processing. When an application is structured per task graph model 2, the percentage improvement in the application's makespan increases rapidly as computational diversity increases. Task graph model 2, however, produces longer makespan and lower percentage improvement in makespan than task graph model 1 when either task execution time range or network size is

increased.

When an application that is structured based on task graph model 3 is compared with one that is structured based on task graph model 1, the former achieves longer makespan by workload distribution because of the increased communication overhead and the increased wait time experienced by tasks as they wait until all their respective predecessors have finished processing. As network range is increased, for instance, model 3 gives longer makespan, higher rate of reduction in makespan with an increase in network range, lower workload delegation, and lower percentage improvement in makespan. A different trend is observed when considering the impact on the master node's energy savings. Due to the following reasons, task graph model 3 performs better than model 1 in terms of master node's energy savings and it performs poorly in terms of makespan. The master node's energy savings is influenced by the master node's communication cost, while makespan is impacted by the communication cost incurred by distributed processing on the network as a whole. Task graph model 3 introduces randomization in the communication between the source task and the remaining tasks, and communication between the sink task and the remaining tasks. This randomization can potentially reduce the communication burden at the master node. However, the increased dependency between the remaining tasks can increase the network's communication costs which, in turn, impacts the application's makespan.

Comparison between round-robin task allocation - scheduling and the proposed adaptive algorithms: In most scenarios, the proposed adaptive algorithms perform better than round-robin task allocation - scheduling. Unlike the round-robin approach, the proposed algorithms do not necessarily allocate the application's tasks to all nodes in the network. They, instead, only choose nodes that minimize the overall objective and ignore the nodes that result in objective values that are far from being optimal. The performance gains achieved by the adaptive approaches over the round-robin approach are mainly due to the exploitation of diversity in the operating environment and the application's concurrency. In scenarios where there is no significant diversity in the environment, the proposed algorithms do not show any significant improvement over the round-robin approach. As diversity is

increased, however, the proposed algorithms provide significant performance gains over the round-robin approach. A round-robin approach does not provide any improvement in an application's makespan when applied to an application which comprises non-concurrent tasks. There is less diversity in link conditions when high transmit signal powers and short network ranges are employed, and all links offer nearly the same communication latency and communication energy consumption.

Experimental versus simulation results: Experimental results gathered from the WCCN prototype demonstrate nearly 80% improvement in the execution time of a face recognition user application when compared with that achieved with traditional local processing. The scalability in the application's makespan with variation in network size, as inferred from simulation results, closely matches with the experimental results.

Chapter 6

A Response Time Model for WCCNs

This chapter presents a mathematical model of the response time experienced by a user application when it is executed in a typical WCCN environment. In this chapter, the WCCN environment under consideration comprises one service provider and a finite number of service requesters, where the service requesters disseminate their respective user application service request messages to the service provider using a simple time division multiple access MAC protocol and they contend for the limited computational resources available in the service requester. Note that, the "service requester" and "service provider" nodes in a general WCCN environment correspond, respectively, to the "master" and "slave" nodes in a WCCN which is setup in a master-slave configuration. The response time model is developed based on a proposed queuing network model of the WCCN environment. The queuing network model mimics queuing behavior in various components in the proposed WCCN architecture. Mixture distributions are employed to model the delay characteristics in individual queues in the queuing network. One such queue models the delay experienced by service request messages in transmitters that employ message buffering, where all request messages that originate from a service requester node are buffered at the node's transmitter before they are communicated to the intended service provider node. The communication latency takes into consideration the transmission success probability experienced by service

request messages in the presence of path loss, fading and noise, where inter-node distances are modeled by a binomial point process. Throughout this chapter, it is assumed that an application service request message comprises a single packet. In all results presented in this chapter, the system model and assumptions presented in Chapter 3 is used along with the assumptions in this chapter.

The main contribution of this chapter is the derivation of the statistical properties of response time, including its probability density function, and analysis of its scalability under different wireless channel conditions. Second, the use of mixture distributions greatly simplifies the response time model and simplifies the derivation of the statistical properties of response time. The distribution function, in particular, is important in assessing response time guarantees under given environmental conditions. Third, the response time model uses stochastic geometry to model the location of a finite number of nodes in a WCCN in order to closely mimic a real world network.

The chapter is organized as follows. The proposed WCCN queuing network model is presented in Section 6.1. Section 6.2 presents a delay distribution model for finite sized queues. In the latter part of this chapter, this model is applied to the communication and computational resource queues of the proposed WCCN queuing network in order to derive the WCCN response time model. Section 6.3 describes the proposed model of communication latency experienced by application service request messages. Stochastic models of the probability of transmission success and the delays experienced in a simple buffered time division multiple access (TDMA) communication scheme are derived in this section. The proposed WCCN response time model is presented in Section 6.4 along with some performance analysis results. Finally, Section 6.5 presents the chapter summary.

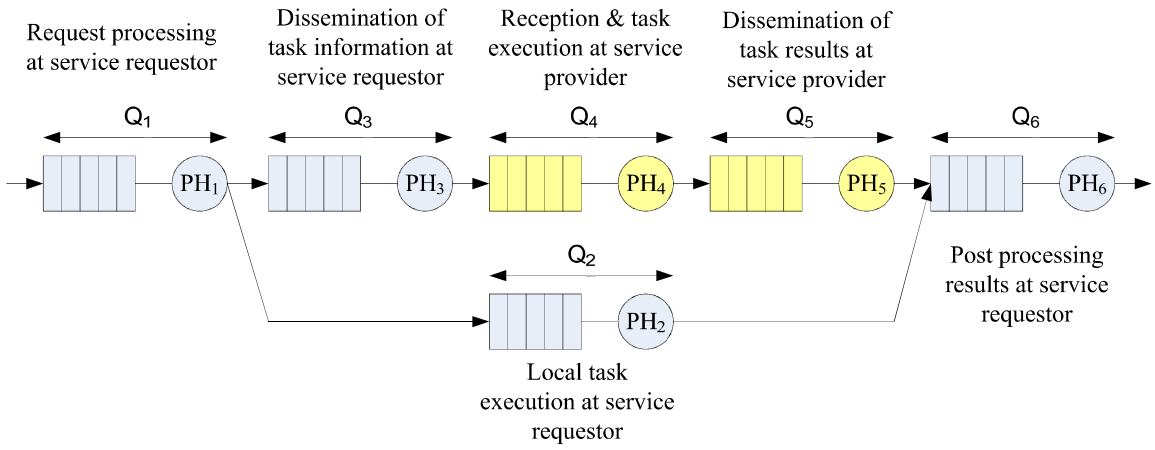


Figure 6.1: The proposed WCCN queuing network model. Queues in yellow are located inside service provider nodes while queues in blue are located inside service requester nodes.

6.1 WCCN Queuing Network Model

This section presents the proposed WCCN queuing network model, where all queues are of finite size and process messages on a first-in first-out basis. The model, as viewed from the perspective of a single service requester, is illustrated in Figure 6.1. The model captures a simple scenario which involves a number of service requesters attempting to access the computational resources of a single service provider, as illustrated in Figure 6.2. Inside a service requester node, requests which are received from user applications are first queued in queue 1, where each request is evaluated for feasibility and workload allocation. Queue 1 resides inside the "Service Request Queue and Processor" component of the proposed WCCN architecture. A request enters either queue 2 or queue 3 based on the workload allocation. A request that is allocated for local processing enters queue 2 and waits until the node's on-board computational resources become available for processing the request. Queue 2 resides inside the "Local Task Executor" component of the service requester node. A request from queue 1 enters queue 3 if it is allocated for remote processing. Queue 3 resides inside the service requester's "Communication Layer". In queue 3, a request is buffered until the required communication resources becomes available to transmit the message to the intended service provider. A request message exits queue 3 either when the transmission of the message

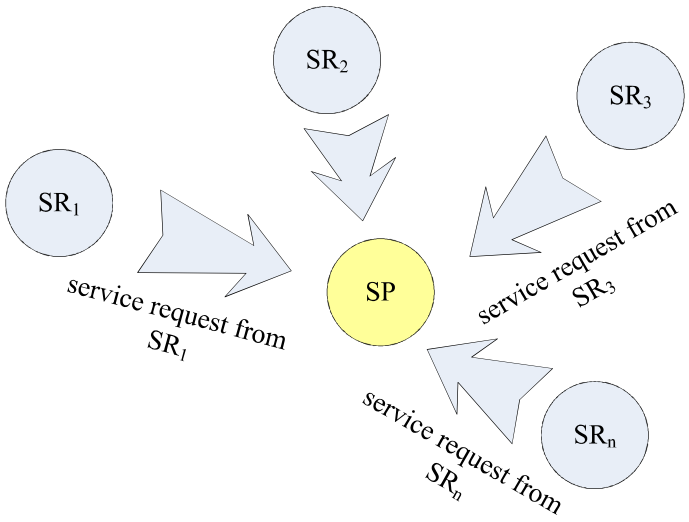


Figure 6.2: A fundamental resource contention scenario in a WCCN.

to the intended node fails due to poor link quality after a finite number of re-transmissions, or after it is successfully transmitted to the intended service provider. If a request message is received successfully by the service provider node and there are available vacancies in queue 4, the request enters queue 4 and waits in the queue until the required computational resources in the service provider become available to process the request. Queue 4 resides in the "Service Request Queue and Processor" component in the service provider. The time taken to process a request in queue 4 is modeled using the queue's service time distribution model. Results generated at the output of a requested service are buffered in queue 5 in the form of a message. Queue 5 resides in the service provider's "Communication Layer". The service time distribution of queue 5 depends on the link connectivity between the service provider and the intended requester. Messages carrying results destined to a service requester exit from queue 2 and queue 5, and enter queue 6. Queue 6 resides inside the "WDC Controller" component in service requester nodes. In queue 6, results undergo post-processing before they are made available to the respective user application that had originally generated the request corresponding to the results. While only the service requester under consideration is shown in the model, traffic injected by other service requesters into the network is accounted in the overall network departure rate, which in turn is the arrival rate to queue 4. The

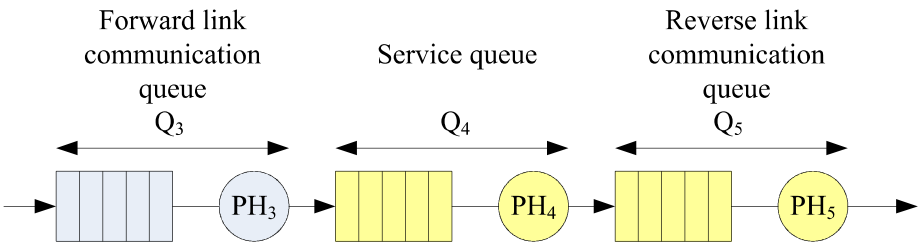


Figure 6.3: Simplified WCCN queuing network model used to model response time.

factors that impact the transmission success of a message and the network departure rate are discussed in Section 6.3.

This fundamental model can be easily scaled up to represent a compounded scenario involving multiple service requesters and service providers, where each service requester disseminates multiple requests to different service providers. A service requester can avail the services of multiple service providers for several reasons. In the case of an application that comprises heterogeneous services, the application's workload is delegated only to the service providers that can provision the required services. Second, multiple service providers may be employed to process an application in order to achieve workload balancing or enhancement in the application's security, where service providers with good metrics, such as wireless link quality, computational resource availability (or resource queue occupancy) and link security, can be allocated proportionally higher number of requests to process. Third, a service requester can leverage diversity in both communication and computational resources across the network. Multiple copies of a request message can be disseminated to different service providers. Furthermore, the proposed model can be realized for different queue disciplines and serves as a foundation for further analysis and design of mechanisms, such as flow control, that are critical to managing service request-provisioning processes. The model presented in Figure 6.3 can be used to analyze WCCN performance metrics, namely, response time, throughput, fairness, and resource availability (or outage probability). A simplified queuing network model, which is shown in Figure 6.3, is used in the rest of this chapter to derive a fundamental response time model for WCCNs.

6.2 Delay Distribution in Finite Sized Discrete-Time Queues

This section presents a proposed delay distribution model for finite - sized discrete - time queues. This model is one of the building blocks of the WCCN response time model. The delay distribution model is proposed with the following assumptions: a. queue size is finite; b. arrival and departure processes of a queue are stationary; c. queue occupancy is a stationary random process, such that it does not vary according to the actual time of arrival of a tagged message; d. messages are serviced on a first-in-first-out basis; e. service time distribution of a queue is memoryless; and f. service times are i.i.d. for all messages. The memoryless property of the service time distribution ensures that the distribution of wait time experienced by a tagged message does not depend on the amount of time elapsed while processing a message that is being serviced at the time when the tagged message enters the queue. In addition, the discrete queue model assumes "late arrival with delayed replacement" [65]. In the late arrival model, messages arrive just prior to the end of a slot, contrary to arriving at the beginning of the slot. A message potentially arrives while a departure event is close to completion. The slot in which the message arrives is not counted towards the message's wait time in the queue. If a message arrives at the first location (or head) of the queue at the end of a slot, it's processing begins only at the beginning of the next slot. Thus, if a message arrives at the end of a slot when the queue is empty and it's processing completes at the end of the next slot, then it's wait time is zero and it's total queue delay, which includes its wait time and its processing time, is 1 slot. In a delayed replacement model, when a message arrives in the final slot of a service time, i.e., the slot during which a departure is bound to happen, the message is not accepted into the queue if the queue is full prior to the departure.

Let N_{queue} denote the number of messages in a queue at an arbitrary time slot. If there is vacancy in the queue, the tagged message enters the queue and experiences a wait time, as it waits in the queue until all N_{queue} messages are serviced. In addition, the tagged message

experiences a service time. Service time is the delay experienced by a message from the time it enters the queue server till it exits the server. Accordingly, the total delay experienced by the tagged message in the queue, denoted by T_{qdelay}, is given by

$$T_{qdelay} = \sum_{j=1}^{N_{queue}+1} T_{svc}^{j}, \tag{6.1}$$

where T_{svc}^{j} denotes the service time of the j^{th} message in the queue. Let T_{sum}^{m} denote a random variable which is computed as a sum of m i.i.d. service time random variables, i.e.,

$$T_{sum}^{m} = \sum_{j=1}^{m} T_{svc}^{j}. \tag{6.2}$$

Let the message arrival rate, service rate, and departure rate at a queue be denoted, respectively, by μ_{arr}, μ_{svc}, and μ_{dep}. Arrival rate is the probability that a message arrives in a time slot. Service rate represents the probability that a message completes processing in an arbitrary time slot. Departure rate captures the rate at which messages exit the queue. Q_{size} denotes the size of the queue, which includes vacancies for messages waiting in the queue and a vacancy for the message that is currently being serviced. The probability that there are i messages in the queue at an arbitrary time is denoted by p_{occup}^{i}. The probability mass function of T_{sum}^{i} is denoted by $f_{T_{sum}^{i}}(t;\ i;\ \mu_{svc})$, i.e., it represents $Pr\{T_{sum}^{i} = t\}$. In Equation 6.1, N_{queue} is a random variable related to queue occupancy and T_{svc}^{j} is a random variable which follows the service time distribution of the queue. Hence, the probability mass function of T_{qdelay} can be expressed as a finite mixture distribution where the mixture weights represent the probability of queue occupancy, as given by

$$f_{T_{qdelay}}(t) = \sum_{i=0}^{Q_{size}-1} p_{occup}^{i} \times f_{T_{sum}^{i+1}}(t;\ i+1;\ \mu_{svc}), \tag{6.3}$$

$$\text{where} \quad \sum_{i=0}^{Q_{size}-1} p_{occup}^{i} = 1. \tag{6.4}$$

In Equation 6.3, T_{sum}^{i+1} includes the service times of all messages ahead of the tagged message in the queue and the service time of the tagged message itself. Queue occupancy at an arbitrary time can be derived based on the queue type, queue size, service distribution, queue discipline, and characteristics of the arrival process [65]. The delay distribution model presented in Equation 6.3 characterizes the total queuing delay experienced by an arbitrary message, given that the message successfully enters the queue. As per the "delayed replacement" assumption, the model ignores the case where the queue is full and the tagged message is blocked from entering the queue. Throughout this chapter, it is assumed that all queues in the WCCN queuing network model are $Geom/Geom/1/K$ queues (geometric arrival process, geometrically distributed service times, single server, and finite queue size), where the message arrival and service time processes follow their respective geometric distributions. Accordingly,

$$T_{svc}^{j} \sim Geom\left(\mu_{svc}\right), \tag{6.5}$$

$$\text{where } Pr\left\{T_{svc}^{j} = t\right\} = \left(1-\mu_{svc}\right)^{t-1} \times \mu_{svc},\ t = 1,\ 2,\ 3, \ldots. \tag{6.6}$$

It is reasonable to assume a geometric arrival process since all operations in a node, including the generation of service requests, are clocked in discrete time. The queue occupancy distribution for a $Geom/Geom/1/K$ queue is derived in Takagi [65]. In this chapter, the message service time is assumed to be unbounded. When a finite limit is placed on the service time, the service time distribution is by itself a mixture distribution. Now, it is known that the sum of i.i.d. geometric random variables follows a negative binomial distribution. Thus, the

$i-1^{th}$ component in the mixture distribution of Equation 6.3 is given by

$$T^i_{sum} \sim NB\left(t;\ i,\ \mu_{svc}\right), \tag{6.7}$$

$$\text{where } f_{T^i_{sum}}\left(t;\ i,\ \mu_{svc}\right) = \begin{pmatrix} t-1 \\ i-1 \end{pmatrix} \mu^i_{svc}\ \left(1-\mu_{svc}\right)^{t-i},\ t=i,\ i+1,\ i+2,\ldots, \tag{6.8}$$

$$\mathbb{E}\left\{T^i_{sum}\right\} = \frac{i}{\mu_{svc}}, \tag{6.9}$$

$$\text{and } Var\left\{T^i_{sum}\right\} = i\,\frac{\left(1-\mu_{svc}\right)}{\mu^2_{svc}}. \tag{6.10}$$

$\mathbb{E}\left\{T^i_{sum}\right\}$, $Var\left\{T^i_{sum}\right\}$, and $F_{T^i_{sum}}\left(t;\ i;\ \mu_{svc}\right)$ denote, respectively, the mean, variance, and cumulative density function (CDF) of T^i_{sum}.

As discussed next, the mixture distribution model is used to derive the statistical properties of queuing delay. The mean (and CDF) can be derived as the convex combination of the mean (and CDF) of the mixture components, as given by

$$\overline{T}_{qdelay} = \sum_{i=0}^{Q_{size}-1} p^i_{occup} \times \mathbb{E}\left\{T^{i+1}_{sum}\right\} = \sum_{t=1}^{\infty} t \times f_{T_{qdelay}}\left(t\right), \tag{6.11}$$

$$F_{T_{qdelay}}\left(t\right) = \sum_{i=0}^{Q_{size}-1} p^i_{occup} \times F_{T^{i+1}_{sum}}\left(t;\ i+1;\ \mu_{svc}\right), \tag{6.12}$$

$$\text{where } \sum_{i=0}^{Q_{size}-1} p^i_{occup} = 1. \tag{6.13}$$

Equation 6.13 ensures that the weights of the mixture distribution are normalized to the probability that the queue does not block an arrival. The variance, specifically the uncondi-

tional variance, of the mixture distribution is computed as given by [66]

$$Var\left\{T_{qdelay}\right\} = \mathbb{E}\left\{Var\left\{T_{sum}^{i+1}/i\right\}\right\} + Var\left\{\mathbb{E}\left\{T_{sum}^{i+1}/i\right\}\right\}, \tag{6.14}$$

where

$$\mathbb{E}\left\{Var\left\{T_{sum}^{i}/N_{queue}\right\}\right\} = \sum_{i=0}^{Q_{size}-1} p_{occup}^{i} \times Var\left\{T_{sum}^{i}/i\right\},$$

$$Var\left\{\mathbb{E}\left\{T_{sum}^{i}/N_{queue}\right\}\right\} = \left(\sum_{i=0}^{Q_{size}-1} p_{occup}^{i} \times \mathbb{E}\left\{T_{sum}^{i}/i\right\}^{2}\right) - \overline{T}_{qdelay}^{2},$$

$$\text{and} \sum_{i=0}^{Q_{size}-1} p_{occup}^{i} = 1.$$

In Equation 6.14, $\mathbb{E}\left\{Var\left\{T_{sum}^{i+1}/i\right\}\right\}$ denotes the expected value of the conditional variances of the mixture components and $Var\left\{\mathbb{E}\left\{T_{sum}^{i+1}/i\right\}\right\}$ denotes the variance of the conditional means of the mixture components, where the respective quantities are conditional on the queue occupancies, denoted by i.

Let ψ_{load} denote the offered load to the queue which is defined as [65]

$$\psi_{load} = \frac{\mu_{arr}}{\mu_{svc}},\ 0 < \mu_{arr}, \mu_{svc} < 1. \tag{6.15}$$

Let P_{blk} and P_{empty} denote, respectively, the blocking probability of a queue and the probability that the queue is empty. In the queuing model, the departure rate is computed as given by

$$\mu_{dep} = \left\{ \begin{array}{ccc} \mu_{arr}\ (1 - P_{blk}) & ; & \psi_{load} > 1 \\ \mu_{arr} & ; & \psi_{load} \leq 1 \end{array} \right\}, \tag{6.16}$$

$$\text{where } P_{blk} = \left\{ \begin{array}{ccc} 1 - \frac{1}{P_{empty} + \psi_{load}} & ; & \psi_{load} > 1 \\ 0 & ; & \psi_{load} \leq 1 \end{array} \right\}. \tag{6.17}$$

The definition of blocking probability for $\psi_{load} > 1$, as given by Equation 6.17, is derived by Takagi [65] (Equation 8.26b of page 338) for a $Geom/Geom/1/K$ queue that uses the

delayed replacement model. Next, the proposed delay distribution model is validated by comparing against simulation results.

Validation of the mixture distribution model A computer simulation model of a $Geom/Geom/1/K$ queue is used to validate the proposed theoretical model. In the simulation model, the "late arrival with delayed replacement" assumption is implemented as follows. A potential departure event is processed first followed by a potential arrival event. In the case of a departure, the queue occupancy is updated only after processing an arrival event. The simulation model and the theoretical model are calibrated such that their respective time scales align. Before calibration, the simulation model is found to differ from the theoretical model by 1 slot time in all simulation runs. This is a consequence of the "late arrival" assumption, where the slot when the message arrives is not counted towards the delay calculation. This calibration is taken into consideration for all results pertaining to the queuing delay model. In order to validate the theoretical model for different arrival rates, queue sizes and mean service times, the simulation run is repeated for the following sets of parameters: (a) queue size of 14 and mean service time of 3 slots; (b) queue size of 14 and mean service time of 10 slots; (c) queue size of 5 and mean service time of 3 slots; and (d) queue size of 5 and mean service time of 10 time slots. Each simulation run is executed for 10^5 successful message departures from the queue and the queuing delay samples gathered from the simulation runs represent the delay experienced by a message conditioned on the fact that the message has successfully entered the queue.

The statistical properties of queuing delay, namely mean, standard deviation, skewness, and kurtosis, are used as metrics to compare the theoretical queuing delay PMF against the corresponding simulation PMF. Skewness and kurtosis of the PMF are computed as given

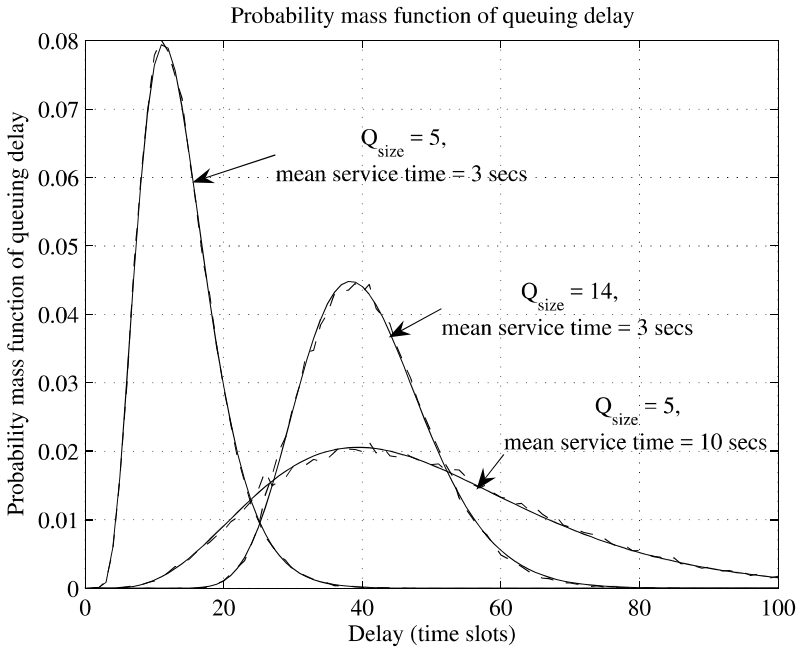


Figure 6.4: Validation of the theoretical probability mass function of queuing delay.

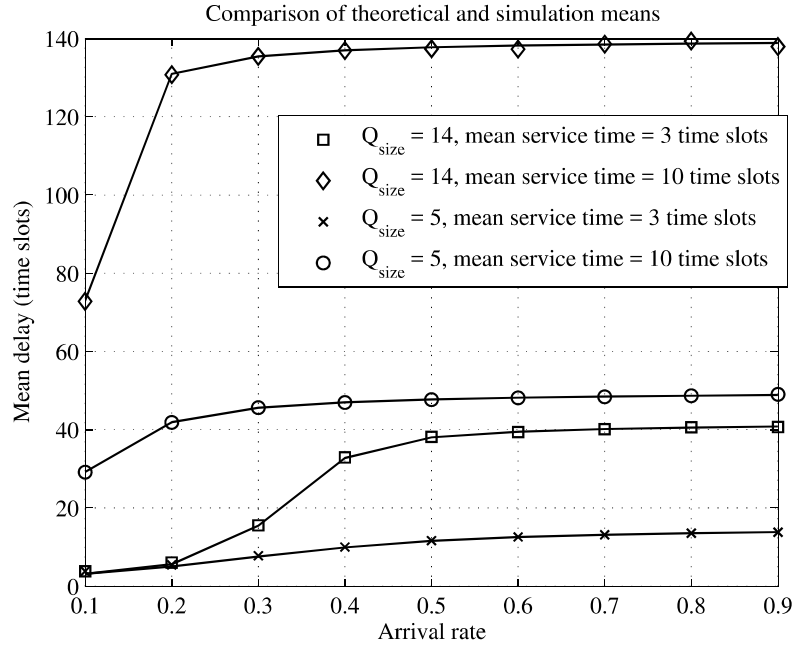


Figure 6.5: Validation of mean delay computed using the proposed queuing delay model.

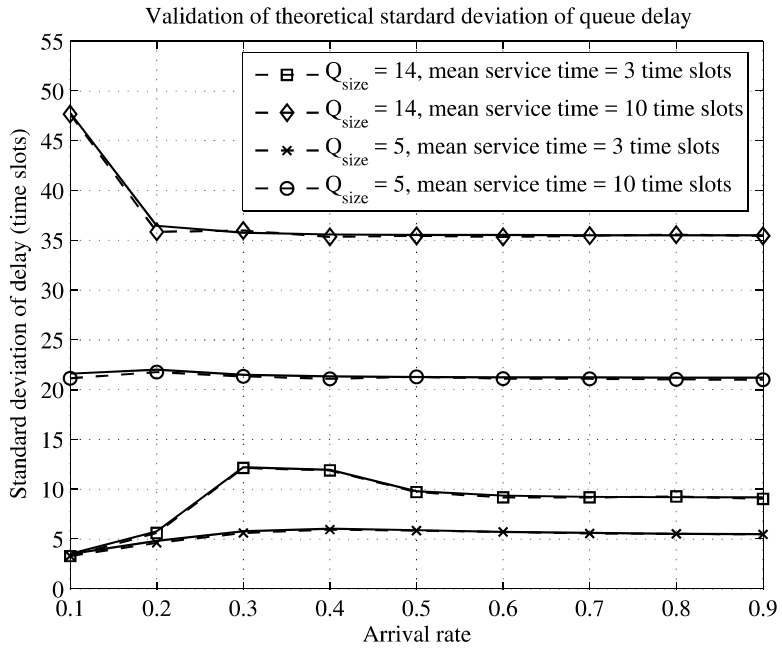


Figure 6.6: Validation of standard deviation of delay computed using the proposed queuing delay model.

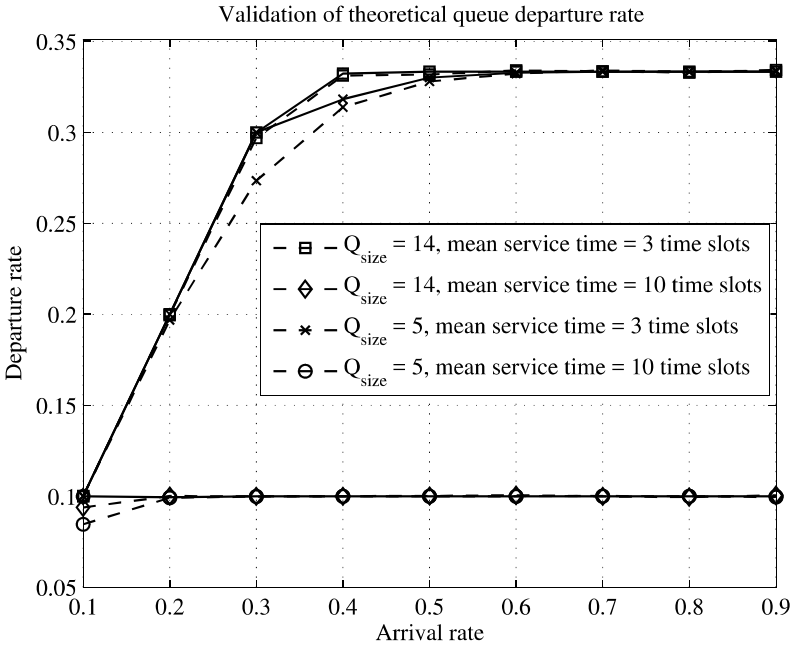


Figure 6.7: Validation of queue departure rate computed using the proposed queuing delay model.

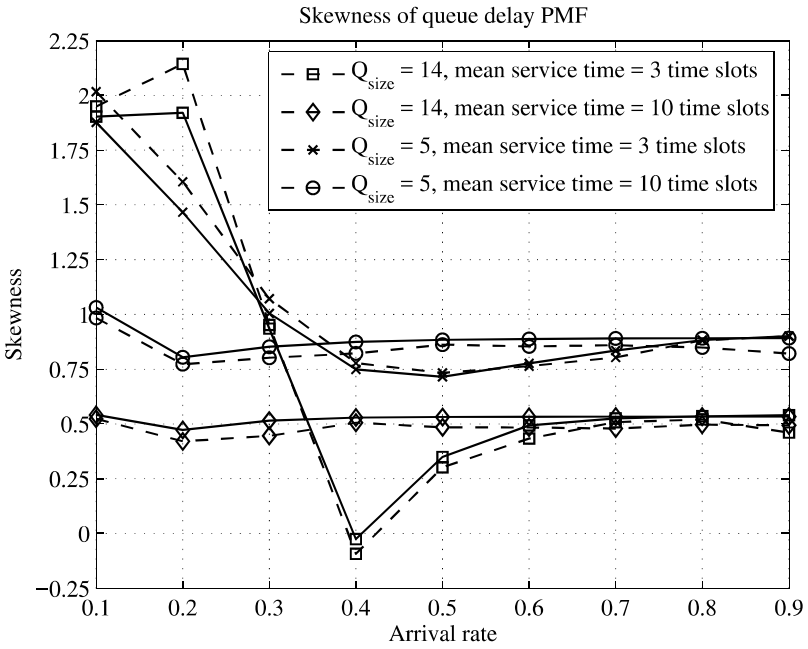


Figure 6.8: Comparison of skewness of simulation and theoretical queuing delay PMFs.

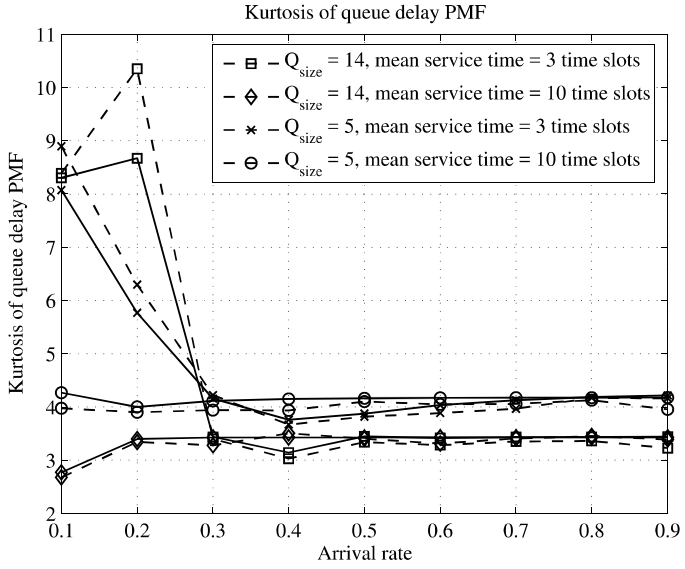


Figure 6.9: Comparison of kurtosis of simulation and theoretical queuing delay PMFs.

by

$$skewness = \mathbb{E}\left\{\left(\frac{t-\overline{T}_{qdelay}}{\sigma_{qdelay}}\right)^3\right\}, \tag{6.18}$$

$$\text{and } kurtosis = \frac{\mathbb{E}\left\{\left(t-\overline{T}_{qdelay}\right)^4\right\}}{\sigma^4_{qdelay}}, \tag{6.19}$$

where σ_{qdelay} is the standard deviation of queuing delay. The theoretical statistical properties are computed using Equations 6.3, 6.11, 6.14, 6.18, and 6.19. The theoretical departure rate is computed using Equation 6.16. The simulation properties are computed from queuing delay samples collected during the simulation runs. The properties are plotted for the 4 simulation runs. Figures 6.4 shows the theoretical and simulation PMF of T_{qdelay}. Figures 6.5, 6.6, 6.7, 6.8, 6.9 show, respectively, the mean, standard deviation, departure rate, skewness, and kurtosis, which are computed from simulation and the analytical equations. In these figures, dotted lines represent the simulation results while solid lines represent the theoretical values. The figures show that the analytically computed statistical properties closely match with those computed using simulation, thereby validating the theoretical model. The performance trends observed with variation in arrival rate and queue size, as shown in these figures, are well studied in queuing theory and are not of interest in this dissertation. The plots are solely used to validate the theoretical model against simulation results under different queuing conditions. Next, the statistics are computed for low values of mean service time. Figures 6.10, 6.11, 6.12, and 6.13 show, respectively, the mean, standard deviation, skewness, and kurtosis of queuing delay for short mean service times ranging from 1 time slot to 2 time slots. These results illustrate a limitation of the proposed queuing delay model. It is observed that the theoretical and simulation values do not match for higher moments of delay, namely, skewness and kurtosis, particularly at low mean service times. In addition, the kurtosis is very high for short mean service times. However, the theoretical and simulation results match closely at mean service times at and above 1.4 time slots.

Next, the delay distribution model presented in this section is used in deriving the com-

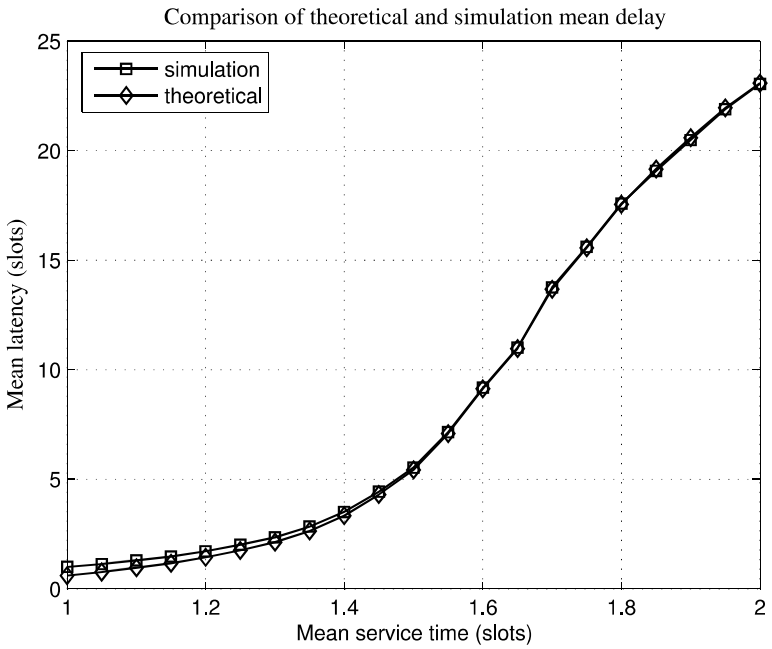


Figure 6.10: Validation of mean delay computed using the proposed queuing delay model for short mean service times.

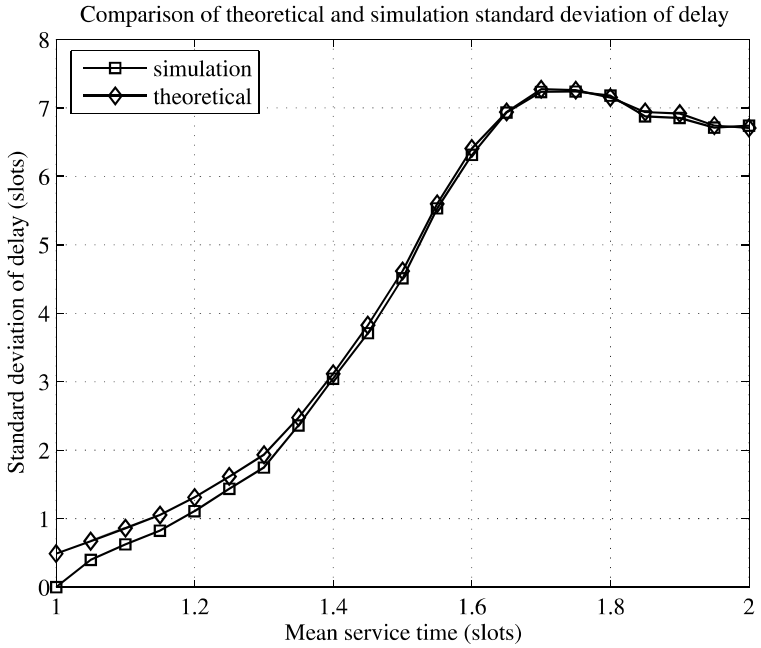


Figure 6.11: Validation of standard deviation of delay computed using the proposed queuing delay model for short mean service times.

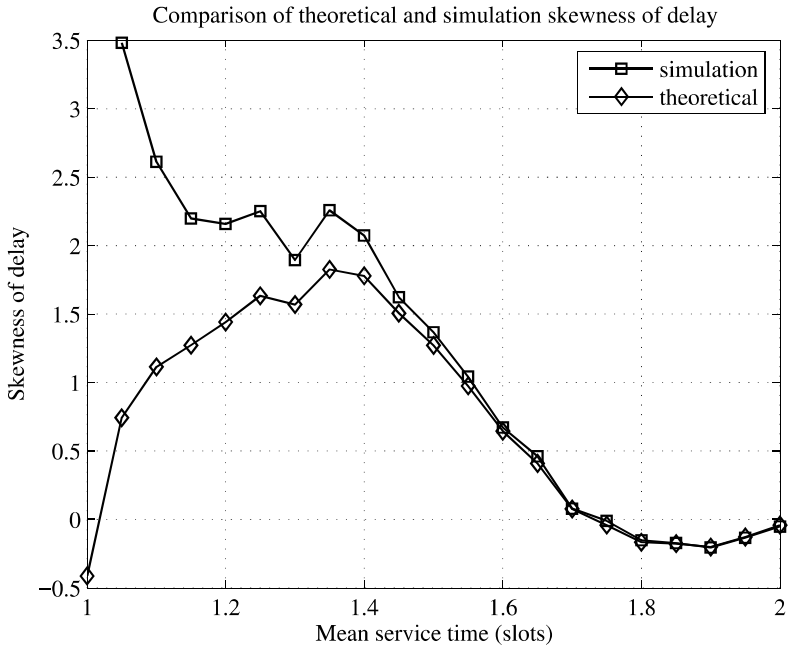


Figure 6.12: Comparison of skewness of simulation and theoretical queuing delay PMFs for short mean service times.

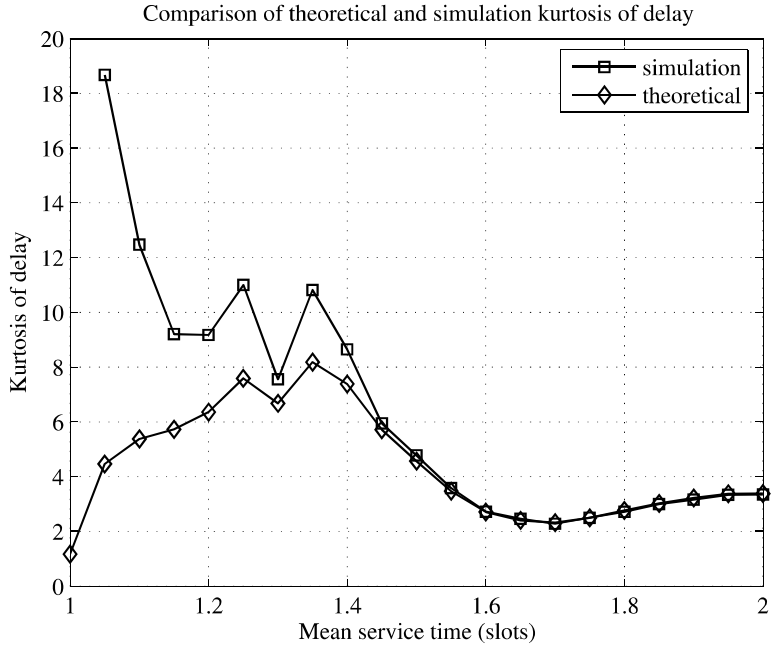


Figure 6.13: Comparison of kurtosis of simulation and theoretical queuing delay PMFs for short mean service times.

munication latency model. The next section describes the various components of the communication latency model, namely, a model of link connectivity (or probability of success of transmissions) for finite sized wireless networks and a model of communication latency experienced by request messages when they are buffered in a service requester's transmitter.

6.3 Communication Latency Model

This section presents a stochastic discrete-time model of communication latency experienced by messages in a TDMA wireless network. The model is developed for the following scenario. The wireless network comprises a finite number of transmitters, where each transmitter attempts to send one or more packets to a common receiver. The transmitters sharing a single wireless channel for communication with the receiver. In the context of the proposed WCCN queuing network model, this applies to a scenario where a finite number of service requester nodes are attempting to transmit their respective service request messages to a single service provider.

In a single-hop wireless link, the main factors that impact the communication latency experienced by a message include propagation delay experienced during every single transmission trial of the message and delay incurred due to re-transmissions. Typically, the transmitter node transmits a packet to the intended receiver node. Upon successfully receiving the packet, the receiver node sends a positive acknowledgment to the transmitter node indicating successful reception of the packet. When no acknowledgment is received or a negative acknowledgment is received, the transmitter assumes that the current trial of packet transmission has failed and then retransmits the same packet, sometimes after a certain backoff time. This procedure repeats either until the maximum number of transmission attempts is reached, or until a packet transmission trial is successful. The probability of success (or link connectivity) of a single packet transmission trial depends on channel characteristics such as path loss, fading, interference, noise, and traffic pattern. MAC layer interference, which

is caused by multiple access of the same wireless channel, is ignored in this model. This is simply because the model is derived for a typical TDMA network where transmissions from different nodes do not overlap in time.

The first step in deriving the communication latency model involves the characterization of distances between individual transmitters and the receiver, as discussed in Section 6.3.1. Using the distance distributions, the next step is to derive the probability of success of a single packet transmission trial in the presence of Rayleigh fading, path loss, and noise, as presented in Section 6.3.2. The probability of success, in turn, impacts the distribution of number of re-transmissions required for successful transmission of a message. In Section 6.3.3, the communication latency model is extended to account for buffering delays experienced by messages in TDMA networks. For the sake of convenience, the notation used in this chapter is slightly varied from the notation used in the system model of Chapter 3. All parameters in this chapter are not indexed by the link index ij, instead they are indexed by the index n of the nearest neighbor node.

6.3.1 Distance Distributions in a Binomial Point Process

A finite network of N_{nodes} transmitting nodes is modeled as a 2-dimensional binomial point process, where the points are uniformly distributed within a circle of radius R_c on a 2-dimensional plane and the receiver is located at the origin. Srinivasa et al. [67] show that the Euclidean distance, denoted by D_n, of the n^{th} nearest node from the origin in a 2-dimensional binomial point process follows a generalized beta distribution [67], as given by

$$f_{D_n}(r) = C_n\, \beta\left(\left(\frac{r}{R_c}\right)^2; a_n, b_n\right), \quad r \in [0, R_c], \quad 1 \le n \le N_{nodes} \tag{6.20}$$

$$\text{where } C_n = \frac{2}{R_c}\,\frac{\Gamma(n+0.5)\;\Gamma(N_{nodes}+1)}{\Gamma(n)\;\Gamma(N_{nodes}+1.5)}$$

$$\text{and } a_n = n+0.5, \quad b_n = N_{nodes} - n + 1. \tag{6.21}$$

Accordingly, the mean distance of the n^{th} nearest neighbor is given by [67]

$$\mathbb{E}\left(D_n\right) = R_c \times \frac{\Gamma\left(n+0.5\right)\ \Gamma\left(N_{nodes}+1\right)}{\Gamma\left(n\right)\ \Gamma\left(N_{nodes}+1.5\right)}. \tag{6.22}$$

The generalized beta distribution in Equation 6.20 can be defined as given by

$$\beta\left(\left(\frac{r}{R_c}\right)^2; a_n, b_n\right) = \frac{1}{B\left(a_n, b_n\right)}\left(\frac{r^2}{R_c^2}\right)^{a_n-1}\left(1-\frac{r^2}{R_c^2}\right)^{b_n-1}, \tag{6.23}$$

where $B\left(a_n, b_n\right)$ is a beta function. Accordingly, the distance distribution can be represented as given by [67]

$$f_{D_n}\left(r\right) = \frac{C_n}{B\left(a_n, b_n\right)}\ \left(\frac{r^2}{R_c^2}\right)^{a_n-1}\ \left(1-\frac{r^2}{R_c^2}\right)^{b_n-1}. \tag{6.24}$$

The normalization factor for the distance distribution of the form presented by Equation 6.24 can be computed as given by

$$\begin{aligned}
C_{norm} &= \int_0^{R_c} f_{D_n}\left(r\right)\ dr \\
&= \frac{C_n}{B\left(a_n, b_n\right)}\ \int_0^{R_c}\left(\frac{r^2}{R_c^2}\right)^{a_n-1}\ \left(1-\frac{r^2}{R_c^2}\right)^{b_n-1}\ dr \\
&= \frac{0.5\ C_n\ R_c}{B\left(a_n, b_n\right)}\int_0^1 t^{a_n-0.5-1}\ \left(1-t\right)^{b_n-1}\ dt \\
&\quad \left(\text{making a variable transformation} \frac{r^2}{R_c^2} \rightarrow t\right) \\
&= \frac{0.5\ C_n\ R_c}{B\left(a_n, b_n\right)}\ B\left(a_n-0.5, b_n\right) \text{(using the integral definition of beta function)} \\
\Rightarrow C_{norm} &= 1 \text{(evaluating beta functions in terms of gamma functions)}
\end{aligned}$$

In this dissertation, an alternate form of the distance distribution is presented. The second form is useful for analytical purposes, as it eases integration. According to Binomial

Theorem,

$$(1-x)^k = \sum_{i=0}^{k} (-1)^i \begin{pmatrix} k \\ i \end{pmatrix} x^i,$$

which is a finite series provided that $k \geq 0$ and k is an integer. By applying binomial theorem to Equation 6.24, the distance distribution can be represented in an alternate form as given by

$$\begin{aligned} f_{D_n}(r) &= \frac{C_n}{B(a_n, b_n)} \left(\frac{r^2}{R_c^2}\right)^{a_n-1} \sum_{i=0}^{b_n-1} (-1)^i \begin{pmatrix} b_n-1 \\ i \end{pmatrix} \left(\frac{r^2}{R_c^2}\right)^i \\ &= \frac{C_n}{B(a_n, b_n)} \sum_{i=0}^{b_n-1} (-1)^i \begin{pmatrix} b_n-1 \\ i \end{pmatrix} \left(\frac{r^2}{R_c^2}\right)^{a_n+i-1}, \end{aligned}$$

where $r^2/R_c^2 \leq 1$ and the power factor $b_n - 1$ $(= N_{nodes} - n)$ is always positive. The alternative form of the distance distribution can be normalized as given by

$$\begin{aligned} C_{norm} &= \int_0^{R_c} f_{D_n}(r)\ dr \\ &= \frac{C_n}{B(a_n, b_n)} \sum_{i=0}^{b_n-1} (-1)^i \begin{pmatrix} b_n-1 \\ i \end{pmatrix} \left(\frac{1}{R_c^2}\right)^{a_n+i-1} \int_0^{R_c} r^{2(a_n+i-1)}\ dr \\ &= \frac{0.5 \times R_c \times C_n}{B(a_n, b_n)} \sum_{i=0}^{b_n-1} (-1)^i \begin{pmatrix} b_n-1 \\ i \end{pmatrix} \frac{1}{a_n+i-0.5} \end{aligned} \tag{6.25}$$

After normalization, the alternative form of the distance distribution is given by

$$f_{D_n}(r) = \frac{\sum_{i=0}^{b_n-1} (-1)^i \begin{pmatrix} b_n-1 \\ i \end{pmatrix} \left(\frac{r^2}{R_c^2}\right)^{a_n+i-1}}{0.5 \times R_c \times \sum_{i=0}^{b_n-1} (-1)^i \begin{pmatrix} b_n-1 \\ i \end{pmatrix} \frac{1}{a_n+i-0.5}} \tag{6.26}$$

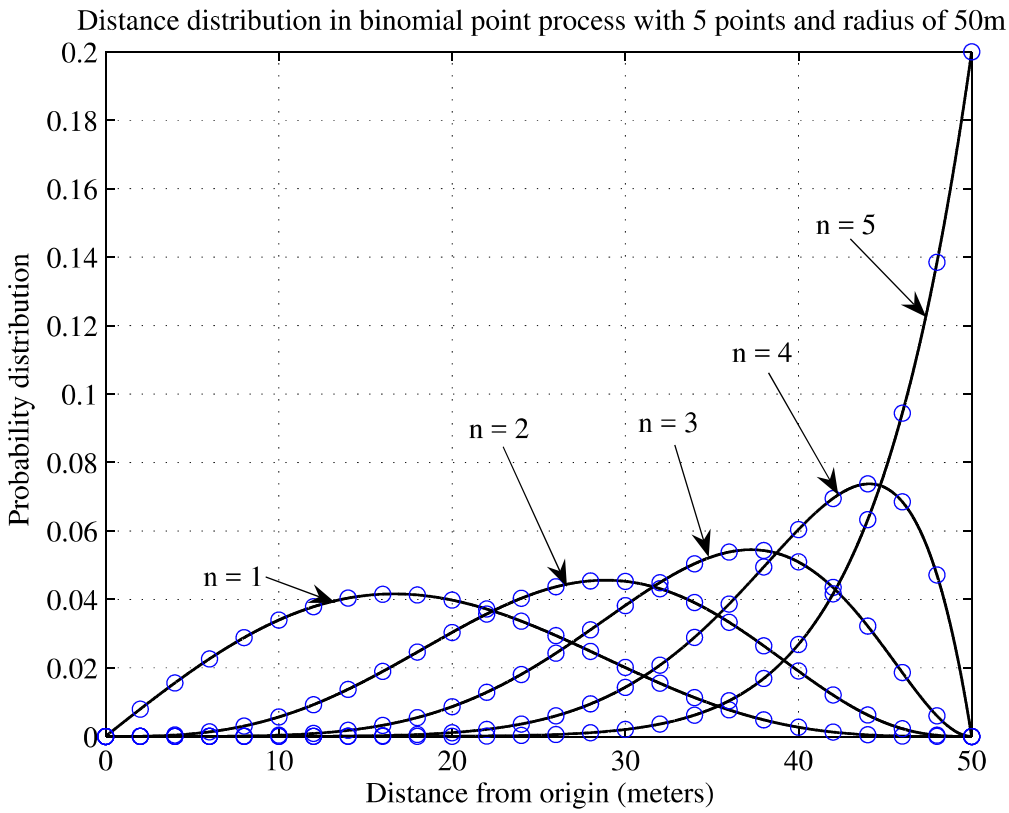


Figure 6.14: Distance distribution of n^{th} closest neighbor in a binomial point process comprising 5 points located within a radius of 50 meters. The solid line and circles plot, respectively, the original and alternate forms of the distance distributions.

Figure 6.14 verifies that the two forms of the distance distribution, as shown by Equations 6.24 and 6.26, yield the same results.

6.3.2 Link Connectivity in Finite Stochastic Networks in the Presence of Path Loss, Fading, and Noise

This section presents the derivation of transmission success probability (or link connectivity) experienced by messages in a finite network when the wireless channel is characterized by path loss, fading, and noise. Let transmitted SNR be denoted by γ_{tx}, signal transmit power measured at the output of a power amplifier be denoted by P_t, noise power be denoted

by P_{noise}, channel fading gain be denoted by g, distance between the n^{th} nearest node and intended receiver be denoted by r_n, path loss exponent be denoted by η_{pl}, minimum required received SNR threshold be denoted by SNR_{min}, receiver bandwidth be denoted by BW, and power amplifier efficiency be denoted by η_{amp}. The received SNR can be computed using Equations A.15 and A.16. Then, the probability of success for the n^{th} nearest node is given by

$$P^n_{success} = Pr\left\{\gamma_{tx}\ g\ {r_n}^{-\eta_{pl}}\ \left(\frac{\eta_{amp}\ P_{noise}}{G_1\ BW}\right) > SNR_{min}\right\}, \tag{6.27}$$

$$\text{where } \gamma_{tx} = \frac{P_t}{P_{noise}}. \tag{6.28}$$

In Equation 6.27, G_1 is a constant which is defined in Equations A.19. For the sake of convenience, let

$$\theta = \frac{SNR_{min}\ G_1\ BW}{\eta_{amp}\ P_{noise}}. \tag{6.29}$$

When the node locations are modeled by a binomial point process and channel fading gain follows the distribution given by Equation 3.10 with $\sigma_g = 1$, the probability of success experienced by the n^{th} closest neighbor node is given by

$$\begin{aligned}
P^n_{success} &= Pr\left\{\gamma_{tx}\ g\ {r_n}^{-\eta_{pl}} > \theta\right\} \\
&= 1 - Pr\left\{g \le \frac{\theta\ {r_n}^{\eta_{pl}}}{\gamma_{tx}}\right\} \\
&= \mathbb{E}_{D_n}\left\{exp\left(-\frac{\theta\ {r_n}^{\eta_{pl}}}{2\ \gamma_{tx}}\right)\right\} \\
&= \int_0^{R_c} exp\left(-\frac{\theta\ {r_n}^{\eta_{pl}}}{2\ \gamma_{tx}}\right)\ f_{D_n}(r)\ dr \\
&= \frac{C_n}{C_{norm}\ B\left(a_n, b_n\right)} \times \\
&\quad \sum_{i=0}^{b_n-1} (-1)^i \begin{pmatrix} b_n - 1 \\ i \end{pmatrix} \left(\frac{1}{{R_c}^2}\right)^{a_n+i-1} \int_0^{R_c} exp\left(-\frac{\theta\ {r_n}^{\eta_{pl}}}{2\ \gamma_{tx}}\right)\ r^{2(a_n+i-1)}\ dr
\end{aligned}$$

$$
\begin{aligned}
&= \frac{C_n}{C_{norm}\, B\left(a_n, b_n\right)} \times \\
&\sum_{i=0}^{b_n-1} (-1)^i \begin{pmatrix} b_n - 1 \\ i \end{pmatrix} \left(\frac{1}{R_c^2}\right)^{a_n+i-1} \left(\frac{2\,\gamma_{tx}}{\theta}\right)^{\frac{1}{\eta_{pl}}(2a_n+2i-1)} \frac{1}{\eta_{pl}} \\
&\int_0^{\frac{\theta\, R_c^{\eta_{pl}}}{2\,\gamma_{tx}}} exp\left(-t\right)\; t^{\frac{1}{\eta_{pl}}(2a_n+2i-1)-1}\, dt \\
&\left(\text{making variable transformation } \frac{\theta\, r_n^{\eta_{pl}}}{2\,\gamma_{tx}} \rightarrow t\right) \\
\Rightarrow P^n_{success} &= \frac{C_n}{C_{norm}\, B\left(a_n, b_n\right)} \frac{1}{\eta_{pl}} \times \sum_{i=0}^{b_n-1} \left\{ (-1)^i \begin{pmatrix} b_n - 1 \\ i \end{pmatrix} \left(\frac{1}{R_c^2}\right)^{a_n+i-1} \right. \\
&\left. \left(\frac{2\,\gamma_{tx}}{\theta}\right)^{\frac{1}{\eta_{pl}}(2a_n+2i-1)} \gamma\left(\frac{1}{\eta_{pl}}\left(2a_n + 2i - 1\right), \frac{\theta\, R_c^{\eta_{pl}}}{2\,\gamma_{tx}}\right) \right\}, \qquad (6.30)
\end{aligned}
$$

where C_{norm} is given by Equation 6.25 and $\gamma\left(\cdot\right)$ represents the lower incomplete gamma function. The limitations of this analytical model are discussed next. In Equation 6.30, the arguments of the incomplete gamma function are always positive. The maximum value that the first argument takes is $2N_{nodes}/\eta_{pl}$. When $2N_{nodes}/\eta_{pl} \rightarrow \infty$ or $2N_{nodes}/\eta_{pl} \rightarrow 0$, the incomplete gamma function, particularly the associated gamma function with argument $2N_{nodes}/\eta_{pl}$, tends to ∞. This can typically occur for extreme channel conditions with either very low or very high values of path loss exponent, and very large network sizes. It is recommended to use $\eta_{pl} > 0.2$ provided that the second argument of the incomplete gamma function holds a reasonable value. When the first argument of the incomplete gamma function takes an acceptable value, the function fails when the value of the second argument is very large. This can occur when the signal is transmitted with a very low transmit power. Furthermore, the computer software should be able to handle very large numbers encountered while computing the factorial of $N_{nodes} \geq 15$. Computer simulation can be used in cases where the analytical model fails. Figure 6.15 shows that the theoretical probability of success, which is computed using Equation 6.30, matches with the simulated probability of success. The results are generated for a network of 15 nodes located within a cluster radius of 200 meters. The transmit SNR is set to 135 dB (transmitted signal power of 16

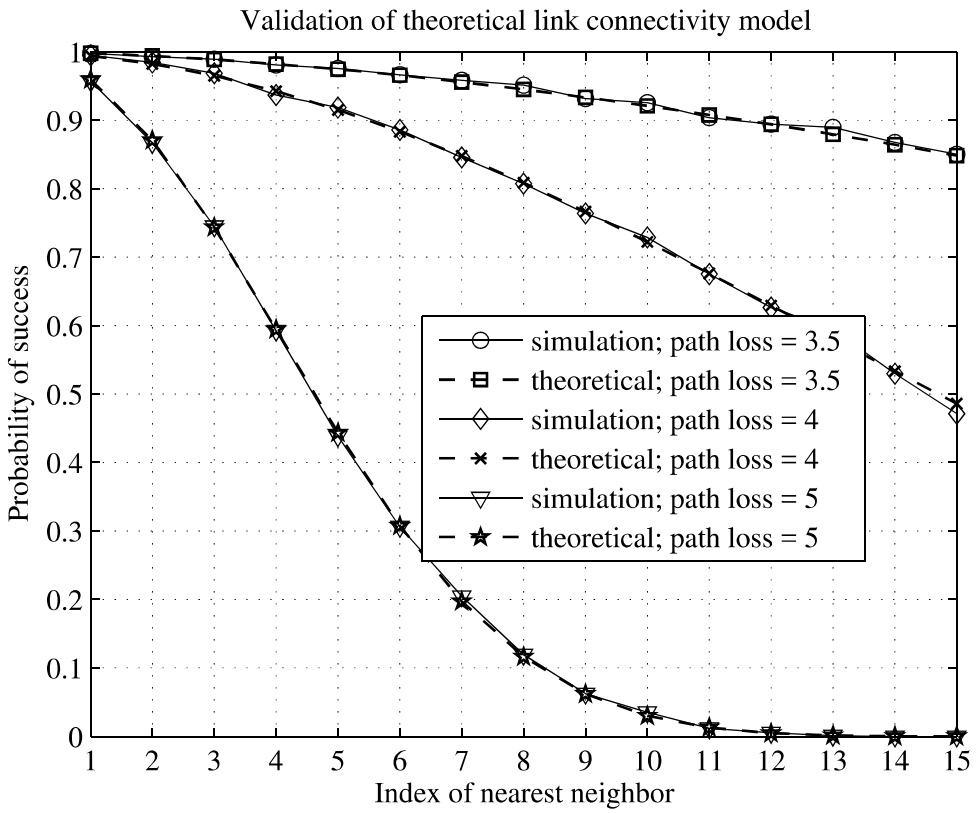


Figure 6.15: Validation of the theoretical link connectivity model.

dBm) and the receiver SNR threshold is set to 15 dB (received signal power of -104 dBm). The values of the remaining parameters are taken from Table 3.1.

Link connectivity, as computed by Equation 6.30, is analyzed for a network comprising 15 nodes located in an area of radius 250 meters, where each node has a receiver SNR threshold of 15 dB (received signal power of -104 dBm), transmit SNR is set to 135 dB (transmitted signal power of 16 dBm), and channel path loss exponent is 4. Figure 6.16 plots the probability of success for all nodes in the network for different network sizes. Interestingly, it is observed that the probability of success for the n^{th} nearest neighbor varies with network size. This is an artifact of the binomial point process model. For a particular network cluster radius, the variance of the distance distribution of every node in the network decreases as the network size increases [67]. When there are only 2 nodes in the network,

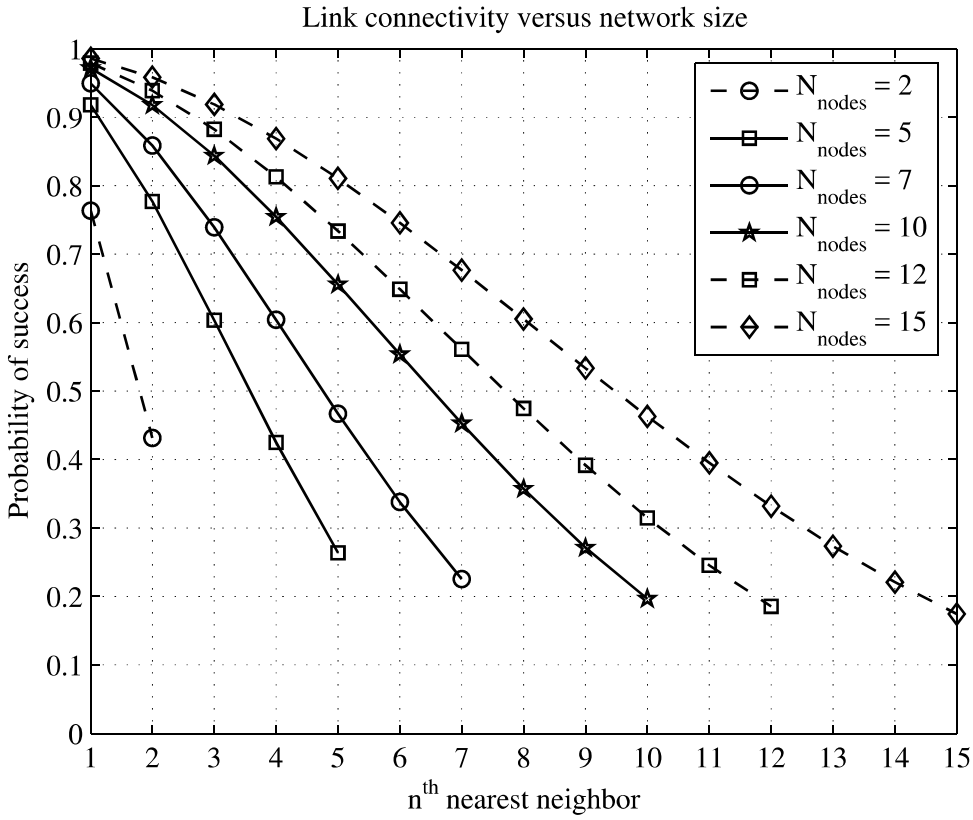


Figure 6.16: Link connectivity for different network sizes.

for instance, the location of each node can vary from being close to the center of the circular cluster to being close to the cluster boundary. In addition, the mean distance of a node from origin also varies with the network size. These factors cause the average probability of success to be reduced drastically. As the network size increases, on the other hand, the locations of the individual nodes are more restricted, thereby resulting in a higher probability of success.

The probability of success achieved for different channel path-loss exponents is plotted in Figure 6.17. It is evident that the probability decreases with increase in path-loss exponent. It gets worse for nodes that are located further away from the 8^{th} nearest neighbor as the channel path-loss exponent is increased to 4. The probability of success curve has a low slope for good channel conditions and is increasingly non-linear as channel conditions degrade. In

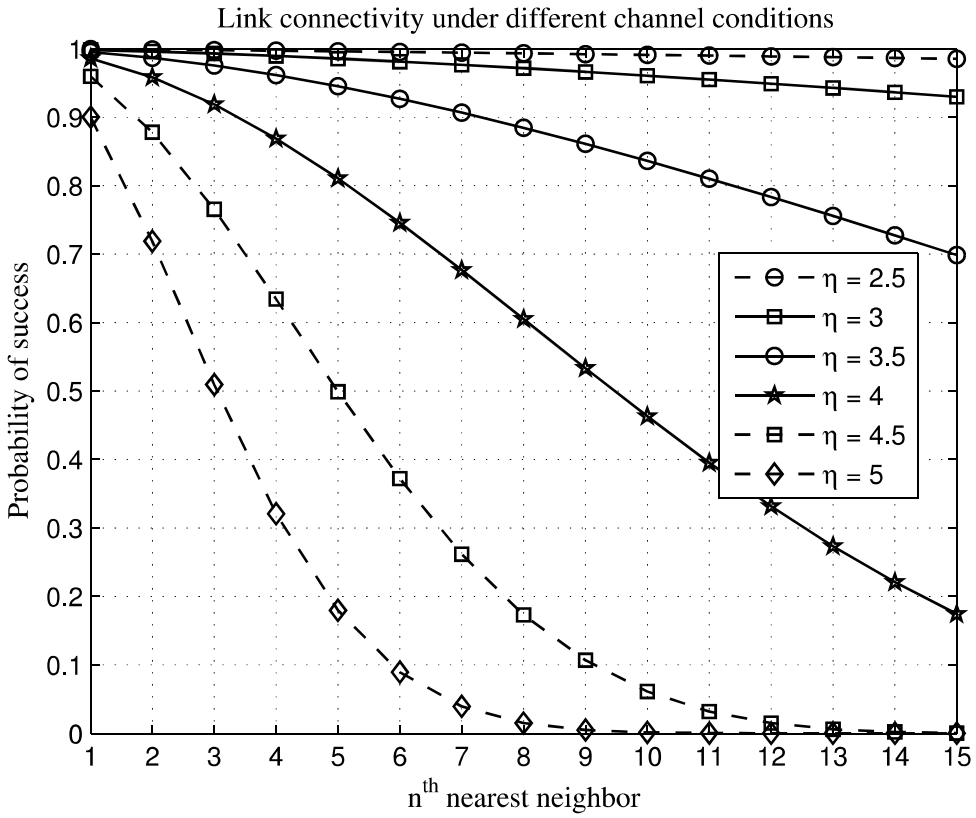


Figure 6.17: Link connectivity achieved under different channel path exponents.

the former case, all nodes in the network enjoy nearly identical link connectivity. Similar trends are observed from Figure 6.18 which plots the probability of success for different transmitter SNRs. A sufficiently high transmit power ensures that all nodes in the network experience nearly the same probability of success regardless of distance.

6.3.3 Delay Distribution in TDMA Networks in the Presence of Packet Buffering, Path Loss, Fading, and Noise

This section derives the PMF of latency experienced by an arbitrary message transmitted in a TDMA network. The model is evolved in stages, by first considering the delay distribution of a single message and extending the model to incorporate the impact of buffering delays.

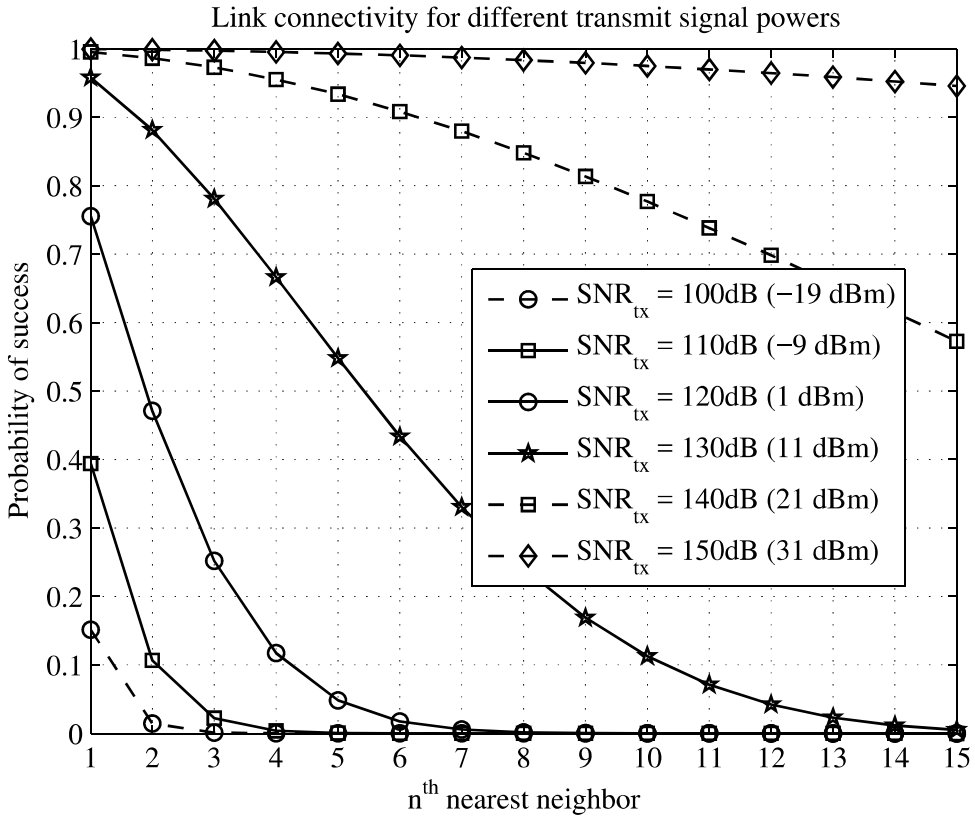


Figure 6.18: Link connectivity achieved with different transmitter SNRs.

Delay distribution of a single message The mean communication latency per message is given by Equation 3.30. Consider a scenario where a slot-based (or discrete time) communication scheme is employed. Assume that the slot time is equal to the packet propagation delay T_{pktp}, which is given by Equation 3.26. The mean communication latency per message experienced by the n^{th} nearest neighbor, denoted by $\overline{T}^{n}_{pktx}$, is expressed in terms of the slot time. Equation 3.30 can be adapted to represent the relationship between mean communication latency per message and link connectivity, as given by

$$\overline{T}^{n}_{pktx} = \frac{1}{P^{n}_{success}}, \tag{6.31}$$

where $P^{n}_{success}$ is the probability of a successful message transmission in an arbitrary time slot as given by Equation 6.30. This inverse relationship between mean delay and link

connectivity is also consistent with past work [68]. Haenggi [68] derives a model of the "local delay" experienced in successfully transmitting a single packet between two neighboring nodes, where the location of the nodes is modeled using a Poisson Point Process.

In the case of a slot-based communication scheme that allows re-transmission of erroneously received messages, the total communication latency experienced by an arbitrary message follows a geometric distribution, as given by

$$Pr\left\{T^n_{pktx} = t\right\} = (1 - P^n_{success})^{t-1} \times P^n_{success}, \text{ where } t = 1,\ 2,\ 3, \ldots. \tag{6.32}$$

In Equation 6.32, it is assumed that the probability of success is independent of the slot time when a packet is transmitted, i.e., message transmissions on different time slots are stochastically independent of each other. When a finite limit is placed on the number of re-transmissions, the latency per message is a mixture distribution.

The communication latency model is extended to a TDMA communication scheme, where all transmitting nodes share a common communication channel in order to communicate with a common receiver and the channel is divided into time slots of equal duration. The slot allocation to nodes is fixed and is performed in a round-robin manner. Message size is fixed and equal for all nodes in the network. Furthermore, the TDMA slot time is fixed and is independent of the number of nodes in the network. Thus, increasing the number of nodes increases the delay experienced by a node between successive re-trials of a message. The delay between successive re-trials of a message is equal to $N_{nodes} - 1$, where it is assumed that a message transmission attempt is considered complete only at the end of a time slot. With these assumptions, the latency model presented in Equation 6.32 can be extended to the TDMA communication scheme to measure the latency experienced by a message that arrives in the slot prior to the one scheduled for its first transmission attempt. The total latency experienced by an arbitrary message in a TDMA network follows a time-scaled

N_1	N_2	N_3	N_4	N_5	N_1	N_2	N_3	N_4	N_5	N_1	N_2	N_3	N_4
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Figure 6.19: An example round-robin transmission schedule of a TDMA network.

geometric distribution, as given by

$$\begin{aligned} Pr\left\{T^n_{pktx} = t\right\} &= (1 - P^n_{success})^{\frac{t+T_s-1}{T_s}-1} \times P^n_{success}, \\ \text{where } t &= 1, 1+T_s,\ 1+2T_s,\ 1+3T_s \ldots. \end{aligned} \tag{6.33}$$

In Equation 6.33, T_s represents the time scaling factor which is expressed in multiples of the TDMA slot time and $T_s = N_{nodes}$. Let X represent a geometrically distributed random variable. Scaling the random variable in time results in a new random variable given by $Y = k\ X$, where k is a scaling factor. Y follows a time-scaled geometric distribution. Ufuktepe [69] provides more information on time-scaled distributions. Next, the latency model presented in Equation 6.33 is extended to incorporate the effect of buffering.

Delay distribution in the presence of buffering In the WCCN queuing network model, TDMA buffering delays can occur in the forward link when a service requester transmits its service requests to a service provider. Consider a TDMA network comprising 5 nodes sharing a common channel to communicate with a receiver node. As shown in Figure 6.19, the slot allocation to the transmitting nodes in the network is fixed and is done in a round-robin manner, i.e., node N_4 transmits in slots 4, 9, 14 and so on. If a tagged message enters N_4's forward link communication queue at the first slot in the TDMA schedule (i.e., time slots 1, 6, 11, and so on), the total latency experienced by the message has the following sample space: $\{2+1,\ 2+6,\ 2+11, \ldots\}$. Similarly, if a message arrives at the second slot in the TDMA schedule, the sample space of latency is given by: $\{1+1,\ 1+6,\ 1+11, \ldots\}$. In every element in the sample space, the first term in the sum, denoted by T_δ, indicates the

number of slots that the tagged message has to wait until it is node N_4's turn to transmit a message from its forward link communication queue. The second term in the sum, denoted by $T^i_{pktotal}$, captures the number of slots that the tagged message has to wait in addition to T_δ until it is successfully transmitted to the receiver. $T^i_{pktotal}$ is computed as the sum total of the transmission delays of all messages located ahead of the tagged message in the queue and the transmission delay of the tagged message itself. T^{ij}_{qpk} denotes the time taken to successfully transmit the j^{th} packet in node N_i's transmitter queue after it begins processing the message. Accordingly, the total forward link communication latency, denoted by T^i_{cmfl}, experienced by an arbitrary message in the i^{th} nearest neighbor node can be modeled as given by

$$T^i_{cmfl} = T_\delta + T^i_{pktotal}, \tag{6.34}$$

$$\text{where } T^i_{pktotal} = \sum_{j=1}^{N_{queue}+1} T^{ij}_{qpk}. \tag{6.35}$$

T^i_{cmfl} and its delay components, namely T_δ and $T^i_{pktotal}$, are implicitly conditioned on the fact that a message that arrives at the queue is able to successfully enter the queue. This is simply because it is of no interest to capture the delay of a message that is blocked from entering the queue. T_δ is assumed to be independent of both the queue occupancy and the queue departure process. $T^i_{pktotal}$ is dependent on the departure process. In this model, it is also assumed that $T^i_{pktotal}$ is independent of the slot when the message arrives at the queue, as well as independent of the slot assignment. As a consequence, T_δ and $T^i_{pktotal}$ are treated as being independent of each other in the model. Ideally, the queue occupancy N_{queue} at the time when the tagged message enters the queue is dependent on the TDMA slot when the message arrives at the queue, mainly because message departures from a node's queue can occur only at the TDMA slot allocated to that node while message arrivals occur in every TDMA slot.

The derivation of the PMF of $T^i_{pktotal}$ is similar to the derivation of the PMF of queuing delay as shown in Section 6.2, with the exception that T^{ij}_{qpk} is characterized by a time-

scaled geometric distribution as given by Equation 6.32. Let Q^i_{cmfl} denote the size of node N_i's forward link communication queue and p^{ij}_{cmflq} denote the probability that there are j messages in node N_i's forward link communication queue. Accordingly, the PMF of $T^i_{pktotal}$, denoted by $f_{T^i_{pktotal}}$, is given by

$$f_{T^i_{pktotal}}(t) = \sum_{j=0}^{Q^i_{cmfl}-1} p^{ij}_{cmflq}\, f_{T^{i,j+1}_{sum}}\left(t;\ j+1;\ P^i_{success}\right), \tag{6.36}$$

$$\text{where } T^{i,j}_{sum} = \sum_{k=1}^{j} T^{ik}_{qpk}, \tag{6.37}$$

$$f_{T^{i,j}_{sum}}\left(x;\ j;\ P^i_{success}\right) = \begin{pmatrix} \frac{(x+T_s-1)}{T_s} - 1 \\ j-1 \end{pmatrix} \left(P^i_{success}\right)^j \left(1-P^i_{success}\right)^{\frac{(x+T_s-1)}{T_s}-j}, \tag{6.38}$$

$$t = 1,\ 1+T_s,\ 1+2T_s, \ldots,$$

$$x = 1+(j-1)\,T_s,\ 1+j\,T_s,\ 1+(j+1)\,T_s,\ 1+(j+2)\,T_s,\ \ldots.$$

The PMF of T_δ is derived next.

Let S_{arr} denote the particular slot in a TDMA schedule period when a tagged message arrives at the queue. S_{arr} is a random variable with sample space given by $\{1,\ 2, \ldots,\ T_s\}$. S_{trial} denotes the TDMA slot when the Bernoulli trial corresponding to the arrival process of the tagged message begins. S_{trial} can take one of the following values: $\{1,\ 2, \ldots,\ T_s\}$, each with a probability given by $1/T_s$. With reference to the TDMA schedule shown in Figure 6.19, for instance, a message is said to arrive in TDMA slot 3 if it arrives in any one of the following discrete time slots: 3, 8, 13, ... Then, the probability that the message arrives in slot S_{arr} for a given S_{trial}, is given by

$$Pr\left\{S_{arr} = s_1 / S_{trial} = s_2\right\} = \sum_{i=0}^{\infty} Pr\left\{N_{trials} = \epsilon + i\,T_s\right\}, \tag{6.39}$$

$$\text{where } Pr\left\{N_{trials} = x\right\} = (1-\mu_{arr})^{x-1}\ \mu_{arr};\ x = 1,\ 2,\ 3, \ldots. \tag{6.40}$$

$$\Rightarrow Pr\left\{S_{arr} = s_1 / S_{trial} = s_2\right\} = \tilde{\mu}_{arr}^{\epsilon-1}\ \mu_{arr}\ \sum_{i=0}^{\infty} \left(\tilde{\mu}_{arr}\right)^{iT_s}, \tag{6.41}$$

$$= \tilde{\mu}_{arr}^{\epsilon-1}\ \mu_{arr}\ \frac{1}{1-\left(\tilde{\mu}_{arr}\right)^{T_s}};\ \text{given that}\ \left(\tilde{\mu}_{arr}\right)^{T_s} < 1, \tag{6.42}$$

$$\text{where}\ \tilde{\mu}_{arr} = 1 - \mu_{arr}. \tag{6.43}$$

In Equation 6.39, N_{trials} denotes the number of Bernoulli trials associated with the message arrival process. ϵ is a positive integer which is a function of S_{arr} and S_{trial}. Referring again to the TDMA schedule shown in Figure 6.19, ϵ takes values 1, 5, 4, 3, 2, respectively, for $s_1 = 1$ and $s_2 = 1,\ 2,\ 3,\ 4,\ 5$. By averaging over all possible values of S_{trial}, the marginal PMF of S_{arr} is given by

$$Pr\left\{S_{arr} = s_1\right\} = \frac{1}{T_s} \sum_{\epsilon=1}^{T_s} \frac{\tilde{\mu}_{arr}^{\epsilon-1}\ \mu_{arr}}{1-\left(\tilde{\mu}_{arr}\right)^{T_s}} \tag{6.44}$$

$$= \frac{\mu_{arr}}{T_s\ \tilde{\mu}_{arr}\ \left(1-\left(\tilde{\mu}_{arr}\right)^{T_s}\right)}\ \sum_{\epsilon=1}^{T_s} \left(\tilde{\mu}_{arr}\right)^{\epsilon} \tag{6.45}$$

$$\Rightarrow Pr\left\{S_{arr} = s_1\right\} = \frac{1}{T_s}. \tag{6.46}$$

Equation 6.46 shows that the probability $Pr\left\{S_{arr} = s_1\right\}$ is no longer dependent on ϵ. The sample space of T_δ is given by $\{0,\ 1,\ 2, \ldots,\ T_s - 1\}$, where the corresponding probabilities have a one-to-one mapping with the probabilities of S_{arr}. Thus, T_δ is uniformly distributed in the interval $[0, T_s - 1]$, i.e.,

$$T_\delta \sim \text{DiscreteUniform}\left(\frac{1}{T_s}\right). \tag{6.47}$$

Finally, the PMF of T^i_{cmfl}, denoted by $f_{T^i_{cmfl}}$, is computed as given by

$$f_{T^i_{cmfl}} = f_{T_\delta} \otimes f_{T^i_{pktotal}}. \tag{6.48}$$

Let ψ^i_{load}, μ^i_{arr}, and μ^i_{dep} denote, respectively, the offered load, arrival rate, and departure

rate of node N_i's forward link communication queue. The departure rate of N_i is computed using Equation 6.16, with the exception that the offered load is computed as given by

$$\psi^i_{load} = T_s \frac{\mu^i_{arr}}{P^i_{success}},\ 0 < \mu^i_{arr} < 1. \tag{6.49}$$

The network throughput or the net departure rate of the TDMA network in the forward link, denoted by ρ_{net}, is computed as given by

$$\rho_{net} = \sum_{i=1}^{N_{nodes}} \mu^i_{dep}. \tag{6.50}$$

The theoretical PMF model and the throughput model are validated against simulation results next.

Validation of the buffered TDMA latency model The TDMA latency PMF model is validated against simulation results w.r.t. the statistical properties of PMF, namely CDF, mean, standard deviation, skewness, and kurtosis. The simulation model discussed in Section 6.2 is extended to include the forward link communication queues of all transmitting nodes in a TDMA network, where the departure processes in the individual queues follow the TDMA schedule of their respective nodes. The simulation model is run until 10^5 messages depart from the farthest neighbor node in the network. The simulation PMF of communication latency and its statistical properties are computed from communication latency samples that are gathered using the simulation model. The theoretical PMF is computed as given by Equation 6.48 and it is, in turn, used to compute the statistical properties. The simulation model is run for a TDMA network comprising 10 transmitting nodes, where the node locations are modeled using a binomial point process that spans a circular network cluster area with a radius of 200 meters with the receiver located at the origin of the circle. Transmitter SNR at the individual nodes is set at 130 dB (i.e., a transmitted signal power of 11 dBm) and the channel path loss exponent is set to 4. The size and arrival rate of the forward link communication queue in each node is set to 14 and 0.6, respectively.

Figure 6.20 plots the simulation and theoretical PMFs of latency experienced by the 4^{th} nearest neighbor node N_4. In these PMFs, the resolution of communication latency is set to the highest possible resolution of 1 time slot. Figure 6.21 plots the latency CDFs for nodes N_1, N_4, and N_6 with the same time resolution. The PMFs and CDFs are plotted for a lower resolution of 10 time slots in Figures 6.22 and 6.23, respectively. When using a high resolution for communication latency in the PMFs, the simulation PMF exhibits saw-tooth like patterns, as shown in Figure 6.20, due to the following reasons. First, it is a known fact that there is a tradeoff between the resolution of the x-axis and that of the y-axis of a histogram. Smith [70] explains that a histogram that is constructed using a large number of bins is prone to simulation noise because only a few samples are collected in each bin. Second, the departure and arrival processes of a node's transmission queue operate at different time scales. The departure process is invoked every T_s time slots whereas the arrival process is invoked every time slot. As a result, the queue occupancy as observed by an incoming message can depend on the TDMA slot when the message enters the queue, especially when using a high arrival rate. The theoretical PMF model, however, does not take this dependence into consideration. When plotting the PMFs at a lower resolution of T_s time slots, the saw-tooth pattern and the effects of simulation noise are no longer observed. In this case, the theoretical and simulation PMFs match closely when computed for different nodes in the network, as shown in Figures 6.22. Figures 6.21 and 6.23 show that the simulation and theoretical CDFs match closely, both at low and high latency resolutions. The figures also show that the CDFs shift to the right as one considers transmitter nodes which are located farther away from the receiver.

Next, the theoretical PMF model is validated in terms of statistical properties for various network sizes. For this purpose, the simulation model is run for larger networks comprising 20 and 30 nodes, respectively, keeping all other simulation parameters the same. Figures 6.24, 6.25, 6.26, and 6.27 show, respectively, the mean, standard deviation, skewness, and kurtosis for different network sizes as computed from the simulation PMF and the theoretical PMF. In these results, the resolution of communication latency is set to 1 slot. Figures 6.24 and

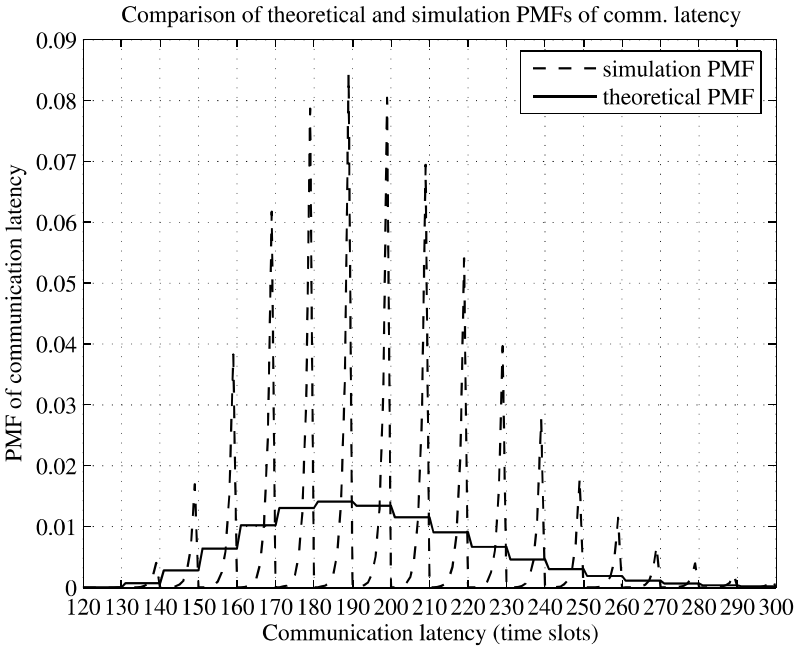


Figure 6.20: Comparison of the theoretical and simulation PMFs of forward link communication latency experienced by node N_4 in a TDMA network comprising 10 nodes, where the PMFs have a resolution of 1 time slot.

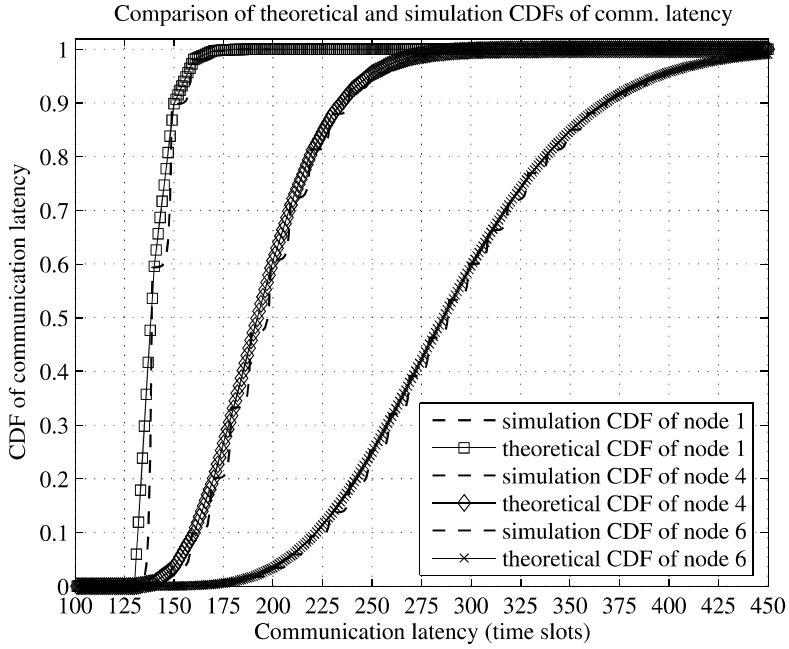


Figure 6.21: Comparison of the theoretical and simulation CDFs of forward link communication latency experienced by nodes in a TDMA network comprising 10 nodes, where the CDFs have a resolution of 1 time slot.

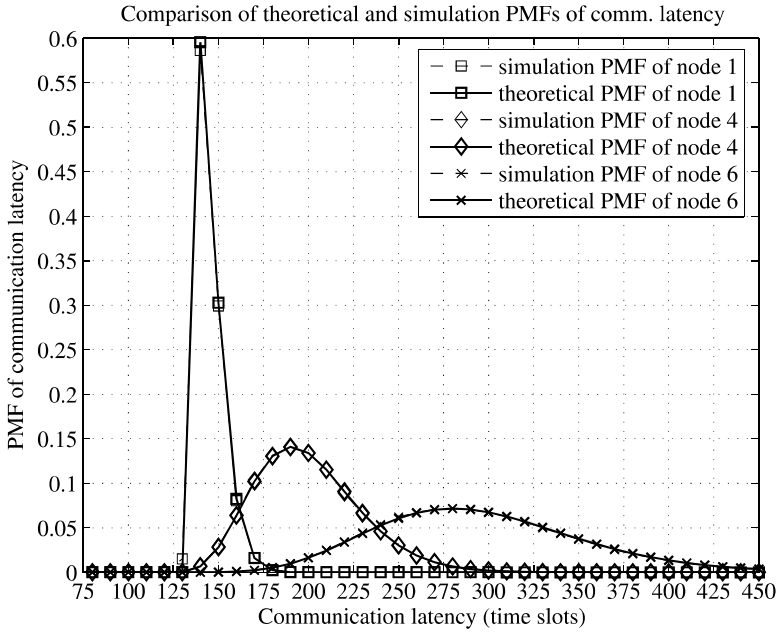


Figure 6.22: Comparison of the theoretical and simulation PMFs of forward link communication latency experienced by nodes in a TDMA network, where the PMFs have a resolution of 10 time slots.

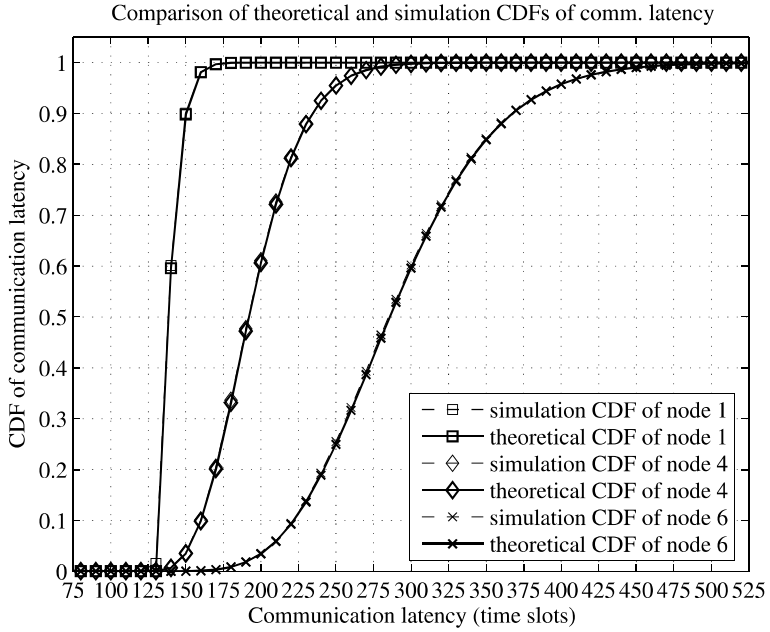


Figure 6.23: Comparison of the theoretical and simulation CDFs of forward link communication latency experienced by nodes in a TDMA network, where the CDFs have a resolution of 10 time slots.

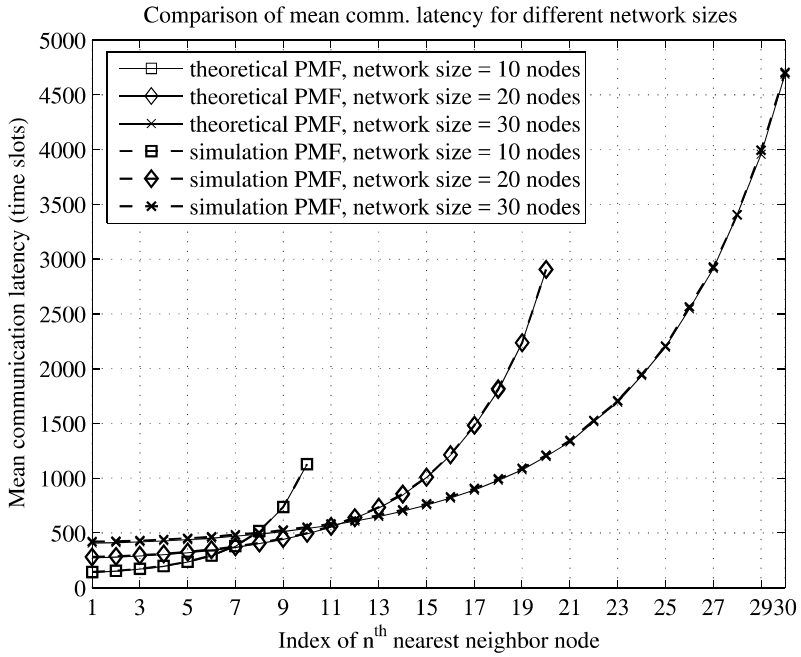


Figure 6.24: Comparison of the theoretical and simulation mean of TDMA forward link communication latency for different network sizes.

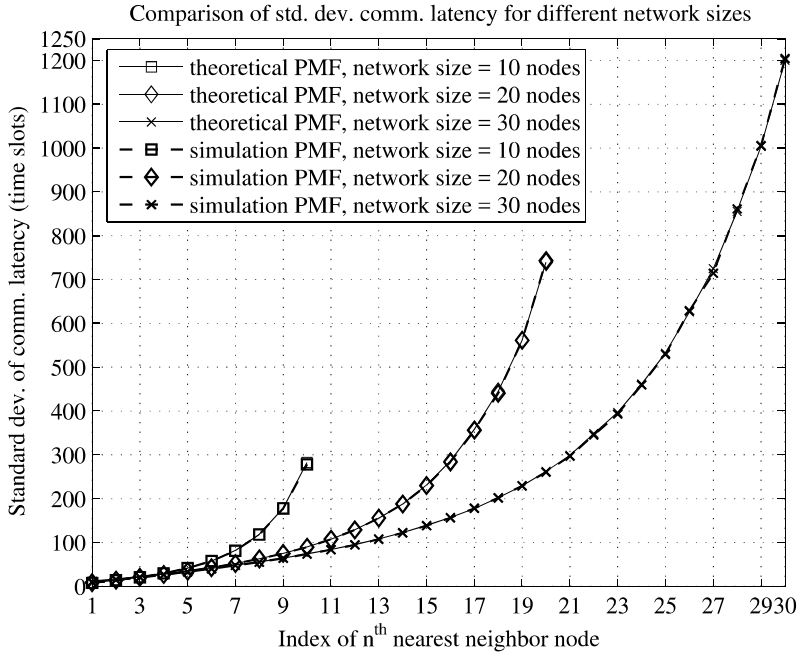


Figure 6.25: Comparison of the theoretical and simulation standard deviation of TDMA forward link communication latency for different network sizes.

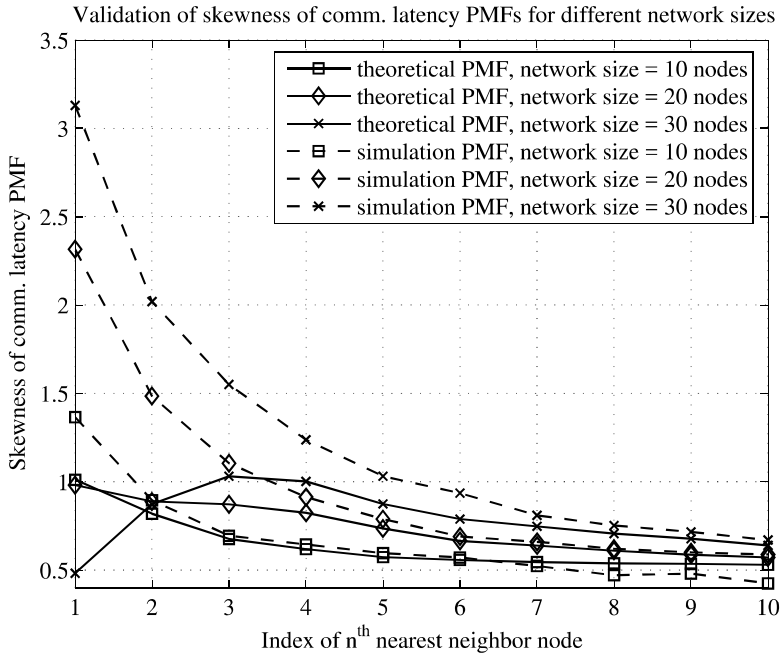


Figure 6.26: Comparison of the skewness of theoretical and simulation PMFs of TDMA forward link communication latency for different network sizes.

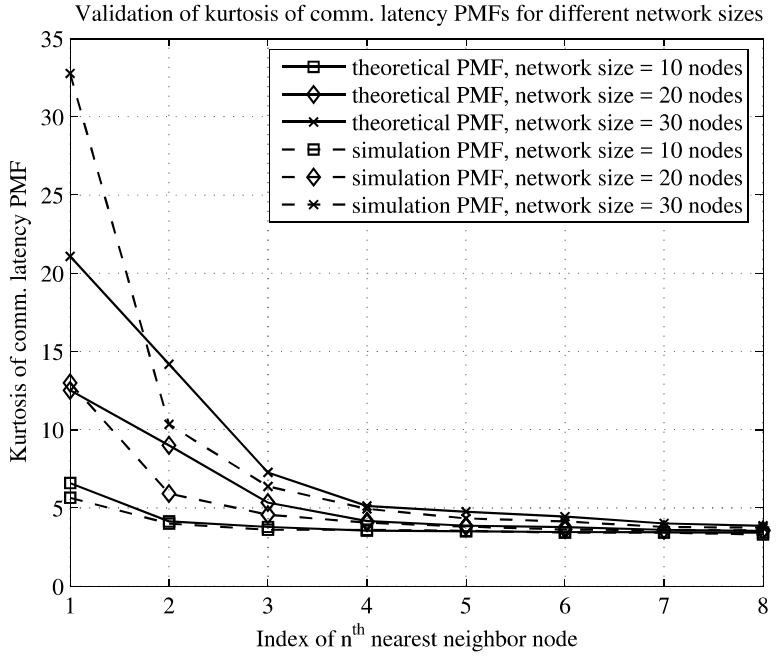


Figure 6.27: Comparison of the kurtosis of theoretical and simulation PMFs of TDMA forward link communication latency for different network sizes.

6.25 show that the theoretical PMF accurately predicts the mean and standard deviation of communication latency experienced by all nodes in a network. The figures also validate the theoretical model for different network sizes. Second, the figures show that the mean and standard deviation of communication latency increase non-linearly as one considers transmitter nodes which are located farther away from the receiver. Third, it is observed that the mean and standard deviation for a particular node varies with the network size. For instance, node N_{20} experiences a mean latency of 3000 time slots and 1250 time slots when the network size is 20 and 30 nodes, respectively. This is merely a consequence of the distance distribution model which is presented in Section 6.3.2. A similar trend is observed in Figure 6.16 when analyzing the impact of network size on a node's link connectivity. From Figures 6.26 and 6.27, it is observed that skewness and kurtosis are unusually high for nodes N_1 and N_2, and the theoretical and simulation values of these parameters do not match for these nodes. This is a direct consequence of a limitation of the queuing delay PMF model, as discussed in Section 6.2. Transmitter nodes that are located close to the receiver have a high probability of message transmission success, which in turn results in a very low mean service time close to 1 time slot. The queuing delay model, which serves as a foundation to the TDMA communication latency model, is shown to perform poorly under low mean service times. For all other nodes which are located farther away from the receiver than node N_2, the skewness and kurtosis of the theoretical PMF closely match with those of the simulation PMF.

Analysis of communication latency The impact of channel conditions on the mean and standard deviation of communication latency experienced by a node in the forward link of a TDMA network is analyzed next. The statistical parameters are computed from the PMF of communication latency which is given by Equation 6.48. Consider a TDMA network comprising 10 nodes located within a cluster radius of 200 meters. All nodes in the network employ a transmitter SNR of 130 dB (i.e., a transmitted signal power of 11 dBm). The size and arrival rate of the forward link communication buffer in each node is set to 14 and

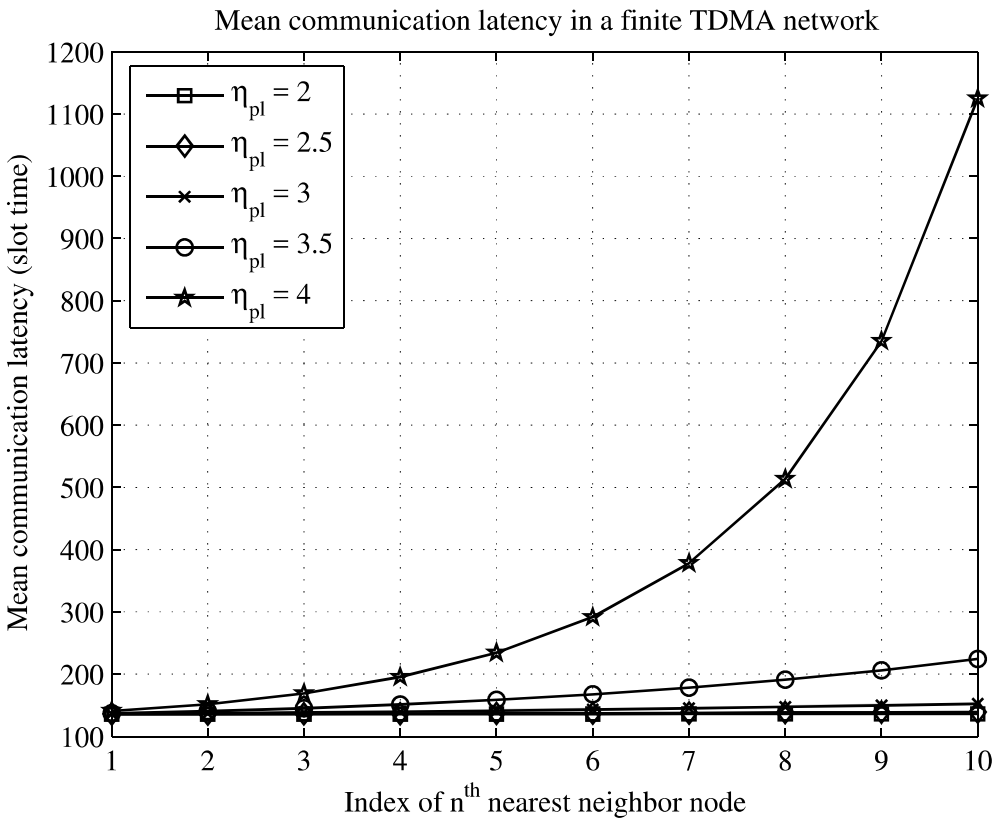


Figure 6.28: Mean communication latency experienced by nodes in a finite TDMA network in the presence of packet buffering, path loss, fading, and noise, for different path loss exponents.

0.6, respectively. The mean and standard deviation of communication latency per message experienced by the nodes is plotted, respectively, in Figures 6.28 and 6.29 for different values of the channel path loss exponent. As expected, the mean and standard deviation of latency increase with the path loss exponent for all nodes in the network, as well as increase as one moves farther away from the receiver. The difference in the mean and standard deviation of latency experienced by the nearest and farthest nodes is not significant for small path loss exponents of up to 2.5. For a path loss of 3.5 and above, however, the difference is significant. Figure 6.30 shows the throughput achieved by all nodes in the network for different path loss exponents, where the throughput of a node is computed using the definition of offered load given by Equation 6.49. The throughput does not vary much between the nearest

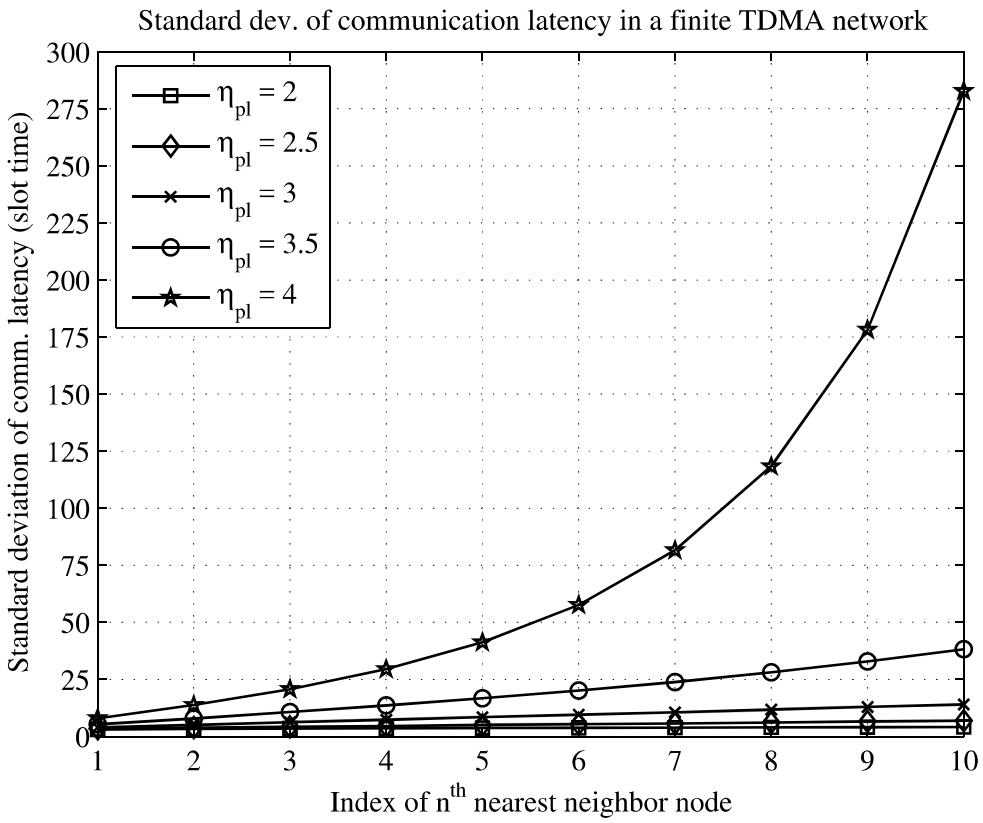


Figure 6.29: Standard deviation of communication latency experienced by nodes in finite TDMA networks in the presence of packet buffering, path loss, fading, and noise, for different path loss exponents.

and farthest nodes for small path loss exponents. It, however, decreases rapidly for the n^{th} nearest node as n increases for a path loss exponent of 3.5 and above. Next, the impact of channel path loss exponent and network size on network throughput is analyzed. The interest in network throughput arises due to the fact that the network inter-departure rate becomes the arrival rate to the service provider's request processing queue in the WCCN queuing network model, as discussed in Section 6.1.

The network throughput is computed using Equation 6.50. In this analysis, the network is located within a radius of 200 meters. All nodes in the network employ a transmitter SNR of 120 dB (approximately 1 dBm of transmitter signal power). The arrival rate and queue size of all transmitter queues is set to 0.1 and 14, respectively. The scalability of

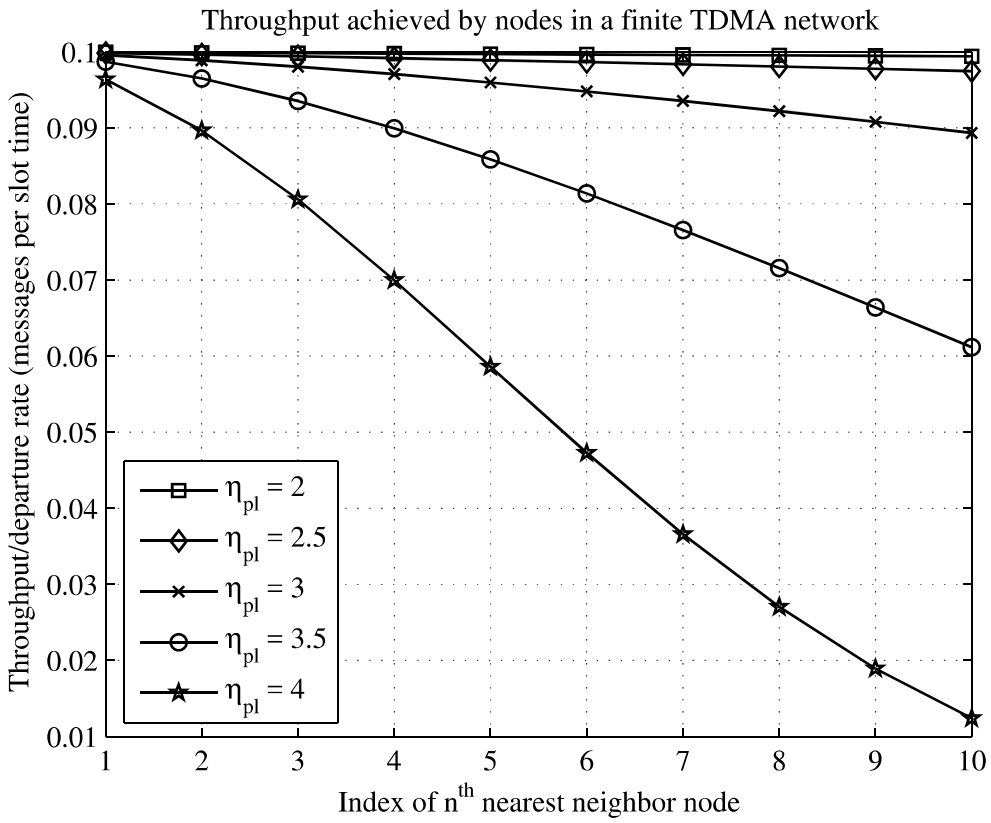


Figure 6.30: Throughput achieved by nodes in finite TDMA networks with packet buffering, fading, and noise, for different path loss exponents.

network throughput with network size (or network density) and channel path loss exponent is shown, respectively, in Figures 6.31 and 6.32. For small path loss exponents, an increase in the network size from 1 to 5 nodes increases the network throughput in a linear manner. For higher values of path loss exponent, say 3.5, this increase in network throughput is significantly smaller. Figure 6.32 shows that network throughput for a certain network size decreases non-linearly as the channel path loss exponent is increased. The network throughput curves for different network sizes converge at high path loss exponents.

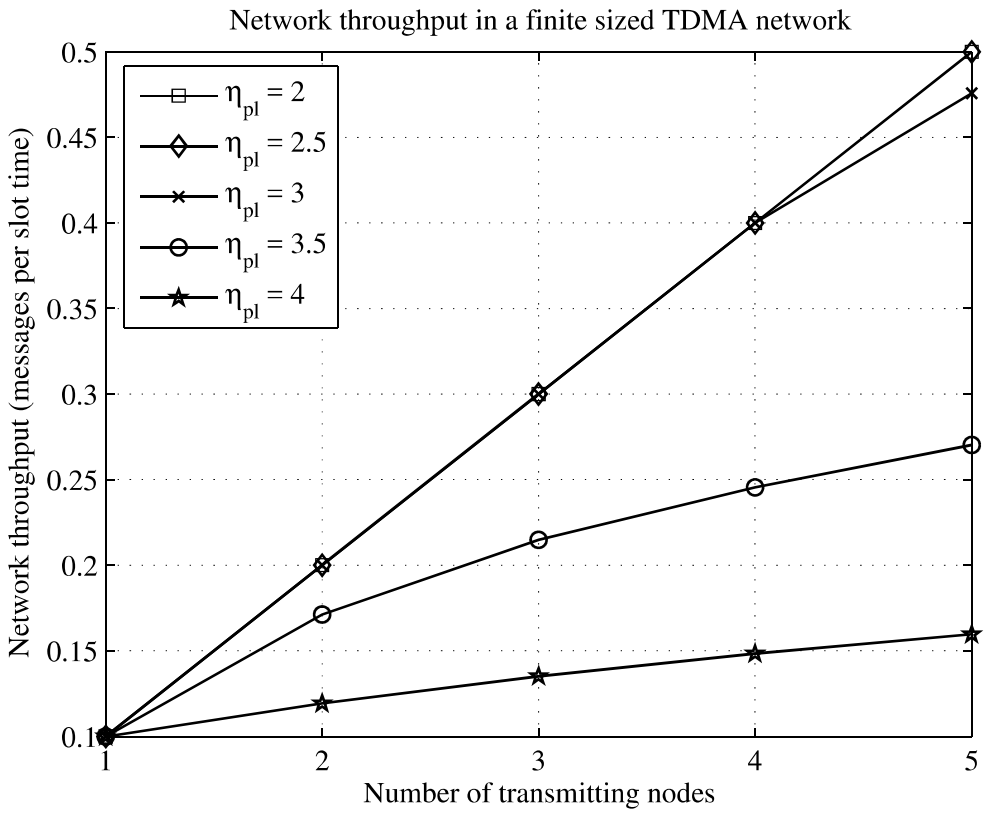


Figure 6.31: Network throughput achieved in finite TDMA networks with packet buffering, fading, and noise, for different network sizes.

6.4 WCCN Response Time Model

This section derives the proposed discrete-time WCCN response time model which computes the overall latency experienced by a user application when it is executed in a typical WCCN environment. The response time model is developed for a simple wireless cloud computing scenario as described by the WCCN queuing network model which is presented in Section 6.1. When a user application is executed in a WCCN environment, the various latencies experienced by the application include the forward link communication latency encountered while sending service requests to a service provider, processing time incurred by the service requests at the service provider, and the reverse link communication latency encountered when the service provider sends the service request's processing results back to

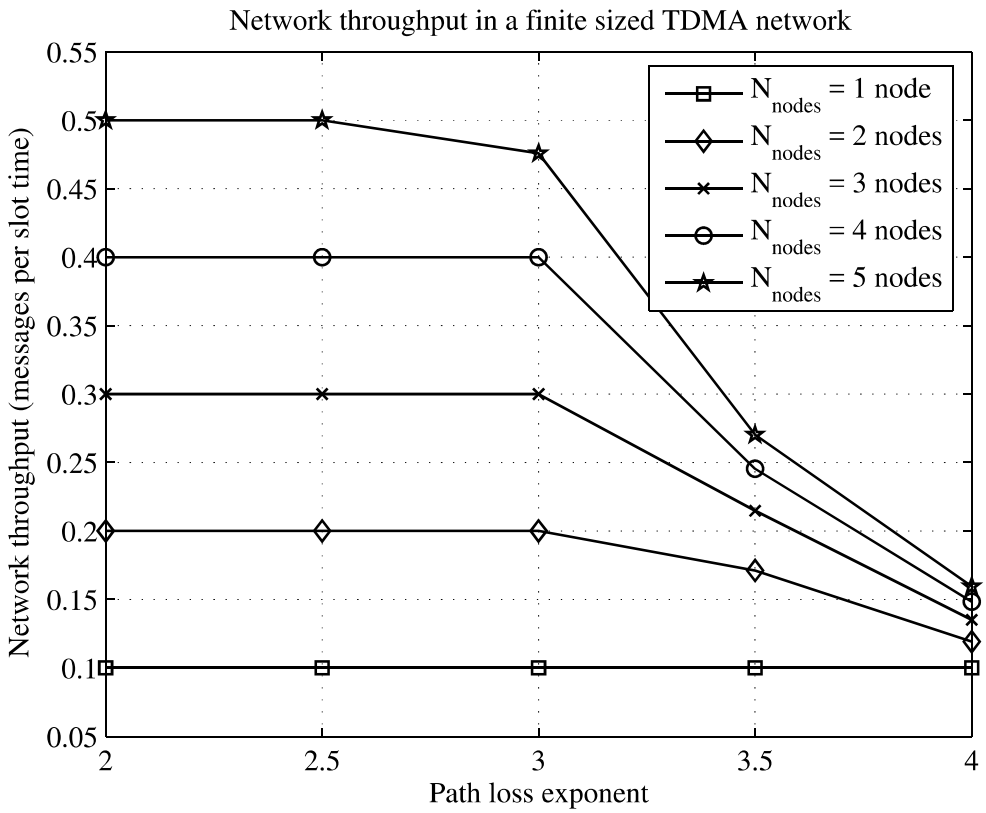


Figure 6.32: Network throughput achieved in finite TDMA networks with packet buffering, fading, and noise, for different path loss exponents.

the service requester. As discussed in Section 6.1, the response time of the application is modeled as the delay experienced in a tandem network comprising 3 queues, namely, the forward link communication queue, the service provider's request processing queue, and the service provider's reverse link communication queue. The forward link communication queue latency is modeled as discussed in Section 6.3.3. Let μ_{cp}, Q_{cp}, and p^j_{cpq} denote, respectively, the message service rate in the service provider's processing queue, the size of the queue, and the probability that there are j messages in the queue. Let μ_{cmrl}, Q_{cmrl}, and p^j_{cmrlq} denote, respectively, the transmission rate of a message in the reverse link queue, the size of the queue, and the probability that there are j messages in the queue. The queues at the service provider are modeled using the queuing latency model, which is presented in Section 6.2, as

given by

$$f_{T_{cpq}}(t) = \sum_{j=0}^{Q_{cp}-1} p_{cpq}^{j} \, f_{T_{sum}^{j+1}}(t;\ j+1;\ \mu_{cp}), \tag{6.51}$$

$$f_{T_{cmrl}}(t) = \sum_{j=0}^{Q_{cmrl}-1} p_{cmrlq}^{j} \, f_{T_{sum}^{j+1}}(t;\ j+1;\ \mu_{cmrl}), \tag{6.52}$$

$$\text{where} \sum_{j=0}^{Q_{cp}-1} p_{cpq}^{j} = 1, \tag{6.53}$$

$$\text{and} \sum_{j=0}^{Q_{cmrl}-1} p_{cmrlq}^{j} = 1. \tag{6.54}$$

In the tandem queue network, the departure rate of the TDMA network, which is computed using Equation 6.50, becomes the arrival rate of messages at the service provider's request processing queue. Similarly, the departure rate of the request processing queue becomes the arrival rate of the reverse link communication queue. The following assumptions are made while deriving the response time model: (a) the three latency components are independent of each other; (b) there can be a loss of messages at all queues, i.e., a message can be lost when it is blocked by a queue as it tries to enter it; (c) there is no channel contention in the reverse link, i.e., the service provider has dedicated links to communicate with service requesters without having to share the channel with any other node; (d) the forward link and reverse link are symmetrical i.e., the channel conditions between two nodes are the same in the forward and reverse links; and (e) the slot times employed by all the queues are equal and synchronized. An additional assumption is that the same throughput is maintained for all nodes in the reverse link, i.e., service time characteristics are identical for messages destined to all service requester nodes. Accordingly, the service rate of the reverse link communication queue is set to a fixed value regardless of which service requester's results message is being transmitted. In this chapter, this value is set to the mean link connectivity in the network.

Now, the response time experienced by an application that is hosted by service requester

node N_i is computed as given by

$$T_{rt}^{i} = T_{cmfl}^{i} + T_{cpq} + T_{cmrl}. \tag{6.55}$$

Thus, the PMF of response time experienced by service requests originating from node N_i is given by

$$f_{T_{rt}^{i}}(t) = f_{T_{cmfl}^{i}} \otimes f_{T_{cpq}} \otimes f_{T_{cmrl}}, \tag{6.56}$$

$$f_{T_{rt}^{i}}(t) = f_{T_{cmfl}^{i}} \otimes \sum_{j=0}^{Q_{cp}-1} \sum_{k=0}^{Q_{cmrl}-1} p_{occup}^{j,k} \times \left(f_{T_{sum}^{j+1}}(t;\ j+1;\ \mu_{cp}) \otimes f_{T_{sum}^{k+1}}(t;\ k+1;\ \mu_{cmrl}) \right), \tag{6.57}$$

$$\text{where } p_{occup}^{j,k} = p_{cpq}^{j} \times p_{cmrlq}^{k}$$

The assumption that the slot times in all the three queues are synchronized is important when computing the convolution in Equation 6.56.

In the scenario where a service requester has a dedicated link for forward link communication with the service provider, the TDMA schedule period can be set to $T_s = 1$. In this case, the response time distribution can be further evaluated as follows. With regards to the response time experienced by node N_x, the sum of independent and heterogeneous pascal random variables can be expressed as a generalized mixture of a finite number of Pascal

distributions [71], as given by

$$
\begin{aligned}
f_{T^x_{rt}}(t) &= \sum_{i=0}^{Q^x_{cmfl-1}} \sum_{j=0}^{Q_{cp-1}} \sum_{k=0}^{Q_{cmrl-1}} p^{i,j,k}_{occup} \times \left(\sum_{l=1}^{N_{conv}} \sum_{m=1}^{r_l} d_{l,m}\, NB(m, p_l) \right), \quad (6.58)\\
\text{where } p^{i,j,k}_{occup} &= p^{xi}_{cmflq} \times p^{j}_{cpq} \times p^{k}_{cmrlq},\\
r_1 &= i+1,\ r_2 = j+1,\ r_3 = k+1,\\
p_1 &= P^x_{success},\ p_2 = \mu_{cp},\ p_3 = \mu_{cmrl},\\
d_{l,m} &= \frac{(-1)^{r_l - m}}{(r_l - m)!\ {q_l}^{r_l - m}\ {p_l}^{m}}\ \Pi_l^{(r_l - m)}\left({q_l}^{-1}\right),\\
\text{and } \Pi_l(t) &= \left(\prod_{s=1}^{N_{conv}} {p_s}^{r_s} \right) \times \prod_{\substack{c=1,\\ c \neq l}}^{N_{conv}} \frac{1}{(1 - q_c t)^{r_c}},
\end{aligned}
$$

where N_{conv} is the number of convolutions involved, which is 3 in this case and $\Pi^{(n)}$ is the n^{th} derivative of $\Pi(t)$. The distributions in the mixture are all pascal distributions. This formulation is preferred over the one provided in Furman [72], where the sum is expressed as an infinite mixture of negative binomial distributions. When applied to the current problem involving 3 convolutions,

$$
\Pi_l(t) = \left(\prod_{s=1}^{3} {p_s}^{r_s} \right) \times \frac{1}{(1 - q_e t)^{r_e}}\ \frac{1}{(1 - q_f t)^{r_f}},
$$
$$
\text{where } 1 \leq e, f \leq 3, \text{ and } e \neq f \neq l,
$$

whose derivative can be evaluated as follows. As per Leibniz rule,

$$
\frac{d^n}{dt^n}(f \cdot g) = \sum_{v=0}^{n} {}^nC_v\ f^{(v)}\ g^{(n-v)},
$$

where f and g are functions of t. Now,

$$
\frac{d^n}{dt^n} \frac{1}{(1 - q_e t)^{r_e}} = {q_e}^n\ \langle r_e \rangle^n\ \frac{1}{(1 - q_e t)^{r_e + n}},
$$

where $\langle r_e \rangle_n$ is a ascending factorial defined as $\langle r_e \rangle^n = r_e \ (r_e + 1) \ldots (r_e + n - 1)$. Thus,

$$\Pi_l^{(r_l - m)} \left({q_l}^{-1} \right) = \left(\prod_{s=1}^{3} {p_s}^{r_s} \right) \times \sum_{v=0}^{r_l - m} \binom{r_l - m}{v} \langle r_e \rangle^v \ \langle r_f \rangle^{r_l - m - v} \ {q_e}^v \ {q_f}^{r_l - m - v} \times \frac{1}{(1 - q_e/q_l)^{r_e + v}} \ \frac{1}{(1 - q_f/q_l)^{r_f + r_l - m - v}}, \quad (6.59)$$

where $r_l - m > 0$. When $r_l - m = 0$, $\Pi_l^{(0)} \left({q_l}^{-1} \right) = \Pi_l \left({q_l}^{-1} \right)$. The CDF, mean and variance of response time are computed as given by

$$F_{T_{rt}^x}(t) = \sum_{i=0}^{Q_{cmfl}^x - 1} \sum_{j=0}^{Q_{cp}-1} \sum_{k=0}^{Q_{cmr}-1} \left(\sum_{l=1}^{N_{conv}} \sum_{m=1}^{r_l} p_{occup}^{i,j,k} \ d_{l,m} \ F(m, p_l) \right) \quad (6.60)$$

$$\overline{T}_{rt}^x = \sum_{i=0}^{Q_{cmfl}^x - 1} \sum_{j=0}^{Q_{cp}-1} \sum_{k=0}^{Q_{cmrl}-1} \left(\sum_{l=1}^{N_{conv}} \sum_{m=1}^{r_l} p_{occup}^{i,j,k} \ d_{l,m} \ \overline{T}(m, p_l) \right), \quad (6.61)$$

$$\text{where} \quad \sum_{l=1}^{N_{conv}} \sum_{m=1}^{r_l} d_{l,m} = 1$$

The response time PMF model presented by Equation 6.56 is validated against simulation results and analyzed next.

Validation and Analysis of WCCN response time The TDMA network simulation model, which is introduced in Section 6.3.3, is extended to include the service provider's request processing and reverse link communication queues. The simulation model is run for a TDMA network comprising 10 transmitting nodes, where the node locations are modeled using a binomial point process that spans a circular network cluster area within a radius of 200 meters with the receiver located at the origin of the circle. Transmitter SNR at the individual nodes is set at 130 dB (i.e., a transmitted signal power of 11 dBm) and the channel path loss exponent is varied. The size and arrival rate of the forward link communication queue in each node is set to 14 and 0.6, respectively. In addition, the size of the service provider's request processing queue and the reverse link communication queue are each set

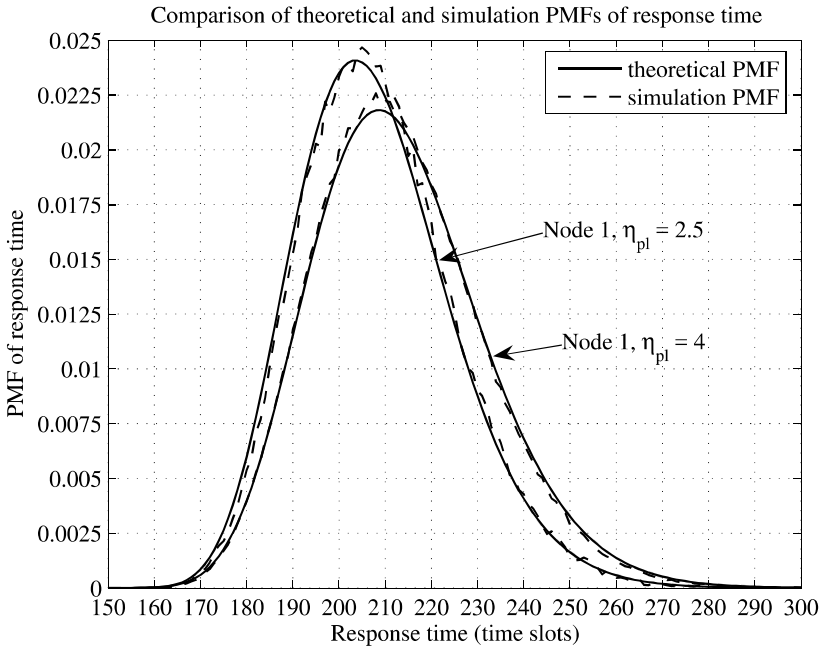


Figure 6.33: Comparison of simulation and theoretical PMFs of node N_1's response time for different channel conditions.

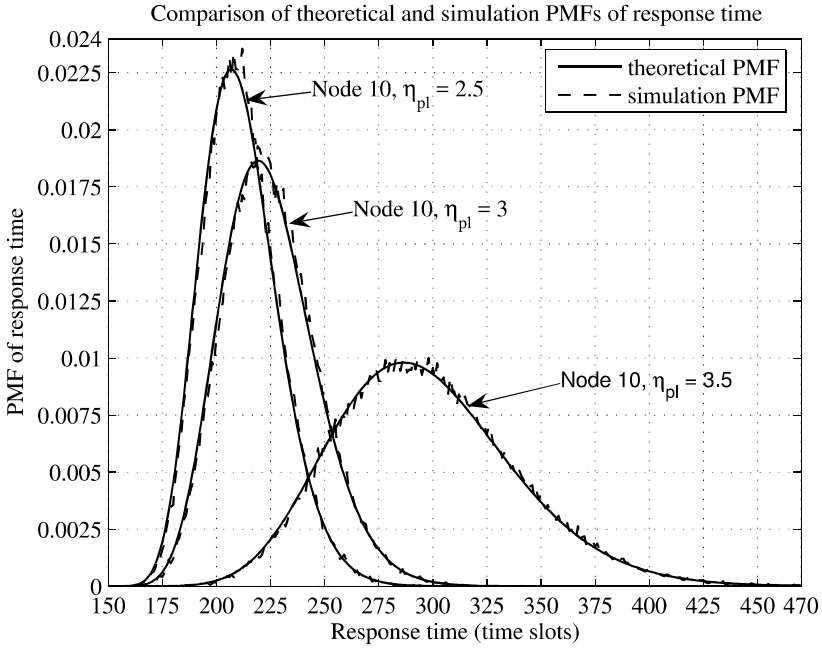


Figure 6.34: Comparison of simulation and theoretical PMFs of node N_{10}'s response time for different channel conditions.

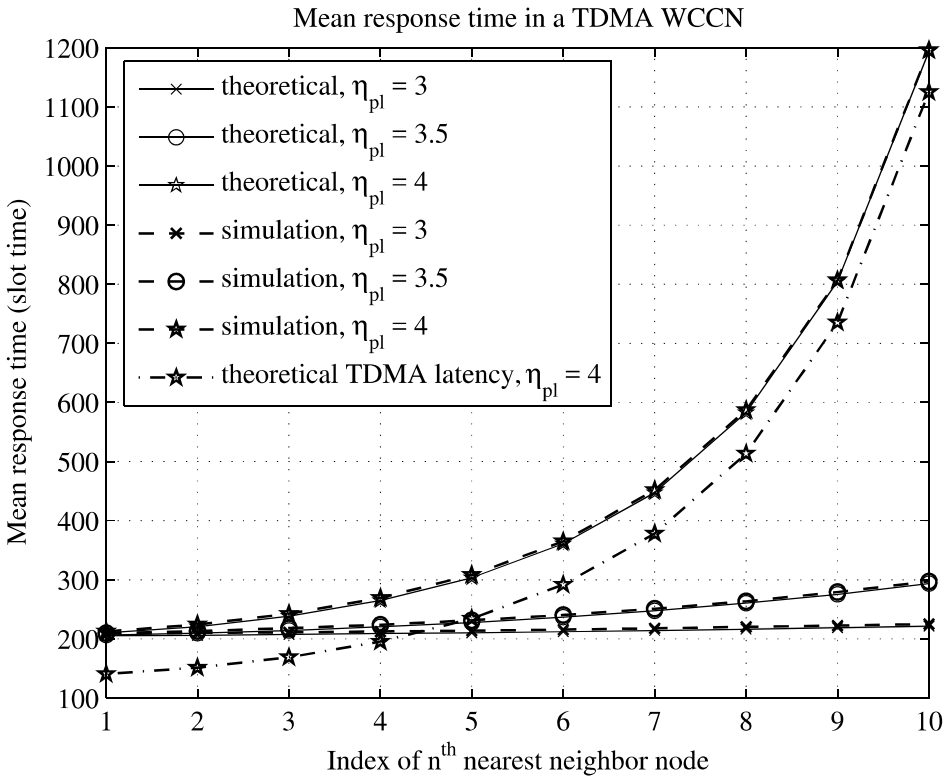


Figure 6.35: Mean response time experienced by a service request in a finite TDMA-based WCCN for different channel conditions.

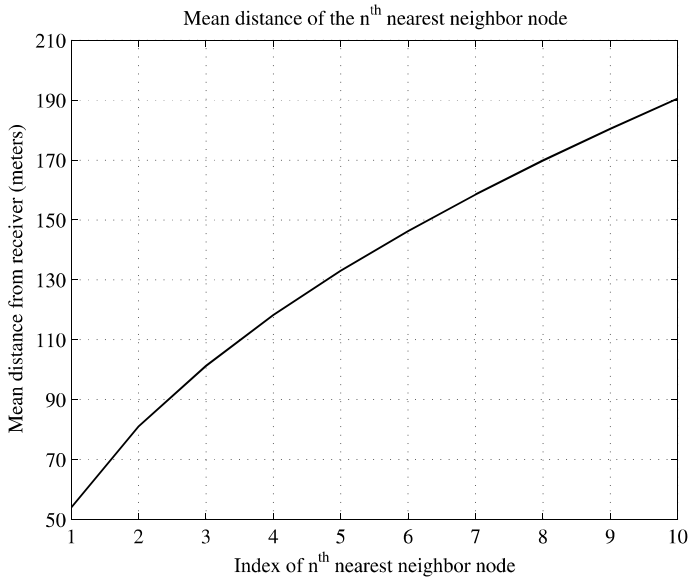


Figure 6.36: Mean distance of n^{th} nearest neighbor in a network that spans a radius of 200 meters.

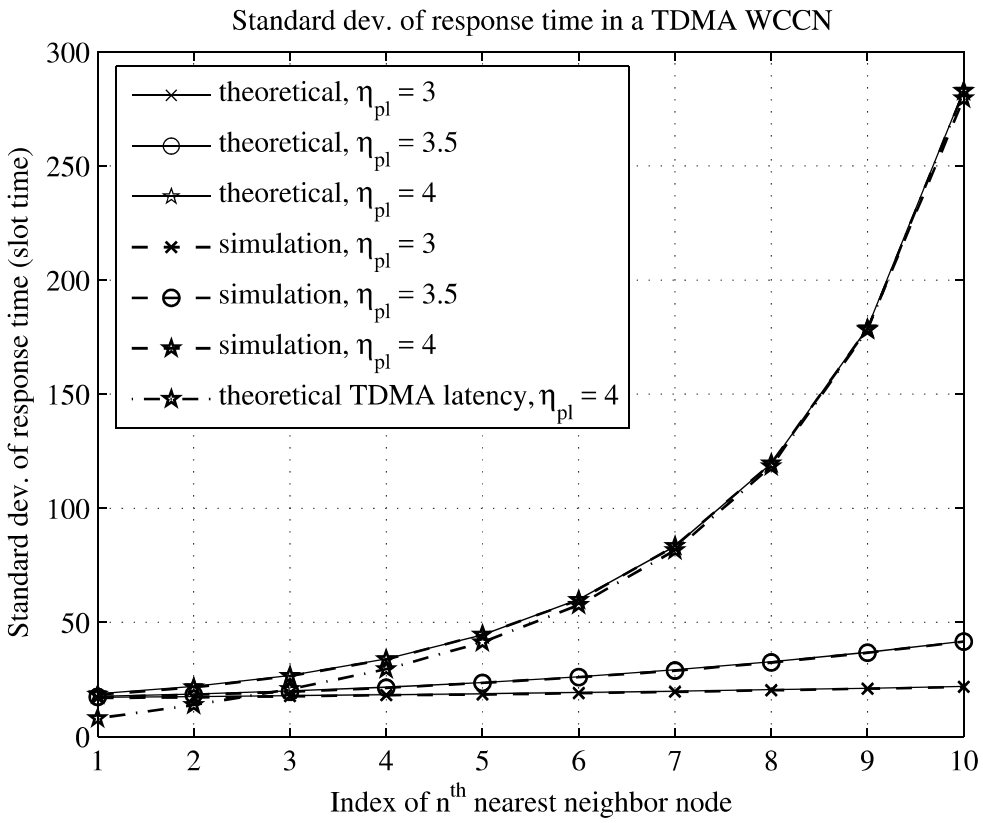


Figure 6.37: Standard deviation of response time experienced by a service request in a finite TDMA-based WCCN for different channel conditions.

to 14. The mean service time to process a service request is set to 5 slot times. Equation 6.56 is used to compute the theoretical PMF of response time, which in turn is used to compute the statistics of response time, namely, mean, standard deviation, skewness, and kurtosis. Figures 6.33 and 6.34 compare the theoretical PMFs with simulation PMFs for extreme scenarios involving the nearest and farthest nodes in the network, as well as low and high path loss exponents. The plots show that the theoretical PMF closely matches with the simulation PMF in all these scenarios. In addition, it is observed from the PMF plots that the PMF shifts to the right either as one moves away from the receiver or as the channel path loss exponent is increased. An increase in path loss exponent also causes the PMF curves to spread out.

Figures 6.35 and 6.37 plot, respectively, the mean and standard deviation of response time

experienced by a service request which originates from the n^{th} nearest neighbor node in a TDMA network for different channel path loss exponents. In addition, the plots show, respectively, the mean and standard deviation of forward link communication latency for comparison purposes. For reference purposes, Figure 6.36 shows the mean distance of all transmitting nodes in the network, which is computed using Equation 6.22. First, the plots show that the theoretical and simulation values of mean and standard deviation match closely for all nodes in the network and for different channel path loss exponents. The simulation and experimental skewness are found to lie in the range of 0.4 - 0.5 for all the scenarios tested. The kurtosis lies in the range of 3.25 - 3.65 both for theoretical and simulation PMFs. Second, it is observed that the trends in the response time curves strongly resemble those observed for forward link communication latency (Figure 6.28). This implies that the characteristics of the underlying communication network strongly dominate the response time characteristics of a user application in a WCCN. A related observation is that the standard deviation of response time does not differ much from the standard deviation of forward link communication latency. This further implies that the uncertainty in response time is mainly contributed by the forward link communications latency component of the response time model. Third, the mean response time experienced by a service request is slightly higher than the mean forward link latency experienced by the service request. This additional delay is due to the delay incurred at the service provider's queues.

6.5 Chapter Summary

This chapter derives a discrete-time stochastic model of the response time experienced by a service request in a fundamental WCCN service request - provisioning scenario. The building blocks of this model are a general queuing delay distribution model, a link connectivity model, and a buffered TDMA communication latency model. In this chapter, these building blocks are derived, verified against their respective simulations, and analyzed for performance trends. The queuing delay distribution model is found to function accurately except at low

mean service times which are less than 1.4 time slots. In all other scenarios, the delay statistics computed using the model closely match with those computed from simulation. The TDMA latency model is impacted by this limitation, particularly when considering nodes that are located close to the intended receiver such that the mean service time (or mean message transmission time) of the forward link communication queue is as low as 1.05 time slots. In addition, a comparison of the PMF computed using the TDMA latency model with that computed using simulation shows that the simulation PMF matches the theoretical PMF only at low resolutions of the x-axis, i.e., when smaller number of histogram bins are employed for the simulation PMF. In spite of the limitation, the statistics of latency, namely CDF, mean, standard deviation, skewness, and kurtosis, are consistent between the theoretical and simulation models.

These limitations do not impact the theoretical response time model. Results produced using the theoretical response time model closely match with those generated using the simulation model for all nodes in the network as well as for different channel conditions. Having validated the response time model, it is used to analyze the impact of wireless channel conditions on response time characteristics. The analysis shows that the response time characteristics are strongly influenced by the characteristics of the underlying wireless network environment. For instance, the mean response time scales non-linearly as one traverses from a node closest to the service requester to a node farthest away from it, particularly under a high path loss exponent of 4.

Chapter 7

Conclusions

This chapter highlights the contributions and conclusions of this dissertation. The dissertation proposes an architectural framework for WCCNs and presents the different phases of its development. The framework enables all nodes in a WCCN to collaborate in order to service user applications and deliver optimal application performance to the user, where the performance metrics include efficiency in energy consumption and execution time incurred by user applications.

In the architectural design phase, the dissertation presents the proposed component-based architectural framework. The framework has provisions for various functionalities which are critical for WCCN operation, such as user application management, services management, WDC resource allocation, cloud environment monitoring, and communication between distributed tasks of a user application. In addition, the architectural design incorporates the notion of multi-layer decision making and control, which enables every node in a WCCN to adapt to the current operating wireless cloud environment and deliver optimal performance. A proof-of-concept prototype of the architecture is demonstrated based on an example use case of the architecture. The prototype serves as a real-world validation of the WDC aspects of the architecture, as well as demonstrates the benefits of workload distribution in WCCNs. Experiments using the prototype have shown that WDC is feasible and benefical

in executing complex user applications in a WCCN environment. Experimental results show that WDC workload distribution improves the makespan of a complex user application by nearly 80% when compared to the traditional approach of local processing. In addition, the results show that an increase in the network size improves makespan. However, there is no significant improvement in makespan when the network size is increased beyond a certain value. These performance trends match with those exhibited by simulation results.

In the mathematical modeling phase, the dissertation presents models of power consumption cost, energy consumption cost, and latency incurred by computational and communication activities associated with the distributed execution of a user application in a WCCN. The models are primarily used in this dissertation in the performance analysis of resource allocation algorithms. The dissertation also derives a discrete-time model of the response time experienced by service requests in a TDMA based WCCN environment and analyzes response time under different operating conditions. Statistical properties of response time, namely the discrete cumulative density function, mean, standard deviation, skewness, and kurtosis, are derived using the proposed model and are found to closely match their respective simulation values. Results generated from the model show that the characteristics of the underlying wireless environment significantly influence the response time characteristics of service requests, particularly under harsh channel conditions. This infers that communication system design plays an important role in influencing the performance of user applications when executed in WCCNs.

In the design and performance analysis phase, the dissertation analyzes the limitations and scalability in the performance benefits achieved by WDC workload distribution in a WCCN, where workload allocation and scheduling algorithms are used to make decisions on workload distribution. Preliminary analysis is performed for simple homogeneous environments using simple uniform workload allocation to analyze the break point when WDC workload distribution is power efficient and energy efficient compared to on-board local processing. Simulation results show that power consumption gains can be achieved by exploiting the non-linearity in processor power consumption characteristics. In addition, results show that

workload distribution is energy efficient when the communication overhead does not dominate over the computational energy consumption. The dissertation also analyzes the benefits of workload distribution when applied to an example tactical UAV scenario, where WDC empowers a network of UAV nodes to process complex data all by itself, without having to depend on remote super nodes for processing the data. The UAV network achieves significant savings in energy consumption and improvement in network lifetime when the network pre-processes data before transmitting it to a remote node over a long range backhaul. The savings are significant in the case of large backhaul ranges, however the performance benefits do not scale linearly with the backhaul range.

The main focus in the design and performance analysis phase is on the design of the proposed workload allocation algorithm and the proposed task allocation-scheduling algorithm, and the evaluation of their performance in heterogeneous WCCN environments. The performance evaluation results analyze the impact of system design parameters on workload allocation performance metrics, namely, master node's energy savings, application's makespan, and percentage improvement in makespan. The system design parameters employed in this analysis are network range, network size, task execution time range, application's workload, transmitted signal power, and energy supply availability. Simulation results show that performance gains can be achieved by exploiting concurrency in user applications' software structure and diversity inherent in a WCCN environment. Computational diversity, diversity in the availability of power-energy resources, and diversity in link quality and inter-node distances are some of the diversity factors that can exist in a WCCN environment. High values of workload size, task execution time range, and network size can improve computational diversity in a WCCN environment. An increase in network size improves diversity and increases the number of nodes located in close proximity to the master node. It also increases the amount of computational resources that can be utilized to exploit concurrency in an application's software structure. There is, however, significant benefit in large network sizes only when there is significantly high computational diversity in both the application and the environment. The master node's energy savings scales non-linearly with the appli-

cation's computational workload, computational power consumption per task, network size, power supply availability, and communication range. An increase in network size, task execution range, or application workload results in a non-linear improvement in the application's makespan until the "law of diminishing returns" is encountered.

This is clearly a work in progress. The dissertation lays the foundation for future developments in the WCCN architectural framework. One of the future plans is to extend the prototype to involve multiple master nodes which contend for the computational resources hosted by slave nodes. In addition, future prototypes are expected to implement the Service Management Layer of the architecture. There is also a challenge in extending the WCCN prototype presented in this dissertation to embedded platforms, where the challenges can be different from those faced with laptop PCs. Preliminary testing of the prototype's image processing application on a E100 Universal Software Radio Peripheral device ($http://www.ni.com/usrp/$), which is an embedded device, has shown that the time taken to process a single image is several times longer than the time taken on a laptop PC. This indicates that there is a greater need for collaborative computing when using resource constraint embedded platforms. It also remains to be seen what new "killer" applications can be developed in the future that can leverage the benefits of WCCNs.

Bibliography

[1] Wikipedia, "Mobile phone - wikipedia, the free encyclopedia," Accessed March 2010.

[2] P. Mell and T. Grance, "The NIST definition of cloud computing," Tech. Rep. Special Publication 800-145, National Institute of Standards and Technology, Gaithersburg, MD, USA, September 2011.

[3] N. A. Lynch, *Distributed Algorithms*. The Morgan Kaufmann Series in Data Management Systems, first ed., March 1996.

[4] M. K. Goff, *Network Distributed Computing: Fitscapes and Fallacies*. Prentice Hall PTR, May 2004.

[5] W. Jia and W. Zhou, *Distributed Network Systems: From Concepts to Implementations*. Springer, first ed., Dec. 2004.

[6] J. Sang, C.-H. Lin, S. Varadaraja, and M.-C. Wang, "Distributed network computing over wireless links," in *International Conference on Parallel and Distributed Systems (ICPADS'97)*, (Seoul, South Korea), pp. 252–259, December 1997.

[7] Y. Xu and H. Qi, "Distributed computing paradigms for collaborative signal and information processing in sensor networks," *Journal of Parallel and Distributed Computing*, vol. 64, pp. 945–959, August 2004.

[8] S. Abdelhak, R. S. Chaudhuri, C. S. Gurram, S. Ghosh, and M. Bayoumi, "Energy-aware distributed QR decomposition on wireless sensor nodes," *The Computer Journal*, 2010.

[9] Y. Yu and V. K. Prasanna, "Energy-balanced task allocation for collaborative processing in networked embedded systems," in *Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool support for embedded systems (LCTES)*, vol. 38, (San Diego, CA, USA), July 2003.

[10] Y. Yu and V. Prasanna, "Power-aware resource allocation for independent tasks in heterogeneous real-time systems," in *Proceedings of Ninth International Conference on Parallel and Distributed Systems (CAPDS)*, pp. 341 – 348, 2002.

[11] Y. Yu and V. K. Prasanna, "Energy-balanced task allocation for collaborative processing in wireless sensor networks," *Journal Mobile Networks and Applications*, vol. 10, February 2005.

[12] A. B. Olsen, F. H. P. Fitzek, and P. Koch, "Evaluation of cooperative task computing for energy aware wireless networks," in *Proceedings of International Workshop on Wireless Ad-Hoc Networking (IWWAN)*, 2005.

[13] T. Xie and X. Qin, "An energy-delay tunable task allocation strategy for collaborative applications in networked embedded systems," *IEEE Transactions on Computers*, vol. 57, pp. 329–343, March 2008.

[14] N. Santoro, *Design and Analysis of Distributed Algorithms*. Wiley-Interscience, October 2006.

[15] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations," *Journal International Journal of High Performance Computing Applications*, vol. 15, pp. 200 – 222, August 2001.

[16] I. Foster, "What is the grid? A three point checklist." Argonne National Laboratory & University of Chicago: www.mcs.anl.gov/ itf/Articles/WhatIsTheGrid.pdf, July 2002.

[17] A. Agarwal, D. O. Norman, and A. Gupta, "Wireless grids: approaches, architectures and technical challenges," *MIT Sloan Working Paper No. 4459-04; Eller College Working Paper No. 1016-05*, January 2004.

[18] L. W. McKnight, J. Howison, and S. Bradner, "Guest editors' introduction: wireless grids-distributed resource sharing by mobile, nomadic, and fixed devices," *IEEE Internet Computing*, vol. 8, pp. 24 – 31, July-Aug 2004.

[19] M. Conti and M. Kumar, "Opportunities in opportunistic computing," *Computer*, vol. 43, pp. 42 – 50, January 2010.

[20] A. Schill, B. Bellmann, W. Bohmak, and S. Kummel, "System support for mobile distributed applications," in *SDNE '95 Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environments*, (Washington, DC, USA), p. 124, IEEE Computer Society, 1995.

[21] D. Datla, X. Chen, T. Tsou, S. Raghunandan, S. M. Hasan, J. H. Reed, B. Fette, C. B. Dietrich, J.-H. Kim, and T. Bose, "Wireless distributed computing: a survey of research challenges," *IEEE Communications Magazine*, January 2012.

[22] E. Kushilevitz and N. Nisan, *Communication Complexity*. Cambridge University Press, December 1996.

[23] E. Kushilevitz and Y. Mansour, "Computation in noisy radio networks," in *Proc. 9th Annual ACM-SIAM Symp. Discrete Algorithms*, (Philadelphia, PA, USA), p. 23643, 1998.

[24] D. Datla, X. Chen, T. R. Newman, J. H. Reed, and T. Bose, "Power efficiency in wireless network distributed computing," in *IEEE Vehicular Technology Conference*, (Anchorage, AK, USA), September 2009.

[25] D. Datla, T. Tsou, T. R. Newman, J. H. Reed, and T. Bose, "Waveform level computational energy management in software defined radios," in *SDR'09 Technical Conference and Product Exposition*, (Washington, DC, USA), December 2009.

[26] M. Uetz, *Algorithms for Deterministic and Stochastic Scheduling*. PhD thesis, Maastricht University, Berlin, Germany, November 2001.

[27] J. Weinman, "Network implications of cloud computing," in *Technical Symposium at ITU Telecom World (ITU WT)*, October 2011.

[28] G. Tel, *Topics in distributed algorithms*. Cambridge International Series On Parallel Computation; Vol. 1, 1991.

[29] D. P. Vidyarthi, B. K. Sarker, A. K. Tripathi, and L. T. Yang, *Scheduling in Distributed Computing Systems: Analysis, Design and Models*. Springer, 2008.

[30] D. Datla, S. M. Hasan, J. H. Reed, and T. Bose, "Fundamental issues of wireless distributed computing in SDR networks," in *SDR'10 Technical Conference and Product Exposition*, (Washington, DC, USA), November 2010.

[31] X. Chen, T. R. Newman, D. Datla, T. Bose, and J. H. Reed, "The impact of channel variations on wireless distributed computing networks," in *IEEE Globecom*, (Honolulu, HI, USA), Dec. 2009.

[32] X. Chen, S. M. Hasan, T. Bose, and J. H. Reed, "Cross-layer resource allocation for wireless distributed computing networks," in *IEEE Radio and Wireless Symposium*, (New Orleans, LA, USA), Jan. 2010.

[33] T. Tsou and J. H. Reed, "Software architecture for cooperative applications," in *SDR'09 Technical Conference and Product Exposition*, (Washington, DC, USA), December 2009.

[34] A. S. Tanenbaum and M. V. Steen, *Distributed Systems: Principles and Paradigms*. Prentice Hall, second ed., October 2006.

[35] A. J. Goldsmith and S.-G. Chua, "Variable-rate variable-power MQAM for fading channels," *IEEE Transactions on Communications*, vol. 45, pp. 1218–1230, October 1997.

[36] B. Sklar, *Digital Communications: Fundamentals and Applications*. Pearson Education, Inc., second ed., 2002.

[37] R. Min and A. Chandrakasan, "A framework for energy-scalable communication in high-density wireless networks," in *Proc. International Symposium on Low Power Electronics and Design ISLPED '02*, pp. 36–41, 2002.

[38] A. Sinha and A. P. Chandrakasan, "Energy aware software," in *Thirteenth International Conference on VLSI Design*, (Calcutta, India), pp. 50–55, 2000.

[39] A. Sinha and A. P. Chandrakasan, "Jouletrack-a web based tool for software energy profiling," in *Proceedings of Design Automation Conference*, pp. 220– 225, 2001.

[40] S. R. Cui, *Cross-Layer Optimization in Energy Constrained Networks*. PhD thesis, Stanford University, June 2005.

[41] S. Cui, A. J. Goldsmith, and A. Bahai, "Energy-efficiency of MIMO and cooperative MIMO techniques in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 1089–1098, AUGUST 2004.

[42] I. 802.11-12/0935ro, "6-10 GHz UWB link budget and discussion," tech. rep., IEEE, July 2012.

[43] A.Godsmith, *Wireless Communications*. Cambridge University Press, 2005.

[44] J. Gaeddert, "Liquid radio." ganymede.ece.vt.edu.

[45] "Open source computer vision." opencv.org.

[46] C. Pullela, L. Xu, D. Chakraborty, and A. Joshi, "A component based architecture for mobile information access," in *Proceedings 2000 International Workshops on Parallel Processing*, (Toronto, Ont. , Canada), pp. 65 – 72, August 2000.

[47] I. Shpantzer, L. Schoenfeld, M. Grindahl, and V. Kelman, "Keyware: an open wireless distributed computing environment," in *Proc. SPIE*, vol. 2601, (Philadelphia, PA, USA), October 1995.

[48] A. Pavan, R. Jha, L. Graba, S. Cooper, I. Cardei, M. Cardei, V. Gopal, S. Parthasarathy, and S. Bedros, "Real-time adaptive resource management," *IEEE Computer Magazine*, vol. 34, pp. 99 – 101, July 2001.

[49] P. Zhang and Z. Yan, "A QoS-aware system for mobile cloud computing," in *IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, (Beijing, China), pp. 518 – 522, September 2011.

[50] N. Fernando, S. W. Loke, and W. Rahayu, "Dynamic mobile cloud computing: ad hoc and opportunistic job sharing," in *IEEE International Conference on Utility and Cloud Computing (UCC)*, (Victoria, New South Wales, Australia), pp. 281 – 286, December 2011.

[51] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, (San Francisco, CA, USA), June 2010.

[52] W. Alsalih, S. Akl, and H. Hassancin, "Energy-aware task scheduling: towards enabling mobile computing over MANETs," in *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium*, April 2005.

[53] C. Jou, "The decision model of task allocation for constrained stochastic distributed systems," *Computers & Industrial Engineering*, vol. 58, pp. 344–351, April 2010.

[54] E. Y. S. L. Perng-Yi Richard MA and M. Tsuchiya, "A task allocation model for distributed computing systems," *IEEE Transactions on Computers*, vol. C-31, pp. 41–47, Jan 1982.

[55] T. Hu, "Parallel sequencing and assembly line problems," *Operations Research*, vol. 9, pp. 841– 848, 1961.

[56] P. Baldi and M. Rosen-Zvi, "On the relationship between deterministic and probabilistic directed graphical models: From bayesian networks to recursive neural networks (elsevier science ltd.)," *Neural Networks - Special issue on neural networks and kernel methods for structured domains*, vol. 18, October 2005.

[57] H. I. Volos and R. M. Buehrer, "Cognitive engine design for link adaptation: an application to multi-antenna systems," *IEEE Transactions on Wireless Communications*, vol. 9, pp. 2902–2913, Sept. 2010.

[58] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *Communications Surveys & Tutorials, IEEE*, vol. 11, no. 1, pp. 116–130, 2009.

[59] W. Headley, J. Reed, and C. da Silva, "Distributed cyclic spectrum feature-based modulation classification," in *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pp. 1200 –1204, March 31 - April 3 2008.

[60] P. Sutton, K. Nolan, and L. Doyle, "Cyclostationary signatures in practical cognitive radio applications," *Selected Areas in Communications, IEEE Journal on*, vol. 26, pp. 13 –24, Jan. 2008.

[61] T. W. Rondeau, *Application of Artificial Intelligence to Wireless Communications*. PhD thesis, Virginia Tech, 2007.

[62] T. R. Newman, B. A. Barker, A. M. Wyglinski, A. Agah, J. B. Evans, and G. J. Minden, " Cognitive engine implementation for wireless multicarrier transceivers," *Wiley Journal on Wireless Communications and Mobile Computing*, vol. 7, no. 9, pp. 1129–1142, 2007.

[63] A. He, J. Gaeddert, K. Bae, T. R. Newman, J. H. Reed, L. Morales, and C. Park, "Development of a case-based reasoning cognitive engine for IEEE 802.22 WRAN applica-

tions," *ACM Mobile Computing and Communications Review*, vol. 13, no. 2, pp. 37–48, 2009.

[64] H. I. Volos and R. M. Buehrer, "On balancing exploration vs. exploitation on a cognitive engine for multi-antenna systems," in *Proceedings of the IEEE Global Telecommunications Conference*, pp. 1–6, Nov. 2009.

[65] H. Takagi, *Queuing Analysis - A Foundation of Performance Evaluation - Volume 3: Discrete-Time Systems*. Amsterdam, The Netherlands: Elsevier Science Publishers B.V., 1993.

[66] S. A. Klugman, H. H. Panjer, and G. E. Wilmot, *Loss Models, From Data to Decisions*. New York: Wiley-Interscience, a John Wiley & Sons, Inc, second ed., 2004.

[67] S. Srinivasa and M. Haenggi, "Distance distributions in finite uniformly random networks: Theory and applications," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 940–949, November 2010.

[68] M. Haenggi, "Local delay in poisson networks with and without interference," in *Forty-Eighth Annual Allerton Conference*, (Allerton House, UIUC, Illinois, USA), September 29 - October 1 2010.

[69] nal Ufuktepe, "Unification of probability theory on time scales," *Advances in Difference Equations*, vol. 2012, December 2012.

[70] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Pub, 1st edition ed., 1997.

[71] J. Mia, W. Shib, and Y. Y. Zhouc, "Some properties of convolutions of pascal and erlang random variables," *Statistics and Probability Letters*, vol. 78, p. 23782387, March 2008.

[72] E. Furman, "On the convolution of the negative binomial random variables," *Statistics & Probability Letters*, vol. 77, p. 169172, 2007.

[73] T. S. Rappaport, *Wireless Communications: Principles and Practice.* Prentice Hall, second ed., 2002.

[74] D. Maniezzo, K. Yao, and G. Mazzini, "Energetic trade-off between computing and communication resource in multimedia surveillance sensor network," in *4th International Workshop on Mobile and Wireless Communications Network*, (Stockholm, Sweden), pp. 373– 376, November 2002.

Appendices

Appendix A

Power and Energy Consumption Models

This chapter presents models of the power and energy consumption in the transmitter module, receiver module, and application processor module of a node. The rest of the chapter is organized as follows. Section A.1 presents the computational power consumption model that is derived from a CMOS-based processor power consumption model. Sections A.2 and A.3 present the power and energy consumption models for a typical wireless transmitter and receiver, respectively.

A.1 Computational Power Consumption

The power consumption model of a CMOS-based processor, which has been proposed by Sinha et al. [38, 39], is given by

$$P_{cp} = C_{switch}\, V_{cp}^{\,2}\, f_{cp} + V_{cp}\, I_{leak}, \tag{A.1}$$

$$\text{where } V_{cp} = 4.081\, nV/Hz \times f_{cp} + 659\, mV, \tag{A.2}$$

$$I_{leak} = 1.196\, mA \times exp\left(\frac{V_{cp}}{552.76\, mV}\right), \tag{A.3}$$

$$\text{and } 59\, MHz \leq f_{cp} \leq 206\, MHz,\ 0.8V \leq V_{cp} \leq 1.5V. \tag{A.4}$$

In Equation A.1, the computational power consumption is represented by P_{cp}, V_{cp} denotes the processor core supply voltage, I_{leak} is the leakage current, and f_{cp} represents the processor clock frequency. The mean total load capacitance switched per clock cycle, denoted by C_{switch}, abstracts the processor activity factor (or utilization) and switching activity. Equation A.2 expresses the linear relationship between the processor clock frequency and the minimum core supply voltage required in order to operate at that frequency.

A macroscopic model that can abstract the microscopic and system level components of computational power consumption is derived from Equations A.1 and A.2. In this derivation, polynomial curve fitting is employed to remove the exponential terms in Equation A.1. The derived computation power consumption model expresses the non-linear polynomial relationship between computational power consumption and the processor's operating clock rate, as given by

$$P_{cp}(f_{cp}) = a_{cp}\, f_{cp}^{\,3} + b_{cp}\, f_{cp}^{\,2} + c_{cp}\, f_{cp} + d_{cp};\ a_{cp},\ b_{cp},\ c_{cp},\ d_{cp} \geq 0, \tag{A.5}$$

where a_{cp}, b_{cp}, c_{cp}, and d_{cp} are coefficients of the computational power consumption model.

A.2 Transmitter Power and Energy Consumption

The power consumption of a typical wireless transmitter, denoted by P_{tx}, is given by

$$P_{tx} = \left(P_{trs} + P_{mixer} + P_{LO} + P^{t}_{filter} + P_{amp}\right) + P_{DAC} + P_{spt}, \quad \text{(A.6)}$$

$$\text{where } P_{amp} = (\eta_{amp}\, P_t) + \beta_{amp}. \quad \text{(A.7)}$$

In Equation A.6, the terms within parenthesis refer to power consumption by the active transmitter RF components. P_{mixer}, P_{LO}, P^{t}_{filter}, P_{DAC}, and P_{spt} represent power consumption by the mixer, local oscillator, transmit filter, digital-to-analog convertor (DAC), and transmitter's signal processing tasks (such as modulation and channel coding), respectively. P_{trs} is the transient power consumption of the radio just after it is switched on and before it is operational. Transient processes such as settling of the phase-locked loop and voltage-controlled oscillator dominate the transient power consumption, while the other components such as the power amplifier (PA) settle fast and their settling time is insignificant [37]. P_{amp} denotes power consumption by the PA in order to produce a power of P_t at the PA output (or at the transmitter antenna input). As shown in Figure A.7, the PA linearized efficiency is characterized by two parameters: linear PA inefficiency denoted by η_{amp} and a constant amplifier inefficiency term denoted by β_{amp} [37]. η_{amp} is the slope of the amplifier's input-output power characteristics and β_{amp} is the minimum power consumption for zero output power determined as the y-intercept in the amplifier characteristics i.e. the amplifier's static power consumption [37]. The relationship between the PA output power, required received signal power, and channel characteristics is discussed next.

A shadowing loss model is considered for the propagation model. It is also assumed that the nodes have omni-directional antennas. The shadowing model consists of two components, namely, the path loss and log-normal fading. The path loss model sets the average received power and the log-normal random variable represents variations in the received power. Let D denote the distance between the transmitter and the receiver. In long range wireless

communication, the receiver is located in the far field of the transmitter. Let D_{far} denote the far-field distance of the transmitting antenna and λ_{sig} denote the wavelength of the transmitted signal. D_{ref} represents a small distance beyond D_{far} where the Frii's model can be used to compute the path loss. By assuming a small distance, it is ensured that the channel is a line-of-sight channel within that distance so that the free space model can be employed with a path loss exponent of 2. Beyond D_{ref}, random shadowing loss can occur and the path loss exponent is greater than 2. In this model, it is also required that the following condition be met: $D > D_{ref} > D_{far} > \lambda_{sig}$. For antennas whose physical dimension exceeds $\lambda_{sig}/2$, D_{far} is defined in terms of the Fraunhofer distance which is given by [73]

$$D_{far} = \frac{2 \times D_{ant}^2}{\lambda_{sig}}, \tag{A.8}$$

$$\text{where } D_{far} > D_{ant} > \lambda_{sig}/2. \tag{A.9}$$

In Equation A.8, D_{ant} represents the antenna's largest physical linear dimension. Assuming an operating frequency of 450 MHz, λ_{sig} is 0.67 m. In order to ensure that $D_{ant} > \lambda_{sig}/2$, it is assumed that D_{ant} is 0.5 meters. As a result, D_{far} is computed to be 0.75 meters. This value of D_{far} meets the following conditions: $D_{far} > D_{ant}$ and $D_{far} > \lambda_{sig}$. Finally, D_{ref} is chosen to be a small distance greater than D_{far}.

Let $P_{coverage}$ denote the coverage probability in a given area populated with obstacles that impose shadow losses to signal propagation. When considering the effects of shadow fading, $P_{coverage}$ is defined as given by

$$P_{coverage} = Prob\left[P_{r,dBm}(D) > P_{rmin,dBm}\right] \tag{A.10}$$

$$= Q\left(\frac{P_{rmin,dBm} - \overline{P}_{r,dBm}(D)}{\sigma_{s,dB}}\right). \tag{A.11}$$

In Equation A.10, $P_{r,dBm}$ denotes the random variable which represents received signal power, $P_{rmin,dBm}$ represents the minimum required received signal power, $\overline{P}_{r,dBm}$ represents the corresponding mean received signal power, and $\sigma_{s,dB}$ represents the standard deviation intro-

duced by shadow fading. Note that the subscripts dB and dBm indicate, respectively, that the parameter is expressed in decibels and dBm. By manipulating Equation A.11, the mean received signal power can be expressed as given by

$$\overline{P}_{r,dBm}(D) = P_{rmin,dBm} - \left(Q^{-1}(P_{coverage}) \times \sigma_{s,dB}\right), \tag{A.12}$$

where Q^{-1} represents the inverse Q–function. When considering the effects of path loss, the mean received power can also be computed as given by

$$\overline{P}_{r,dBm}(D) = \overline{P}_{r,dBm}(D_{ref}) - 10\,\eta_{pl}\,log_{10}\left(\frac{D}{D_{ref}}\right), \tag{A.13}$$

$$\text{where } \overline{P}_r(D_{ref}) = \frac{P_t\,G_t\,G_r\,\lambda_{sig}^2}{(4\,\pi\,D_{ref})^2}. \tag{A.14}$$

In Equation A.13, $\overline{P}_{r,dBm}(D_{ref})$ denotes the mean received signal power at D_{ref} distance from the transmitter and η_{pl} denotes the channel path loss exponent. Equation A.14 shows how $\overline{P}_r(D_{ref})$ is computed according to Frii's free space propagation model. G_t and G_r denote, respectively, the transmitter and receiver antenna gains. By replacing $\overline{P}_{r,dBm}$ in Equation A.13 with Equation A.12, we get an expression for the PA output power as given by

$$\begin{aligned} P_{t,dBm} = {} & P_{rmin,dBm} - \left(Q^{-1}(P_{coverage}) \times \sigma_{s,dB}\right) - 10\,log_{10}\left[\frac{G_t\,G_r\,\lambda_{sig}^2}{(4\,\pi\,D_{ref})^2}\right] \\ & +10\,\eta_{pl}\,log_{10}\left(\frac{D}{D_{ref}}\right) + LM_{dB}, \end{aligned} \tag{A.15}$$

where LM denotes the link margin. Note that the link margin has been added to the PA output power model in order to account for the miscellaneous losses in the system that are not explicitly modeled in Equation A.15. It accounts for processes that randomly influence the received signal power such as hardware process variations, filter losses, shadowing loss, and other additive background noise or interference [41].

Let P_{noise} and SNR_{min} denote, respectively, the mean receiver noise power and minimum required received signal power-to-noise power ratio. SNR_{min} is a function of the required bit-error-rate (BER), transmitter and receiver waveform (modulation, channel coding and signal processing), and channel conditions. The minimum required received signal power (receiver sensitivity) P_{rmin} is given by

$$\begin{aligned} P_{rmin} &= SNR_{min} \times P_{noise} && \text{(A.16)} \\ \text{where } P_{noise} &= N_{psd} \times BW, && \text{(A.17)} \\ \text{and } N_{psd} &= k_B \times T_{noise} \times NF. && \text{(A.18)} \end{aligned}$$

In Equation A.18, N_{psd} denotes the thermal noise power spectral density and BW is the receiver RF bandwidth. In Equation A.18, k_B is the Boltzmann's constant, T_{noise} is the receiver noise temperature, and NF is the receiver noise figure.

Using Equations A.6, A.15, A.16, the transmitter power consumption for a given radio system can be expressed as a function of D, BW, and SNR_{min} as given by

$$\begin{aligned} P_{tx} &= (G_1\, SNR_{min}\, D^{\eta_{pl}}\, BW) + G_2, && \text{(A.19)} \\ \text{where } G_1 &= \frac{\eta_{amp}\, N_{psd}\, (\sigma_s)^{-Q^{-1}(P_{coverage})}\, (4\pi)^2}{G_t\, G_r\, \lambda_{sig}^2\, D_{ref}^{\eta_{pl}-2}}\, LM, && \text{(A.20)} \\ \text{and } G_2 &= P_{trs} + P_{DAC} + \beta_{amp} + P_{mixer} + P_{LO} + P_{filter}^{t}. && \text{(A.21)} \end{aligned}$$

G_1 and G_2 are constants in the model. The power consumption by signal processing tasks at the transmitter is not taken into consideration in the model. Maniezzo [74] presents a high level communication power consumption model similar to the one presented in Equation A.19.

Power consumption in digital-to-analog converter The power consumption of a binary-weighted current-steering Digital-Analog Converter (DAC) is given by [40]

$$\begin{aligned} P_{DAC} &= P^s_{DAC} + P^d_{DAC}, && \text{(A.22)} \\ \text{where } P^s_{DAC} &= 0.5 \times V_{supply} \times I_{LSB} \times (2^{n_{dac}} - 1) && \text{(A.23)} \\ \text{and } P^d_{DAC} &\approx n_{dac} \times C_{prstc} \times (2\,BW + f_{corner}) \times V^2_{supply}. && \text{(A.24)} \end{aligned}$$

In Equation A.22, P^s_{DAC} and P^d_{DAC} denote the static and dynamic power consumption components, respectively. I_{LSB}, V_{supply}, n_{dac}, C_{prstc}, and f_{corner} denote, respectively, the unit current source corresponding to the least significant bit in the DAC circuit, the DAC supply voltage, the resolution of the DAC (i.e., number of bits representing amplitude), the parasitic capacitance of each switch, and the corner frequency of flicker noise.

Transmitter energy consumption The energy consumption at the transmitter is given by

$$\begin{aligned} E_{tx} &= E_{tx1} + \left((P_{tx} - P_{trs}) \times T_{tx}\right), && \text{(A.25)} \\ \text{where } E_{tx1} &= P_{trs}\, T_{trs} + P_{static}\, T_{idle}. && \text{(A.26)} \end{aligned}$$

In Equation A.25, E_{tx1} is a constant term in the energy consumption model that is independent of the transmission time and T_{tx} represents the total time taken to transmit all the information bits successfully. T_{idle} and T_{trs} represent, respectively, the idle time and transient time in the RF circuit. P_{static} is the static power consumption by the RF circuit, especially the PA, when it is powered-up but idle.

A.3 Receiver Power and Energy Consumption

Let P_{rx} and E_{rx} denote, respectively, the total power and energy consumption at a typical wireless receiver. The total power and energy consumption at the receiver is given by

$$\begin{aligned}
P_{rx} &= \left(P_{LNA} + P_{mixer} + P_{LO} + P^{r}_{filter}\right) + P_{trs} + P_{ADC} + P_{spr}, && \text{(A.27)}\\
E_{rx} &= E_{rx1} + (T_{rx} \times E_{rx2}) + E_{spr} && \text{(A.28)}\\
\text{where } E_{rx1} &= (P_{trs} \times T_{trs}) + (P_{static} \times T_{idle}), && \text{(A.29)}\\
E_{rx2} &= \left(P_{LNA} + P_{mixer} + P_{LO} + P^{r}_{filter}\right) + P_{ADC}, && \text{(A.30)}\\
E_{spr} &= E_{bdec} \times N_{data}, && \text{(A.31)}\\
\text{and } P_{spr} &= \frac{E_{bdec}}{T_{sprb}}. && \text{(A.32)}
\end{aligned}$$

In Equation A.27, the term in parenthesis represents the total power consumption by the active receiver RF components. P_{LNA}, P_{mixer}, P_{LO}, P^{r}_{filter}, P_{ADC}, and P_{spr} denote, respectively, the power consumption by the low noise amplifier, mixer, local oscillator, and receiver RF filter, analog-to-digital convertor (ADC), and receiver signal processing tasks (such as channel decoding). In Equation A.28, E_{rx1} and E_{rx2} are constant factors defined by Equations A.29 and A.29, respectively. E_{spr} and T_{rx} denote, respectively, the energy consumption by the receiver signal processing tasks and total time taken to receive the information bits successfully. It is assumed that the transmission and reception times are approximately equal. E_{bdec} and T_{sprb} denote, respectively, the energy and time consumption to decode an information bit.

Analog-to-digital convertor The power consumption of a typical ADC is given by [40]

$$P_{ADC} \approx \frac{3 \times V^{2}_{supply} \times L_{chlen} \times (2\ BW + f_{corner})}{10^{(-0.1525\ n_{adc} + 4.838)}}, \qquad \text{(A.33)}$$

where L_{chlen} denotes the ADC's minimum channel length and n_{adc} represents the ADC's resolution.

Receiver signal processing The energy consumption at the receiver for viterbi decoding using the StrongARM SA-1110 is given by [37]

$$\begin{aligned} E_{bdec} &= C_{swbit}\, V_{cp}^2 \,+\, T_{sprb}\, V_{cp}\, I_{leak} \qquad \text{(A.34)} \\ \text{where } C_{swbit} &= 51.6\, nF \,\times 2.62^{K_{cl}} \\ T_{sprb} &= 219\, nsec \,\times 2.99^{\beta_{spr}}, \\ \text{and } \beta_{spr} &= K_{cl} \frac{206\, MHz}{f_{cp}}. \end{aligned}$$

In Equation A.34, V_{cp} and I_{leak} are defined as given by Equations A.2 and A.3. C_{swbit} denotes the switch capacitance per bit and K_{cl} denotes the decoder's constraint length.

Appendix B

Notation

Table B.1 summarizes the notation throughout this dissertation. Note that, i, j, and k in the subscript or superscript of a symbol are used to denote an index such as a node index or a task index and represent the parameter's dependence on the node or task index. These indices may be dropped when the parameter represented by the symbol is independent of the node or task index. For instance, P_{cp}^{ij} denotes the power consumed while executing task F_i in node N_j while P_{cp} represents the same power consumption when it does not vary between tasks or nodes as in the case of a homogeneous network and a user application comprising a homogeneous set of tasks. Next, the mean of a random variable is represented by placing a bar on the symbol, as in $\overline{\gamma}_{rx}$ where γ_{rx} denotes the received signal power-to-noise power ratio.

Table B.1: Notation employed in this dissertation.

Symbol	Definition
α_{pc}	Power control scaling factor
β_{amp}	Constant inefficiency term in the power amplifier model
$\beta\,(x, y, z)$	Generalized beta distribution with parameters x, y, z
	Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
γ_{rx}	Received signal power-to-noise power ratio
γ_{tx}	Transmitted signal power-to-noise power ratio
δ	Improvement in objective computed between consecutive iterations of proposed workload allocation algorithm
η_{amp}	Power amplifier inefficiency
η_{pl}	Channel path loss exponent
θ	A parameters in the link connectivity model
λ_{sig}	Transmitted signal wavelength
μ_{arr}	Message arrival rate at a queue
μ^{i}_{arr}	Arrival rate of node N_i's forward link communication queue
$\tilde{\mu}_{arr}$	Probability that a message does not arrive in an arbitrary time slot
μ_{cmrl}	Message service rate in the reverse link communication queue at a service provider
μ_{cp}	Message service rate in the request processing queue at a service provider
μ_{dep}	Message departure rate at a queue
μ^{i}_{dep}	Departure rate of node N_i's forward link communication queue
μ_{svc}	Message service rate at a queue
ρ	Variable to denote current iteration of proposed workload allocation algorithm
ρ_{net}	Network throughput or the net departure rate of the TDMA network in the forward link
σ_{qdelay}	Standard deviation of queuing delay
σ_s	Standard deviation of lognormal shadowing
σ_g	Standard deviation of channel fading (Rayleigh) gain
ψ_{load}	Offered load to a queue

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
ψ_{load}^{i}	Offered load in node N_i's forward link communication queue
a_{cp}^{ij}	Power consumption model coefficient for the execution of F_i in N_j
a_n	A parameter of the distance distribution model
$B\left(x,y\right)$	Beta function with parameters x and y
b_{cp}^{ij}	Power consumption model coefficient for the execution of F_i in N_j
b_{ij}	Number of bits of data transferred from task i's output to task j's input
BM_{ij}	Branch metric associated with the allocation of F_i to N_j, used in proposed workload allocation algorithm
BW	Wireless system communication bandwidth
b_n	A parameter of the distance distribution model
c_{cp}^{ij}	Power consumption model coefficient for the execution of F_i in N_j
C_{cpe}^{i}	Computational energy energy consumptions of N_i for a partial allocation scheme in the proposed workload allocation algorithm
$C_{min}(\rho)$	Minimum WDC energy cost achieved for ρ^{th} iteration of the proposed workload allocation algorithm
C_{net}	Total cost computed for a partial allocation in the proposed workload allocation algorithm
C_n	A parameter of the distance distribution model
C_{norm}	A parameter of the distance distribution model
C_{prstc}	Parasitic capacitance of each switch in DAC
C_{swbit}	Switch capacitance per bit in receiver signal processor
C_{switch}	Average capacitance being switched per processor cycle
C_{T}^{i}	Total energy energy consumptions of N_i for a partial allocation scheme in the proposed workload allocation algorithm
D_{ij}	Distance between transmitter node N_i and receiver node N_j

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
D_{ant}	Antenna's largest physical linear dimension
d_{cp}^{ij}	Power consumption model coefficient for the execution of F_i in N_j
D_{far}	Transmitter far-field distance
D_n	Euclidean distance of the n^{th} nearest node from the origin in a binomial point process
D_{ref}	Close-in reference distance from transmitter's location
E	Set of all edges in a task graph
E_{ap}	Maximum WDC energy cost tolerated by user application
E_{bdec}	Energy to decode an information bit
E_{cmb}^{ij}	Total energy consumption per channel bit for communication on link L_{ij}
E_{cp}^{ij}	Energy consumed for processing F_i in N_j
E_{cpt}^{i}	Energy available for computing purposes in node N_i
E_{cp}^{total}	Energy consumed to complete processing user application
E_{node}^{i}	Total energy consumption in node N_i
E_{nodeT}^{ijk}	Total energy consumption in node N_k incurred due to allocation of F_i to N_j
E_{nw}	Total energy consumption in the WDC network
E_{pktrl}^{ijk}	Total energy consumption for the first k transmission trials an arbitrary packet over link L_{ij}
E_{pktx}^{ij}	Total energy consumption for successfully transmitting an arbitrary packet over link L_{ij}
E_{pktx}^{ijk}	Total energy consumption for successfully transmitting the k^{th} packet of a data file over link L_{ij}
E_{remote}	Total energy consumption of a WCCN when adopting remote data processing in the UAV network example of Section 3.9

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
E_{rx}^{ij}	Total energy consumption for the successful reception of a data file over link L_{ij}
E_{rxb}^{ij}	Energy consumption at the receiver for successfully transmitting an arbitrary channel bit over link L_{ij}
E_{rx1}	Constant in receiver energy consumption model
E_{rx2}	Dynamic coefficient in receiver energy consumption model
E_{sav}^{m}	Energy savings achieved in the master node by employing WDC
E_{sav}^{uav}	Percentage energy savings achieved by the UAV network in the UAV network example of Section 3.9
E_{spr}	Receiver signal processing energy consumption
E_{supply}^{i}	Remaining energy supply in i^{th} node in the network
E_{tx}^{ij}	Total energy consumption for the transmission of a data file over link L_{ij}
E_{tx1}	Constant in transmitter energy consumption model
E_{txb}^{ij}	Transmitter energy consumption for successfully transmitting an arbitrary channel bit over link L_{ij}
E_{T}^{ij}	Total energy consumption associated with execution of F_i in node N_j
$\acute{E}_{T}^{ij}$	Energy consumption term that takes into consideration the energy consumption costs associated with execution of F_i in node N_j, and the cost incurred by interaction between task F_i and the sink task
$E_{txRemote}$	Energy consumption to transmit processed data to a remote node when the UAV network pursues the in-network processing option, in the UAV network example of Section 3.9
$E_z(\cdot)$	Exponential integral function
E_{wdc}	Total energy consumption of a WCCN when adopting in-network data processing in the UAV network example of Section 3.9

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
EL_i	Estimated level of task F_i
F	Set of all tasks in a task graph
$\acute{F}$	Set of unscheduled tasks of a task graph
f_{corner}	Corner frequency of the flicker noise in DAC
f^i_{cp}	Processor clock frequency in node N_i in the network
f^i_{cpd}	Processor clock frequency employed in node N_i when using WDC
f^i_{cpl}	Processor clock frequency employed in node N_i when processing all tasks locally
f_{cpmax}	Maximum supported processor clock frequency
$f_{E^{ij}_{pktrl}}$	Probability distribution of E^{ij}_{pktrl}
$f_{E^{ij}_{pktx}}$	Probability distribution of E^{ij}_{pktx}
$f_g(g)$	Probability distribution of channel fading gain
$F_g(g)$	Cumulative distribution function of channel fading gain
F_i	Symbolic representation of task i in an application's task graph
$f_{T_{cpq}}$	PMF of latency experienced by an arbitrary message in the request processing queue at a service provider
$F_{P_{cpt}}$	CDF of P_{cpt}
$f_{T^i_{cmfl}}$	PMF of T^i_{cmfl}
$f_{T_{cmrl}}$	PMF of latency experienced by an arbitrary message in the reverse link communication queue at a service provider
$f_{T^i_{pktotal}}$	PMF of $T^i_{pktotal}$
$f_{T_{qdelay}}(t)$	Probability mass function of T_{qdelay}
$F_{T_{qdelay}}(t)$	Cumulative density function of delay experienced by an arbitrary message in a queue

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
$f_{T^i_{rt}}$	PMF of response time experienced by a service request originating from node N_i
$f_{T^i_{sum}}(t)$	Probability mass function of T^i_{sum}
$f_{T^{i,j}_{sum}}$	PMF of $T^{i,j}_{sum}$
$F_{T^i_{sum}}(t)$	Cumulative density function of T^i_{sum}
f_{sig}	Transmitted signal center frequency
FT^{ij}_{cp}	Computational finish time of F_i when executed in N_j
$F\acute{T}^{ijk}_{cm}$	Estimated finish time of the communication activity originating from task F_i and destined to sink task F_k given that F_i is allocated to node N_j
FT^{ijk}_{cm}	Finish time of a communication activity originating from F_k destined to F_i when F_i is allocated to N_j
G_1	Constant in transmitter power consumption model
G_2	Constant in transmitter power consumption model
G_3	A variable in the adaptive transmitter power consumption model defined by Equation 3.14
G_{cp}	Symbolic representation of application's task graph
G_{cm}	Symbolic representation of network's communication graph
g_{ij}	Channel fading gain in link L_{ij}
g_{min}	Lowest channel fading gain that can be supported by the system
G_r	Receiver antenna gains
G_t	Transmitter antenna gains
$H(g)$	Heaviside step function of g
I_{leak}	Processor leakage current
I_{LSB}	DAC unit current source corresponding to the least significant bit

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
k_B	Boltzmann's constant
K_{cl}	Receiver decoder constraint length
K_m	Modulation index
L	Set of all wireless links in a communication graph
L_{ij}	Symbolic representation of wireless link between N_i and N_j in a communication graph
L_{chlen}	Minimum channel length in ADC
L_{sorted}	Set of topologically sorted tasks of a task graph
LM	Link margin
M_k	Number of immediate successors of task F_k
N	Set of all nodes in a communication graph
n_{adc}	ADC resolution
N_{bits}	Number of information bits input/output to/from a task when $N_{bits}^{in} = N_{bits}^{out} = N_{bits}$
N_{bits}^{in}	Number of bits of data input to a task
N_{bits}^{out}	Number of bits of data generated as a result of processing a task
N_{bpp}	Number of bits used to represent a single FFT point
$N_{channels}$	Number of RF frequency channels available for communication
N_{cycles}^{ij}	Number of processor cycles consumed for processing F_i in N_j
n_{dac}	DAC resolution
N_{data}	Total number of uncoded bits (including protocol and application data bits) that constitute a user application data file associated with a single communication activity
N_G^i	Set of candidate nodes that qualify for the execution of F_i
N_i	Symbolic representation of node i in a communication graph

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
N_L^i	Set of candidate nodes that have not been evaluated for F_i's allocation in the proposed workload allocation algorithm
N_{pdata}	Number of uncoded bits (including protocol and application data bits) in a single packet
N_{pkts}	Number of packets in a single user application data file
N_{psd}	Thermal noise power spectral density
N_{pts}	Number of FFT points
N_{nodes}	Number of nodes in a WCCN cluster
N_{queue}	Number of messages in a queue in an arbitrary time slot
N_{tasks}	Total number of tasks in an application's task graph
N_{cpw}^{mr}	Computational workload assigned to master node
N_{cpw}^{sl}	Total computational workload assigned to all slave nodes together
N_{trials}	Number of Bernoulli trials associated with the arrival process of a message
NF	Receiver noise figure
O_{ij}	Objective function associated with allocation of F_i to node N_j
P_{ADC}	ADC circuit power consumption
P_{amp}	Power amplifier power consumption
P_{blk}	Blocking probability of a queue
p_{cmflq}^{ij}	Probability that there are j messages in node N_i's forward link communication queue
$P_{coverage}$	Coverage probability
p_{cmrlq}^{j}	Probability that there are j messages in the reverse link communication queue at a service provider
P_{cp}^{ij}	Power consumed for processing F_i in N_j

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
p^j_{cpq}	Probability that there are j messages in the request processing queue at a service provider
P^i_{cpt}	Power available for computing purposes in node N_i
P_{DAC}	DAC circuit power consumption
P^s_{DAC}	DAC circuit static power consumption
P^d_{DAC}	DAC circuit dynamic power consumption
P_{empty}	Probability that a queue is empty
P^r_{filter}	Receiver RF filter power consumption
P^t_{filter}	Transmitter RF filter power consumption
P_{LNA}	Low noise amplifier circuit power consumption
P_{LO}	RF local oscillator circuit power consumption
P_{mixer}	RF mixer power consumption
P^i_{node}	Total power consumption in node N_i
P_{noise}	Mean receiver noise power
P_{nw}	Total power consumption of the network
p^i_{occup}	Probability that there are i messages in a queue
$p^{j,k}_{occup}$, $p^{i,j,k}_{occup}$	Mixture weights in the response time PMF model
P^{cp}_{outage}	Outage probability of power resources for computational purposes
P_r	Random variable representing received signal power
P_{rmin}	Minimum required received signal power
P^{ij}_{rx}	Power consumption at the receiver when N_i is transmitting to N_j
P^m_{sav}	Power savings achieved in master node by employing WDC
P^{nw}_{sav}	Power savings achieved in a WCCN by employing WDC
P_{spt}	Transmitter signal processing power consumption
P_{static}	RF circuit static power consumption

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
$P^{ij}_{success}$	Probability of success of signal transmission over link L_{ij}
P^{i}_{supply}	Remaining power supply in node N_i in the network
P^{min}_{supply}	Minimum remaining power supply of a node in the network
P^{max}_{supply}	Maximum remaining power supply in a node in the network
P_t	Power dissipation at output of power amplifier
P_{trs}	RF transient power consumption
P^{ij}_{tx}	Power consumption at the transmitter when N_i is transmitting to N_j
P^{ijk}_{tx}	k^{th} instance of random variable P^{ij}_{tx}
P_{txout}	Power dissipated at output of transmitter antenna
$Pred\{F_i\}$	Set of immediate predecessors of task F_i in a task graph
Q^{i}_{cmfl}	Size of node N_i's forward link communication queue
Q_{size}	Size of a queue
Q_{cmrl}	Size of the reverse link communication queue at a service provider
Q_{cp}	Size of the request processing queue at a service provider
R_{baud}	Radio modem transmission symbol rate (or baud rate)
R_c	Radius within which all nodes in the network are located
R_{code}	Channel coding rate
R_{comp}	Ratio between N^{out}_{bits} and N^{in}_{bits}
R_δ	Minimum value of δ expected by the user application.
RT^{i}_{cm}	Ready time of communication from F_i
RT^{ij}_{cp}	Ready time of F_i when allocated to N_j
r^{i}_{cpt}	Power available for computational purposes in node N_i for a given outage probability
R_{txout}	Radio transmission rate expressed in bits per second

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
S_{arr}	Random variable which represents the slot in a TDMA schedule when a tagged message arrives
S_{trial}	The TDMA slot when Bernoulli trials corresponding to the arrival of a tagged message begin
SNR_{min}	Minimum required received signal power-to-noise power ratio
ST_{cp}^{ij}	Computational start time of F_i when executed in N_j
ST_{cm}^{ijk}	Start time of communication from F_k to F_i when F_i is executed in N_j
T_{δ}	Number of slots that a tagged message has to wait until it is the node's turn to transmit a message from its forward link communication queue
T_{bit}^{ij}	Time taken to successfully transmit an arbitrary channel bit from N_i to N_j
T_{bitp}	Deterministic propagation time per channel bit
T_{cmstp}^{ij}	Stop time of communication from F_i to F_j
T_{cmstr}^{ij}	Start time of communication from F_i to F_j
T_{cpstp}^{i}	Stop time of task F_i
T_{cpstr}^{i}	Start time of task F_i
T_{cp}^{ij}	Time consumed to complete processing F_i in N_j
T_{cpd}^{ij}	Time consumed to process task F_i in node N_j when clock frequency is f_{cpd}^{j}
T_{cpl}^{ij}	Time consumed to process task F_i in node N_j when clock frequency is f_{cpl}^{j}
T_{cp}^{total}	Total time consumed in executing a user application
T_{cpmax}	Range of execution times of tasks in a task graph
T_{cmfl}^{i}	Total forward link communication latency experienced by an arbitrary message in the i^{th} nearest neighbor node
T_{idle}	RF circuit idle time
T_{life}^{r}	Network lifetime achieved with the remote processing option in the UAV network example of Section 3.9

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
T^{w}_{life}	Network lifetime achieved with the in-network processing option in the UAV network example of Section 3.9
T_{lifesv}	Percentage improvement in lifetime achieved by the UAV network in the UAV network example of Section 3.9
T_{noise}	Receiver noise temperature
T_{pktp}	Deterministic propagation time per packet
$T^{i}_{pktotal}$	Number of slots that the tagged message has to wait in addition to T_{δ} until it is successfully transmitted to the receiver
T^{ij}_{pktx}	Time taken to successfully transmit an arbitrary packet over link L_{ij}
T^{ijk}_{pktx}	Time taken to successfully transmit the k^{th} packet in a data file over link L_{ij}
T^{n}_{pktx}	Communication latency per message experienced by the n^{th} nearest neighbor
$\overline{T}^{n}_{pktx}$	Mean communication latency per message experienced by the n^{th} nearest neighbor
T_{qdelay}	Total delay experienced by an arbitrary message in a queue
$\overline{T}_{qdelay}$	Mean delay experienced by an arbitrary message in a queue
T^{ij}_{qpk}	Time taken to successfully transmit the j^{th} packet in node N_i's transmitter queue after it begins processing the message
T_{sprb}	Time required per information bit for receiver signal processing
T_{trs}	RF circuit transient time
T^{ij}_{rx}	Time taken to successfully receive a user application data file over link L_{ij}
T_{s}	The period of a TDMA schedule
T^{i}_{supply}	Lifetime of the power supply in node N_i
T^{i}_{sum}	Sum of i i.i.d. service time random variables

Continued on next page

Table B.1 – continued from previous page

Symbol	Definition
$T_{sum}^{i,j}$	Sum of j message transmission times in node N_i's queue
T_{svc}^{j}	Service time of the j^{th} message in a queue
T_{tx}^{ij}	Time taken to successfully transmit a user application data file over link L_{ij}
TS_{ch}^{i}	Time stamp of channel i
TS_{cp}^{i}	Time stamp of computing resources in node N_i
TS_{cm}^{i}	Time stamp of communication resources in node N_i
TS_{cht}^{ijr}	Channel r's timestamp when considering the allocation of F_i to N_j
TS_{cmt}^{ijs}	Timestamp of node N_s's communication resources when considering the allocation of F_i to N_j
U	Set of all possible paths from source to sink tasks
U_j	j^{th} subset of U
U_{ij}	A subset of U_j comprising tasks along the j^{th} path in the task graph starting from F_i down to the sink task
V_{cp}	Processor core supply voltage
V_{supply}	Voltage supply to ADC and DAC circuits
w_1,w_2	Weights assigned to objectives in a multi-objective problem
x_{ij}	Binary-valued decision variable which represents the allocation of F_i to N_j
X_{max}	Maximum number of packet transmission attempts permitted per packet
X_{pktx}^{ij}	Random variable representing the number of transmission attempts of an arbitrary packet over link L_{ij}
X_{tx}^{ij}	Total number of packet transmission attempts corresponding to the transfer of a single data file
$X_{\rho,i}$	Variable that stores the index of the node assigned to F_i in ρ^{th} iteration of the proposed workload allocation algorithm
y_{ijk}	Takes value 1 iff communication from F_i to F_j is allocated to channel k