Efficient Resource Allocation Schemes for Wireless Networks with with Diverse Quality-of-Service Requirements

Akshay Kumar

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

T. Charles Clancy, Chair
Ravi Tandon, Co-Chair
Jeffrey H. Reed
Michael S. Hsiao
Michael R. Taaffe

July 15, 2016
Blacksburg, Virginia

Keywords: Quality-of-Service, Dynamic Resource Allocation, Cross-Layer Optimization, Distributed Storage, M2M Communication, Delay-Optimal Scheduler

Efficient Resource Allocation Schemes for Wireless Networks with Diverse Quality-of-Service Requirements

Akshay Kumar

(ABSTRACT)

Quality-of-Service (QoS) to users is a critical requirement of resource allocation in wireless networks and has drawn significant research attention over a long time. However, the QoS requirements differ vastly based on the wireless network paradigm. At one extreme, we have a millimeter wave small-cell network for streaming data that requires very high throughput and low latency. At the other end, we have Machine-to-Machine (M2M) uplink traffic with low throughput and low latency. In this dissertation, we investigate and solve QoS-aware resource allocation problems for diverse wireless paradigms.

We first study cross-layer dynamic spectrum allocation in a LTE macro-cellular network with fractional frequency reuse to improve the spectral efficiency for cell-edge users. We show that the resultant optimization problem is NP-hard and propose a low-complexity layered spectrum allocation heuristic that strikes a balance between rate maximization and fairness of allocation. Next, we develop an energy efficient downlink power control scheme in a energy harvesting small-cell base station equipped with local cache and wireless backhaul. We also study the tradeoff between the cache size and the energy harvesting capabilities. We next analyzed the file *read* latency in Distributed Storage Systems (DSS). We propose a heterogeneous DSS model wherein the stored data is categorized into multiple classes based on arrival rate of *read* requests, fault-tolerance for storage etc. Using a queuing theoretic approach, we establish bounds on the average *read* latency for different scheduling policies. We also show that erasure coding in DSS serves the dual purpose of reducing *read* latency and increasing the energy efficiency.

Lastly, we investigate the problem of delay-efficient packet scheduling in M2M uplink with heterogeneous traffic characteristics. We classify the uplink traffic into multiple classes and propose a proportionally-fair delay-efficient heuristic packet scheduler. Using a queuing theoretic approach, we next develop a delay optimal multiclass packet scheduler and later extend it to joint medium access control and packet scheduling for M2M uplink. Using extensive simulations, we show that the proposed schedulers perform better than state-of-the-art schedulers in terms of average delay and packet delay jitter.

Acknowledgments

I owe my deepest gratitude to Dr. T. Charles Clancy and Dr. Ravi Tandon, for their valuable guidance, technical inputs and encouragement throughout my course of study at Virginia Tech. They allowed me a free hand in exploring many ideas and I have gained immensely from them. I would also like to thank my committee members Dr. Jeffrey H. Reed, Dr. Michael R. Taaffe and Dr. Michael S. Hsiao for their assistance in finalizing this dissertation. I would like to express my gratitude to Dr. Ahmed Abdelhadi for being a great mentor and for numerous technical discussions and suggestions that have found their way into this dissertation.

I would like to thank my colleagues Priyabrata Parida, Mehdi Naderi Soorki, Amuru Sai Dhiraj, Mehrnaz Afshang and Avik Sengupta for discussions on various topics on wireless communications. I greatly appreciate the help of the Hume Center and Wireless@VT staff including Ms. Nancy Goad, Ms. Hilda Reynolds, Ms. Leslie Sullivan and Ms. Sonya Rowe for taking excellent care of many of our administrative needs.

Most of all, I would like to thank my father Karambir, my mother Anita, and my brother Ajay, whose constant and limitless support, motivation, and unwavering belief in me had a great part in nurturing my dreams and bringing this work to completion.

Funding Acknowledgments

This work was supported in part by Award No. 1134843 awarded by the National Science Foundation, Security and Software Engineering Research Center.

Contents

1 Introduction **1**

1.1 Contributions 2

1.1.1 Dynamic spectrum allocation in macrocell network 2

1.1.2 Energy-efficient power control in small cells 3

1.1.3 Read latency analysis in Distributed Storage Systems 4

1.1.4 Delay-efficient packet schedulers for M2M uplink 5

1.1.5 Joint delay-efficient packet scheduling and channel assignment for M2M uplink 6

1.2 List of Publications 7

2 Background **9**

2.1 Dynamic Programming 9

2.2 Branch-and-Bound for MINLP problems 11

2.3 Delay characterization using Queuing theory 12

2.3.1 FCFS scheduling 14

2.3.2 Non-Preemptive resume priority scheduling 15

2.3.3 Preemptive priority scheduling 16

3 Dynamic Spectrum Allocation in a Macrocell Network **17**

3.1 Cooperative Spectrum Sharing Architecture 18

3.1.1 Spectrum Lease Request 19

3.2 Resource Allocation at the LTE Network 21

3.2.1 System Model . . . 22
3.2.2 Propagation and Rate Models . . . 25
3.2.3 Joint DRA Problem Formulation . . . 26
3.2.4 Layered FFR based DRA Problem Formulation . . . 28
3.3 Simulation Results . . . 33
3.3.1 Variation in MME Transmission Permission . . . 35
3.3.2 5 MHz Leased Spectrum, Transmission in all Cells . . . 36
3.3.3 Leased Spectrum BW Varies, Transmission in all cells . . . 39
3.4 Conclusions . . . 40

4 Power Control in Cache Enabled Energy Harvesting Small Cells 41
4.1 System Model . . . 42
4.2 Problem Formulation and Solution Methodology . . . 46
4.3 Simulation Results . . . 48
4.4 Conclusions . . . 50

5 Read Latency in Heterogeneous Distributed Storage Systems 52
5.1 System Model . . . 53
5.1.1 Latency and Energy Efficiency . . . 55
5.2 Preliminaries . . . 59
5.2.1 Lower Bound on Average Latency . . . 59
5.2.2 Upper Bound on Average Latency . . . 60
5.3 Main Results . . . 61
5.3.1 Main Results . . . 62
5.4 Quantitative Results and Discussion . . . 64
5.4.1 Impact of fault-tolerance and service rate . . . 65
5.4.2 Impact of coding on energy efficiency and storage space . . . 65
5.4.3 Impact of number of servers in DSS . . . 67
5.4.4 Impact of general service time . . . 67

5.4.5 Impact of heavy-tailed arrival distribution 68
5.4.6 Impact of number of redundant requests 70
5.5 Conclusions 71

6 Delay-Efficient Packet Scheduler for M2M Uplink 72
6.1 System Model 73
6.2 Problem Formulation 75
6.2.1 Utility Functions 75
6.2.2 System utility function 77
6.2.3 Delay-optimal scheduling policy 77
6.3 Proposed Heuristic Packet Scheduler 77
6.3.1 Service order between PU and ED classes 78
6.3.2 Service order among PU classes 78
6.3.3 Service order among ED classes 78
6.3.4 Service order among packets of same class 79
6.3.5 Optimization search space 79
6.4 Simulation Results 79
6.4.1 Toy Case 1: One PU and One ED class 80
6.4.2 Toy Case 2: Two PU and Two ED classes 82
6.5 Conclusions 86

7 Joint Delay-Optimal MAC and Packet Scheduler for M2M Uplink 87
7.1 System Model 89
7.2 Problem Formulation 91
7.2.1 System utility function 91
7.2.2 Single-stage Optimization Problem 92
7.2.3 Iterative Optimization Problem 93
7.3 Optimal Scheduler 96
7.3.1 Complexity Analysis 97

7.4 Jointly Optimal MA-AS Scheduler 99
7.4.1 Centralized Optimization at M2M AS 99
7.4.2 Distributed Optimization 100
7.4.3 Complexity Analysis 102
7.5 Joint Subcarrier Allocation and Packet Scheduler 103
7.5.1 Centralized Optimization at M2M AS 103
7.5.2 Proposed Distributed Optimization Framework 105
7.5.3 Complexity Analysis 106
7.6 Numerical Results 108
7.6.1 Multiclass Scheduler at AS 108
7.6.2 Joint Multiclass Scheduler at MAs and AS 111
7.6.3 Joint Subcarrier Assignment and Packet Scheduling 112
7.7 Conclusions 115

8 Conclusions **117**

Appendices **119**

A Proofs of Results on Read Latency **120**
A.1 Proof of Lemma 5.1: Stability Conditions 120
A.1.1 FCFS Scheduling 120
A.1.2 Priority Scheduling 121
A.2 Proof of Theorem 5.1: Upper Bound on Average Latency 122
A.2.1 FCFS Scheduling 122
A.2.2 Non-preemptive Priority Scheduling 123
A.2.3 Preemptive Priority Scheduling 124
A.3 Proof of Theorem 5.2: Lower Bound on Average Latency 125
A.3.1 FCFS Scheduling 125
A.3.2 Non-preemptive Priority Scheduling 126
A.3.3 Preemptive Priority Scheduling 128

Bibliography **130**

List of Figures

2.1 A heterogeneous queueing system with R traffic classes. 14

3.1 Spectrum Sharing Framework. 18

3.2 (a) The Fractional Frequency Reuse - 3 scheme in a 7 cell cluster; (b) Mapping of location indices of restricted transmission to eNB sectors. 23

3.3 (a) Comparison of Average Sum Throughput of the System with varying MME transmission permissions. The symbols representing different schemes are: $\star$ - Scheme 1; $\triangledown$ - Scheme 1 Bintprog; $\circ$ - Naive FFR-3 Scheme.; (b) Boxplot showing the 25th and 75th percentile, the mean (circle) and median (line) of the rates. The line extending from the boxes show the outliers. 35

3.4 (a) Comparison of Average System Throughput of different schemes with time. (b) Comparison of fraction of UEs served by the schemes across time. 36

3.5 Comparison of Average Sum Throughput of the System. The number in the bars represent the average number of users served by the schemes. 37

3.6 Comparison of Average Jain Index of the System. 38

3.7 Comparison of Average Per UE Sum Throughput of Cell Center vs Cell Sectors. 38

3.8 Comparison of Average Sum Throughput across time with changing lease bandwidths. 39

4.1 System Model 43

4.2 Energy dynamics for $M = 80$ and $\lambda = 0.1$. 48

4.3 Impact of energy harvesting and cache size on available energy. 49

4.4 Impact of energy harvesting and caching on throughput. 50

4.5 Effect of cache size and energy harvesting on energy available in system. . . 51

5.1 System Model for a Heterogeneous Distributed Storage System. 54

5.2 MDS codes for data storage in a two-class Fork-Join system. 55
5.3 System state evolution: two-class FJ system with FCFS. 55
5.4 Variation of total power consumption of DSS across multiple busy periods and idle periods. The switch to idle state happens only for idle periods with duration greater than d_l but it then results in a wake-up latency of w_l. . . . 58
5.5 Markov chain for a $(3, 2)$ Fork-Join system. 59
5.6 Latency of a data-class increases with increase in its code-rate and decreases with increase in service rate. 66
5.7 Tradeoff between energy efficiency of DSS and storage space per file with variation in code-rate. 66
5.8 Energy efficiency increases and attains a maxima as number of servers is increased while latency behaves in an inverse fashion. 67
5.9 A light tailed general service distribution (Pareto-distribution with $\alpha = 6$) results in monotonically decreasing latency as a function of code-rate. Energy efficiency follows an inverse behavior. 68
5.10 A heavy tailed general service distribution (Pareto-distribution with $\alpha = 1.1$) results in minimal latency and maximum energy efficiency point as the code-rate is increased. 69
5.11 A heavy tailed inter-arrival time distribution (Pareto-distribution with $\alpha = 1.5$) results in monotonically increasing latency (and monotonically decreasing energy efficiency) as the code-rate is increased. 69
5.12 Sending redundant requests reduces latency and improves energy efficiency. . 70

6.1 System model illustrating queuing process at M2M AS. 74
6.2 Utility function for PU packet of class i. 76
6.3 Utility function for ED packet of class i. 76
6.4 Selection of the optimal PU latency threshold l_t as a function of b. 81
6.5 Impact of criticality of ED application (set by a) on system utility. 82
6.6 Impact of ED arrival rate on the system utility. 83
6.7 System utility with delay-homogeneous M2M traffic. 83
6.8 System utility with delay-heterogeneous M2M traffic. 84
6.9 Impact of using optimal latency threshold for ED class 1. 85

6.10 Utility performance for small but equal PU failure penalty. 85

6.11 Utility performance for unequal PU failure penalty. 86

7.1 System model for a heterogeneous M2M uplink system showing the queuing process at M2M aggregators and the M2M application server. The different packet colors at AS indicate different delay classes. 90

7.2 Sigmoidal utility function for delay-sensitive traffic. 91

7.3 Sigmoidal utility function for delay-tolerant traffic. 92

7.4 Complexity of iterative and direct optimization schemes. 98

7.5 Sequence diagram illustrating the Distributed optimization at n^{th} iteration. . 102

7.6 Information exchange overhead as R is varied. 104

7.7 Comparison of system utility for $R = 4$ classes. 109

7.8 Delay variance of class 1 for different schedulers. 110

7.9 Delay variance of class 4 for different schedulers. 110

7.10 Convergence performance of distributed optimization for geometric packet size distribution. 111

7.11 System utility for joint schedulers with $[M, R] = [4, 4]$. 112

7.12 System utility for joint subcarrier allocation and packet scheduler. 113

7.13 Subcarrier allocation to MAs as eSM traffic at MA 2 is increased. We plot the results for the following: (a) centralized integer subcarrier allocation 'o', (b) real-valued subcarrier allocation '$\times$', (c) distributed integer subcarrier allocation '+'. 114

7.14 Iterative procedure of distributed subcarrier allocation to MAs when MA 2 has 4% eSM traffic. The eSM traffic at MA 2 is kept constant at 4%. Iteration 0 corresponds to the minimally feasible assignment used to initialize Algorithm 5. 115

A.1 Enhanced two-class FJ system with FCFS. 123

A.2 Example illustrating the operational meaning of variable $\mathcal{R}_s^i$, where i is the class id and s is the processing stage. The decreasing order of class priority is $1 > 2 > 3 > 4 > 5$. 127

List of Tables

3.1 Table of Notations 24
3.2 Decision Variables for Joint Resource Allocation 26
3.3 Details of Simulated Resource Allocation Schemes 34
3.4 Simulation Parameters 34

7.1 List of M2M aggregators serving the different regions. 112
7.2 List of important M2M use-cases and their characteristics. 113

Chapter 1

Introduction

Mobile network operators routinely face the problem of efficiently allocating resources[1] to its customers to satisfy their demands. The Quality-of-Service (QoS) requirements of customers (such as maximum packet delay, packet loss, data rate etc.) are fairly heterogeneous and depend on the set of applications being run at the mobile terminal. For instance, voice traffic needs a guaranteed data rate to ensure low latency but they can tolerate high packet error rate without any discernible impact on the quality of the communication. On the other hand, TCP based services such as email and file transfers work well with a best effort service but require very low packet error rate. Therefore, the cellular traffic is usually characterized into multiple QoS classes based on the minimum data rate, maximum packet delay and maximum packet error rate [1].

Efficient resource allocation to satisfy the QoS needs of users is an important but challenging problem. Significant research has been done on resource allocation for wireless networks. Joint power and bandwidth allocation for rate maximization has been studied using various optimization tools [2], game theory [3, 4], reinforcement learning [5] and auction theory [6]. Another line of work has focused on energy-efficient resource allocation for green wireless networks (See [7] and references therein). However, there still exist many challenges.

The total resources available to a wireless network are limited. The network operator aims to maximize the revenue obtained from a fixed set of resources and thus provisions more resources to users whose QoS requirements can be met with minimum resources. For instance, while allocating resources to meet the user capacity requirements, it assigns higher power and bandwidth to the users closer to the base station due to their low average path-loss and thus higher spectral efficiency. This however is unfair to the cell-edge users as they do not get the service level which they paid for. Therefore, it is important to develop resource allocation scheme that achieve a balance between throughput maximization and fairness of resource allocation. It is equally important to ensure a low implementation complexity of

[1] Wireless resources is an umbrella term comprising of spectral, energy or computational resources available to the network.

such schemes for their successful practical implementation.

Furthermore, it is anticipated that the business use-cases for the fifth generation (5G) mobile networks will be characterized by extremely high reliability, connectivity and network capacity requirements [8]. These 5G requirements would push the performance envelope of current cellular systems even further, making it more difficult to allocate the limited wireless resources. It would therefore require a fundamental restructuring and redesigning of the existing wireless networks. Therefore, new resource allocation schemes need to be developed that are tailored to the unique characteristics and requirements of these new wireless network paradigms such as Machine-to-Machine (M2M) communications [9], small-cells network [10], content delivery networks with edge caching [11]. For instance, M2M networks comprise of very large number of networked sensing devices working together to provide smart services and applications such as smart grid, industrial automation, Digital billboards etc. M2M devices generate sporadic data with small payload size (of typically few hundred bytes) and usually have very low latency requirements.

In this dissertation, we investigate and solve various resource allocation problems in current and future wireless network paradigms with heterogeneous traffic characteristics and QoS requirements. We propose distributed and layered resource allocation architectures to reduce the implementation complexity of the proposed resource allocations schemes. We next outline the specific contributions of this dissertation.

1.1 Contributions

Chapter 2 provides the required background and theory used for solving the resource allocation problems studied in this dissertation. Chapter 3–Chapter 7 describe the major contributions of this dissertation. Conclusions and directions for future work are presented in Chapter 8. The major contributions of this dissertation are briefly described below.

1.1.1 Dynamic spectrum allocation in macrocell network

The exponential increase in mobile data consumption has made the spectrum scarcity a serious concern for mobile network operators. However, the current spectrum allocated to government agencies is highly underutilized [12] and therefore the FCC has mandated it to be shared with cellular systems that need additional spectrum. Cognitive Radio (CR) technology has been popularized so that the primary user spectrum can be leased to the secondary user in a dynamic fashion. The main constraint is that the interference caused to the spectrum owner should remain below a tolerable threshold. Once the cellular network is equipped with sufficient (licensed and leased) spectrum, it is faced with the task of allocating the spectrum to meet the demands of its customers.

From the network operator's perspective, it is rewarding to allocate the spectrum such that the system throughput is maximized as that would maximize their return on spectrum investment. However, this results in an unfair allocation to cell-edge users which typically have low Signal-to-Interference-Noise-Ratio (SINR) due to high average path loss and Inter-Cell Interference (ICI). To reduce the disparity between the average and cell-edge spectral efficiencies, various interference alignment schemes such as LTE Inter-Cell Interference Coordination (ICIC) and LTE-Advanced enhanced ICIC (eICIC) have been proposed. However their usefulness is limited due to their high implementation complexity and additional (base-station) coordination requirements it places on the cellular infrastructure. Fractional Frequency Reuse (FFR) [13] is another technique that can be used to mitigate the ICI to cell-edge users. It can be used in conjunction with interference coordination schemes for maximum cell-edge performance gains. In FFR, the cell center and sectors are assigned a set of orthogonal subcarriers which then get reused in the adjacent cells. This significantly reduces the number of first-tier interferers for cell edge users from 6 to 2.

In this dissertation, we discuss an end-to-end solution for cooperative spectrum sharing and resource allocation in a LTE network. We consider a dynamic FFR-3 scheme for resource allocation [14, 15]. The available (leased and licensed) spectrum is allocated using a layered resource allocation heuristic that first assigns spectrum to the base-stations at cell-center and cell-sectors across the network which in turn assign it to their customers. The LTE network dynamically allocates resources by accounting for the geographical (primary) spectrum restrictions, time-varying capacity demands and channel conditions. We show that the proposed resource allocation schemes work as a perfect tradeoff between greedy rate maximization and fair but low sum-throughput static frequency allocation schemes. More details are presented in Chapter 3.

1.1.2 Energy-efficient power control in small cells

As the global mobile traffic continues to grow exponentially, the wireless industry is preparing itself for staggering 1000 fold capacity increase. More than 50% of this is accounted for by the mobile video traffic with tremendous data-rate requirements [16]. It is not feasible for the current macrocellular network architecture to satisfy such gigantic capacity improvements. Therefore, network densification by extensive deployment of small cells with millimeter Wave access links is being sought after as a promising solution [17]. Small-cells is an umbrella term for low-powered base-stations with small coverage zone ranging from 10 m to several hundred meters. Although the transmit power of an individual small cell can be as low as 25 dBm depending on its coverage area, the overall energy footprint in a dense deployment scenario can be enormous. Due to the low energy budget of each small-cell, renewable energy powered small cells provide an attractive and green alternative to resolve this energy crisis. Feasibility studies conducted on the practicality of various green energy sources for running femtocell operations have provided encouraging results [18].

Another bottleneck to guarantee high data rate service is the shared backhaul capacity among the small-cells. To relieve the capacity-constrained backhaul, caching of popular user data at the small-cell base station (SBS) has been suggested in [11, 19]. An added benefit of caching user data is that it can significantly cut down the file download delay for users [20–22]. Nevertheless, while interesting, these works primarily focus on content placement and update so as to minimize the total delay in downloading the user data. The impact of caching on the energy consumption and backhaul usage for the small cell network has not been explored yet.

In this dissertation, we develop an *online* energy-efficient downlink power control scheme for an energy harvesting SBS that is endowed with caching capabilities and a wireless, capacity-limited backhaul. Energy arrivals are random and modeled as Poisson process [23–25]. The popularity distribution of user data is modeled using Zipf's law and the set of popular user data is cached at the SBS [11, 26]. We consider a novel utility function for the SBS that dynamically adjusts the SBS energy efficiency as a function of the remaining energy in battery. We formulate the power control problem as a (discounted) infinite horizon dynamic programming problem and solve it numerically using the value iteration algorithm. We show that for a given system energy-efficiency requirement, the cache size at SBS can be traded off with the capability of energy harvesting equipment. More details are presented in Chapter 4.

1.1.3 Read latency analysis in Distributed Storage Systems

Cloud based storage systems are emerging to gain significant prominence due to their highly virtualized infrastructure that presents cost-effective and simple to use elastic network resources. The backbone infrastructure of the cloud is comprised of distributed storage systems (DSS), in which the data is stored and accessed from commodity storage disks. Coding of data across distributed disks provides fault tolerance by providing reliability against unexpected disk failures, such as in Google File System [27]. Besides providing fault tolerance and minimizing storage cost, another important aspect which deserves equal, if not more attention is that of *latency performance* of such systems.

The primary focus in literature has been on homogeneous erasure-coded DSS [28–30], wherein the data of each user is stored across n disks (or servers) using a (n, k) optimal Maximum-Distance-Separable (MDS) code. MDS codes are preferred over mere replication-based storage as they provide higher fault tolerance at the same storage cost. The homogeneous DSS model assumed in existing works does not make any distinction among the properties of read requests such as arrival rate, packet size, fault-tolerance requirements etc. However, the data stored and accessed from the cloud can be broadly classified into two categories [31, 32]:

- *Hot-data*: this could refer to data which is frequently accessed (i.e., a higher job request rate). Furthermore, it is desirable to provide higher redundancy/fault tolerance when storing such data.

- *Cold-data*: this could refer to data which is infrequently accessed or archival data. Such data does not necessarily mandate to be coded and stored with higher fault tolerance, as it is seldom accessed by the users.

For instance, the leading cloud storage providers such as Amazon S3, Windows Azure etc. allow their customers to choose from multiple storage classes that differ in redundancy/fault-tolerance, data availability and access latency [33, 34]. They have a low redundancy storage class designed for infrequently/archival data.

Motivated by this, in this dissertation we propose a $(n, k_1, k_2, ..., k_R)$ multi-tenant DSS model for R distinct data classes, a generalization of the homogenous (n, k) DSS in [28]. Data of class i ($\forall i \in \{1, 2, ..., R\}$) is stored across n servers using a (n, k_i) erasure (MDS) code which is directly linked to the storage and fault-tolerance requirements for class i. Using a queuing theoretic approach, we establish upper and lower bounds on the average read latency for each class for different scheduling policies. We show that erasure coding in DSS serves the dual purpose of reducing latency and increasing energy efficiency of DSS. We investigate the impact of number of servers, job arrival distribution on the average latency and energy efficiency of the system. We show that sending redundant requests reduces latency and increasing energy efficiency. More details are presented in Chapter 5.

1.1.4 Delay-efficient packet schedulers for M2M uplink

M2M communications are becoming increasingly ubiquitous due to their vast number of residential and industrial use-cases such as in smart homes, vehicle tracking, industrial automation etc. Since M2M applications provides intelligent services by aggregating and processing the sensory data, the M2M uplink is more demanding than the downlink and thus has garnered more attention from the M2M research community. The M2M uplink traffic from the sensors exhibits heterogeneity in several dimensions such as maximum packet delay, packet size and arrival rate etc. For example, consider the M2M traffic generated by smart meters in a residential area [35]. As per the UCA OpenSG specification (described in [36]), at one extreme are the firmware and software updates with maximum latency of 5 s, payload less than 50 B and arrivals less than 0.5 messages/day/device. On the other extreme are meter reading messages with high payload of around 1.2 kB, maximum packet delay greater than 60 s and 6 messages/day/device. Furthermore with increase in network size, the available computational and (wireless) communication resources gets shared among a large number of sensors. This makes it harder to provide real-time QoS due to simultaneous access attempts from multiple sensors [37]. This necessitates the need for designing delay-efficient M2M uplink packet scheduler that (preferably) are agnostic to the M2M application, communication standard and hardware-software architecture.

Most of the existing M2M packet schedulers are designed for specific wireless technology such as LTE (see [38] and references therein). These schedulers are essentially heuristics that use

variants of Access Grant Time Interval algorithm [39] to allocates resources to the sensors over a periodic time-interval in a fixed or dynamic manner. Another line of work considers the design of packet schedulers for real-time control applications (see [40] and references therein). However, they can not accommodate the multi-dimensional heterogeneity of the M2M uplink traffic. Besides these works, a number of state-of-the-art packet schedulers have been designed for general queuing networks such as fair queuing [41, 42], weighted round robin [43], and deficit round robin [44].

In this dissertation, we consider a general M2M network wherein the uplink traffic from sensors is first aggregated at M2M aggregators (MAs) and then processed at an M2M Application Server (AS) residing in the M2M core. The delay-heterogeneous sensor data is classified into multiple Periodic Update (PU) and Event Driven (ED) traffic classes [45]. We propose a novel delay-efficient multiclass packet scheduling heuristic that exploits the differences in delay-utility of PU and ED traffic to maximize a proportionally-fair system utility metric. The PU packets that are not serviced before deadline are cleared to reduce congestion and thus service delay for backlogged packets. Since successive PU failures may be detrimental for critical M2M applications, we penalize them to minimize their occurrences. Using extensive simulations, we show that the proposed scheduler has a superior delay-performance compared to popular scheduling policies. More details are presented in Chapter 6.

1.1.5 Joint delay-efficient packet scheduling and channel assignment for M2M uplink

We next use queueing theory to analytically characterize the average delay for different M2M traffic categories and then use it to develop delay-optimal M2M packet schedulers. We assume a renewal (Poisson) arrival process [46] for each traffic class as it lends itself easily for delay characterization. The Poisson arrival process has been shown to be a fairly good fit for the aggregated traffic at the MAs [24]. Our goal is to obtain a delay-optimal scheduler that maximizes a proportionally-fair system utility metric. We show that this problem is equivalent to solving for the optimal fraction of time-sharing between different priority scheduling policies such that it maximizes the system utility. We reduce the computational complexity of determining the optimal scheduler (without loss in optimality) by reformulating the original problem as an iterative optimization problem.

We then extend our work to the problem of jointly optimal packet scheduler at the MAs and AS. We initially formulate a single-stage convex optimization problem to be solved centrally at the AS. We then propose a low-complexity, distributed and iterative optimization approach. However, the distributed optimization requires a Poisson arrival process at AS. Lastly, we extend our work to the joint problem of channel assignment for MA-AS links and packet scheduler at MAs and AS. The centralized optimization problem at AS is a mixed integer non-linear programming problem and solved using the Branch and Bound method.

Since this is a NP-hard problem, to reduce its complexity, we propose a low-complexity, distributed optimization framework.

Using Monte-Carlo simulations, we verify the correctness of the analytical result for proposed delay-optimal scheduler and show that it outperforms other state-of-the-art schedulers. It also results in a much lower delay jitter for the delay-sensitive traffic which is highly desirable. But this is achieved at the expense of higher delay jitter for delay-tolerant traffic which is usually not a big concern due to its delay-tolerant nature. We then show that the centralized joint MA-AS packet scheduler also has superior delay-performance as compared to other schedulers and the distributed optimization scheduler results in optimal (near-optimal) delay-performance for exponential (constant) service distribution. More details are presented in Chapter 7.

1.2 List of Publications

The work in this dissertation has resulted in following publications.

- **Journals**

  1. **A. Kumar**, A. Sengupta, R. Tandon and T. Clancy, "Dynamic Resource Allocation for Cooperative Spectrum Sharing in LTE Networks," Accepted for publication in *IEEE Transactions on Vehicular Technology*.
  2. **A. Kumar**, R. Tandon and T. Clancy, "On the Latency and Energy Efficiency of Distributed Storage Systems," Accepted for publication in *IEEE Transactions on Cloud Computing*.
  3. **A. Kumar**, A. Abdelhadi and T. Clancy, "A Delay Optimal Multiclass Packet Scheduler for General M2M Uplink," Submitted to *IEEE Internet of Things Journal*, 2016

- **Conferences**

  1. **A. Kumar**, R. Tandon and T. Clancy "On the Latency of Heterogeneous MDS queue," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 2375-2380, Dec. 2014.
  2. **A. Kumar** and W. Saad, "On the tradeoff between energy harvesting and caching in wireless networks," in *IEEE International Conference on Communication Workshop (ICCW)*, pp. 1976-1981, June 2015.
  3. **A. Kumar**, A. Abdelhadi and T. Clancy, "An Online Delay Efficient Packet Scheduler for M2M Traffic in Industrial Automation", in *IEEE International Systems Conference*, pp. 1-6, April 2016.

4. **A. Kumar**, A. Abdelhadi and T. Clancy, "A Delay-Optimal Packet Scheduler for M2M Uplink," Accepted for publication in *IEEE MILCOM* 2016.
5. **A. Kumar**, A. Abdelhadi and T. Clancy, "A Delay Efficient Multiclass Packet Scheduler for Heterogeneous M2M Uplink", Accepted for publication in *IEEE MILCOM* 2016
6. **A. Kumar**, A. Abdelhadi and T. Clancy, "A Delay-Efficient MAC and Packet Scheduler for Heterogeneous M2M Uplink", Submitted to *IEEE GLOBECOM Workshop on the Internet of Everything* 2017

Chapter 2

Background

In this chapter, we provide the required background and theory needed to solve the resource allocation problems discussed in this dissertation. Almost all of the resource allocation problems can be translated into an optimization problem with desired objective (such as system sum throughput, energy efficiency) subject to certain system constraints and Quality-of-Service (QoS) requirements (such as minimum data-rate requirement, maximum acceptable packet delay) of users.

In some cases, these optimization problems are convex and can be solved easily using the Lagrangian multiplier method [47] or other techniques [48]. Furthermore the optimal solution can be verified using the KKT conditions [49]. However, most of the resource allocation problems require combinatorial optimization to make a decision on which user/application a particular resource element should be assigned. These problems are mixed integer non-linear programs (MINLP) and are often variants of the Knapsack problem which has been proven to be NP-hard [50, 51]. Therefore, in Section 2.1 and Section 2.2, we review Dynamic Programming and the Branch-and-Bound method respectively, that have been used in this dissertation to solve the combinatorial resource allocation problems.

For applications in which packet delay is important for user applications, the resource allocation problem requires the characterization of the average delay or delay jitter either as part of the objective function or as the constraints of the optimization problem. Queuing theory is a powerful analytical tool for delay characterization and a brief overview is provided in Section 2.3.

2.1 Dynamic Programming

The theory for Dynamic programming (DP) was pioneered by Richard Bellman [52]. DP is a powerful mathematical technique used to address sequential decision making for a time-

dynamical system. It involves decomposing the original complex optimization problem into a sequence of multiple simpler subproblems. The DP problems involve a dynamical system that expresses the evolution of some variables, collectively referred to as system's *state*, under the influence of a set of actions made at discrete instances of time. Here we restrict ourselves to a infinite (time) horizon, discrete-time and discrete-state dynamical system as used in Chapter 4.

Let $\mathcal{S}$ and $\mathcal{A}$ denote the state-space and action-space respectively for the dynamic system that evolves as,

$$x_{k+1} = f_k(x_k, a_k, w_k),\ k = 0, 1, \cdots \tag{2.1}$$

where k is discrete-time index, $x_k \in \mathcal{S}$ is system state, $a_k \in \mathcal{A}$ denotes the control or action taken and w_k is a random parameter with known probability distribution. Alternatively, the impact of w_k on the system evolution can be expressed by a probability transition matrix $\mathcal{P}(x_{k+1}/x_k, a_k)$, from the current state x_k to the next state x_{k+1} under action a_k. Let $U_k(x_k, a_k)$ denote the system utility at each time instant k.

DP involves closed-loop optimization wherein the information from the previous stage is used to improve the quality of the decision in the next stage. Therefore in DP, rather than finding the optimal values of the action taken in each stage, the goal is to establish an optimum policy for selecting at each stage k, an action a_k for each possible state x_k in the state-space $\mathcal{S}$. To this end, we define the *value* of each state x_k under policy π as,

$$V_\pi(x_k) = U_k(x_k, a_k) + \delta \sum_{x_{k+1}} \mathcal{P}(x_{k+1}/x_k, a_k) V_\pi(x_{k+1}), \tag{2.2}$$

where $\delta \in [0, 1)$ is a discount factor to account for infinite-time horizon. Then the optimum policy for state x_k, $\pi^*(x_k)$, is the one that maximizes its *value*. Mathematically we have,

$$\pi^*(x_k) = \arg\max_{a_k}\ U_k(x_k, a_k) + \delta \sum_{x_{k+1}} \mathcal{P}(x_{k+1}/x_k, a_k) V_{\pi^*}(x_{k+1}). \tag{2.3}$$

Although there exists several algorithms to find the optimal policy π^*, value iteration and policy iteration are the most popular techniques. Value iteration [53] is an iterative procedure for determining the optimal expected *value* of each state using the *values* for the neighboring states. The optimal *values* can then be substitute in Eq. 2.3 to obtain the optimal policy. The algorithm stops when the change in *value*of each state for two consecutive iterations is below a prespecified threshold ϵ, i.e.,

$$\arg\max_{x_k}\ |V^{n+1}(x_k) - V^n(x_k)| < \epsilon, \tag{2.4}$$

where $V^n(x_k)$ denotes the *value* of state x_k at the end of n^{th} iteration. The downside of value iteration is its slow convergence. It has been observed that the policy becomes optimal long before the expected *values* of states have converged [54].

This motivates us to explore another technique to determine the optimal policy, namely the policy iteration [55]. The initial policy is usually set as the one that maximizes the current utility for each state. Then in each iteration, two steps are performed - value determination and policy improvement. Value determination entails calculating the expected value for each state under the given policy using Eq. (2.2). The update values are then used to determine if any policy improvement is possible. The procedure terminates when policy stabilizes. It has been shown that policy iterations converges fairly quickly compared to value iteration [56]. However, the downside is that the search space for the optimal policy in the policy determination step can be extremely huge.

2.2 Branch-and-Bound for MINLP problems

The Branch-and-Bound (BnB) is a non-heuristic enumeration-based method for finding global optimum of (NP-hard) MINLP problems. It maintains provable lower and upper bounds on the global objective value and terminate with ϵ-suboptimality. The basic idea behind BnB algorithm is to develop tight upper and lower bounds on the optimal solution by repeatedly partitioning the feasible region to improve the bounds until the gap between the bounds is less than certain tolerance ϵ [57].

Without loss of generality, we next consider a minimization MINLP problem to demonstrate the BnB process[1]. The goal is to find global minima of some function $f(\bar{x}, \bar{y})$, where $\bar{x}$ is a set of continuous variables and $\bar{y}$ is a set of integer variables. The minimization is subject to certain constraints on $\bar{x}$ and $\bar{y}$, which generates the feasible region $\mathcal{S}_0$. Define $\Phi_{\min}(\mathcal{S})$ to be the optimal objective over the region $\mathcal{S}$. Therefore the global optimal value $f^* = \Phi_{\min}(\mathcal{S}_0)$. Let $\Phi_{\text{lb}}(\mathcal{S})$ and $\Phi_{\text{ub}}(\mathcal{S})$ denote the lower and upper bounds on $\Phi_{\min}(\mathcal{S})$. The lower bound can be easily determined by solving the convex relaxation of the original problem by allowing the integer $\bar{y}$ to take on continuous values. By rounding off the continuous $\bar{y}$ to closest integer, we get the upper bound. We next detail the steps of BnB procedure as given in [57].

1. Determine lower and upper bounds on f^*.
   - Set $L_1 = \Phi_{\text{lb}}(\mathcal{S}_0)$ and $U_1 = \Phi_{\text{ub}}(\mathcal{S}_0)$.
   - Terminate if $U_1 - L_1 \leq \epsilon$.
2. Partition $\mathcal{S}_0$ into two regions: $\mathcal{S}_0 = \mathcal{S}_1 \cup \mathcal{S}_2$.
3. Compute $\Phi_{\text{lb}}(\mathcal{S}_i)$ and $\Phi_{\text{ub}}(\mathcal{S}_i)$, $i = 1, 2$.
4. Update lower and upper bounds on f^*.

[1]Similar process can be applied for maximization problem with only difference is that the role of lower and upper bounds gets reversed.

- Update lower bound: $L_2 = \min\{\Phi_{\text{lb}}(\mathcal{S}_1), \Phi_{\text{lb}}(\mathcal{S}_2)\}$
- Update upper bound: $U_2 = \min\{\Phi_{\text{ub}}(\mathcal{S}_1), \Phi_{\text{ub}}(\mathcal{S}_2)\}$
- Terminate if $U_2 - L_2 \leq \epsilon$.

5. Refine partition, by splitting $\mathcal{S}_1$ or $\mathcal{S}_2$, and repeat steps 3 and 4.

At each refinement step, we need to select - (a) which partition to split, (b) which variable to split, (c) where to split (what value of variable). A good heuristic is to split the partition with smallest lower bound, at the midpoint of the longest dimension or variable. The downside of BnB algorithm is that it is slow and has exponential worst case performance. To improve the performance, we can eliminate or *prune* the partitions where the lower bound is greater than the current global upper bound.

2.3 Delay characterization using Queuing theory

Queuing of transmitted packets[2] occurs commonly in wireless networks due to the finite (packet) processing capacity of various nodes in the network. To characterize a queuing system, we have to identify the probabilistic properties of the arrival process, service times and service disciplines.

The arrival process is usually characterized by the packet inter-arrival times which is assumed to be independent and identically distributed (i.i.d.) random variables. Similarly the service process is characterized by the distribution of service time and is again assumed to be i.i.d. for each packet. In general, there may be multiple servers that simultaneously process the queued jobs. The capacity of queuing system is defined as the maximum number of customers that can be allowed in the system including the job being served. There are several service disciplines to process the jobs in queue such as First-Come-First-Serve (FCFS), Last-In-First-Out, Random Service etc. If all the jobs arriving in system are identical with equal service importance, then FCFS is the most appropriate policy as it is fair to all the jobs. A queuing system can be succinctly described using the Kendall's notation as $A/B/m/K/n/D$, where

- A and B denote the inter-arrival and service time distribution respectively.
- m denotes the number of parallel servers.
- K and n denote the system capacity and population size respectively.
- D denotes the service discipline.

[2]In this dissertation, we use the terms - packets, messages and job requests interchangeably in the context of queuing systems.

Exponential (M) and deterministic (D) distributions are commonly used for inter-arrival and service time distribution. All other distributions are collectively notated by the general distribution G. For our work, the population size n and capacity K is assumed to be infinite and the service discipline is FCFS. Therefore, they are omitted from the notation.

Now let A be a random variable (r.v.) denoting the inter-arrival time and S be a r.v. for the service time. Let $\lambda = 1/\mathbb{E}[A]$ be the job arrival rate and $\mu = 1/\mathbb{E}[S]$ denote the service rate. Then, for the queuing system to be stable (i.e., bounded queue length), the server utilization $\rho = \lambda/\mu$ should be less than unity.

Queuing systems are typically analyzed by modeling them as a Markov chain where the states represent the number of jobs in the system. We are typically interested in the steady-state analysis, i.e., when the system has reached equilibrium and the probabilities of system being in a certain state becomes constant. The quantities of interest include waiting time distribution, average waiting time or average number of jobs in the system.

Little's law [58] is popularly used to relate the average waiting time in queue W and average number of jobs in queue N_q at steady state and is simply given by $N_q = \lambda W$. Similarly, the average time spent in system W and number of jobs in system N are related as $N = \lambda T$. Also, it is straightforward to see that $T = W + \frac{1}{\mu}$.

M/M/1 queue: It is the simplest non-trivial queue. The response time T is exponentially distributed with mean $1/(\mu - \lambda)$ [59]. Then from Little's law, we have $N = \rho/(1-\rho)$.

G/M/1 queue: Due to the non-Poisson arrival process, the PASTA theorem is not applicable to the system at arbitrary time instants. Therefore, the system is analyzed using the Imbedded Markov Chain [60] constructed at the arrival instant of jobs. The average waiting time W_q is given by

$$W = \frac{\sigma_A}{\mu(1-\sigma_A)}, \tag{2.5}$$

where $\sigma_A^2 = \mathbb{E}[A^2] - (\mathbb{E}[A])^2$ is the variance of inter-arrival time. The waiting time distribution is given by

$$f_W(t) = (1-\sigma_A)\delta(t) + \mu\sigma_A(1-\sigma_A)\mathrm{e}^{-\mu(1-\sigma_A)t}, t \geq 0. \tag{2.6}$$

M/G/1 queue: The difficulty in M/G/1 analysis stems from the non-memoryless service distribution and the state description would depend on both the number of jobs in system and amount of service provided to the job in server. The M/G/1 queue can be analyzed using residual life arguments or Imbedded Markov chain approach. The average waiting time W_q is given by the P-K formula [61, 62] as

$$W = \frac{\lambda\mathbb{E}[S^2]}{2(1-\rho)}. \tag{2.7}$$

In general, it is difficult to find the delay distribution. However, the Laplace Transform

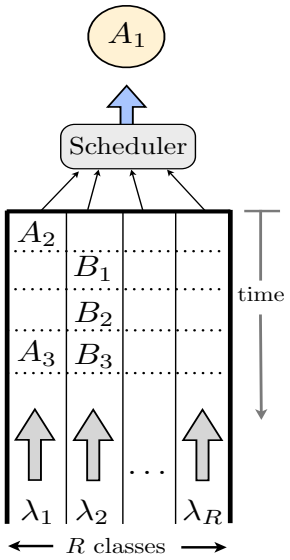


Figure 2.1: A heterogeneous queueing system with R traffic classes.

(L.T.) of the waiting time can be determined as

$$\mathcal{L}_Q(s) = \frac{s(1-\rho)}{s - \lambda + \lambda \mathcal{L}_S(s)}, \tag{2.8}$$

where $\mathcal{L}_S(s)$ is the L.T. of the service distribution.

G/G/1 queue: An exact analysis for G/G/1 queue is not possible but there exist useful bounds on the average waiting time in queue [63],

$$\frac{\lambda \sigma_S^2 - \mathbb{E}[S](2-\rho)}{2(1-\rho)} \leq W \leq \frac{\lambda(\sigma_S^2 + \sigma_A^2)}{2(1-\rho)}, \tag{2.9}$$

where $\sigma_S^2 = \mathbb{E}[S^2] - (\mathbb{E}[S])^2$ is the variance of service time.

We next review the delay characterization for M/G/1 system with heterogeneous traffic that is classified into R traffic classes as shown in Fig. 2.1. The arrivals for class i is Poisson with rate λ_i. Let S_i be a r.v. representing the service time for a job of class i and follows a general distribution G. The service times both within and across the classes are independent of each other. We assume FCFS for job arrivals within each class, whereas for scheduling jobs of different classes we consider three popular policies - FCFS, preemptive priority and non-preemptive priority system.

2.3.1 FCFS scheduling

The fraction of jobs of class i is $p_i = \lambda_i / \sum_{r=1}^{R} \lambda_r$. Now since the service time for each class i follows general distribution, the overall service time across all classes S also follows a

general distribution. The probability that S takes on the value of S_i is p_i $\forall i$. Therefore the probability distribution function (pdf) of S is $f_S(s) = \sum_{r=1}^{R} p_r f_{S_r}(s)$.

Then the first and second moment of S are simply

$$\mathbb{E}[S] = \sum_{r=1}^{R} p_r \mathbb{E}[S_r], \;\; \mathbb{E}[S^2] = \sum_{r=1}^{R} p_r \mathbb{E}[S_r^2]. \tag{2.10}$$

Now the average waiting time for any job is given by P-K formula (in Eq. 2.7) as

$$W_{\text{FCFS}} = \frac{\sum_{r=1}^{R} \lambda_r \mathbb{E}[S_r^2]}{2\left(1 - \sum_{r=1}^{R} \lambda_r \mathbb{E}[S_r]\right)} \tag{2.11}$$

Therefore, the average response time for jobs of class i is sum of their average waiting time and average service time as $T^i_{\text{FCFS}} = \mathbb{E}[S_i] + W_{\text{FCFS}}$.

2.3.2 Non-Preemptive resume priority scheduling

For priority queueing systems, without loss of generality, we assume the following priority order: $1 > 2 \cdots > R$, where $a > b$ denotes that class a has higher priority than class b. Again using the residual time analysis for each M/G/1 class, the average waiting time in queue for class i can be shown to be [59, Chap. 3],

$$W_i = \frac{X}{\left(1 - \sum_{r=1}^{i-1} \rho_r\right)\left(1 - \sum_{r=1}^{i} \rho_r\right)}, \tag{2.12}$$

where X is the mean residual service time. Due to the non-preemptive service paradigm, the residual service time for class i is due to all the classes in the system. Therefore we have,

$$X = \frac{1}{2} \sum_{r=1}^{R} \lambda_r \mathbb{E}[S_r^2]. \tag{2.13}$$

Then the average response time for jobs of class i is simply,

$$T^i_{\text{NPQ}} = \mathbb{E}[S_i] + \frac{X}{\left(1 - \sum_{r=1}^{i-1} \rho_r\right)\left(1 - \sum_{r=1}^{i} \rho_r\right)}. \tag{2.14}$$

2.3.3 Preemptive priority scheduling

We again assume the priority order: $1 > 2 \cdots > R$. The average response time for class i is shown to be [59, Chap. 3],

$$T^i_{\mathrm{PQ}} = \frac{X_i + \left(1 - \sum_{r=1}^{i} \rho_r\right) / \mu_i}{\left(1 - \sum_{r=1}^{i-1} \rho_r\right)\left(1 - \sum_{r=1}^{i} \rho_r\right)}, \tag{2.15}$$

where X_i is the mean residual service time for jobs of class i. Due to the preemptive service paradigm, the residual service time for class i is only due to the higher priority classes and class i itself. Therefore, we have,

$$X_i = \frac{1}{2} \sum_{r=1}^{i} \lambda_r \mathbb{E}[S_r^2]. \tag{2.16}$$

Chapter 3

Dynamic Spectrum Allocation in a Macrocell Network

The FCC spectrum policy task force has reported that a significant amount of the current licensed spectrum is sporadically utilized [12] and the fixed spectrum assignment policy should be modified in order to support the ever growing number of wireless devices. The key enabler in breaking the spectrum gridlock is Cognitive Radio (CR) technology, which provides the ability to share the wireless channel with Secondary Users' (SU) in an opportunistic manner also popularly known as Dynamic Spectrum Access (DSA) [64]. The main constraint for the secondary devices (or cognitive radios) is to guarantee that the interference caused to the Primary Users' (PU) needs to be below a tolerable threshold [65].

In this chapter, we consider a spectrum sharing architecture with an arbitrary primary network and a LTE based secondary cellular network. We employ a hybrid DSA approach in which the information about availability of (primary) spectrum (obtained through spectrum sensing) at LTE network is fused with an interference map based Geo-location DataBase (GDB) present at the (primary) Spectrum Accountability Server (SAS) [66]. The SAS then leases a certain chunk of primary spectrum to the LTE network for a specified lease duration. The LTE network controller is then faced with the decision of allocating the available (licensed and leased) spectrum to the customers.

Most of the works on resource allocation in cellular systems either aim to maximize system throughput or provide fair resource allocation to users. Resource allocation schemes that greedily maximize the system sum throughput in LTE network have been proposed in [13, 67–70]. However, this results in lower QoS to the cell edge users due to increased path-loss and Inter-Cell Interference (ICI). ICI coordination was proposed in [71] for LTE downlink. The ICI to cell-edge users can be further mitigated by adopting Fractional Frequency Reuse (FFR) schemes [13] instead of frequency reuse factor of 1 that is being used in LTE. The FFR scheme partitions a cell into cell center and cell sectors near cell-edge. The cell center and sectors are assigned a set of orthogonal subcarriers which then get reused in the adjacent

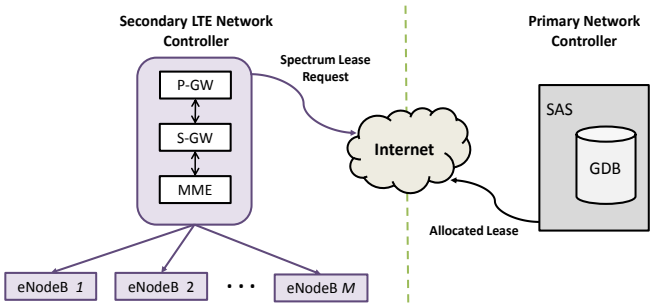


Figure 3.1: Spectrum Sharing Framework.

cells. This significantly reduces the number of first-tier interferers for cell edge users from 6 to 2. Fair resource allocation schemes have been studied in [72, 73] and specifically through the lens of cooperative game theory in [74–76]. However, for practical resource allocation schemes, it is important to achieve a balance between the two extremes: maximizing network throughput and fair resource allocation to users.

In this chapter, we discuss an end-to-end solution for hybrid Dynamic Spectrum Access (DSA) based spectrum sharing and resource allocation in a LTE network. We consider a dynamic Fractional Frequency Reuse 3 (FFR-3) scheme for resource allocation [14, 15]. The available (leased and licensed) spectrum is allocated using a layered resource allocation heuristic that first assigns spectrum to the cell centers and sectors across the network and then to the users within each region. The LTE network dynamically allocates resources (i.e., available legacy and leased spectrum bands) by taking into account the spectrum regulatory restrictions, current traffic demands and channel conditions and minimizing long term ICI. We show that the proposed resource allocation schemes work as a perfect tradeoff between greedy rate maximization and fair but low sum-throughput static frequency allocation schemes.

The rest of this chapter is organized as follows. Section 3.1 discusses the overall spectrum sharing architecture. Section 3.2 describes the resource allocation problem formulation and the layered resource allocation heuristic/algorithm. The simulation results are discussed in Section 3.3. Section 3.4 concludes the chapter.

3.1 Cooperative Spectrum Sharing Architecture

The cooperative spectrum sharing scheme comprises of two main mechanisms: the spectrum lease request and dynamic resource allocation. The lease request is sent from a secondary network to a primary spectrum server, which in turn evaluates the request and sends back a response granting full, partial or no access to the requested spectrum. We follow a spectrum sharing architecture utilizing the existing LTE/LTE-A framework [77], which is shown in Figure 3.1. A variation of this architecture had been considered in [78] where the secondary

small cell radio access networks perform spectrum sensing and aggregate that data in a spectrum database which then uses that information for many tasks including channel allocation. Our candidate spectrum sharing architecture comprises of two key elements:

- *Primary* Spectrum Accountability Server (SAS): responsible for monitoring the current activity of primary users operating in their assigned primary bandwidth. Upon receiving a spectrum lease request from the Packet Data Network Gateway (P-GW) of a LTE network, the SAS determines available PU bands by querying a GDB that assists the SAS in identification of available primary bands at a particular location and time.
- *Secondary* Spectrum Manager[1] (SSM): controls a set of eNBs. It receives real time information regarding the spectrum demand and local sensing decisions (about availability of PU bands) from each of the eNBs. It then prepares a spectrum lease request and sends it to the SAS. Upon receiving the spectrum access authorization from the SAS, it *efficiently* allocates the available spectrum among the eNBs.

We classify the spectrum lease request as either "structured" or "unstructured" [79] depending upon the lease format required by the SAS. In a structured request, in addition to specifying the spectrum requirement, the MME also reports a desired subset of available PU bands that if allocated would maximize its utility. In an unstructured request, the MME requests for a particular amount of spectrum without specifying a set of desired PU bands. Abhay in [80] discussed a lease structure similar to our "unstructured" least request. Both these requests also have corresponding time and geographic specifications tagged to them. For the scope of this work, we assume that a structured lease request is sent by the MME to the SAS.

3.1.1 Spectrum Lease Request

In this section, a procedure for obtaining the spectrum lease from the SAS is discussed. Joslyn et. al. in [81] proposed a "Protocol to Access White Space Database (PAWS)" in the context of TV band White Spaces. A similar message passing protocol is adopted in this work. The secondary network senses the primary spectrum and aggregates the spectrum sensing inputs from the eNBs at the MME. It then formulates a spectrum demand, determines the required lease duration and finally prepares a spectrum lease request with the consolidated information.

[1]Depending on the future wireless standards, the SSM might exist as a separate functional block or it may be combined with the Mobile Management Entity (MME) present in the LTE network. However for this work, we assume the functionality of SSM resides within the MME. Henceforth, we use MME instead of the acronym SSM.

Local Multi-band Sensing and Decision Aggregation at MME

We assume that each eNB is equipped with sensors to locally sense the presence of PUs in the PU bands. We also assume that there is no collaboration from adjoining eNBs in making sensing decisions at each eNB i.e., in this work, we adopt a centralized decision fusion model as opposed to a distributed decision feedback. The motivation behind this approach is to leverage the inherent spatial and spectral diversity. Due to varying channel conditions at the eNBs, it might happen that the adjoining eNBs in the network sense a particular PU channel busy; but the eNB of interest might find the PU channel available. Thus it helps to avoid missing potential transmission opportunities in PU bands. As described later, the sensing decisions of the eNBs are ultimately combined at the MME. Further, the SAS is the decision making authority for the purpose of leasing the PU bands and hence we assume that an appreciable performance degradation is avoided even after adopting a centralized decision fusion model. The added benefit of centralized model is clearly the reduced complexity/overhead in making sensing decisions and the resulting decrease in feedback. The sensing period is typically interleaved with the transmission interval (for uplink/downlink user data/control traffic). The sensing time for each band in principle would be different because of the unequal bandwidths of the different PU bands. Depending upon the kind of signal being transmitted by an active primary user, the optimal detection strategy would in general differ. For instance, the PU signal could be either wideband/narrowband signal, frequency swept signal, frequency hopping signal etc. For each of these, the eNB may choose to employ a different detection strategy such as energy detection, matched filter, cyclostationary feature detection etc. [82]. For the scope of this work, we do not detail these signal dependent detection strategies.

After completion of local sensing, there are two extremes for information sharing with the MME: soft-decision vs. hard-decision combining [83]. While soft decisions increase the quality of decision making at the MME, communication overhead between the eNBs and the MME becomes prohibitive. A more favorable choice is to communicate only the final 1-bit hard decisions to the MME. At the MME, an optimum fusion rule based on minimizing Bayes risk [84] can be adopted. Basically, the MME makes its decision for each PU band, by comparing a weighted sum of the individual hard decisions to a threshold, assigning a larger weight to more reliable measurements.

The spectrum lease issued by the SAS would typically have a cost (per unit time per unit bandwidth) associated with them. So it is important for the MME to estimate the average network load and average time duration for which additional spectrum is required. The average total spectrum needed to support the active UEs can be estimated based on the capacity requirements and channel conditions of the active UEs in each cell averaged over time. The MME then estimates the additional spectrum (difference of the total bandwidth needed and the available legacy spectrum) that needs to be leased, W_{lease}, to support the active User Equipment's (UEs). Using wideband sensing, the MME can also evaluate which of the available PU bands would best serve the purpose of fulfilling additional capacity requirement.

The MME also estimates the duration, T_{lease}, for which the additional bandwidth will be required by utilizing higher layer information about the usage characteristics of the network in terms of apps/voice data being used within a given cell. This information is aggregated at MME and a T_{lease} is calculated. When there is a cost associated with the leasing of PU bands, this process would ensure that the MME makes the best possible lease request given current average demands and channel conditions.

Constructing Lease Request

The MME prepares a set $\mathcal{R}$ of most *useful* available PU bands to be included in the spectrum lease request. The total expected bandwidth of $\mathcal{R}$ should be as close as possible to the additional bandwidth demand W_{lease}. Finally the MME prepares a "structured" spectrum lease request [79] triplet $\{\mathcal{R}, W_{lease}, T_{lease}\}$ which is transmitted to the SAS. We believe the SAS might enable this option, so that MME can provide a list of preferred PU bands to the SAS in addition to the amount of spectrum needed and the time duration for the requested spectrum. The MME prefers some PU bands over the other available PU bands because the achievable capacity per unit bandwidth in these two cases may be significantly different due to frequency selective fading or other channel variations. In case of an "unstructured" lease request, the MME does not send the information about the available (primary) channels of interest to the SAS. In the next section, the proposed resource allocation scheme for the secondary network is discussed.

3.2 Resource Allocation at the LTE Network

The SAS replies to the spectrum lease request triplet $\{\mathcal{R}, W_{lease}, T_{lease}\}$ with its lease decision triplet $\{\mathbb{R}, \mathbb{L}, \mathbb{P}_{\mathbb{L}}\}$. The contents of the lease decision are as follows:

- $\mathbb{R} \subseteq \mathcal{R}$ is the set of PU bands in which the SU is allowed to transmit. When all PU bands are available $\mathbb{R} = \mathcal{R}$.
- $\mathbb{L}$ is the set of location indices with which eNBs are allowed to transmit.
- $\mathbb{P}_{\mathbb{L}}$ is the set of power constraints with which eNBs can transmit at the given locations.

Assuming that set $\mathcal{A}$ denotes a GDB, which contains location indices for all local geographical areas under the control of the secondary network, we have $\mathbb{L} \in \mathcal{A}$.

In general this decision triplet can be of two types:

- *Unrestricted Access:* The MME can be granted unrestricted access to the requested set of bands $\mathbb{R}$. In this case the secondary MME is given access to transmit with its

desired power over the allocated sub-bands for the entire leasing period T_{lease}. Thus $\mathbb{P}_{\mathbb{L}}$ does not contain any constraints and the set $\mathbb{L}$, in this case, is equal to the local GDB set $\mathcal{A}$.

- *Restricted Access:* Alternately, the secondary MME can be granted restricted access to the sub-bands $\mathbb{R}$. The set $\mathbb{L}^c \in \{\mathcal{A} - \mathbb{L}\}$, then denotes the set of locations in which the eNBs are not allowed to transmit i.e., locations at which a primary user may be present and $\mathbb{P}_{\mathbb{L}}$ are the powers with which the eNBs located in $\mathbb{L}$ can transmit. Additionally, the MME is expected to periodically sense the allotted bands for the presence of a primary and transmit accordingly, so as not to cause any interference to the PU.

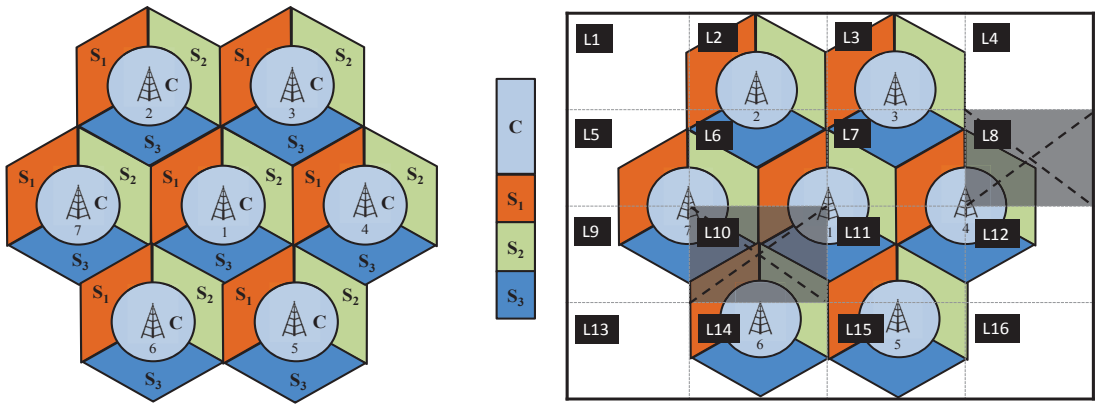
When the MME receives the decision triplet $\{\mathbb{R}, \mathbb{L}, \mathbb{P}_{\mathbb{L}}\}$, it maps the location indices from the SAS to the location of its eNBs from the GDB. Then the power constraints for each index are mapped to the eNBs occupying the given location index. In a LTE network, the function of the legacy Radio Network Controller (RNC) is divided between the eNB and the MME [85]. Our proposed DRA scheme uses control plane (CP) signaling from MME to eNB and UEs to *efficiently* allocate resources using *minimal* feedback from eNBs to the MME. In the following sub-sections, we discuss in detail the system model and the channel and propagation models used for our work.

3.2.1 System Model

A LTE network is considered as the SU network. In the LTE physical layer, the available frequency band is divided into sub-carriers (channels), each of bandwidth $\Delta f = 15$ kHz [85]. 12 contiguous sub-carriers are grouped together to form the Physical Resource Blocks (PRBs) which have a bandwidth of 180 kHz. PRBs are the smallest resource units that can be allocated to an UE and each has a duration of 0.5 ms. Two PRBs are grouped to form a schedule block (SB) which covers a duration of 1ms called a transmit time interval (TTI).

We consider a hexagonal M-cell wrap-around universe[2] with L sectors per cell. In order to reduce ICI, we consider a dynamic FFR-3 scheme [14, 86] for allocating the S PRBs within each of the M cells. This is illustrated in Figure 3.2a for a $M = 7$ cell cluster. Using this scheme, different Frequency Reuse Factors (FRF) can be used at the cell-center and at the cell-edge while still achieving an overall reuse factor of 1 within the network. In the FFR-3 scheme, each cell is divided into 2 main zones : the cell-edge zone and the cell-center zone. The cell-edge zone is further subdivided into $L(= 3)$ sectors. In Figure 3.2a, C corresponds to the cell center zone while the set $\{S_1, S_2, S_3\}$ corresponds to the sectors of the cell-edge zone. The idea here is that at any vertex of a given cell, the 6 sectors sharing the vertex also share 3 different bands such that no two adjacent sectors have the same spectrum assignment. This resource allocation ensures that a given cell uses all the available frequency bands for

[2]In a wrap-around universe, edge-of-the-world effects are accounted for by considering each cell as a center of a M cell cluster and thus each cell faces uniform inter-cell interference.



(a) (b)

Figure 3.2: (a) The Fractional Frequency Reuse - 3 scheme in a 7 cell cluster; (b) Mapping of location indices of restricted transmission to eNB sectors.

transmission. Thus in the FFR-3 scheme, the spectral efficiency and frequency diversity are increased while minimizing ICI.

Given a cell radius of R, it has been shown in [14, 15] that a radius of $\approx 0.6R$ for the cell center zone is optimum with respect to the network throughput. In LTE, Evolved Packet Core (EPC) provides LoCation Services (LCS) [85] which is capable of estimating the position of UEs within the cell using methods like Assisted GPS, Observed Time Difference of Arrival, Enhanced Cell ID etc. Based on the location estimate, the MME classifies the UEs as cell-center or sector users. Table 3.1 lists the notations used in the following sections for modeling the resource allocation problem.

Resource Allocation Framework

After receiving the spectrum allocation from the SAS, the MME allocates received spectrum among the M $(= 7)$ eNBs associated with it. Suppose, within the available frequency band, there are a total of K sub-carriers placed Δf apart and these are grouped to form N PRBs (where $N = K/12$). It is assumed that the MME can map the location indices of restricted transmission provided by the SAS to a sector of an eNB. This is illustrated in Figure 3.2b with a simple example. Suppose the set $\mathbb{L} = \{L1, L2, .., L7, L9, L11, .., L16\}$ represents the permissible transmission zones. In this case the restricted set $\mathbb{L}^c = \{L8, L10\}$. This is marked by the cross-signs within the grid in Figure 3.2b. The MME maps this to the sectors of the cells that fall within this restricted region. In this case, Sector 1 of eNB_1 and eNB_6, Sector 2 of eNB_4, eNB_6 and eNB_7 and Sector 3 of eNB_1 and eNB_7 are the restricted sectors. Also included in the restricted set are the cell-center zones of eNB_1, eNB_4, eNB_6 and eNB_7. Here, we assume a hard decision is made on inclusion of a zone in the restricted set i.e., even if a part of a cell-edge/cell-center zone is within the location index of an SAS restriction, the

j	UE Index
k	Sub-carrier index
m	eNB/cell Index
n	PRB Index
c	Cell-center
l	Cell sector Index $\in \{1, 2, ..., L\}$
J	Total Number of UEs
K	Total Number of sub-carriers
M	Total number of eNBs/cells
N	Total Number of PRBs
L	Total Number of sectors
Δf	sub-carrier spacing
Υ	Set of users
C_j	Minimum Capacity requirement per user
$\mathcal{C}^c$	Set of PRBs assigned to cell-center zone
$\mathcal{C}^l$	Set of PRBs assigned to l-th sector of the cell-edge zone
$\mathcal{I}^c$	Set of interferers for cell-center users
$\mathcal{I}^l$	Set of interferers for cell-edge users of l-th sector

Table 3.1: Table of Notations

entire zone is placed in the restricted set.

The set of S PRBs is subdivided into $(L+1)$ sets. Cell-center PRBs $\in \mathcal{C}^c$ and cell-edge sector PRBs $\in \mathcal{C}^l$, where $l = [L] \triangleq 1, \cdots, L$ is the sector index. The intra-cell interference is zero i.e., $\mathcal{C}^l \bigcap \mathcal{C}^c = \{\}$ and $\bigcap_{l=1}^{L} \mathcal{C}^l = \{\}$, as each set uses orthogonal frequencies. These assignments are replicated over all the cells as per the FFR-3 scheme. Let Υ_m be the set of all UEs in the m-th cell and $J_m = |\Upsilon_m|$ be the number of UEs in the m-th cell. The total number of UEs in the network $J = \sum_{m=1}^{M} J_m$ and the set of all UEs is $\bigcup_{m=1}^{M} \Upsilon_m$. For the FFR-3 scheme, let Υ_m^c be the set of cell-center UEs in the m-th cell and $J_m^c = |\Upsilon_m^c|$. Also, let Υ_m^l be the set of UEs in the l-th sector of the cell-edge zone of the m-th cell and $J_m^l = |\Upsilon_m^l|$. Thus the total number of UEs within the cell-center zones of all cells in the network is given by $J^c = \sum_{m=1}^{M} J_m^c$. The number of UEs in the l-th sector of the cell-edge zones of all cells in the network is given by $J^l = \sum_{m=1}^{M} J_m^l$.

Now, let $\mathcal{I}_m^c$ denote the set of interferers for a UE in the cell-center zone and $\mathcal{I}_m^l$ be the set of interferers for the l-th sector of the $m-$th cell-edge zone. For a 7 cell network shown in Figure 3.2a, an UE in the cell-center zone has interference from all the 6 adjacent cells as all the cell-centers use the same PRB set. An UE in the l-th sector of the cell-edge zone has two interferers. For example in Figure 3.2a, the UEs in sector 2 in cell 1 faces interference from the sectors 2 of cells 6 and 7. This is illustrated by the shaded sectors. Thus for eNB_1, $\mathcal{I}_1^c = \{\text{eNB}_m\}$, $m = 1, .., 7$, for an UE in the cell-center zone, while $\mathcal{I}_1^1 = \{\text{eNB}_m\}$, $m = 6, 7$

for UE in sector 2 of the cell-edge zone.

3.2.2 Propagation and Rate Models

The channel gain for UE j on PRB n, sub-carrier k (within the PRB, $k = 1, ..., 12$) from the serving eNB m is given by:

$$G_{j,m,n,k} = 10^{-\frac{\Gamma_{j,m,n,k}(r)}{10}} \times S_{j,m} \times \zeta_{j,n,k} \tag{3.1}$$

where $\Gamma_{j,m,n,k}(r)$ is the pathloss (in dB) at a distance r, $S_{j,m}$ is the shadowing coefficient and $\zeta_{j,n,k}$ is the fading coefficient. The pathloss is modeled using the COST-Hata Model [87].

This model is valid for frequency bands $1500 - 2000$ MHz. However, in [88], it was shown that the model also gives valid results for the 3.5 GHz range which we consider in this work. In case of frequencies $150 - 1500$ MHz, the Okamura-Hata model [89] should be used. The shadowing coefficient $S_{j,m}$ is modeled as a correlated log-normal distribution while the fading coefficient $\zeta_{j,n,k}$ is assumed to be Rayleigh distributed. The corresponding Signal-to-Interference-plus-Noise Ratio (SINR) for the k-th sub-carrier within the n-th PRB is then given by:

$$\gamma_{j,n,k} = \frac{G_{j,m,n,k} P_m}{N_0 \Delta f + \sum_{(q \neq m,\ q \in \mathcal{I})} G_{j,q,n,k} P_q} \tag{3.2}$$

where N_0 is the noise power spectral density and P_i is the power per sub-carrier for the m-th serving eNB. In the case of power restrictions placed by the SAS, the maximum permissible transmit power of the m-th eNB, P_m^{max}, is mapped by the MME from the power constraint set $\mathbb{P}_{\mathbb{L}}$. At the network level, we assume that the power on each downlink sub-carrier is equal i.e., Uniform Power Distribution (UPD). Thus w.r.t equation (3.2), we have $P_m = P_m^{max}/K$. The interferer set for UE $j \in \bigcup_{m=1}^{M} \Upsilon_m^c$ is $\mathcal{I}^c$ and for an UE $j \in \bigcup_{m=1}^{M} \Upsilon_m^l$ is $\mathcal{I}_m^l$. Using Exponential Effective SINR Mapping (EESM) or a similar technique [90, 91], the sub-carrier SINRs of the n-th PRB can be mapped to an effective SINR for the entire PRB:

$$\gamma_{j,n} = -\ln\left(\frac{1}{K}\sum_{k=1}^{K} e^{-\gamma_{j,n,k}}\right) \tag{3.3}$$

Employing continuous rate adaptation [13], the effective SINR of the j-th UE for the n-th PRB at any time instant t can be mapped to an instantaneous achievable data rate as follows:

$$R_{j,n,t} = K \Delta f \log_2(1 + \lambda \gamma_{j,n,t}) \tag{3.4}$$

where, λ is a constant related to the target Bit Error Rate (BER) as $\lambda = (-1.5)/(\ln(5\ \text{BER}))$ and $\gamma_{j,n,t}$ is the SINR of the UE to eNB at the instant t. The instantaneous achievable rate of UEs belonging to the cell-center zone of all cells is given by $R_{j,n,t}^c$ and for UEs in the $l-$th cell-sector zone is given by $R_{j,n,t}^l$, for the j-th UE and the n-th PRB.

$x^i_{j,m,n} \in \{0,1\}$	Variable modeling PRB assignment to UEs;
$\eta^i_{m,n} \in \{0,1\}$	Variable modeling MME's decision to transmit in cell-center/sector zone;
where	$i \in \{c,l\}$ and $l = 1,2,3$

Table 3.2: Decision Variables for Joint Resource Allocation

3.2.3 Joint DRA Problem Formulation

In this section, a FFR based dynamic PRB allocation problem for the secondary LTE downlink network is formulated. The formulation represents the joint objective of the MME as well as the eNB. The main objective of the DRA is to maximize the system data rate while individual UEs' capacity requirements are satisfied. While this approach helps in optimizing the sum rate of the network, it degrades performance of the cell-edge UEs (rate deprived users) as in [13]. This is because rate maximization is done on the basis of instantaneous achievable rates which are generally worse for the cell edge users and thus their QoS is compromised. In our formulation, we account for this problem by introducing a fairness factor $d^c_{j,m}$ for the j-th cell center zone UE and $d^l_{j,m}$ for the j-th cell-edge zone UE located in the l-th sector of the $m-$th cell. The fairness factor for the j-th UE, in general, is modeled as:

$$d^i_{j,m} = \frac{\bar{R}^i_{avg,m}}{(R_{j,m} + \epsilon)} \quad \forall j \in \Upsilon^i_m \tag{3.5}$$

where, $m = 1, ..., M;\ i \in \{c,l\};\ l = 1,2,3$ and $\bar{R}^c_{avg,m}$, $\bar{R}^l_{avg,m}$ are the average throughput, over a time window (of the order of a fraction of the lease duration), of all UEs in the cell-center and l-th sector of cell-edge zone respectively. $R_{j,m}$ is the average throughput of the UE j. The factor ϵ is an arbitrarily small positive constant which prevents the fairness factor from blowing up to infinity for severely rate deprived users. In practical applications, the factor $d^i_{j,m}$ ensures that the instantaneous achievable rate of rate deprived cell-edge users are scaled by a larger factor than those of fulfilled users. Thus it enforces fair allocation at the scheduler by ensuring cell-edge users also get preference as compared to cell-center users who have better achievable rates owing to better channel conditions which result from their closeness to the eNB. Before expounding on the actual problem formulation, some decision variables are introduced in Table 3.2. When $x^c_{j,m,n} = 1$, PRB n is assigned to the UE j which is located in the $m-$th cell-center zone. Since $\mathcal{C}^c \bigcap \mathcal{C}^l = \{\}$, it follows that when $\overline{x^c_{j,m,n}} = 1$, then the PRB is assigned to an UE j located in the cell-edge zone i.e., $x^l_{j,m,n} = 1$. The binary decision variables $\eta^c_{m,n}, \eta^l_{m,n}$ model the decisions at the MME for the m-th eNB and the $n-$th PRB. It is supposed that the MME combines the SAS and local sensing decisions to arrive at a consolidated decision for the presence or absence of PU in cell m. $\eta^c_{m,n}, \eta^l_{m,n} = 1$ indicates that primary or federal user is absent on the $n - th$ PRB in the cell-center/sector region of the m-th cell and the MME of the secondary LTE network can issue a permission to transmit in the $n-$th PRB.

The joint resource allocation problem can thus be stated as:

$$\max_{x^c_{j,m,n}, x^l_{j,m,n}} \underbrace{\sum_{m=1}^{M}\sum_{j\in\Upsilon^c_m}\sum_{n=1}^{N} x^c_{j,m,n}\frac{\eta^c_{m,n}d^c_{m,j}R^c_{j,m,n,t}}{\sum_j d^c_j}}_{\text{cell-center zone}} + \underbrace{\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{l=1}^{L}\sum_{j\in\Upsilon^l_m} x^l_{j,m,n}\frac{\eta^l_{m,n}d^l_{j,m}R^l_{j,m,n,t}}{\sum_j d^l_j}}_{\text{cell-edge zone}} \tag{3.6}$$

$$\text{s.t.} \sum_{j\in\Upsilon^c_m} x^c_{j,m,n} \in \{0,1\}, \quad m=[M]; \quad n=[N], \tag{3.7}$$

$$\sum_{l=1}^{L}\sum_{j\in\Upsilon^l_m} x^l_{j,m,n} + \sum_{j\in\Upsilon^c_m} x^c_{j,m,n} = 1, \quad m=[M];\ n=[N], \tag{3.8}$$

$$\sum_{m=1}^{M}\sum_{j\in\Upsilon^c_m} x^c_{j,m,n} \in \{0,M\}, \quad n=[N], \tag{3.9}$$

$$\sum_{n=1}^{N} x^c_{j,m,n}R^c_{j,m,n,t} <= C^c_{m,j,t}, \quad m=[M];\ j=[|\Upsilon^c_m|], \tag{3.10}$$

$$\sum_{n=1}^{N} x^l_{j,m,n}R^l_{j,m,n,t} <= C^l_{m,j,t}, \quad m=[M];\ l=[L];\ j=[|\Upsilon^l_m,|], \tag{3.11}$$

where we have used the notation $[X] \triangleq \{1,2,\ldots,X\}$ and $C_{m,j,t}$ is the minimum capacity requirement for the j-th user in m-th cell at time t. The objective function (3.6) maximizes the weighted sum rate of the system. The weights used for resource allocation at cell-center/cell-edge take into account the following considerations.

- The MME and SAS decisions for transmissions in a PRB n of the leased spectrum in the cell-center/cell-edge of cell m in the LTE network are fused in the binary variable $\eta^c_{m,n}$.
- A fairness factor is incorporated in the weights to ensure a fair allocation and better QoS for the cell-edge users which could be rate deprived. The normalization is done to ensure that the maximization problem does not attain extreme values when the fairness factor of a particularly rate deprived UE attains a very large value.

The constraints are given by (3.7)-(3.11):

- The constraints in (3.7) and (3.8) enforces that a PRB can be assigned to only one user within a cell in the cell-center zone and the l-th sector of the cell-zone respectively by forcing the sum of the assignment variables $x^c_{j,m,n}, x^l_{j,m,n}$ to be either 0 or 1 for the $n-th$ assigned PRB for all UEs ($j \in \Upsilon^c_m$ or $j \in \Upsilon^l_m$) belonging to the center/sector in cell m.

- The constraints (3.9)) ensures that if a PRB is assigned to either cell-center or any of the three cell-edge groups, it should be reused in the respective groups in all the cells in the network.
- The constraints (3.10) and (3.11) enforces the DRA to satisfy the minimum rate requirement for each user in cell-center and cell-sectors respectively.

The joint maximization in (3.6) evaluates the optimal binary vectors $x^c_{j,m,n}, x^l_{j,m,n}$. It is a integer (binary) programming problem which is a NP-hard problem [92]. Therefore, to reduce the complexity of the joint problem and offer a tractable solution, we formulate a layered FFR based DRA solution which results in a near-optimal solution to the joint optimization problem.

3.2.4 Layered FFR based DRA Problem Formulation

We now formulate the layered DRA problem wherein the joint binary assignment problem is layered into two different assignment problems. Each of the sub-problem is still NP-hard. Therefore, we propose a novel heuristic solution approach which has much lower complexity as compared to the optimal strategy. The design of the layered shared spectrum scheduler follows the approach presented in [13]. The first layer is a MME network level heuristic and finds the optimum allocation of PRBs to one of $(L+1)$ sets (4 sets in this case). The MME computes the assignment of each PRB to the set $\mathcal{C}^c$ or to the sets $\mathcal{C}^l$ using the assignment variables x^c_n and x^l_n. These assignments are reused within each of the M cells. For allocating resources, the MME DRA needs the achievable rates for all the PRBs i.e., R^c_n, R^l_n, the average fairness factors D^c_m, D^l_m and the capacity requirements C^c_m, C^l_m for the cell-center and cell-edge zones respectively. Thus the MME needs $M(N+2)(L+1)$ values for every execution of the MME DRA heuristic. The second layer is a eNB level algorithm and allocates the PRBs from the sets $\mathcal{C}^c$ and $\mathcal{C}^l$ to the individual UEs within the eNB. The eNB maintains time averages of achievable rates for every UE-PRB pair. The additional requirement that this scheme places on the existing standards is that averages are calculated [13] for the cell-center zone UEs and the cell-sector zone UEs. In the next sections, the layered heuristic/algorithm is discussed in detail.

MME-Level DRA Heuristic

The MME-level (referred hereon as Level 1) DRA heuristic assigns the available PRBs within the subgroups $\mathcal{C}^c$ and $\mathcal{C}^l$, $l = [L]$ according to the dynamic FFR-3 scheme. This heuristic makes these assignments based on the average rates achievable if a PRB is assigned to cell center/sectors. In order to ensure QoS of the rate deprived cell-edge users, the fairness factor is considered. It thus increases the sum-rate of the network while being fair to the cell-edge

Algorithm 1 Heuristic for PRB Allocation at Secondary LTE MME

Inputs: $R_n^c,\ R_n^l,\ C_m^l, C_m^c,\ \left(\eta_{m,n}^c, \eta_{m,n}^l\right) \in \{0,1\},\ D_m^c,\ D_m^l, \quad \forall\ j = [J]\ ;\ \ n = [N];\ \ l = [L];\ m = [M]$

Outputs: $x_n^c,\ x_n^l, \qquad \forall\ n = [N];\ l = [L]$

Initialization:

$x_n^c = 0;\ x_n^l = 0;\ g^c = 0;\ g^l = 0; \qquad \forall\ n = [N];\ l = [L]$

$W_n^c = \sum_{m=1}^{M} \left(\frac{D_m^c\ \eta_{m,n}^c\ R_n^c}{\sum_{m=1}^{M} D_i^c \cdot M} \right), \qquad \forall n$ ▷ Utility value for assigning PRB $n \rightarrow \mathcal{C}^c$

$W_n^l = \sum_{m=1}^{M} \left(\frac{D_j^l\ \eta_{m,n}^l\ R_n^l}{\sum_{m=1}^{M} \sum_{l=1}^{L} D_m^l \cdot M} \right), \qquad \forall n, l$ ▷ Utility value for assigning PRB $n \rightarrow \mathcal{C}^l$

$U_n^l = \frac{W_n^l - W_n^c}{\max_l W_n^l}, \qquad l = [L];\ n = [N]$ ▷ Fractional utility gain

$C^l = \sum_{m=1}^{M} C_m^l, \quad \forall l$ ▷ Capacity requirement per cell-edge sector

$C^c = \sum_{m=1}^{M} C_m^c$ ▷ Capacity requirement for cell-center

$\mathcal{Z} = \{1, ..., N\}, \quad \mathcal{L} = \{1, ..., L\}$ ▷ Set of PRBs and sectors respectively

Begin Heuristic:

for $n = 1 \rightarrow N$ **do**
  Find sectors $\mathcal{L}^{'}$, that satisfy $C^l - g^l > 0, \quad l \in \mathcal{L}$
  if $\mathcal{L}^{'} \neq \{\}$ **then**
    Find the sector, PRB pair (l^*, n^*) that satisfies $\max_{l^{'},n} (U_n^{l^{'}}), \quad l^{'} \in \mathcal{L}^{'},\ n \in \mathcal{Z}$
    if $U_{n^*}^{l^*} < 0$ **and** $C^c - g^c \geq 0$ **then**
      $x_{n^*}^c = 1$ ▷ Assign n^*-th PRB to the cell-center
      $g^c = g^c + \sum_{m=1}^{M} \left(\frac{\eta_{m,n^*}^c\ R_{n^*}^c}{M} \right)$ ▷ Update assigned rate for cell-center
    else
      $x_{n^*}^{l^*} = 1$ ▷ Assign n^*-th PRB to the l^*-th sector of cell-edge zone
      $g^{l^*} = g^{l^*} + \sum_{m=1}^{M} \left(\frac{\eta_{m,n^*}^{l^*}\ R_{n^*}^{l^*}}{M} \right)$ ▷ Update assigned rate for l^*-th cell-edge sector
    end if
  else Find the n^*-th PRB which satisfies $\max_n(W_n^c)$ and assign it to cell center.
  end if
  Remove n^* from $\mathcal{Z}$
end for

and cell-center users. It optionally satisfies the minimum per user capacity requirement C_j, while making the assignments. The Level 1 DRA heuristic is presented in Algorithm 1.

For the Level 1 DRA heuristic the binary decision variables are modified to model the assignment of a PRB to one of the $L+1$ sets. We have $x_n^c = 1$ if PRB n is assigned to the cell center set & $x_n^l = 1$ if the $n-$th PRB is assigned to the $l-$ th sector. The MME transmission decision variables are also modified to η_m^c, η_m^l such that they are 1 if MME decides to transmit in the $m-$th cell center/sector zones. This variable is used in the rate maximization problem along with the average fairness factors. The steps of the heuristic are discussed in some detail below:

- **Input:** average achievable rates for each PRB for a given cell-center/sector zone, minimum capacity requirements for the centers and sectors of each cell, the average fairness factors and MME transmission decisions for all the cell-center and sector zones.
- **Outputs:** assignment vector $\overrightarrow{\mathbf{X}}^c = \{x_n^c | n = [N]\}$ and the binary matrix $\mathbf{X}^e = \{x_n^l | n = [N]; l = [L]\}$. The assignments satisfy the constraints that $\mathcal{C}^l \bigcap \mathcal{C}^c = \{\}$ and $\bigcap_{l=1}^{L} \mathcal{C}^l = \{\}$.
- **Initialization:** The heuristic aims to assign each available PRB to one of the $L+1$ available sets in the FRF-3 scheme.
  - The assignment variables x_n^c, x_n^l are initialized to zero signifying that the sets $\mathcal{C}^i, i \in \{c, l\}$ are empty.
  - Two variables g^c and g^l are declared and initialized to zero. They model the increase in the capacity of the cell-center and l-th sector of the cell-edge zone respectively for the case when the PRB n is assigned to either set.
  - The utility values W_n^c and W_n^l measure the utility of assigning a PRB n to the cell-center zone or to the l-th sector of the cell-edge zone
  - The variable U_n^l holds the fractional utility gain for the case when the $n-$th PRB is assigned to the l-th cell-sector zone as opposed to the cell-center zone. A negative value of the fractional utility gain for all l indicates that the allocation of the PRB to the cell-edge zone will lead to a lesser utility than assigning it to the center zone.
- **Heuristic:** The heuristic begins by checking if there are any cell-sectors whose minimum capacity requirements are more than zero.
- If it finds capacity deficient cell-sectors, it next finds the PRB-sector pair (n^*, l^*) that maximizes U_n^l. If maximum fractional utility is negative and the center is still capacity deficient, then the PRB n^* is assigned to the cell-center zone. Else, if the center's capacity demand is satisfied, the PRB is assigned to the l^*-th sector zone. This provides a measure of fairness to the cell-edge users while still maintaining the capacity requirements for the cell-center users.

- Conversely, if in step 2, all the sectors are capacity satisfied and the set $\mathcal{L}^{'} = \{\}$, then the remaining PRBs are assigned to the cell-center zone. This effectively helps maximize the system throughput as the center users typically have better achievable rates as compared to edge users.
- g^c and g^l are updated after each assignment (steps 6 and 10). It is to be noted that the fairness factors are not considered when updating the increase in capacity due to assignment of the PRB to a zone. This is because the the increase in capacity only depends on the achievable average rate of the given zone and the MME transmission decisions within a cell.
- Remove the selected n^* from set of available PRBs and repeat the heuristic for next assignment. The heuristic runs N times.

The average rates, fairness factors and capacity constraints used by the Level 1 DRA heuristic are averaged across the total number of users in the system at any given TTI and then time averaged over a time window of T_{MME} TTIs, where T_{MME} is the number of TTIs after which the MME-level heuristic runs. The Level 1 allocation thus changes every T_{MME} TTIs.

Complexity of Level 1 DRA heuristic: The MME-level DRA heuristic described above performs, for a total of N PRBs, N iterations. Within each iteration it performs a 2 dimensional search over PRBs and sectors to find the optimum n^*, l^*. Thus the heuristic has $\mathcal{O}(N^2)$ complexity. We can modify the heuristic to make it *linear* in complexity. At each iteration, the n-th PRB can be set as the n^*-th PRB i.e., the search for the optimal PRB is eliminated. The heuristic then performs N iterations and for each iteration a constant L searches. The complexity order thus becomes $\mathcal{O}(N)$. It has been shown in Section 3.3, that both heuristics perform nearly identically in terms of system sum throughput and fairness. This is because the parameters used in the heuristic are double averages over time and the number of UEs at each TTI. Thus, the achievable rates for the individual PRBs for a given cell and sector do not vary vastly in magnitude. Thus we favor the use of the linear complexity heuristic as opposed to the more general polynomial complexity heuristic.

eNB-Level DRA Algorithm

At the eNB level, the DRA algorithm runs at every TTI and assigns PRBs to the UEs with each of center and sectors zones. The eNB-DRA (referred hereon as Level 2 DRA) takes into account the individual UE's capacity constraints, instantaneous achievable rates, and fairness factors and aims to maximize the sum throughput of the sector or center zone. The assignment problem then calculates $x^i_{j,m,n}$ given $n \in \mathcal{C}^i$ and $j \in \Upsilon^i_m$ $i \in \{c, l\}$ and $l = 1, 2, 3$. Thus the sectors and centers are allocated PRBs independently of each other within each cell provided $\eta^c_{m,n}, \eta^l_{m,n}$ are not zero. For each cell-center/sector at the t-th TTI, the optimization

problem can be defined as[3]:

$$\max_{x_{j,m,n,t}} \sum_{j\in\Upsilon_m}\sum_{n\in\mathcal{C}} x_{j,m,n,t}\frac{\eta_{m,n}d_{j,m,t}R_{j,m,n,t}}{\sum_j d_{j,m,t}} \tag{3.12}$$

$$\text{s.t.} \quad \sum_{j\in\Upsilon_m} x_{j,m,n,t} \in \{0,1\}, \quad n = 1, ..., |\mathcal{C}|, \tag{3.13}$$

$$\sum_{n=1}^{|\mathcal{C}|} \left(x_{j,m,n,t}R_{j,m,n,t}\right) <= C_{j,m,t}, \quad j = 1, ..., |\Upsilon_m| \tag{3.14}$$

The constraints are given by equations (3.13)-(3.14). The first constraint ensures that a given PRB is assigned to a single UE with the selected zone. The second constraint ensures that the capacity demands of the individual UEs are satisfied. $R_{j,m,n,t}, d_{j,m,t}$ are calculated similar to equations (3.4)-(3.5) respectively at each time instant t. The problem described in equation (3.12) is a general scheduling and rate maximization problem and given the Level 1 PRB assignments, this can be solved by any scheduler proprietary to a LTE service provider. The novelty in this scheduler comes from the use of a weighted linear function as the objective to achieve fairness and throughput maximization. This leads to a reduction in complexity as a linear problem is solved. Since this is an integer assignment problem, standard binary integer programming solutions can be used for solving it.

Metrics for Feedback to the Level 1 DRA heuristic

The eNB calculates the instantaneous achievable rates and fairness factors for all UEs within the given cell and sector for use at the Level 2 algorithm. The following metrics are also evaluated by the eNB for feedback to the Level 1 heuristic. Here, the average is taken over the number of users at a given instant. Instantaneous average achievable rates for the cell center/sector zones:

$$\bar{R}^i_{n,t} = \frac{1}{M}\sum_{m=1}^{M}\sum_{j=1}^{|\Upsilon^i_m|}\frac{R_{j,m,n,t}}{|\Upsilon^i_m|}, \quad \forall j \in \Upsilon^i_m. \tag{3.15}$$

Instantaneous average fairness factors:

$$\bar{d}^i_{m,t} = \sum_{j=1}^{|\Upsilon^i_m|}\frac{d^i_{j,m,t}}{|\Upsilon^i_m|}, \quad \forall j \in \Upsilon^i_m. \tag{3.16}$$

[3]In this formulation, we discard the notations differentiating the cell-center and sector zones. It is assumed that the given sets are of either type based on where the algorithm is being deployed.

Instantaneous average capacity requirements:

$$\bar{C}^i_{m,t} = \sum_{j=1}^{|\Upsilon^i_m|} \frac{C^i_{j,m,t}}{|\Upsilon^i_m|}, \quad \forall j \in \Upsilon^i_m, \quad \text{where } i \in \{c, l\}, \quad l = 1, 2, 3. \tag{3.17}$$

The instantaneous values are stored for T_{MME} TTIs. At the end of T_{MME} TTIs, the instantaneous average values are time averaged to produce the average rates R^c_n, R^l_n, the average fairness factors D^c_m, D^l_m and the average capacity requirements C^c_m, C^l_m. The local sensing decisions from the eNBs are also fed back to the MME. The amount of feedback needed by the proposed layered heuristics is thus drastically reduced as only 6 values need to be fed back as opposed to the joint problem where per user data needs to be fed back to the MME.

The time scale of operation of the heuristics can be made dynamic by having the Level 2 DRA query the MME every TTI to check if the Level 1 allocation has changed. If not, the Level 2 algorithm can continue with the existing Level 1 allocation. Else, it can start over with the new allocation sets. At the network level, the MME can also check if a new allocation adds more utility to the previous one and only then forward this allocation to the eNBs. The next section discusses in detail, the simulation results for the proposed layered DRA scheme.

3.3 Simulation Results

In this section, the proposed schemes are compared against the traditional frequency reuse 1 and a static Naive FFR-3 scheme which allocates resources in Level 1 statically while using the proposed Level 2 algorithm. The metrics for comparing these schemes are sum throughput, fairness in allocation and the average number of users served. We additionally use an alternate implementation of the Level 1 heuristic for comparison purposes. In this implementation, an optimal binary integer programming solution is used for the PRB assignments at the MME level similar to the Level 2 algorithm. The objective function is the maximization of average achievable rates weighted by the average fairness factors and subject to per center/sector capacity constraints. This solution is optimal but exponentially complex. We show that our scheme while being linear in complexity, achieves comparable performance while serving a similar percentage of UEs within the system.

In the Naive FFR-3 scheme the available PRBs are divided equally into 4 sets and assigned to sectors and centers. It is shown that our scheme is a tradeoff between the reuse 1 scheme which is unfair to cell-edge UEs and the Naive FFR-3 scheme, which owing to its static and equal resource allocation to sectors and centers, is fair. Our scheme achieves a tradeoff in fairness and throughput. We also show the variations in the system as the MME transmission decisions as well as the federal leased bandwidth changes. The fairness is measured in terms of the average number of users served with a non-zero data rate and the average Jain Index

Scheme Name	**Scheme Details**
Scheme 1	Proposed Level 1 heuristic with $\mathcal{O}(N^2)$
Scheme 1 LC	Proposed Level 1 linear complexity heuristic
Scheme 1 Bintprog	Optimal solution to Level 1 using integer programming
Trad Scheme	Traditional reuse 1 scheme
Naive FFR-3	FFR-3 scheme with static Level 1 allocation

Table 3.3: Details of Simulated Resource Allocation Schemes

Channel Model	COST-Hata Model for metropolitan areas
Center Freq of Legacy Spectrum	1800 MHz
Center Freq of Leased Spectrum	3500 MHz
Sub-carrier separation (Δf)	12 kHz
LTE system BW	5 MHz (25 PRBs)
No. of cells in network	7
Radius of each cell	2 km
Height of eNB	80 m
Height of UE	10 m
Radius of cell-center	$0.6R$(cell radius)
eNB transmit power	40 dBm (maximum)
UEs per center/sector zone	$\sim$ Uniform $[6, 10]$
$\sigma_{\text{Shadowing}}$	7 dB
Lease duration	100 TTIs
T_{MME}	10

Table 3.4: Simulation Parameters

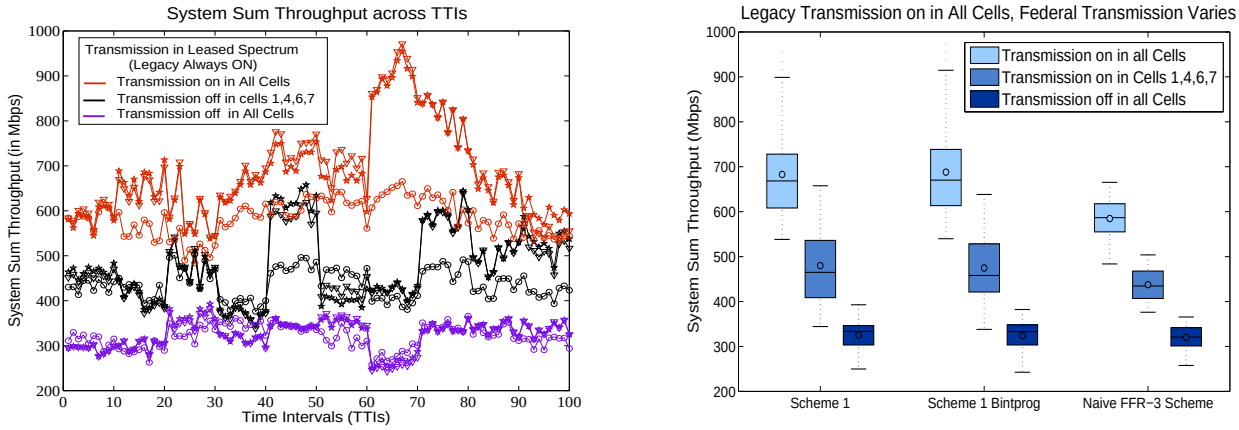
(JI) [85]. It is given by:

$$\mathcal{JI}(r_1, r_2, ..., r_J) = \frac{\left(\sum_{j=1}^{J} r_j\right)^2}{J \cdot \sum_{j=1}^{J} {r_j}^2} \tag{3.18}$$

where r_j is the rate/throughput of the j-th user and J is the total number of UEs. A higher value of Jain Index indicates a better fairness measure.

Table 3.3 lists the different schemes used for the simulations. The schemes in Table 3.3 do not use the fairness factor weighted eNB level algorithm and are used for comparing the fairness achieved at the Level 1 DRA only. A variation of each scheme using fairness factor at Level 2 (eNB level) is also simulated to compare the overall end-to-end system. The simulation parameters used are as given in Table 3.4.

For the simulations, we consider a 7 cell wrap-around universe to cancel edge-of-the-world effects. Also, when a transmission is not permitted in a cell/sector, we consider that the primary is active in the cell and it causes an increase in the noise floor for the adjacent cells



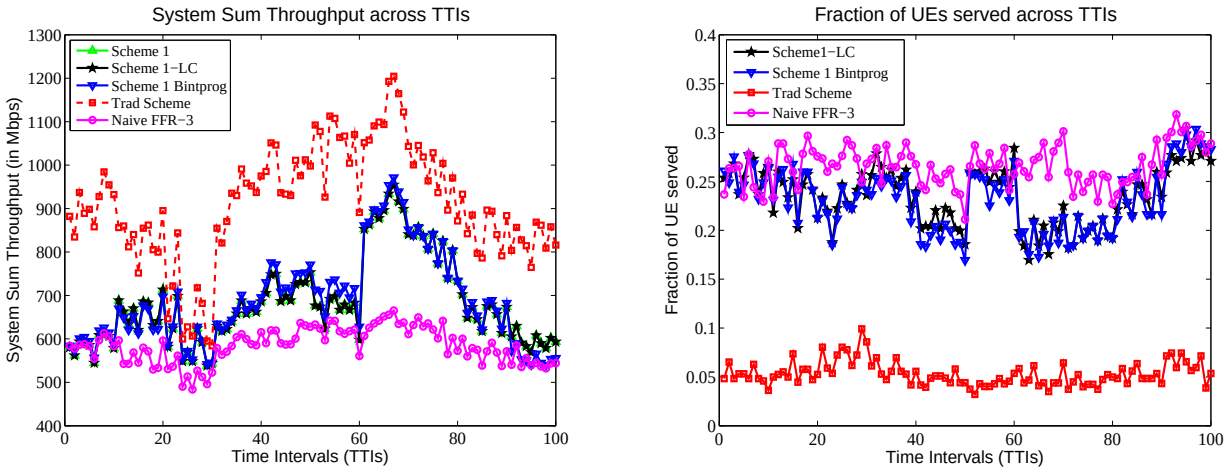
(a) (b)

Figure 3.3: (a) Comparison of Average Sum Throughput of the System with varying MME transmission permissions. The symbols representing different schemes are: $\star$ - Scheme 1; $\triangledown$ - Scheme 1 Bintprog; $\circ$ - Naive FFR-3 Scheme.; (b) Boxplot showing the 25th and 75th percentile, the mean (circle) and median (line) of the rates. The line extending from the boxes show the outliers.

that are using leased primary spectrum. We assume that the SAS can at most lease out 5 MHz of primary spectrum. The (legacy) LTE spectrum is set to 5 MHz around 1800 MHz band. We consider the case where the demand in the cluster is very high and the MME requests the entire available primary bandwidth. The SAS either gives access to the entire 5 MHz of spectrum or a part of it. We also note that given the high demand, not all users can be served with non-zero rate when rate-maximization is employed. For all the subsequent comparisons, we assume the LTE system's operating bandwidth (legacy + leased) is fixed in order to make comparisons fair. The simulation results are presented in the following subsections.

3.3.1 Variation in MME Transmission Permission

In Figure 3.3a, we show the impact of varying the MME transmission decisions on the system sum throughput of the LTE network. The transmission decisions are a result of fusion of local sensing decisions and the SAS permissions. We use the information in Figure 3.2b, for simulating the scenario when the usage of leased spectrum is restricted to a part of the network. Here, transmission on leased spectrum is prohibited in cells $1, 4, 6$ and 7. The other two cases are when transmission in the leased spectrum is allowed in all cells and when it is not allowed in any of the cells. It is to be noted that the cells can still transmit always on their licensed legacy spectrum. It can be seen that the system throughput changes predictably i.e., it is minimum for the schemes when no transmission is permitted and maximum when all cells can transmit. The third case with partial transmission is intermediate. It should be



(a) (b)

Figure 3.4: (a) Comparison of Average System Throughput of different schemes with time. (b) Comparison of fraction of UEs served by the schemes across time.

noted that the relative performance of all the schemes are similar in the three cases. This is further highlighted by the boxplot in Figure 3.3b which shows the variability in the rates for each scheme and transmission permission. Thus, in order to analyze the difference between the respective schemes, we can choose any one of these permission models. Thus without loss of generality and to keep results tractable, for comparing different schemes, we choose the case when all cells are allowed to transmit in the leased spectrum. We next look at cases where the SAS leases out the entire 5 MHz or a fraction of it.

3.3.2 5 MHz Leased Spectrum, Transmission in all Cells

In Figure 3.4a, the average system throughput across time is shown for the cases when fairness is not used in the Level 2 Algorithm. We note that *Scheme 1* and *Scheme 1 LC* are identical in performance. This is attributed to the fact the utility of assigning a PRB to a UE remains almost the same for adjacent PRBs. Thus, searching to find a n^* does not hold much utility. Henceforth *Scheme 1* and *Scheme 1 LC* are treated identically. We also see that the *Trad Scheme* achieves the best throughput performance while the *Naive FFR-3* scheme is the worst. This is expected as the reuse 1 scheme greedily maximizes the rate while the Naive scheme allocates PRB among center/sectors uniformly without any bias and the Level 2 algorithm maximizes the rates given the allocation. The implications can be observed in Figure 3.4b in terms of the fraction of UEs served. The traditional scheme serves a tiny fraction of UEs who have the best channels. The static allocation of the Naive FFR-3 scheme implies that it serves a higher fraction of users. These are the two extreme cases. The proposed schemes present a trade-off. They achieve higher throughput while serving a number of users comparable to the naive FFR-3 scheme. It can also be observed that the

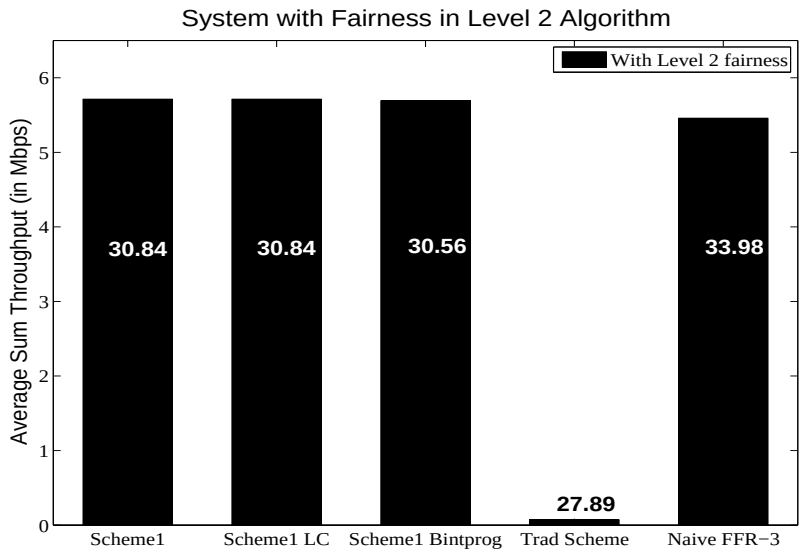


Figure 3.5: Comparison of Average Sum Throughput of the System. The number in the bars represent the average number of users served by the schemes.

more complex *Scheme 1 Bintprog* performs marginally better if not identically in terms of average throughput and fraction of users served.

In Figure 3.5, we evaluate different schemes with Level 2 fairness incorporated. The results are averaged over time and users and hence reflect the average performance of each scheme over space and time. In the subsequent figures, we use "Average" to mean the time average over the users in system unless specified otherwise. It can be readily observed that the use of Level 2 fairness at the eNB reduces the data rate drastically. However the percentage of UEs served increases. However, in the case of the traditional scheme the percentage of UEs served increases but it assigns almost negligible rate to the UEs resulting in a very low average system sum throughput. Figure 3.6 shows the average Jain Index for the different schemes. This fairness metric reflects the conclusion from the previous results. In case of no Level 2 fairness, the traditional scheme is the least fair while the naive FFR-3 scheme's static allocation leads to higher fairness. The proposed schemes are a tradeoff between the two extreme cases. In the case when the Level 2 fairness is used, the traditional scheme increases in fairness but it comes at the cost of very low throughput. In both cases the *Scheme 1* and *Scheme 1 LC* have identical fairness. They both achieve slightly higher fairness than the *Scheme 1 Bintprog*.

The average per user throughput in the center zone vs the cell-edge zones is shown in Figure 3.7 for the case of no Level 2 fairness. It can be seen that the proposed schemes provide almost the same capacity to the edge UEs as the naive scheme. The cell-center allocation however results in more bandwidth in the proposed schemes. The traditional reuse one scheme in this case gives almost the entire capacity to the cell-center UEs and ignores the

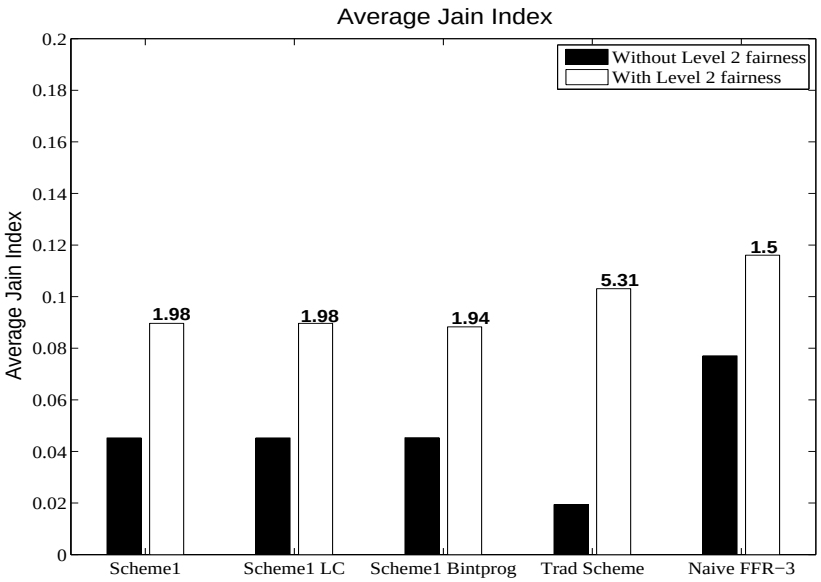


Figure 3.6: Comparison of Average Jain Index of the System.

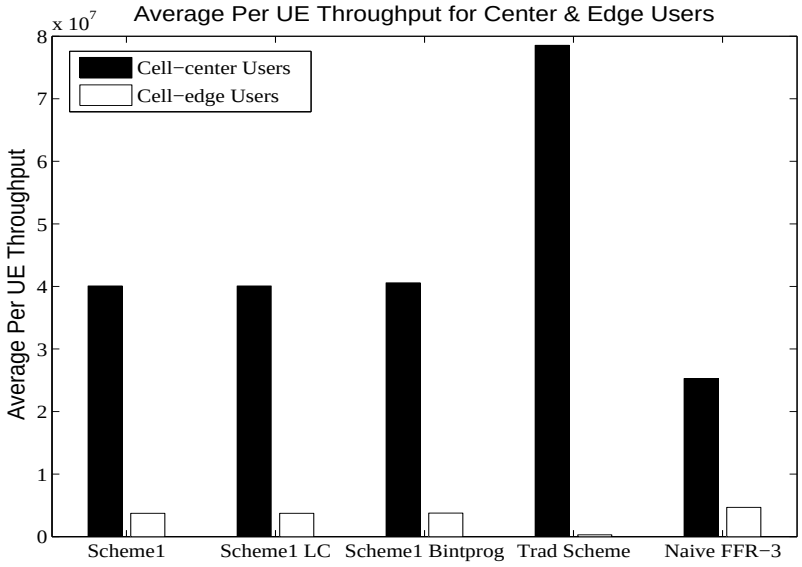


Figure 3.7: Comparison of Average Per UE Sum Throughput of Cell Center vs Cell Sectors.

cell-edge UEs.

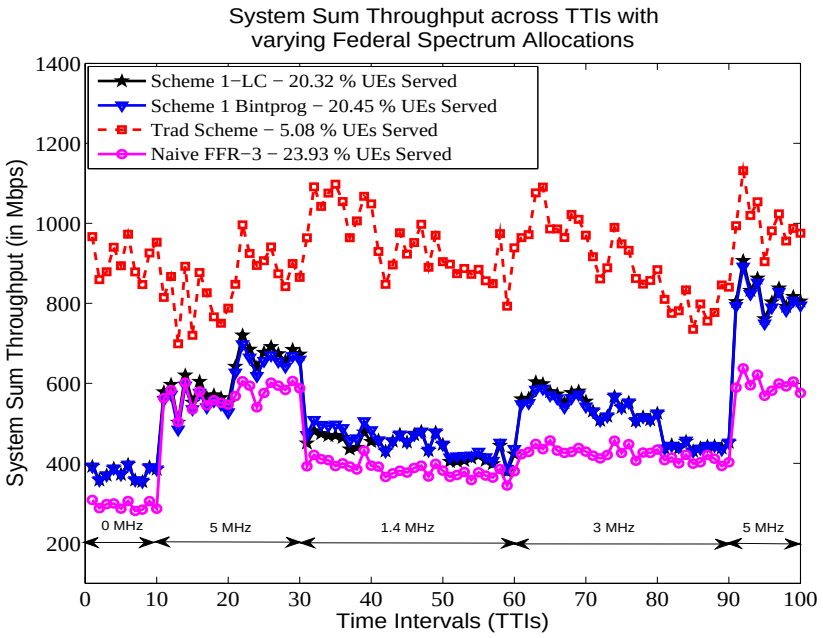


Figure 3.8: Comparison of Average Sum Throughput across time with changing lease bandwidths.

3.3.3 Leased Spectrum BW Varies, Transmission in all cells

In this case, we assume that the bandwidth allocated to the secondary network by the SAS is varying with time. In Figure 3.8, we start with no leased spectrum for the first 10 time intervals. Then the allocation changes to 5 MHz for the next 20 time intervals and then goes down to 1.4 MHz for the next 30 time intervals and so on. The lease is requested and granted on a preemptive basis and each time the allocation changes, the secondary network runs an initialization in order to calculate the time averaged rates and fairness factors. During this initialization phase, the old allocation can continue. In Figure 3.8, we observe that there is a sharp change in throughput for *Scheme1* and *Scheme1 Bintprog* when allocations change. In case of the *Trad Scheme*, the change in throughput is not as apparent. This implies that the traditional scheme makes lesser use of the costly leased spectrum at the cell edges. It can also be seen that the traditional scheme gives a consistently higher sum throughput but serves a lesser fraction of UEs owing to its greedy nature. Interestingly, the low complexity algorithm and the bintprog solution yield almost identical results. The naive FFR-3 on the other hand allocates lesser throughput on a consistent basis.

In future, when the Carrier-Aggregation becomes a feature in LTE-A networks, the proposed scheme would be ideally suited to take advantage of it. When the available spectrum at MME is aggregated from widely separated frequency bands, the static allocation of the naive FFR-3 scheme will not be practical and the proposed schemes will perform consistently better. Thus our algorithm is ideally suited to network sharing in LTE-A networks.

3.4 Conclusions

In this chapter, a novel layered dynamic resource allocation scheme for spectrum sharing in LTE/LTE-A has been proposed. A model for a spectrum sharing architecture has been discussed, where an access request is based on current demand and usefulness of a requested frequency band. The secondary network controller (MME) fuses the transmission permissions from the SAS and local sensing decisions to form transmission decisions. The proposed MME-level heuristic not only mitigates ICI and increases throughput, but also gives importance to rate-deprived users by allocating PRBs based on an average fairness factor. The complexity of the MME level heuristic is polynomial and at best linear. Both these low complexity heuristic/algorithm achieve system throughput close to the high complexity algorithms presented in literature. The eNB level allocation also takes into account the fairness factor and uses a weighted linear objective to allocate resources optimally. The system as a whole has been shown to provide a tradeoff in fairness and throughput compared to the extreme cases of the traditional frequency reuse 1 scheme and the naive static FFR3 scheme.

Chapter 4

Power Control in Cache Enabled Energy Harvesting Small Cells

The demand for mobile data particularly video content has been growing significantly in recent years and is projected to continue like this in future [16]. In order to meet the extremely high data rate requirements of mobile traffic, the use of millimeter Wave (mmW) access links in a dense small cell network has been proposed in literature [17]. Since the range of small cells is roughly 10 m to 100 m, mmW combined with massive MIMO techniques could be used to focus extremely high data rate beams on the mobiles and thus meet the QoS requirements of mobile video content consumers. Another hindrance to high data rate service is the limited backhaul capacity since it gets shared among a large number of small cell base-stations (SBS).

To decrease the traffic load from the capacity-constrained backhaul, caching of popular user data at the SBS has been suggested in [11, 19]. It studied the problem of optimal content placement across distributed caches such that the expected sum delay in downloading users data is minimized. Related works are discussed in [20–22]. Nevertheless, while interesting, these works primarily focus on content placement and update so as to minimize the total delay in downloading the user data. The impact of caching on the energy consumption and backhaul usage for the small cell network is yet to be explored.

The design of energy efficient transmission schemes is becoming increasingly important for mobile networks. According to a Ericsson report on energy emissions [93], about 10% of total Information and Communications Technology (ICT) emissions are attributable to cellular networks, and more than half of that is due to the radio access network. Although the power consumption of a single SBS is low (around 2 W for a typical femtocell), the overall energy consumption of the small cell network will be high due to their dense deployment. As a result, it is desirable to exploit off-grid and green energy sources such as solar energy to power the small cell networks. Mao et. al. in [18] used case studies to analyze the practical feasibility of various green energy sources for small cells and concluded that both solar and

wind energy are viable solutions for powering up SBS's. Hence, it is important to design energy efficient transmission techniques to reduce probability of power outage events and thus satisfy QoS guarantees.

In this chapter, we develop an *online* energy-efficient power control scheme for an energy harvesting SBS that is endowed with caching capabilities and a wireless, capacity-limited backhaul. Energy arrival are modeled as Poisson process and the popularity distribution of user data is modeled using Zipf's law. The utility function of the SBS is a generalized energy-efficiency function that alters the degree of energy efficiency depending upon the current battery level. The power control problem is formulated as a (discounted) infinite horizon dynamic programming problem and solved numerically using the value iteration algorithm. We show that for a given QoS requirement of users, the cache size at SBS can be traded off with the capability of energy harvesting equipment.

The rest of the chapter is organized as follows. Section 4.1 introduces the system model. In Section 4.2, we formulate the optimization problem to determine the transmit powers on downlink and backhaul. We then propose a value iteration algorithm to solve this problem. Section 4.3 presents simulation results. Finally Section 4.4 draws some conclusions.

4.1 System Model

Figure 4.1 shows the system model of an SBS serving N_u mobile users. The SBS is equipped with a local cache of size M and is connected to the core network using a wireless (licensed spectrum) backhaul link. We also assume that the SBS is self-powered and it obtains its power supply through an energy harvesting device that is connected to a rechargeable battery with capacity $E_{\max}$.

In this model our goal is to study the impact of caching and energy harvesting on resource allocation in a SBS. In our case, resources consist of power and bandwidth allocated to the users. The scheduler at the SBS performs resource allocation at the start of each time slot of duration T seconds. Let $E_k \in [0, E_{\max}]$ be the energy remaining in the battery at the start of the k^{th} time slot. Then the evolution of E_k over time can be written recursively as,

$$E_{k+1} = \min(E_k - P_k T + \tilde{E}_k, E_{\max}), \quad \forall\ k = 0, 1, 2, ... \tag{4.1}$$

Here, $\tilde{E}_k$ is a random variable denoting the energy harvested during the k^{th} slot. The energy arrivals are typically modeled as a Poisson process with mean λ [23–25]. Let q be the amount of energy harvested at each arrival, which depends on the capability of the energy harvesting device present at the SBS. So $\tilde{E}_k = N_a q$, where N_a is the number of arrivals in time T with a mean value of λT. The variable $P_k \in [0, \min(P_{\max}, \frac{E_k}{T})]$ in (4.1) represents the total transmit power of the SBS during the k^{th} slot. $P_{\max}$ is the maximum transmit power constraint on the SBS, but the actual upper bound is $\min(P_{\max}, \frac{E_k}{T})$ because the energy remaining in the battery cannot be negative at any instant. The total power P_k can be written in terms of

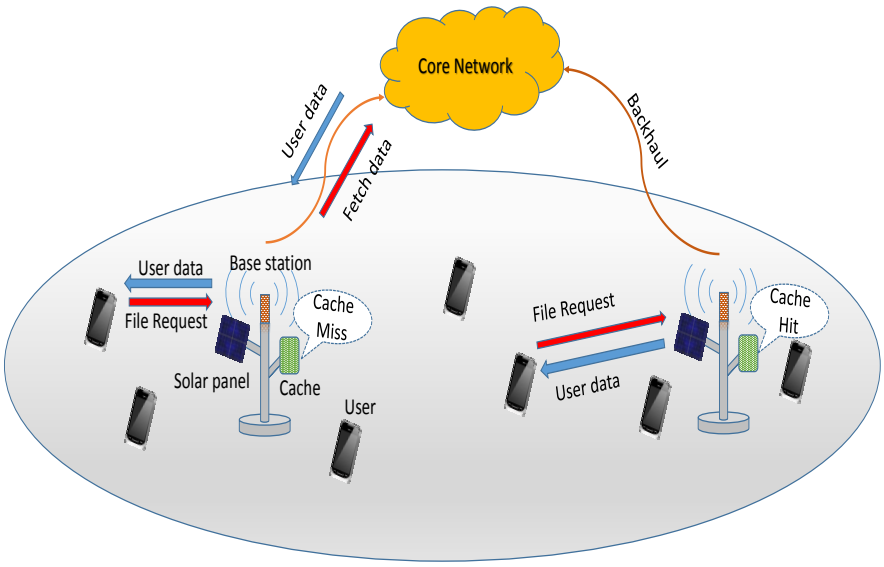


Figure 4.1: System Model

its constituents as,

$$P_k = \sum_{i=1}^{N_u} P_{i,k} + \epsilon P_{b,k}, \tag{4.2}$$

where $P_{i,k}$ is the transmit power on downlink for user i during the k^{th} slot. Similarly, $P_{b,k}$ is the SBS transmit power spent on fetching the user requested content from the core network using the backhaul. The backhaul link is accessed with a probability ϵ, which is related[1] to the probability of cache miss, ϵ_m, such that $\epsilon = 1 - (1 - \epsilon_m)^{N_u}$. We assume that the SBS caches the M most popular files among the users, because the probability of users requesting these files is more than other files and thus minimizes the backhaul usage for fetching the requested content from the core. Therefore, caching the popular data is likely to result in higher energy savings and lower latency than just randomly caching user content. We model the content popularity distribution using Zipf's law which states that the frequency of an element (file) of rank j in a set of R elements is given as,

$$f(j; s, R) = \frac{1/j^s}{\sum_{r=1}^{R} 1/n^s}, \tag{4.3}$$

where $s > 1$ is a decay constant characterizing the *peakiness* of the distribution: a large value indicating that a very small number of files account for majority of data traffic. The rank of a file indicates its relative popularity, with a lower rank indicating higher popularity. The probability of cache miss, ϵ, is the probability that one of the files that has not been cached

[1]The backhaul is accessed for fetching the files from the core network if atleast one of files requested by the N_u users is not present in the cache. The probability that all the users find their requested files in cache is $(1 - \epsilon_m)^{N_u}$. Therefore probability of using backhaul is $1 - (1 - \epsilon_m)^{N_u}$.

is requested by the user and can be calculated as,

$$\epsilon = \sum_{j=M+1}^{R} f(j; s, R) = \frac{\sum_{j=M+1}^{R} 1/j^s}{\sum_{r=1}^{R} 1/n^s}. \tag{4.4}$$

For a given resource (bandwidth, power) allocation and battery level, we define the utility of the SBS at time slot k as,

$$U_k(\boldsymbol{w}_k, \boldsymbol{p}_k, \boldsymbol{h}_k, E_k) = \frac{\sum_{i=1}^{N_u} W_{i,k} \log_2(1+\gamma_{i,k})}{\left(\sum_{i=1}^{N_u} P_{i,k}\right)^{g(E_k)}}, \tag{4.5}$$

where $W_{i,k}$ is the bandwidth allocated to the i^{th} user and the backhaul respectively during the k^{th} slot[2]. For simplicity, we have $\boldsymbol{w}_k = [W_{1,k}, W_{2,k}, \cdots, W_{N_u,k}]$ and $\boldsymbol{p}_k = [P_{1,k}, P_{2,k}, \cdots, P_{N_u,k}]$. The variables $\gamma_{i,k}$ in (4.5) denotes the received signal-to-noise ratio (SNR) at the i^{th} user at k^{th} slot, and are given by,

$$\gamma_{i,k} = \frac{P_{i,k} \|h_{i,k}\|^2}{d_i^{\alpha} \sigma^2}, \tag{4.6}$$

where $h_{i,k}$ denotes the downlink channel between the SBS and i^{th} user. d_i is the distance[3] between the SBS and the i^{th} user and α is the path loss exponent. All the channels are assumed to be i.i.d., Rayleigh fading channels. σ^2 is the variance of the Gaussian noise.

When the user-requested data is not present in the local cache at SBS, then the requests must be forwarded to the core network via backhaul. When it receives the data from core network, the SBS transmits it back on the downlink[4]. Let $\gamma_{b,k}$ denote the received SNR at the core network when the backhaul is used to fetch the files from the core network. $\gamma_{b,k}$ can be written as,

$$\gamma_{b,k} = \frac{P_{b,k} \|h_{b,k}\|^2}{d_b^{\alpha} \sigma^2}, \tag{4.7}$$

where $h_{b,k}$ denotes the Rayleigh fading channel between the SBS and the core network. d_b is the distance between the SBS and the core network. The SBS transmit power on backhaul, $P_{b,k}$, should be large enough so that $\gamma_{b,k} \geq \gamma_{\min}$ at each time slot k. Otherwise, the receiver at core network cannot correctly decode the files requested by the users.

The function $g(E_k)$ in (4.5) alters the degree of energy efficiency in next time slot depending upon the current battery level, E_k. Although many functions can be used, we define $g(E_k)$ as,

$$g(E_k) = a + \frac{(b-a)E_k}{E_{max}}, \tag{4.8}$$

[2]In LTE, the scheduler allocates the bandwidth to a UE at each TTI in multiples of Physical Resource Block (PRB), where a PRB comprises of 12 subcarriers each 15 kHz wide.

[3]We assume that the users are stationary over the time duration of interest.

[4]The contents of the cache may be updated when new data is downloaded using backhaul, if file popularity distribution has changed over the course of time.

where a and b are constants ($a > b > 0$), that determine the extent to which E_k influences energy efficiency. We choose this function because it is desirable to be more energy efficient when battery level is low, due to uncertainty in harvested energy. Clearly, the function $g(E_k)$ in (4.8) satisfies this and is a non-increasing function of E_k. Also, both $g(E_k)$ and U_k are well-defined at all values of E_k.

We now determine the average utility of the SBS, $U_k^{\text{avg}}(\boldsymbol{w}_k, \boldsymbol{p}_k, E_k)$, by averaging over all possible channel realizations as following,

$$U_k^{\text{avg}}(\boldsymbol{w}_k, \boldsymbol{p}_k, E_k) = \mathbb{E}_{\boldsymbol{h}_k}\left(U_k(\boldsymbol{w}_k, \boldsymbol{p}_k, \boldsymbol{h}_k, E_k)\right), \tag{4.9}$$

$$= \frac{\mathbb{E}_{\boldsymbol{h}_k}\left(\sum_{i=1}^{N_u} W_{i,k}\log_2(1+\frac{P_{i,k}\|h_{i,k}\|^2}{d_i^{\alpha}\sigma^2})\right)}{\left[\sum_{i=1}^{N_u} P_{i,k}\right]^{g(E_k)}}, \tag{4.10}$$

where $\mathbb{E}_{\boldsymbol{h}_k}(.)$ denotes the expectation operation over the channel vector $\boldsymbol{h}_k = [h_{1,k}, h_{2,k}, \cdots, h_{N_u,k}]$. Equation (4.10) follows from substituting (4.5) and (4.6) in (4.9). Since we assumed the channel to be Rayleigh fading, the squared norm of the channel, $H_{i,k} = \|h_{i,k}\|^2$, is exponentially distributed with some rate parameter, $\mu \ \forall \ i, k$, and is given by,

$$f_{H_{i,k}}(x) = \mu \mathrm{e}^{-\mu x} \ \forall \ i, k. \tag{4.11}$$

Then using (4.10) and (4.11), we have,

$$U_k^{\text{avg}}(\boldsymbol{w}_k, \boldsymbol{p}_k, E_k) = \frac{\sum_{i=1}^{N_u} W_{i,k}\int_0^{\infty}\log_2(1+\frac{P_{i,k}x}{d_i^{\alpha}\sigma^2})\mathrm{e}^{-\mu x}dx}{\frac{1}{\mu}\left[\sum_{i=1}^{N_u} P_{i,k}\right]^{g(E_k)}}. \tag{4.12}$$

Let us solve the integral $I = \int_0^{\infty}\log_2(1+\frac{P_{i,k}x}{d_i^{\alpha}\sigma^2})\mathrm{e}^{-\mu x}dx$. Let $r = \frac{P_{i,k}}{d_i^{\alpha}\sigma^2}$. Applying integration by parts to I, we have,

$$I = \frac{1}{\mu\ \ln(2)}\left(-\ln(1+rx)\mathrm{e}^{-\mu x}\Big|_0^{\infty} + \int_0^{\infty}\frac{r}{1+rx}\mathrm{e}^{-\mu x}dx\right), \tag{4.13}$$

$$= \frac{\mathrm{e}^{\frac{\mu}{r}}}{\mu\ \ln(2)}\int_{\frac{\mu}{r}}^{\infty}\frac{\mathrm{e}^{-z}}{z}dz = \frac{\mathrm{e}^{\frac{\mu d_i^{\alpha}\sigma^2}{P_{i,k}}}\Gamma(0, \frac{\mu d_i^{\alpha}\sigma^2}{P_{i,k}})}{\mu\ \ln(2)}, \tag{4.14}$$

where (4.14) follows from (4.13) by noting that $\lim_{x\to\infty} -\ln(1+rx)\mathrm{e}^{-\mu x} = 0$ (using L'Hospital's rule) and by setting $z = \mu(x+\frac{1}{r})$. Then noting that the integral is a upper incomplete Gamma function, $\Gamma(0, \frac{\mu}{r})$ and substituting back the value of r, we get the final expression in (4.14). Substituting the solution of the integral in (4.12) we get,

$$U_k^{\text{avg}}(\boldsymbol{w}_k, \boldsymbol{p}_k, E_k) = \frac{\mu\sum_{i=1}^{N_u}\left(W_{i,k}\mathrm{e}^{\frac{\mu d_i^{\alpha}\sigma^2}{P_{i,k}}}\Gamma(0, \frac{\mu d_i^{\alpha}\sigma^2}{P_{i,k}})\right)}{\ln(2)\left[\sum_{i=1}^{N_u} P_{i,k}\right]^{g(E_k)}}. \tag{4.15}$$

4.2 Problem Formulation and Solution Methodology

In this section, we first formulate the optimal power control problem at the SBS as an infinite horizon dynamic programming[5](DP) problem and then present a solution approach for finding the optimal transmit power at each time slot.

For the system model described in the last section, the problem at the SBS is that of optimal power control $\boldsymbol{p}_k^*$ for given energy E_k and channel state $\boldsymbol{h}_k$ at time slot k. It is determined by solving for the power control scheme that maximizes the sum of current and (discounted) future utility. Mathematically this is given as,

$$\boldsymbol{p}_k^*(E_k, \boldsymbol{h}_k) = \arg\max_{\boldsymbol{p}_k} \quad U_k(\boldsymbol{w}_k, \boldsymbol{p}_k, E_k, \boldsymbol{h}_k) + \delta \sum_{E_{k+1}} \mathcal{P}(E_{k+1}/E_k, \boldsymbol{p}_k) V_{\pi^\dagger}(E_{k+1}), \tag{4.16}$$

such that

$$p_{i,k} \geq 0, \forall\ i = 1, 2, \cdots, N_u, , \tag{4.17}$$

$$\sum_{i=1}^{N_u} P_{i,k} + \epsilon P_{b,k} \leq \min(P_{\max}, \frac{E_k}{T}). \tag{4.18}$$

The backhaul power, $P_{b,k}$ in (4.18) is set such that $\gamma_{b,k} = \gamma_{\min}$. We set $P_{b,k} = P_{i,k} = 0$, when $\frac{\min(P_{\max}, \frac{E_k}{T})\|h_{b,k}\|^2}{d_b^\alpha \sigma^2} < \gamma_{\min}$.

$\delta \in [0, 1)$ is a discount factor for the utility of future states. $\mathcal{P}(E_{k+1}/E_k, \boldsymbol{p}_k)$ denotes the transition probability from energy state E_k to E_{k+1} when the transmit power in the k^{th} slot is $\boldsymbol{p}_k$. $V_{\pi^\dagger}(E_{k+1})$ is the *value* of state E_{k+1} under the transmission policy $\pi^\dagger(E_k)$ and is defined iteratively as,

$$V_{\pi^\dagger}(E_k) = U_k^{\text{avg}}(\boldsymbol{w}_k, \boldsymbol{p}_k, E_k) + \delta \sum_{E_{k+1}} \mathcal{P}(E_{k+1}/E_k, \boldsymbol{p}_k) V_{\pi^\dagger}(E_{k+1}), \tag{4.19}$$

where $\boldsymbol{p}_k = \pi^\dagger(E_k)$. Note that the value function uses the average utility function defined in (4.9).

The policy $\pi^\dagger(E_k)$ is the one that maximizes the value function $V_\pi(E_k)$, subject to certain power constraints. So we have,

$$\pi^\dagger(E_k) = \arg\max_{\boldsymbol{p}_k} \quad U_k^{\text{avg}}(\boldsymbol{w}_k, \boldsymbol{p}_k, E_k) + \delta \sum_{E_{k+1}} \mathcal{P}(E_{k+1}/E_k, \boldsymbol{p}_k) V_{\pi^\dagger}(E_{k+1}), \tag{4.20}$$

subject to (4.17), (4.18).

The backhaul power, $P_{b,k}$ in (4.18) is set such that $\frac{P_{b,k}}{\mu d_b^\alpha \sigma^2} = \gamma_{\min}$.

[5]The SBS continually serves the users that are associated to it as long as the energy in the battery is above a particular threshold, so it is reasonable to consider an infinite-horizon problem.

Algorithm 2 Proposed Value Iteration Algorithm

Procedure: Value_Iteration $(\mathcal{E}, A, \mathcal{P}, U^{\text{avg}}, \theta)$
Inputs
$\mathcal{E} = \{0, 1, \cdots, E_{\max}\}$ is the set of all energy states
A is the set of all transmit power vectors, $\boldsymbol{p}$
$\mathcal{P}$ is the state transition function specifying $\mathcal{P}(E^{'}/E, \boldsymbol{p})$
U^{avg} is the utility function $U(\boldsymbol{p}, E, \boldsymbol{h})$
θ is a threshold, $\theta > 0$
Outputs
$\pi^{\dagger}[\mathcal{E}]$ is optimal policy
$V[\mathcal{E}]$ is the value function
Local
real array $V_m[\mathcal{E}]$ is a sequence of value functions
action array $\pi^{\dagger}[\mathcal{E}]$
Begin Algorithm:
$V_0[E] \leftarrow 0 \ \ \forall E \in \mathcal{E}$
$m \leftarrow 0$
repeat
$m \leftarrow m + 1$
for $E = 0$ to $E_{\max}$ **do** ▷ Update value of each state
$V_m(E) = \max_{\boldsymbol{p} \in A} \ U^{\text{avg}}(\boldsymbol{w}_k, \boldsymbol{p}, E) + \delta \sum_{E^{'}} \mathcal{P}(E^{'}/E, \boldsymbol{p}) V_{m-1}(E^{'})$
end for
until $\forall E, \ |V_m[E] - V_{m-1}[E]| < \theta$
for $E = 0$ to $E_{\max}$ **do** ▷ Determine the optimal policy
$\pi^{\dagger}(E) = \arg\max_{\boldsymbol{p}_k \in A} U^{\text{avg}}(\boldsymbol{w}_k, \boldsymbol{p}, E) + \delta \sum_{E^{'}} \mathcal{P}(E^{'}/E, \boldsymbol{p}) V_m(E^{'})$
end for
return $\pi^{\dagger}, V_m$

It is not possible to obtain a closed form solution to (4.16), because the objective function depends on the *value* of next state which in turn depends on the *value* of next-to-next stage and so on. Therefore, we need to solve the DP problem numerically. Policy iteration and value iteration are the two popular algorithms for numerically solving the DP. Policy iteration converges quickly, but it requires search over set of all feasible policies which is typically a very large set. On the other hand, value iteration iteratively updates the value of each state by determining the actions that would maximize the value of that state. Therefore, the search is over a much smaller set. The drawback is slow convergence. We decided to use value iteration to solve the DP in (4.16). However, before doing that, we would need to discretize the state space and action space. The detailed algorithm is described in Algorithm 2.

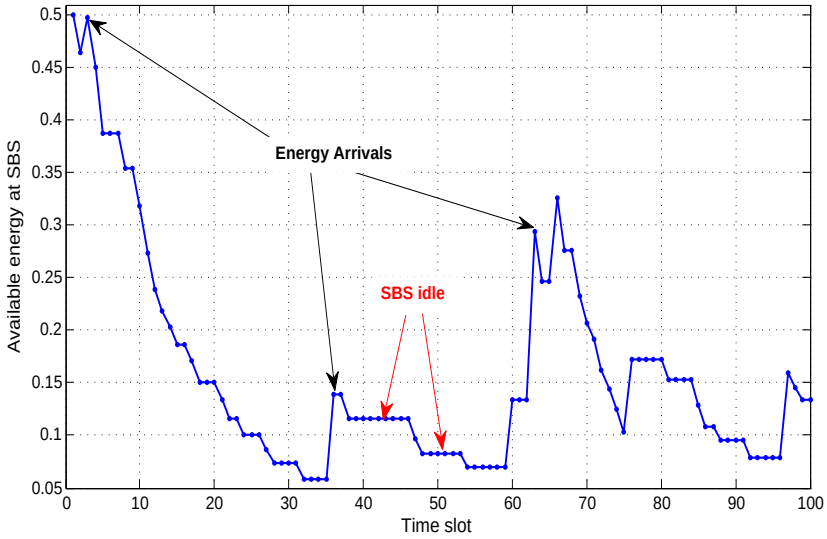


Figure 4.2: Energy dynamics for $M = 80$ and $\lambda = 0.1$.

4.3 Simulation Results

For simulations, we consider a rural small cell network in which the SBS is located at the center and N_u users are deployed randomly within a circle of radius 800 m. The separation between the SBS and core[6] is set to 3 km. The battery capacity $E_{\max}$ is 2 J and the maximum transmit power of the SBS, $P_{\max}$ is 0.8 W. The amount of harvested energy per arrival is set to $q = 80$ mJ, while the time slot duration is set to $T = 1$ ms. The total number of files that the users can request, N, is assumed to be very large for practical purposes. These files are assumed to be of same size and the cache size M represents number of files cached. The exponent s in (4.3) is set to 2. The fading parameter μ in (4.11) is 1. The constants a and b in (4.8) are set to 0.18 and 0.03 respectively. The discount factor $\delta = 0.7$. The minimum SNR for backhaul, $\gamma_{\min}$ is set to 8 dB. The value of path loss exponent,α, is set to 3. The noise variance σ^2 is set to -90 dBmW. Since the focus of this work is to understand the impact of energy harvesting and caching on the downlink resource allocation at SBS, for simplicity, we assume equal bandwidth, $W = 100$ kHz, for each user and backhaul.

Figure 4.2 shows the variations in the energy available at SBS over time. The available energy decreases monotonously with time except for the time instants with energy arrivals or low battery level. When the battery level is low, the SBS enters energy-saving mode i.e., it remains idle when channel conditions are bad and until more energy is harvested.The rate of decrease in energy decreases with decrease in available energy at the SBS. This is due to the function $g(E_k)$ in in (4.5) that makes the SBS more energy-efficient when the battery level is low. Figure 4.2 also indicates the energy arrivals and time slots for which the SBS is

[6]If the separation between SBS and core is huge, relay nodes or macroBS may be deployed to relay the backhaul data to the core.

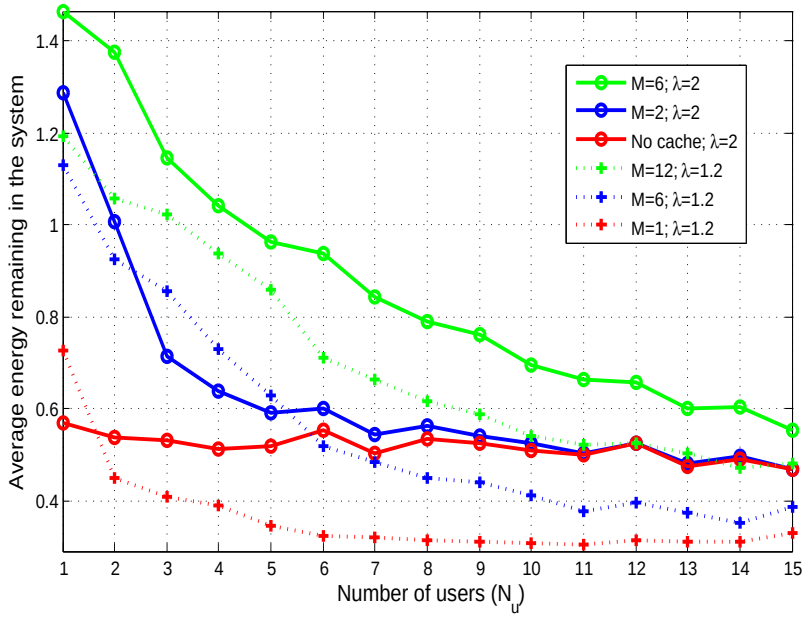


Figure 4.3: Impact of energy harvesting and cache size on available energy.

idle (no transmission). We note that energy arrivals are infrequent because λ is small.

Figure 4.3 shows the plot of average energy *remaining* in the battery as a function of the number of users for different cache size M and energy harvesting rate λ. We note that for a given energy harvesting rate, the available energy in the battery decreases sharply as the cache size is decreased. This is because the probability of miss, ϵ_m increases with the decrease in cache size. Therefore, the SBS will start using the backhaul more aggressively and, thus the energy expenditure increases. For $N_u = 10$ and $\lambda = 2$ s^{-1}, caching two files as opposed to none, results in 40% higher energy savings. In Figure 4.3, we also note that, as N_u increases, the energy expenditure increases due to higher probability of backhaul usage, given by $\epsilon = 1 - (1 - \epsilon_m)^{N_u}$. Finally, we note that the degradation in (energy) performance due to the decrease in energy arrival rate ($\lambda : 2 \rightarrow 1.2$) can be compensated by increasing the cache size ($M : 2 \rightarrow 6 \rightarrow 12$).

Figure 4.4 shows the average sum throughput of users as the number of users varies for different cache size. The rate of energy arrivals is $\lambda = 2$ s^{-1}. We note that sum throughput increases with the increase in cache size. This is due to the fact that the frequency of backhaul access becomes smaller and thus more power is available for downlink transmissions. In this figure, we also compare the throughput performance of our proposed power control scheme with a sum throughput maximization scheme[7]. We note that, contrary to intuition, our scheme results in higher throughput (20% gain at $(N_u, M) = (11, 6)$) than the sum throughput maximization scheme. This is because in case of sum throughput maximization scheme, the SBS transmits at its maximum power level while being oblivious to the channel

[7]The sum throughput maximization scheme can be obtained from the utility function defined in (4.5) by setting the denominator to 1.

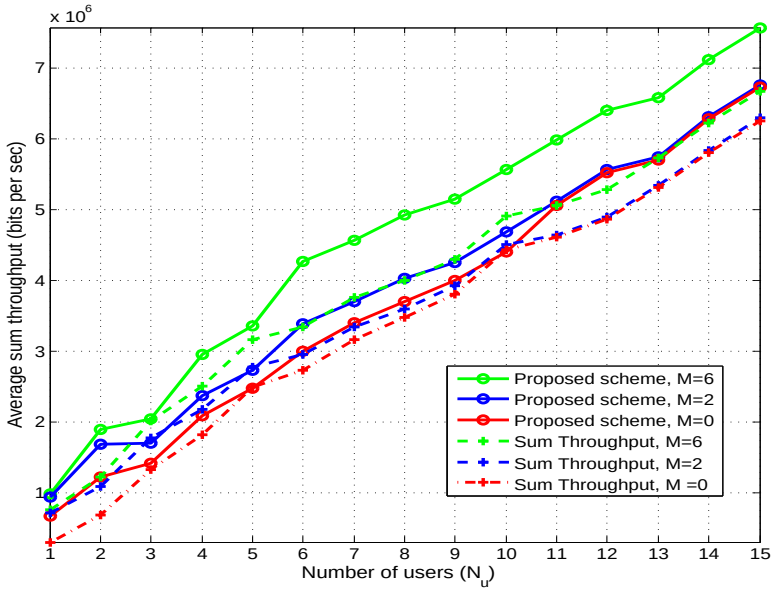


Figure 4.4: Impact of energy harvesting and caching on throughput.

state information. In the case of a deep fade, this will exhaust the battery power. If this is followed by a period of low energy arrivals, then the SBS will not be able to transmit at high power levels even in excellent channel conditions. This results in a significant loss in the sum throughput compared to the proposed scheme.

In Figure 4.5, we show the impact of jointly varying the cache size M and the energy harvested per arrival q on the energy availability at SBS. N_u is set to 15. We note that increasing M from 1 to 160 or q from 0.1 to 0.9, increases the energy availability. Increasing q however has a more pronounced effect than increasing M. This is because we cache files in order of decreasing popularity, so increasing M results in diminishing returns. In order to achieve a desired (energy) performance, the cache size and energy harvesting capacity can be traded off. This is illustrated in Fig. 4.5 by considering the points $(M, q) = (40, 0.7)$ and $(120, 0.5)$ which correspond to roughly same available energy of 0.63 J. The practical implication of this result is that a network operator can make a better decision (after factoring in the storage and energy harvesting equipment costs) about the desired cache size and size of energy harvesting device that achieves desired performance.

4.4 Conclusions

In this chapter, we have presented an energy efficient power control scheme for an energy harvesting SBS equipped with a wireless backhaul and local storage. We have formulated the power control problem as a discounted infinite horizon dynamic programming problem and solved it numerically using value iteration algorithm. Using simulations, we have revealed key

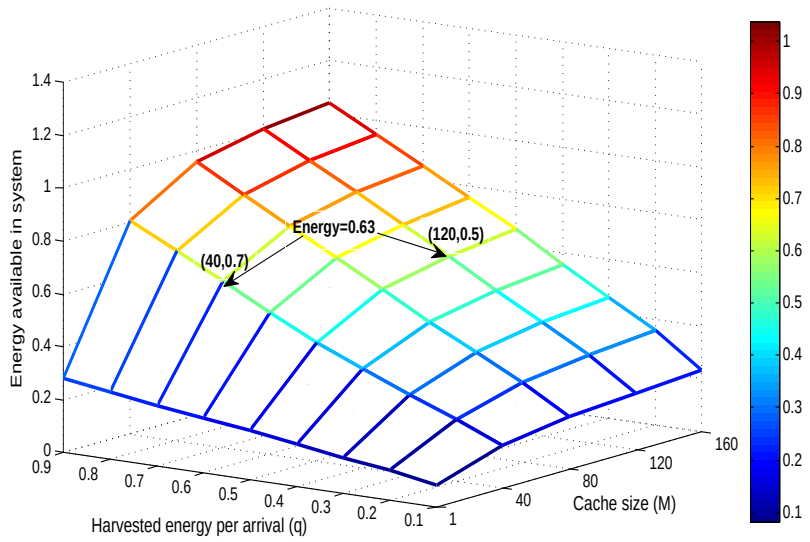


Figure 4.5: Effect of cache size and energy harvesting on energy available in system.

insights on the impact of caching and energy harvesting on the energy consumption at SBS as the number of users in system is varied. Increasing cache size or energy harvesting rate results in increased energy availability. The proposed scheme also provides higher throughput as compared to the baseline sum throughput maximization scheme. We have also shown that the cache size can be bargained for energy harvesting capability and vice versa, while still meeting the desired system performance.

Chapter 5

Read Latency in Heterogeneous Distributed Storage Systems

Cloud based storage systems are emerging to gain significant prominence due to their highly virtualized infrastructure that presents cost-effective and simple to use elastic network resources. The backbone infrastructure of the cloud is comprised of distributed storage systems (DSS), in which the data is stored and accessed from commodity storage disks. Coding of data across distributed disks provides fault tolerance by providing reliability against unexpected disk failures, such as in Google File System [27]. There has been a recent paradigm shift from classical replication based codes to erasure, and more generally, regenerating codes, not only because they provide higher fault tolerance at the same storage cost but also in part due to the novel and efficient repair mechanisms [94], [95], [96]. Besides providing fault tolerance, minimizing storage cost and designing efficient repair mechanisms, another important aspect which deserves equal, if not more attention is that of *latency performance* of such systems.

The primary focus in literature has been on a homogeneous (n, k) DSS wherein all the read requests are considered identical [28–30]. Each file is split it into k equal-sized fragments which are then stored across n servers using a (n, k) Maximum-Distance-Separable (MDS) code to provide storage redundancy and thus fault-tolerance against disk failures. However, in practice, the frequency of read requests and degree of storage redundancy depends on the popularity of the file. A more popular file will be requested more (higher arrival rate) and will stored with higher redundancy or fault-tolerance. Therefore, in this chapter, we propose a $(n, k_1, k_2, ..., k_R)$ heterogeneous DSS model that stores the data of R data-classes characterized with different job[1] arrival rates and redundancy requirements set by using a (n, k_i) code for the i^{th} class.

The key contributions of this chapter are:

[1] In this chapter, we use the terms 'job' and 'read requests' interchangeably.

- A multi-tenant DSS is proposed and analyzed through the Fork Join framework to account for the heterogeneity in job arrival rates and fault-tolerance requirements of different data classes.
- A *data throughput* based energy efficiency metric is defined for the heterogeneous DSS operating under any given scheduling policy. For the special case of single server and data class, we showed that the average latency and energy efficiency of DSS are closely related to each other. Therefore, using a queuing-theoretic approach, we provided lower and upper bounds on the average latency for jobs of class i ($\forall i \in \{1, 2, ..., R\}$) in the proposed F-J framework under various scheduling policies such as First-Come-First-Serve (FCFS), preemptive and non-preemptive priority scheduling policies.
- We studied the impact of varying the code-rate on the latency, energy efficiency and network bandwidth consumed by DSS. Increasing code-rate reduces latency and increases energy efficiency. However, this comes at the cost of increased storage space and (write) bandwidth. We also obtained interesting insights from investigating the impact of varying the number of servers, heavy-tail arrival/service distributions in the DSS.
- Lastly, we studied the impact of varying the number of redundant requests (sending requests to more than k servers for (n, k) MDS code) to the DSS. We observed that sending redundant requests reduces latency and increases energy efficiency. Thus, full redundancy results in minimum latency and maximum energy efficiency for each data-class.

The rest of this chapter is organized as follows. Section 5.1 presents the heterogeneous DSS model and introduces the energy efficiency metric for DSS. Section 5.2 reviews existing queuing theoretic latency analysis for homogenous DSS. Section 5.3 presents the main results and proofs for latency analysis of proposed heterogenous DSS. Then in Section 5.4, we use Monte-Carlo simulations to evaluate the accuracy of latency bounds and the relationship between latency and energy efficiency of DSS as number of servers, code-rate, arrival rate etc. is varied. Lastly Section 5.5, we draws some conclusions.

5.1 System Model

A heterogeneous multi-tenant $(n, k_1, k_2, ..., k_R)$ DSS (shown in Fig. 5.1) consists of n servers that store the data of R distinct classes. The R classes differ from each other in the fault-tolerance, storage requirements and frequency of access of the stored data. The data of class i (which is assumed to be of size l_i) is partitioned into k_i equal size fragments and then stored across n servers using a (n, k_i) Maximum-Distance-Separable (MDS) code. Thus each server stores, $1/k_i$ fraction of original data. The arrival process for request of class i is assumed to be Poisson with rate λ_i. The service time at each server is assumed to follow an exponential

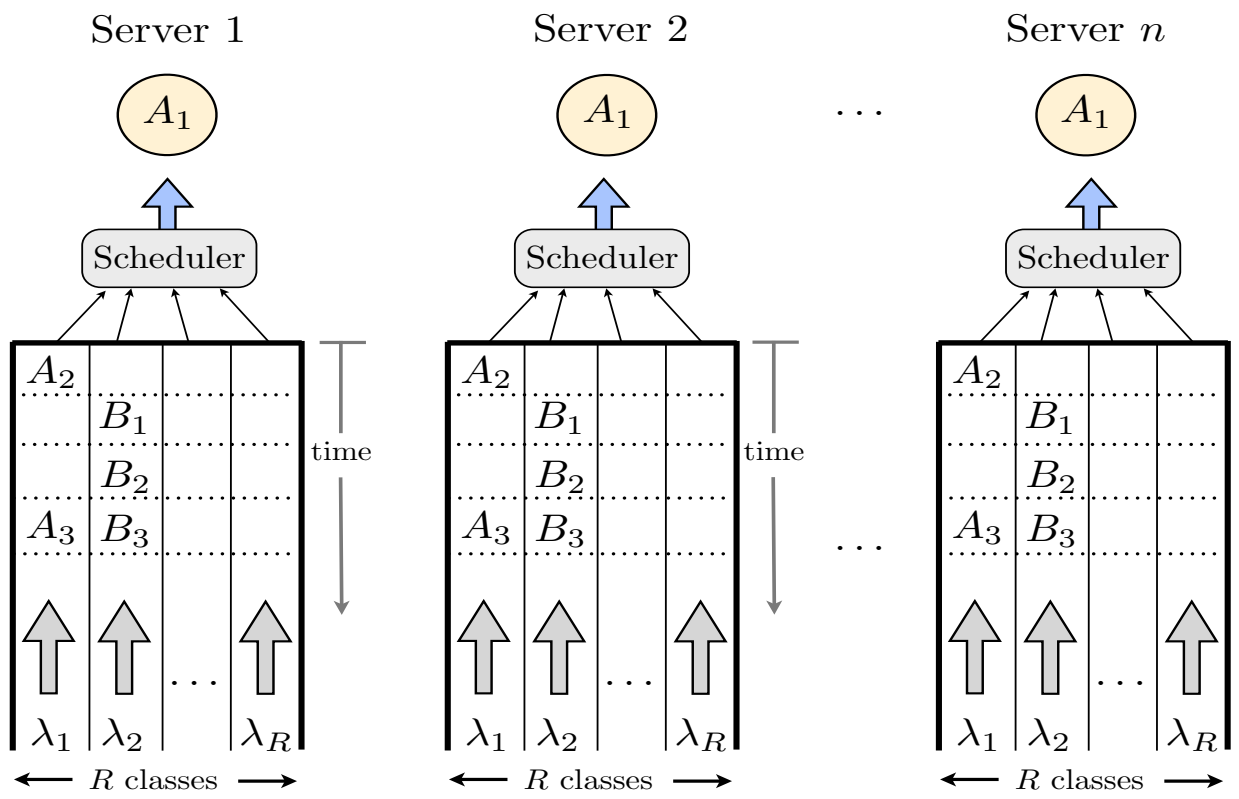


Figure 5.1: System Model for a Heterogeneous Distributed Storage System.

distribution with service rate μ (per unit file size) [97]. The effective service rate at any server for jobs of class i is $\mu_i = \frac{k_i \mu}{l_i}$ since each server stores $1/k_i$ fraction of data.

Example 5.1. *We now present a representative example to illustrate the system model. Consider a* $(n, k_1, k_2) = (3, 2, 1)$ *two-class DSS. Data for the two classes A and B are encoded across* $n = 3$ *servers using* $(3, 2)$ *and* $(3, 1)$ *MDS codes respectively as shown in Fig. 5.2. Let* A_1 *and* B_1 *denote two files of class A and B respectively that need to be coded and stored across the servers. Then for the* $(3, 2)$ *MDS code,* A_1 *is split into two sub-files,* A_{11} *and* A_{12}*, of equal size and are stored on any two servers (servers 1 and 2 in Fig. 5.2). Then the remaining server (i.e. server 3) stores* $A_{11} \oplus A_{12}$*. Thus each server stores half the size of original file and the entire file can be recovered from any two servers. The* $(3, 1)$ *MDS code for file* B_1*, is a simple replication code in which each server stores the copy of entire file of class B and thus can be recovered by accessing the data from any one server.*

The evolution of system state in this example, depends on the local scheduling policy at each server. Although there exists various scheduling policies, in this work we consider First-Come-First-Serve (FCFS), preemptive and non-preemptive priority queuing policies at each server. In FCFS scheduling, all data classes are equal priority. At each server, the job that enters first in the buffer is served first. In a priority queuing policy, the data classes are

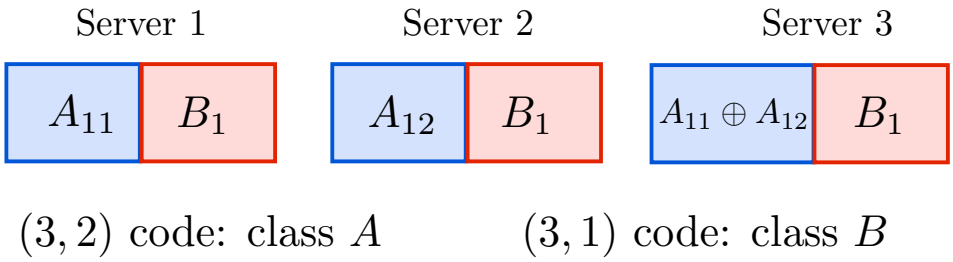


Figure 5.2: MDS codes for data storage in a two-class Fork-Join system.

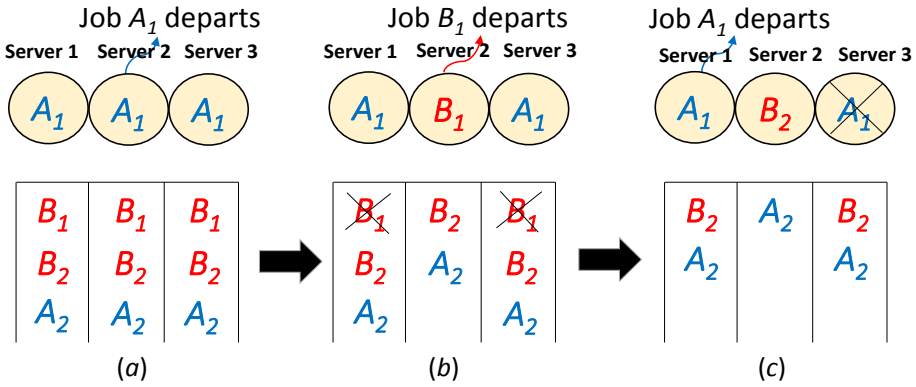


Figure 5.3: System state evolution: two-class FJ system with FCFS.

assigned different priority levels. A job of a particular class will be served only when there are no outstanding jobs of classes with higher priority level. A priority queuing policy is further classified as preemptive or non-preemptive based on whether or not the job in server can be preempted by a job of higher priority level.

Fig. 5.3(a)-(c) illustrates the evolution of system state under the FCFS policy. After server 2 finished job A_1 in Fig. 3(a), B_1 enters server 2 and is finished in the next state (Fig. 3(b)) while other servers still process A_1. Since $k_B = 1$, the remaining two copies of B_1 immediately exit the system. Finally in Fig. 3(c) server 1 finishes A_1 and since $k_A = 2$, A_1 exits at server 3.

5.1.1 Latency and Energy Efficiency

Definition 1. *For a* $\{\lambda_i, l_i, \mu, (n, k_1, k_2, ..., k_R)\}$ *DSS, the average latency of class* i *under some scheduling policy* $\mathcal{P}$ *is defined as*

$$T^i_{\mathcal{P}} = T^i_{s,\mathcal{P}} + T^i_{q,\mathcal{P}}, \tag{5.1}$$

where $T^i_{s,\mathcal{P}}$ and $T^i_{q,\mathcal{P}}$ are the average service time and waiting time (in queue) for a job of class i respectively.

For a $\{\lambda_i, l_i, \mu, (n, k_1, k_2, ..., k_R)\}$ DSS operating under scheduling policy $\mathcal{P}$, a relevant metric for measuring the energy efficiency, $\mathcal{E}_\mathcal{P}$, of the DSS is the data transfer throughput metric [98]. It is defined as the limiting ratio of the amount of data processed, $D_\mathcal{P}(t)$, by the DSS to the energy consumed, $E_\mathcal{P}(t)$, by the DSS in a infinitely large time duration t. It has units of bits/Joule. Now $D_\mathcal{P}(t)$ is simply,

$$D_\mathcal{P}(t) = \sum_{i=1}^{R} l_i N_i(t), \tag{5.2}$$

where $N_i(t)$ is the number of jobs of class i processed by DSS in a time interval t. In order to determine $E_\mathcal{P}(t)$, we model the power consumption of the DSS as follows:

- To reduce power consumption, the servers are equipped with the dynamic voltage/frequency scaling (DVFS) mechanism and low-power states [99]. The DVFS mechanism reduces operating voltage and CPU processing speed (or frequency) in step to reduce utilization and hence increase power savings.
- The power consumed by a server in any state is the sum of power consumed by the CPU and the platform which comprises of chipset, RAM, HDD, Fan etc. The power consumed by the CPU and platform in a given state is assumed to be same across all the n servers.
- The power consumed by a server (CPU and platform) while being in active and low-power state is denoted by P_{on} and P_{off} respectively. A server is in active mode during the busy periods (i.e., there are outstanding jobs waiting for service). In general, at the end of a busy period, a server remains active for a while and then enters a sequence of low-power states staying in each for a predetermined amount of time. For ease of analysis, we lump them into a single low-power state with constant CPU power, C_l and constant platform power, P_l. After the busy period is over, the server remains in active mode for d_l and then enters the low-power state[2]. When the busy period restarts, the server incurs a wake-up latency w_l in which it consumes active mode power, but is not capable of processing any job requests. Fig. 5.4 explains this using an example.
- The CPU power during active mode, C_a is proportional to $V^2 f$, where V is the supply voltage and f is the CPU operating frequency[3] ($f \in [0, 1]$) and are set by the DVFS mechanism. Further, we assume that V is proportional to f [99]. So $C_a = C_0 f^3$, for some maximum power C_0. The power consumed by the platform during active mode, P_a, is constant.

[2]As a consequence, if the duration of idle period (time between end of a busy period and start of the next one) is smaller than d_l, then the server always remains active).

[3]Due to this, the effective service rate for class i becomes $\mu_i = f k_i \mu / l_i$.

- $t_{\text{busy}}^{i,j,k}$ denotes the duration of time for which the k^{th} server is *busy* serving j^{th} job of i^{th} class.
- $t_{\text{idle}}^{i,j,k}$ denotes the duration of idle period after the k^{th} server finished the j^{th} job of i^{th} class.

Using the above notations, the active mode power per server is $P_{\text{on}} = C_a + P_a = C_0 f^3 + P_a$. Similarly, $P_{\text{off}} = C_l + P_l$. Consider any time duration t of interest during the operation of DSS. During this period, the total time for which the DSS is in active mode, t_a, is sum total (across all servers) of all busy periods plus the active mode time before entering low-power state. Mathematically, we have,

$$t_a = \sum_{i=1}^{R} \sum_{j=1}^{N_i(t)} \sum_{k=1}^{n} t_{\text{busy}}^{i,j,k} + \max\,(0, t_{\text{idle}}^{i,j,k} - d_l) \tag{5.3}$$

The total time for which DSS is in low-power state, t_l is,

$$t_l = nt - t_a. \tag{5.4}$$

We have now the following definition of energy efficiency of a DSS.

Definition 2. *For a $\{\lambda_i, l_i, \mu, (n, k_1, k_2, ..., k_R)\}$ DSS, the energy efficiency of the DSS under some scheduling policy $\mathcal{P}$ is defined as,*

$$\mathcal{E}_{\mathcal{P}} = \lim_{t\to\infty} \frac{D_{\mathcal{P}}(t)}{E_{\mathcal{P}}(t)}, \tag{5.5}$$

$$= \lim_{t\to\infty} \frac{\sum_{i=1}^{R} l_i N_i(t)}{P_{on} t_a + P_{off} t_l}, \tag{5.6}$$

where (5.6) follows from (5.5) using (5.2). The expressions for t_a and t_l are given in (5.3) and (5.4) respectively.

Next in order to highlight the relationship between the average latency and energy efficiency of a DSS, we consider the special case of a M/M/1 system and a single data-class. For tractability of analysis, here we assume that d_l, w_l and P_{off} are all 0. Then from the Definition 1 for average latency[4], we have,

$$T = T_s + T_q, \tag{5.7}$$

$$= \frac{1}{\mu'} + \frac{\lambda}{\mu'(\mu' - \lambda)} = \frac{1}{\mu' - \lambda}, \tag{5.8}$$

[4]In this special case, the scheduling policy $\mathcal{P}$ and class index i are not relevant and hence dropped from 5.1.

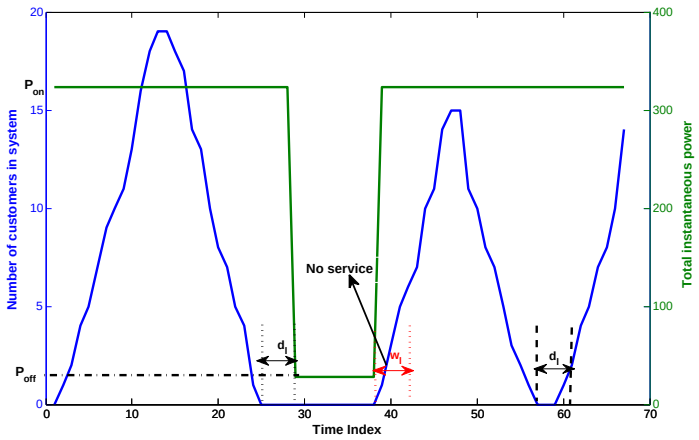


Figure 5.4: Variation of total power consumption of DSS across multiple busy periods and idle periods. The switch to idle state happens only for idle periods with duration greater than d_l but it then results in a wake-up latency of w_l.

where (5.8) follows from (5.7) by noting that for a M/M/1 system, the mean service time is $T_s = \frac{1}{\mu'}$ and mean waiting time is $T_q = \frac{\lambda}{\mu'(\mu'-\lambda)}$. Here, $\mu' = \frac{\mu f}{l}$ is the effective service rate. Here, The energy efficiency is computed using (5.6) as

$$\mathcal{E} = \lim_{t\to\infty} \frac{lN(t)}{P_{\text{on}} t_{\text{a}} + P_{\text{off}} t_{\text{l}}}, \tag{5.9}$$

$$= \lim_{t\to\infty} \frac{lN(t)}{P_{\text{on}} \sum_{i=1}^{N(t)} T_{s,i}}, \tag{5.10}$$

$$= \frac{l}{P_{\text{on}} \lim_{t\to\infty} \frac{\sum_{i=1}^{N(t)} T_{s,i}}{N(t)}}, \tag{5.11}$$

$$= \frac{l}{P_{\text{on}} T_s}, \tag{5.12}$$

where (5.10) follows from (5.9) by noting that t_{on} is sum of service time of each of $N(t)$ jobs (denoted by $T_{s,i}$ for the i^{th} job) and by neglecting the power consumed when server is idle i.e., $P_{\text{off}} = 0$. Then (5.12) follows from (5.11) from the definition of average service time. Thus the energy efficiency is inversely related to the average service time of jobs. It is difficult to find a closed form expression for the energy efficiency of a heterogeneous DSS but the general trend of inverse proportionality between latency and energy efficiency continues to hold true as verified through extensive simulations in Section 5.4. The average latency is also directly related to the average service time[5]. Therefore, we conclude that energy efficiency and average latency of a DSS are closely related to each other. Henceforth, we focus on the latency analysis of a heterogeneous DSS.

[5]Queuing delay depends on job arrival rate and service time. So the latency which is sum of queuing delay and service time directly depends on service time.

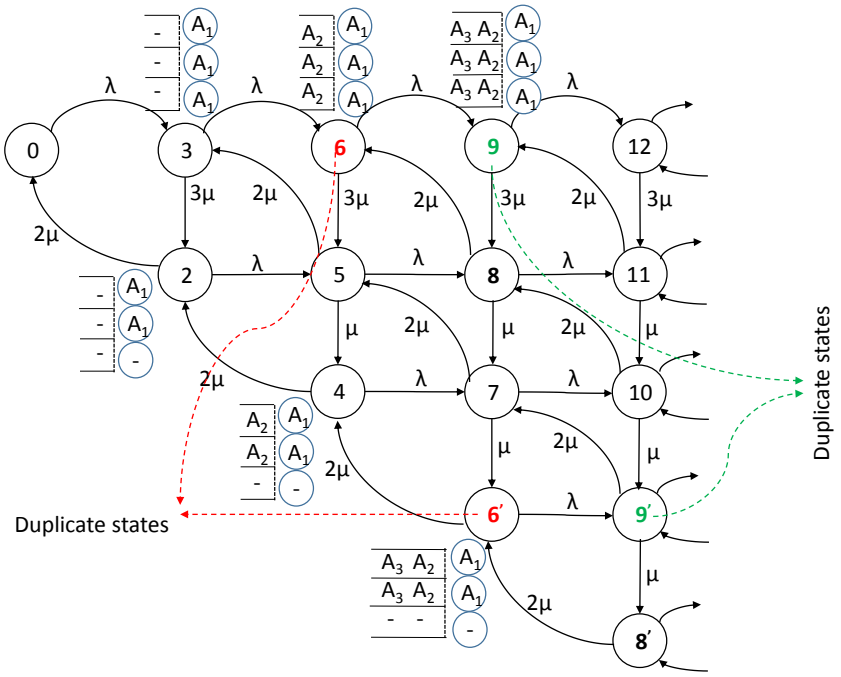


Figure 5.5: Markov chain for a $(3, 2)$ Fork-Join system.

5.2 Preliminaries

An exact latency analysis of the (n, k) DSS is prohibitively complex because the Markov chain has a state space with infinite states in at least k dimensions. This is exemplified in Figure 5.5 which shows the Markov chain evolution for a $(3, 2)$ DSS. Each state is characterized by the number of jobs in the system. The arrival and service rates of jobs are λ and μ respectively. We note that as more jobs enter the system, the Markov Chain starts growing in two-dimensions and results in multiple states with the same number of jobs in the system such as states 6 and $6^{'}$. Thus, we note that an exact analysis of the F-J system is very complex. Therefore, we review existing upper- and lower-bounds for the average latency of homogeneous DSS.

5.2.1 Lower Bound on Average Latency

In a (n, k) DSS, a job is considered finished when k out of n servers finish that job. This is equivalent to each job going through k stages sequentially, where the transition from one stage to the next occurs when one of the remaining servers finishes a sub-task of the job [100]. We note that at any stage s, the maximum possible service rate for a job that is not finished yet is $(n - s + 1)\mu^{'}$, where $\mu^{'} = \frac{fk\mu}{l}$. This happens when all the remaining sub-tasks of a job are at the head of their queues. Thus, we can enhance the latency performance in each stage s by approximating it with a M/M/1 system with service rate $(n - s + 1)\mu^{'}$. Then,

the average latency of the original system (denoted by T), can be lower bounded as

$$T \geq T_{\text{LB}} = \sum_{i=1}^{k} \frac{1}{(n-i+1)\mu' - \lambda}, \tag{5.13}$$

where T_{LB} denotes the lower bound on the average latency of the F-J system.

5.2.2 Upper Bound on Average Latency

To upper-bound the performance of the (n, k) F-J system, we degrade its performance by approximating it with a (n, k) Split-Merge (SM) system, proposed in [28]. In the (n, k) SM system, after a server finishes a copy of a job, it is blocked and not allowed to accept new jobs until all k copies of the current job are finished. When k copies of a job are finished, the copies of that job at remaining $n - k$ servers exit the system immediately. The SM system thus can be modeled as a M/G/1 system with arrival rate λ and a service distribution that follows k^{th} order statistics [101] and is described here for reference.

Let X_1, X_2, ..., X_n be n i.i.d random variables (r.v.). Now if we order the r.v.'s in ascending order to get, $X_{1,n} < X_{2,n} \cdots < X_{k,n} \cdots < X_{n,n}$, then the distribution of the k^{th} smallest value, $X_{k,n}$, is called the k^{th} order statistics. The pdf of $X_{k,n}$ is given by[6]

$$f_{X_{k,n}}(x) = \binom{n}{k-1, 1, n-k} F_X(x)^{k-1} (1 - F_X(x))^{n-k} f_X(x) \tag{5.14}$$

where $F_X(x)$ and $f_X(x)$ are the cumulative density function and pdf of X_i respectively for all i. The average latency of the F-J system is thus upper-bounded by the average latency for the SM system, T_{SM} as,

$$T \leq T_{\text{SM}} = \underbrace{\text{E}[X_{k,n}]}_{\text{service time}} + \underbrace{\frac{\lambda \left[\text{V}[X_{k,n}] + \text{E}[X_{k,n}]^2\right]}{2(1 - \lambda \text{E}[X_{k,n}]))}}_{\text{waiting time}}, \tag{5.15}$$

where the average service time is simply the expectation, $\text{E}[X_{k,n}]$ and the average waiting time for a M/G/1 system given by the P-K formula in (2.11). Now if X_i is exponential with mean $1/\mu'$ (where $\mu' = \frac{fk\mu}{l}$), then the mean and variance of $X_{k,n}$ are given by,

$$\text{E}[X_{k,n}] = \frac{H^1_{n-k,n}}{\mu'}, \text{V}[X_{k,n}] = \frac{H^2_{n-k,n}}{\mu'^2}, \tag{5.16}$$

[6]The result in (5.14) can be understood as follows. First select groups of $k-1$, 1, and $n-k$ servers out of n servers in $\binom{n}{k-1,1,n-k}$ possible ways. Then the pdf of service time for the singled-out server is simply $f_X(x)$. Now since X_i are i.i.d random variables, the probability that the selected $k-1$ servers finish their jobs before the singled-out server is $F_X(x)^{k-1}$. Similarly, the probability that $n-k$ servers finish their jobs after the singled-out server is $(1 - F_X(x))^{n-k}$.

where $H^z_{x,y}$ is a generalized harmonic number of order z defined by

$$H^z_{x,y} = \sum_{j=x+1}^{y} \frac{1}{j^z}, \tag{5.17}$$

for some positive integers x, y and z.

5.3 Main Results

Section 5.2 presented bounds on the average latency for the (n, k) F-J system. To extend the lower-bound result (5.13) to a heterogeneous FJ system, a naive approach would be to approximate it with a homogenous FJ system with jobs of class i only while evaluating the lower bound on average latency of class i. Thus a naive lower-bound on the average latency for jobs of class i is,

$$T^i_{\text{naive}} \geq \sum_{j=0}^{k_i-1} \frac{1}{(n-j)\mu_i - \lambda_i}. \tag{5.18}$$

This lower bound holds true irrespective of the scheduling policy used in the heterogeneous system. However, this is a loose bound as it ignores the dependency of response time for a job of class i on the jobs of other classes in the system which compete for the service at the same server.

Therefore, through a rigorous latency analysis of various scheduling policies, we next account for this inter-dependency in average latency of different classes and present lower and upper bounds for the heterogeneous FJ system. To this end, we first define a set of variables for a compact presentation of the results. The operational meaning of these variables will become clear when we present the proof of the results.

- (n, k_i) is the MDS code used to store data of class i.
- l_i is the file-size for class i.
- λ_i is the arrival rate for jobs of class i.
- $\mu_i = k_i f \mu / l_i$ is the effective service rate for jobs of class i, where μ is the service rate per unit file size.
- $\rho_i = \frac{\lambda_i}{\mu_i}$ is the server utilization factor for class i.
- $\mathcal{S}_i = \sum_{r=1}^{i} \rho_r H^1_{n-k_r,n}$.

5.3.1 Main Results

Lemma 5.1 gives the stability conditions of the heterogeneous DSS for various scheduling policies. The upper- and lower-bounds on the average latency for various scheduling policies are presented in Theorem 5.1 and 5.2 respectively. The proofs are presented in Appendix A.

Lemma 5.1. *For a $(n, k_1, k_2, ..., k_R)$ Fork-Join system to be stable, the following condition must be satisfied at each node.*

- *FCFS scheduling*

$$\left(\sum_{r=1}^{R} k_r \lambda_r\right)\left(\sum_{r=1}^{R} \frac{\lambda_r l_r}{k_r}\right) < nf\mu \sum_{r=1}^{R} \lambda_r. \tag{5.19}$$

- *Preemptive/Non-preemptive priority scheduling*

$$\sum_{r=1}^{R} \lambda_r l_r < nf\mu. \tag{5.20}$$

Next, to upper-bound the average latency, we extend the Split-Merge (SM) system (defined in Section 5.2.2) to R data classes, keeping the scheduling policy same as that for the original system. Then for a given scheduling policy, the upper-bound on average latency is basically the average latency of the corresponding SM system. This in turn is sum of the average service time and waiting time which can be determined by noting the equivalence between the SM system as a M/G/1 system as described in Section 5.2.2. We thus obtain the following upper-bounds on the average latency for different scheduling policies.

Theorem 5.1. *The average latency for job requests of class i in a $(n, k_1, k_2, ..., k_R)$ Fork-Join system is upper-bounded as follows:*

- *FCFS scheduling*

$$T^i_{\text{FCFS}} \leq \underbrace{\frac{H^1_{n-k_i,n}}{\mu_i}}_{\textit{Service time}} + \underbrace{\frac{\sum_{r=1}^{R} \lambda_r [H^2_{n-k_r,n} + (H^1_{n-k_r,n})^2]/{\mu_r}^2}{2\,(1-\mathcal{S}_R)}}_{\textit{Waiting time}}. \tag{5.21}$$

  The bound is valid only when $\mathcal{S}_R < 1$.

- *Non-preemptive priority scheduling*[7]

$$T^i_{\text{N-PQ}} \leq \frac{H^1_{n-k_i,n}}{\mu_i} + \frac{\sum_{r=1}^{R} \lambda_r [H^2_{n-k_r,n} + (H^1_{n-k_r,n})^2]/{\mu_r}^2}{2\,(1-\mathcal{S}_{i-1})\,(1-\mathcal{S}_i)}. \tag{5.22}$$

  The bound is valid only when $\mathcal{S}_i < 1$.

- *Preemptive priority scheduling*[7]

$$T^i_{\text{PQ}} \leq \frac{H^1_{n-k_i,n}}{\mu_i\,(1-\mathcal{S}_{i-1})} + \frac{\sum_{r=1}^{i} \lambda_r [H^2_{n-k_r,n} + (H^1_{n-k_r,n})^2]/{\mu_r}^2}{2\,(1-\mathcal{S}_{i-1})\,(1-\mathcal{S}_i)}. \tag{5.23}$$

  The bound is valid only when $\mathcal{S}_i < 1$.

We now define an additional set of variables for compact presentation of the results in Theorem 2.

- Without loss of generality, assume the classes are relabeled such that $k_1 \leq k_2 \leq ... \leq k_R$. Then for class i, we define c_s as,

$$c_s = \begin{cases} 0, & 1 \leq s \leq k_1 \\ 1, & k_1 < s \leq k_2 \\ \vdots & \\ i-1, & k_{i-1} < s \leq k_i \end{cases}. \tag{5.24}$$

- At a stage s, let $\mathcal{R}^i_s$ denote the set of classes with priority higher than class i and that have not been finished yet.

- $t_{s,i} = \frac{\lambda_i}{(n-s+1)\mu_i}$ at stage s and class i.

- $\mathcal{Z}^i_s = 1 - \sum_{r \in \mathcal{R}^i_s} t_{s,r}$ at stage s and class i.

For obtaining a lower-bound on the average latency, we enhance the performance of the original system similar to the process described in Section 5.2.2. The processing of a job of class i is modeled as completing k_i sequential stages (or sub-tasks). Then we enhance the latency performance for job of class i in stage s by assuming the maximum possible service rate for it, i.e, $(n-s+1)\mu_i$. However, at stage s, there may also be unfinished sub-tasks of jobs of other classes which can be served with maximum possible service rate of $(n-s+1)\mu_j$, where $j \neq i$. Due to this, we model the performance of each enhanced stage as a M/G/1 system. We thus obtain the following lower-bounds on the average latency for different scheduling policies.

[7]Without loss of generality, we set the classes in the order of decreasing priority as $1 > 2 > \cdots > R$.

Theorem 5.2. *The average latency for job requests of class i in a $(n, k_1, k_2, ..., k_R)$ Fork-Join system is lower-bounded as follows:*

- *FCFS scheduling*

$$T^i_{\text{FCFS}} \geq \sum_{s=1}^{k_i} \left(\underbrace{\frac{t_{s,i}}{\lambda_i}}_{\text{service time}} + \underbrace{\frac{\sum_{r=c_s+1}^{R} \frac{t^2_{s,r}}{\lambda_r}}{1 - \sum_{r=c_s+1}^{R} t_{s,r}}}_{\text{waiting time}} \right). \tag{5.25}$$

- *Non-Preemptive priority scheduling*[7]

$$T^i_{\text{N-PQ}} \geq \sum_{s=1}^{k_i} \left(\frac{t_{s,i}}{\lambda_i} + \frac{\sum_{r=c_s+1}^{R} \frac{t^2_{s,r}}{\lambda_r}}{\mathcal{Z}^i_s \left(\mathcal{Z}^i_s - t_{s,i} \right)} \right). \tag{5.26}$$

- *Preemptive priority scheduling*[7]

$$T^i_{\text{PQ}} \geq \sum_{s=1}^{k_i} \left(\frac{t_{s,i}}{\lambda_i \mathcal{Z}^i_s} + \frac{\sum_{r \in \mathcal{R}^i_s \cup i} \frac{t^2_{s,r}}{\lambda_r}}{\mathcal{Z}^i_s \left(\mathcal{Z}^i_s - t_{s,i} \right)} \right). \tag{5.27}$$

5.4 Quantitative Results and Discussion

In this section, we use Monte-Carlo simulations of a heterogeneous Fork-Join system to study the impact of varying various system parameters on the average latency of different classes and the energy efficiency of DSS. For simplicity, the number of data classes is set to 2. Data of class 1 is stored using $(n, k_1) = (10, 5)$ MDS code. Data of class 2 is stored using $(10, k_2)$ MDS code where k_2 is varied from 1 to 10. Arrival rates for the two classes are set as: $\lambda_1 = 0.15$ and $\lambda_2 = 0.5$. The job size for both the classes is set to 1 kilobits. Job requests for both the classes are served using full redundancy (i.e., $r_1 = r_2 = n$). We set the power consumption parameters by using the data for Intel Xeon family of CPUs[8] and associated platform components [99]. Therefore, we set $C_0 = 203.13$ W, $P_a = 120$ W, $C_l = 15$ W, $w_l = 6$ s, and $P_l = 13.1$ W. The CPU frequency, f is set to 1 unless mentioned otherwise.

[8]We use the power consumption parameters of "Deeper Sleep" state for our low-power state.

5.4.1 Impact of fault-tolerance and service rate

The behavior of average latency with respect to change in fault-tolerance k, is governed by two opposing factors.

- Increasing k reduces the number of servers available for serving the next job in queue, thus resulting in an increase in latency.
- Increasing k increases the effective service rate ($k\mu$) of each server as each server stores a smaller fraction ($\frac{l}{k}$) of the job. This decreases the average latency.

Fig. 5.6 shows the average latency for jobs of class 2 versus k_2 for the FCFS system with $\mu = 1/6$ and 1[9]. The file size for both classes are equal to 1 kb. We note that the average latency increases on increasing k_2. This is because μ is large enough, so the increment in latency due to the first factor dominates the decrease in latency due to the second factor. We also note that the bounds are somewhat loose at high values of k_2 and low values of μ. In particular, the lower bound becomes loose because at each processing stage of the serial Fork-Join system, the difference between the actual service rate and its bound at s^{th} stage of processing (i.e. $(n - s + 1)\mu_i$) for jobs of class i increases with increase in k and decrease in μ. Similarly the upper bound becomes worse because the service time lost due to blocking increases significantly at low μ and high k values. This is because the remaining sub-tasks are served really slow (low μ) and the blocking continues until a large number of sub-tasks (high k) are finished. Finally, as expected, we note that the naive lower bound on latency of class 2 is loose as compared to the proposed lower bound for the FCFS system.

5.4.2 Impact of coding on energy efficiency and storage space

Fig. 5.7 illustrates the impact of varying the code rate (k/n for (n, k) MDS code) on the latency, energy efficiency and network bandwidth of system. At one extreme is the $(n, 1)$ replication code with code rate $1/n$ that has minimum latency (see Fig. 5.6) and maximum energy efficiency. This is because we just wait for any one server to finish the job. For a fixed n, low latency translates to higher data throughput and hence higher energy efficiency. However, the total storage space per file for $(n, 1)$ code is nl where l is file size. Hence the (write) network bandwidth is maximum at $k = 1$. At the other extreme is (n, n) code with no fault-tolerance and no storage overhead (storage size is l). But it suffers from high latency and low energy efficiency. This is because we need to wait for all the servers to finish their sub-tasks of a job before the job is completed. Hence latency and throughput suffers which in turn decreases energy efficiency.

[9]In our work, we set file size to be multiples of 1 kilobits. So μ is defined per kilobit.

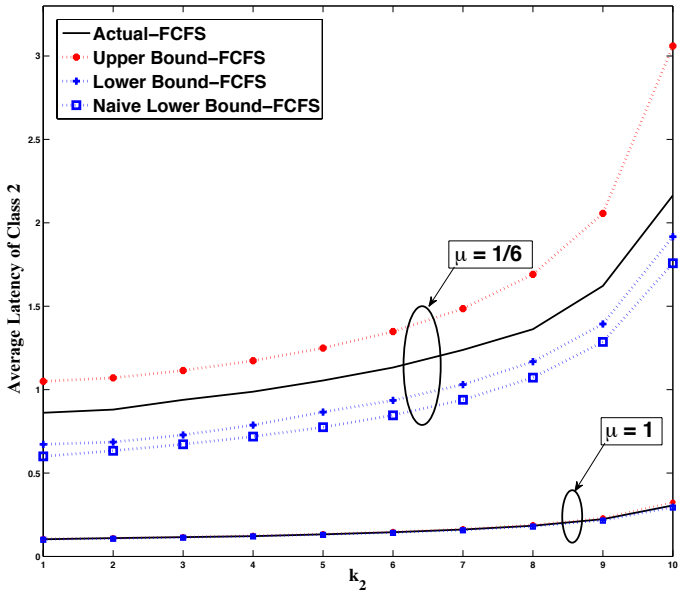


Figure 5.6: Latency of a data-class increases with increase in its code-rate and decreases with increase in service rate.

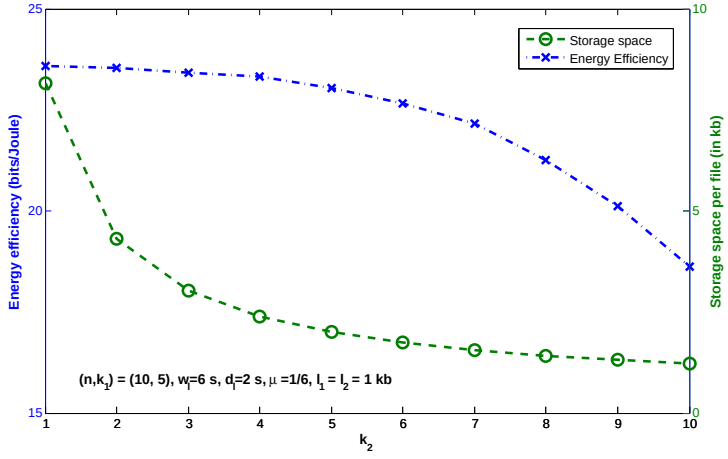


Figure 5.7: Tradeoff between energy efficiency of DSS and storage space per file with variation in code-rate.

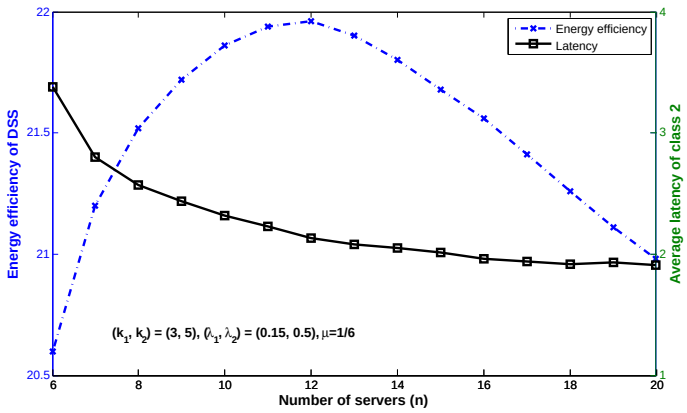


Figure 5.8: Energy efficiency increases and attains a maxima as number of servers is increased while latency behaves in an inverse fashion.

5.4.3 Impact of number of servers in DSS

Fig. 5.8 shows the impact of increasing the number of servers (n) on the latency and energy efficiency of DSS, while all other system parameters are kept constant. We observed that for low values of n, increasing n increases the energy efficiency. This is because of more servers available to serve the job which reduces average latency and thus increase the throughput. The increase in throughput due to lower latency outweighs the increase in energy consumption due to higher n. Hence the overall effect is that energy efficiency increases. However at high values of n, increasing n results in diminishing returns in latency and throughput. This is because latency improvement is limited by effective service rate ($k\mu/l$) and not the number of servers. At very large n, the energy consumption becomes quite significant. Therefore, the energy efficiency begins to decrease at large n. We thus conclude that **there is an optimum value of n that maximizes energy efficiency and has near minimal latency**.

5.4.4 Impact of general service time

In most of the practical DSS, the service times are not exponentially distributed but rather have heavy-tail which means that there is a significant probability of very large service times. Pareto distribution has been found to be a good fit for service time distribution in practical DSS [102, 103]. Its cumulative distribution function is given by

$$F_S(s) = \begin{cases} 0 & \text{for } s < s_m \\ 1 - \left(\frac{s_m}{s}\right)^{\alpha} & \text{for } s \geq s_m \end{cases} \tag{5.28}$$

Here α is shape parameter and x_m is the scale parameter. As the value of α decreases the service becomes more heavy-tailed and it becomes infinite for $\alpha \leq 1$. Figures 5.9 and 5.10

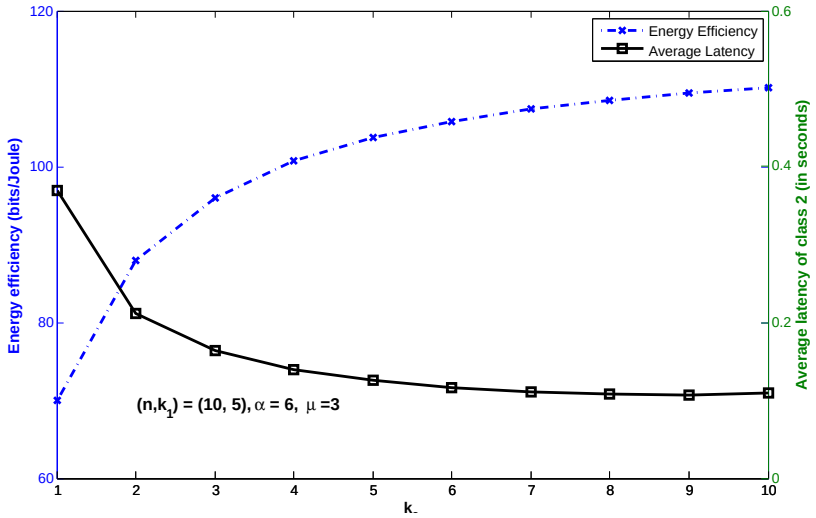


Figure 5.9: A light tailed general service distribution (Pareto-distribution with $\alpha = 6$) results in monotonically decreasing latency as a function of code-rate. Energy efficiency follows an inverse behavior.

show the impact of Pareto service distribution on the latency and energy efficiency of DSS for $\alpha = 1.1$ and 6 respectively. At $\alpha = 6$, the service distribution is not very heavy-tailed. So increasing k_2 reduces latency of jobs of class 2 due to increase in their effective service rate $(k_2\mu f/l_2)$. However, at $\alpha = 1.1$, the service time distribution becomes very heavy-tailed, so as k_2 becomes large, the increase in service time due to waiting for more servers (larger k) outweighs the decrease due to higher effective service rate. In both cases, we note that latency behaves inversely to the change in latency. We note that as k_2 increases from 1 to 10, energy efficiency first starts increasing, reaches a maximum and then starts decreasing for large k. We conclude that **for heavy-tailed service distribution, there exists an optimal code-rate that yield maximum energy efficiency and minimum latency for heavy-tailed service times**.

5.4.5 Impact of heavy-tailed arrival distribution

Fig. 5.11 illustrates the impact of a general (Pareto) arrival time distribution on the latency and energy efficiency of DSS. We observed that when distribution becomes heavy tailed, latency increases (and energy efficiency decreases) with increase in code rate. The heavy-tailed arrival distribution results in occasional very large inter-arrival time, however the arrival rate remains the same. Since it does not influence significantly the service dynamics, we observe that the latency increases with increase in code-rate similar to the M/M/1 case (in Fig. 5.6). Since latency increases, energy efficiency decreases with increase code-rate similar to previous results.

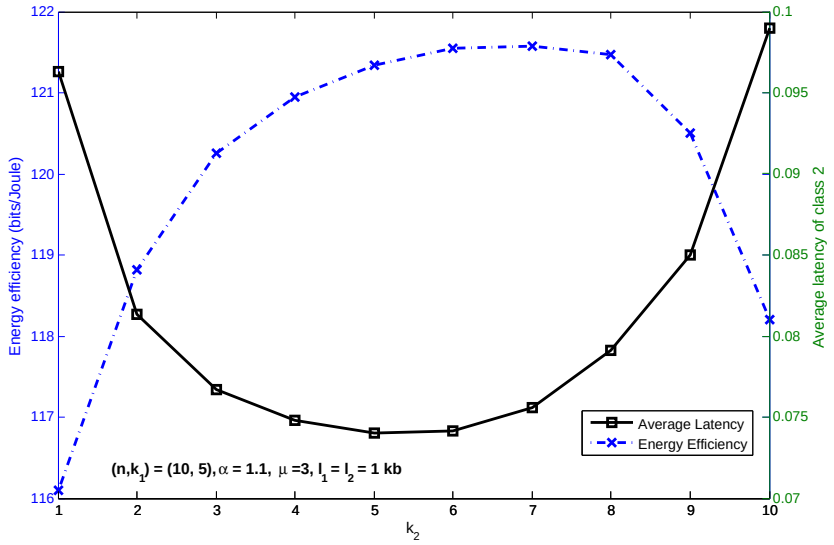


Figure 5.10: A heavy tailed general service distribution (Pareto-distribution with $\alpha = 1.1$) results in minimal latency and maximum energy efficiency point as the code-rate is increased.

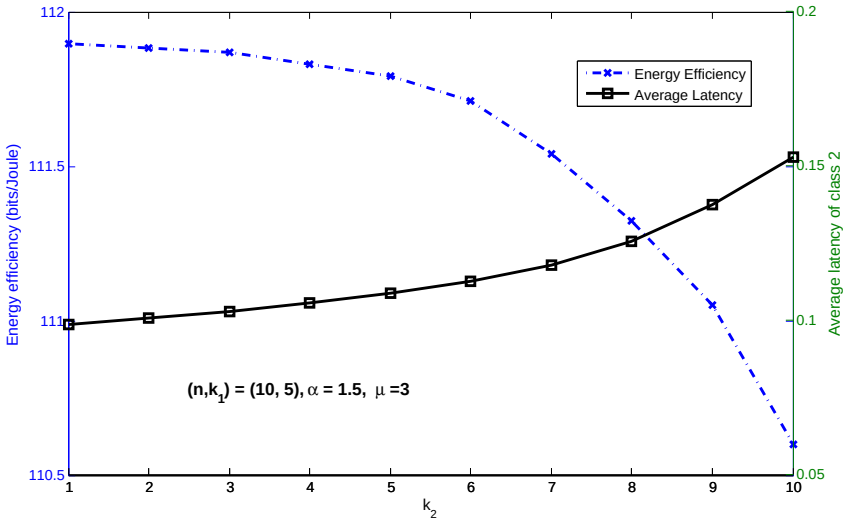


Figure 5.11: A heavy tailed inter-arrival time distribution (Pareto-distribution with $\alpha = 1.5$) results in monotonically increasing latency (and monotonically decreasing energy efficiency) as the code-rate is increased.

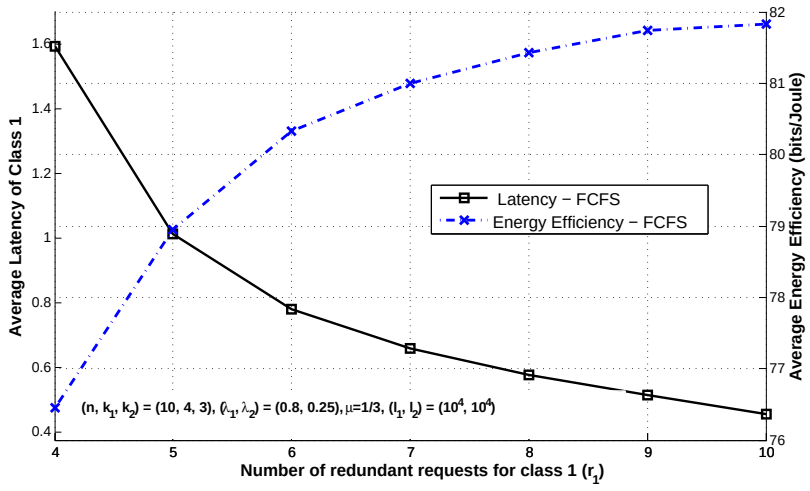


Figure 5.12: Sending redundant requests reduces latency and improves energy efficiency.

5.4.6 Impact of number of redundant requests

We now explore the impact of varying the number of redundant requests (i.e., sending job requests to more than k servers) on the average latency and energy efficiency of DSS. The behavior of latency is governed by two opposing factors.

- Increasing the number of redundant requests reduces the service time because there are more servers available that simultaneously process the same job. This reduces the service time of each job. It increases the energy efficiency because the servers can process more requests per unit time.
- On the other hand, increasing the number of redundant requests reduces the number of servers available for serving the next job in queue, thus resulting in increase of size of queue at the servers. This results in loss of throughput and hence a plausible decrease in energy efficiency.

As it turns out that the first factor is more dominant than the second one, thereby resulting in an overall reduction in latency (increase in energy efficiency) by increasing the number of redundant requests. This behavior can be observed in Fig. 5.12 which shows the average latency of class 1 and energy efficiency of DSS for FCFS scheduling. In this figure, the redundancy for class 1, r_1, is varied from 4 to 10 and the redundancy of class 2 is set to $r_2 = 10$.

5.5 Conclusions

In this chapter, we proposed a novel multi-tenant DSS model and analyzed the energy efficiency of the system via lens of system latency. In the proposed heterogeneous DSS, each data class can possibly have different job arrival rate, job size and its data can be stored with a different fault-tolerance requirement by coding it with appropriate (n, k) MDS code. In order to evaluate the impact of various parameters of DSS on its energy efficiency, we defined a *data throughput* based energy efficiency metric for any given scheduling policy. We analytically established that the energy efficiency of DSS is inversely related to the system latency for a special case. This motivated us to further investigate the impact of various parameters on the relationship between the latency and energy efficiency of the DSS. Therefore, using a queuing-theoretic approach, we obtained bounds on the average latency for FCFS, preemptive and non-preemptive priority queuing policies. We verified the accuracy of the bounds for different settings of system parameters. The bounds, in general, are tight at high values of service rate, μ and low values of k. We also noted that the proposed lower bounds are tighter than a naive lower bound that follows directly from the work in [28].

Using simulations, we investigate the relationship between average latency of data classes and energy efficiency of DSS under various setting of system parameters. We found that increasing the coding rate reduces the network bandwidth but increases latency and decreases energy efficiency. We also found that there exists an optimal number of servers which maximizes energy efficiency and results in near minimal latency. We observed that for heavy-tailed service distribution (which is the case for practical DSS), there exists an optimal code-rate that yield maximum energy efficiency and minimum latency. Lastly, we studied the impact of sending redundant requests on the average latency of that data class and the energy efficiency of DSS. We noted that increasing redundancy for a data class helps to reduce its average latency and as a consequence, the overall latency decreases and energy efficiency of DSS increases.

Chapter 6

Delay-Efficient Packet Scheduler for M2M Uplink

Machine-to-Machine (M2M) communications is becoming increasingly ubiquitous due to its vast number of residential and industrial applications such as in smart homes, vehicle tracking, industrial automation etc. Since M2M applications provides intelligent services by processing the sensory data to monitor and control the nodes, the M2M uplink is more demanding than the M2M downlink and thus attracted more attention from the M2M research community. In most of the M2M applications, the uplink traffic generated from the sensors is delay-heterogeneous and can be classified as either non real-time (no deadline for task completion) or *soft* real-time (decreased utility if deadline not met) or *firm* (zero utility if deadline not met). For instance, the messages from a instrumented protective system have *firm* real-time service requirements whereas preventive maintenance applications are typically served in non real-time. With increase in network size, the available computational and communication resources are shared among a large number of sensors making it even more harder to provide real-time service [37]. This necessitates the need for designing delay-efficient packet scheduler for M2M uplink and that preferably are agnostic to the M2M application, communication standard and hardware-software architecture.

Most of the existing M2M packet schedulers are designed for specific wireless technology such as LTE (see [38] and references therein) and do not completely characterize the heterogeneity in M2M uplink traffic. Another line of work considers the design of packet schedulers for real-time control applications (see [40] and references therein) that guarantee the schedulability of critical tasks while providing best-effort service for others. The limitation of these works is that they assume the relative deadline for periodic tasks is equal to their periods, all aperiodic tasks are aggregated into a single class and the minimum inter-arrival time of an aperiodic task equals its deadline.

In this chapter, we develop a delay-efficient mutliclass packet scheduler for M2M uplink that overcomes some of the limitations of the existing packet schedulers. To begin with, we use

source M2M traffic model in [45] to classify the data from sensors as Periodic Updates (PU) or Event Driven (ED). The PU arrivals are periodic with *firm* service deadline while the ED arrivals are random with *firm*/*soft* real-time/non real-time service requirements. We map the contrasting delay requirements of PU and ED packets onto step and sigmoidal utility functions respectively. The goal of proposed scheduler is to maximally satisfy the delay requirement of all classes which translates to maximizing a proportionally-fair system utility metric. This is achieved by exploiting the heterogeneity in the utility functions of different traffic classes. Specifically, the step utility function for PU packets allow us to serve the ED packets while ensuring that the PU packets meet their deadline. This results in a higher utility for the ED packets without compromising the utility for PU packets.

Additionally we remove PU packets that fail to meet their deadline. This reduce network congestion and improve overall system utility. Furthermore, for critical M2M applications, successive PU failures are highly detrimental to system performance. Hence, we penalize successive PU failures more to minimize their occurrence. Using extensive simulations, we show that the proposed scheduler has a superior delay-performance compared to popular scheduling policies such as First-Come-First-Serve (FCFS), Earliest Due Date (EDD), fixed priority scheduler etc. Another salient feature of the proposed scheduler is that it is general enough to be used across different M2M applications, M2M communication standards and hardware-software M2M architectures.

The rest of this chapter is organized as follows. Section 6.1 introduces the system model, defines the PU, ED utility functions and introduces a proportionally fair system utility metric. Then in Section 6.2, we formulate the system utility maximization problem and in Section 6.3 describe the proposed packet scheduling heuristic. Section 6.4 presents simulation results. Finally Section 6.5 draws some conclusions.

6.1 System Model

Fig 6.1 shows the system model for a heterogeneous M2M uplink from N sensors to a M2M application server (AS) via multiple M2M Aggregators (MAs). The aggregated M2M traffic at AS is classified into R_{pu} PU and R_{ed} ED classes. Here, we assume that there exists multiple sensors with similar arrival period and service deadlines for PU packets and therefore the PU traffic at AS can be clustered into R_{pu} classes. Similarly we cluster the aggregated ED traffic into R_{ed} classes based on the closeness in the characteristics of ED traffic. The period and relative deadline for i^{th} PU class are T^i_{pu} and l^i_d respectively. The ED arrivals is modeled as a Poisson process with rate λ^i_{ed} for i^{th} class.

The service times for i^{th} PU and ED class are assumed to be exponential with rates μ^i_{pu} and μ^i_{ed} respectively. In general, the total time spent by a packet in the system, T, can be written as sum of following components,

$$T = T_{\text{trans}} + T_{\text{prop}} + T_{\text{cong}} + T_{\text{queue}} + T_{\text{ser}}, \tag{6.1}$$

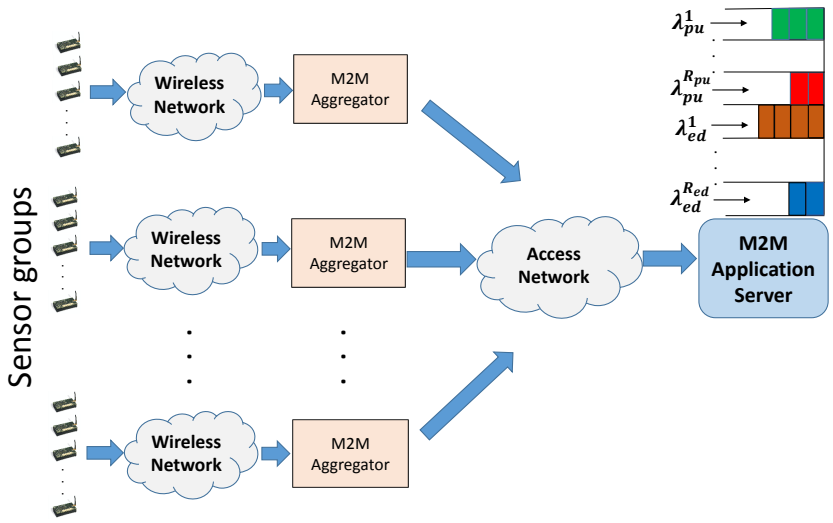


Figure 6.1: System model illustrating queuing process at M2M AS.

which denote the following component delays:

- T_{trans}: Packet transmission time at the sensors and MAs.
- T_{prop}: Propagation delay from the sensors to MAs and from MAs to M2M AS.
- T_{cong}: Congestion delay due to shared wireless channel in large-scale sensor network.
- T_{queue}: Queuing delay at the MAs and M2M AS.
- T_{ser}: Processing time for the packet at the M2M AS.

In this work, we ignore all the terms except for the queuing and service delay at the M2M AS. Due to relatively small payload for M2M traffic, the packet transmission time are small relative to queuing delay and thus can be ignored. We assume sufficient number of wireless resources for both access and core network to ignore the congestion delay. Now the scheduling policy at the M2M AS should be chosen as to maximally satisfy the delay requirements of all of the PU and ED classes.

6.2 Problem Formulation

6.2.1 Utility Functions

PU utility

As stated previously, the PU packets need to be processed within a prespecified time interval at the end of which there is no utility in serving the packet. For a PU packet of i^{th} class with latency l^i_{pu}, we define the utility function (see Fig. 6.2) as,

$$U^i_{\text{pu}}(l^i_{\text{pu}}) = \begin{cases} 1 & \text{if } l^i_{pu} < l^i_d \\ 0 & \text{if } l^i_{pu} \geq l^i_d, \end{cases} \tag{6.2}$$

where l^i_d is the latency deadline at which utility drops to 0.

We next define a penalty function to limit the number of successive PU failures. Let r_i denote the run-length[1] of a sequence of PU failures for i^{th} class, then the penalty for the run (besides the utility loss for each PU failure) is,

$$P^i_{\text{pu}}(r_i) = r_i - {r_i}^{\gamma_i},\ r_i \geq 1 \tag{6.3}$$

where $\gamma_i \geq 1$ indicates the severity of the penalty for i^{th} class and is chosen based on the criticality of the application. For instance, consider the smart-grid uplink traffic which besides other messages, consists of periodic real-time pricing (RTP) updates and periodic meter readings [36]. The RTP updates are highly critical messages and failure to process them within a particular time can potentially result in significant economic loss [104]. On the contrary, failure to process periodic meter readings in time is not a big concern. It can easily be estimated using the meter reading of last successfully received (at the M2M application server) meter reading and any differences in the estimate and actual reading can be adjusted in future billing cycle. Therefore, the penalty (γ_i) for RTP failures should be higher than that for meter readings. The special case of $\gamma_i = 1$ corresponds to when there is no penalty for successive PU failures.

ED utility

Based on the M2M application, the delay requirements of ED traffic may vary from *firm* to non real-time service requirements. Sigmodial functions are fairly versatile to model diverse Quality-of-Service requirements by appropriately varying their parameters [105,106].

[1] A run is a maximal length sequence of identical outcomes within a larger sequence. For example, let [$SSFFSFSFFFS$] denote a sequence of PU success (S) and failure (F). Here, we have three runs of PU failures with length r_i equal to 1, 2 and 3 respectively.

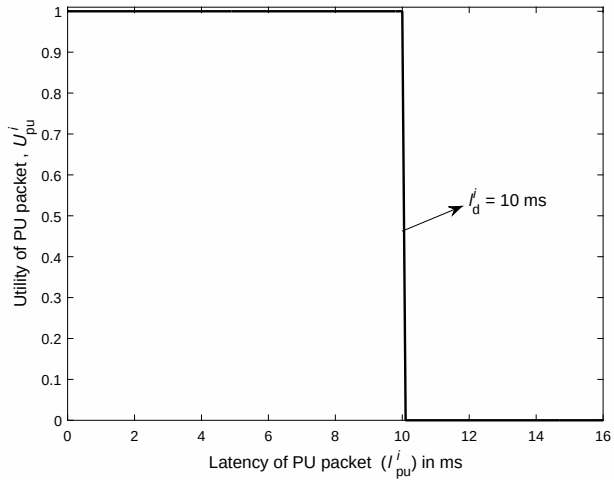


Figure 6.2: Utility function for PU packet of class i.

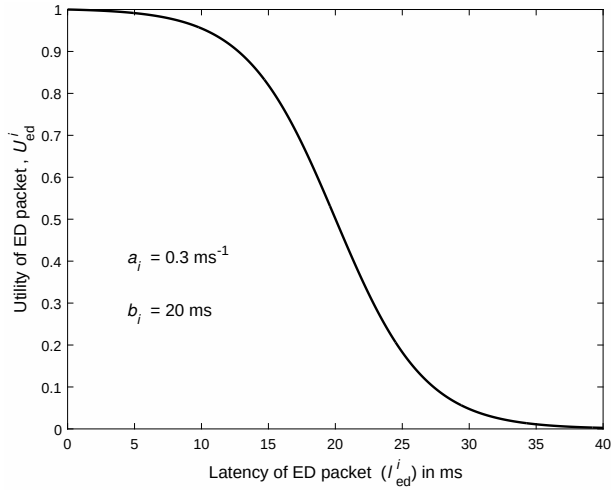


Figure 6.3: Utility function for ED packet of class i.

Therefore, we use sigmoidal function to model the utility of i^{th} ED class (see Fig. 6.3) as,

$$U^i_{\text{ed}}(l^i_{\text{ed}}) = 1 - c_i \left(\frac{1}{1 + \text{e}^{-a_i(l^i_{\text{ed}} - b_i)}} - d_i \right), \tag{6.4}$$

where, $c_i = \frac{1+\text{e}^{a_i b_i}}{\text{e}^{a_i b_i}}$ and $d_i = \frac{1}{1+\text{e}^{a_i b_i}}$. The parameter a_i is the utility roll-off factor that signifies the criticality of the application while the parameter b_i roughly indicates a soft latency deadline.

6.2.2 System utility function

For a given scheduling policy $\mathcal{P}$, let $\hat{U}^i_{\text{pu}}(\mathcal{P})$ and $\hat{U}^i_{\text{ed}}(\mathcal{P})$ denote the average utility of i^{th} PU and ED class respectively, as given by,

$$\hat{U}^i_{\text{pu}}(\mathcal{P}) = \lim_{T_s \to \infty} \frac{\sum_{j=1}^{M^i_{\text{pu}}(T_s)} U^i_{\text{pu}}(l^{ij}_{\text{pu}}) + \sum_{j=1}^{R_i(T_s)} P^i_{\text{pu}}(r_{ij})}{M^i_{\text{pu}}(T_s)}, \tag{6.5}$$

$$\hat{U}^i_{\text{ed}}(\mathcal{P}) = \lim_{T_s \to \infty} \frac{\sum_{j=1}^{M^i_{\text{ed}}(T_s)} U^i_{\text{ed}}(l^{ij}_{\text{ed}})}{M^i_{\text{ed}}(T_s)}, \tag{6.6}$$

where $M^i_{\text{pu}}(T_s)$ and $M^i_{\text{ed}}(T_s)$ are the number of packets from i^{th} PU and ED class served in time T_s. The variable r_{ij} denotes the length of j^{th} sequence or run of PU failures for class i and $R_i(T_s)$ denotes the number of PU failure sequences in time T_s. Lastly, l^{ij}_{pu} and l^{ij}_{ed} denote the latency of the j^{th} packet i^{th} PU and ED class respectively.

We next aggregate the average PU and ED utilities into the following proportionally-fair system utility metric.

$$V(\mathcal{P}) = \prod_{i=1}^{R_{\text{pu}}} (\hat{U}^i_{\text{pu}})^{\beta^i_{\text{pu}}}(\mathcal{P}) * \prod_{i=1}^{R_{\text{ed}}} (\hat{U}^i_{\text{ed}})^{\beta^i_{\text{ed}}}(\mathcal{P}), \tag{6.7}$$

where the parameters β^i_{pu} and β^i_{ed} denote the relative priority assigned to i^{th} PU and ED class respectively.

6.2.3 Delay-optimal scheduling policy

Our objective is to determine the optimal scheduling policy $\mathcal{P}$ at M2M AS that maximizes the proposed system utility metric, $V(\mathcal{P})$. If the arrival time and service times of packets of each PU and ED class are known *apriori*, then the optimal scheme is to process packets of i^{th} PU class such that their service is completed exactly at their deadline. However, due to random ED arrivals and random service times, it is not possible to determine an online optimal scheduler [107]. Therefore, we propose an online heuristic scheduler with a goal to maximize system utility, as described in the next section.

6.3 Proposed Heuristic Packet Scheduler

The proposed scheduler dynamically schedules packets of different classes using delay-efficient heuristics that aim to maximize the proposed system utility metric. Therefore, the backlogged packets at the AS may preempt the packet in service depending on the following scheduling heuristics.

6.3.1 Service order between PU and ED classes

From the PU utility function in Fig. 6.2, it is clear that there is no added utility of processing it before the deadline l_d^i. On the contrary, the ED utility decreases monotonically with increase in latency. Therefore, the proposed scheduler should prioritize service to backlogged ED packets until the delay for PU packet of class i exceeds certain delay threshold $l_t^i (l_t^i < l_d^i)$. Note that, due to randomness in arrival and service times of packets, it is not possible to guarantee that all PU packets will meet their deadline. However, the value of l_t^i is chosen such that the majority of PU packets meet their deadline.

6.3.2 Service order among PU classes

For resolving service contention between packets of different PU classes, we give higher (preemptive) priority to the PU class with higher penalty factor γ_i and lower service rate μ_{pu}^i. This is because higher γ_i for PU class indicates that it is less robust to PU failures and therefore, should be served first. We prioritize service to PU class with lower service rate (i.e., higher average service time) because it is more likely to be unsuccessful if its service is delayed. Again due to the *firm* utility function for PU classes, if there are backlogged ED packets, then the priority order between two PU classes comes into effect only when packets of each PU class exceeds its corresponding latency threshold.

6.3.3 Service order among ED classes

We now discuss how the proposed scheduler determines the order of service between packets of different ED classes. We assign higher priority to ED classes that are more delay sensitive (i.e., have higher a_i and/or lower b_i). Unlike the earlier heuristics, the (preemptive) priority order among ED classes comes into effect immediately at the arrival of ED packet. However, if the latency of i^{th} ED class exceeds a certain threshold δ_i while there are backlogged packet of lower priority ED classes, then its gets preempted by the ED class with the highest priority among lower priority (backlogged) ED classes. By definition, the threshold $\delta_i = \infty$ for ED class with least priority.

The preemption policy for ED classes exceeding their threshold δ_i ensures that ED packets with unusually large service time do not hog the server which will greatly increase the latency for other ED classes. Thus, the proposed heuristic aims to reduce the latency of delay-sensitive ED class without incurring very large latency for delay-tolerant classes.

6.3.4 Service order among packets of same class

We serve the packets of a given PU or ED class using FCFS policy. Although the Shortest Processing Time (SPT) algorithm is optimal for minimizing the average latency [108], we cannot implement it here because we assume random service time for packets and hence this information is not known *apriori* to the scheduler. Since the average service time is same for all packets of a particular class, FCFS is the best policy in this case.

6.3.5 Optimization search space

For the proposed scheduler to maximize the system utility, we need to determine the jointly optimal value of PU and ED thresholds i.e., l_t^i, δ_i. However, the computational complexity for this optimization may be huge if we perform a brute force search over all permissible values of thresholds l_t^i and δ_i, particularly for large number of PU and ED classes. Therefore, we next present the heuristics to reduce the search space for optimization, without sacrificing much on the delay-performance.

Search space for the PU threshold: The unconstrained search space for the PU threshold for class i is $l_t^i \in [0, l_d^i]$. We have assumed the service time for PU packets of class i to be exponentially distributed with rate μ_{pu}^i. So, the 99 percentile for service time is roughly $t_{99}^i = 4.6/\mu_{\text{pu}}^i$. Therefore, we reduce the search space for class i to $l_t^i \in [l_d^i - t_{99}^i, l_d^i]$ without significantly affecting its delay-performance. The computational savings are significant for PU class with large deadline (l_d^i) and high service rate (μ_{pu}^i).

Search space for the ED threshold: The unconstrained search space for the i^{th} ED class threshold is $\delta_t^i \in [0, \infty]$. Let $l_{\text{ed},x}^i$ denote the latency at which the utility of packet of i^{th} ED class equals x and can be easily determined using (6.4). We set the upper bound on δ_t^i to $l_{\text{ed},0.01}^i$. Since a class i packet on average takes $t_{\text{av}}^i = 1/\mu_{\text{pu}}^i$ time at server, a reasonable lower bound for δ_t^i is $\min(0, l_{\text{ed},0.99}^i - t_{\text{av}}^i)$. Hence, the reduced search space for class i is $\delta_t^i \in [\min(0, l_{\text{ed},0.99}^i - t_{\text{av}}^i), l_{\text{ed},0.01}^i]$.

Lastly, the proposed scheduler drops the backlogged PU packets that have exceeded their latency deadline as there utility drops to 0. However, this helps clear congestion and thus reduce latency for other backlogged packets. The proposed scheduler is described in detail in Algorithm 3.

6.4 Simulation Results

In this section, we use Monte-Carlo simulations to evaluate the delay performance of our scheduler against various standard scheduling policies such as FCFS, EDD, Preemptive priority scheduling. The simulation time T_s is set to a large value (40 s) to ensure a steady-state

Algorithm 3 Proposed Mutliclass Packet Scheduler

Begin Algorithm:

while there are backlogged packets **do**
Remove the failed PU packets.
Find classes with packet delay exceeding threshold.
if some PU classes exceed their threshold **then**
Serve the PU class with highest priority among PU class that exceed their threshold.
else if no PU or ED class exceed its threshold **then**
Serve the ED class with overall highest priority.
else if no PU but some ED classes exceed their threshold **then**
Serve the ED class with highest priority among ED classes that do not exceed their threshold.
else
Continue processing the current packet in server.
end if
end while

queuing behavior. The number of sensors is set to $N = 500$ and all classes are assigned equal priority (i.e. equal β_i), unless mentioned otherwise. Although, the proposed scheduler is designed for any number of PU and ED classes, here we consider two toy cases with $R_{\text{pu}} = R_{\text{ed}} = 1$ and $R_{\text{pu}} = R_{\text{ed}} = 2$ to gain practical insights about its performance with respect to existing schedulers.

6.4.1 Toy Case 1: One PU and One ED class

We set the PU and ED arrival period as $T_{\text{pu}} = 500$ ms and $\lambda_{\text{ed}} = 0.0068$ packet/ms respectively for each of the N sensors. The service rate for PU and ED class are 1 packet/ms and 0.5 packet/ms respectively. The latency deadline for PU is $l_d = 10$ ms. For ED utility function, we set $a = 1$/ms and $b = 20$ ms unless mentioned otherwise.

Impact of latency threshold and ED utility parameter, b

Fig. 6.4 illustrates the impact of varying the PU latency threshold l_t in proposed scheduler (without PU drop) for different values of parameter b. We note that for a given b, there exists an optimal value of PU threshold that maximizes the system utility. As b becomes large, the ED packets becomes more delay tolerant and prioritizing them over PU traffic does not increase its utility appreciably. Also since the PU utility is fairly constant at low values of threshold, the the system utility does not change appreciably at high b and low l_t.

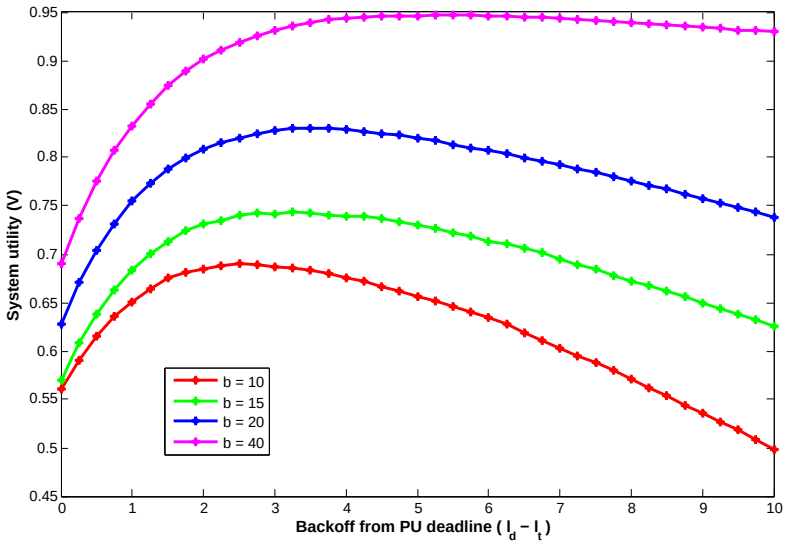


Figure 6.4: Selection of the optimal PU latency threshold l_t as a function of b.

Impact of ED utility parameter, a

Fig. 6.5 illustrates the dynamics of system utility as the ED utility parameter a is varied. As a is increased, ED utility function more brick-walled with the step at latency of b. We observe that the system utility is maximum for all the scheduler when a is close to 0 because the ED utility function is highly delay-tolerant with utility constant at 1. As a is increased, the utility function is no longer constant and therefore the system utility start decreasing and attains a minima. Further increasing a, increases the system utility which eventually saturates. This is because at high a, the ED utility tends to approach a brick-wall function and is approximately constant until the latency becomes equal to b. We also note that, the proposed scheduler performs better than the other scheduler for all values of a. This is again because it intelligently chooses the PU threshold l_t based on a and b and steals time from PU service in order to schedule the ED packet.

Impact of ED arrival rate

Fig. 6.6 shows the impact of increasing the frequency of ED arrivals. Clearly, the proposed scheduler always results in a higher system utility compared to all other schemes. The performance gap widens at large λ_{ed} because with increasing system load, the intelligent utility-based scheduling of PU and ED traffic becomes more critical. Also at large λ_{ed}, the number of PU failures increases and the proposed scheduler with PU drop performs much better since it relieves congestion at the M2M Application Server.

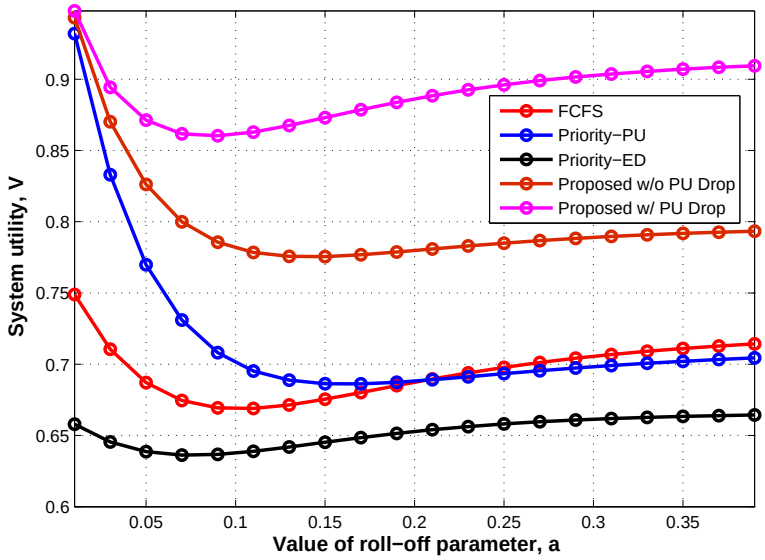


Figure 6.5: Impact of criticality of ED application (set by a) on system utility.

6.4.2 Toy Case 2: Two PU and Two ED classes

We assume that 300 sensors transmit packets of PU class 1 and the remaining 200 transmit packets of PU class 2. For ED traffic, we assume 150 sensors for class 1 and the remaining 350 sensors for class 2. The PU period for each sensor of class 1 and 2 are $T^1_{\text{pu}} = 1.8$ s and $T^2_{\text{pu}} = 0.5$ s respectively. The overall arrival rate for ED class 2 is $\lambda^2_{\text{ed}} = 0.1$ packet/ms whereas λ^1_{ed} is varied from 0.1–0.25 packet/ms. The service rates for packet of all classes is set to 1 packet/ms.

The latency deadline for PU class 1 and 2 are $l^1_d = 4$ ms and $l^2_d = 8$ ms respectively. The utility function parameters for ED class 1 and 2 are set to $[a_1, b_1] = [1, 10]$ and $[a_2, b_2] = [0.7, 20]$ respectively, unless mentioned otherwise. The penalty parameters are set to $\gamma_1 = \gamma_2 = 1$ unless mentioned otherwise. We initially fix the latency threshold for ED class 1 and 2 to $\delta_1 = b_1 + (4/a_1)$ and $\delta_2 = \infty$ respectively and later study the performance improvement by optimizing δ_1 over $[\min(0, l^1_{\text{ed},0.99} - t^1_{\text{av}}), l^1_{\text{ed},0.01}]$, where $t^1_{\text{av}} = 1$ ms in our simulations. For each simulation scenario, we determine the optimal latency threshold for i^{th} PU class by searching over the region $[l^i_d - t^i_{99}, l^i_d]$.

Impact of delay heterogeneity of classes

Fig. 6.7 shows plot of system utility for an approximately homogeneous system with $T^1_{\text{pu}} = T^2_{\text{pu}} = 0.8$ s, $[l^1_d, l^2_d] = [7.8, 8]$ ms, $[a_1, a_2] = [0.65, 7]$ ms^{-1} and $[b_1, b_2] = [19, 20]$ ms. We then make the system more heterogeneous by reverting to the default system parameters. The impact of delay-heterogeneity on delay-performance of the system can be observed in Fig. 6.8.

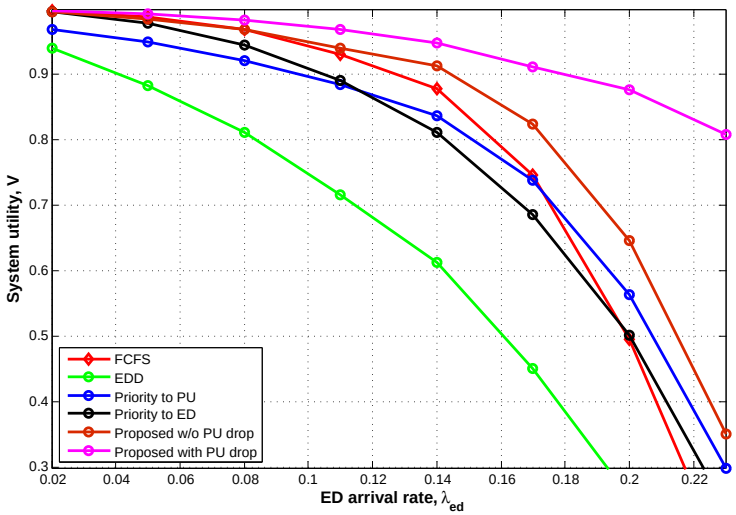


Figure 6.6: Impact of ED arrival rate on the system utility.

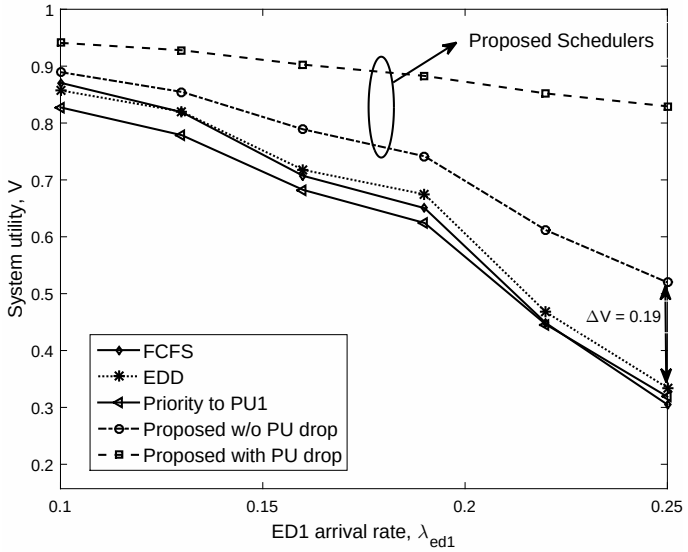


Figure 6.7: System utility with delay-homogeneous M2M traffic.

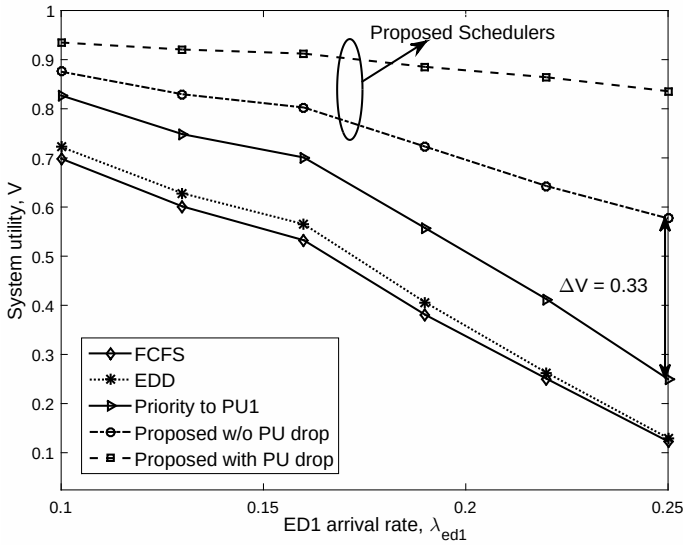


Figure 6.8: System utility with delay-heterogeneous M2M traffic.

We note that the performance gap between the proposed scheduler and the other schemes, ΔV, is much larger for heterogeneous system ($\Delta V = 0.33$) as compared to homogeneous system ($\Delta V = 0.19$). This is because in the proposed scheduler, the role of assigning different priority levels to classes becomes much more important with increase in heterogeneity. Lastly, as expected, dropping the failed PU packets further improves the performance of proposed scheduler due to reduced congestion.

Impact of choice of ED threshold

Fig. 6.9 shows the impact of selecting optimal threshold δ_1 on the system utility. We note that the performance of proposed scheduler is significantly improved ($\Delta V = 0.60$) compared to Fig. 6.8 ($\Delta V = 0.33$) wherein δ_1 was arbitrarily set to $b_1 + (4/a_1)$. This is because selecting optimal[2] δ_1 prevents latency of ED class 2 from growing very large when λ^1_{ed} becomes very large relative to λ^2_{ed}.

Impact of penalizing PU failures

We now consider the impact of penalizing PU failures on the performance improvement of different schedulers. Fig. 6.10 shows the plot of system utility when the penalty factor is small but equal for both PU classes ($\gamma_1 = \gamma_2 = 1.2$). We observe that while the utility for other schedulers take a performance hit and approach 0 at $\lambda_{\text{ed}_1} = 0.25$, the proposed scheduler is fairly robust to the penalization. This is due to multiple degrees of freedom in the form of PU latency thresholds which are re-calibrated to maximize system utility. However, other

[2]Here we refer to optimality over reduced search space for δ_1.

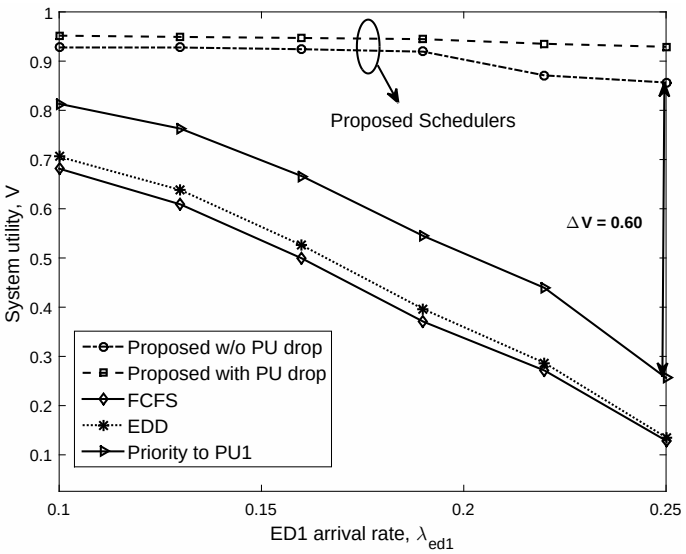


Figure 6.9: Impact of using optimal latency threshold for ED class 1.

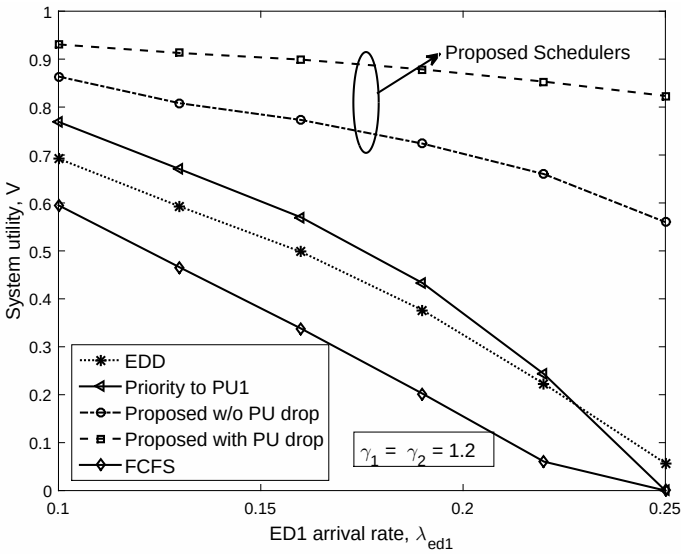


Figure 6.10: Utility performance for small but equal PU failure penalty.

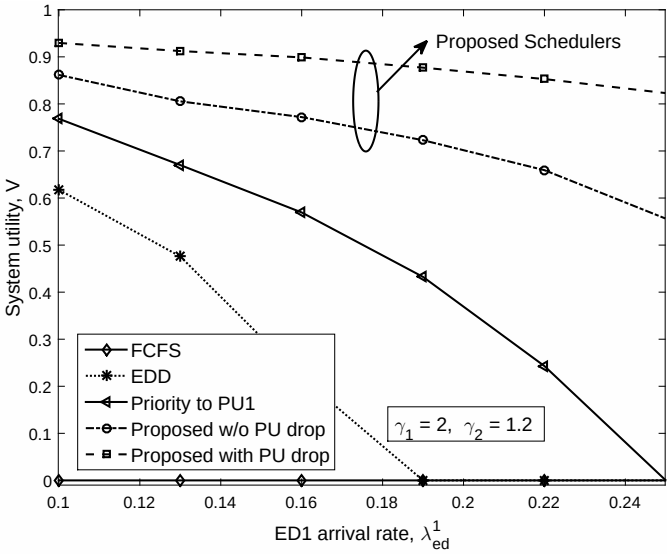


Figure 6.11: Utility performance for unequal PU failure penalty.

schedulers lack this flexibility and hence their performance suffers. Penalizing PU class 1 more, worsens the performance of other schedulers as shown in Fig. 6.11. But due to robustness of the proposed scheduler, its performance does not change appreciably.

6.5 Conclusions

In this chapter, we presented a online delay-efficient multiclass scheduler for delay-heterogeneous M2M uplink traffic. The data from each sensor is classified into PU and ED traffic. Furthermore, due to the delay-heterogeneity across the sensors, the aggregated traffic at M2M application server is classified into multiple PU and multiple ED classes. We use step and sigmoidal functions to represent the different utility of service to PU and ED classes. The average utility of each class is finally aggregated into a proportionally-fair system utility metric. The proposed delay-efficient scheduler uses heuristics that aim to maximize system utility metric. Specifically, it exploits the *firm* utility for PU packets by prioritizing service to ED data as long as we meet the deadline for the PU data. However with increase in network size, an increasingly number of PU packets fail to meet their deadline. We remove the failed PU packets to reduce congestion and improve overall system utility. We also introduce a novel penalty function as part of PU utility to reduce burst PU failures for critical applications. Using extensive simulations, we did a comprehensive delay-performance analysis of the proposed scheduler with respect to other schedulers. We note that the proposed scheduler outperforms other schedulers and is fairly robust to heterogeneity in latency requirements and penalization for PU failures.

Chapter 7

Joint Delay-Optimal MAC and Packet Scheduler for M2M Uplink

The Machine-to-Machine (M2M) market has witnessed a steady exponential growth over the past few years due to its plethora of both commercial and residential use-cases such as in smart healthcare, vehicle tracking, smart home, security and surveillance etc. [109, 110]. The M2M uplink traffic triggered from the sensors exhibits heterogeneity in several dimensions such as maximum packet delay, packet size and arrival rate etc. For example, consider the M2M traffic generated by smart meters in a residential area [35]. As per the UCA OpenSG specification (described in [36]), at one extreme are the firmware and software updates with maximum latency of 5 s, payload less than 50 B and arrivals less than 0.5 messages/day/device. On the other extreme are meter reading messages with high payload of around 1.2 kB, maximum packet delay greater than 60 s and 6 messages/day/device.

Motivated by this problem, in Chapter 6, we presented a novel delay-aware multiclass packet scheduling heuristic at M2M Application Server (AS) wherein we mapped the delay-requirements of classes onto utility function and developed strategies to maximize a proportionally fair system utility metric. We classified the sensory traffic into Periodic Updates (PU) and Event Driven (ED) messages as in [45]. In this chapter, our focus is on developing delay-optimal packet scheduling schemes using a queuing theoretic framework. However, since the periodic arrivals for PU is a non-renewal arrival process [46], it does not lend itself easily for analytical characterization of average packet latency. Therefore, in this chapter, we assume a Poisson arrival process (a renewable process) for each category of sensor traffic which is a fairly good fit for the aggregated traffic at the M2M Aggregators (MAs) [24].

We consider a general M2M network wherein the uplink data from a local group of sensors is first aggregated at a M2M aggregator (MA) and then the data from multiple MAs is finally processed at an Application Server (AS) residing in the M2M core network. The uplink sensor traffic is classified into multiple Quality-of-Service (QoS) classes based on the packet arrival rate, payload size and maximum packet delay requirements of different ap-

plications. We use a general sigmoidal function to map the delay-requirement onto utility function for each class. We then define a delay-optimal scheduler as the one that maximizes a proportionally-fair system utility metric. We note that the average delay for each class under any work-conserving[1] scheduling policy can be realized by appropriately time-sharing between all possible preemptive[2] priority scheduling policies. Therefore, we determine the optimal scheduler by solving for the optimal fraction of time-sharing between different priority scheduling policies that results in maximum system utility. We significantly reduce the computational complexity of determining the optimal scheduler by reformulating and solving the original single-stage optimization problem as an iterative optimization problem. This reduction in complexity is achieved without loosing the delay-optimality of the iterative scheduler.

We then extend our work to the problem of jointly optimal multiclass scheduler at the MAs and AS. To solve it, we first formulate a single-stage convex optimization problem that is solved centrally at AS. We also propose an alternative low-complexity, distributed and iterative optimization approach. However, the distributed optimization requires the per-class arrival process at AS to be Poisson. We next extend our work to the joint problem of channel assignment for MA-AS links and packet scheduler at MAs and AS. To solve it, we first formulate a mixed integer non-linear programming problem that is solved centrally at AS using the Branch and Bound method. We then propose a low-complexity, distributed optimization framework with optimal delay-performance when the service time distribution is exponential.

Using Monte-Carlo simulations, we verify the correctness of the analytical result for proposed optimal scheduler at AS and show that it outperforms other state-of-the-art schedulers such as WRR, max-weight scheduler, WFS and fixed priority scheduling. It also results in a much lower delay jitter for the delay-sensitive traffic which is highly desirable. But this is achieved at the expense of higher delay jitter for delay-tolerant traffic which is usually not a big concern due to its delay-tolerant nature. We then show that the centralized joint MA-AS packet scheduler has superior delay-performance as compared to other schedulers and the distributed optimization scheduler results in optimal (near-optimal) delay-performance for exponential (constant) service distribution.

The proposed M2M packet scheduler is agnostic to the communication standard and the hardware-software architecture used for M2M network. It can also adapt to time-varying characteristics of M2M traffic by either solving the optimization problem on-the-fly or looking up for the optimal scheduler using a Look-Up-Table which can be populated during a training phase prior to the deployment of the proposed scheduler.

The rest of the paper is organized as follows. Section 7.1 introduces the system model for a generic M2M uplink. Then in Section 7.2, we define the utility functions for each

[1] A work-conserving scheduler does not result in server being idle while there are jobs in the queue waiting for service.

[2] Hereafter, we drop the qualifier ‘ preemptive ’for succinctness.

class, formulate both the single-stage and iterative utility maximization problem. We prove the iterative optimization problem is convex, and solve it to obtain the optimal scheduler in Section 7.3. We then present the centralized and distributed optimization approach in Section 7.4 for finding the jointly optimal scheduler at MAs and AS. Then in Section 7.5, we study the problem of joint channel assignment for MA-AS links and packet scheduling at MAs and AS. Section 7.6 presents the simulation results. Finally Section 7.7 draws some conclusions.

7.1 System Model

Fig. 7.1 shows the system model for a general M2M uplink system. The data transmitted from each local group of sensors is aggregated at a M2M aggregator (MA) and classified into R classes based on the packet size and packet delay requirements. We assume that the packet arrival process for class i at k^{th} MA is Poisson with rate λ_{ik} [24]. The data from the set of M MAs is concurrently transmitted to the M2M Application Server (AS) using a set of orthogonal channels[3]. This avoids packet collisions between different MA-AS links due to simultaneous transmission, without significantly increasing the complexity at MAs or AS. Furthermore, since the M2M traffic from multiple sensors is aggregated at each MA, assigning dedicated resources for each MA-AS link would not result in any significant resource wastage.

We assume that the total spectrum available for the MA-AS communication is divided into N orthogonal subcarriers, each of bandwidth W. The link between k^{th} MA and AS is allocated N_k subcarriers, depending on the incoming load and traffic delay-requirements at the MA. We assume that each MA transmits at a nominal power to achieve a certain subcarrier signal-to-noise ratio (SNR) η at the AS under Additive White Gaussian Noise (AWGN) channel conditions. The channel between the MAs and AS is assumed to be independent slow Rayleigh fading since the MAs and AS are stationary nodes. Furthermore, we assume flat fading because of the small bandwidth requirement for the M2M system relative to the microwave carrier frequencies. Then the resultant capacity per subcarrier at k^{th} MA, $C_{s,k}$, is given by Shannon's capacity formula as,

$$C_{s,k} = W \log_2(1 + \|h_k\|^2 \eta) \text{ bits/sec}, \tag{7.1}$$

where h_k is the channel gain at a given instant between the k^{th} MA and the AS. The total capacity for k^{th} MA is simply $C_k = N_k C_{s,k}$ bits/sec.

The packets of each class i from the M MAs are then aggregated at AS resulting in a Poisson process for class i with rate $\lambda_i = \sum_{k=1}^{M} \lambda_{ik}$. The packet size may vary for different applications but we assume that the packet size for all the M2M applications mapped to

[3]For instance, this can be achieved using OFDMA across the MAs to transmit the buffered packets to AS

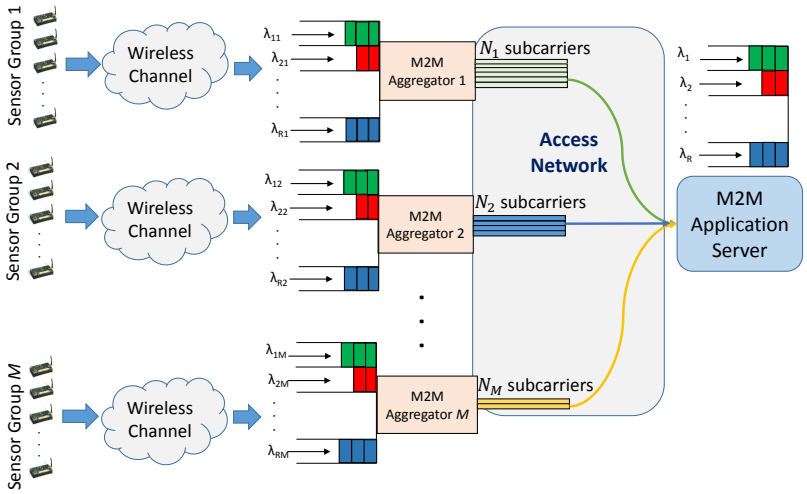


Figure 7.1: System model for a heterogeneous M2M uplink system showing the queuing process at M2M aggregators and the M2M application server. The different packet colors at AS indicate different delay classes.

the same class is approximately same. Therefore, the packet size for class i is modeled as a constant of s_i bits. Then the packet transmission rate for class i at k^{th} MA is $\mu_{ik} = C_k/s_i$ packets/sec. Let C_o denote the service rate at AS, then the the service rate for class i at AS is $\mu_i = C_o/s_i$ packets/sec.

The total time spent by a packet in the system consists of transmission time at sensor and MA, propagation and congestion delay for the sensor-MA link and MA-AS link, queuing delay at MA and AS and lastly, the service time at AS. The propagation delay is usually negligible compared to other delay components and can be safely ignored. The packet transmission time at sensor is also negligible due to small packet size relative to the channel capacity. Furthermore, due to low data rate for traffic at each sensor and small number of sensors in each group, the congestion delay for the sensor-MA link can also be ignored. However, due to traffic aggregation at each MA, the MA-AS link is usually resource constrained. Therefore, we need to account for queuing and service delay at both MA and AS. Since, we assume dedicated orthogonal channels for the MA-AS link, we ignore the congestion delay for the MA-AS link.

We first study the basic packet scheduling problem at AS accounting only for the queuing and service delay at AS. We then extend this analysis to a joint packet scheduler for MAs and AS in Section 7.4. Lastly in Section 7.5, we study the problem of joint subcarrier assignment and packet scheduling at MAs and AS.

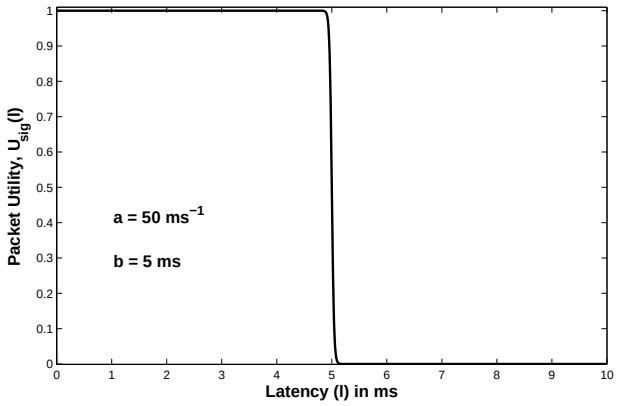


Figure 7.2: Sigmoidal utility function for delay-sensitive traffic.

7.2 Problem Formulation

We first use a generic sigmoidal function [105, 106] to map the latency[4] requirements of i^{th} class, l_i, onto a utility function as,

$$U_i(l_i) = 1 - c_i \left(\frac{1}{1 + \mathrm{e}^{-a_i(l_i - b_i)}} - d_i \right) \tag{7.2}$$

where, $c_i = \frac{1+\mathrm{e}^{a_i b_i}}{\mathrm{e}^{a_i b_i}}$ and $d_i = \frac{1}{1+\mathrm{e}^{a_i b_i}}$. Note that $U_i(0) = 1$ and $U_i(\infty) = 0$. The parameter a_i is the utility roll-off factor and the inflection point for U_i occurs at $l_i = b_i$.

The sigmoidal function is versatile to represent diverse delay requirements by appropriately choosing the parameters a and b. For high a and low b, the utility function becomes 'brick-walled' (see Fig. 7.2) and is a good fit for delay-sensitive applications. On the other hand, at low a and high b, the sigmoidal utility function is a good fit for delay-tolerant applications as shown in Fig. 7.3.

7.2.1 System utility function

For a given scheduling policy $\mathcal{P}$, we define a proportionally fair system utility function as,

$$V(\mathcal{P}) = \prod_{i=1}^{R} \mathbf{U_i}^{\beta_i}(\mathcal{P}), \tag{7.3}$$

[4]We use the terms 'delay' and 'latency' interchangeably. Both refer to the sum of queuing delay and service time.

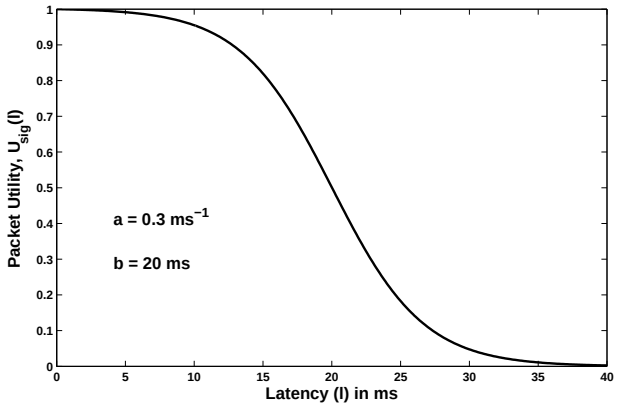


Figure 7.3: Sigmoidal utility function for delay-tolerant traffic.

where $\mathbf{U_i}(\mathcal{P})$ is the average utility of traffic of i^{th} class in the steady state given as,

$$\mathbf{U_i}(\mathcal{P}) = U_i \left(\lim_{T_\text{s} \to \infty} \frac{\sum_{j=1}^{M_i(T_\text{s})} l_i^j(\mathcal{P})}{M_i(T_\text{s})} \right), \tag{7.4}$$

where $M_i(T_\text{s})$ is the number of packets of class i served in time T_s and l_i^j is the latency of the j^{th} packet of i^{th} class. The parameters β_i indicates the relative importance of utility of i^{th} class towards the system utility.

7.2.2 Single-stage Optimization Problem

We now formulate a single-stage optimization (**SSO**) problem to solve for the optimal scheduling policy. We first define the sets $\mathsf{R} = \{1, 2, \cdots, R!\}$ and $\mathcal{R} = \{1, 2, \cdots, R\}$. Assume that in a sufficiently large time interval[5], the AS serves the buffered traffic using j^{th} priority order[6] for γ_j fraction of the time $\forall\ j \in \mathsf{R}$. Then we can write the following equation for the average latency l_i of class i,

$$l_i = \sum_{j=1}^{R!} \gamma_j l_{i,j}^s \ \forall\ i \in \mathcal{R}, \tag{7.5}$$

where $l_{i,j}^s$ represents the average latency of class i using j^{th} priority order in the direct optimization problem.

[5]We assume the time interval under consideration is large enough to observe steady state queuing behavior.

[6]There are $R!$ different priority orders for a set of R classes. For instance if $R = 3$, the different priority orders are $123, 132, 213, 231, 312, 321$, where pqr denotes that class p has higher priority than class q which in turn is greater than priority of class r.

Let $\mathcal{H}_{i,j}$ denote the set of classes with higher priority than class i in the j^{th} priority order. Then using the existing results on latency of M/G/1 preemptive priority queuing systems by Bertsekas et. al. [59], we get,

$$l_{i,j}^{s} = \frac{D_{\mathcal{H}_{i,j},i} + (\rho_{\mathcal{H}_{i,j}} - \rho_i)(1/\mu_i)}{\rho_{\mathcal{H}_{i,j}}(\rho_{\mathcal{H}_{i,j}} - \rho_i)} \ \forall \ i, \tag{7.6}$$

where $\rho_i = \lambda_i/\mu_i$ is the AS utilization factor for class i and $\rho_{\mathcal{H}_{i,j}} = 1 - \sum_{m \in \mathcal{H}_{i,j}} \rho_m$. Since each class is a M/D/1 queue, the residual time $D_{\mathcal{H}_{i,j},i}$ is given by,

$$D_{\mathcal{H}_{i,j},i} = \sum_{k \in \mathcal{H}_{i,j} \cup \{i\}} \frac{\rho_k}{2\mu_k}. \tag{7.7}$$

Since this scheduling policy is characterized by $\gamma = \{\gamma_j : j \in \mathsf{R}\}$, the system utility in (7.3) becomes,

$$V(\gamma) = \prod_{i=1}^{R} \mathbf{U_i}^{\beta_i}(\gamma). \tag{7.8}$$

Now the optimal scheduling policy is determined by solving for γ that maximizes $V(\gamma)$. Exploiting the strictly increasing nature of logarithms, we get the following utility-maximization problem,

$$\begin{aligned} \textbf{SSO:} \max_{\gamma} \quad & \log\big(V(\gamma)\big) = \sum_{i=1}^{R} \beta_i \log\big(U_i(\gamma)\big) \\ \text{s.t.} \quad & \gamma_j \geq 0 \ \forall \ j \in \mathsf{R}, \\ & \sum_{j=1}^{R!} \gamma_j = 1. \end{aligned} \tag{7.9}$$

We note that the optimization is done over $R!$ variables and thus its complexity is enormous even at moderate values of R. For instance, for $R = 7$ the number of variables is 5040.

7.2.3 Iterative Optimization Problem

To reduce the complexity of the above optimization problem, we reformulate it as an iterative optimization (**IO**) problem. Let $\mathcal{A}_r$ be set of any r $(2 \leq r \leq R)$ classes and $\mathcal{A}_r^{\mathsf{c}} = \mathcal{R} \backslash \mathcal{A}_r$, where $A \backslash B$ denotes the set difference $A - B$. We assume that each class in $\mathcal{A}_r^{\mathsf{c}}$ is assigned higher priority than classes in $\mathcal{A}_r$. If AS serves class $i \in \mathcal{A}_r$ with highest priority in $\mathcal{A}_r$ for α_i fraction of the time, then the average latency l_i of class i is given by,

$$l_i = \alpha_i l_{i,i} + \sum_{j \in \mathcal{A}_r, j \neq i} \alpha_j l_{i,j}^{*} \ \forall \ i \in \mathcal{A}_r, \tag{7.10}$$

where $l_{i,i}$ denotes the latency of the class i when it is served with highest priority. $l^*_{i,j}$ denotes the optimal latency for the class i obtained on solving the $r-1$ class system with $\mathcal{A}_{r-1} = \mathcal{A}_r \backslash \{j\}$. Let us define $l_{\mathcal{A}_r} = \{l_i : i \in \mathcal{A}_r\}$ and $l^*_{\mathcal{A}_r \backslash \{j\}} = \{l^*_{i,j} : i \in \mathcal{A}_r, i \neq j\}$.

Now reapplying the result for latency of M/D/1 preemptive priority queuing systems similar to (7.6), we get,

$$l_{i,i} = \frac{D_{\mathcal{A}_r^c,i} + (\rho_{\mathcal{A}_r^c} - \rho_i)(1/\mu_i)}{\rho_{\mathcal{A}_r^c}(\rho_{\mathcal{A}_r^c} - \rho_i)} \ \forall \ i \in \mathcal{A}_r. \tag{7.11}$$

For the special case of $r = 2$ in (7.10), we note that $l^*_{i,j}$ is equivalent to a $l_{i,i}$ in $r = 1$ system with $\mathcal{A}_r = \{i\}$. Hence, we obtain $l^*_{i,j}$ explicitly using (7.11) as,

$$l^*_{i,j} = \frac{D_{\mathcal{A}_r^c,i} + \rho_j/\mu_j + \rho_{\mathcal{R}}/\mu_i}{\rho_{\mathcal{R}}(\rho_{\mathcal{R}} + \rho_i)}, \tag{7.12}$$

Since this scheduling policy is characterized by $\alpha_{\mathcal{A}_r} = \{\alpha_i : i \in \mathcal{A}_r\}$, the system utility in (7.3) becomes,

$$V(\alpha_{\mathcal{A}_r}) = \prod_{i \in \mathcal{A}_r} \mathbf{U_i}^{\beta_i}(\alpha_{\mathcal{A}_r}). \tag{7.13}$$

Now the optimal scheduling policy for a r class system with given $\mathcal{A}_r$, is determined by solving for $\alpha_{\mathcal{A}_r}$ that maximizes $V(\alpha_{\mathcal{A}_r})$. Exploiting the strictly increasing nature of logarithms, we obtain the following r-class optimization subproblem,

$$\begin{aligned} \textbf{IO:} \max_{\alpha_{\mathcal{A}_r}} \quad & \log\big(V(\alpha_{\mathcal{A}_r})\big) = \sum_{i \in \mathcal{A}_r} \beta_i \log\big(U_i(\alpha_{\mathcal{A}_r})\big) \\ \text{s.t.} \quad & \alpha_i \geq 0 \ \forall \ i \in \mathcal{A}_r, \\ & \sum_{i \in \mathcal{A}_r} \alpha_i = 1. \end{aligned} \tag{7.14}$$

Theorem 7.1. *The iterative optimization problem **IO** in* (7.14) *is convex.*

Proof. The proof outline begins with proving that $f^i(\alpha_{\mathcal{A}_r}) = \beta_i \log\big(U_i(\alpha_{\mathcal{A}_r})\big)$ is concave $\forall\ i \in \mathcal{A}_r$ and for all possible $\mathcal{A}_r$. Now the Hessian matrix of f^i at the point $\alpha_{\mathcal{A}_r}$ is defined as,

$$H^i(\alpha_{\mathcal{A}_r}) = \begin{pmatrix} f^i_{11}(\alpha_{\mathcal{A}_r}) & f^i_{12}(\alpha_{\mathcal{A}_r}) & \cdots & f^i_{1r}(\alpha_{\mathcal{A}_r}) \\ f^i_{21}(\alpha_{\mathcal{A}_r}) & f^i_{22}(\alpha_{\mathcal{A}_r}) & \cdots & f^i_{2r}(\alpha_{\mathcal{A}_r}) \\ \vdots & \vdots & \ddots & \vdots \\ f^i_{r1}(\alpha_{\mathcal{A}_r}) & f^i_{r2}(\alpha_{\mathcal{A}_r}) & \cdots & f^i_{rr}(\alpha_{\mathcal{A}_r}) \end{pmatrix},$$

where $f^i_{jk} = \frac{\partial^2 f^i}{\partial \alpha_j \partial \alpha_k}$.

On solving, we get the following result,

$$f^i_{jk}(\alpha_{\mathcal{A}_r}) = \frac{-\beta_i\theta_i a_i^2 \zeta_{i,j}\zeta_{i,k}}{(1+\theta_i)^2}, \tag{7.15}$$

where $\zeta_{i,j} = l^*_{i,j}$ when $i \neq j$ and is $l_{i,i}$ for $i = j$. We define $\theta_i = \mathrm{e}^{-a_i(l_i - b_i)}$.

Therefore, $H^i(\alpha_{\mathcal{A}_r})$ is given by, $H^i(\alpha_{\mathcal{A}_r}) =$

$$\frac{-\beta_i\theta_i a_i^2}{(1+\theta_i)^2}\begin{pmatrix} \zeta^2_{i,(1)} & \zeta_{i,(1)}\zeta_{i,(2)} & \cdots & \zeta_{i,(1)}\zeta_{i,(r)} \\ \zeta_{i,(1)}\zeta_{i,(2)} & \zeta^2_{i,(2)} & \cdots & \zeta_{i,(2)}\zeta_{i,(r)} \\ \vdots & \vdots & \ddots & \vdots \\ \zeta_{i,(1)}\zeta_{i,(r)} & \zeta_{i,(2)}\zeta_{i,(r)} & \cdots & \zeta^2_{i,(r)} \end{pmatrix}, \tag{7.16}$$

where for ease of representation, we defined the shorthand notation $(k) = \mathcal{A}_r(k)$, i.e., the k^{th} element of set $\mathcal{A}_r$. Now to prove that f^i is a concave function, it is sufficient to prove that H^i is a Negative Semi-Definite (NSD) matrix for all $\alpha_{\mathcal{A}_r}$ that satisfies the constraints in (7.14). Let Δ_k denote a k^{th} order principal minor of H^i. Then H^i is NSD if and only if $(-1)^k\Delta_k \geq 0$ for all principal minors of order $k = 1, 2, \cdots, r$.

From (7.16), we obtain the following, $\Delta_1 =$

$$\begin{aligned} &\frac{-\beta_i\theta_i a_i^2\zeta^2_{i,(1)}}{(1+\theta_i)^2}, \frac{-\beta_i\theta_i a_i^2\zeta^2_{i,(2)}}{(1+\theta_i)^2}, \cdots, \frac{-\beta_i\theta_i a_i^2\zeta^2_{i,(r)}}{(1+\theta_i)^2}, \\ &\Delta_2 = 0, \Delta_3 = 0, \cdots, \Delta_r = 0. \end{aligned} \tag{7.17}$$

Clearly all $\Delta_1 < 0$ and all higher order principal minors are 0. Therefore $H^i(\alpha_{\mathcal{A}_r})$ is NSD for all $\alpha_{\mathcal{A}_r}$. Hence, $f^i(\alpha_{\mathcal{A}_r})$ is concave for all $i \in \mathcal{A}_r$. This implies that the aggregated utility natural logarithm $\log\big(V(\alpha_{\mathcal{A}_r})\big) = \sum_{i=1}^r f^i(\alpha_{\mathcal{A}_r})$ is concave.

Therefore, the optimization problem in (7.14) has concave objective function with affine constraints. Hence, the optimization problem in (7.14) is convex. □

Corollary 7.1. *The direct optimization problem* ***SSO*** *in* (7.9) *is also convex.*

Proof: We note that the problem **SSO** in (7.9) is a special case of the problem **IO** in (7.14). Specifically, the set $\mathcal{A}_r$ in **IO** is equal to $\mathcal{R}$ in **SSO**. Further, each latency equation in (7.5) is a function of $R!$ optimization variables γ, whereas in (7.10) it is a function of r variables $\alpha_{\mathcal{A}_r}$. We can consider this as $\alpha_{\mathcal{A}_r}$, $l^*_{i,j}$ and $l_{i,i}$ in **IO** being replaced by γ, $l^s_{i,j}$ and $l^s_{i,i}$ in **SSO** respectively. Hence, the problem **SSO** is just a special case of **IO** with same structure.

Now from Theorem 7.1, we note that the optimization problem in (7.14) is convex. Therefore, we can conclude that the direct optimization problem **SSO** in (7.9) is also convex.

7.3 Optimal Scheduler

We now solve the dual problem of (7.14) to determine the optimal solution. We first define the Lagrangian as,

$$L(\alpha_{\mathcal{A}_r}, \eta) = \sum_{i \in \mathcal{A}_r} \beta_i \log\left(U_i(\alpha_{\mathcal{A}_r})\right) - \eta \left(\sum_{i \in \mathcal{A}_r} \alpha_i - 1 \right), \tag{7.18}$$

where $\eta \geq 0$ is a Lagrange multiplier. The dual problem is formulated as follows,

$$\begin{aligned} &\min_{\eta} \max_{\alpha_{\mathcal{A}_r}} L(\alpha_{\mathcal{A}_r}, \eta) \\ &\text{s.t.} \quad \eta \geq 0, \alpha_i \geq 0, \ \forall \ i \in \mathcal{A}_r. \end{aligned} \tag{7.19}$$

At the optimal solution $(\alpha^*_{\mathcal{A}_r}, \eta^*)$, we have,

$$\frac{\partial}{\partial \alpha_i} \left[\log(V(\alpha_{\mathcal{A}_r})) - \eta \left(\sum_{j \in \mathcal{A}_r} \alpha_j - 1 \right) \right]_{\alpha^*_{\mathcal{A}_r}, \eta^*} = 0, \tag{7.20}$$

$$\sum_{j \in \mathcal{A}_r} \frac{\beta_j}{U_j(\alpha_{\mathcal{A}_r})} \frac{\partial}{\partial \alpha_i} U_j(\alpha_{\mathcal{A}_r}) \Big|_{\alpha_{\mathcal{A}_r} = \alpha^*_{\mathcal{A}_r}} = \eta^* \ \forall \ i. \tag{7.21}$$

Using (7.2) and (7.10), we have,

$$\frac{\partial}{\partial \alpha_i} U_j(\alpha_{\mathcal{A}_r}) = \frac{-c_j \beta_j a_j \mathrm{e}^{-a_j(l_j - b_j)} \zeta_{j,i}}{\left(1 + \mathrm{e}^{-a_j(l_j - b_j)}\right)^2} \ \forall \ i. \tag{7.22}$$

Substituting (7.22) in (7.21), followed by some algebraic manipulations, we get,

$$\sum_{j \in \mathcal{A}_r} \frac{k_{j,i}}{1 + \mathrm{e}^{-a_j(l_j - b_j)}} \Big|_{\alpha_{\mathcal{A}_r} = \alpha^*_{\mathcal{A}_r}} + \eta^* = 0 \ \forall \ i. \tag{7.23}$$

where $k_{j,i} = \beta_j a_j \zeta_{j,i}$ is a constant.

Also, at the optimal solution $(\alpha^*_{\mathcal{A}_r}, \eta^*)$, we have,

$$\frac{\partial}{\partial \eta} \left[\log(V(\alpha_{\mathcal{A}_r})) - \eta \left(\sum_{j \in \mathcal{A}_r} \alpha_j - 1 \right) \right]_{\alpha^*_{\mathcal{A}_r}, \eta^*} = 0, \tag{7.24}$$

$$\implies \sum_{j \in \mathcal{A}_r} \alpha^*_j = 1. \tag{7.25}$$

We determine the optimal solution by solving for $(\alpha^*_{\mathcal{A}_r}, \eta^*)$ using (7.23) and (7.25) simultaneously. The constraint $\alpha^*_i \geq 0$ is enforced by setting $\alpha^*_i = 0$ if it is negative and then we solve again for $\alpha^*_{\mathcal{A}_r}$. Now the optimal latency for r class system, $l^*_{\mathcal{A}_r}$, is obtained by setting $\alpha_{\mathcal{A}_r} = \alpha^*_{\mathcal{A}_r}$ in (7.10).

The iterative algorithm for determining the optimal multiclass scheduler is described in Algorithm 4. The function optSch is initialized with $\mathcal{A}_r = \mathcal{R}$, i.e., $r = R$. At the r-class iteration, we iterate through each class $i \in \mathcal{A}_r$, assign it as highest priority in $\mathcal{A}_r$ which is equivalent to moving it from $\mathcal{A}_r$ to $\mathcal{A}^{\mathsf{c}}_r$. We then determine the optimal scheduler for resultant $r-1$ class system, $\mathcal{A}_{r-1} = \mathcal{A}_r \backslash \{i\}$. This downward recursion is performed until $r = 2$ for which the optimal solution is determined using procedure outlined in Section 7.3. Then the optimal solution from $r-1$ class system, $(l^*_{\mathcal{A}_{r-1}}, \alpha^*_{\mathcal{A}_{r-1}})$ is returned upwards to to determine the optimal scheduler for r class system, $\mathcal{A}_r$. Finally, the $\alpha^*_{\mathcal{A}_r} \forall r, \mathcal{A}_r$ are aggregated to determine the fraction of time the scheduler at AS uses each of the $R!$ priority orders. We denote this by $\gamma = \{\gamma_i : i \in \mathsf{R}\}$, as described in Section 7.2.2.

Algorithm 4 Proposed Iterative r-Class Scheduler

```
function optSch(A_r)
    if length(A_r) > 2 then
        for i ∈ A_r do
            if i < max(A_r^c) then
                Reuse l*_{A_r\{i}} and α*_{A_r\{i}}.
            else
                (l*_{A_r\{i}}, α*_{A_r\{i}}) = optSch(A_r\{i})
            end if
            Determine l_{i,i} using Eq (7.11).
        end for
    else                                              ▷ For r = 2
        Use (7.11) and (7.12) for l_{i,i}, l*_{i,j}.
    end if
    Use method in Section 7.3 to solve for l*_{A_r} and α*_{A_r}.
return (l*_{A_r}, α*_{A_r}).
end function
```

7.3.1 Complexity Analysis

We now compare the complexity of solving the single-stage and iterative optimization problems.

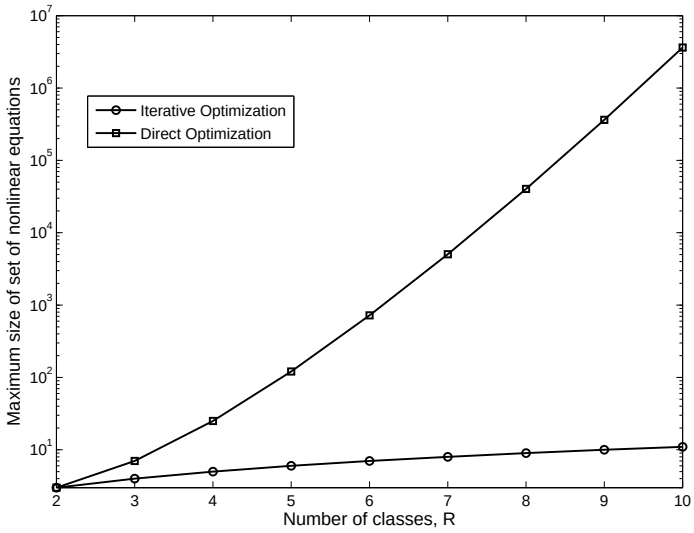


Figure 7.4: Complexity of iterative and direct optimization schemes.

Single-stage optimization

For a R class system, there are $R!$ unique priority orders among the classes and we need to determine the optimal fraction of time γ_i^* the scheduler serves using the i^{th} priority order. Thus, we need to solve a set of $R! + 1$ nonlinear equations to determine γ^*. Hence, the computational complexity of determining the optimal scheduler is enormous, even at moderate number of classes, R.

Iterative optimization

In this case, we solve optimization problems for r class system, $\forall\ \mathcal{A}_r, \forall\ 2 \leq r \leq R$. Since the priority order among the classes in $\mathcal{A}_r^{\mathsf{c}}$ does not effect the latency of classes in $\mathcal{A}_r$, the number of r class optimization problems are $\binom{R}{r}$, each requiring us to solve a set of $r + 1$ nonlinear equations. Thus, the iterative algorithm scales well with the number of classes by solving multiple but quite small optimization problems. The maximum number of simultaneous nonlinear equations is $R + 1$ for $r = R$. This complexity reduction using iterative optimization is illustrated in Fig. 7.4 that plots the maximum number of simultaneous nonlinear equations that needs to be solved in both schemes as the number of classes R are increased.

Clearly, the maximum complexity of an optimization problem for iterative optimization is negligible compared to the direct optimization, even though it requires us to solve a large number of optimization problems.

7.4 Jointly Optimal MA-AS Scheduler

We now extend the proposed scheduler to design a jointly-optimal packet scheduler at M MAs and AS. In our system model, we assumed a Poisson arrival process and constant packet size for each QoS class at each MA. Therefore, the queuing process for any class at any given MA is a M/D/1 process and hence the aggregated traffic for each class at AS becomes a G/D/1 process.

7.4.1 Centralized Optimization at M2M AS

We extend the analysis in Section 7.2.2 to formulate a joint utility-maximization problem to be solved centrally at AS in a single stage. We first define the set $\mathcal{M} = \{1, 2, \cdots, M\}$. Let the k^{th} MA adopt the j^{th} non-preemptive priority order[7] for $\gamma_{j,k}$ fraction of the time and result in average latency of $l^s_{i,j,k}$ for class $i, \forall\ i \in \mathcal{R}, \forall\ j \in \mathsf{R}$ and $\forall\ k \in \mathcal{M}$. Then the average latency l_i of class i is given by,

$$l_i = \sum_{j=1}^{R!} \gamma_j l^s_{i,j} + \sum_{k=1}^{M} w_{i,k} \sum_{j=1}^{R!} \gamma_{j,k} l^s_{i,j,k} \ \forall\ i, \tag{7.26}$$

where the first component represents the latency experienced at AS as in (7.5) and the weight $w_{i,k} = \lambda_{ik}/\lambda_i$ represents the contribution of k^{th} MA towards overall latency of class i. Let $\mathcal{H}^k_{i,j}$ denote the set of classes with higher priority than class i in the j^{th} priority order at k^{th} MA. Using the existing result on latency of M/D/1 nonpreemptive priority system [59], we have,

$$l^s_{i,j,k} = \frac{1}{\mu_{ik}} + \frac{1}{\rho_{\mathcal{H}^k_{i,j}}(\rho_{\mathcal{H}^k_{i,j}} - \rho_{ik})} \sum_{m \in \mathcal{R}} \frac{\rho_{mk}}{2\mu_{mk}} \ \forall\ i, \tag{7.27}$$

where $\rho_{ik} = \lambda_{ik}/\mu_{ik}$ is the utilization factor for class i at k^{th} MA and $\rho^k_{\mathcal{H}_{i,j}} = 1 - \sum_{m \in \mathcal{H}^k_{i,j}} \rho_{mk}$.

A closed-form expression for $l^s_{i,j}$ exists only if we have a M/G/1 queuing system at AS and is given in (7.6). However, due to the assumption of constant packet sizes, the queuing process at MAs and AS is M/D/1 and G/D/1 respectively. In this case, the latency $l^s_{i,j}$ can be determined numerically by simulating the queuing processes.

Let $\hat{\gamma} = \{\gamma_j : j \in \mathsf{R}\}$ and $\hat{\gamma_k} = \{\gamma_{j,k} : j \in \mathsf{R}\}$. Then the joint scheduling policy is characterized by $\gamma = \{\hat{\gamma}, \hat{\gamma_k} : k \in \mathcal{M}\}$. Then, similar to Section 7.2.2, we obtain the following

[7] We assume non-preemptive priority scheduling at each MA to prevent discontinuities in transmission of a packet over the wireless channel.

Centralized Optimization problem,

$$\begin{aligned} \max_{\gamma} \quad & \log\big(V(\gamma)\big) = \sum_{i=1}^{R} \beta_i \log\big(U_i(\gamma)\big) \\ \text{s.t.} \quad & \gamma_j, \gamma_{j,k} \geq 0 \ \forall \ j \in \mathsf{R}, \forall \ k \in \mathcal{M} \\ & \sum_{j=1}^{R!} \gamma_j = 1, \sum_{j=1}^{R!} \gamma_{j,k} = 1 \ \forall \ j, \ \forall \ k \in \mathcal{M}. \end{aligned} \tag{7.28}$$

Since this optimization problem has $(M+1)R!$ variables, it is prohibitively complex to solve, even at moderate values of M and R.

7.4.2 Distributed Optimization

Since the complexity of the centralized optimization is prohibitively high, we reformulate it into $M+1$ optimization subproblems to be solved separately at AS and the M MAs in an iterative fashion. At the start of each iteration, the AS broadcasts to the MAs its locally optimal scheduler[8] from the previous iteration. Each MA also has the information about the locally optimal scheduler at other MAs in the previous iteration. Then each MA simultaneously solves the centralized optimization problem in (7.28) by fixing the scheduling decisions for other MAs and AS and solving for its locally optimal scheduler using the iterative optimization framework presented in Section 7.2.3. Then each MA broadcasts its locally optimal scheduler to other MAs and AS. The AS on receives the broadcasts from MAs solves for its locally optimal scheduler. We show that we can completely decouple the optimization problems at MAs and AS, given that the arrival process at AS is represented by a Poisson process.

We now describe the optimization sub-problem at m^{th} MA and AS at n^{th} iteration.

Optimization at m^{th} MA

We now refer to the iterative optimization framework presented in Section 7.2.3. For a given set $\mathcal{A}_r$ of r classes, if the m^{th} MA serves class $i \in \mathcal{A}_r$ with highest priority for α_i fraction of

[8]Here locally optimal scheduler means the optimal scheduler at a node determined by fixing the scheduling decisions at other node.

time, then using (7.10) and (7.26), the average latency l_i of class i is,

$$\begin{aligned} l_i &= \underbrace{\sum_{j=1}^{R!} \gamma_j^{n-1} l_{i,j}^s + \sum_{k=1,k\neq m}^{M} w_{i,k} \sum_{j=1}^{R!} \gamma_{j,k}^{n-1} l_{i,j,k}^s}_{\psi_{im}^{n-1}} \\ &\quad + w_{i,m}(\alpha_{i,m} l_{i,i,m} + \sum_{j\in\mathcal{A}_r, j\neq i} \alpha_{j,m} l_{i,j,m}^*) \ \forall \ i \in \mathcal{R}, \\ &= \psi_{im}^{n-1} + w_{i,m}(\alpha_{i,m} l_{i,i,m} + \sum_{j\in\mathcal{A}_r, j\neq i} \alpha_j l_{i,j,m}^*), \end{aligned} \tag{7.29}$$

where γ_j^{n-1} and $\gamma_{j,k}^{n-1}$ denotes the optimal value of γ_j and $\gamma_{j,k}$ at $(n-1)^{\text{th}}$ iteration. Here $\alpha_{i,m}$, $l_{i,i,m}$ and $l_{i,j,m}^*$ have same meaning as $\alpha_i, l_{i,i}$ and $l_{i,j}^*$ in Section 7.2.3, but are defined for m^{th} MA. Similarly, we define $\alpha_{\mathcal{A}_r}^m = \{\alpha_{i,m} : i \in \mathcal{A}_r\}$.

Since we assume the packet size is constant, the queuing process at AS is G/D/1. Therefore, the arrival process for each class and hence the latency terms $l_{i,j}^s$ at AS are functions of the scheduling policy $\hat{\gamma}_k$ at the k^{th} MA $\forall$ k. Given that the arrival process for each class at AS is a Poisson process, we can decouple the optimization problems at each MA and AS as $l_{i,j}^s$ is independent of the scheduling policy $\hat{\gamma}_k$ and thus the term ψ_{im}^{n-1} can be treated as a constant.

We note that (7.29) is similar to (7.10) except for the additive constant ψ_{im}^{n-1} and multiplicative constant $w_{i,m}$. Therefore using the procedure in in Section 7.2.3, the resultant r-class optimization problem at m^{th} MA can also be easily shown to be convex from Theorem 7.1 and thus solved iteratively using Algorithm 4. Finally, the optimal solution at each iteration of Algorithm 4 is aggregated to determine the fraction of time the scheduler at m^{th} MA uses each of the $R!$ priority orders during the n^{th} iteration, i.e., $\hat{\gamma}_m^n = \{\gamma_{j,m}^n : j \in \mathsf{R}\}$.

Optimization at AS

For a given set $\mathcal{A}_r$ of r classes, if the AS serves class $i \in \mathcal{A}_r$ with highest priority for α_i fraction of time, then using (7.10) and (7.26), the average latency l_i of class i at n^{th} iteration is,

$$\begin{aligned} l_i &= \alpha_i l_{i,i} + \sum_{j\in\mathcal{A}_r, j\neq i} \alpha_j l_{i,j}^* + \underbrace{\sum_{k=1}^{M} w_{i,k} \sum_{j=1}^{R!} \gamma_{j,k}^n l_{i,j,k}^s}_{\psi_i^n}, \\ &= \alpha_i l_{i,i} + \sum_{j\in\mathcal{A}_r, j\neq i} \alpha_j l_{i,j}^* + \psi_i^n \ \forall \ i \in \mathcal{R}, \end{aligned} \tag{7.30}$$

where the last term ψ_i^n is a constant. Again, for reasons mentioned earlier, the resultant r-class optimization problem at AS is convex and is solved iteratively using Algorithm 4.

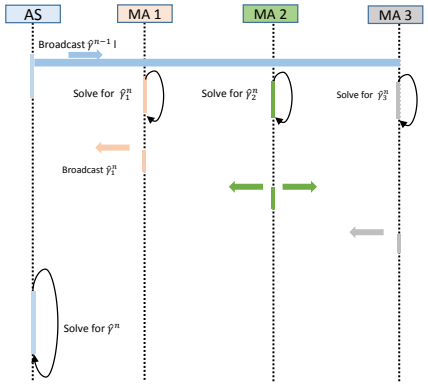


Figure 7.5: Sequence diagram illustrating the Distributed optimization at n^{th} iteration.

Finally, the optimal solution at each iteration of Algorithm 4 is aggregated to determine the fraction of time the scheduler at AS uses each of the $R!$ priority orders during the n^{th} iteration, i.e., $\hat{\gamma}^n = \{\gamma_j^n : j \in \mathsf{R}\}$.

The distributed optimization process at n^{th} iteration for $M = 3$ is illustrated in Fig. 7.5. A simple way to initialize the local scheduling decisions, prior to the 1^{st} iteration, is by setting $\gamma_j^0 = 1/R!\ \forall\ j \in \mathsf{R}$ for AS and $\gamma_{j,k}^0 = 1/R!\ \forall\ j \in \mathsf{R}$ for the k^{th} MA $\forall\ k \in \mathcal{M}$.

7.4.3 Complexity Analysis

We now compare the complexity of the centralized and distributed optimization problem for a M2M uplink with M MAs and R classes.

Centralized Optimization

Here, we formulate a single-stage optimization problem at AS with $(M+1)R!$ variables as can be noted from Eq. (7.28). On solving the Lagrangian dual problem similar to the procedure in Section 7.3, the problems reduces to solving a set of $(M+1)R! + M + 1$ nonlinear equations simultaneously.

Distributed Optimization

In the distributed optimization framework, at each iteration, we adopt the iterative optimization approach to determine the locally optimal scheduler at the MAs and AS. Again as discussed in Section 7.3.1, we solve $\binom{R}{r}(M+1)$ optimization problems for r-class system

($\forall\ 2 \leq r \leq R$), each requiring to solve a set of $r+1$ simultaneous equations. However, the maximum number of simultaneous nonlinear equations solved at any MA or AS is still $R+1$, which is much smaller than that needed for centralized optimization.

We now analyze the overhead due to information exchange among the MAs and AS during each iteration. We note that, the set $\alpha_{\mathcal{A}_r}$ and $\alpha^k_{\mathcal{A}_r}\ \forall\ \mathcal{A}_r,\ \forall\ r$ is a complete characterization of $\hat{\gamma}$ and $\hat{\gamma}^k$ respectively. So we can either choose to broadcast $\alpha_{\mathcal{A}_r}, \alpha^k_{\mathcal{A}_r}$ or $\hat{\gamma}, \hat{\gamma}^k$.

Case 1: Broadcast $\alpha_{\mathcal{A}_r}, \alpha^k_{\mathcal{A}_r}$
Each node solves $\binom{R}{r}$ r-class optimization subproblems. The total number of optimization variables for R-class problem at any node is,

$$N_R = \sum_{r=2}^{R} \binom{R}{r}(r-1), \tag{7.31}$$

$$= (R-1)(2^{R-1}-1) - 2^{R-1} + R, \tag{7.32}$$

where in Eq. 7.31, we consider $r-1$ variables for r-class problem as the sum of the optimization variables in each subproblem is constant (equal to 1). Therefore, the total number of variables transmitted over-the-air due to M MAs and AS is $N_1 = (M+1)N_R$.

Case 2: Broadcast $\hat{\gamma}, \hat{\gamma}^k$
In this case, each node broadcasts $R! - 1$ variables. Thus a total of $N_2 = (M+1)(R!-1)$ variables are transmitted.

Fig. 7.6 plots the number of optimization variables broadcasted in each case as R is varied. We note that the number of variables in two cases are same upto $R = 3$ and then it grows drastically in case 2. At $R = 8$, the number of variables in case 2 is roughly 52 times more than that in case 1. Therefore, we choose to broadcast $\alpha_{\mathcal{A}_r}, \alpha^k_{\mathcal{A}_r}$ for information exchange in each iteration.

7.5 Joint Subcarrier Allocation and Packet Scheduler

We now extend our work to jointly determine the delay-optimal subcarrier allocation and packet scheduling for the MA-AS system.

7.5.1 Centralized Optimization at M2M AS

We now formulate the joint optimization problem to be solved centrally at the M2M Application Server (AS). The problem formulation follows on similar line as in Section 7.4.1. The difference is that now the service rates at MAs are not fixed but depends on the number of subcarriers (N_k) assigned to the MAs which is determined by solving the following joint

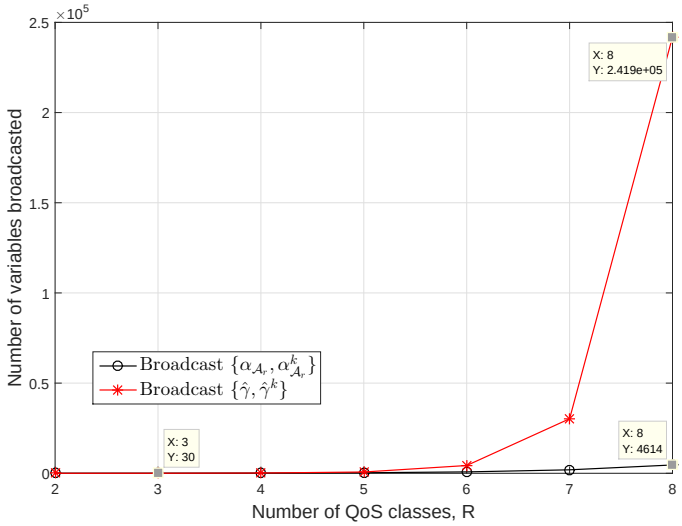


Figure 7.6: Information exchange overhead as R is varied.

optimization problem.

$$\begin{aligned} \max_{\gamma_j, \gamma_{j,k}, N_k} & \sum_{i \in \mathcal{R}} \beta_i \log\left(U_i(\gamma_j, \gamma_{j,k}, N_k)\right) \\ \text{s.t.} & \gamma_j, \gamma_{j,k} \geq 0 \ \forall \ j \in \mathsf{R}, \forall \ k \in \mathcal{M} \\ & N_k \in \mathbb{Z}^+, \forall \ k \in \mathcal{M} \\ & \sum_{j \in \mathsf{R}} \gamma_j = 1, \sum_{k \in \mathcal{M}} N_k = N \\ & \sum_{j \in \mathsf{R}} \gamma_{j,k} = 1 \ \forall \ k \in \mathcal{M}. \end{aligned} \tag{7.33}$$

We note that $\gamma_j, \gamma_{j,k}$ are continuous variables whereas N_k are integer variables. Therefore, this optimization problem is a mixed integer nonlinear programming (MINLP) problem and are in general shown to be NP-hard. Furthermore, this optimization problem has a total of $(M+1)R! + N$ variables, which is huge even at moderate values of M and R. Thus, the centralized joint optimization problem is prohibitively complex to solve and hence necessitates the need of alternative low-complexity optimization procedures.

Also, as pointed out earlier in Section 7.4.1, there is no closed form expression for average latency of classes at AS if packet size is assumed constant, i.e.,for a G/D/1 queuing process at AS. Resorting to numerical solution for approximating the latency expression becomes extremely difficult here due to variable MA service rate. Therefore, we assume the packet size for each class to be geometrically distributed, resulting in an exponential service time distribution and thus M/M/1 arrival process at MAs and AS. We show later in Fig. 7.11 that the approximation of exponential service distribution results in a near-optimal delay

performance.

We next present a novel low-complexity iterative optimization framework that is implemented in a distributed fashion across the MAs and AS.

7.5.2 Proposed Distributed Optimization Framework

We reformulate the joint channel assignment and packet scheduling problem as a two-stage distributed optimization problem. Stage 1 involves solving a distributed optimization subproblem at each MA to determine the delay-efficient subcarrier assignment to MAs. This is then fed as input to Stage 2 which determines the jointly optimal scheduling problem at MAs and AS while fixing the subcarrier assignment from Stage 1. Since the Stage 2 is same as discussed previously in Section 7.4.2, here we focus only on Stage 1. We will show later in Section 7.6 that distributed optimization approach does not result in same as the optimal solution to the centralized channel assignment and packet scheduling.

Stage 1: Subcarrier Allocation to MAs

We initialize Stage 1 with a *minimally-feasible* assignment of subcarrier which is defined as follows,

Definition 3. *Consider a M2M uplink with M MAs, R traffic classes with λ_{ik} and $\mu_{ik} = \frac{N_k C_s}{s_i}$ denoting the arrival and service rate of i^{th} class at k^{th} MA respectively. Let $\mathcal{N} = \{N_1, N_2, \cdots, N_m\}$ denote the set of any unconstrained integer subcarrier assignment. Then the minimally-feasible subcarrier assignment $\mathcal{N}^{mf} = \{N_1^{mf}, N_2^{mf}, \cdots, N_m^{mf}\}$ is defined such that if there exists a subcarrier assignment $\mathcal{N}$ with $N_k < N_k^{mf}$ for any $k \in \mathcal{M}$, then $\sum_{i \in \mathcal{R}} \frac{\lambda_{ik}}{\mu_{ik}} > 1$ and thus results in a unstable queueing system at k^{th} MA.*

Thus the *minimally-feasible* assignment is the smallest possible subcarrier assignment at each MA beyond which it would result in an unbounded queue at that MA. It is straightforward to prove that the *minimally-feasible* assignment is unique.

Next, for the *minimally-feasible* assignment, the k^{th} MA $\forall$ k solves a locally optimal packet scheduling problem (**DO-k**) without any knowledge of scheduling policy at other MA or AS. We enable this be setting l_i as the average latency of i^{th} class at k^{th} MA as,

$$l_i = w_{i,k} \sum_{j \in \mathsf{R}} \gamma_{j,k} l_{i,j,k} \ \forall \ i \in \mathcal{R}, \tag{7.34}$$

Therefore, we obtain the following optimization problem at k^{th} MA.

$$\begin{aligned} \textbf{DO-k:} \max_{\gamma_{j,k}} \quad & Z_k^{\text{mf}} = \sum_{i \in \mathcal{R}} \beta_i \log\left(U_i(\gamma_{j,k}, N_k^{\text{mf}})\right) \\ \text{s.t.} \quad & \gamma_{j,k} \geq 0 \ \forall \ j \in \mathsf{R} \\ & \sum_{j \in \mathsf{R}} \gamma_{j,k} = 1. \end{aligned} \tag{7.35}$$

Let $\hat{Z}_k^{\text{mf}}$ denote the value of objective for k^{th} MA at the optimal solution and let $\hat{Z}^{\text{mf}} = \{\hat{Z}_1^{\text{mf}}, \hat{Z}_2^{\text{mf}}, \cdots, \hat{Z}_M^{\text{mf}}\}$. To further reduce the computation complexity associated with the optimization problem **DO-k**, we adopt the iterative optimization scheme discussed in our previous work.

Next, we assign the remaining subcarriers one by one to the MA that results in the maximum increase in system utility[9] Thus, the subcarrier assignment is an iterative process with $\tilde{N} = N - \sum_{k \in \mathcal{M}} N_k^{\text{mf}}$ iterations. During each iteration, we solve the optimization problem at k^{th} MA similar to Eq. (7.35) (again using the iterative approach in our previous work) only if it is assigned a subcarrier in the previous iteration, thus avoiding solving unnecessary optimization problems. The proposed subcarrier assignment heuristic is described in Algorithm 5. The inputs to the heuristic are the *minimally-feasible* subcarrier assignment $\mathcal{N}^{\text{mf}}$ and the resultant optimal objective value at MAs $\hat{Z}^{\text{mf}}$. The output is the number of subcarriers assigned to each MA, $\mathcal{N}$.

Stage 2: Packet Scheduler at MAs and AS

The Stage 2 problem involves determining the jointly optimal packet scheduling scheme at MAs and AS while fixing the subcarrier allocation as the output of Stage 1. This problem can be solved with low-complexity in a distributed and iterative fashion across MAs and AS as discussed in Section 7.4.2.

7.5.3 Complexity Analysis

We now compare the complexity of the centralized and distributed frameworks for joint subcarrier assignment and packet scheduler in a M2M uplink with M MAs, R classes and N subcarriers.

[9]Since the capacity of each subcarrier is same for AWGN case, it is irrelevant which subcarrier gets assigned to a MA; rather what matters is the number of subcarriers assigned to each MA.

Algorithm 5 Proposed Subcarrier Assignment Heuristic

function OPTCHASS($\mathcal{N}^{\mathrm{mf}}, \hat{Z}^{\mathrm{mf}}$)
  Set $\mathcal{N} = \mathcal{N}^{\mathrm{mf}}$, $\hat{Z} = \hat{Z}^{\mathrm{mf}}$.
  Set $\tilde{N} = N - \sum_{k \in \mathcal{M}} N_k^{\mathrm{mf}}$.
  Set update flag $f_k = 1 \; \forall \; k \in \mathcal{M}$.
  for $i = N - \tilde{N} + 1 : N$ **do** ▷ Iterate over remaining subcarriers
    for $k = 1 : M$ **do** ▷ Iterate over each MA
      if $f_k = 1$ **then**
        Solve **DO-k** with $N_k + 1$ subcarriers.
        Calculate the differential utility $\Delta \hat{Z}_k$.
      end if
    end for
    Find the MA, k^*, with largest $\Delta \hat{Z}_k$.
    Set $f_k = 0, \forall \; k \neq k^*$.
    Set $N_{k^*} = 1$ & $\hat{Z}_{k^*} = \hat{Z}_{k^*} + \Delta \hat{Z}_{k^*}$.
  end for
  return $\mathcal{N}$.
end function

Centralized Optimization

Here, we formulate and solve a single-stage MINLP optimization problem at AS with $(M + 1)R! + N$ variables. Since this is a NP-hard problem, the computational complexity is prohibitively high even at moderate values of M, R and N.

Distributed Optimization

Here we discuss the computational complexity of proposed subcarrier assignment heuristic (Stage 1) detailed in Algorithm 5. The complexity of Stage 2 is same as described in Section 7.4.3. Algorithm 5 is initialized with a *minimally-feasible* subcarrier assignment $\mathcal{N}^{\mathrm{mf}}$ as defined in Definition 3. The second input argument $\hat{Z}^{\mathrm{mf}}$ is the result of solving the local optimization problem **DO-k** (defined in Eq. 7.35) at the k^{th} MA $\forall k \in \mathcal{M}$ with N_k^{mf} subcarriers. Then, we allocate the remaining $\tilde{N}$ subcarriers one-by-one to the MAs. The first iteration requires k^{th} MA $\forall k \in \mathcal{M}$ to solve the optimization problem **DO-k** with $N_k^{mf} + 1$ subcarriers. Thus the complexity of first iteration is same as that of determining $\hat{Z}^{\mathrm{mf}}$. Then in each of $\tilde{N} - 1$ iterations, we solve **DO-k** only at the MA which is allocated an new subcarrier in the previous iteration.

Putting everything together, we solve $2M + \tilde{N} - 1$ instances of the optimization problem **DO-k** using the iterative procedure described in Section 7.2.3. The computational complexity of each problem is described previously in Section 7.3.1 and requires solving a maximum of $R + 1$ simultaneous nonlinear equations.

We next analyze the information exchange overhead in Stage 1 of distributed optimization

framework. The corresponding analysis for Stage 2 has been done in Section 7.4.3. Based on the channel conditions and arrival rate of different classes at each MA, the AS determines the *minimally-feasible* subcarrier assignment and feeds back this information to the MAs. Then the procedure for allocating the remaining $\tilde{N}$ subcarriers is done in $\tilde{N}$ iterations. In the first iteration, all MAs compute and transmit their corresponding $\Delta\hat{Z}_k$ to AS. The AS then determines the MA index with largest $\Delta\hat{Z}_k$ and broadcasts this MA index to the MAs. On receiving the message from AS, the MA with the allocated subcarrier updates its $\Delta\hat{Z}_k$ for next iteration, while other MAs simply ignore the AS message. Thus for each of the remaining $\tilde{N}-1$ iterations, the k^{th} MA updates and transmits $\Delta\hat{Z}_k$ to AS only if it was assigned a subcarrier in the last iteration.

Putting everything together, we note that the AS transmits M messages with initial subcarrier assignment to each MA and $\tilde{N}$ messages with MA index that is allocated next subcarrier. The The total number of $\Delta\hat{Z}_k$ values transmitted by all MAs to the AS are $M+\tilde{N}-1$.

7.6 Numerical Results

We now use Monte-Carlo simulations to compare the system utility and delay jitter performance of proposed scheduler against various state-of-the-art schedulers, namely WRR, WFS, max-weight scheduler and priority scheduling schemes. We consider 4 QoS classes classes with arrival rate as $[\lambda_2,\lambda_3,\lambda_4]=[0.02,0.04,0.05]$ s^{-1} , packet size as $[s_1,s_2,s_3,s_4]=[143,111,83,67]$ B and service rate $r=100$ B/s. The utility function parameters are as follows: $[a_1,a_2,a_3,a_4]=[0.35,0.4,0.45,0.7]$ s^{-1}, $[b_1,b_2,b_3,b_4]=[2,1.9,1.8,1]$ s and $[\beta_1,\beta_2,\beta_3,\beta_4]=[0.35,0.45,0.5,0.8]$. For WRR, the weights for class i are inversely proportional to the delay requirement (set as $b_i+\frac{4}{a_i}$) and the packet size s_i. After normalizing to integer values, we get the weights for WRR as $[52,76,112,223]$. For WFS, the weights are calculated similarly, except that we do not factor in the packet sizes. The weights for WFS are $[745,840,936,1489]$.

7.6.1 Multiclass Scheduler at AS

Fig. 7.7 compares the delay-performance of various scheduling schemes by plotting system utility metric as a function of λ_1. Firstly, we note that the theoretical result (both single-stage and iterative optimization) for proposed scheduler matches perfectly with the simulation, thus validating the correctness of our analysis. As expected, the system utility decreases with increase in λ_1 for all schedulers due to larger queuing delays. However, the proposed scheduler always outperforms other schedulers and the performance gap widens with increasing λ_1. This is because it prioritizes service to classes that are delay-sensitive and have greater impact of system utility. Therefore, the delay performance of proposed scheduler does not decrease significantly with increase in λ_1 due to the low-priority delay tolerant traffic of class 1. For the same reason, we note that prioritizing class 1 results in worst performance and

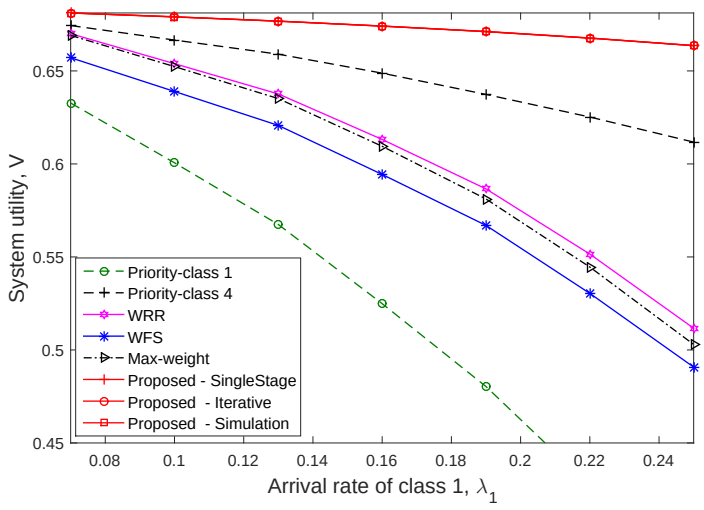


Figure 7.7: Comparison of system utility for $R = 4$ classes.

priority to class 4 performs really well and is second to only the proposed scheduler. This is because it gives equal priority to traffic of rest of the classes rather than prioritizing them based on their delay-sensitivity.

The performance of WRR and WFS is bad despite assigning higher weight to delay-sensitive traffic due to the usage of non-optimal weights. Even if we had used optimal weights, the performance of WRR and WFS would still remain sub-optimal because of their goal to achieve weighted fairness among different classes. Lastly, max-weight scheduling increasingly prioritizes class 1 over other classes as λ_1 increases due to its largest average queue size. Since class 1 is most delay-tolerant, this results in sharp performance loss with increasing λ_1.

A low packet delay jitter, in general, is a desirable feature for scheduling policies. This is all the more important for delay-sensitive traffic where high delay jitter can lead to packets exceeding their delay budget. Therefore, we study the delay jitter (as measured by packet delay variance) of class 1 and 4 for different schedulers as shown in Fig. 7.8 and 7.9 respectively. As suggested by intuition, we note that prioritizing traffic of a class results in its minimum delay variance.

As desired, the proposed scheduler results in near-minimal delay variance for the delay-sensitive class 4, far better than delay variance for other schedulers. This is because it gives high priority to class 4. Although, the proposed scheduler performs poorly on delay variance of class 1, this is not of grave concern due to its delay-tolerant and is also assigned least importance towards system utility. However, if low delay variance of class 1 is important, it can still be achieved at the cost of higher average delay of class 1, by using a delay-jitter regulator for class 1 at AS. Lastly, we infer that the proposed scheduler and priority scheduler always result in the extreme delay variances for classes. On the contrary, WFS and WRR due to their round robin service operation, results in moderate delay-variance for all classes.

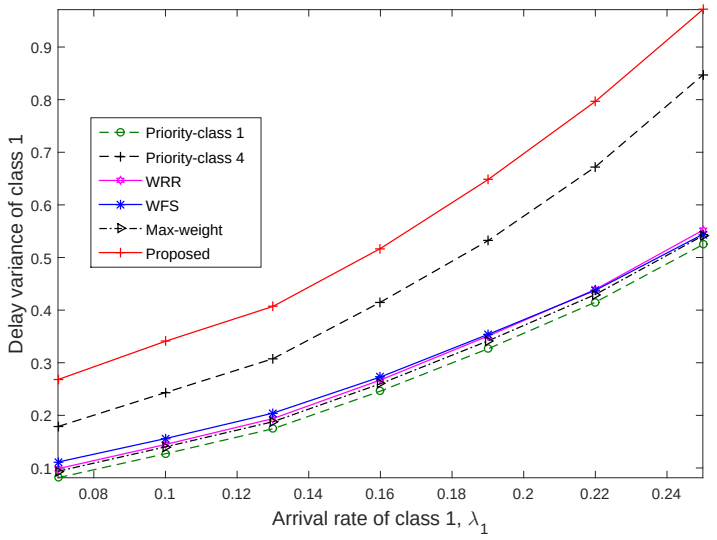


Figure 7.8: Delay variance of class 1 for different schedulers.

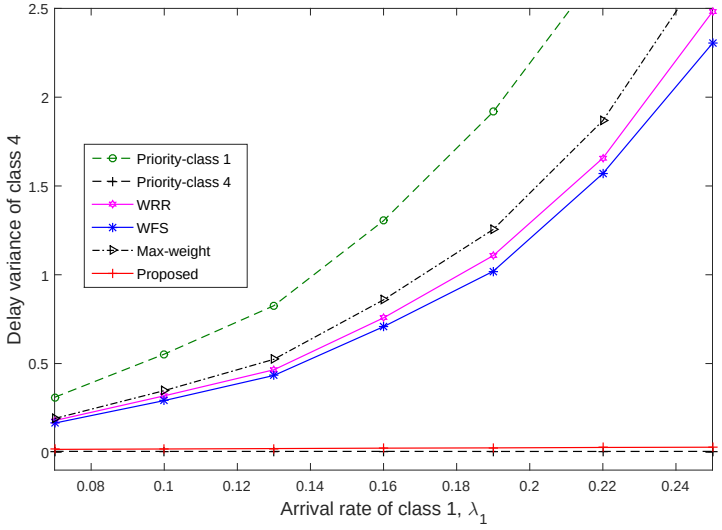


Figure 7.9: Delay variance of class 4 for different schedulers.

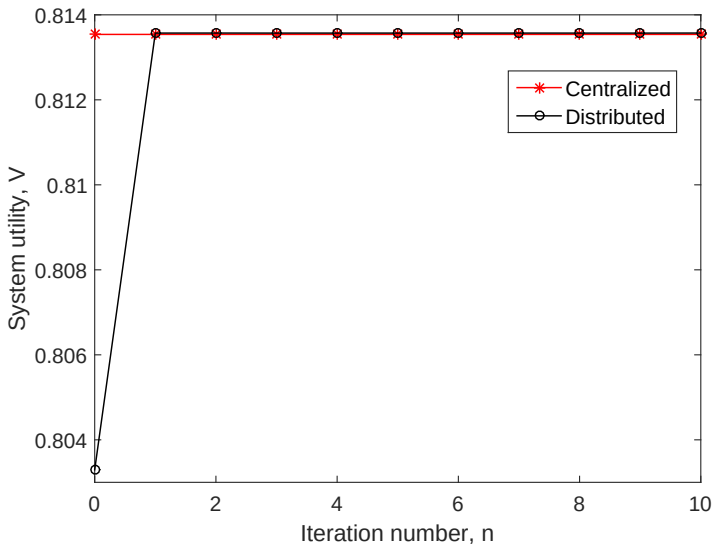


Figure 7.10: Convergence performance of distributed optimization for geometric packet size distribution.

7.6.2 Joint Multiclass Scheduler at MAs and AS

Next we numerically evaluate the joint scheduling problem with $M = 4$ MAs and $R = 4$ classes. The utility parameters of each class, i.e., a, b and β remain unchanged. The arrival rate for different classes at each MA is expressed as a matrix given below,

$$\begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} & \lambda_{14} \\ \lambda_{21} & \lambda_{22} & \lambda_{23} & \lambda_{24} \\ \lambda_{31} & \lambda_{32} & \lambda_{33} & \lambda_{34} \\ \lambda_{41} & \lambda_{42} & \lambda_{43} & \lambda_{44} \end{bmatrix} = \begin{bmatrix} 5 & 4 & 4 & 3 \\ 4 & 1 & 3 & 7 \\ 3 & 8 & 5 & 2 \\ 1 & 3 & 1 & 4 \end{bmatrix} \times 10^{-2} \text{ s}^{-1}.$$

The service rate at AS and MAs is $[r, r_1, r_2, r_3, r_4] = [40, 34, 20, 60, 30]$ B/s and packet sizes for 4 classes are $[s_1, s_2, s_3, s_4] = [14, 11, 8, 6]$ B.

Now, we expect the distributed optimization to be optimal when the arrival process at AS is actually Poisson. Using Burke's reversibility theorem [111], this implies that the queuing process at each AS should be M/M/1, which in turn translates to geometric packet size distribution for each class. Using Fig. 7.10, we verify that this is indeed the case. We note that the system utility for distributed optimization converges quickly (in just a single iteration) to the centralized optimization result. Here, we had initialized $\gamma_j^0 = 1/R! \ \forall \ j \in \mathsf{R}$ for AS and $\gamma_{j,k}^0 = 1/R! \ \forall \ j \in \mathsf{R}$ for the k^{th} MA $\forall \ k \in \mathcal{M}$, assuming that each node (MA or AS) has no information about arrival rate of classes at other nodes.

We now extend the previously discussed state-of-the-art schedulers to the joint scheduling problem, by simply reapplying them at each MA and AS. Fig. 7.11 compares the system utility of various joint-scheduling schemes as λ_{11} is varied while other parameters are kept con-

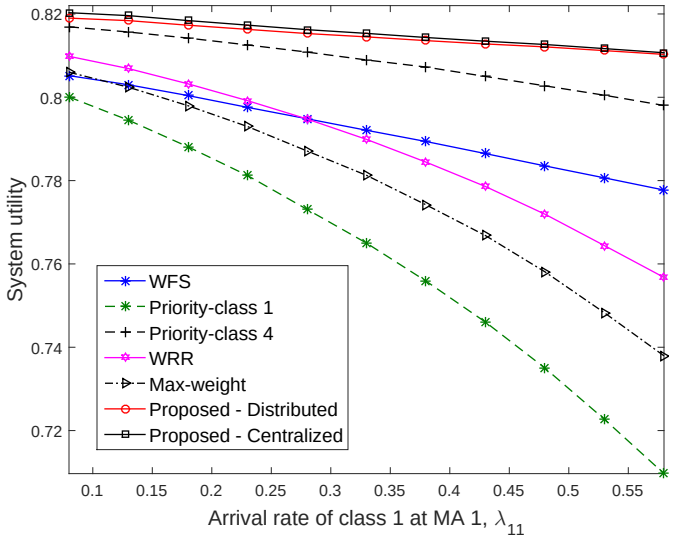


Figure 7.11: System utility for joint schedulers with $[M, R] = [4, 4]$.

MA Id	Region Served	Number of SMs
1	Large industrial plant	1700
2	Sparse residential area	100
3	Medium-size residential area	800
4	70% residential and 30% commercial area	1000

Table 7.1: List of M2M aggregators serving the different regions.

stant. We note that the centralized optimization scheduler has superior delay-performance as compared to other schedulers and the distributed optimization scheduler results in a near-optimal delay-performance. The loss in optimality is solely due to the Markovian arrival process assumption at AS.

7.6.3 Joint Subcarrier Assignment and Packet Scheduling

In this section, we evaluate the performance of proposed joint subcarrier assignment and packet scheduling scheme. We consider smart metering (SM) as a representative M2M application scenario and set the parameters in our model in accordance with the UCA OpenSG smart grid networks system requirements specification [36].

We consider that the uplink SM traffic from 4 different regions in a city is aggregated as specified in Table 7.1. The SM traffic is heterogeneous and comprises of the 4 important use-cases (i.e., $R = 4$ QoS classes) as listed in Table 7.2 along with payload size and latency requirements (utility parameters a_i and b_i) and arrival rate per device [112]. We assume that a certain fraction of the SMs in each region are enhanced Smart Meters (eSM) that provide detailed and frequent measurement reports (Class 1 in Table 7.2) of the power quality

M2M Use Case	Payload	(a, b, β)	Arrivals/sec/SM
eSM reporting	2500 B	(4,1,0.95)	1
Real-time pricing	25 B	(1.5,5,0.85)	$1.1 * 10^{-3}$
Meter reading (on-demand)	2400 B	(0.8,5,0.65)	$2.89 * 10^{-7}$
Regular meter readings	1560 B	(0.05,70,0.35)	$2.78 * 10^{-4}$(com.)/ $6.94 * 10^{-5}$(res.)

Table 7.2: List of important M2M use-cases and their characteristics.

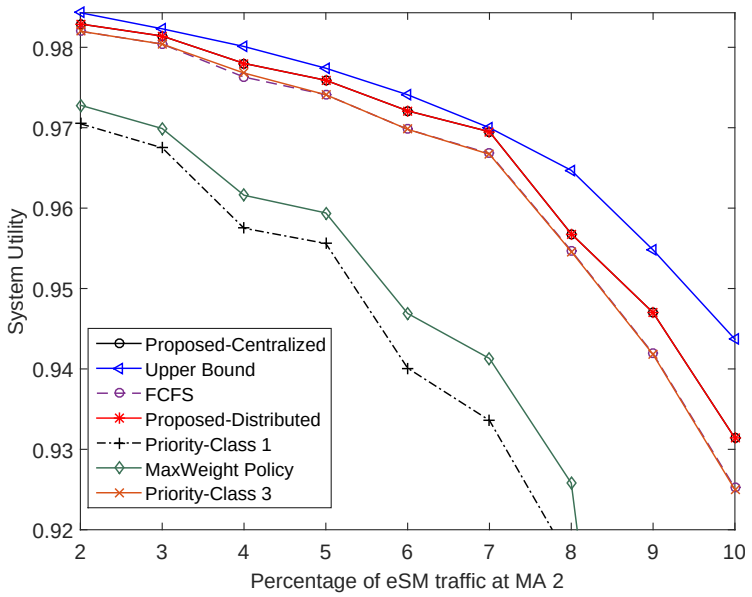


Figure 7.12: System utility for joint subcarrier allocation and packet scheduler.

parameters necessary for real-time monitoring and state estimation [112]. The exact fraction of eSM traffic is a design parameter and depends on several factors. However, for simplicity, we assume that eSM constitutes 3 % of overall traffic for MA 1, 3 and 4.

The subcarrier bandwidth, W, is set to 15 kHz and the received SNR at the AS under AWGN condition is $\eta = 10$ dB. We assume block i.i.d. Rayleigh fading for each MA-AS channel link. We repeat the proposed procedure for the optimal subcarrier allocation and packet scheduler capacity each time the channel changes significantly between any MA-AS link. For our simulation, we have $[\|h_1\|^2, \|h_2\|^2, \|h_3\|^2, \|h_4\|^2] = [3.205, 0.6454, 1.12, 1.778]$ and thus subcarrier capacity as $[C_{s,1}, C_{s,2}, C_{s,3}, C_{s,4}] = [75.66, 43.47, 54.13, 63.47]$ kbps. The AS service rate is set to $C_o = 4$ Mbps. The total number of subcarriers is $N = 40$.

We solve the centralized MINLP problem using the branch and bound method [57].

Fig. 7.12 shows the plot of system utility as the eSM traffic at MA 2 is increased from 2% to 10%. We note that the distributed implementation is as efficient as the centralized scheme and is close to an upper-bound obtained by convex relaxation of the centralized optimization by allowing for continuous subcarrier allocation. We also compare it against

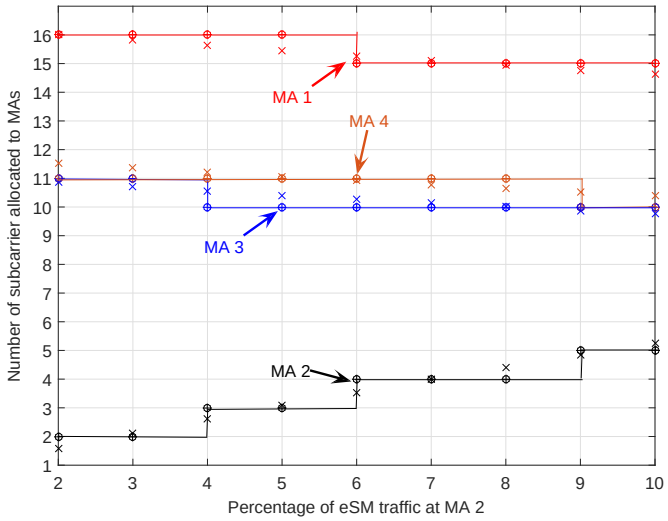


Figure 7.13: Subcarrier allocation to MAs as eSM traffic at MA 2 is increased. We plot the results for the following: (a) centralized integer subcarrier allocation 'o', (b) real-valued subcarrier allocation '$\times$', (c) distributed integer subcarrier allocation '+'.

the system utility obtained for FCFS, priority queueing and max-weight scheduling policies at MAs and AS. In each case, the proposed scheme performs better and the performance gap widens as eSM traffic at MA 2 is increases. Note that the plots are not smooth due to integer constraints on the subcarrier allocation.

We note that the eSM reports have very high arrival rate, payload size compared to other classes. Thus the max-weight policy which serves the longest queue, selects mostly the eSM reports for service, resulting in poor system utility. As expected, this effect is more pronounced as eSM traffic at MA 2 is increased. Priority to class 1 results in an even worse performance as it provides exclusive service to eSM reports. On the other hand, FCFS results in fairly good performance as it provides service opportunity to different classes in proportion to their arrival rate. Similarly prioritizing the low arrival rate class 3 improves its utility without much effect on other classes. In this case, the traffic of class 1, 2 and 4 is served in a FCFS manner.

Fig. 7.13 illustrates the dynamics of subcarrier allocation to MAs as the eSM traffic at MA 2 is increased from 2% to 10%. We first note that MA 2 is assigned the least number of subcarriers because it provides service to a sparse residential area with just 100 SMs. As eSM traffic at MA 2 is increased, more subcarriers are assigned to MA 2 due to real-time, low latency requirements and large payload of eSM traffic. Since the total number of subcarriers is fixed, this results in reduction of subcarriers allocated to other MAs. Now MA 3 has the least amount of traffic (or SMs) among MA 1, 3 and 4. Therefore, it is the first one to get hit and is followed by MA 1 and MA 4. Lastly, we note that the centralized and distributed subcarrier allocation results in same subcarrier assignment, which is also fairly close to the

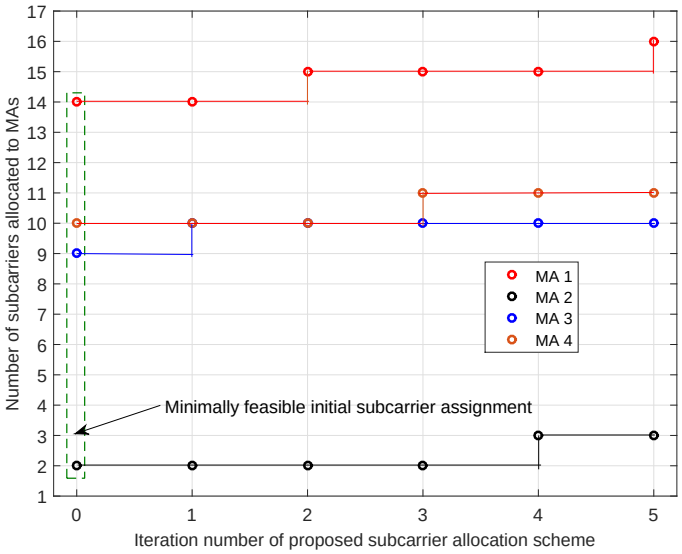


Figure 7.14: Iterative procedure of distributed subcarrier allocation to MAs when MA 2 has 4% eSM traffic. The eSM traffic at MA 2 is kept constant at 4%. Iteration 0 corresponds to the minimally feasible assignment used to initialize Algorithm 5.

continuous subcarrier allocation.

Fig. 7.14 illustrates the subcarrier allocation at end of each iteration in the proposed distributed optimization framework. The iteration 0 corresponds to the minimally feasible assignment used to initialize Algorithm 5. We note the subcarrier assigned to MA 2 (which has least amount of M2M traffic) remains fairly constant at 2 except towards the end of heuristic when it is assigned 3 subcarriers.

7.7 Conclusions

In this chapter, we studied delay-optimal MAC and packet scheduling strategies for a M2M uplink with heterogeneous arrivals at a M2M Application Server (AS) via multiple M2M Aggregators (MAs). We classified the uplink traffic into multiple QoS classes based on the M2M data characteristics such as maximum delay budget, packet size and arrival rate requirements. The packet delay requirements for each class are equivalently represented by sigmoidal utility functions. We then proposed a delay-optimal multiclass packet scheduler at AS by iteratively searching for the optimal fraction of time-sharing between all preemptive priority scheduling policies, such that it maximizes a proportionally-fair system utility metric. The computational complexity of the optimal scheduler is significantly reduced by iteratively solving multiple, small optimization problems rather than a single big optimization problem. We then extended our work to the design of joint packet scheduler at the MAs

and AS. We first considered centralized single-stage optimization at AS and then proposed a near-optimal distributed iterative optimization framework which allows each MA and AS to solve for its locally optimal scheduler while fixing the local scheduler at other nodes. The distributed optimization has low computational complexity, with minimal information exchange overhead and converges quickly to the optimal result for Poisson arrival process at AS.

Using Monte-Carlo simulations, we verified the optimality of the iterative scheduler and showed that it outperforms various state-of-the-art schedulers (such as WRR, WFS, max-weight and fixed priority scheduler) in terms of proposed system utility metric. It also results in near-minimal delay jitter for delay-sensitive traffic at the expense of somewhat higher delay jitter for delay-tolerant traffic. Finally, for the joint packet scheduling, we showed that the proposed scheduler (from centralized optimization) has superior delay performance as compared to other schedulers and that the distributed optimization results in a optimal (near-optimal) delay performance for exponential (constant) service time distribution.

Chapter 8

Conclusions

In this dissertation, we investigated and solved several resource allocation problems in wireless networks with diverse Quality-of-Service (QoS) requirements such as maximum tolerable delay or minimum capacity requirements of applications.

First, we studied cross-layer resource allocation in a macro-cellular LTE network that operates on a combination of licensed and leased spectrum. We incorporated the dynamics in available spectrum, user capacity requirements and assumed a Fractional Frequency Reuse architecture to improve the spectral efficiency for cell edge users. We proposed a low-complexity layered resource allocation heuristic that strikes a balance between rate maximization and fairness of resource allocation. Next, we studied a energy-harvesting small-cell network equipped with caching capabilities to reduce backhaul load and file download delay. Using Dynamic programming, we developed an *online* energy-efficient power control scheme and studied the tradeoff between cache size and energy harvesting capabilities for given QoS requirement of users.

We next analyzed the file download latency from a DSS in order to provide overall latency guarantees to the users. We proposed a heterogeneous DSS model that stores the data of multiple data-classes, each characterized by different arrival rate of file access requests, file size, and storage redundancy requirements. Using a queuing theoretic approach, we established upper and lower bounds on the average file access latency for each class for different scheduling policies. We also showed that erasure coding in DSS serves the dual purpose of reducing latency and increasing energy efficiency.

Lastly, we investigated the design of delay-efficient packet schedulers for a Machine-to-Machine (M2M) network with heterogeneous traffic characteristics such as packet arrival rate, maximum tolerable delay and payload size. We classified the uplink traffic into multiple QoS classes and proposed a proportionally-fair delay-efficient heuristic packet scheduler at M2M application server. Using a queuing theoretic approach, we next developed a delay optimal multi-class packet scheduler at M2M aggregators (MAs) and M2M application

server (AS). This was later extended it to joint medium access control and packet scheduling for M2M uplink. In each case, we proposed a low-complexity distributed optimization framework to be implemented at MAs and AS. Using extensive simulations, we showed that the proposed schedulers perform better than state-of-the-art schedulers in terms of average delay and packet delay jitter.

Future Work: In our work on M2M uplink scheduler, we assumed that in the M2M core, the data is processed by an M2M application server. However, for certain data-logging type applications, the data is first stored (*written*) in M2M cloud and is later pulled back (*read*) by a back-end application for analysis [113]. Furthermore, due to the high-reliability and availability requirements for M2M applications, the data needs to be stored redundantly across multiple servers in M2M cloud. Due to the massive volume of M2M uplink data, the overall M2M storage costs can grow very large. Therefore, this is a multi-faceted problem wherein we need to find a balance between storage, reliability/availability and *read*/*write* delay requirements. Among other things, this would require characterization of write latency in heterogeneous DSS which has not yet been studied in literature.

Appendices

Appendix A

Proofs of Results on Read Latency

A.1 Proof of Lemma 5.1: Stability Conditions

A.1.1 FCFS Scheduling

Consider any server in the $(n, k_1, k_2, ..., k_R)$ Fork-Join system. Jobs of class r enter the queue with rate λ_r. Each new job of class r exits the system when k_r sub-tasks of that job are completed. The remaining $n - k_r$ sub-tasks are then cleared from the system. Thus for each job of class r, $\frac{(n-k_r)}{n}$ fraction of the sub-tasks are deleted and hence the effective arrival rate of jobs of class r at any server is $\lambda_r \left(1 - \frac{n-k_r}{n}\right) = \frac{k_r \lambda_r}{n}$. Thus the overall arrival rate at any server, λ_{eff}, is

$$\lambda_{\text{eff}} = \sum_{r=1}^{R} \frac{k_r \lambda_r}{n}. \tag{A.1}$$

Let S denote the service distribution for a single-server FCFS system serving R data classes. Then from (2.10), the mean service time at a server is

$$E[S] = \sum_{r=1}^{R} p_r E[S_r] = \sum_{r=1}^{R} \frac{\lambda_r}{\mu_r \sum_{r=1}^{R} \lambda_r}, \tag{A.2}$$

where the assumption that the service time for a job of class i is exponential with rate μ_r. To ensure stability, the net arrival rate should be less than the average service rate at each

server. Thus from (A.1) and (A.2) the stability condition of each queue is

$$\sum_{r=1}^{R} \frac{k_r \lambda_r}{n} < \left(\sum_{r=1}^{R} \frac{\lambda_r}{\mu_r \sum_{r=1}^{R} \lambda_r} \right)^{-1},$$

Since $\mu_r = \frac{f k_r \mu}{l_r}$ and the term $\sum_{r=1}^{R} \lambda_r$ is a constant, with simple algebraic manipulations we arrive at

$$\left(\sum_{r=1}^{R} k_r \lambda_r \right) \left(\sum_{r=1}^{R} \frac{\lambda_r l_r}{k_r} \right) < n f \mu \sum_{r=1}^{R} \lambda_r. \tag{A.3}$$

This completes the proof of stability condition for FCFS scheduling.

A.1.2 Priority Scheduling

Consider any node in the $(n, k_1, k_2, ..., k_R)$ Fork-Join system. Jobs of class r enter the queue with rate λ_r. Each new job of class r exits the system when k_r sub-tasks of that job are completed. The remaining $n - k_r$ sub-tasks are then cleared from the system. Thus for each job of class r, $\frac{(n-k_r)}{n}$ fraction of the sub-tasks are deleted and hence the effective arrival rate of jobs of class r at any node is $\lambda_{\text{eff}}^r = \lambda_r \left(1 - \frac{n-k_r}{n}\right) = \frac{k_r \lambda_r}{n}$. The stability condition for a priority queue is that the overall server utilization should be less than 1. If the condition is violated, the queues belonging to a priority level lower than some limit k will grow without bound [59, Section 3.5.3]. Mathematically the stability condition is,

$$\sum_{r=1}^{R} \rho_r < 1, \tag{A.4}$$

where $\rho_r = \frac{\lambda_{\text{eff}}^r}{\mu_r}$ is the server utilization factor for class r. Substituting the expression for ρ_r with $\mu_r = \frac{f k_r \mu}{l_r}$ in (A.4) and rearranging terms, we get the stability condition of each queue as

$$\sum_{r=1}^{R} \lambda_r l_r < n f \mu. \tag{A.5}$$

A.2 Proof of Theorem 5.1: Upper Bound on Average Latency

A.2.1 FCFS Scheduling

The FCFS system can be modeled as a M/G/1 queuing system with arrival rate $\lambda = \sum_{r=1}^{R} \lambda_r$ and a general service time distribution S. Then the average latency for a job of class i in a FCFS scheduling system is given by (2.11) as,

$$T^i_{\text{fcfs}} = \mathrm{E}[S_i] + \frac{\sum_{r=1}^{R} \lambda_r \left[\mathrm{V}[S_r] + \mathrm{E}[S_r]^2\right]}{2\left(1 - \sum_{r=1}^{R} \lambda_r \mathrm{E}[S_r]\right)}. \tag{A.6}$$

To obtain an upper bound on the average latency, we degrade the FJ system in the following manner. For a job of class i, the servers that have finished processing a sub-task of that job are blocked and do not accept new jobs until k_i sub-tasks of that job have been completed. Then the sub-tasks at remaining $n - k_i$ servers exit the system immediately. Fig. A.1 illustrates this process using Example 5.1. When A_1 is finished at server 2, it is blocked (see Fig. 7(b)) until another $k_A = 2$ copies are finished. Now this performance-degraded system can be modeled as a M/G/1 system where the distribution of the service process, S_i, follows k_i^{th} ordered statistics as described in Section 5.2.2. Now for any class i, the service time at each of the n servers is exponential with mean $1/\mu_i$. Hence from (5.16), the mean and variance of S_i are,

$$\mathrm{E}[S_i] = \frac{H^1_{n-k_i,n}}{\mu_i},\ \mathrm{V}[S_i] = \frac{H^2_{n-k_i,n}}{\mu_i^2}. \tag{A.7}$$

Substituting (A.7) in (A.6), we get the following upper bound on average latency:

$$T^i_{\text{FCFS}} \leq \underbrace{\frac{H^1_{n-k_i,n}}{\mu_i}}_{\text{service time}} + \underbrace{\frac{\sum_{r=1}^{R} \lambda_r [H^2_{n-k_r,n} + (H^1_{n-k_r,n})^2]/{\mu_r}^2}{2\left(1 - \mathcal{S}_R\right)}}_{\text{waiting time}}, \tag{A.8}$$

where $\mathcal{S}_R = \sum_{r=1}^{R} \rho_r H^1_{n-k_r,n}$ and $\rho_r = \lambda_r/\mu_r$. This concludes the proof of upper bound on the average latency for FCFS scheduling.

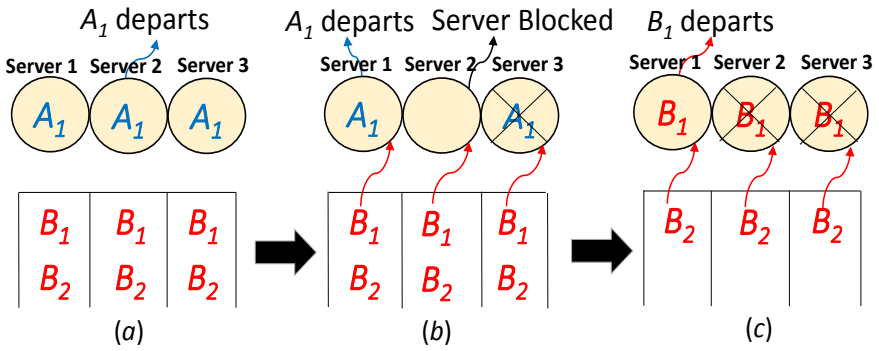


Figure A.1: Enhanced two-class FJ system with FCFS.

A.2.2 Non-preemptive Priority Scheduling

Let S_i be a random variable representing the service time for a job of class i. Then the average latency for a job of class i in a non-preemptive priority scheduling system[1] is given by Eq. 2.14 as,

$$T^i_{\text{N-PQ}} = \mathrm{E}[S_i] + \frac{\sum_{r=1}^{R} \lambda_r \mathrm{E}[S_r^2]}{2(1 - \sum_{r=1}^{i-1} \lambda_r \mathrm{E}[S_r])(1 - \sum_{r=1}^{i} \lambda_r \mathrm{E}[S_r])}, \tag{A.9}$$

$$= \mathrm{E}[S_i] + \frac{\sum_{r=1}^{R} \lambda_r ((\mathrm{E}[S_i])^2 + \mathrm{V}[S_i])}{2(1 - \sum_{r=1}^{i-1} \lambda_r \mathrm{E}[S_r])(1 - \sum_{r=1}^{i} \lambda_r \mathrm{E}[S_r])}. \tag{A.10}$$

To obtain an upper bound on the average latency, we degrade the FJ system in the following manner. For a job of class i, the servers that have finished processing a sub-task of that job are blocked and do not accept new jobs until k_i sub-tasks of that job have been completed. Then the sub-tasks at remaining $n - k_i$ servers exit the system immediately. For jobs of class i, this performance-degraded system can be modeled as a M/G/1 system where the distribution of the service process, S_i, follows k_i^{th} ordered statistics as described in Section 5.2.2. Now for for any class i, the service time at each of the n servers is exponential with mean $1/\mu_i$. So the mean and variance of S_i are given by (A.7). Substituting (A.7) in (A.10), we get the following upper bound on average latency:

$$T^i_{\text{N-PQ}} \leq \overbrace{\underbrace{\frac{H^1_{n-k_i,n}}{\mu_i}}_{\text{service time}} + \underbrace{\frac{\sum_{r=1}^{R} \lambda_r [H^2_{n-k_r,n} + (H^1_{n-k_r,n})^2]/{\mu_r}^2}{2\,(1 - \mathcal{S}_{i-1})\,(1 - \mathcal{S}_i)}}_{\text{waiting time}}}^{\text{Average latency of degraded system}}, \tag{A.11}$$

[1]The decreasing order of class priority is $1 > 2 > \cdots > R$.

where $\mathcal{S}_i = \sum_{r=1}^{i} \rho_r H^1_{n-k_r,n}$ and $\rho_r = \lambda_r/\mu_r$.

A.2.3 Preemptive Priority Scheduling

Let S_i be a random variable representing the service time for a job of class i. Then the average latency for a job of class (priority level) i in a preemptive priority scheduling system[1] is given by Eq. 2.15 as,

$$T^i_{\mathrm{PQ}} = \frac{\mathrm{E}[S_i]}{1 - \sum_{r=1}^{i-1} \lambda_r \mathrm{E}[S_r]} + \frac{\sum_{r=1}^{i} \lambda_r \mathrm{E}[S_r^2]}{2\left(1 - \sum_{r=1}^{i-1} \lambda_r \mathrm{E}[S_r]\right)\left(1 - \sum_{r=1}^{i} \lambda_r \mathrm{E}[S_r]\right)}, \tag{A.12}$$

$$= \frac{\mathrm{E}[S_i]}{1 - \sum_{r=1}^{i-1} \lambda_r \mathrm{E}[S_r]} + \frac{\sum_{r=1}^{i} \lambda_r \left(\left(\mathrm{E}[S_i]\right)^2 + \mathrm{V}[S_i]\right)}{2\left(1 - \sum_{r=1}^{i-1} \lambda_r \mathrm{E}[S_r]\right)\left(1 - \sum_{r=1}^{i} \lambda_r \mathrm{E}[S_r]\right)}. \tag{A.13}$$

To obtain an upper bound on the average latency, we degrade the FJ system in the following manner. For a job of class i, the servers that have finished processing a sub-task of that job are blocked and do not accept new jobs until k_i sub-tasks of that job have been completed. Then the sub-tasks at remaining $n - k_i$ servers exit the system immediately. For jobs of class i, this performance-degraded system can be modeled as a M/G/1 system where the distribution of the service process, S_i, follows k_i^{th} ordered statistics as described in Section 5.2.2. Now for any class i, the service time at each of the n servers is exponential with mean $1/\mu_i$. Hence the mean and variance of S_i are given by (A.7). Substituting (A.7) in (A.13), we get the following upper bound on average latency:

$$T^i_{\mathrm{PQ}} \leq \overbrace{\underbrace{\frac{H^1_{n-k_i,n}}{\mu_i\left(1 - \mathcal{S}_{i-1}\right)}}_{\text{service time}} + \underbrace{\frac{\sum_{r=1}^{i} \lambda_r [H^2_{n-k_r,n} + (H^1_{n-k_r,n})^2]/{\mu_r}^2}{2\left(1 - \mathcal{S}_{i-1}\right)\left(1 - \mathcal{S}_i\right)}}_{\text{waiting time}}}^{\text{Average latency of degraded system}}. \tag{A.14}$$

where $\mathcal{S}_i = \sum_{r=1}^{i} \rho_r H^1_{n-k_r,n}$ and $\rho_r = \lambda_r/\mu_r$.

A.3 Proof of Theorem 5.2: Lower Bound on Average Latency

A.3.1 FCFS Scheduling

For the purpose of obtaining a lower bound on the average latency of class i, using insights from Section 5.2.1, we map the parallel processing in the proposed FJ system to a sequential process consisting of k_i processing stages for k_i sub-tasks of a job of class i. The transition from one stage to the next occurs when one of the remaining servers finishes a sub-task of the job. Let c_s denotes the number of classes that are finished before start of stage s, defined in (5.24). The processing in each stage s corresponds to a single-server FCFS system with jobs of all but classes $1, 2, \cdots, c_s$. Then, using (2.11) for the FCFS sub-system at stage s, the average latency for a sub-task of a job of class i in stage s is given by,

$$T^i_{\text{FCFS},s} = \mathrm{E}[S^s_i] + \frac{\lambda \mathrm{E}[(S^s)^2]}{2(1-\lambda \mathrm{E}[S^s]))}, \tag{A.15}$$

where S^s is a r.v. denoting the service time for any sub-task in stage s and S^s_i denotes the service time for a sub-task of class i in stage s. Now the moments of S^s and S^s_i are related to each other in the same way as the moments of S and S_i in (2.10). So we have,

$$\mathrm{E}[S^s] = \sum_{r=c_s+1}^{R} p_r \mathrm{E}[S^s_r], \quad \mathrm{E}[(S^s)^2] = \sum_{r=c_s+1}^{R} p_r \mathrm{E}[(S^s_r)^2]. \tag{A.16}$$

Substituting (A.16) in (A.15), we get

$$T^i_{\text{FCFS},s,c_s} = \mathrm{E}[S^s_i] + \frac{\sum\limits_{r=c_s+1}^{R} \lambda_r \mathrm{E}[(S^s_i)^2]}{2\left(1 - \sum\limits_{r=c_s+1}^{R} \lambda_r \mathrm{E}[S^s_i]\right)}. \tag{A.17}$$

Now we note that at any stage s, the maximum possible service rate for a job of class j that is not finished yet is $(n-s+1)\mu_j$. This happens when all the remaining sub-tasks of job of class j are at the head of their buffers. Thus, we can enhance the latency performance in each stage s by approximating it with a M/G/1 system with service rate $(n-s+1)\mu_j$ for jobs of class j. Then, the average latency for sub-task of job of class i in stage s is lower bounded as,

$$T^i_{\text{FCFS},s,c_s} \geq \frac{1}{(n-s+1)\mu_i} + \frac{\sum\limits_{r=c_s+1}^{R} \frac{\lambda_r}{(n-s+1)\mu_r^2}}{1 - \sum\limits_{r=c_s+1}^{R} \frac{\lambda_r}{(n-s+1)\mu_r}}, \tag{A.18}$$

Finally, the average latency for class i in this enhanced system is simply $\sum_{s=1}^{k_i} T^i_{\text{FCFS},s,c_s}$. This gives us

$$T^i_{\text{FCFS}} \geq \underbrace{\sum_{s=1}^{k_i} \frac{t_{s,i}}{\lambda_i}}_{\text{service time}} + \underbrace{\sum_{s=1}^{k_i} \left(\frac{\sum_{r=c_s+1}^{R} \frac{t_{s,r}}{(n-s+1)\mu_r}}{1 - \sum_{r=c_s+1}^{R} t_{s,r}} \right)}_{\text{waiting time}},$$

where $t_{s,i} = \frac{\lambda_i}{(n-s+1)\mu_i}$. This concludes the proof of lower bound on the average latency for FCFS scheduling.

A.3.2 Non-preemptive Priority Scheduling

For the purpose of obtaining a lower bound on the average latency of class i, using insights from Section 5.2.1, we map the parallel processing in the heterogeneous FJ system to a sequential process consisting of k_i processing stages for k_i sub-tasks of a job of class i. The transition from one stage to the next occurs when one of the remaining servers finishes a sub-task of the job. Since classes are relabeled such that $k_1 \leq k_2 \leq ... \leq k_R$, all jobs of classes 1 to i get finished when stage k_i is finished. However due to this relabeling of classes, this new class 1 is not necessarily the one with highest priority.

Let c_s denote the number of classes that are finished before start of stage s, given by (5.24). At any stage s, let $\mathcal{R}^i_s$ denote the set of classes with priority higher than class i and have atleast one sub-task remaining to be completed. Fig. A.2 illustrates the operational meaning of $\mathcal{R}^i_s$ using a toy example. Fig. 13(a) specifies the MDS codes corresponding to 5 data classes and suppose we are interested in the latency of class 3 marked in red. Fig. 13(b) shows the state of the system in stage $s = 1$. The classes are reordered in order of increasing k values. Since $s = 1$, no class is finished yet and $c_1 = 0$. The last column shows the set $\mathcal{R}^i_1$ for a class i with unfinished jobs. Fig. 13(c) shows the state of system at stage $s = 5$. Since $k_5 = 4$, all jobs of class 5 are finished and exits the system, so $c_5 = 1$. However since class 5 has lowest priority, $\mathcal{R}^i_5$ for a remaining class i is same as $\mathcal{R}^i_1$. Fig. 13(d) shows the state of the system at stage, $s = 6$. Since $k_2 = 5$, all jobs of class 2 are finished and exits the system, so $c_6 = 2$. Since class 2 is higher priority than class 3, $\mathcal{R}^3_6$ is now $\{1\}$.

Now using (A.9), we get the mean completion time for sub-task of a job of class i in stage s as,

$$T^i_{\text{N-PQ},s,c_s,\mathcal{R}^i_s} = \mathrm{E}[S^s_i] + \frac{\sum_{r=c_s+1}^{R} \lambda_r \mathrm{E}[(S^s_r)^2]}{2\left(1 - \sum_{r\in\mathcal{R}^i_s} \lambda_r \mathrm{E}[S^s_r]\right)\left(1 - \lambda_i \mathrm{E}[S^s_i] - \sum_{r\in\mathcal{R}^i_s} \lambda_r \mathrm{E}[S^s_r]\right)}, \tag{A.19}$$

Class Id (i)	MDS code
1	(10, 8)
2	(10, 5)
3	(10, 6)
4	(10, 7)
5	(10, 4)

(a)

Class Id (i)	Subtasks left	R_1^i
5	4	{1,2,3,4}
2	5	{1}
3	6	{1,2}
4	7	{1,2,3}
1	8	{ϕ}

(b)

Class Id (i)	Subtasks left	R_5^i
2	1	{1}
3	2	{1,2}
4	3	{1,2,3}
1	4	{ϕ}

(c)

Class Id (i)	Subtasks left	R_6^i
3	1	{1}
4	2	{1,3}
1	3	{ϕ}

(d)

Figure A.2: Example illustrating the operational meaning of variable $\mathcal{R}_s^i$, where i is the class id and s is the processing stage. The decreasing order of class priority is $1 > 2 > 3 > 4 > 5$.

where S_i^s is a random variable denoting the service time for a sub-task of class i in stage s. Unlike (A.9), the summation in the numerator starts from $c_s + 1$ because classes 1 through c_s have been completed at stage s. Also unlike (A.9), the summation in the denominator is over the set $\mathcal{R}_s^i \subseteq \{1, 2, \cdots, i-1\}$ because some of the higher priority (relative to class i) classes may have been finished at start of stage s.

Now at any stage s, the maximum possible service rate for a job of class j that is not finished yet is $(n-s+1)\mu_j$. This happens when all the remaining sub-tasks of job of class j are at the head of their buffers. Thus, we can enhance the latency performance in each stage s by approximating it with a M/M/1 system with service rate $(n-s+1)\mu_j$ for jobs of class j. The average latency for sub-task of job of class i in stage s is thus lower bounded as,

$$T^i_{\text{N-PQ},s,c_s,\mathcal{R}_s^i} \geq \overbrace{\underbrace{\frac{1}{(n-s+1)\mu_i}}_{\text{service time}} + \underbrace{\frac{\sum_{r=c_s+1}^{R} \frac{\lambda_r}{(n-s+1)\mu_r^2}}{\left(1 - \sum_{r\in\mathcal{R}_s^i} \frac{\lambda_r}{(n-s+1)\mu_r}\right)\left(1 - \frac{\lambda_i}{(n-s+1)\mu_i} - \sum_{r\in\mathcal{R}_s^i} \frac{\lambda_r}{(n-s+1)\mu_r}\right)}}^{\text{waiting time}}}^{\text{Average latency of enhanced system}}.$$

Finally, the average latency for class i in this enhanced system is simply $\sum_{s=1}^{k_i} T^i_{\text{N-PQ},s,c_s,\mathcal{R}_s^i}$.

Thus we have,

$$T^i_{\text{N-PQ}} \geq \sum_{s=1}^{k_i} \left(\frac{t_{s,i}}{\lambda_i} + \frac{\sum_{r=c_s+1}^{R} \frac{t_{s,r}}{(n-s+1)\mu_r}}{\mathcal{Z}^i_s \left(\mathcal{Z}^i_s - t_{s,i} \right)} \right), \tag{A.20}$$

where $t_{s,i} = \frac{\lambda_i}{(n-s+1)\mu_i}$ and $\mathcal{Z}^i_s = 1 - \sum_{r \in \mathcal{R}^i_s} t_{s,r}$.

A.3.3 Preemptive Priority Scheduling

For the purpose of obtaining a lower bound on the average latency of class i, using insights from [100], we map the parallel processing in the heterogeneous FJ system to a sequential process consisting of k_i processing stages for k_i sub-tasks of a job of class i. The transition from one stage to the next occurs when one of the remaining servers finishes a sub-task of the job. Since classes are relabeled such that $k_1 \leq k_2 \leq ... \leq k_R$, all jobs of classes 1 to i get finished when stage k_i is finished. However due to this relabeling of classes, this new class 1 is not necessarily the one with highest priority.

Let c_s denote the number of classes that are finished before start of stage s, given by (5.24). At any stage s, let $\mathcal{R}^i_s$ denote the set of classes with priority higher than class i and have atleast one sub-task remaining to be completed. The operational meaning of R^i_s is explained in Fig. A.2. Then using (A.12), the mean completion time for sub-task of a job of class i in stage s is given by,

$$T^i_{\text{PQ},s,c_s,\mathcal{R}^i_s} = \frac{\text{E}[S^s_i]}{1 - \sum_{r \in \mathcal{R}^i_s} \lambda_r \text{E}[S^s_r]} + \frac{\lambda_i \text{E}[(S^s_i)^2] + \sum_{r \in \mathcal{R}^i_s} \lambda_r \text{E}[(S^s_r)^2]}{2 \left(1 - \sum_{r \in \mathcal{R}^i_s} \lambda_r \text{E}[S^s_r] \right) \left(1 - \lambda_i \text{E}[S^s_i] - \sum_{r \in \mathcal{R}^i_s} \lambda_r \text{E}[S^s_r] \right)}, \tag{A.21}$$

where S^s_i is a random variable denoting the service time for a sub-task of class i in stage s. Unlike (A.12), the summation terms in the numerator and denominator are over the set $\mathcal{R}^i_s \subseteq \{1, 2, \cdots, i-1\}$ because some of the higher priority (relative to class i) classes may have been finished at start of stage s.

Now we note that at any stage s, the maximum possible service rate for a job of class j that is not finished yet is $(n-s+1)\mu_j$. This happens when all the remaining sub-tasks of job of class j are at the head of their buffers. Thus, we can enhance the latency performance in each stage s by approximating it with a M/M/1 system with service rate $(n-s+1)\mu_j$ for jobs of class j. The average latency for sub-task of job of class i in stage s is thus lower

bounded as $T^i_{\mathrm{PQ},s,c_s,\mathcal{R}^i_s} \geq$,

$$\overbrace{\underbrace{\frac{1}{(n-s+1)\mu_i\left(1-\sum\limits_{r\in\mathcal{R}^i_s}\frac{\lambda_r}{(n-s+1)\mu_r}\right)}}_{\text{service time}}+\underbrace{\frac{\frac{\lambda_i}{(n-s+1)\mu_i^2}+\sum\limits_{r\in\mathcal{R}^i_s}\frac{\lambda_r}{(n-s+1)\mu_r^2}}{\left(1-\sum\limits_{r\in\mathcal{R}^i_s}\frac{\lambda_r}{(n-s+1)\mu_r}\right)\left(1-\frac{\lambda_i}{(n-s+1)\mu_i}-\sum\limits_{r\in\mathcal{R}^i_s}\frac{\lambda_r}{(n-s+1)\mu_r}\right)}}_{\text{waiting time}}}^{\text{Average latency of enhanced system}}.$$

Finally, the average latency for class i in this enhanced system is simply $\sum\limits_{s=1}^{k_i} T^i_{\mathrm{PQ},s,c_s,\mathcal{R}^i_s}$. Thus we have,

$$T^i_{\mathrm{PQ}} \geq \sum_{s=1}^{k_i}\left(\frac{t_{s,i}}{\lambda_i\mathcal{Z}^i_s}+\frac{1-\mathcal{Z}^i_s+\frac{t_{s,i}}{(n-s+1)\mu_i}}{\mathcal{Z}^i_s\left(\mathcal{Z}^i_s-t_{s,i}\right)}\right), \tag{A.22}$$

where $t_{s,i}=\frac{\lambda_i}{(n-s+1)\mu_i}$ and $\mathcal{Z}^i_s = 1-\sum\limits_{r\in\mathcal{R}^i_s} t_{s,r}$.

Bibliography

[1] "Technical Specification Group Services and System Aspects; Policy and charging control architecture," 3GPP, Tech. Rep. TS 23.203 V8.9.0, March 2010.

[2] R. Madan and S. Ray, "Uplink resource allocation for frequency selective channels and fractional power control in lte," in *IEEE International Conference on Communications*, June 2011, pp. 1–5.

[3] D. Gesbert, S. G. Kiani, A. Gjendemsjo, and G. E. Oien, "Adaptation, Coordination, and Distributed Resource Allocation in Interference-Limited Wireless Networks," *Proceedings of the IEEE*, vol. 95, no. 12, pp. 2393–2409, Dec 2007.

[4] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter, "A survey on networking games in telecommunications," *Elsevier Computer and Operations Research*, vol. 33, no. 2, pp. 286–311, Feb. 2006.

[5] A. Feki and V. Capdevielle, "Autonomous resource allocation for dense lte networks: A multi armed bandit formulation," in *IEEE Personal Indoor and Mobile Radio Communications*, Sept 2011, pp. 66–70.

[6] A. Subramanian, M. Al-Ayyoub, H. Gupta, S. Das, and M. Buddhikot, "Near-optimal dynamic spectrum allocation in cellular networks," in *IEEE DySPAN*, Oct 2008, pp. 1–11.

[7] I. Ahmed, M. Butt, C. Psomas, A. M. I. Krikidis, and M. Guizani, "Survey on Energy Harvesting Wireless Communications: Challenges and Opportunities for Radio Resource Allocation," *Elsevier Computer Networks*, vol. 88, pp. 234–248, Sep. 2015.

[8] A. Annunziato, "5G Vision: NGMN - 5G Initiative," in *IEEE Vehicular Technology Conference*, May 2015, pp. 1–5.

[9] "M2M White Paper: The Growth of Device Connectivity," The FocalPoint Group, Tech. Rep., May 2010.

[10] "Small cells, whats the big idea?" Small Cell Forum, Tech. Rep., Feb 2014.

[11] N. Golrezaei, A. Molisch, A. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, Apr. 2013.

[12] "Spectrum policy task force report," *FCC*, no. 02-155, 2002.

[13] S. Ali and V. Leung, "Dynamic frequency allocation in fractional frequency reused ofdma networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 8, pp. 4286 –4295, Aug. 2009.

[14] N. Saquib, E. Hossain, and D. I. Kim, "Fractional frequency reuse for interference management in lte-advanced hetnets," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 113–122, 2013.

[15] T. Novlan, J. Andrews, I. Sohn, R. Ganti, and A. Ghosh, "Comparison of fractional frequency reuse approaches in the ofdma cellular downlink," in *IEEE GLOBECOM*, Dec. 2010, pp. 1 –5.

[16] "Cisco visual networking index: Global mobile data traffic forecast update, 20142019," Feb 2014. [Online]. Available: http://goo.gl/ULXROo

[17] S. Hur, T. Kim, D. Love, J. Krogmeier, T. Thomas, and A. Ghosh, "Millimeter wave beamforming for wireless backhaul and access in small cell networks," *IEEE Transactions on Communications*, vol. 61, no. 10, pp. 4391–4403, October 2013.

[18] Y. Mao, Y. Luo, J. Zhang, and K. B. Letaief, "Energy harvesting small cell networks: feasibility, deployment, and operation," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 94–101, 2015.

[19] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *IEEE INFOCOM*, March 2012, pp. 1107–1115.

[20] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive content download and user demand shaping for data networks," *Accepted for Publication in IEEE/ACM Transactions on Networking*.

[21] M. Ji, G. Caire, and A. Molisch, "Fundamental limits of distributed caching in d2d wireless networks," in *IEEE Information Theory Workshop*, Sept 2013, pp. 1–5.

[22] A. Pingyod and Y. Somchit, "Content updating method in femtocaching," in *International Joint Conference on Computer Science and Software Engineering (JCSSE)*, May 2014, pp. 123–127.

[23] O. Ozel, K. Tutuncuoglu, J. Yang, S. Ulukus, and A. Yener, "Transmission with energy harvesting nodes in fading wireless channels: Optimal policies," *arXiv*, vol. abs/1106.1595, 2011. [Online]. Available: http://arxiv.org/abs/1106.1595

[24] H. S. Dhillon, H. Huang, H. Viswanathan, and R. A. Valenzuela, "Fundamentals of Throughput Maximization With Random Arrivals for M2M Communications," *IEEE Transactions on Communications*, vol. 62, no. 11, pp. 4094–4109, Nov 2014.

[25] E. Gelenbe, "A sensor node with energy harvesting," *SIGMETRICS*, vol. 42, no. 2, pp. 37–39, Sep 2014.

[26] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube Network Traffic at a Campus Network - Measurements, Models, and Implications," *Compuer Networks*, vol. 53, no. 4, pp. 501–514, March 2009.

[27] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, Oct 2003.

[28] G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE Journal on Selected Areas in Communication*, May 2014.

[29] N. B. Shah, K. Lee, and K. Ramchandran, "The MDS queue," *arXiv*, vol. abs/1211.5405, 2012.

[30] N. Shah, K. Lee, and K. Ramchandran, "When do redundant requests reduce latency ?" in *Allerton*, Oct 2013, pp. 731–738.

[31] J. Levandoski, P.-A. Larson, and R. Stoica, "Identifying hot and cold data in main-memory databases," in *Proc. of IEEE International Conference on Data Engineering*, April 2013, pp. 26–37.

[32] D. Gibson, "Is your data hot, warm, or cold ?" IBM Data Magazine, 2012. [Online]. Available: http://ibmdatamag.com/2012/06/is-your-big-data-hot-warm-or-cold/

[33] "How aws pricing works," July 2014. [Online]. Available: http://media.amazonwebservices.com/AWS_Pricing_Overview.pdf

[34] "Google cloud storage - pricing," May 2010. [Online]. Available: https://cloud.google.com/storage/docs/storage-classes

[35] J. J. Nielsen, G. C. Madueo, N. K. Pratas, R. B. Srensen, C. Stefanovic, and P. Popovski, "What can wireless cellular technologies do about the upcoming smart metering traffic?" *IEEE Communications Magazine*, vol. 53, no. 9, pp. 41–47, September 2015.

[36] E. Hossain, Z. Han, and H. V. Poor, *Smart Grid Communications and Networking*. Cambridge University Press, 2012.

[37] "Study on RAN Improvements for Machine-type communications," 3GPP, Tech. Rep. TR 37.868, 2012.

[38] A. Gotsis, A. Lioumpas, and A. Alexiou, "Evolution of packet scheduling for machine-type communications over lte: Algorithmic design and performance analysis," in *IEEE Globecom Workshop*, Dec 2012, pp. 1620–1625.

[39] S.-Y. Lien and K.-C. Chen, "Massive Access Management for QoS Guarantees in 3GPP Machine-to-Machine Communications," *IEEE Communications Letters*, vol. 15, no. 3, pp. 311–313, March 2011.

[40] G. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, 3rd ed. New York: Springer, 2011.

[41] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," in *SIGCOMM*, 1989.

[42] A. Greenberg and N. Madras, "How Fair is Fair Queueing?" in *Proc. Performance*, 1990.

[43] J. Nagle, "On Packet Switches with Infinite Storage," *IEEE Transactions on Communications*, vol. 35, no. 4, pp. 435–438, Apr 1987.

[44] M. Shreedhar and G. Varghese, "Efficient Fair Queuing Using Deficit Round-Robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, Jun 1996.

[45] N. Nikaein, M. Laner, K. Zhou, P. Svoboda, D. Drajic, M. Popovic, and S. Krco, "Simple traffic modeling framework for machine type communication," in *International Symposium on Wireless Communication Systems*, Aug 2013, pp. 1–5.

[46] E. Çinlar, "Exceptional paper—markov renewal theory: A survey," *INFORMS Management Science*, vol. 21, no. 7, pp. 727–752, Mar. 1975.

[47] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

[48] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2009.

[49] H. W. Kuhn and A. W. Tucker, "Nonlinear Programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, 1951, pp. 481–492.

[50] G. B. Dantzig, "Discrete-Variable Extremum Problems," *Operations Research*, vol. 5, no. 2, pp. 266–277, 1957.

[51] D. Pisinger, "Algorithms for Knapsack Problems," Ph.D. dissertation, University of Copenhagen, Denmark, 1995.

[52] R. Bellman, "The Theory of Dynamic Programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 11 1954.

[53] ——, *Dynamic Programming.* Princeton University Press, 1957.

[54] M. L. Puterman, *Markov Decision Processes: discrete stochastic dynamic programming.* Wiley, 1994.

[55] E. Pashenkova, I. Rish, and R. Dechter, "Value iteration and policy iteration algorithms for markov decision problem," 1996.

[56] S. J. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[57] S. Boyd and J. Mattingley, "Branch and bound methods," 2003.

[58] "A Proof for the Queuing Formula: L = W," *Operations Research*, vol. 9, no. 3, pp. 383–387, 1961.

[59] D. Bertsekas and R. Gallager, *Data Networks (2nd Ed.).* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992, ch. Delay Models in Data Networks, pp. 203–206.

[60] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain," *The Annals of Mathematical Statistics*, vol. 24, no. 3, pp. 338–354, 09 1953.

[61] F. Pollaczek, "Über eine aufgabe der wahrscheinlichkeitstheorie. i," *Mathematische Zeitschrift*, vol. 32, no. 1, pp. 64–100, 1930.

[62] A. Y. Khintchine, "Mathematical theory of a stationary queue," *Matematicheski Sbornik*, vol. 39, pp. 73–84, 1932.

[63] J. F. C. Kingman, "The single server queue in heavy traffic," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 57, pp. 902–904, 10 1961.

[64] J. Mitola III, "Software radios-survey, critical evaluation and future directions," in *National Telesystems Conference*, May 1992, pp. 13/15 –13/23.

[65] T. Clancy, "Achievable capacity under the interference temperature model," in *IEEE International Conference on Computer Communications*, May 2007, pp. 794 –802.

[66] V. Osa, C. Herranz, J. Monserrat, and X. Gelabert, "Implementing opportunistic spectrum access in lte-advanced," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012:99, no. 1, 2012.

[67] R. Kwan, C. Leung, and J. Zhang, "Resource allocation in an lte cellular communication system," in *IEEE International Conference on Communications*, 2009, pp. 3915–3919.

[68] A. G. Marqus, L. M. Lopez-Ramos, G. B. Giannakis, and J. Ramos, "Resource allocation for interweave and underlay CRs under probability-of-interference constraints." *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 10, pp. 1922–1933, 2012.

[69] L. M. Lopez-Ramos, A. G. Marqués, and J. Ramos, "Jointly optimal sensing and resource allocation for multiuser overlay cognitive radios," *IEEE Journal on Selected Areas in Communications (submitted)*, 2012.

[70] C.-H. Chang, H.-L. Chao, and C.-L. Liu, "Sum throughput-improved resource allocation for lte uplink transmission," in *IEEE Vehicular Technology Conference*, Sept 2011, pp. 1–5.

[71] "Physical layer aspects for Evolved UTRA, (release 7)." 3GPP, Tech. Rep. TR 25.814, Oct. 2007.

[72] W. Xuan-li, Z. Wan-jun, and W. Wei, "Throughput and fairness-balanced resource allocation algorithm in td-lte-advanced relay-enhanced network," in *International Workshop on High Mobility Wireless Communications (HMWC)*, Nov 2013, pp. 82–86.

[73] A. Abdel-Hadi and C. Clancy, "A utility proportional fairness approach for resource allocation in 4g-lte," in *International Conference on Computing, Networking and Communications*, Feb 2014, pp. 1034–1040.

[74] M. Iturralde, A. Wei, T. Ali Yahiya, and A.-L. Beylot, "Resource allocation for real time services using cooperative game theory and a virtual token mechanism in lte networks," in *IEEE Consumer Communications and Networking Conference*, Jan 2012, pp. 879–883.

[75] S. Vatsikas, S. Armour, M. De Vos, and T. Lewis, "A fast and fair algorithm for distributed subcarrier allocation using coalitions and the nash bargaining solution," in *IEEE Vehicular Technology Conference*, Sept 2011, pp. 1–5.

[76] L. Anchora, L. Badia, and M. Zorzi, "A framework for scheduling and resource allocation in lte downlink using nash bargaining theory," in *International Conference on Communications*, June 2011.

[77] J. Deaton, R. Irwin, and L. DaSilva, "The effects of a dynamic spectrum access overlay in lte-advanced networks," in *IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, 2011, pp. 488–497.

[78] S. Sen, T. Zhang, M. Buddhikot, S. Banerjee, D. Samardzija, and S. Walker, "A dual technology femto cell architecture for robust communication using whitespaces," in *IEEE DySPAN*, Oct 2012, pp. 242–253.

[79] A. Amanna, J. Mitola, R. Tandon, T. Clancy, J. Reed, R. McGwier, A. Sengupta, and A. Kumar, "Method for heterogeneous spectrum sharing between commercial cellular operators and legacy incumbent users in wireless networks," US utility filing 13/835,012, March 2013.

[80] A. Karandikar. (2013) Regulation in spectrum. [Online]. Available: http://www.nicf.gov.in/pdf/spectrum_management/regulation_in_spectrum.pdf

[81] S. Das (ed.), J. Malyar, and D. Joslyn. (2012) Device to database protocol for white space draft-das-paws-protocol-02. Internet draft. [Online]. Available: http://tools.ietf.org/html/draft-das-paws-protocol-02

[82] D. Arianandа, M. K. Lakshmanan, and H. Nikookar, "A survey on spectrum sensing techniques for cognitive radio," in *International Workshop on Cognitive Radio and Advanced Spectrum Management*, May 2009, pp. 74–79.

[83] J. Shen, S. Liu, R. Zhang, and Y. Liu, "Soft versus hard cooperative energy detection under low snr," in *Third International Conference on Communications and Networking*, Aug. 2008, pp. 128–131.

[84] P. K. Varshney, *Distributed Detection and Data Fusion.* New York:Springer-Verlag, 1997.

[85] S. Sesia, I. Toufik, and M. Baker, *LTE, The UMTS Long Term Evolution: From Theory to Practice*, ser. Wiley InterScience online books. Wiley, 2009.

[86] T. Novlan, R. Ganti, A. Ghosh, and J. Andrews, "Analytical evaluation of fractional frequency reuse for ofdma cellular networks," *IEEE Transactions on Wireless Communications*, vol. 10, no. 12, pp. 4294 –4305, Dec. 2011.

[87] "Evolution of land mobile radio (including personal communication)," ICT COST Action 231, 1993. [Online]. Available: http://cordis.europa.eu/project/rcn/16733_en.html

[88] M. Alshami, T. Arslan, J. Thompson, and A. Erdogan, "Evaluation of path loss models at wimax cell-edge," in *IFIP International Conference on New Technologies, Mobility and Security*, Feb. 2011, pp. 1–5.

[89] J. Seybold, *Introduction to RF Propagation.* Wiley, 2005.

[90] Ericsson, "System-level evaluation of ofdm - further considerations," *TSG-RAN WG1 35, Lisbon, Portugal, TR R1-031303*, Nov 2003.

[91] L. Wan, S. Tsai, and M. Almgren, "A fading-insensitive performance metric for a unified link quality model," in *IEEE Wireless Communications and Networking Conference*, vol. 4, April 2006, pp. 2110 –2114.

[92] U. Faigle, W. Kern, and G. Still, "Integer programming," in *Algorithmic Principles of Mathematical Programming*, ser. Kluwer Texts in the Mathematical Sciences. Springer Netherlands, 2002, vol. 24, pp. 173–196.

[93] "Ericsson energy and carbon report," Nov 2014. [Online]. Available: http://goo.gl/d3Qoz3

[94] N. Shah, K. Rashmi, P. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *IEEE Information Theory Workshop*, Jan 2010, pp. 1–5.

[95] V. Cadambe, S. Jafar, and H. Maleki, "Minimum repair bandwidth for exact regeneration in distributed storage," in *IEEE Wireless Network Coding Conference*, June 2010, pp. 1–6.

[96] K. V. Rashmi, N. Shah, and P. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5227–5239, Aug 2011.

[97] G. Liang and U. Kozat, "Tofec: Achieving optimal throughput-delay trade-off of cloud storage using erasure codes," in *Proc. of IEEE INFOCOM*, April 2014, pp. 826–834.

[98] G. Schulz, *The Green and Virtual Data Centre*. New York, NY: CRC/Auerbach, 2009, ch. Measurement, Metrics, and Management of IT resources.

[99] Y. Liu, S. Draper, and N. S. Kim, "Sleepscale: Runtime joint speed scaling and sleep states management for power efficient data centers," in *IEEE International Symposium on Computer Architecture*, June 2014, pp. 313–324.

[100] E. Varki, A. Merchant, and H. Chen, "The M/M/1 fork-join queue with variable sub-tasks," March 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.100.3062

[101] S. Ross, *A first course in probability*. Prentice-Hall Inc., 2002.

[102] M. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, Dec 1997.

[103] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 251–262, Aug 1999.

[104] R. Pratt. (2011, July) Real-Time Pricing and Demand Response. https://goo.gl/2TwH0c.

[105] A. Abdelhadi and T. Clancy, "A Utility Proportional Fairness Approach for Resource Allocation in 4G-LTE," in *IEEE International Conference on Computing, Networking, and Communications (ICNC), CNC Workshop*, 2014.

[106] ——, "A Robust Optimal Rate Allocation Algorithm and Pricing Policy for Hybrid Traffic in 4G-LTE," in *IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2013.

[107] T. Tia, J. Liu, and M. Shankar, "Algorithms and optimality of scheduling soft aperiodic requests in fixed-priority preemptive systems," *Real-Time Systems*, vol. 10, pp. 23–43, Jan 1996.

[108] L. Schrage, "A Proof of the Optimality of the Shortest Remaining Processing Time Discipline," *Operations Research*, vol. 16, no. 3, pp. 687–690, 1968.

[109] GSMA Intelligence. (2014, Feb.) From concept to delivery: the M2M market today. https://goo.gl/yFmi5s.

[110] Machina Research. (2015, May) M2M growth necessitates a new approach to network planning and optimisation. https://goo.gl/CDmd6M.

[111] P. J. Burke, "The Output of a Queuing System," *Operations Research*, vol. 4, no. 6, pp. 699–704, Dec 1956.

[112] J. J. Nielsen, G. C. Madueo, N. K. Pratas, R. B. Srensen, C. Stefanovic, and P. Popovski, "What can wireless cellular technologies do about the upcoming smart metering traffic?" *IEEE Communications Magazine*, vol. 53, no. 9, pp. 41–47, September 2015.

[113] Aeris. (2016) AerCloud M2M Cloud Platform / IoT Platform. [Online]. Available: http://www.aeris.com/technology/aercloud/