A Reference Model and Architecture for Future Computer Networks

Hoda Mamdouh Hassan

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
Computer Engineering

Mohamed Eltoweissy, Chair
Scott Midkiff
Luiz DaSilva
Ing-Ray Chen
Amir Zaghloul
Moustafa Youssef

May 26th, 2010
Blacksburg, Virginia

Keywords: Bio-inspired Computer Network Design, Complex Adaptive Systems, Computer Network Architecture, Network Reference Model, Protocol Design

A Reference Model and Architecture for Future Computer Networks

Hoda Mamdouh Hassan

ABSTRACT

The growing need for a trustworthy Future Internet demands evolutionary approaches unfettered by legacy constrains and concepts. The networking community is calling for new network architectural proposals that address the deficiencies identified in present network realizations, acknowledge the need for a trustworthy IT infrastructure, and satisfy the society's emerging and future requirements. Proposed architectures need to be founded on well-articulated design principles, account for network operational and management complexities, embrace technology and application heterogeneity, regulate network-inherent emergent behavior, and overcome shortcomings attributed to present network realizations.

This dissertation presents our proposed clean-slate Concern-Oriented Reference Model (CORM) for architecting future computer networks. CORM stands as a guiding framework from which network architectures can be derived according to specific functional, contextual, and operational requirements or constraints. CORM represents a pioneering attempt within the network realm, and to our knowledge, CORM is the first reference model that is bio-inspired and derived in accordance with the Function-Behavior-Structure (FBS) engineering framework.

CORM conceives a computer network as a software-dependent complex system whose design needs to be attempted in a concern-oriented bottom-up approach along two main dimensions: a vertical dimension addressing structure and configuration of network building blocks; and a horizontal dimension addressing communication and interactions among the previously formulated building blocks. For each network dimension, CORM

factors the design space into function, structure, and behavior, applying to each the principle of separation of concerns for further systematic decomposition.

In CORM, the network-building block is referred to as the Network Cell (NC), which represents CORM's first basic abstraction. An NC's structure and inherent behavior are bio-inspired, imitating a bacterium cell in a bacteria colony, thus it is capable of adaptation, self-organization and evolution. An NC's functional operation is defined by CORM second basic abstraction; the ACRF framework. The ACRF framework is a conceptual framework for network-concerns derived according to our interpretation of computer network requirement specifications. CORM networks are recursively synthesized in a bottom-up fashion out of CORM NCs. CORM addresses the multi-dimensionality of computer networks by modeling the network structure and behavior using a network structural template (NST), and an information flow model (IFM), respectively. Being developed according to a complex system paradigm, CORM refutes the long endorsed concept of layering, intrinsically accounts for emergent behavior, and ensures system integrity and stability.

As a reference model, CORM is more typical of conventional engineering. Therefore it was validated using the FBS engineering framework. However, the behavior to be realized in CORM-based networks was substantiated and evaluated by deriving CellNet, our proposed CORM-based network architecture. CellNet-compliant protocols' behavioral adaptation and modification were illustrated and evaluated through simulation.

CORM will have a profound impact on the operation and behavior of computer networks composing the Internet. By introducing awareness adaptability and evolvability as network intrinsic features, CORM-based Internet will proactively respond to changes in operational contexts, underlying technologies, and end user requirements. A major direction in CORM future work would be to detail the IFM component.

Dedication

To my caring parents,
who taught me the essence of life,
To my loving husband,
who shares my woes and triumphs,
To Lobna, Ahmed and Abdel Badie
whose smiles shine my days,
With Love and devotion
I dedicate to thee.

Acknowledgements

Begin in the name of God most Gracious most Merciful

First and foremost, all thanks are due to Allah. None of this would have been possible if it was not for GOD's blessing grace and mercy. So all prays and thanks are to GOD most Gracious most Merciful.

To me attaining a PhD degree was a far fetched dream, however, with the help, assistant and encouragement of a lot of people, my far fetched dream turned to reality, and now many thanks are due.

I would like to thank my dear parents for their continuous support and care that they have willingly devoted for me and my family throughout this long journey. I really appreciate their efforts, and I hope that my achievements would make them proud of me. I also would like to thank my dear sister, who has always been praying for me.

To my loving, caring, and wonderful husband, Mahmoud; you are a blessing from GOD. I have to admit that I owe to you all what I have achieved. It is you, after God, who made this dream possible. Your continuous support, care, and help have always inspired me and gave the strength to go on. No word can express how grateful I feel for your support and care.

To my loving and wonderful children, Ahmed, Abdel Badie, and Lobna, I really owe you a lot of thanks for bearing with me throughout this journey. I know I was not present for you as much as I should, but you were always understanding, bearing with me, and willing to help.

I was very fortunate to have the opportunity to work with Dr. Mohamed Eltoweissy as my advisor. Dr Mohamed has always managed to stimulate my thinking, inspire me to explore new horizons, and challenged me to sharpen my thoughts and endeavors. Dr Mohamed has greatly impacted my reasoning and exalted my research skills. I will always remember our long discussions, during which Dr Mohamed was very patient and instructive. His insightful comments has always enlightened my perspectives,

and had a profound effect on the quality of my work. Dr Mohamed I will always be grateful to you.

I would like to thank all my committee members for their guidance and comments, which have certainly improved the quality of my work, and I appreciate the time and effort that they alloted for me, in spite of their busy schedule.

Very special thanks goes to Dr Sedki Riad, who has initiated the VT-MENA program. Dr. Sedki you have made my dream possible. If it was not for the VT-MENA program I would have never had a chance to acquire a PhD degree from such a respectable university, nor would I have had the chance to enjoy learning and researching in such a compelling environment as that present in Virginia Tech.

I am also very grateful and indebted to Dr. Yasser Hanafy, the VT-MENA program director in Egypt. It was Dr Yasser who admitted me to the VT-MENA program. So if it was Dr Sedki who made my dream possible, it was Dr Yasser who made my dream come true. Dr Yasser I am very thankful for your efforts and endeavors to make this program a success here in Egypt, in spite of all the obstacles and difficulties you faced. I would also like to extend my thanks to all the VT-MENA professors who taught me during my course work.

Last, but never least I would like to thank Cindy Hopkins for her support and help in dealing with a lot of logistics, and for her patience in answering my questions and clarifying a lot of the administrative issues.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Think outside the limitations of existing systems – imagine what might be possible"

Vinton G. Cerf

The research in computer networks is at a critical turning point. The research community is endeavoring to devise future network architectures that address the deficiencies identified in present network realizations, acknowledge the need for a trustworthy IT infrastructure, and satisfy the society's emerging and future requirements [1, 2, 3]. Considering the lessons from the past and evaluating the outcomes and contributions of contemporary research literature, the community concluded that the advent of a trustworthy Future Internet cannot be achieved "by the current trajectory of incremental changes" and point solutions to the current Internet, but rather "more revolutionary paths need to be explored" [1, 2, 3]. Proposed architectures need to be grounded on well-articulated design principles, account for network operational and management complexities, embrace technology and application heterogeneity, regulate network-inherent emergent behavior, and overcome shortcomings attributed to present network realizations.

Resorting to revolutionary paths suggests the inadequacy of currently adopted concepts, principles, mechanisms, and/or implementations. Analyzing the path that led to the current state of the art, we postulate that present network realizations are the outcome of incremental research efforts and endeavors exerted during the inchoative stage of computer network design aiming to interconnect architecturally disparate networks into one global network. That goal was achieved through the introduction of the Transmission Control Protocol (TCP) [4]. TCP was introduced as a flexible protocol allowing for inter-process communication across networks, while hiding the underlying network differences. TCP was later split into TCP and IP leading to the derivation of the layered Internet TCP/IP suite. As such, the TCP/IP suite defined the Internet system, which was regarded as a vehicle to interconnect diverse types of networks. However, the astounding success of the TCP/IP suite in interconnecting networks resulted in adopting the TCP/IP suite as the de facto standard for inter-computer communication within networks, as well as across different networks, diminishing the need for a separate research addressing the requirements and specifications for internally designing networks. Focusing primarily on interconnection, TCP/IP networks possessed intelligence at the network edges, while regarding the network core as a "dump forwarding machine", thus introducing the End-to-End (E2E) design principle, a fundamental principle for TCP/IP networks [5]. Influenced by both the TCP/IP layered architecture and the E2E design principle, network designers and protocol engineers conformed to a top-down design strategy as the approach to architect networks. Moreover, with the introduction of the layered OSI

model, the top-down layered approach in network design and protocol engineering was emphasized further, in spite of the fact that the OSI was developed as an "Interconnection Architecture", i.e. an architecture facilitating the interaction of heterogeneous computer networks rather than an architecture for building computer networks [6].

Despite the outstanding success of its realizations, we argue that the Internet-layered model was deficient in representing essential network aspects necessary for network design and subsequent protocol engineering, which led to the shortcomings evident in present networks. First, the traditional "cloud model" derived from the E2E principle abstracts the Internet layout as core and edge, thus failing to express the network's topological, social, or economical boundaries. Second, resource management as a function is absent from the Internet-layered model, resulting in introducing point solutions to handle resource-management functions such as admission control, traffic engineering, and quality of service. Furthermore, the Internet-layered model prohibits vertical function integration, which is an obstacle when designing performance aspects that need to span across layers, resulting in numerous proposals for cross-layer designs. Moreover, the Internet model does not express network behavior nor allow for its customization according to context and requirements. A deficiency that led to two undesirable effects: (1) IP-based networks exhibit a defining characteristic of unstable complex systems— a local event can have a destructive global impact [7], and (2) lack of support for mobility, security, resilience, survivability, etc.— main features for a pervasive trustworthy infrastructure.

In response to observed network liabilities, research efforts have sprouted a plethora of architectural proposals aiming to overcome shortcomings evident in network realizations. Two main approaches can be identified. The first preserves the layered protocol structure, yet allows for vertical integration between protocols operating at different layers to induce context awareness and adaptability. The second approach targets clean-slate designs, primarily ignoring the need for backward compatibility. Nevertheless, we argue that the majority of contemporary architectural proposals remain idiosyncratic, satisfying a particular service model, tailored to specific set of requirements, or fettered by implicit assumptions [8]. Targeting causes rather than symptoms, we opine that deficiencies apparent in current network realizations can be traced back to the following underlying causes.

- The general trend towards network science and engineering lacks a systematic formalization of core principles that expresses essential network features required to guide the process of network design and protocol engineering;
- The prevalence of a top-down design approach for network architecture demonstrated as confining intelligence to network edges, and maintaining a dump core; and
- The absence of a general reference model that embodies core network design principles and acknowledges the multidimensionality in design entailed in architecting computer networks that reach beyond core networking requirements

1.1 Research Statement

This dissertation presents our proposed clean-slate Concern-Oriented Reference Model (CORM) for architecting future computer networks. CORM stands as a guiding framework from which several network architectures can be derived according to specific functional, contextual, and operational requirements or constraints. CORM represents a pioneering attempt within the network realm, and to our knowledge, CORM is the first reference model that is bio-inspired where its derivation process conforms to the Function-Behavior-Structure (FBS) engineering framework [9].

CORM derivation process was initiated by identifying two core network-design principles that, we assert, are applicable to all computer networks regardless of their size purpose or capabilities. The first principle states that *a computer network is a complex system*, while the second states that *a computer network is a distributed software system*. From a complex system perspective, computer networks need be composed of autonomous entities capable of emergent behavior that can act coherently to perform the global system function in spite of intricate interactions occurring at the micro and macro level. The first principle is motivated by the fact that the Internet in general, and the World Wide Web in specific, have been commonly characterized as complex systems [10], and complex systems are informally described as a network of autonomous components interacting in a nonlinear fashion [10]. However, to our knowledge, complex system characteristics have not been synthesized as primary design features for computer networks. The second principle can be easily justified by noting that all network protocols are composed of software code running on several computing machines. As a distributed software system, computer network need to be designed according to Software Engineering (SE) principles and concepts. Accordingly, we have extensively incorporated the principle of Separation of Concerns (SoC), which primarily addresses system conceptual decomposition and guarantees the development of loosely-coupled highly cohesive systems, into CORM [11].

Guided by our proposed principles, we formulated a Concern-Oriented Bottom-Up design methodology for deriving CORM. The Bottom-Up approach is motivated by our first design principle in general, and network composability of autonomous entities in specific, thus accentuating the importance of the entities composing the network system. These network-building entities need to imitate complex *adaptive* system (CAS) entities (CAS is a complex system whose emergent behavior always lead to overall system stability, in contrast to unstable complex systems whose emergent behavior may result in system meltdown. In this document the term complex system indicates CAS unless otherwise stated), by possessing adaptability, self-organization and evolvability as intrinsic features [12]. The network will then be recursively synthesized from these network-building entities in a bottom-up approach substantiating the two main network-dimensions; a vertical dimension that addresses structure and configuration of network building entities, and a horizontal dimension that addresses communication and interactions among the previously formulated building entities. For these synthesized networks, the Concern-Oriented paradigm represents our vision in network functional as

well as structural decomposition realized at the micro (network-building entities) as well as at the macro (network horizontal and vertical dimensions) level.

Adopting a bottom-Up approach, stipulated, as our first task, the delineation of the network-building entity, in terms of structure, function, and behavior. The structure of the network-building entity was inspired by observations recorded in a recent study on primordial bacterial cells capable of evolution [13], thus introducing the first basic CORM abstraction; the Network Cell (NC). As for the functions to be performed by the NC, these were derived from our interpretation of the network requirement specification. We postulated that the basic requirement for computer networks can be expressed by the following statement: "*The network is a communication vehicle allowing its users to communicate using the available communication media*". Accordingly, we identified the network users, the communication media, and the communication logic as the primary impetuses for computer networks, out of which we derived the second basic CORM abstraction; the ACRF framework that depicts the network functional operation in terms of four concerns: Application Concern (ACn), Communication Concern (CCn), Resource Concern (RCn), and Federation Concern (FCn). These concerns are to be realized along the vertical and the horizontal network dimensions, at the micro and macro levels. The behavior of the NC will be, on one hand, realized as the outcome of its structure, and on the other hand, affected by its assigned function(s), and can be defined in terms of performance aspects and operational parameters.

As presented, CORM refutes the long endorsed concept of layering, intrinsically accounts for emergent behavior, and ensures system integrity and stability. System integrity means having, or being conceived of having, a unified overall design, form or structure [14]. In CORM-based networks, system integrity stems from network component congruency. On the other hand, stability refers to the ability of the system to maintain stable global patterns in spite of the unpredictable interactions occurring at the local level where elements composing the system operate at conditions that are far from equilibrium [15]. In CORM-based network stability stems from the emergent behavior property of CAS that allows system components to adapt to their environment and further evolve to better fit their context, in terms of resources available, technologies used, running applications, etc. [15], [16].

CORM validation was a challenging task that we faced in this dissertation. We posit that models can be classified into definitional and descriptive. A definitional model is more typical of conventional engineering −it expresses required characteristics of a system at an appropriate level of abstraction [92]. A descriptive model, on the other hand, captures observed high-level behavior of a system [92]. Being a reference model for computer networks, CORM can be considered a definitional model. CORM expresses the required CAS characteristics and network functional decomposition through its basic abstraction units (NC and ACRF), and enforces them to be synthesized into the network fabric by construction. Therefore, we resorted to validate CORM and the derivation of its basic abstractions using an engineering model. For this purpose we used the Function-Behavior-Structure (FBS) engineering framework presented in [9], which is applicable to

any engineering discipline, for reasoning about and explaining the nature and process of design [78]. On the other hand, the behavior to be realized in CORM-based networks was substantiated and evaluated by deriving CellNet, a CORM-based architecture. CellNet-compliant protocols' behavioral adaptation and modification were illustrated through simulation.

1.2 Contributions

In this dissertation we adopted a systematic approach to derive a clean-slate network reference model that strives to address the requirements and meet the challenges posed by the network community. Our contribution are summarized in the following subsections.

1.2.1 Intellectual Merits

1.2.1.1 Formulation of Network Fundamental-Design Principles

In several contemporary proposals, design principles are stated in terms of requirements or expected network behavior, which results in idiosyncratic designs that satisfy a particular service model, address an individual problem, or are tailored to a specific set of requirements. In crafting CORM design principles we attempted to express the most fundamental characteristics of computer networks regardless of their size, purpose, or operational context. We conceived the network as a software-dependent complex system. From a complex system perspective, computer network design need to handle complexity and account for emergent behavior. From a software-dependent perspective, computer networks need to be designed as distributed software systems. Thus SE concepts and principles need to be applied to computer network protocols.
Our design principles led to two consequences

1. Adopting a bottom-up approach to network composition: Our bottom-up approach focused primarily on designing the network-building block from which the network will be recursively synthesized. Constructing the network recursively using a bottom-up approach guaranteed network component congruency.
2. Adopting a concern-oriented paradigm to network functional decomposition: The concern-oriented functional decomposition was applied to our derived network specification requirement statement and to the capabilities of our adopted model of CAS (bacteria cells).

1.2.1.2 Building an Evolvable Bio-Inspired Clean-Slate Reference Model for Future Computer Networks

In this dissertation we derive a concern-oriented reference model (CORM) for future computer networks. CORM encompasses our design principles, addresses network multi-dimensionality and stands as a guiding framework from which several network architectures can be derived according to specific functional, contextual, and operational

requirements or constraints. CORM defines the network along two main dimensions: a vertical dimension addressing structure and configuration of network-building blocks, and a horizontal dimension addressing communication and interactions among the previously formulated building blocks. For each network dimension, CORM factors the design space into function, structure and behavior, applying to each the principle of separation of concerns (SoC) for further systematic decomposition. CORM is composed of three main components; the network-concerns conceptual framework, the network structural template, and the information flow model. CORM network-building blocks represents CORM first basic abstraction and are referred to as Network Cells (NCs). An NC behavior is bio-inspired, mimicking a bacterium cell in a bacteria colony. NCs are thus capable of adaptation, self-organization and evolution. NCs functional operation is defined by CORM second basic abstraction; the ACRF framework. ACRF framework is a conceptual framework for network-concerns derived according to our interpretation of computer network requirement specifications. CORM networks are recursively synthesized in a bottom-up fashion out of CORM NCs substantiating the vertical and horizontal dimensions of the network, demonstrating intrinsic adaptable and evolvable behavior fostered by their basic building blocks; the NCs.

1.2.1.3 Deriving and Evaluating a CORM-Based Network Architecture

We conjectured that network realizations derived from CORM-based architectures can adapt to context changes and further evolve by inducing online modifications to the network logic executed by network elements, allowing these elements to operate according to learned optimal values. To justify our conjecture, we derived CellNet; a CORM-based architecture. CellNet was derived using CORM abstractions (NC, Ncomp, and ACRF) to define architectural entities, and their functionalities. In deriving CellNet we aimed to devise a minimal architecture interoperable with the Internet protocol suite. CellNet was simulated to illustrate behavioral modification prospects of CORM-based architectures.

1.2.2 Broader Impact

CORM, our presented concern-oriented reference model, impacts the network realm at an architectural level as well as at the network realization level.

At the architectural level, CORM refutes the long-endorsed network architectural concept of layering by introducing a new abstraction unit, the NC, that stands in contrast to a layer. NC is a self-contained entity encompassing network-concerns. An NC can exist and operate autonomously while being aware of its state, context and peers. This stands in direct contrast to a layer that is totally oblivious of its the state, context and other layers except for those directly adjacent to itself. Moreover, a layer cannot exist or operate autonomously. Adopting an NC as the basic abstraction unit, rather than a layer, CORM builds the network recursively in a bottom-up approach defining the network component (Ncomp), network and internetwork all to be made of NCs. Thus establishing

network system integrity due to network building-element congruency. CORM bottom-up approach in network design and construction contradicts the E2E principle that has been central to the Internet design and repudiates the prevailing top-down design approach that abstracts a network in terms of an internetwork. CORM accentuates network awareness, adaptability and evolution, by incorporating CAS characteristics into the its basic abstraction unit as primary features introducing monitoring and regulation as first class architectural constructs intrinsic to the NC. CORM addresses the multi-dimensionality required in network design by expressing the network in terms of function structure and behavior thus adopting an engineering perspective to network design and architecture.

At the network realization level, CORM will have a profound impact on the operation and behavior of computer networks composing the Internet. By introducing awareness adaptability and evolvability as network intrinsic features, CORM-based Internet will proactively respond to changes in operational contexts, underlying technologies, and end user requirements. Internet self-awareness and adaptation will alleviate the burdens of network-management and resource-provisioning, deliver better security, resilience and survivability in response to attacks and faults, and maintain overall stability due to continuous adaptation to changing conditions. Finally, evolution will minimize human intervention delivering an Internet capable of integrating learned knowledge from past experiences into operational protocol logic.

1.3 Document Organization

This dissertation is organized as follows: Chapter 2 presents the necessary context for our discussion of network design principles by providing an overview of the state of art in the field of computer networks starting with the initial efforts that guided the development of the Internet, pointing out the different problems faced nowadays and attempted solutions as well as surveying some initiatives for Future Internet architecture. Chapter 3 details our proposed network design principles that have guided this research, argues for our adopted Bottom-up Concern-Oriented design methodology, and introduces CORM, our proposed Concern-Oriented Reference Model for architecting computer networks. Chapter 4 overviews the Function-Behavior-Structure engineering framework and applies it to CORM and CORM derivation process. Chapter 5 presents and evaluates CellNet, a CORM-based architecture. Chapter 6 concludes the dissertation and highlights future work.

Chapter 2

Background and Related Work

"In the spider-web of facts, many a truth is strangled."
Paul Eldridge

The Internet as we know today is an outcome of a sequence of amendments, accretions and innovations that have sprung up and steered the use of the Internet into directions which were not initially anticipated. However, this sea of changes has pushed the Internet to its limits marking it inadequate for sustaining the future. A state that motivated a plethora of research endeavors; some attempting point solutions, while others aimed for architectural alternatives to the current Internet model. Believing that a thorough understanding of the different concepts and factors that shaped the current Internet provide valuable lessons to be incorporated in future Internet designs, this chapter overviews the state of art in the Internet research arena starting by the preliminary efforts exerted while the packet-switched network research was at its infancy, analyzing the different design decisions and circumstances that shaped the Internet model delivering the present realizations of the Internet today, and concluding by a synopsis of the different initiatives proposing novel design concepts, principles, and architectures targeting the Future Internet.

2.1 Packet-Switched Networks: A Glimpse in History

The origin of packet switched networks dates back to the early 1960's when the first packet switched network was developed in the US to provide a reliable communication media that can sustain any level of link destructions. Back then, there were no clear design principles that guided network development, but rather it was an incremental process subject to trial and error. The idea of reliable data networks motivated the development of several packet-switched networks in other countries, and the Internet was motivated by the idea of interconnecting these desperate networks together. Following is an overview of the different projects that contributed to the birth of the Internet.

2.1.1 RAND's Distributed Communications (1960-1964) [17, 18]

In early 1960's Paul Baran introduced the basic concepts and requirements for designing a survivable distributed digital data communications network that can stand enemy attacks. He presented a distributed network – a network which will allow several hundred major communication stations to talk with one another – that can withstand almost any degree of destruction to individual components without the loss of end-to-end

communications [18]. Survivability is guaranteed through redundant links since each node would be connected to several of its neighboring nodes in a sort of lattice-like configuration giving it the choice of using several possible routes to send and receive data. Communication is carried out through a network of unnamed nodes that would act as switches routing "message blocks" from one node to another towards the final destination. The unnamed switching nodes would use a scheme called "Hot potato routing," which is a simple rapid store and forward switching mechanism to handle all forms of digital data using a standard format message block. Message blocks are formed by dividing the original message at the sender, and flow independently through the network to be rejoined again to form the original message at the destination. Baran coined his proposal as "distributed adaptive message block switching," which is the basis for packet switching as we know it today. Yet the term "packet" was introduced by Donald Davis, who independently devised a very similar system at the National Physical Lab (NPL) in the UK as presented next [19].

2.1.2 NPL (1965-1973) [19]

In 1965 Donald Davis at the National Physical Lab (NPL) in Britain proposed the possibility of building a nation wide digital data communication network. Inspired by the time-sharing systems, Davis envisioned a data communication network that can share a number of digital links thus solving the problem of unreliable links, as well as cutting down on the cost of data communication. His proposal divides messages to be exchanged into a number of individual packets, and then uses routing protocols to get the packets from source to destination through independent routes. The network in Davis architecture is logically divided into two sections; Node computer and Interface computers. Yet, in actual implementation, both the Node computer and the Interface computer can be running on the same device. The Node computers represent the core of the network joined by several links. The Node computers are responsible for routing packets within the network using packet switching technique similar to what Baran previously proposed. The Interface computers sit at the boundary of the network handling a mixed collection of subscribers in a geographical area. The Interface computers are responsible for putting messages into a well defined format, including routing data, before passing it into the network. Davis architecture used time-division multiplexing to allow multiple users to take turns transmitting portion of their messages. The first experimental network devised according to Davis model was called Mark I. Mark I was fully operational by 1971 providing NPL researchers remote access to computers for writing and running programs, querying databases, sharing files, and special services such as "desk calculator". Mark I was upgraded to Mark II in 1973 that provided faster services and remained in service at NPL until 1986. NPL networks were instrumental in passing on the knowledge of packet switching to the eventual ARPANET.

2.1.3 Cyclades (1972-1980's) [20, 21]

Cyclades is the French version of the digital data communication networks. The project was initiated sometime in 1970 after a French delegation visited the United States, and discovered the work on the ARPANET. As a result several reports were generated that aroused the interest in France for a French instantiation of a heterogeneous computer network to serve, as well as experiment with, at universities and research centers; the outcome was the Cyclades network. Although Cyclades had its roots in the ARPANET, it had a major contribution in shaping the ARPANET subsequent development into the global Internet as we know it today. That's why we will mention it before divulging into the interesting details of the ARPANET evolvement.

Being developed at an era of relatively mature communication principles and networking concepts, Cyclades was the first network that exhibited an architectural design dividing the networking functions into three independent layers; Application, Transport and Data Transmission.

Data Transmission layer is the bottom-most layer in Cyclades referred to as Cigale. Cigale is a packet-switching network providing basic functions, such as routing and forwarding to entities located both outside and inside Cigale. When designing Cigale, designers aimed to keep it simple to avoid duplication of functions between various layers, as well as to allow the possibility of interconnection with other networks. In the context of other networks, Cigale itself is a router, allowing the recursive definition of networks.

Transport Layer is the middle layer of the overall Cyclades architecture residing directly above Cigale. It provides inter-process communications facility among transport entities called Transfer Stations (ST). STs are pieces of software running on hosts providing transport service to the Application layer. STs implement a transport protocol defining message exchange procedures, as well as message formats. At this layer, functions such as connection setup, fragmentation and reassembly of packets, error control and error detection, flow and congestion control are implemented.

Application Layer is the higher most layer of the Cyclades architecture. It instantiates the End to End protocols for the Cyclades Network. Two main application protocols devised for Cyclades were the virtual Terminal Protocol and the File Transfer protocol.

Major contributions of the Cyclades networks to the development of the ARPANE,T and eventually the Internet include the following:

- Early conceptualization of layering;
- Defining the network in terms of complex intelligent edges connected by a totally unreliable simple data transmission layer;
- First true implementation of a network designed to take advantage of the characteristics of datagrams, utilizing an out of order delivery service and adaptive routing;
- Providing useful insights with respect to flow and congestion control through the implementation of Channel Load Limiter protocol to control network traffic using

choke packets, as well as developing the modern-day "Sliding Window" Protocol that was later adapted to TCP;

- Introducing hierarchical Addressing schemes.

Unfortunately, due to continuous reduction of funding, the French development of packet switching technology and products withered, and the Cyclades network was abandoned in the early 1980's.

2.1.4 ARPANET (1966-1989) [22-28]

The ARPANET was developed after ARPA. ARPA (Advanced Research Project Agency) was first realized as a military project initiated by President Eisenhower as a reaction to the Soviet launch of Sputnik. Its first purpose was to counter the Soviet threat [22]. In late 60's ARPA awarded the contract for ARPANET to BBN, to construct a physical network. The ARPANET's purpose was to provide fast reliable communication between heterogeneous host machines [18]. APRANET was a packet-switched store-and-forward network. It borrowed the packet switching paradigm from the RAND's Distributed communication project led by Paul Baran. ARPANET started with only four nodes (1968-1969), each was a small processor called Interface Message Processors (IMP). The IMPs were connected to a leased common carrier circuit to form a subnet. Computers were hooked to the ARPANET through the IMPs subnet that provided an invisible means of transmitting messages from source to destination [23]. An IMP would send and receive data, check for errors, retransmit in the case of errors, route data, and verify message delivery. During the early 70's, there was continuous growth in computer networks. ARPANET expanded, and different improvements and additions were made to its protocols. The most important of which was the Network Control Protocol NCP, the first basic email system, and the first File Transfer Protocol. In 1972 ARPANET was publicly demonstrated at the International Conference on Computer Communication in Washington leading to increasing interest in computer networks [24]. In 1973, the first attempt at internetworking the ARPANET with the Packet Radio Network took place. It was then realized that a general internetworking protocol is required for linking different national networks. This was the motivation for designing the TCP protocol. In 1974 Bob Kahn and Vint Cerf wrote a paper titled "A Protocol for a Packet Network Intercommunication" designing a transmission-control protocol that is not tailored to a specific network, in contrast to the NCP that was designed primarily for the ARPANET [25]. By 1977, the TCP operation began over the ARPANET linking it to Packet Radio Net, and Satellite Network (SATNET) through gateways[24]. In 1978, TCP was split into two protocols; the Internet Protocol (IP) that will deal with routing, and TCP protocol that will take care of the packetization, error control, retransmission and reassembly [25]. Due to its success in inter-netting networks, the TCP/IP became the preferred military protocol in 1980 resulting in the official transition of ARPANET from NPC to TCP/IP protocols on the 1st of January 1983. ARPANET kept operational until it was finally shutdown in 1989.

The ARPANET protocols had a layered relationship as seen in Figure1. The core protocol is the IMP-IMP (not shown in the figure), next the Host-IMP and the Host-Host Protocols.

IMP-IMP protocol: The IMP-IMP protocol implemented a reliable store-and-forward packet switching network. The protocol differentiates between the intermediate IMPs and source/destination IMPs in terms of responsibilities. The former comprised the forwarding engine of the network performing error control, route determination and packet forwarding along the path from source IMP to destination IMP. While the latter was responsible for end-to-end connection management, message fragmentation and reassembly [26].

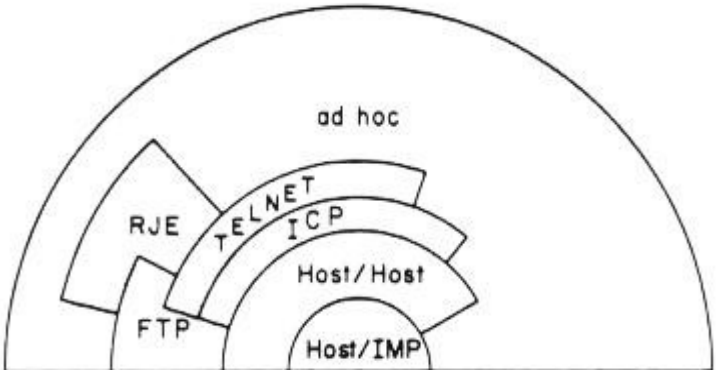

Figure 2.1:Layered Relationship of ARPANET Protocols [25]

Host-IMP protocol: The Host-IMP protocol was responsible to regulate the communication of messages between hosts and IMPs on a network. A host would pass its messages that need to be delivered to another host on the network to its IMP. The host IMP will send the message to the destination IMP. The destination IMP would then send acknowledgements back to the host on the status of the messages. If a message was successfully received by the destination host, the destination IMP would send a RFNM (ready for next message), and if a message was lost in the network the destination IMP would send an incomplete transmission message. The destination IMP was responsible for the reassembly of the received packets into the original message, and sending the fully assembled message to its host. An IMP was also able to block incoming messages from its host for various reasons [27].

Host-Host protocol: The host-to-host protocol was implemented to provide inter-process communication over the network, since end users will be running multiple independent processes that need to use the network concurrently. The main host-to-host protocol was the NPC protocol that was later replaced by the TCP protocol [28].

2.1.5 TCP/IP and the Internet

The adoption of TCP/IP as the “transport protocol” for *inter-networking* different networks could be considered the initiation of the Internet as we now it today. In 1981,

TCP/IP was incorporated into distributions of Berkeley Unix, initiating broad diffusion in the academic and scientific research communities [22]. A year later, SUN Microsystems distributed SUN workstations running UNIX with TCP/IP for free, pushing more momentum in TCP/IP realization [22]. In early 1980's, networks developed around the US became TCP/IP aware. The most prominent was the CSNET. CSNET was a network developed mainly to connect computer science departments to provide services to computer science research group in the US [29]. CSNET played a central role in popularizing the Internet outside the ARPANET, eventually connecting more than 180 institutions and tens of thousands of new users, who in turn went on to promote the awareness and spread of the growing network [30]. By 1986, other countries built gateways between their national networks and the US Internet, and the Internet came to mean the international network dissolving national boundaries [24].

2.1.6 Open Systems Interconnection (OSI) Reference Model

In 1977, the International Organization for Standardization (IOS) realized the need for standardizing the rules of interaction between the evolving heterogeneous networks. The OSI reference model was an attempt to address network interconnection need. The OSI reference model focuses on the external behavior of networks visible to other interconnecting network with no concern to the internal structure or functioning of individual networks. A protocol suite was supposed to be developed according to the layered architecture proposed by the OSI model. Yet, the whole project was eventually abandoned due to the proliferation and success attributed to the TCP/IP protocol suite in substantiating a global interconnected network [31].

2.2 Network Architecture and the Layered Model

As reported in the previous historical overview, several disparate experimental computer networks were developed, each with its own architecture, before the realization of the Internet as a global network of networks. The first interconnection among *architecturally* different computer networks was conceived by the introduction of the Transmission Control Protocol, which sole purpose was to allow the external interaction of the connected networks without being aware of their internal structure. As such, the Transmission Control Protocol was required to be as flexible as possible, allowing for the integration of a number of separately administered networks into a common utility. To realize such required flexibility, the Transmission Control Protocol was further split into IP and TCP introducing the layered IP/TCP protocol suite that defined the Internet architecture. Although layering was not realized as one of the main design principles of the Internet, yet it became one of its defining characteristics as quoted from [32].

> The design philosophy [of the Internet] has evolved considerably from the first proposal to the current standards. For example, the idea of the datagram, or connectionless service, does not receive particular emphasis in the first paper, but has come to be the defining characteristic of the protocol. Another example is

> the layering of the architecture into the IP and TCP layers. This seems basic to the design, but was also not a part of the original proposal. These changes in the Internet design arose through the repeated pattern of implementation and testing that occurred before the standards were set.

According to the previous quote, the Internet layered architecture was incrementally developed to define an "Interconnection Architecture", i.e. an architecture that facilitates the interaction of heterogeneous computer networks, *not* architecture for building computer networks. Yet, the astounding success of the TCP/IP suite in interconnecting networks undermined the need for a separate research addressing the requirements and specifications for internally architecting networks. As such, TCP/IP was taken as the de facto model for internally building networks. Moreover, the OSI model when released, stressed the idea of the layered architecture strengthening further the TCP/IP layered approach in network design and protocol engineering.

As an "Interconnection Architecture" the layered model was developed and standardized according to a set of requirements that expressed the high level goals of the Internet operation and function, which were eventually documented as the Internet design principles to be presented in the next section.

2.3 The Internet Design Principles

The fundamental goal of the Internet was to develop an effective technique for multiplexing utilization of existing interconnected networks [32], while taking into consideration the following features; survivability in the face of failure, allowing variability in the type of services supported and network technologies used, and distributed management and control. To satisfy these high level goals more specific principles where crafted. The main of these principles that strongly guided the design of Internet protocols and shaped its advancements are the end to end principle, the end-to-end connectivity and best effort datagram model.

2.3.1 The End-to-End Principle

The end-to-end principle is considered to be a central principle to the Internet design. It states that

> mechanisms should not be placed in the network if it can be placed at the end node, and the core of the network should provide a general service, not one that is tailored to a specific application.[33]

Adopting the end-to-end principle had several advantages [34];

- Minimizing redundancy, as functions performed at the edge are not repeated in the core leading to reduced complexity in the network core.
- Increasing reliability as applications running at the edge do not depend on the operation of application-specific functions in the core.

- Supporting application innovation and diversity since a general core endorses new applications without any required changes.

A consequence of the end-to-end principle is a network core completely oblivious to traffic passing through it. Hence, the network was described as a "dumb forwarding machine"; whatever goes in the network comes out, a property known as network transparency [35]. Network transparency served well for edge evolvability allowing diverse applications to simultaneously exist at network edges. Examples include Mail, World Wide Web, and Peer-to-Peer applications. Moreover, end-to-end principle satisfied the other design goals, as it implies that end running application do not depend on the network to preserve state information pertaining to on-going communication between any two endpoints. Any loss of state information associated with a communicating entity implies that the entity itself is lost. A reliability approach referred to as "fate-sharing" [32]. However, over the last few years, a set of new requirements for the Internet applications, users and even providers have emerged, and it was argued that these newly emerging requirements will be best served by adding new mechanisms in the network core. Hence, leveraging network core awareness to traffic characteristics and needs. Examples of new requirements include securing end-points against adverse communication, guaranteed delivery model for supporting real-time applications, and incorporating management tools for easing the connection of novice users, as well as aiding network operators in diagnosing network problems and misconfigurations. In spite of the stressing need for a more traffic aware core, it is argued that the end-to-end principle is 'still valid and powerful, but need(s) a more complex articulation in today's world' [33]. In fact, the end-to-end principle should not be considered 'an absolute rule, but rather a guideline that helps in application and protocol design analysis; one must use some care to identify the end points to which the argument should be applied' [36]. Hence we can deduce that we need not abandon the end-to-end principle, but rather find ways to control the transparency model that it delivers [37], or

> What is needed is a set of principles that interoperate with each other—some built on the end to end model and some on a new model of network-centered function. In evolving that set of principles, it is important to remember that, from the beginning, the end to end arguments revolved around requirements that could be implemented correctly at the end-points; if implementation inside the network is the only way to accomplish the requirement, then an end to end argument isn't appropriate in the first place. The end to end arguments are no more "validated" by the belief in end-user empowerment than they are "invalidated" by a call for a more complex mix of high-level functional objectives [34].

2.3.2 The End-to-End Connectivity

End-to-end connectivity implied that any two hosts connected to any of the networks that compose the Internet can communicate. This entailed identifying a unit for message exchange, integrating the disparate disjoint and separately administered networks to form one homogenous communication substrate, and the need to uniquely

identify each and every host capable of communication. Datagrams were chosen to be the unit of message exchange, and will be discussed in the next section. Disparate networks were integrated by deploying network gateways that interconnected using a generic network protocol; the Internet Protocol (IP). Network hosts were assigned unique logical identifiers, which were addresses extracted from a global address space. The IP protocol used these logical identifiers to locate communicating hosts. To comply with the end to end principle, the IP protocol operation depended on minimal state information in the form of routing and forwarding tables stored in intermediate nodes. These tables were generated and used by routing protocols that discover, configure and maintain routes to define paths connecting communicating hosts. Moreover, both the routing and IP protocols were oblivious to the data being carried in datagrams, thus preserving the network core transparency.

Although the end-to-end connectivity is a central principle to any network that aims to achieve universal connectivity, yet, wrong interpretations, assumptions and decisions led to a constrained implementation of end-to-end connectivity on the Internet. The first problem came about with IP address depletion due to the unwise allocation of the address space. As a solution, the address space was split into globally routable addresses to be used on the public Internet, and local private addresses to be used within intranets, with Network Address Translation (NAT) devices sitting at the borders of private intranets to translate non-routable local IP addresses to globally routable IP addresses and vice versa. Thus allowing IP addresses within the global address space to be shared and reused, disrupting the end-to-end connectivity model. The second problem was introduced with the need to support communicating end-points (host/application) mobility and multi-home feature. A situation that led to the location identifier split concept. The assumption that an IP address represents both a communication endpoint identity and location is no more valid, since with mobility an end-point changes its location thus loses its identity. Location/identifier split entails using name-independent routing schemes [38] where a communication end-point will be identified using a flat, topologically agnostic label, and located using a topologically informative label. Research efforts in the field of compact routing shows that achieving efficient scalable name-independent schemes is an issue and we quote their conclusion here

> We thus conclude that in order to find approaches that would lead us to required routing scalability, we need some radically new ideas that would allow us to construct convergence-free, “updateless” routing requiring no full view of network topologies.-[38]

The previous conclusion could be an indication that end-to-end connectivity need not depend on global namespaces and homogenous gluing protocol, but rather other structures need to be defined. Parallel to that line of reasoning several research proposals presented the concept of regions or realms [39] [40], where a realm or region is defined using a set of characteristics including protocols used, addressing schemes adopted, policies implemented etc… Realms need not be aware of internal structure of each other, yet, they can interact, co-exist and overlap. Inter-realm communication is performed

using translation mechanisms defined by the communicating realms that best fit their internal structures. Such region and realm concept endorse the heterogeneity envisioned in present and future networks.

2.3.3 Best Effort Datagram Model

Datagram was chosen to be the unit of message exchange over the Internet for several reasons [32]; first datagrams were crucial for achieving reliability (fate-sharing) since datagram networks need not store essential state information about on going connections. This means that the Internet can be reconstituted after a failure without concern about state. Second, datagram provides a basic building block over which several types of services can be implemented by providing "best effort" delivery. Thus it was possible to build out of datagrams a service that was reliable or a service which traded reliability for the primitive delay characteristics of the underlying network substrate. Third, the datagram model did not place any requirements on underlying networks, allowing a wide variety of networks to be incorporated into the Internet. The variable-sized datagram, as a basic building block for communication, served its purpose well for network edge services by providing a general solution to the problems of statistical multiplexing, fine-grained congestion control and quality of service, and support for a wide range of applications. Yet, when considering network core services such as resource management, accounting, and traffic engineering, a unit of representation other than datagrams may be required. This unit of representation will be able to identify sequence of datagrams traveling from source to destination. This sequence of datagrams is referred to as "flow" in [32]. To be able to manage datagram flows, gateways need to store flow states in order to remember the nature of the flows passing through them. Yet, these flow states need to be stored as "soft state" to retain the core flexibility and survivability. Using datagram flows as the unit of representation in the network centre, allow multiplexing on packet aggregates, producing overall traffic patterns characterized by short term fluctuations when compared to the bursty and intermittent traffic patterns visible near the edge [37]. Thus reasoning on aggregates, rather than per-packet basis, yields better control over network core behavior, and better use of resources. At present, the Internet architecture does not recognize packet aggregates of flows, but rather lower level mechanisms such as MPLS are used to manipulate them. Therefore it has been urged in [37] that any future architecture should include some means to name and reason about aggregates as fundamental, first class objects, so that important practical objectives such as traffic engineering, resource management, on-demand bandwidth acquisition, and QoS control can be expressed within the architecture.

2.3.4 Conclusion

Based on the layered architecture and the aforementioned design principles, the Internet delivered most of its intended goals connecting disparate computer networks and fostering a wide range of applications and services. Yet, it has been argued that these

design principles, as much as they had contributed to the Internet success by allowing application diversity and innovation; they had equally constrained the evolution of the Internet core, rendering it biased to those applications that can tolerate its oblivious nature. Striving to address the Internet limitations, designers sought to augment the Internet protocol suite with point solutions and patchwork of technical embellishments that introduced new challenges to the problem space motivating a plethora of alternative network architectural proposals targeting computer networks in general and the Internet in particular. The following two sections discuss the present Internet limitations and the different proposals and initiatives aiming to introduce architectural innovations as presented in literature.

2.4 Present Internet Limitations

The limitations of the Internet architecture can be attributed to several factors; some pertain to the underlying layered architecture and some are due to implementation decisions. In this section, we aim to highlight these factors. An extensive discussion can be found in the references

2.4.1 Layering and Architectural Limitations

2.4.1.1 Strict Ordering

The concept of layering is a well-established paradigm for functional decomposition, [42]. Yet, practical experience shows that layering might not be the most effective modularity for protocol suite implementation [42]. Developing protocol suites according to a layered architecture has the benefit of functional separation allowing protocols residing within each layer to operate on a subset of the networking function. However, layered protocol suites impose a strict sequential order on protocol execution, which may conflict with the efficient engineering of end systems, thus constraining optimization opportunities that could have been realized in absence of such ordering [42].

2.4.1.2 Vertical Integration

The layered architecture abstracts the basic hierarchy of communication into horizontal layers, which fails to express important system aspects such as performance, security and management [35]. These functions can not be confined to a single layer nor can they be abstracted as a horizontal layer. On the contrary, these functions need to span all layers of the communication hierarchy. Such vertical integration of layers needs to be expressed by the architecture so as to be well designed and engineered in network protocols.

2.4.1.3 Layer Granularity

The layered architecture only defines interlayer boundaries leaving the internals of each layer undefined. This leads to two main consequences; first functions implemented within each layer have undefined dependencies and are addressed as monolithic blocks [35]. For example, the routing function can be divided into several tasks, such as route discovery, route selection, route management, and packet forwarding. However, these functions are collectively implemented into a single routing protocol excluding the notion of choosing among different implementations of the same task as selection is often done at design phase. In addition, such engineering approach hinders development and extensibility, since a change in one of these tasks necessitates the replacement of the whole protocol [35]. Second, undefined internal structure of a layer leads to primitive interfaces for inter-layer communication. Even for adjacent layers, protocols are engineered to operate oblivious of each other's state regardless of the dependencies that exist among them. For example, TCP protocol at the transport layer, translates packet losses as network congestion although it could be due to host mobility at the lower layers.

2.4.1.4 Network Component Distribution

The layered architecture failed to capture the distributed nature of network components which happens to be very important when engineering protocols. Network component distribution can be expressed along three dimensions; a physical dimension, a logical dimension, and a control dimension. The physical dimension expresses the geographical proximity of the network components (How different parts of the network, devices or modules, are close to each other?). The logical dimension expresses the topological layout of the network (How system components (devices or modules) relate to and interact with each other and with what constraints?). The control dimension addresses the fact that different parts of the network are owned/operated by different administrations. For example, two computers may be physically close, yet, topologically separated by many hops, or two computers are physically separated by several miles, yet, are under the same administrative control. To appreciate the importance of considering network distribution in protocol engineering, we note that failing to capture these dimensions in the network architecture caused a major flaw in the design of scalable routing protocols presently used on the Internet [38]. Routing protocols are designed based on the idea of hierarchal clustering of nodes, meaning that nearby nodes are grouped into clusters, and clusters into super-clusters, and so on in a bottom-up fashion. Thus hierarchal clustering abstracts out unnecessary topological details about remote portions of the network and allows building hierarchal routing schemes that use address aggregations for nodes, attaining scalable routing tables. Yet, it has been shown that the Internet topology is scale-free, i.e. it does not exhibit the hierarchal structure assumed by hierarchal routing, rendering hierarchal routing schemes inefficient when applied to Internet-like topologies.

From the above discussion we see that network architecture needs to judiciously express the physical, logical and administrative layout of network components (devices or modules) to guide designers and system engineers to develop scalable as well as efficient protocols suited to the different network-layouts

2.4.1.5 Separation of Data, Control and Management Planes

Using a layered model to guide protocol design fails to capture the fact that information exchanged within the network need to occur along different planes. Three planes can be identified; data (User) plane, control plane, and the management plane. The data plane represents the information flowing in the network without being interpreted and has no significance except for the end applications running on top of the sender and receiver protocol stack. The control plane is parallel to the data plane, where control information is exchanged between peer protocols on both sides of the communication path to handle the data flow. The management plane is a distributed plane that intersects with both the data and control planes. It has a holistic view of all flows administered into the network and how these flows should be served by the different components constituting the network. The management plane mainly decides among the different options that the network can provide its users at a specific moment of time giving the network load and resources. The management plane depends greatly on monitoring functions and network state acquisition. In that sense, we can say that the management plane is composed of a monitoring plane, a knowledge plane, and a reasoning element. The monitoring plane is responsible of acquiring information about network state and context, and feeding this information to the reasoning element. The knowledge plane is where the network past experiences and history can be stored for any future reference. The reasoning element is in control of both the monitoring and knowledge planes reading from the former the instant states and conditions and storing in the latter the inferred decisions and learned experiences. Failing to explicitly represent the different planes required for network operation and management, protocols designed according to the layered model tightly couple the control and management information to the data path limiting both their scope and effect. For example the SNMP cannot interface with the application layer protocols directly to inform a browser that a server is down or the IP is invalid, and the control information for each layer is stripped off before reaching a higher layer in the stack. That's why when implementing congestion control using Explicit Congestion Notification (ECN) the network layer protocol had to access the header of the transport layer to make the effect.

2.4.1.6 Information Flow and Information Sharing

A related concept to the different planes identified in the previous point is the concept of information flow and information sharing. Layer boundaries, in the layered architecture were chosen to minimize the information flow across interfaces, and intra/inter-layer communication was limited to a minimum set of primitives through packet header exchange. Packet headers are metadata exchanged between protocols

residing in the same layer or in adjacent layers. Packet headers are used by peer protocols indicating how the accompanying payload needs to be handled. They provide no insight for performance levels expected or required by end applications, nor do they reveal operation problems encountered, or opportunities offered by lower protocol implementation. Packet headers are stripped off before a packet is passed to a higher layer or treated as payload as packets pass down to lower layers. As such the layered architecture was built on the concept of isolation where upper layers information is completely masked form lower layers and lower layer information is never retained as packets move upwards.

2.4.2 Limitations Induced due to Implementation Decisions

2.4.2.1 Implicit Assumptions about the Operating Environment

The Internet was mainly developed to support the transfer of data, which was mainly text, between wired networks. Accordingly two implicit assumptions were incorporated into the design of Internet protocols; first, the only service model addressed was the best effort service since text transfers are delay tolerant. Second, the underlying communication technology uses static wired links. The first assumption was invalidated by the need for Quality of Service transfers to support real-time applications. This led to the introduction of two protocols at the network layer, the DiffServ and InterServ protocols. Both provide a level of guaranteed delivery by classifying the traffic into flows or classes. Yet both protocols were confronted by the layered architecture model, as they need indication from the application as to which class or flow the traffic belongs to, thus violating the strict layer design.

With the advancement of wireless technology and the proliferation of electronic devices equipped with multiple wireless network interfaces into our daily life, the second assumption was no longer valid. Protocols developed to operate in an end-to-end wired paradigm, assuming stable link conditions in terms of bandwidth, error rates, and transmission latencies, are now required to support communication over wireless links as well as endorsing new communication paradigms that depart from the traditional wired model. Wireless communication allows end hosts to roam around while switching connection from one wireless interface to another. Operating over wireless links require protocol implementations to address the following issues:

- Dynamic operation in a wireless environment: Stable link conditions can no longer be assumed for wireless links. Moreover, when compared to wired links, wireless links are characterized by lower bandwidth, higher error rates, and variable transmission delays. Protocols designed to be oblivious to these changing link conditions operate poorly in wireless environments.
- Accommodating end user mobility: Two forms of mobility are defined in [42]; user mobility and network mobility. User mobility supports switching between devices, as well as migrating sessions while getting the same personalized

services. Network mobility enables the users' devices to move around in different networks and maintain connectivity.

- Manage scarce resources: In wireless systems, power consumption and bandwidth allocation need to be carefully managed. On one hand, power management extends the operation life time of wireless devices as well as increases network capacity. On the other hand, bandwidth allocation enhances overall network performance [43].

As for adapting to new communication paradigms, the following communication concepts need to be realized in protocol implementations;

- The "always best-connected" concept implicit in fourth Generation wireless systems (4G), where wireless terminals are required to select the best access method/interface available and connect using it [44].
- Opportunistic communication where two nodes are enabled to exchange messages with each other even if a route connecting them never existed [45]. One form of such communication is Delay Tolerant Networks (DTN) defined as an overlay on top of regional networks, including the Internet allowing communication over disparate networking environments.

Incorporating both wireless link adaptation and new communication paradigms into existing protocols called for a new design concept known as cross layer design. Cross layer designs depart from the strict layered model, and allow protocols at different layers to interact and communicate.

2.4.2.2 Proliferation of middle boxes

One of the main concepts of the Internet was to develop a simple flexible core that is oblivious to the data being transferred while providing basic communication services and keeping control at the edges [35]. With the emergence of ISP and the realization of the Internet as a commercial product, ISPs need to provide higher level, user-visible services that offer the possibility of product differentiation and higher profits. This motivation led them to implement services in the part of the network they own, thus pushing control into the core by introducing what is called "Middle boxes". Middle boxes such as firewalls and NAT devices do not fit into the current layered architecture. So ISPs, in their drive to remain profitable, are motivated to violate the most basic of the Internet's architectural principles [35].

2.5 Architectural Innovations

Several alternatives to the layered architecture have been proposed even before the advancements of new wireless technologies and the introduction of new services that request support for a wide range of media (voice, video, graphics, and text) and access patterns (interactive, bulk transfer, real-time rendering, opportunistic). The argument presented back then was that layering may not be the most effective modularity for implementation, since it unnecessarily constrains or complicates the engineering

alternatives available to the implementer [41]. However, the reluctance to change working implementations and long-standing inter-layer interfaces often led designers to apply short-term solutions by inserting functionalities between existing layers, rather than modifying them. Nevertheless, the proliferation of wireless communication and demanding needs of new Internet services motivated individual researchers to propose alternatives to the layered architecture. Some presented point solutions, while others aimed at providing a general framework for network design. Some preserved the layered structure, while others took a "clean-slate" approach to their design. However, more influential research attempting architectural innovations aimed at Future Internet was conducted by the US NSF FIND [46] and EU FIRE [47] initiatives. This section surveys the different proposals presented in contemporary research literature. Proposals will be discussed from two points of views; the first classifies the presented designs as incremental or clean-slate. The second view classifies the design according to the network dimension addressed. We identify two network dimensions along which most architectural proposals can be classified; a vertical dimension addressing structure and configuration of protocols, and a horizontal dimension addressing communication in terms of routing functions.

2.5.1 Cross-Layer Designs

Cross-Layer Designs (CLD) defy the strict layered architecture and allow protocols at nonadjacent layers to interact and communicate. CLD concept was motivated by the severe performance deterioration of traditional protocols when operating in wireless environments. In order to cope with the unique and dynamic nature of wireless links, a protocol architecture that incorporates CLD has been frequently proposed. The majority of CLD presented in the research literature address a specific requirement of the wireless communication such as QoS, TCP performance, mobility, resource management, etc. However several general designs that targeted the network architecture as a whole were also presented and those are the designs that will be considered in this section.

The first attempt to incorporate a degree of layer interaction was proposed in [48] through the use of Hints and Notifications (HAN) exchanged between layers above and below the network layer. Exchange was done using option fields in the IP header and ICMP messages. Hints are provided by upper layers to the link layer to guide it in its choice of error coding, modulation techniques and retransmission policies. Notifications, on the other hand, are passed from link layer to the upper layer as the reason for packet drops and the level of link congestion experienced. All exchanges are done at the network layer.

In [49] a more comprehensive cross layer interaction scheme was presented and referred to as Cross-LAyer Signaling Shortcuts (CLASS). The basic idea of CLASS was to break the layer ordering constraints, while keeping the layering structure and propagating cross-layer messages through local out-band signaling shortcuts.

Interlayer Coordination Model [44] was the first to present a cross-layer framework for the layered architecture, where a central entity manages the different layer

interactions. Four planes are identified along which interactions are required across the protocol stack. These are Security plane, QoS plane, mobility plane and link adaptation plane. Protocols at different layers, whose operations will be affected or can affect the operation at the previously mentioned planes, expose their internal state to the management entity (ME), which on its turn, responds with the appropriate action.

Reference [50] introduces ÉCLAIR, which is an architecture for an asynchronous cross layer feedback system where protocol adaptation is done out of phase with the protocol operation. ÉCLAIR splits the cross layer system into two subsystems; Tuning Layers (TL) and Optimizing Subsystem (OSS). A TL is an extension appended to each layer of the protocol stack, and is aware of the data structures used to store the control information of each protocol within a given layer. OSS is composed of several Protocol Optimizers (POs). Each PO contains the algorithm for a particular cross layer optimization. TL exports an API to PO, which, on its turn, needs to register with the TL for information about a specific event. On receiving feed back from the TL, the PO executes the required optimizing code to modify protocol behavior.

Another cross layer framework is presented in [51]. It uses a server local to the mobile host to coordinate interaction among nonadjacent layers using coordination clients that are added to each of the protocol stack layers. The cross-layer clients abstract the corresponding layer states as well as exchange event messages with the coordination server residing on the node. Event messages reveal the different events occurring at each layer. The coordination server, on its turn, decides on the action to be taken according to the different activities running in the node, and communicates the decision to the cross-layer clients, which can update the state parameters of executing protocols.

Table 2.1 identifies the design rationale for the mentioned CLDs, while Table 2.2 lists the different parameters that were most frequently mentioned in CLD proposals. (PHY, MAC, NET, TRANS, and APP are abbreviations for the physical, medium access control, network, transport, and application layers of the TCP/IP protocol suite respectively).

CLDs target the network vertical dimension addressing primarily the configuration of the on node network stack. CLDs aim to preserve the layered architecture, yet, reduce the rigid order imposed by the layered paradigm, as well as alleviate the constraint imposed on vertical layer integration. CLDs are aimed primarily at wireless environments and address the need for inducing adaptability in face of the unstable wireless link conditions by allowing protocols at nonadjacent layers to communicate and expose their internal states. Preserving the layered architecture, CLDs are compatible with the current network realizations, which ease their integration into the TCP/IP suite. Yet, it is argued that integrating several CLDs with the Internet protocol suite may work at cross purposes, leading to unintended side effects, and eventually overall performance deterioration. Besides, stifling future development since changes can no longer be confined to one layer [52].

2.5.2 FIND Proposals

SILO architecture [53], a FIND proposal, aims to integrate cross-layer designs and optimizations into the Future Internet. The SILO architecture main components are services, a control agent, and a set of rules and constraints that dictates services associations and orderings. In SILO a service is an abstract fine-grained, well-defined, self-contained, atomic function to be performed on application data relevant to a specific communication task. A service is described using an ontology of functions, interfaces, and control parameters called knobs. A service implementation is called a method. Several methods (implementations) could exist for the same service. Methods are ordered to build vertical stacks called *silo*. The control agent is an entity that resides inside a node hosting the SILO architecture. The control agent dynamically custom-builds a silo for each initiated connection according to service associations and ordering rules, while taking into consideration requirements, policies and resources. Each instantiated silo is associated with a given traffic stream, possesses a state that is a union of all constituents methods states, as well as any shared state resulting from cross method interactions. A silo persists for the duration of the connection. The control agent is capable of adjusting the knobs of an instantiated silo during runtime to optimize performance. Moreover, the control agent may optionally be able to communicate with other control agents residing on other nodes in the network in order to further optimize a silo behavior.

RNA [54] is another FIND proposal based on recursive composition of a single configurable protocol structure. RNA avoids recapitulation of implementation as well as encouraging a cleaner cross-layer interaction, since the use of a single meta-protocol module facilitates the inter-protocol interactions at different layers. RNA uses a single meta-protocol as a generic protocol layer. This meta-protocol includes a number of basic services, as well as hooks to configurable capabilities. This meta-protocol provides the building block from which layers are instantiated forming dynamic stacks. Each layer of the stack is tuned to the properties of the layers below it. RNA uses a MultiDomain Communications Model (MDCM), which is a structured abstract template to delineate both next-hop and next-layer resolution, where either can be chosen at runtime.

SILO and RNA have been presented as clean-slate architectural attempts towards Future Internet. However, layering, as a design paradigm, is still *the* prevailing model. An essential goal of both proposals is to gracefully embrace cross layering into the present network stack. In [53] the authors argues in favor of layering for modularization and abstraction. They propose “rejuvenating the layering concept” by relaxing the traditional rigid boundaries and strict ordering among layers and allowing any set of services to be selected dynamically for a particular task. While in [54] the protocol layering is still the basic design but it provides freedom in protocol organization to create dynamic custom-build stacks. Although considered clean-slate architectures, we argue that by adhering to layered stacks as the underlying model, both proposals might suffer form shortcomings attributed to the Internet model. First, both architectures do not give guidance to engineers as how to handle cross interests among composed protocols: The single control

agent in SILO, as presented, is a monolithic unit representing a single point of failure for all protocols working under its control, as well as imposing scalability problem as service diversity, granularity, and operational parameters increase. Needless to say the expected code complexity, since it is required to be aware of all services, their compositional logic, polices and constrains. As for RNA MDCM, we note that by confining the logic for horizontal and vertical interlayer communication into a single entity is a very challenging task that is error prone. Furthermore, it lacks explicit representation for interactions leaving it to be decided on at runtime. This allows for implicit assumptions to creep into protocol design and implementations. Second, both architectures have undermined monitoring and resource management failing to express both functions as first class architectural constructs. Finally, as presented, both architectures focus on the vertical dimension of the network without suggesting how the horizontal dimension will be incorporated.

FIND proposals targeting the horizontal network dimension are Content Centric Networking (CCN) and Scalable Internet Architecture presented in [55] and [56] respectively. CCN makes the address in packets correspond to information or elements reachable on the Internet rather than machines. It proposes a layered node model that mimics in structure the TCP/IP layering model but differs in layer responsibilities. The waist of the CCN layered model exists at the content layer replacing IP. CCN introduces security as a standalone layer above the content layer. In Scalable Internet Architecture there was no direct reference to node structure. The main focus was on the addressing schemes and how it maps to the socioeconomic and business dimensions realized in the present Internet.

2.5.3 FIRE Proposals

ANA [57] is an EU initiative aiming to frame autonomic networking principles and architecture, as well as tp build a prototype for autonomic networks. ANA main architectural abstractions are, Functional Blocks (FB), Information Dispatch Point (IDP), Information Channels (IC), and network and node Compartments. ANA architecture is built on the concept of indirection manifested through the use of IDPs. An IDP represents the entry point to all ANA's abstractions and allows for transparent change in the data flow path. An FB in ANA is an information processing unit that implement data transmission function, or additional functionalities such as traffic monitoring. FBs existing within local scope of each other exchange data through the IDPs attached to each directly. While for remote FBs each FB's IDP need to be first attached to the IDP of an IC. To support communication between any pair of networking functionalities ANA's constructs export a generic API. Scope in ANA is defined using the concept of a compartment, which represents a set of FBs, IDPs and ICs with some commonly agreed set of communication principles, protocols and policies. Two types of compartments are defined in ANA. The node compartment represents a local and private execution environment for ANA clients, while the network compartment represents a network

instance encompassing several ANA nodes, and involve communication across an underlying network infrastructure. ANA's network compartments exist at the waist of the TCP/IP model and provides the means to federate multiple heterogenous networks. Internally, a network compartment has a variety of choices regarding naming, addressing policies, packet headers, etc. Externally, each network compartment supports a generic compartment API, which provides the glue that permits to build complex network stacks and packet processing paths.

Indirection in ANA gives a solution to data flow flexibility, however, the ANA's framework does not give guidance as to how each network internally should be built, reasoning that leaving internals unspecified provides flexibility and accounts for future change. To the contrary, we argue that, to support future changes, the community's compelling need is a clear definition of ground network design principles and specifications, according to which all networks need to be developed and synthesized. These ground principles and specifications will be common to all networks regardless of network's heterogeneity in terms of purpose, size, underlying technology, or running applications. Defining network core-design principles will promote global network system integrity leading to seamless integration and federation. Furthermore, although developed primarily towards autonomic behavior, ANA's abstractions do not enforce, or even imply monitoring activities, which is a principal functionality for supporting autonomic behavior.

Haggle [58], is a FIRE architectural proposal targeting Pocket Switched Networks. Haggle is a non-layered architecture that enables applications to be infrastructure-independent. Haggle architecture targets mobile node configuration dividing the on node communication functionalities into four modules; delivery, user data, protocols, and resource management. Haggle architecture is motivated by the need to support mobile applications. An important concept that Haggle introduces is the Application Data Unit (ADU). ADU is the unit of transmission in the Haggle architecture, and as the name suggests an ADU is a data item meaningful to the end application. Haggle was presented at an abstract level with no details on realizations in terms of protocols and their interactions.

BIONETS [59] is another EU research project focusing around the definition of a biologically-inspired design toolkit for autonomic networks and services. At the network level, the BIONETS system is composed of clouds of mobile nodes that are connected among themselves, but potentially disconnected from any IP network or backbone, plus any number of sensors or embedded sources of contextual data. At the application and user level the BIONETS system is comprised of users interacting with each other and with a range of services and applications that are supported by the local cloud and by the other sensors. BIONETS overcomes device heterogeneity and achieves scalability via an autonomic and localized peer-to-peer communication paradigm. Services in BIONETS are also autonomic, and evolve to adapt to the surrounding environment, like living organisms evolve by natural selection. Biologically-inspired concepts permeate the network and its services, blending them together, so that the network moulds itself to the

services it runs, and services, in turn, become a mirror image of the social networks of users they serve.

BIONETS [59] project provide very useful and interesting insights into different aspects of computer communication. BIONETS research highlights the social dimension of technology in supporting sustainable business models. In addition, it incorporates biological, physical and mathematical models into the design of user level, service level and network level protocols. BIONETS does not provide a standalone framework for network design and protocol engineering. However, we, conjecture that BIONETS contribution in terms of bio-inspired theoretical foundations, principles and protocols can be further culminated if incorporated within a bio-inspired network architecture.

2.5.4 Protocol Environment and Roles

Protocol environment present an alternative to layered architecture. The main idea of protocol environments is to break the communication hierarchy with no implied sequence for communication function execution. This approach in protocol design, as opposed to the layered approach, deals with protocols as standalone entities, completely decoupled, with each entity being responsible for a primitive function. More complex functionalities can be attained by instantiating the required entities. Protocol environment was proposed in [60] and [61]. Flexible Protocol Stacks (FSA) proposal in [60] introduces flexibility to the OSI protocol stack by modeling it as a 'protocol environment' populated by standard communication functionalities represented by protocol entities. The aim is to enable the running application to tailor the underlying protocol stack from different standard protocol entities at run time through a bootstrapping procedure according to the application's needs. The protocol environment can be viewed as communication server encompassing a repository of static protocol entities. The server regulates the use and instantiation of protocol entities on demand depending on the availability of the communication service requested, and the configuration of the real system. In [61] component-based design is used to map coarse-grained protocols to fine-grained decoupled components that could be configured and re-used in different contexts. In the context of component-based design, a protocol environment is a component that integrates all end-to-end protocol functionality from other decoupled components according to the needs of a particular application.

This design concept has the advantage of providing complete freedom to structure how the communication process will be performed. Yet, it does not offer guidance as to how this can be achieved, leaving the decision to the implementation phase. Thus architecturally the design dose not constraint engineering options. Yet, lack of clear guidance as how the protocol entities interaction will be performed may result in wrong implementation decisions that do not preserve the flexibility offered in the architectural model. In addition, the more a system is broken down into components the more complex to tackle interactions will be, since more interfaces need to be defined. On the other hand, interfaces are where standardization takes place implying that changes ought to be minimal, if any, rendering the system hard to evolve and innovate [35]

Another proposed alternative to the layered architecture of the present network, is Role Based Architecture discussed in [62]. Instead of using protocol layers, RBA organizes communication using functional units that are called roles. Roles are not generally organized hierarchically, so they may be more richly interconnected than traditional protocol layers. It is argued in [45] that RBA is a major improvement in comparison to layered architecture by allowing explicit signaling of functionalities, and allowing components compromising the network to be explicitly identified. For explicitly identifying network components, RBA re-modularizes current "large" protocols such as IP, TCP and HTTP into somewhat smaller units that are addressed to specific tasks as packet forwarding, fragmentation, flow control, byte-stream packetization, request web page, or suppress caching. Each of these functions will be performed by a specific role in RBA model. A protocol header in RBA no longer forms a "stack", but rather a heap. That is, packet headers are replaced by a container that can hold variable-sized blocks of metadata, referred to as Role Specific Headers (RSH). These blocks may be inserted, accessed, modified and removed in any order by modular protocol units. In RBA, there are no imposed rules on the execution of protocol modules; however an appropriate partial ordering is required among specific roles.

Protocol environments and roles address the vertical dimension of the network concentrating basically on node protocol configuration. Both are clean-slate proposals departing form layering. However, as presented, both proposals are not sufficient as guiding models for network architecting and protocol engineering.

2.5.5 Architectural Comparisons

Table 2.3 summarizes the architectural differences and similarities for each of the overviewed proposals. The following can be deduced form the table

- By examining the design objectives of the reviewed proposals, one finds that they focus on a single problem rather than presenting a holistic perspective toward network design. The most commonly stated objective targets protocol realizations and flexible implementations.
- Several clean-slate designs are still adhering to concepts and abstractions of the legacy Internet, taking them for granted to be the starting point of any new design.
- The need for cross-protocol interactions is highly stressed as indicated in the number of designs directed towards cross-layer designs. Some designs propose a central entity to manage the interaction, while others adopt a distributed approach.
- Flexible protocol composition and alternative data flow paths have been proposed by several designs highlighting the need for allowing protocol suites to be dynamically created, while supporting a variety of service models
- Explicit out of band signaling emphasizes the need for separating control and data planes

Other notions presented include the need to incorporate a social-economical dimension into network realizations, the separation of name and location, and incorporating bio-inspired behavior into protocol operation.

2.6 Conclusion

In this chapter we focused on exploring the alternative network designs from the time when computer networks were considered as an experimental endeavor, to the present, where the Internet has proliferated everyday life. We overviewed the factors and decisions that shaped the Internet rendering it in its present state. Our main observation is that in spite of the extensive research in the area of computer networks, the general trend towards network science and design lacks a systematic formalization of core principles to guide the process of network engineering. These core principles should point out the main features and characteristics that need to exist in any network design. Second, in spite of the limitations attributed to layered architecture, and the call to abandon layering, layering, as a model, is still the *prevailing* design paradigm in most proposals demonstrating *the* reference model to which all other models, proposals or ideas are compared. Third, to our knowledge, proposals described in the literature do not present an overarching network model, since they primarily address a single network dimension, while ignoring the other.. Fourth, with the increasing complexity of computer networks, there is a stressing need for self-monitoring and regulation to alleviate the burden of laborious network management and continuous human intervention. However, monitoring as an architectural principle has not been emphasized or expressed as a standalone functional requirement

Cross Layer Design	Design Rationale
Hints and Notifications (HAN) [48]	Enabling a backward-compatible partial deployment of inter-layer communication
Cross Layer Shortcut Signaling (CLASS) [49]	Propagating cross-layer messages through local out-of-band signaling shortcuts, thus breaking the layer ordering constraints while keeping the layered structure
Interlayer Coordination Model [44]	Coordinating among different functions that need cross layer interaction for improving overall performance. The coordination is done by introducing a central entity that will receive notifications about different events and invoke the appropriate cross layer management algorithms accordingly
ÉCLAIR [50]	➢Rapid Prototyping: enable easy development of new cross layer feedback algorithms, independent of existing protocol stacks ➢Minimum Intrusion: Enable interfacing with existing stack without any significant changes ➢Portability: Enable easy porting to multiple systems ➢Efficiency: Enable efficient implementation of cross layer feedback
Cross-layer Coordination Framework [51]	Defining a framework that manages the interaction between different layers of the protocol stack while satisfying ➢Modularity: It is desirable to preserve the modularity of the protocol stack while enabling interaction between them ➢Scalability: easily adding several new adaptations algorithms to a particular layer or different layers without affecting overall performance ➢Flexibility: No performance degradation due to Cross layer interactions

Table 2.1: Cross Layer Designs' Rationale

Parameter	From	To	Purpose
			Lower to Upper
Channel State Information (CSI)	PHY	MAC	CSI such as BER and SNR are used by the MAC layer to decide on the optimum transmit rate.
Energy level	PHY	MAC	According to the energy level, the MAC decides on the data rate and transmit power to be used in transmitting packets considering application tolerance in terms of packet losses and received errors
Channel State (bandwidth, error rate)	MAC	APP	APP decides on compression, FEC, and ARQ techniques
		TRANS	Adapt the congestion window and the RTO to channel state
		NET	Build QoS aware routes
Packet Size	MAC	APP	APP decides on the packetization and FEC strategies
Link Delay	MAC	NET	NET calculates end-to-end path delay
Contention level expressed as increase in backoff	MAC	TRANS	TCP slows down its rate as well as increases the RTT so as not to retransmit packets whose ACK has been delayed due to MAC contention
Connection status	MAC	NET	➢NET responds to layer 2 PRE and POST handoff triggers to perform seamless handoffs ➢Update one hop neighbor list used in routing tables
		TRANS	TCP can correctly interpret packet losses and invokes congestion avoidance mechanisms only in case of congestion
Operation Mode	MAC	TRANS	TCP need to adjust the congestion avoidance mechanism according to the type of network (wired, infrastructure wireless, infrastructure-less wireless) in which it is operating
Path delay in the wired part of the route	NET	TRANS	TCP estimates link capacity according to total path delay
Route Connection Status	NET	TRANS	TCP needs to be informed in case of handoff so as to correctly interpret packet losses
			Upper to Lower
Delay/Jitter limit	APP	MAC	➢Adapt the retransmission strategy ➢Adapt data transmit rate
		NET	Build QoS aware route
Packet priority	APP/ TRANS	MAC	According to packet priority the MAC layer can schedule transmission for flows from different applications and for packets pertaining to the same flow.
Packet content			MAC adapts the number of retransmission attempts according to the packet content. TCP packets are not to be discarded, UDP packets need to conform to jitter limit set by the APP

Table 2.2: CLD Frequently Used Parameters

Proposed Design		Network Dimension	Unit of Abstraction	Approach	Design Objective
Cross-layer Designs	HAN [48]	Vertical	Layers	Incremental	Facilitate inter-layer communication through option field in Packet Headers
	CLASS [49]	Vertical	Layers	Incremental	Facilitate inter-layer communication through out-band signaling
	Interlayer Coordination Model [44]	Vertical	Layer	Incremental	Facilitate inter-layer communication through a central managing entity
	ÉCLAIR [50]	Vertical	Layers	Incremental	Architecturally allowing asynchronous cross layer feedback, where protocol adaptation is done out of phase with the protocol operation
	Cross-layer Coordination Framework [51]	Vertical	Layers	Incremental	Facilitate inter-layer communication through a central entity (server) and layer representatives (layer client)
FIND Proposals	SILO [53]	vertical	Layered Services	*Described as Clean-slate	Gracefully accommodating cross-layer design into the Internet architecture
	RNA [54]	Vertical	Layers	*Described as Clean-slate	Retrofits the notion of choices in the protocol stack while encouraging a cleaner cross-layer interaction
	CCN [55]	Horizontal	Layers	*Described as Clean-slate	Routes on Content rather than machine addresses
	Scalable Internet Architecture [56]	Horizontal	undefined	**Described as Clean-slate	Mapping addressing schemes to socioeconomic and business layouts
FIRE Proposals	ANA [57]	Horizontal	Functional Blocks	Clean-slate	Frame autonomic networking principles and architecture
	Haggle [58]	Vertical	Modules	Clean-slate	Targets Pocket Switched Networks
	BIONETS [59]	Horizontal	Undefined	Not Applicable	Bio-inspired design toolkit for autonomic networks and services
Protocol Roles and Environment	FSA [60]	Vertical	Protocol entities	Clean-slate	Enable the running application to tailor the underlying protocol stack from different standard protocol entities at run time
	Component-based Protocol Structure [61]	Vertical	Protocol Component	Clean-slate	Create a Protocol environment encompassing end-to-end functionalities required by a running applications
	RBA [62]	Vertical	Roles represented as modular protocols units	*Described as Clean-slate	Explicit signaling of functionalities and allowing components compromising the network to be explicitly identified

Table 2.3: Architectural Comparisons

* These designs although described as Clean-slate are still based on the layered paradigm

** The main innovation is in the flexible addressing scheme however still it is still based on the legacy protocol stack

Chapter 3

CORM: Design Principles, Approach, and Model

"An awareness of fundamental design principles enables engineers to engage in the highest level of decision-making – to which they can then bring their professional skill and training"

Royal Academy of Engineering

Designing future computer networks dictates an eclectic vision capable of encompassing ideas and concepts developed in contemporary research, yet unfettered by today's operational, and technological constraints. However, unguided by a clear articulation of core design principles, the process of network design may be at stake of falling into similar pitfalls and limitations attributed to current network realizations. Therefore in the process of deriving CORM, we were determined to enunciate core design principles that express the most fundamental characteristics of computer networks, regardless of their size, purpose or operational context.

In this chapter we present CORM derivation process, starting with delineating our design principles and approach, then proceeding to detail the different components of our reference model and its distinctive features.

3.1 CORM Design Principles

In crafting CORM design principles, we aimed to address computer network design at the highest level of abstraction. On the one hand, distributed computer networks, such as the Internet, stand as a typical example of complex systems [10], [64]. On the other hand, computer networks operations and performance rely fundamentally on protocols, which can be described as distributed software running on different nodes constituting the network. Therefore, computer networks are a typical example of distributed software-dependent systems. From these two observations, we assert that a computer network is a software-dependent complex system, leading to our two ground design principles detailed in the following subsections.

3.1.1 Principle I: A Computer Network is a Complex System

The study of Complex systems has been the focus of interdisciplinary research. Its importance rises from the fact that different diverse social, natural, economic and physical systems have been shown to exhibit several commonalities. Such commonalities

have promoted the interest of researchers to come up with a unified framework within which different complex systems can be analyzed and formulated. So far there is no consensus on the definition of Complex Systems in the literature although their general properties are well recognized.

Informally, complex systems can be defined as a large network of *relatively* simple components with no central control in which emergent complex behavior is exhibited as a result of component interactions.

> The complexity of the system's global behavior is typically characterized in terms of the patterns it forms, the information processing that it accomplishes, and the degree to which this pattern formation and information processing are *adaptive* for the system - that is, increase its success in some evolutionary or competitive context. [10].

This definition highlights three characteristics for complex systems.

- Complex systems are composed of relatively large number of autonomous entities: This property implies the distributed control and structure of complex systems, as well as their indeterminate nature. Entities in a complex system exist independent of each other (no insinuated global structure or imposed hierarchy). Yet, act interdependently, affecting and getting affected by each other;
- Complex systems exhibit complexity: Quantitatively, complexity refers to the amount of information required to depict the system at the micro and macro levels [65]. Complexity arises, not only from the myriad intricate interactions occurring at the micro level among the system components, but more notably, form the feedback and influence of the macro level resultant behavior on decisions taken at the micro level. In other words, the mapping from individual actions to collective behavior is non-trivial giving the system its discernible emergent behavior property; and
- Emergent behavior: In complex systems, emergent behavior refers to the ability of the system's components to change/evolve their structures and/or functions without external intervention as a result of their interactions, or in response to changes in their environment [10]. Emergent behavior results in global level system stability, in spite of operating away from equilibrium at the local level. Therefore, complex systems are frequently described as being at the "edge of chaos"[67]. A metaphor used to describe the system's reaction to minor context/environmental changes, by shifting from one state of order to another to better cope with the induced changes. Emergent behavior can be further classified as self-organization, adaptation or evolution [12].
    - Self-organization refers to changes in component individual behavior due to inter-component communication.
    - Adaptation refers to changes in components' behavior in response to changes in the surrounding environment. Both self-organization and adaptation imply

information propagation and adaptive processing through feedback mechanisms.

- Evolution refers to a higher form of intelligent adaptation and/or self organization of components in response to changes by accounting on previously recorded knowledge from past experience(s). Evolution usually implies the presence of memory elements, as well as monitoring functions in evolvable components.

After delineating the properties of complex systems, we need to highlight the following points:

- We have previously pointed out that current network realizations, based on the Internet model, have been cited as an example of complex systems. However, in contrast to complex system emergent behavior, present network realizations are characterized by emergent ill-behavior, where a local event may have a destructive global effect realized as cascading meltdowns that might require human intervention for correct network operation [7], [67]. Furthermore, up to our knowledge, adaptation in present network realizations is crude (e.g. TCP aggressive cut down on congestion window size regardless of the reason for packet drops), lacking the evolvability capability intrinsic to all complex systems. This renders the network to be oblivious to changing patterns and trends in operating conditions and performance requirements. In the literature, the term Adaptive Complex System (CAS) has been used to designate systems that exhibit emergent behavior in contrast to those that exhibit emergent ill-behavior, and likewise we will be using the term CAS hereafter to refer to complex systems that possess emergent behavior.
- In context of CAS, stability is different from equilibrium. Stability refers to the ability of the system to maintain stable global patterns, in spite of the unpredictable interactions occurring at the local level. These overall stable patterns are a direct consequence of the local chaotic agitation, where elements composing the system operate at conditions that are far from equilibrium [15]. In other words, for CAS to be stable, its components' states alternate between order and chaos.
- CAS strive to fit their context. However, fitness within the context of CAS means to be able to find approximate solutions to difficult problems, rather than being able to find *the* best solution. Finding *the best* solution may be impossible due to the multitude of possibilities, limited time allowed for reaching a decision, and restricted vision of context [68].

3.1.2 Principle II: A Computer Network is a Distributed Software System

Software Engineering (SE) refers to using systematic, disciplined, quantifiable approaches to the development, operation and maintenance of software [69]. To manage

software functional complexity, SE defines the concept of Separation of Concerns (SoC), which is a general problem-solving technique that addresses complexity by cutting down the problem space into smaller more manageable, loosely-coupled, and easier to solve sub-problems, thus allowing for better understanding and design [11]. SoC is a dual facet concept addressing concerns from two different views. The first view identifies concerns according to the system requirement specifications. It classifies concerns into core concerns and crosscutting concerns. Core concerns refer to core functional requirements of a system that can be identified with clear cutting boundaries. Hence, can be represented in separable modules or components resulting in loosely coupled systems. On the other hand, crosscutting concerns are nonfunctional aspects of the system that span over multiple modules or components, trying to manage or optimize the core concerns, and if not carefully represented, result in scattering and tangling of system behavior [11]. In that respect, concerns will be identified according to the domain in which the system will operate. The second view of the SoC concept addresses system configuration differentiating between system components, representing loci of computations, decisions and states, and system connectors, representing component interactions facilitating information flow and state communication [70].

3.2 CORM Methodology: Concern-Oriented Bottom-Up Design Approach

Our proposed Concern-Oriented Bottom-Up design approach follows directly from our design principles. The Bottom-Up approach is derived from property 1 in Principle I, and the second view of SoC concept in Principle II. Both accentuated the significance of the basic building block or entity composing the network system. The Concern-Oriented paradigm represents our vision in network functional and structural decomposition.

Computer networks need to be designed along two main dimensions; a vertical dimension that addresses structure and configuration of network building blocks, and a horizontal dimension that addresses communication and interactions among the previously formulated building blocks. In adopting a Bottom-Up approach, our main focus will be the vertical network dimension delineating the network building block responsibilities and capabilities. From a CAS perspective, these network building blocks need to possess adaptability, self-organization and evolvability as intrinsic features. The network will then be constructed by composition from these building blocks. From a SE perspective, applying the second view of the SoC principle, network building blocks will represent the loci of computation, decisions and states encompassing the network concerns (core as well as crosscutting concerns), while their interactions and communications identify network connectors instantiating the network horizontal dimension.

Our Concern-Oriented Bottom-Up design methodology does not differentiate between network core and network edge in terms of capabilities, thus it contradicts the

End-to-End (E2E) principle that has been central to the Internet design. The E2E principle dictates that 'mechanisms should not be placed in the network if they can be placed at the end nodes, and that the core of the network should provide a general service, not one that is tailored to a specific application'[36]. It has been argued that the E2E principle has served the Internet well by keeping the core general enough to support a wide range of applications. However, we contend that, taken as an absolute rule, the E2E principle constrained core evolvability rather than fostered its capabilities ***rendering the Internet biased to those applications that can tolerate its oblivious nature*** forcing designers and protocol engineers to adopt point solutions to compensate for core deficiencies. Another consequence to our proposed bottom-up network composition is contradicting the prevailing misconception of abstracting a network in terms of an inter-network. Adopting a bottom-up approach to network composition implies recursive construction of the inter-networks from networks, which are likewise recursively constructed form network components, which are constructed from one or more network building blocks.

3.3 CORM: A Concern-Oriented Reference Model for Computer Networks

A network reference model is an abstract representation of a network. It conveys a minimal set of unifying concepts, axioms, and relationships to be realized within a network [63]. For expressing a multi-dimensional system such as a computer network, multiple abstract representations are required to capture the network from different perspectives. CORM abstracts a computer network in terms of its requirement specifications, structure and Interactions, represented as the network-concerns conceptual framework, the network structural template, and the information flow model. We hypothesize that these three CORM facets are necessary and sufficient for depicting a frame of reference for designing computer networks. Deriving the proof of this statement will constitute part of our future work.

Focusing primarily on the vertical dimension of the network, our research in this dissertation details the first two components of CORM; the concern-oriented conceptual framework and the network structural template. As for the horizontal network dimension, we will provide a synopsis of the information flow model which is a major subject in our future work.

3.3.1 ACRF: The Conceptual Framework for Network Concerns

The conceptual framework was derived according to our interpretation of the network requirement specifications. We postulate that the basic requirement for computer networks can be expressed by the following statement:

"***The network is a communication vehicle allowing its users to communicate using the available communication media***".

Accordingly, we identify the network users, the communication media, and the communication logic as the primary requirements, which the network design need to address and plan for. Applying the concept of SoC to our analysis, we identify four main network concerns; Application Concern (ACn), Communication Concern (CCn), Resource Concern (RCn), and Federation Concern (FCn). The first three are core network concerns encompassing the network functional requirements, while the fourth is a crosscutting concern (non-functional requirement) representing the area of intersection or common interests among core concerns. Elaborating on each concern we have:

- The ACn encompasses the network utilization semantics; the logic and motivation for building the network, where different network-based end-applications (network users) can be manifested.
- The CCn addresses the need for network route binding to provide an end-to-end communication path allowing network elements to get connected (communication logic)
- The RCn focuses on network resources, whether physical or logical, highlighting the need for resource management to efficiently address different trade-offs for creating and maintaining network resources (communication media).
- Finally, FCn orchestrates interactions, resolving conflicts and managing cross interests, where areas of overlap exist among the aforementioned concerns.

Our network concern identification is manifested as the conceptual framework for network concerns, referred to hereafter as ACRF, to which network global behavior needs to be mapped, i.e. ACRF represents the network genetic code to be realized along both network dimensions, vertically on the network components and horizontally among network components.

Analyzing the Internet model (vertical dimension) and present network realization (horizontal dimension) with respect to the ACRF framework, we note that the Internet layered model vertically accounted for the ACn, CCn, and RCn, but did not satisfy concern separation; a single concern was split along two layers. Moreover the Internet layered model did not account for the FCn limiting inter-layer communication to a minimum set of primitives conducted by the use of protocol headers. Figure 3.1 illustrates the ACRF mapping to Internet layers. As for the horizontal dimension, present network realizations account for both ACn and CCn realized as servers and server farms for the former, and routers, switches, and DNS for the latter. Yet, the RCn and FCn are missing from present realizations leading to point solutions for resource and network management respectively.

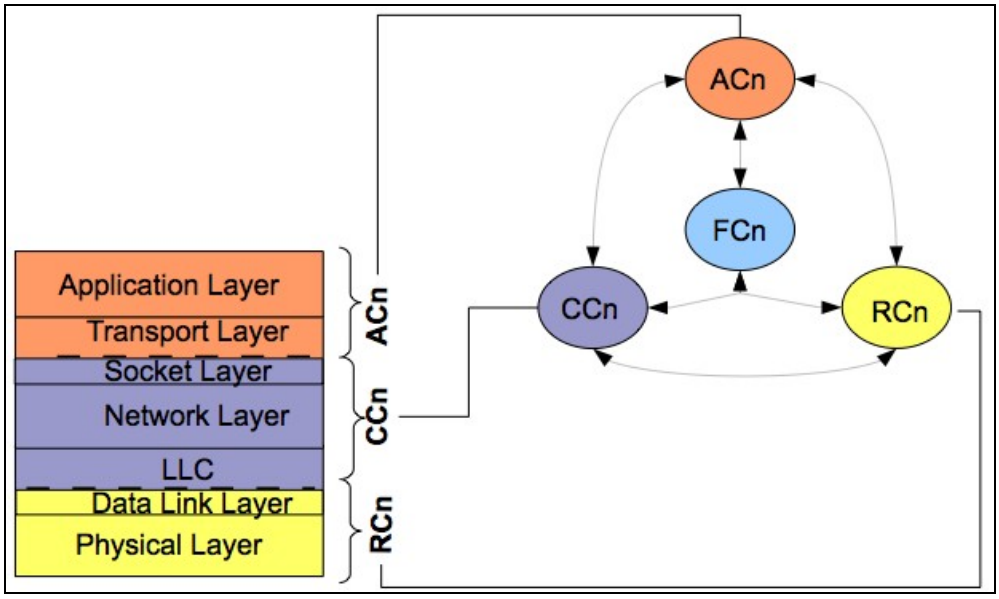

Figure 3.1: ACRF Mapping to the Internet Layered Model

3.3.2 NST: Network Structural Template

CORM networks are composed of network components that communicate through communication substrate. The network components are composed of one or more active network building blocks, where computations and decisions take place, while the communication substrate represents a passive media for information flow and exchange. Being the primary constituents of a software-dependent complex system, CORM-based network building blocks need to possess adaptability, self-organization and evolvability as intrinsic features, thus mimicking bacterial cells in a bacterial colony; our adopted model of CAS [13]. Accordingly, we define the Network Cell (NC), CORM-based network building block, as a self-contained computational/decision entity capable of monitoring its state, adapting to perceived conditions, inferring decisions, recording its experience, and eventually evolving through self-learning and intelligent adaptations. One or more NCs make up a Network Component (Ncomp), which we define as the basic network entity capable of end-to-end communication. The internal structure of an NC and bottom-up network composition template are detailed in the following subsections

3.3.2.1 The Network Cell:

The Network Cell structure and behavior are inspired by observations recorded in a recent study on primordial bacteria that provided a vivid representation of the main features that need to be present in entities composing a self-engineering CAS [13]. According to this study the following four cell capabilities are essential for a network to possess emergent behavior.

- The cell should be aware of its surrounding environment as well as of its inner states;
- The cell should be able to reason about its perceived states and decide on its course of action that best serves its goal function;
- The cell should be able to memorize previously inferred decisions and learn from past experience; and
- The cell should be able to communicate and cooperate with other cells to achieve the high level goals of the whole system.

To beget these capabilities, we construct the NC out of four units: the Interface Unit (IU), the Monitoring Unit (MU), the Regulatory Unit (RU) and the Execution Unit (EU). NC structure is shown in Figure 3.2, which represents the structure of a generic NC. By generic we mean that this structure will be common to all NCs regardless of their assigned responsibilities or roles. The NC has two modes of concurrent operations; intrinsic operation and functional operation. The intrinsic operation is again common to all NCs and represents the NC's genetic blueprint that can be regarded as the sequence of actions and rules that the NC must obey throughout its lifetime. On the other hand, the functional operation of the NC is assigned to it on its creation and prescribes the role that the NC must perform. This includes the function realized and instruction set to be executed by all units. Once the NC is assigned a functional role, it turns to be a specialized NC. Therefore, the generic NC is just a template out of which all specialized NC can be derived. It is also possible for a specialized NC to change its function during its lifetime or alternate between different functions depending on its assigned role(s). Once an NC materializes by assuming a functional role, it will be assigned a portion of the system resources according to its functional needs.

Following is an outline for the intrinsic operation of each of the units shown in Figure 3.2:

- *Interface Unit (IU):* The IU is the NC boundary allowing it to communicate with the outside world (environment or peer NCs). Through IU the NCs receive and transmit different forms/representations of data (states, instructions, control, content, etc.).
- *Monitoring Unit (MU):* The MU is responsible for monitoring the states of the input/output flows directed into or out of the NC, as well as the assigned recourses usage level. MU will extract state information and represent this information in a quantifiable format. These quantified states are then stored in the state database to be retrieved upon requests received form Regulatory Unit.
- *Regulatory Unit (RU):* The RU has two regulatory cycles one is inherent and the other is initiated. The inherent cycle is always in operation, and checks that the resource usage levels and performance parameters are always within the set thresholds. The initiated cycle is either triggered by requests received from EU asking for advices and recommendations for performance enhancement, or due to

performance deterioration realized through monitoring. Accordingly, RU inspects environmental parameters to infer the reasons that accounted for performance deterioration. This step may lead to communication with neighboring peers requesting their views of the environment. RU has the capability of gaining knowledge and learning from past experience, which it records in the Knowledge/Experience database. Therefore, RU provides educated recommendations to EU to optimize its operation. In addition, RU may update some of the threshold levels according to its inferred decisions.

- *Execution Unit (EU):* The EU is responsible for executing the function assigned to the NC. Functions assigned to an NC are usually accompanied by a pool of libraries that can be used to formulate different ways of accomplishing the required function. EU is composed of two main components; the Logic Component (LC) and the Resource Use Controller Component (RUCC). LC is the part responsible for performing the NC function. LC starts by creating a set of instructions that best accomplishes the required function using the code library pool. It also requests RU recommendations, thus incorporating both RU knowledge and experience, as well as accounting for environmental alterations. Once LC receives feedback recommendations from RU, it might update its tailored instruction set to fit the inferred operational status. RUCC, on the other hand, is responsible for managing resources assigned to the NC. RUCC works together with LC to ensure optimal internal resource usage and distribution. RUCC is also responsible for estimating the required level of resources for the NC operation as a whole, and thus requests the estimated resources from the system.

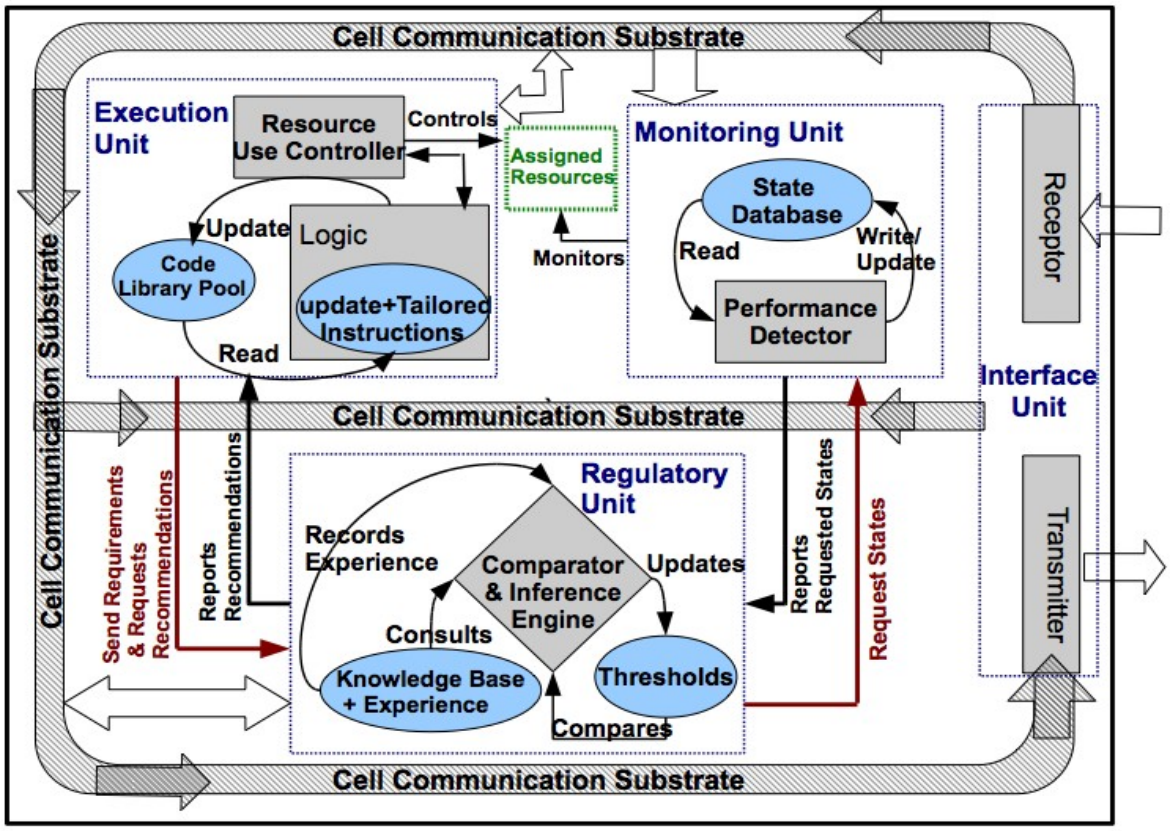

Figure 3.2: The Network Cell

The NC units communicate through the *cell communication substratum* that represents the media into which information is temporarily deposited to be later consumed. The function assigned to a specialized NC will further identify operational aspects of each of the NC units such as the parameters to be monitored and states recorded by MU, the threshold values to be used by RU, and the code library pool out of which the logic is tailored and executed by EU. Recalling that the ACRF represents the network genetic code to be realized throughout the network dimensions, and the NC represents the basic building block encompassing the network-concern space for CORM networks, we note that any functional role assigned to the NC will be decomposed according the ACRF as illustrated in Figure 3.3.

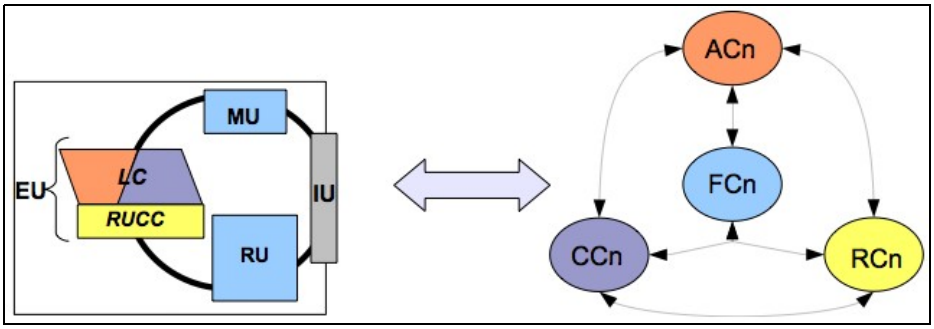

Figure 3.3: ACRF realization within CORM-NC

3.3.2.2 Network Compositional Logic

Network compositional logic defines the bottom-up network construction out of network entities identifying the different interaction boundaries that can occur among network entities. Network entities can be classified, as mentioned before, into computational/decision building blocks (NC and/or Ncomp) that produce and/or consume network data, or passive media communication substrate, where network data can flow. Network Compositional Logic stems from our bottom-up definition of network and inter-network construction. We define a computer network as two or more Ncomp connected by a communication substratum, where Ncomp interactions are sustained, despite the heterogeneity of the hardware, middleware, and software of the connected Ncomps. While an inter-network is defined as two or more networks connected by communication substrate, where interactions among Ncomps residing within each of the connected networks are sustained, despite the heterogeneity of the hardware, middleware, and software employed by the Ncomp composing the connected networks.Ncomps communicating over the inter-network perceive the connected networks as a single network. Integrating NC, Ncomp, network and inter-network definitions into the Network Composition Logic we derive CORM NST and define it, using EBNF, as follows:

CORM NST EBNF formal Definition:

Notations

- Trailing * means repeat 0 or more times
- Trailing + means repeat 1 or more times

Abbreviations

- NC = Network Cell
- CCS = Cell Communication Substratum
- Ncomp = Network Component
- Net = Network
- NCS = Network Communication Substratum
- INet = Inter-network

Grammar Definitions

- NC = MU RU EU IU CCS

- Ncomp = NC (CCS NC)*
- Net = Ncomp (NCS Ncomp)+
- INet = Net (NCS Net)+

= Ncomp (NCS Net NCS)+ Ncomp

3.3.3 IFM: The Information Flow Model

Information exchange is the essence of computer communication bringing life to the network. It is *the* activity to be observed, measured and evaluated indicating network behavior. Information exchange incorporates two main aspects; the first deals with "meaning" or content of information, and the second deals with actual exchange or "communication" identifying the communicating points and path.

The Information Flow Model captures the aspects of information exchange by defining two main sub-models: Data Representation sub-model (DR) and Data Communication sub-model (DC). Data representation and communication in CORM exist at the vertical network dimension; within the NC between units and between different NCs making up a Ncomp, as well as along the horizontal network dimension; between Ncomps in the same network or across networks. The DR sub-model will provide categorization for the different types of information flowing in the system abiding by the ACRF framework. Therefore, DR is mainly concerned with the "meaning" of information flowing within the network system. The DR sub-model need to handle complexity in terms of the amount of information required to depict the system states at the macro and micro level, taking decisions on the details that need to be exposed and those that need to be suppressed. The DC sub-model, on the other hand, is concerned with communication aspects including interface compatibilities, data formating across different communication boundaries and majorly routing functions encompassing addressing, naming and forwarding. Similar to DR, the DC sub-model will need to address characteristics of complex systems, such as the free scale small world layout [10], when devising the routing functions. The DC sub-model represents a major part of the CCn to be realized by CORM NCs and Ncomps.

3.4 CORM Features

Being developed according to a software-dependent CAS paradigm, CORM refutes the long endorsed concept of layering, presenting the NC as a novel abstraction unit. To our knowledge, CORM is the first reference model that addresses the need for engineering for emergent behavior by accentuating monitoring, knowledge acquisition, and regulation as first class intrinsic features of the basic abstraction unit. Furthermore, we argue that CORM maintains system integrity due to network construction congruency, where Ncomps, networks and inter-networks are defined recursively in terms of the basic abstraction unit – the NC. In addition to the previously mentioned features, CORM facets acknowledge the multidimensionality of the networks, and accounts for concepts and

notions proposed by contemporary designs and architectures including protocol composability out of fine-grained micro-protocols, dynamic protocol adaptation, protocol extensibility and flexibility, cross interest management and control, context awareness through monitoring, resource management as a standalone requirement, and inspired biological behavior and evolution.

Table 3.1 highlights the differences between CORM and the more conventional layered network models (e.g. Internet, OSI, ATM, etc...)

Features/Network Models	CORM	Layered Models
Basic Abstraction Unit (BAU)	NC	Layer
Operation of BAU	Independent: CORM-NC can exist and operate by itself	Dependent: a single layer can never exist or operate by itself
BAU responsibilities	➢Execution of assigned network function ➢Self-monitoring and regulation	➢Execution of assigned network function
BAU Relationships	Interdependent: NC realizes other NCs and cooperate to maximize the over all performance by adapting to context	Incognizant: A layer at one level uses services from the layer below and provides services to the layer above. However, it is unaware of the state of adjacent layers nor realizes the presence of other layers that are at one level further
System level awareness	Global awareness: CORM NCs have a sense of global system goal	Unaware of the global system: Awareness is restricted to layer boundaries
Network Composition	Bottom up recursive	Top-down incremental overlaying

Table 3.1: CORM vs Layered Network Models

3.5 Addressing Future Requirements and Paradigms in CORM

Trustworthiness, virtualization and mobility are frequently mentioned in discussions and proposals targeting future network design in general and Future Internet in specific. Trustworthiness, is considered a central issue to any future network design, taking precedence over any other requirement as stated in the recent NSF summit

summary :‘NSF has stated that while networks with different objectives may have to meet different requirements, no proposals should ignore the requirement for trustworthy operation and suitable security’ [1]. Mobility, is another emphasized requirement, defined as one of the objectives of a Future Internet, where the network-vision for the future will be ‘a network in which mobility is the norm’ [1]. Mobility as a requirement raises other related issues, such as identity, routing, intermittent communication models, and location awareness, while sophisticating others, especially issues related to trustworthiness. Virtualization, on the other hand is not a requirement, but rather a scheme that allows multiple heterogeneous networks to coexist and operate over present network infrastructures [71]. Network virtualization has attracted significant attention in the debate on how to model future networks. Architectural purists view network virtualization as a tool for experimenting with newly proposed architectures; where as pluralists conceive virtualization as a fundamental diversifying attribute of the future architecture itself [71]. In this section we discuss Trustworthiness, Mobility and Virtualization from CORM perspective.

3.5.1 Trustworthiness in CORM

Trustworthiness is a multidimensional concept that have recently gained the attention of the network community. A trustworthy system is described as one that does what people expect it to do—and not something else—despite environmental disruption, human user and operator errors, and attacks by hostile parties [72]. The operation of a trustworthy system is expected to exhibit correctness, reliability, security, privacy, safety, and survivability. As stated in [73], multiple researchers have provided different perspectives on trustworthiness in computer networks. According to which, the basic description of trustworthy networks is; a network where ‘ [the] behaviors and results of network and users are foreseeable and controllable.’ Security is a major aspect of trustworthy behavior that is recurrently emphasized. Within the context of trustworthy networks, the scope of security goes beyond the conventional information security relating to user authentication, access control, accountability, confidentiality, integrity, and privacy [73]. It further involves reducing vulnerabilities of, and mitigating systematic attacks on, network systems. Other aspects of trustworthiness in computer networks, as mentioned in literature, are resilience and controllability. Resilience refers to the ability of the network to operate and maintain an acceptable level of service under the presence of adverse conditions [74]. These conditions could be intentionally or unintentionally created. Resilience is considered a superset of survivability [74]. Survivability in networks is defined as the capability of a system to fulfill its mission, in a timely manner, in the presence of threats such as targeted attacks or large-scale natural disasters [75]. As for controllability, it refers to the presence of effective mechanisms that can identify as well as respond to misbehavior and ill-behavior in network operation. Controllability relies on monitoring system-states to gather trustworthiness information [73].
Accordingly, we can conclude that trustworthiness is not an on/off feature, but rather it is an outcome of behavioral traits that the network need to possess, customize, and

continuously refine, rendering the resultant network behavior to be perceived at different trustworthy levels. We argue that trustworthiness definition in literature is not rigorously delineated. Thus formalized definition for trustworthiness can be considered as the first step towards achieving trustworthy networks.

Trustworthiness in CORM can be realized through defining a trustworthiness profile using the DR sub-model. A trustworthiness profile will encompass parameters form different operating protocols and their range of acceptable values. The created trustworthiness profile will then be continuously monitored, and its parameters evaluated to ensure their operation are within accepted value ranges, using the monitoring and regulation capabilities of CORM. Furthermore, in CORM, different trustworthiness profiles can be created according to the active applications and their operational context, thus providing a flexible way of customizing trust definition according to needs and operating conditions. Thus, we argur that in CORM, Trustworthiness is intrinsically enabled by design. In this respect we are in line with the Hegelian antithesis group design proposal [1], since in CORM, we provide tools that can define, build, and evaluate trust as needed.

3.5.2 Mobility in CORM

The need for mobility was realized with the advent of mobile devices equipped with wireless communication capabilities. The proliferation of such devices in the current Internet have shifted the communication norm from the traditional static model to a dynamic model. Mobility in current Internet is realized through add-on point solutions at the network layer and the application layer. Network layer mobility is achieved by the use of mobile IP (MIP), while application layer mobility is made possible by the use of Session Initiation Protocol (SIP).

Mobility raised several issues in the current implementation, such as the separation of naming from addressing, dealing with intermittent connections, and location-aware modes for routing, while emphasizing further the need for an architecture that address security and privacy concerns [1].

We insinuate that mobility is a behavioral aspect of the communication function, which can be addressed in CORM through the IFM. In the present state of our research, we have not fleshed out the IFM, however, we acknowledge the importance of communication as a standalone function by defining the DC sub-model. In CORM, the communication function can exhibit different behaviors, including mobility, which will be captured in the communication profiles created by the DC sub-model. To have a clear vision and perception of the IFM, we needed first to delineate the properties, capabilities and structures of the entities (NC, Ncomp and the network) composing the vertical network dimension, on which the IFM will operate. That is why we deferred the derivation of the IFM, expounding it, as future work, after depicting the different elements of the vertical network dimension represented by the NST and ACRF

3.5.3 Virtualization in CORM

Virtualization as a concept was invented more than 30 years ago to provide a way to share expensive computing resources [76]. It was then realized that virtualization brings about other important advantages, recommending virtualization mechanisms to be employed within the context of networks. Virtualization in computer networks allows multiple heterogenous network architectures to exist on a shared single substrate as virtual networks (VN) [73]. Each VN will be assigned a portion of the underlying physical resources, and allow the co-existence of multiple new custom protocols and architectures to be deployed independently without disruptions [77]. Accordingly, it has been argued that network virtualization provides flexibility, promotes diversity, and promises security and increased manageability [73]. At present virtualization as a mechanisms is regarded from two different perspectives. On the one hand, the architectural "purist" views virtualization as a tool for architecture evaluation and periodic deployment of successive, singular Internet architectures. On the other hand, "pluralist" seek to make virtualization an architectural attribute of the Internet arguing that virtualization mitigates the ossifying forces of the current Internet and enables the continual introduction of innovative network technologies [73]. However, we argue that as advantageous as it may seem, virtualization does not provide radical solutions to any of the current Internet limitations, nor alleviate the need for new paradigms in deriving novel network architectures. On the contrary, employing virtualization as an architectural concept may lead to several levels of indirections introducing further complexities, and accruing inefficiencies. Thus, we maintain that virtualization should not be regarded as an alternative to Future architectures, but rather a transitional and/or experimental tool. Before divulging into details to support our stand, we will provide a brief overview of virtualization, its advantages and current challenges.

Network virtualization is defined by the decoupling of the roles of the traditional Internet Service Provider (ISPs) into two independent entities: infrastructure providers (InPs), and service providers (SPs). InPs deploy and manage the underlying physical network resources. They offer their resources through programmable interfaces to different SPs. SPs, on the other hand, leases resources from one or more infrastructure providers to create virtual networks and deploy customized protocols and end-to-end services to end-users, or to other SPs [74, 78]. Therefore, virtualization can be regarded as classifying the network into Transport substratum, and Service substratum, where each substratum can evolve independently.

Several advantages are attributed to virtualization in literature. Primarily, Virtualization allows the evolution of communication technology, while largely reusing deployed infrastructure thus reducing the economic barrier for technical evolution [77]. Through virtualization network sharing is possible where different networking services are provided by different SPs on a common physical infrastructure. Virtualization provide flexibility at the SP level, where different architectures and customized protocols can be deployed within the boundaries of a VN [74, 78]. Isolating VNs improves fault-tolerance, security and privacy, since failures in one VN, due to attacks or

misconfigurations, are contained within the VN boundaries and do not affect other co-existing VNs [73]. More importantly, virtualization has the potential to break up the "deployment stalemate" observed today in the Internet by reducing the need to create broad consensus among the multitude of stake holders with diverging interests that make up today's Internet. By decoupling the infrastructure from the services and offering the ability to rent "slices" of the network infrastructure, virtualization can provide the opportunity to roll out new architectures, protocols, and services without going through the slow and difficult process of creating a consensus among stake holders [77]. However to realize virtualization benefits several challenges need to be addressed including the following list. For full analysis we refer the reader to [74, 78].

- Resource and Topology discovery: Providing a wide area end-to-end services SPs need to rely on different resources owned by different InPs, as well as interact with co-existing SPs to discover their topologies. On the other hand, to allocate resources to SPs, InPs must be able to discover both the topology and available resources that they manage and interact together to enable end-to-end VNs
- Resource Allocation and Management: Efficient allocation and scheduling of physical resources among multiple VN request is extremely important to maximize the number of co-existing VNs. However, several issues arise; first scalability and robustness increase in difficulty with increasing the number of supported VNs. Second the allocation of resources with constraints on virtual nodes and virtual links is known to be an NP-hard problem.
- Interoperability: Interoperation among InPs, InPs and SPs, and SPs is crucial to allow for network connectivity. In IP network interoperability has been solved by IP. However in VN environments, diversity of architectures and technologies may lead to network fragmentation.
- Signaling and Bootstrapping: Before creating a VN, an SP must already have network connectivity to the InPs in order to issue its requests. This introduces circularity where network connectivity is a prerequisite to itself. There must also be bootstrapping capabilities to allow SPs to customize the virtual nodes and virtual links allocated to them through appropriate interfaces.

From the above brief overview of virtualization, we argue that virtualization can be regarded as an operational environment aimed mainly for resource (network infrastructure) sharing. However, virtualization introduces additional problems to network realizations. Renowned for providing security, we argue that virtualization does not mitigate security attacks within a single VN, nor prevents attacks on the underlying InPs infrastructure. Furthermore, virtualization as a mechanisms, does not alleviate any of the reported Internet limitations or challenges, such as mobility handling, routing table convergence, separation of naming from addressing, etc … [73]. Accordingly, we assert that the strength of virtualization lies in being an experimental tool for evaluating new network architectures, as well as providing a transition environment hosting both old and new network realizations. However, virtualization can never eradicate the need for defining new network architectures or reference models such as CORM. On the contrary,

designing the virtualization environment according to CORM provides a systematic way in dealing with resource management and sharing, which have been the driving motive for the initiation of virtualization concepts in the first place. Resource management is a well realized concept in CORM by identifying resources as a main concern to be addressed along both network dimensions. Moreover, virtualization environment need a way to dictate polices. In CORM policy definition and management can be realized through the application concern. Finally, connectivity and interoperability among components in the virtualization environment can be achieved through the communication concern. For an example of CORM-based networks that can provide a better understanding of CORM concern-oriented design we refer the reader to chapter 5.

3.6 Summary

In this chapter we presented CORM design principles, approach and components. CORM conceives computer networks as a software-dependent complex system whose design need to be attempted in a concern-oriented bottom-up approach. CORM addresses the multidimensionality in networks detailing the network from three different perspective; configuration and structure of the basic building block, network composition, and network component interactions. The chapter concludes by highlighting CORM distinctive features, and sheds the light on how future requirements and paradigms can be addressed in CORM-based network realizations

Chapter 4
CORM: Casting the FBS Engineering Framework on Computer Network Design

"Function, structure, behavior, and relationships form the foundation of the knowledge that must be represented for specific design process to be able to operate on them."

John Gero [9]

Validating CORM has been a challenging task. In contemporary literature, validation of network architectural proposals have been, to a great extent, attempted through descriptive approaches including simulation, prototyping or mathematical modeling. This can be attributed to the fact that these proposals have been directed to address a particular problem or tailored to satisfy a specific set of requirements. Thus their validation focused to illustrate the behavioral modifications attained due to operational adjustments of current network realizations. These adjustments were accomplished through the manipulation of protocol interactions, performance parameters, or execution logic. However, in case of CORM, we aim to (1) validate CORM compliance to our proposed design principles, (2) validate CORM expressiveness, as a reference model, to delineate a computer network. For this purpose we sought to validate CORM (CORM basic abstractions and CORM components) using the Function-Behavior-Structure engineering framework.

The Function-Behavior-Structure framework (FBS) has been developed by the Australian design scientist John Gero and his colleagues [9]. It is credited to be applicable to any engineering discipline, for reasoning about and explaining the nature and process of design [78]. The advantage of the FBS lies in its explicitness in describing the design process. It clearly delineates and defines the design process in terms of eight steps [79]. In this chapter, we overview the FBS engineering framework and its accompanying design steps [9]. We aim to validate CORM and the derivation process of its basic abstractions using the FBS framework. We argue that CORM derivation process coincides with the FBS model. Furthermore, we present CORM as a realization of the FBS framework within the context of computer network design.

4.1 The Function-Behavior-Structure Engineering Framework

According to Gero, designers design by positing functions to be achieved and producing descriptions of artifacts capable of generating these functions. Thus the metagoal of design is to transform a set of function F (generally referred to as requirements) into a design description D. The purpose of D is to transform sufficient

information about the designed artifact so that it can be manufactured, fabricated, or constructed. However, translating requirements, expressed as F, to an artifact's design description D, is not a straightforward task. Thus further explorations are required to attain the transformation

$$F \rightarrow D.$$

On the other hand, the artifact to be produced using D will be realized in the physical world. Thus D needs to express the artifact in terms of composing elements and their relationships. The description of the artifact's elements and their relationships is referred to as the artifact's structure S. Thus the design description D of an artifact can be partially derived, by delineating the artifact's structure S, yielding the transformation

$$S \rightarrow D$$

Another way of deriving the structure of an artifact S is by using a *catalog lookup,* where we find a structure associated to a certain function providing the direct transformation

$$F \rightarrow S$$

However, this transformation occurs at the element level of an artifact, and it does not represent the general case of design. Nevertheless, both F and S can be defined within the context of the target artifact behavior. The behavior of structure Bs can be directly derived from structure yielding the transformation

$$S \rightarrow Bs$$

while F can be expressed in terms of expected behavior Be, where Be provides the syntax by which the semantics represented by F can be achieved

$$F \rightarrow Be$$

Comparing Bs to Be determines the appropriateness of S in fulfilling F yielding the following transformations

$$F \rightarrow Be \rightarrow S \rightarrow Bs \rightarrow D$$

Here, requirements specified through a set of functions F will be expressed in terms of expected behavior Be. This expected behavior Be is used in the selection and combination of structure S based on a knowledge of the behaviors Bs produced by S to be defined in D. However, when S is synthesized, its accompanying Bs might not be quite what was expected leading to several reformulations of all elements of the model (F, Be, S, Bs, and D). Figure 4.1 illustrates the design activity encompassing FBS elements and processes. Table 4.1 lists the eight processes of the FBS model.

4.2 Derivation Process for CORM Basic Abstraction Unit

Our derivation process of CORM NC coincides with the design steps proposed by the FBS framework. The inception point of our design activity was marked by our design principles. According to which, computer networks need to be designed as software-dependent complex adaptive systems that exhibit emergent behavior. CORM design principles formed our first set of requirements F_1 and expected behavior Be_1 as follows

F_1 = Complex adaptive systems (autonomous entities, complexity)

Be_1 = Emergent Behavior (adaptation, self-organization, evolution)

Shifting to the structure that can deliver F_1 and Be_1, we attempted a *catalog lookup* by exploring natural complex systems, and studying the structure (S), and the individual behavior of their components (Bs). Our research led us to a recent study on

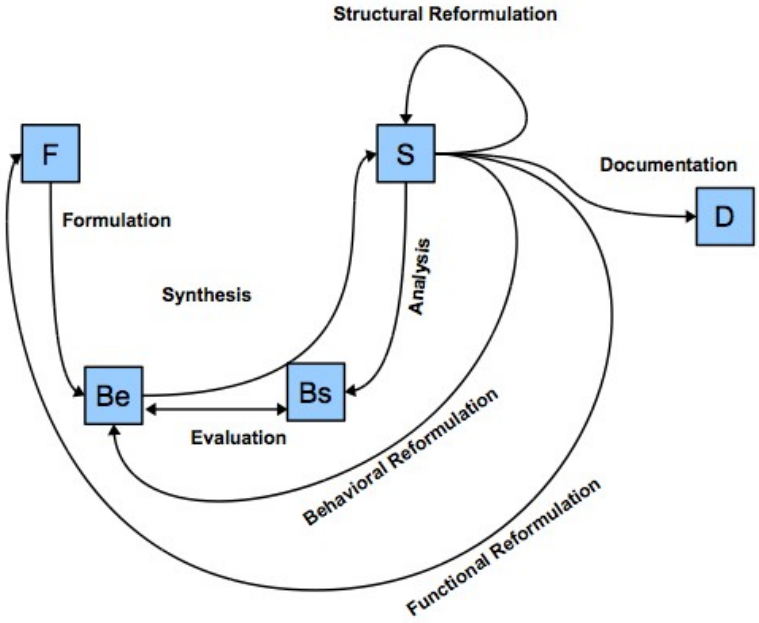

Figure 4.1: John Gero's Function-Behavior-Structure framework [9, 79]

Process	Transformation	Explanation
Formulation	F → Be	Transformation of the posited functions into behaviors that are expected to enable these functions.
Synthesis	Be → S	Transformation of these expected behaviors into a structure that is intended to exhibit these behaviors.
Analysis	S → Bs	Derivation of the actual behaviors of the structure
Evaluation	Bs ↔ Be	Comparison of the actual and expected behaviors
Documentation	S → D	Production of Design description
Structural Reformulation	S → S'	Changes in design state space in terms of structural variables or their values expressing structural evolution and refinement over time

Process	Transformation	Explanation
Behavioral Reformulation	S → Be'	Changes in the design state space in terms of behavioral variables or their values leading to changes/evolution of expected behavior
Functional Reformulation	S → F'	Changes in design state space in terms of functional variables or their values illustrating evolution of requirements

Table 4.1: The eight steps of the FBS model [9, 79, 80]

primordial bacterial colonies and their fascinating capabilities in self-engineering [13]. This point of our research marked our first functional reformulation, where we needed to formulate new requirements F_2 for designing a network cell that mimics the bacterium cell behavior Be_2. Accordingly, we synthesized the structure S_2 from Be_2 presenting the NC (detailed in chapter 3)

$$\mathbf{F_2} = \text{Self-engineering NC}$$

$$\mathbf{F_2 \rightarrow Be_2}$$

$\mathbf{Be_2}$ = Bacterium cell behavior (Self-monitoring, Self-regulation and decision-making, experience recording and memorization, communication and cooperation with peers and environment to perform high level goals of the whole colony)

$$\mathbf{Be_2 \rightarrow S_2}$$

$\mathbf{S_2}$ = IU, MU, RU, EU, CCS

However, F_2, Be_2, and S_2 needed further reformulation to detail network requirements. At this point we defined the network requirement specification that led to the derivation of the ACRF framework for network concerns yielding a new set of requirements F_3.

$$F_3 = \text{ACn, CCn, RCn, FCn}$$

F_3 was integrated with F_2, and superimposed over our previously defined Be_2, and S_2 to customize each towards computer network context in which they all will be realized. This led to the derivation of CORM NC, where F_3, F_2 and Be_2 are expressed within the NC structure S_2

CORM NC delineates the basic abstraction unit from which the network can be recursively built. However, at this point of our research, we still have not completely defined Bs for CORM NC, since this will involve defining performance variables and their range of values for the software code that will be running within each unit of the NC structure. Nevertheless, we accounted for Bs by defining the IFM that constitutes a major part of our future work.

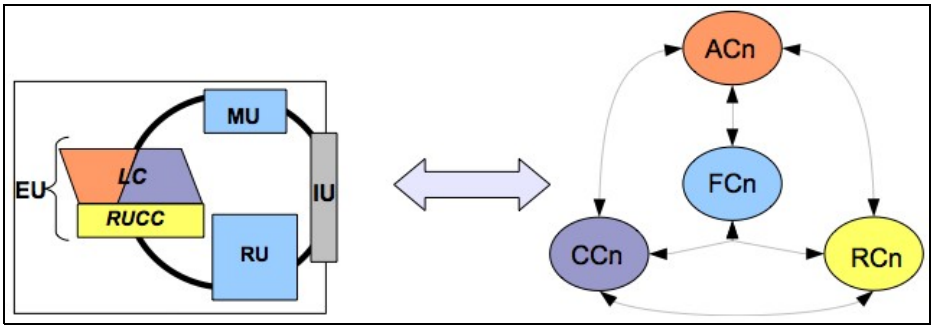

Figure 4.2: F_2 and Be_2 are expressed within CORM NC structure S_2

4.3 CORM: A Realization of the FBS Engineering Framework for Architecting Computer Networks

Computer networks, similar to any artifact need to be identified in terms of function, structure, and behavior. Most of the problems pointed out in present network realizations stem from the lack of behavioral and structural specifications. For example, routing function is defined as a means to discover routes within the network between two points. To delineate the routing function, the structure on which the search will be performed and the criteria for searching should be identified. The criteria for searching is what we refer to as behavior. In other words, the aspects the routing function should look for in a route, and according to which, the operation of the routing function can be understood and designed. On the other hand, aspects of the route are tied to the structure of the network itself. In networks, structure need to be addressed at different level of abstractions. Networks are made of devices and protocols that run on them. The relations of these devices is realized and defined according to the semantics of the protocols that define them for each other. Thus network structure need to be defined according to node logical as well as physical proximity. Therefore, we argue that any network model not only needs to delineate functions of the network, but more importantly needs to specify the structure on which these functions will operate and the parameters of operation.

Within the context of the previous discussion, we argue that CORM components account for network function structure and behavior. Network requirements are captured by the ACRF framework, while network structure is represented by the NST. Network behavioral aspects are to be identified by DR submodel, while network interactions are to be captured by DC submodel. Furthermore, adopting the cell paradigm for modeling network building blocks, emergent behavior capability is enabled throughout the network. However, we note that CORM, as a reference model, is at least three levels of abstraction away from any physical realization. Its purpose is to provide a common conceptual framework that can be used consistently across different implementations, and it is of particular use in modeling-specific solutions [63]. Accordingly architectures derived from CORM need to define Be and Bs within the context of their application.

4.4 Summary

In this chapter we overviewed the FBS engineering framework and its accompanying design processes. We presented the derivation of CORM NC, CORM's basic abstraction unit, in light of the FBS processes. It has been argued that FBS engineering framework is applicable to any engineering discipline. We conjecture that CORM is a realization of the FBS engineering framework within the context of computer network design.

Chapter 5

CellNet: An Architecture Derived From CORM

CORM, as a reference model, stand as a conceptual representation of a computer network expressing underlying design principles in terms of network abstract constructs, and their relationships. Several network architectures can be derived from CORM by further detailing requirements for the network to be realized. We conjecture that network realizations derived from CORM-based architectures can adapt to context changes and further evolve by inducing online modifications to the network logic executed by network components, allowing these components to operate according to learned optimal values. In this chapter, we present CellNet as an example for CORM-based network architecture. CellNet was our preliminary effort in materializing our perception of CORM [80] [81]. CellNet is simulated on the ns2 simulator [82] to illustrate the behavioral adaptation and modification of CellNet-compliant protocols in two case studies. The first demonstrates the adaptation capabilities of CellNet-compliant protocols in stabilizing TCP when operating in wireless ad hoc networks [83], [84], [85]. The second demonstrates CellNet-compliant protocols capabilities in defying the instability induced due the unintended consequences of a cross-layer design [52].

5.1 From CORM to CellNet

Three main aspects are involved when deriving an architecture: modularity, functionality, and interoperability. Modularity breaks the system into parts, functionality verifies that all required functions have been incorporated in the system using the parts defined, and interoperability ensures that the defined parts are capable of interacting and communicating [35]. Architectures derived from CORM will use CORM abstractions (NC, Ncomp, and ACRF) to define system parts, and their functionalities. Interoperability will be achieved through interface specifications for the defined basic abstractions.

CellNet was derived to have the minimal configuration required to perform general network functionalities, while ensuing loose coupling and high cohesion among the derived components. Furthermore, CellNet was required to operate within the Internet context. Accordingly, CellNet modularity and functionality is achieved by defining CellNet Ncomp to be composed of core-concerns NCs, namely resource oriented cell (ROC), communication oriented cell (COC), and application oriented cell (AOC). ROC, COC, and AOC have the structure of the generic CORM NC, but each is specialized to

perform a dedicated core-concern function executed by the logic in the EU, and further decomposed according to the ACRF framework. As for the FCn, we argue that it exits within each of the core concern cells by definition through the existence of the MU and the RU units. These two units collaborate to manage the crosscutting interests existing between cells in the Ncomp, as well as among Ncomps residing together on the same node/network/internetwork. Alternatively, the crosscutting concern management can be manifested by instantiating one or more federation oriented cell(s) (FOC) on the Ncomp, thus allowing for a more sophisticated interaction management. Figure 5.1 illustrates CellNet architecture and how it maps to the TCP/IP stack. Figure 5.2 illustrates a Ncomp composed of an AOC, COC, ROC and FOC. In CellNet, the number of cells instantiated to form a Ncomp is a design decision. However, operating within the context of the Internet, CellNet defines the minimum configuration for an edge Ncomp as one instantiation of each of the core-concerns NCs (AOC,COC, and ROC), while the minimum configuration for a core Ncomp is a single instantiation of a COC and an ROC.

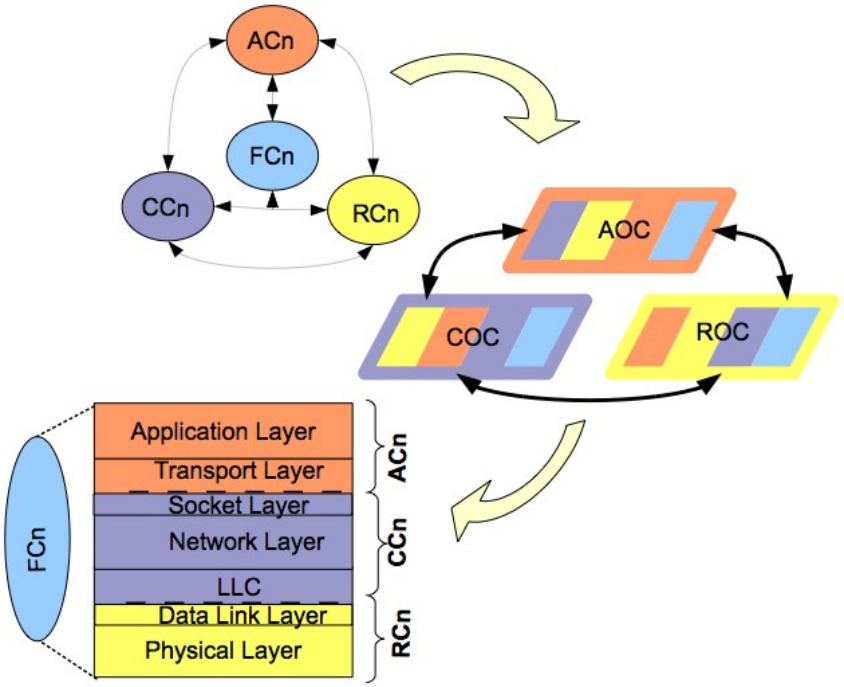

Figure 5.1: CellNet Basic Architecture Mapped to TCP/IP Stack

For network composition, CellNet imports CORM NST, expanding its definition to include AOC, COC, ROC and FOC. In addition, CellNet NST captures the notion of the Internet gateway, which is a common construct connecting TCP/IP networks, by introducing the internetwork communication substratum. Accordingly, the CellNet NST, using EBNF, is defined as follows:

CellNet NST EBNF Formal Definition

Notations:

Trailing * means repeat 0 or more times.

Trailing + means repeat 1 or more times

Abbreviations:

- AOC = Application Oriented Cell
- COC = Communication Oriented Cell
- ROC = Resource Oriented Cell
- FOC = Federation Oriented Cell
- CCS = Cell Communication Substratum
- Ncomp = Network component
- NCS = Network Communication Substratum
- Net = Network
- ICS = Inter-network Communication Substrate
- INet = Inter-network

Network Definition:

1. AOC = EU RU MU IU CCS
2. COC = EU RU MU IU CCS
3. ROC = EU RU MU IU CCS
4. FOC = EU RU MU IU CCS
5. Ncomp = AOC* COC+ ROC+ FOC* CCS
6. Net = Ncomp (NCS Ncomp)+
7. ICS = (NCS Ncomp)* NCS
8. INet = Net (ICS Net)+

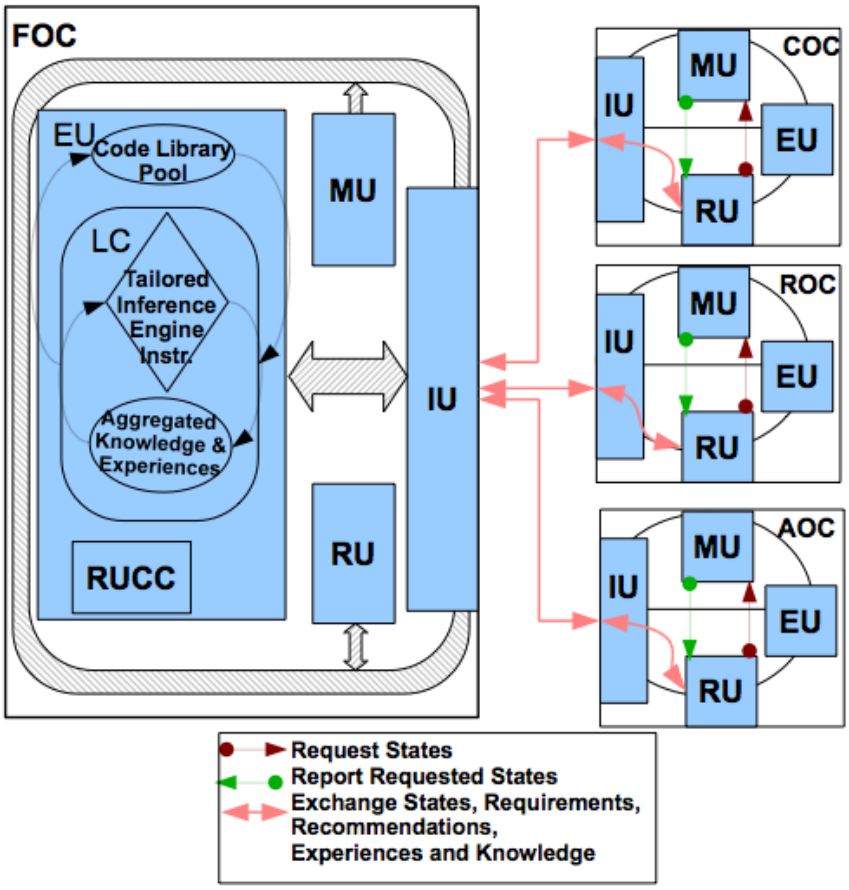

Figure 5.2: CellNet Ncomp

5.2 CellNet Protocols

Designing CellNet protocols needs a shift in the protocol engineering paradigm to acknowledge both the NC structure and the ACRF framework. As an illustration, we present the following example for a configuration instance of CellNet protocols performing a file transfer. We note that in our example we aim to present a conjecture of how protocols might be engineered, rather than delving into implementation details

Assume an end user initiated a file transfer (FT) between two nodes that are residing in a network that is CellNet enabled. According to CellNet architecture, to perform a file transfer, an edge Ncomp will be instantiated on both the source and destination nodes, while intermediate Ncomps will be active on the nodes along the path. In this example, the term end user may refer to a human user, the OS, or any another application. The operational parameters we refer to in this example are a subset of the actual parameters that need to be involved in the file transfer operation, and their use in the example are just for the purpose of illustrating our perception. The responsibilities of each of the initiated CellNet NCs will be as follows:

1. FT-AOC:

Source and destination FT-AOC will be, respectively, the source and the sink of the packet flow pertaining to the file. We envision the functional classification within the FT-AOC according to ACRF will be as follows:

- ACn: The ACn code will be executed in the AOC EU LC. It will receive from the end user an application profile that depicts the file transfer instantiation. This will include the level of reliability required for the file transfer. The level of reliability will reflect how the tailored transport service will set packet formats, decide on retransmission policy, handle acknowledgements and corrupted packets.
- CCn: The CCn code will be executed in the AOC EU LC. It will receive from the end user a communication profile that indicates the destination of file transfer. Other parameters might be an upper threshold on the transfer duration and the priority of the transfer.
- RCn: The code for the RCn will be executed in the AOC RUCC. It is responsible for estimating and managing the resources assigned by the OS to the instantiated AOC
- FCn: The execution of the FCn code is distributed between the MU and the RU. MU will be responsible for monitoring all performance parameters. These parameters could be specific to the file transfer, or general parameters reflecting the operation of the EU. These monitoring parameters are specified once the NCs get specialized and are subject to adjustments/amendments if required. The RU will constantly check the performance of the file transfer operation in specific, and the EU operation in general, by comparing the values of the monitored parameters to thresholds values. Part of the threshold values are fed into the RU when the NC is specialized and part are to be negotiated with other NCs as will be explained shortly. The RU interferes to adjust either the file transfer operation, or the EU operation, in case the monitored values fall below the indicated thresholds. Furthermore, the RU will decide on any optimizations required for improving performance, resolving any cross interests that might rise among the core concerns within the AOC or among the AOC and other NCs. In this presented case of file transfer, the RU will create two profiles based on the user profiles that were fed to the AOC. The first profile will be fed to the COC and the other will be fed to the ROC. These profiles will include the parameters, whose values need to be decided upon after consulting either or both NCs. For example, the ROC profile could include the packet formats, and FEC technique. These two parameters represent a cross interest between the AOC-ACn and the ROC-ACn, since both are affected by the transmission technology used and the type of the file to be transferred. Therefore, their values need to be decided upon in tandem. For the COC profile, this could include the level of transport service reliability and flow priority. Both parameters will indicate the route quality to be discovered and the handling of the flow transmission.

2. FT-COC:

FT-COC is responsible for setting up the communication path from source to destination, and managing the transfer of packets across the network. The FT-COC functional classification according to ACRF is as follows:

- ACn: The code for ACn will be executed in the COC EU LC and is responsible for setting the routing protocol policies, partially based on the COC profile received form the AOC, which indicates the quality of the route that need to be set up for a given destination and the priority of the flow. ACn will also decide on internal COC policies such as how the routing table should be built (i.e., what are the parameters for choosing one route over another), how the routes will be maintained, when a route will be purged, how to manage the neighbor table, when to consider a route to a neighbor to be down, and so forth.
- CCn : The code for CCn will be executed in the COC EU LC and is responsible of defining and discovering routes based on the ACn requirements and policies. ACn requirements and polices will assist the CCn to decide on the appropriate Routing protocol (RT) to be instantiated. The choice of the RT could depend on the RTs available on the node on which the Ncomp is residing, and previously learned knowledge about each RT's operation. On the other hand, a default RT could be chosen and its operation will be adapted according to the sensed context and the fed in requirements and policies. Route definition and discovery mainly depends on link definition, which is part of the CCn responsibility in CellNet. Thus introducing a cross interest between COC and ROC in case of wireless links to be handled through the FCns of both NCs as will be mentioned shortly. The CCn will also resolve routes, manage the sending and receiving of route requests and replies, communicate with neighbors, forward received packets and so forth.
- RCn: The code for RCn will be executed in the COC EU RUCC, and it is responsible for estimating and managing the resources assigned by the OS to the instantiated COC.
- FCn: MU will be responsible for monitoring all performance parameters, whether pertaining to the specific function assigned to the COC (communication through route definition, discovery, and maintenance), or the operation of the EU in general. Monitoring parameters are to be specified once the NCs get specialized and are subject to adjustments/amendments if required. The RU will constantly check the performance of the communication related functions in specific, and the EU operation in general, by comparing the values of the monitored parameters to thresholds values defined in the RU. The RU interferes to adjust either the communication related functions, or the EU operation, if the monitored values fall below the indicated thresholds. Furthermore, the RU will decide on any optimizations required for improving performance, resolving any cross interests that might rise among the core concerns within COC, or among the COC and other NCs as in the case of operating over wireless links (e.g., memory usage and route purging interval, power level and hello message frequency, etc).

In case of wireless links, link definition and quality is greatly affected by the transmission power used. However, power transmission level is controlled by the ROC. Therefore the RU of the COC and ROC need to negotiate for deciding on the optimal value that enhances the operation of both. Such optimizations will depend on the information and recommendations stored in the RU knowledge base, which could have been fed on specialization or inferred through learning.

3. FT-ROC:

In present implementation the network system is integrated within the OS and does not have direct control on resources, such as the device memory, CPU cycles or overall device power. However, it has full control on network interfaces. We argue that such setup actually hinders the capabilities and evolution of the network system, as well as resource management, since effective management of resources needs a holistic approach. Being an integrated part of the OS, the network system resource management is confined to the resources assigned to it be the OS to operate the network interfaces, and these are the resources that will be managed and controlled by the ROC.

- ACn: The code of the ACn will be executed in the ROC EU LC. It represents the logic that sets the policies for managing active interfaces on a device. ACn policies and choices will depend on the capabilities of the underlying transmission technology used as well as the profile received from other NCs (AOC, COC). As mentioned before, the packetization and FEC techniques, for example, need to be collaboratively chosen by the ROC and AOC.
- CCn: CCn: The code for the CCn will be executed in the ROC EU LC. It controls the actual transmission and reception functions of the NIC including tweaking the power level at which each transmission and reception operation will be performed, managing collisions, capture, adjusting power to SNR detected, etc. Again, CCn choices will be affected by the COC requirements as previously mentioned.
- RCn: The code for RCn will be executed in the ROC EU RUCC and it is responsible for estimating as well as managing the resources assigned by the OS to the ROC
- FCn: FCn on ROC will have similar operations regarding monitoring and regulation as mentioned in 1 and 2 above. What we want to stress again that through the negotiation between ROC RU and AOC and COC RUs values for common interest parameters can be set, modified or redefined. These values will then be used by the code executed in the EUs (whether for the ACn or CCn) for all NCs

Engineering protocols to conform to CellNet call for new methodologies and techniques beyond the scope of this research work. Yet, we argue that CellNet capabilities in terms of adaptation and evolution can be induced into present protocol implementations by embedding legacy protocols into specialized CellNet NCs according to CellNet ACRF concern framework, while maintaining legacy protocol APIs in addition to the NCs' APIs. Such arrangements pave the road for practical realization of CellNet-

aware networks. Such embedments have the effect of augmenting existing legacy protocols with self-monitoring functions and regulatory logic, yielding CellNet-compliant protocols. CellNet-compliant protocols override the incognizant execution of legacy protocols by controlling operational parameters and tweaking performance to operational context. As for inter cell communications, part of the space allocated for the application payload in legacy packets can be used for inter-Ncomp communication. This layout was used to simulate CellNet protocols in the ns2 simulator [82] providing a pragmatic way for CellNet evaluation.

5.3 CellNet Evaluation

The main focus of this evaluation is to assess the adaptation and evolution capabilities attained by embedding legacy protocols into CellNet specialized NCs. Our evaluation was conducted through simulations since our main concern was to demonstrate the change in the protocol model behavior as a proof of concept, rather than address a specific protocol implementation. Two case studies will be presented. The first illustrates the adaptation of CellNet-compliant transport protocol operation to network capacity in ad hoc networks, by raising the transport protocol awareness to the level of MAC contention. The second demonstrates the behavioral modification of CellNet-compliant protocols and consequent evolution through learning. These modifications counteracts the TCP flow instability manifested as end-to-end throughput oscillations resulting from unintended consequences of a cross-layer design presented in [52]. Both case studies were simulated in an ad hoc network using the ns2 simulator [82]. The reason for choosing ad hoc network as the setting for our simulations are two fold. First, proposals introducing adaptation to Internet protocols have been, to a great extent, motivated by the incognizant legacy protocol operation over wireless links [44], [51]. Second, the adverse effects of wireless links on legacy protocol operation are exacerbated in ad hoc network environments [83], [84], [85].

5.3.1 ns2 Network Simulator Overview

ns2 is a discrete event simulator targeted at networking research [82]. ns2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks. ns began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995, ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently, ns development is supported through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. ns has always included substantial contributions from other researchers, including wireless code from the UCB Daedalus and CMU Monarch projects and Sun Microsystems [82].

Our choice for ns2 as our network simulator was driven by two factors: first, our second case study is based on the cross-layer algorithm developed in [52], in which the authors used ns2 for their simulation to show the adverse effect of cross-layer designs. Since we will be adopting their algorithm in our baseline simulation, we were inclined to use the same simulator to obtain comparable results. Second, it has been mentioned in [86] that ns2 is the most commonly used simulator for MANET research. By adopting ns2 as our simulator, we can, as future work, experiment with different other scenarios presented in literature to further acquire a better understanding of the effect of tuning different protocol parameters to protocol operations.

In our simulations, we used ns2.31over MAC OS X version 10.5.8. ns2 comes with a test suite composed of validation scripts, which we executed without errors. To simulate wireless nodes, we used the CMU extension, which includes a set of mobile ad hoc network routing protocols and an implementation of BSD's ARP protocol, as well as, an 802.11 MAC layer and three propagation models [87]. The MAC layer implements the complete IEEE 802.11 standard MAC protocol DCF to accurately model the contention of nodes for the wireless media. All nodes communicate with half duplex wireless radios that approximate the DSSS radio interface (Lucent WaveLan direct-sequence spread-spectrum) with a bandwidth of 2Mb/s and a nominal transmission radius of 250m [87]. It was mentioned in [88] that the results obtained form ns2 wireless simulations in ad hoc network settings match fairly closely results obtained from actual testbeds, suggesting that the simulation of the 802.11 DCF (CMU extension) do not contain major errors.

In our simulations, we used the one-way TCP implementation, which reflects a simplified underlying model of one-way data transfer of fixed size packets. This model attempts to capture the essence of the TCP congestion and error control behaviors, but it is not intended to be a faithful replica of any real-world TCP implementation [87]. Thus, the model does not contain a dynamic window advertisement, or a segment number and ACK number computations entirely in packet units, and there is no SYN/FIN connection establishment. However, the TCP model implementation performs congestion control and round-trip time estimation similar to the version of TCP released with the 4.3BSD “Tahoe” UNIX system release from UC Berkeley. Therefore, it is called in the simulator Tahoe TCP. For more information on TCP implementation details, we refer the reader to the ns documentation [87]. Using model implementation rather than real-world implementation allowed us to concentrate on the general TCP protocol behavior rather than confining our results to specific TCP variants. We argue that the results obtained from the TCP general model can be applied to all variants, since the model captures the main essence of the protocol’s behavior.

5.3.2 General Simulation Design

In our evaluations, we are interested in the steady state performance. Therefore, we ignore transient states by deleting initial values recorded for performance parameters

detailed in each case. For each simulation scenario, we conduct 10 independent replications seeding the random number generator (RNG) as indicated in [89] to ensure the absence of correlations among the generated random numbers. Baseline scenarios are executed using the protocols supplied by ns2, which simulate the behavior of legacy protocols. The same scenarios are then executed after embedding the legacy protocols into CellNet specialized NCs. These will be hereafter referred to as CellNet scenarios. For each set of replications, the confidence interval at 95% is calculated using t-distribution.

5.3.3 Case Study I:Achieving Flow Stability Through Transport Protocol Adaptation

This case study casts network stability onto transport protocol congestion control mechanisms. As mentioned in [90],

> network stability is frequently associated with rate fluctuations or variance. Rate variations can result in fluctuations in router queue size and therefore queue overflows. These queue overflows can cause loss of synchronizations across coexisting flows and periodic under-utilization of link capacity, both of which are considered to be general signs of network instability.

Thus, one way to achieve stability is to avoid network congestion by correctly estimating network capacity and adjusting the flow rate accordingly. For legacy protocols, the flow rate is either controlled by the TCP congestion window or by the application running over UDP. In the case of TCP, a congestion window is a sliding window on the TCP buffer that stores the application data to be transferred over the network. The congestion window starts at a size of one segment and probes the network capacity by increasing the window size (multiplicatively during slow start and additively during congestion avoidance) with each acknowledged segment until the first loss is realized, which results in reducing the congestion window size (window size reduction will depend of the TCP variant; in Tahoe it goes back to one segment). Optimally, the congestion window size should be proportional to the network capacity, derived as the bandwidth-delay product of the links along the path. However, TCP cannot estimate the optimal size for its congestion window, but has to wait for losses to realize that the window has exceeded the optimal size. Losses at the sender side can be realized either proactively, by receiving duplicate acknowledgement from the receiver for the last received segment, or reactively, though a round-trip time out (RTO). To calculate the latter, TCP estimates the end-to-end round-trip time (RTT) taken by a segment and its acknowledgement, and if an acknowledgement for a sent segment fails to be received within the estimated RTT then an RTO event has occurred. In this respect, we note that TCP's congestion control is reactive; packets need to be lost before TCP can realize congestion, as well as incognizant; congestion window size is aggressively reduced regardless of the reasons imputed to packet drops. For UDP, congestion control mechanism is not provided leaving it to be solely implemented by the end running applications, which attempt to estimate

the RTT to adjust the flow rate to the available network capacity. Accordingly, RTT estimation is a determinant factor in adjusting flow rates to network capacity, and hence achieving network flow stability.

Network instability due to flow-rate fluctuations is exacerbated in ad hoc networks due to hop-by-hop MAC acknowledgements at wireless links [88]. Hop-by-hop MAC acknowledgements have a direct effect of artificially increasing the RTT, which is translated by TCP and/or end applications over UDP as an increase in network capacity. A direct consequent of a large RTT would be an increase in the load induced into the network. However, the increased load brings no gains in throughput, but rather results in more packets being queued at intermediate nodes along the path, causing higher contention among adjacent nodes [83]. Eventually, packets will be dropped due to repeated collisions resulting in loss of synchronizations across coexisting flows and periodic under-utilization of link capacity – signs of network instability as previously noted. Since TCP is the prevalent transport protocol used for numerous IP-network applications, TCP instability over ad hoc networks has been the focus of several research work [83], [84], [85]. It was noted in [84] that the main cause of packet drops in wireless networks is due to contention rather than congestion. Yet, these contention drops are not tuned to the TCP operation. A realization that led the authors in [84] to propose two simple link-layer designs to make contention drops beneficial to TCP flows. The first, which they referred to as link-layer random early dropping (LRED), controls the TCP window size by tuning the link-layer dropping probability according to perceived channel contention. The second, called Adaptive pacing, is enabled by LRED when a node finds that the average transmission count exceeds a threshold. The basic idea is to make a node back off an addition packet transmission time, in addition to its current deferral period, when necessary. In [85], the authors examine the TCP instability problem from the routing viewpoint, arguing that a lack of coordination among the TCP and routing layer leads TCP to take the available network resources as much as possible leaving, little that might otherwise be used for routing maintenance affecting the end-to-end connection and eventually hurting the TCP performance. The authors proposed a fractional window increment scheme, called FeW, that allows a fractional increment of the TCP congestion window thus reducing the growth rate for the congestion window, and allowing gentle probing to the network so as not to cause instability. In [83], the TCP instability problem is classified into intra flow and inter flow instability, where the former instability is caused by the interaction of nodes belonging to the same TCP flow, and the latter occurs due to interactions among nodes pertaining to different flows at the MAC layer. For the intra flow, the authors proposed a cross-layer design to raise the awareness of TCP to the contention level at the MAC layer. Each node, in the path from sender to receiver, calculates the time required to gain the channel to transmit a TCP packet and adds it to the value stored at a new field in the packet header. This new field had been added to packet headers to hold the aggregated time values for packet transmission. The receiver then transmits the aggregated time value to the sender in the ACK packets. As for the

inter flow instability, they proposed a new scheme for the MAC back-off algorithm to tackle the channel access fairness among contending nodes.

Presented proposals sought different approaches to handle the TCP instability in ad hoc networks. LRED and adaptive pacing in [84] focused on the link layer, FeW in [85] applied changes to the TCP protocol, and [83] presented a cross-layer design incorporating TCP at the transport layer and MAC at the link layer. In all cases, raising the awareness of protocols working at different layers to their operational environment was essential. In spite of achieved improvements, we argue that the presented mechanisms all target a single problem (TCP instability), and provided a single point solution (adapting TCP traffic load to network capacity). Expanding the instability problem to all protocols operating over ad hoc networks, we argue that to attain stability, any protocol implementing transport functionality needs to be aware of the contention level present at the wireless links – thus correctly estimating the RTT and accordingly adapting its flow to the available network capacity. In other words, our independent variable will be the load offered to the network by the transport function measured as the size of congestion window for reliable transmission protocol, and as the rate of data transfer for streaming and real-time protocols. While our dependent variable will be the MAC contention level, measured as suggested in [83], as the time required by the node to acquire the channel. To maintain stability, the MAC contention level need to be monitored along the path and load offered need to be adjusted accordingly.

5.3.3.1 Case Study I Simulation Design

The purpose of this simulation is to illustrate the monitoring and adaptation capabilities of CellNet-specialized NCs (CellNet scenario) to attain stability in contrast to the incognizant operation of legacy protocols (baseline scenario). Simulation scenarios will incorporate both TCP and UDP traffic over a chain topology composed of five nodes, where each node can only access its direct neighbor. Chain topologies have been reported as challenging due to the interference induced by the successive transmission of packets pertaining even to a single flow, thus increasing MAC contention level and probability of packet drops [84, 85]. TCP traffic will be induced using an FTP application, while UDP traffic will be induced using a constant bit rate (CBR) application. In CellNet scenarios, the principal monitored parameter is the MAC contention level realized at the wireless links. MAC contention is measured on each Ncomp and along the path from the source to destination. Each Ncomp will be aware of the contention level on its direct link towards its next hop neighbors as well as the contention level realized along the paths towards both the source and destination. For TCP protocol, the application-oriented cell (AOC-TCP) encapsulating the TCP protocol on the source Ncomp will use the reported contention level along the path to derive the optimal size for the congestion window. For UDP protocol, the application-oriented cell (AOC-CBR) encapsulating the CBR application on the source will incorporate the contention level in the packet transfer rate

calculation.

The total simulation time is 1003 sec and is composed of three stages each running for 300 sec and two intervals. In the first stage, a single TCP flow is running over the chain topology from source (node 0) to destination (node 4) for 300 sec. The second stage introduces another TCP flow running from the same source to the same destination for the next 300 sec. After the second TCP flow stops, we introduce a wait interval (Interval 1) for 100 sec, allowing the first TCP flow to stabilize. The third stage introduces a CBR flow over UDP for another 200 sec, after which we allow the first TCP flow to stabilize again for 100 sec (Interval 2) before the simulation ends. The last 3 sec are introduced to allow for any in-flight packets to reach the sink. The CBR flow rate has been adapted separately to the chain topology in a trial simulation to ensure that any packet drops realized are due to the flow interactions not due to the overload of CBR traffic. The simulation parameters are summarized in Table 5.1.

Simulation Parameters	
No. of Nodes	5 nodes arranged in chain topology
Area	1200 x 1200 m^2
Total Simulation Duration ➢ Stage 1 = Stage 2 ➢ Stabilization Interval 1 ➢ Stage 3 ➢ Stabilization Interval 2	1003 sec 300 sec 100 sec 200 sec 100 sec
Transmission range	250 m
Propagation model	Two ray ground
Wireless Link Bandwidth	2Mb/sec
MAC protocol	CMU extension
Routing protocol	AODV
Transport protocols	Tahoe TCP, UDP
Application inducing packet flows	CBR over UDP, FTP over TCP
No. of flows	3 (2 FTP and 1 UDP)
CBR packet flow rate	110 Kbits/sec

Table 5.1: Case Study I Simulation Parameters

5.3.3.2 Case Study I Simulation Results and Analysis

The main focus of this evaluation is to compare the performance of CellNet protocols to legacy protocols, and assess the gains of CellNet protocol adaptation to

MAC contention level. In our evaluation we care for steady state performance; thus, we exclude the values recorded for the first 5 secs during which the TCP protocol in baseline simulations will be operating in slow start phase, and in CellNet simulations the NCs will be gathering monitoring data. Our performance metrics, which are derived from the TMRG RFC 5166 “Metrics for the Evaluation of Congestion Control Mechanisms” [90], are throughput, packet losses, and protocol stability. For throughput and packet losses, we record the congestion window size in packets (TCP packets are 1000 bytes long), total number of duplicate acknowledgements, and sink throughput in Kbits/sec for TCP. As for UDP, we record throughput, and packet losses in Kbits/sec recorded at the receiver. For stability, we calculate the coefficient of variation (CoV) for TCP and UDP throughput received at the receiver. For these four metrics we report the grand average (Av), the standard deviation (Std) and 95% confidence interval (CI) calculated for each simulation phase (Stage 1, Stage 2, Interval 1, Stage 3, and Interval 2) over all replications in Table 5.2, Table 5.3, Table 5.4, Table 5.5, Table 5.6, and Table 5.7.

For a single flow, CellNet TCP throughput is higher than legacy TCP as indicated in Table 5.2, in spite of a smaller congestion window size. Moreover, the number of packets dropped for CellNet TCP is below that dropped by Tahoe TCP, indicating better network capacity estimation as well as better resource usage.

Performance Parameters		Tahoe TCP	CellNet TCP
Congestion Window Size (Packets)	Av	5.57	3.06
	Std	0.11	0.11
	CI	[5.65, 5.5]	[3.14, 2.98]
Sink Throughput (Kbits/sec)	Av	136.09	156.23
	Std	1.62	0.54
	CI	[137.25, 134.94]	[156.62, 155.85]
Duplicate Acknowledgements (ACK Packets)	Av	381	19
	Std	38	5
	CI	[408, 345]	[22, 15]
CoV	Av	0.48	0.11
	Std	0.02	0.01
	CI	[0.49, 0.46]	[0.12, 0.10]

Table 5.2: Stage 1 Performance Parameters

For Stage 2, as seen in Table 5.3, both CellNet and Tahoe TCP maintain fairness among the two flows in the sense that the average throughput of both flows, and the throughput CI is within close values. However, CellNet TCP still performs better with

Performance Parameters		Tahoe TCP1	CellNet TCP1	Tahoe TCP2	CellNet TCP2
Congestion Window Size (Packets)	Av	3.83	2.37	3.92	2.25
	Std	0.44	0.06	0.48	0.05
	CI	[4.15, 3.51]	[2.41, 2.33]	[4.261, 3.58]	[2.29, 2.21]
Sink Throughput (Kbits/sec)	Av	66.39	73.28	67.4	75.05
	Std	12.59	1.33	12.84	1.76
	CI	[75.39, 57.38]	[74.23, 72.32]	[76.58, 58.21]	[76.31, 73.79]
Duplicate Acknowled-gements (ACK Packets)	Av	376	95	358	92
	Std	66	21	137	13
	CI	[424, 329]	[110, 81]	[456, 260]	[102, 83]
CoV	Av	0.98	0.44	0.96	0.43
	Std	0.16	0.02	0.15	0.02
	CI	[1.1, 0.87]	[0.46, 0.42]	[1.07, 0.85]	[0.45, 0.42]

Table 5.3: Stage 2 Performance Parameters

Performance Parameters		Tahoe TCP	CellNet TCP
Congestion Window Size (Packets)	Av	5.5	2.96
	Std	0.22	0.18
	CI	[5.66, 5.34]	[3.08, 2.83]
Sink Throughput (Kbits/sec)	Av	138.49	156.39
	Std	2.22	0.7
	CI	[140.08, 136.9]	[156.88, 155.9]
Duplicate Acknowledgements (ACK Packets)	Av	120	5
	Std	0.02	1.93
	CI	[135, 104.7]	[6, 3]
CoV	Av	0.45	0.1
	Std	0.02	0.2
	CI	[0.47, 44]	[0.12, 0.09]

Table 5.4: Interval 1 Performance Parameters

less packet losses indicated by the number of duplicate acknowledgements recorded.

In Interval 1, we are testing TCP response to changes in network load. Results in Table 5.4 show that both TCP protocols regained their previous values recorded in Stage 1

except for duplicate acknowledgements indicating less packet drops. In spite of the lower number of duplicate acknowledgements recorded for TCP Tahoe, CellNet TCP has recorded far fewer duplicate acknowledgements and higher throughput, indicating better adaptation to network load.

Performance Parameters		Tahoe TCP	CellNet TCP
Congestion Window Size (Packets)	Av	2.21	2.35
	Std	0.21	0.8
	CI	[2.36, 2.06]	[2.41, 2.3]
Sink Throughput (Kbits/sec)	Av	8.04	69.11
	Std	1.53	4.09
	CI	[9.14, 6.95]	[72.04, 66.19]
Duplicate Acknowledgements (ACK Packets)	Av	63.6	71.2
	Std	45.34	11.24
	CI	[96.03, 31.17]	[79.24, 63.16]
CoV	Av	2.57	0.45
	Std	0.3	0.07
	CI	[2.79, 2.36]	[0.5, 0.4]

Table 5.5: Stage 3 TCP Performance Parameters

In stage 3, CellNet transport protocols outperform legacy transport protocols. As shown in Tables 5.5 and 5.6, CellNet TCP and UDP evenly shared the network capacity, achieving similar throughput as indicated by the throughput average and confidence interval. For baseline simulation, the UDP protocol acquires most of the bandwidth, leaving a very narrow strand for TCP (8 Kbits/sec for TCP average throughput versus 113.66 Kbits/sec for UDP average throughput). Furthermore, for lost packets, CellNet UDP recorded less than legacy UDP. However, for TCP the number of duplicate acknowledgements for CellNet TCP are more than Tahoe TCP – but taking into consideration that the throughput of the latter is far below the former, this difference is expected. It can be argued that applications running over UDP are most properly time stringent, therefore a larger portion of the bandwidth should be given to UDP. We argue that, in the present simulation both UDP and TCP flows have the same priority, and therefore should get equal shares of the network capacity. However, if such a setting is required, it can be easily induced by raising the awareness of the CellNet NCs to the different priorities of the flows administered to the network, and different shares will be assigned accordingly.

Similar to Stage 1 and Interval 1, in Interval 2, the TCP flow is the single flow active in the network until the end of the simulation. We note that in the three stages, CellNet TCP outperforms Tahoe TCP when comparing the performance parameters. Furthermore, the variation in the CellNet TCP performance parameters during these three stages is minimal when compared to Tahoe TCP.

Performance Parameters		CBR over UDP	CBR over CellNet UDP
Throughput (Kbits/sec)	Av	113.66	69.25
	Std	1.25	2.5
	CI	[114.55, 112.76]	[71.0, 67.5]
Packet Loss (Kbits/sec)	Av	9.23	5
	Std	2.01	0.94
	CI	[10.67, 7.79]	[5.66, 4.32]
CoV	Av	0.15	0.27
	Std	0.02	0.3
	CI	[0.16, 0.13]	[0.29, 0.25]

Table 5.6: Stage 3 UDP Performance Parameters

Performance Parameters		Tahoe TCP	CellNet TCP
Congestion Window Size (Packets)	Av	5.35	2.94
	Std	0.37	0.26
	CI	[5.61, 509]	[3.13, 2.75]
Sink Throughput (Kbits/sec)	Av	124	156.17
	Std	8.17	1.32
	CI	[129.84, 118.16]	[157.12, 155.23]
Duplicate Acknowledgements (ACK Packets)	Av	124	4
	Std	23.46	4.64
	CI	[140, 107]	[8, 2]
CoV	Av	0.58	0.11
	Std	0.08	0.03
	CI	[0.63, 0.52]	[0.13, 0.9]

Figure 5.7: Interval 2 Performance Parameters

Finally, for flow throughput variance, CellNet TCP throughput recorded lower CoV in all simulation phases when compared to Tahoe TCP throughput. However, for UDP flows, CellNet UDP throughput recorded a slightly higher CoV in stage 3 thanlegacy UDP. This can be attributed to the fact that the flow rate for the CBR application running over legacy UDP has been previously adjusted to the network capacity. Therefore, in Stage 3, being oblivious to any accompanying flows, legacy UDP flow takes over most of the available bandwidth thus achieving very low rate variation indicated by the low CoV average and confidence interval. On the other hand, due to monitoring and regulation in CellNet, CellNet UDP is aware of the accompanying TCP flow, and therefore the network bandwidth is shared between the two flows – resulting in slightly higher throughput fluctuation for CellNet UDP when compared to legacy UDP.

To conclude, the presented results demonstrate the adaptation capability of CellNet protocols to better fit their operational context, thus sustaining network stability and bringing about gains in terms of throughput and bandwidth utilization.

5.3.4 Case Study II: Achieving Stability Through Protocol Evolution

The presented case study highlights the need for protocol evolution to address feature interactions and cross interests that might occur among operating protocols. Within the context of network protocols, we define protocol evolution as a higher form of intelligent protocol adaptation in response to context changes by accounting for previously recorded knowledge from past experiences. Accordingly, in the presented case study, CellNet protocols will learn optimal values for operational parameters within the given context at a global scope. They will then tweak their subsequent operations accordingly. Such behavior contrasts with cross layer designs that apply unadaptable optimizations at a local scope without considering the consequences at a global scope. The case study deals with two main problems. The first is cross interests involved in power management, and the second is legacy protocols' inevolvable behavior.

When investigating the power-management problem, one finds that it is a typical case of a cross-concern problem that needs to be addressed from different perspectives. Referring to ACRF framework, four perspectives are identified: a resource perspective; a communication perspective; a network utilization perspective; and an interaction perspective. From a resource perspective, power is a valuable resource that needs to be efficiently managed while maintaining a balance between optimization and performance tradeoffs. This balance needs to be identified and continuously redefined by the needs of the utilization perspective representing the primary purpose of the device's operation. From a communication perspective, in wireless environments, power defines the links in terms of characteristics and range, which is the main entity out of which paths are created allowing networking among dispersed nodes. Thus, to manage this intersection of interests in power, a cross-concern perspective needs to be in place. However, legacy

protocols have been developed to minimize, if not completely exclude, interactions and thus fail to handle problems requiring cross interest management. Furthermore, due to their incognizant operation, legacy protocols fail to adapt and consequently evolve.

The baseline simulation of the presented case study was deliberately contrived in [52] to show the pitfalls of cross-layer adaptations that apply unalterable optimizations at a local scope without considering the effect at a global scope. Optimizations at local scope might result in unintended consequences due to protocols working at cross interests. The authors in [52] adopted an algorithm that tunes the transmission power of a node according to the number of surrounding neighbors in an effort to minimize MAC contention while maintaining network connectivity in ad hoc networks. The scenario presents a cross-layer design integrating the operation of the Network, MAC, and Physical layers. It was primarily devised for UDP traffic and was presented in [91]. The simulation results thus presented in [91] showed the advantage of using a cross layer power-adaptation algorithm in reducing the over-all power consumption in ad hoc networks, while improving the end-to-end network throughput in comparisons to typical ad hoc network settings, where all mobile nodes use the same transmit power. Yet, applying this power-adaptation algorithm to TCP traffic, the authors in [52] illustrated the end-to-end throughput fluctuation due to network oscillation between connectivity and disconnectivity adversely affecting TCP performance, as the devised cross-layer algorithm did not take the Transport layer end-to-end throughput into consideration.

5.3.4.1 Case Study II Simulation Design

The case study is composed of two simulation scenarios: the basic power scenario; and the CellNet scenario. For both scenarios, the simulation parameters and setup are based on those presented in [52] with the minor change of using AODV protocol with link-layer detection for neighbors instead of DSDV protocol with Hello messages. The topology is made of 23 nodes arranged in parallel columns in an area of 500 x 500 m^2 as shown in Figure 5.3. The cross-layer algorithm logic, as stated in [52], is made of two nested loops. The outer loop choses an upper bound on the number of neighbors that each node should have. Neighbor detection is performed at the Link layer. This upper bound is referred to as *target degree*. The inner adaptation loop adjusts the transmit power level at the physical layer to achieve the previously set target degree. Each node has 6 transmit power levels corresponding to 50 m, 90 m, 130 m, 170 m, 210 m, and 250 m when the two-rayground propagation model is used. The outer loop sets a target degree once every 90 sec. The target degree starts at 2 and is increased when the recorded throughput at the network layer is zero, signaling a disconnected network, or as long as the recorded throughput is increasing. The inner loop sets the transmission power every 15 sec to adjust the number of neighbors detected at the Link layer to the target degree. One TCP flow is active from node 0 to node 3. The simulation runs for 500 sec. Table 5.8 summarizes the simulation parameters.

Figure 5.3: Node setup for Case Study II [52]

For the baseline power scenario, the cross-layer algorithm was active throughout the simulation producing similar results to those [52] as shown in Figure 5.4. When comparing the throughput in Figure 5.4 with that provided in [52] and illustrated in Figure 5.5, some differences can be seen. This can be attributed to the following

1. The authors in [52] did not mention the routing protocol used for their simulation, but most probably the routing protocol used was the DSDV, since it is the routing protocol to which the authors had augmented power management in their subsequent research. Since the DSDV is a proactive protocol, i.e. routes are available regardless there is a need for it or not, route discovery phase takes shorter time than reactive protocols such as AODV, providing more stable routes and less down time seen as less fluctuations at the sink.
2. Although the topology setting for the nodes was illustrated in [52], yet the exact distances between the different nodes were not mentioned. Thus the topology used in our simulation is similar, but not exactly the same. Changes in node position may result in changes in the end output.

Simulation Parameters	
No. of Nodes	23
Area	500 x 500 m^2
Total Simulation Duration	500 sec
Transmission ranges	50, 90, 130, 170, 210, 250 m
Propagation model	Two ray ground
Wireless Link Bandwidth	2Mb/sec
MAC protocol	CMU extension
Routing protocol	AODV
No. of flows	1 TCP flow
Target degree starting value	2
Outer loop duration	90 sec
Inner loop duration	15 sec

Table 5.8: Case Study II Simulation Parameters

Despite these differences, our baseline simulation illustrates the adverse effect that the power adaptation algorithm has on the end to end throughput.

For the CellNet scenario, an FOC was instantiated at the edge Ncomps and intermediate Ncomp. The performance parameters that will be monitored by the MU of the NCs are: the power level that results in minimum MAC contention while sustaining next-hop transmission at the ROC; the next hop neighbor associated with each power level used and the number of data packets received at the COC; and the TCP congestion window size at the source AOC. The simulation is divided into two phases, each 250 sec in length. The first phase is an adaptation-learning phase during which the cross layer power-adaptation algorithm controlled the transmission power. Meanwhile, the AOC at the source adjusted the TCP window according to the reported path capacity as estimated by COCs along the path (as explained in case study I). Concurrently, MU at each NC records values assumed by the specified performance parameters. In Phase 2, the FOC on each Ncomp picks the optimal values for the performance parameters (i.e., the power level at which least MAC contention was recorded, the corresponding next hop neighbor on the route from source to destination, and the corresponding throughput. Throughput is measured at the source Ncomp as TCP congestion window size, and at intermediate Ncomp as the number of data packets received at the COC). The FOC sustains these values, overriding any changes attempted by the cross-layer power-adaptation algorithm, AODV, or TCP.

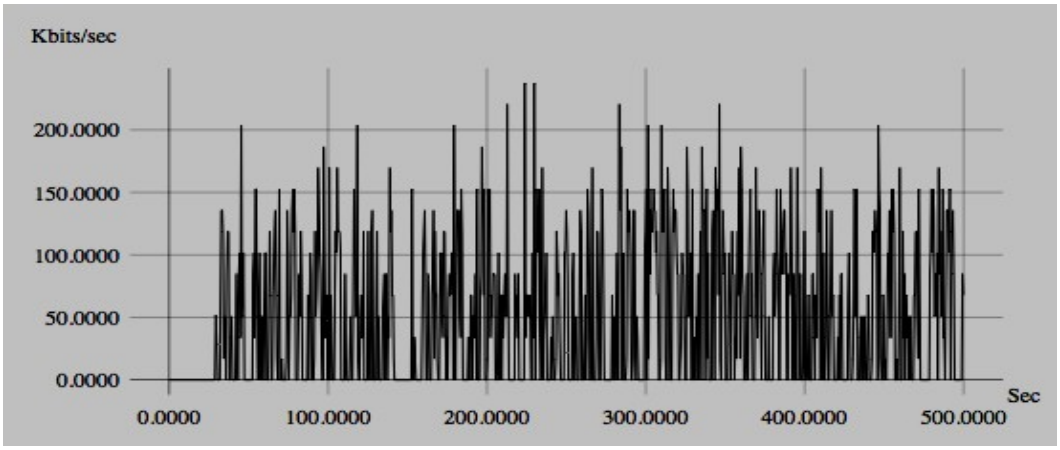

Figure 5.4: TCP Sink Throughput Using Our Baseline Power Scenario

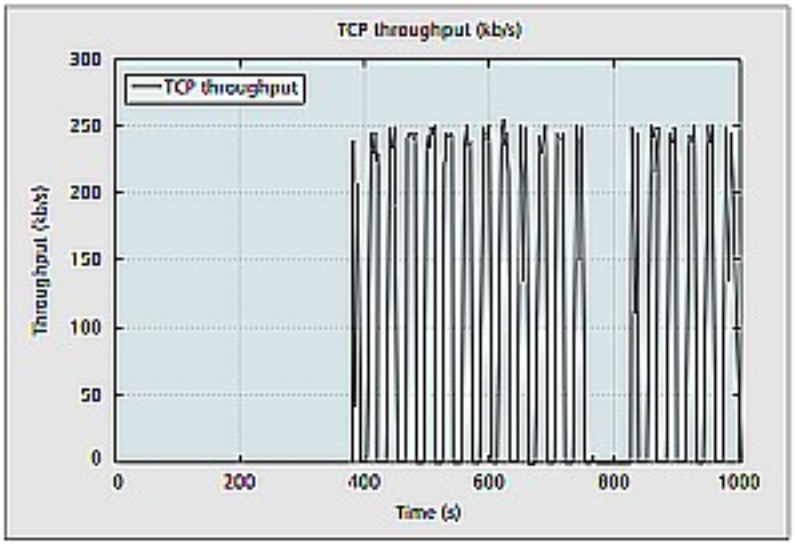

Figure 5.5: TCP Sink Throughput as presented in [52]

5.3.4.2 Case Study II Simulation Results and Analysis

The main focus of the simulation is to illustrate CellNet NCs' capability in recording their states, adapting to context, and recalling the optimal values for performance parameters for enhancing their operations thus better fitting their contexts. Figures 5.6 and 5.7 show the TCP congestion window and resulting sink throughput for one of the replications simulating the basic power scenario. while Figures 5.8 and 5.9 show the corresponding CellNet adapted TCP congestion window and resulting sink throughput. Table 5.9 compares the performance of the basic power scenario and the CellNet scenario expressed in terms of the TCP congestion window size in packets and sink throughput in Kbits/sec, for which we report the grand average (Av) and 95% CI. We also report the CoV for the TCP throughput measured for the duration of each phase (phase 1 and phase 2) for both baseline and CellNet scenarios in Table 5.10.

Performance Parameters		Basic Power Sc. TCP	CellNet Sc. TCP	
			Phase I	Phase 2
Congestion Window size (packets)	Av	5.46	2.86	2.53
	CI	[6.19, 4.74]	[3.04, 2.67]	[3.04, 2.03]
Sink Throughput Kbits/sec	Av	107.25	164.15	175.12
	CI	[131.34, 83.16]	[189.51, 138.79]	[210, 140.24]

Table 5.9: Basic Power Scenario vs. CellNet Scenario

CoV / Simulation Duration		Phase I	Phase II
TCP Sink in the basic scenario	Av	1.03	0.65
	CI	[1.42, 0.64]	[0.88, 0.42]
TCP Sink in the CellNet scenario	Av	0.45	0.08
	CI	[0.51, 0.39]	[0.10, 0.06]

Table 5.10: CoV for TCP Sink Throughput recorded for Basic Power Sc. Vs. CellNet Scenario

We note the following:

- In the CellNet scenario during Phase 1, the TCP congestion window shows less oscillation than that for the baseline scenario as indicated by the ranges of the CI in Table 5.10. This is also evident in Figure 5.8, indicating the advantage of monitoring and regulation in CellNet protocols to stabilize their performance by adjusting to operational context in rapidly changing context. As for phase 2 the sink throughput in the CellNet scenario shows stability reflected as the very low values of the CoV CI.
- In Table 5.9, the CellNet TCP congestion window in Phase 1 and Phase 2 is smaller in size than that recorded for Tahoe TCP in the basic power scenario. Yet, the CellNet sink throughput is higher in both phases. This can be attributed to the fact that CellNet protocols are capable of correctly estimating path capacity and efficiently utilizing the available bandwidth, rather than wasting the bandwidth by overestimating path capacity, and pumping into the network more packets, which are eventually dropped at intermediate nodes.
- FOC was able to learn optimal performance values, and incorporate this knowledge into CellNet protocols at run time, thus stabilizing performance in a highly variable context.

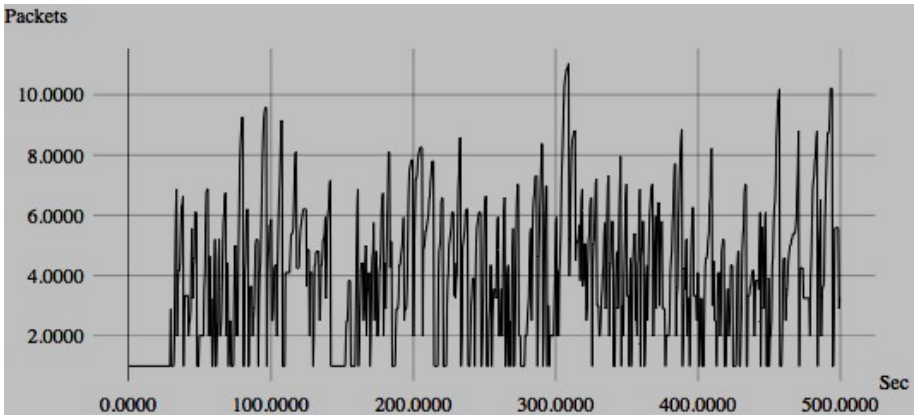

Figure 5.6: Basic Power Scenario TCP Congestion Window Oscillations

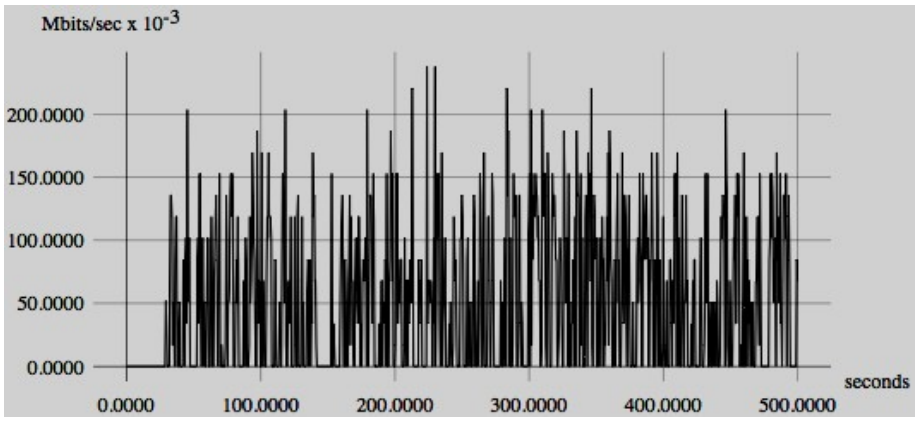

Figure 5.7: Basic Power Scenario Sink Throughput Oscillations

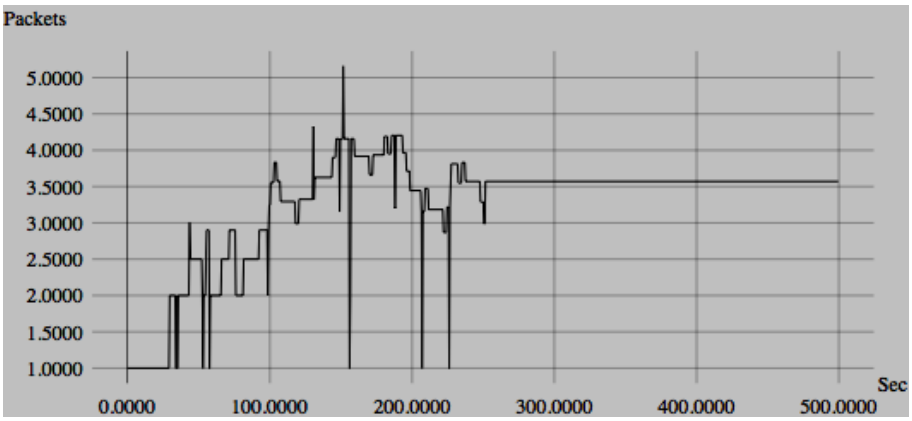

Figure 5.8: CellTCP Congestion Window

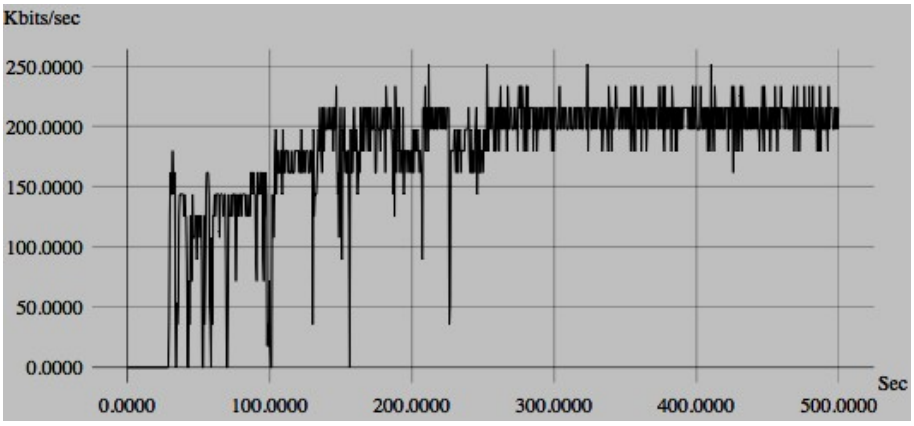

Figure 5.9: Cell Sink Throughput

5.4 Conclusion

In this chapter we presented CellNet, a CORM-based network architecture. CellNet is derived as a minimal configuration architecture to operate within the context of the Internet protocol suite. Engineering protocols to conform to CellNet call for new methodologies and techniques beyond the scope of this research work. However, CellNet-compliant protocols can be developed by embedding legacy protocols into specialized CellNet NCs according to CellNet ACRF concern framework. This setup was used to simulate CellNet over the ns2 simulator. Simulation results shows the capabilities of CellNet-compliant protocols to adapt to context and further evolve by learning optimal values for performance parameters through experience.

Chapter 6

Conclusion and Future Work

"A man's reach should exceed his grasp, or what are the heavens for?"
Vinton G. Cerf

6.1 Summary

The Internet as we know today is an outcome of a sequence of amendments, accretions and innovations that have sprung up and steered the use of the Internet into directions that were not initially anticipated. However, this sea of changes has pushed the Internet to its limits marking it as inadequate for sustaining the future. Such state of the art has urged the network research community to devise future network architectures that can address current Internet deficiencies, acknowledge the need for a trustworthy IT infrastructure, and satisfy the society's emerging and future requirements. Proposed architectures need to be grounded on well-articulated design principles that account for network operational and management complexities, embrace technology and application heterogeneity, regulate network-inherent emergent behavior, and overcome shortcomings attributed to present network realizations. Complying with the network community's architectural request, we present CORM, a clean-slate network reference model for architecting future computer networks. CORM stands as a guiding framework from which several network architectures can be derived according to specific functional, contextual, and operational requirements or constraints. CORM represents a pioneering attempt within this realm, and to our knowledge, CORM is the first reference model that is bio-inspired and derived in accordance with the Function-Behavior-Structure (FBS) engineering framework. CORM conceives computer networks as a software-dependent complex system whose design need to be attempted in a concern-oriented bottom-up approach along two main dimensions: a vertical dimension addressing structure and configuration of network building blocks; and a horizontal dimension addressing communication and interactions among the previously formulated building blocks. CORM addresses this multidimensionality in networks detailing the network in terms of function, structure, and behavior, represented as network-concerns conceptual framework (ACRF), the network structural template (NST), and the information flow model (IFM) respectively. In this dissertation, our main focus was the vertical dimension of the network. Thus our research detailed the first two components of CORM: ACRF and NST. As for the horizontal network dimension, we provided a synopsis of the information flow model, which represents a major area of our future work. Being developed according to a complex system paradigm, CORM refutes the long endorsed concept of layering, intrinsically accounts for emergent behavior, and ensures system integrity and stability. CORM, as a reference model, was validated using the FBS engineering framework, and

its adaptation and evolution capabilities were evaluated through simulating CellNet – a CORM-based architecture.

6.2 Contributions

In this dissertation we adopted a systematic approach to derive a clean-slate network reference model that strives to address the requirements and meet the challenges posed by the network community. Our contribution can be summarized as follows:

- Formalization of network core-design principles: We derived two design principles that expressed the most fundamental characteristic of computer networks regardless of their size, purpose, or operational context. Our design principles conceive computer networks as software-dependent complex systems. This led us to incorporate complex system characteristics as well as SE concepts and principles into our design. A direct consequence of our design principle was a paradigm shift in the network design methodology. We adopted a recursive bottom-up approach in composing the network. Our approach contrasts with the prevailing top-down approach adopted in realizing computer networks. Furthermore it abandons the E2E principle, which is a central principle to the Internet design, by obliterating the difference between the network core and edge.
- Building an evolvable bio-inspired clean-slate network reference model: CORM, our derived concern-oriented reference model, refutes layering, and defines the Network Cell (NC) as the basic abstraction unit for network modeling. The NC structure and inherent behavior are inspired by observations recorded in a recent study on primordial bacterial cells capable of evolution. NC function is derived form the ACRF framework. ACRF framework is the second basic CORM abstraction that depicts computer network functional operation in terms of network concerns derived according to our network requirement specification statement. CORM networks are recursively synthesized from NCs substantiating the vertical and horizontal dimensions of the network. Being formed from NCs, CORM networks possess adaptation, self-organization, and evolution as intrinsic features, thus they are capable of accommodating change.
- Derivation of a CORM-based network architecture and evaluating behavioral modification prospects through simulation: CellNet, a CORM-based network architecture was derived as a minimal architecture to operate within the context of Internet protocol suite. Interoperability between CellNet and Internet protocols will be achieved by encapsulating Internet protocols within CellNet NCs, while maintaining their APIs in addition to the CellNet NCs' APIs. CellNet simulation provided preliminary results that show the adaptation as well as evolution capabilities of protocols designed according to the CORM paradigm.

6.3 Future Work

A major direction in our future work would be to detail the IFM component of CORM posing the following open research issues.

1. Network system profiling: A major constituent of the DR sub-model is devising network knowledge-base (NKB) schema for profiling network entities, protocols, and interactions. Entities can be represented in terms of their capabilities and resources, protocols can be expressed in terms of their performance parameters, and interactions can be modeled in terms of patterns they form (out of which behavioral descriptors can be identified). Out of these elements, key performance indicators (KPI) will be extracted for measurements and evaluation.
2. Network system profile representation: Devise a representation schema for the NKB to be the base for interface design and engineering. The schema needs to address the variability, diversity, and immensity of information existing within the NKB. The representation schema is required to address the need for information abstraction and hiding to handle complexity without disrupting system awareness and sensitivity to context. Furthermore, the representation schema needs to be expressive with no implicit assumptions about the information format or contents. In other word, the design of the representation schema needs to adopt a minimal architecture approach.
3. Devise network evaluation schema: What cannot be measured cannot be evaluated. Accordingly, elements of the derived NKB need to be quantified and measured. Thus, measurement procedures as well as evaluation calibration techniques need to be devised.
4. CORM-compliant protocols: CORM-compliant protocols require a paradigm shift in protocol engineering methodologies. As such, we invite SE practitioners to provide their insights in developing concepts, tools, and methodologies to assist in engineering CORM-compliant protocols.
5. Designing the DC sub-model: A plethora of proposals have addressed the challenging area of network routing in terms of naming addressing and forwarding functions. In CORM, we need to tackle these topics from a FBS perspective taking into considerations the DR sub-model, derived NKB, and representation schema, (behavioral aspects) as well as complex system layout models (structural aspect of CAS) that have already been identified in CAS research literature.
6. Modeling CAS structure: The NST derived in this research work addresses mainly point-to-point connections. We need to expand our NST grammar definition to present other forms of communication including point-to-multipoint and multipoint-to-multipoint connections.
7. Evaluating CORM-derived architectures using Formal Methods: Formal Methods is an approach in SE to assess the correction of software specifications, and

design. Formal methods can be used to assess architectures based on CORM as well as verifying derived protocols.

8. CORM prototyping: CORM-compliant platforms and devices are required for testing and evaluating CORM-based networks and architectures. We conjecture, that CORM-based network simulation results will motivate the interest in constructing CORM-compliant platforms where comparisons of CORM-based network realizations can be performed to test and evaluate proposals for trustworthy future computer networks.

References

[1] D. Clark, “NSF Future Internet Summit,” Meeting Summary, Washington, DC October 12-15, 2009. Version 7.0 of January 5, 2010.

[2] A. Feldmann, “Internet Clean-Slate Design: What and Why?,” ACM SIGCOMM Computer Communication Review,vol.37, no. 3, July 2007

[3] Information and Communication Technologies. The European FIRE Initiative Future Internet Research and Experimentation - An Overview -. Viewed 2010 May 14 Available: http://cordis.europa.eu/fp7/ict/fire/overview_en.html

[4] V. Cerf and R. kahn, “A Protocol for Packet Network Intercommunication,” IEEE Transaction on Communication, vol. 22 No 5 May 1974.

[5] J. Saltzer, D. Reed, and D. Clark, “End-To-End Arguments in System Design,” ACM Transactions on Computer Systems, vol. 2, pp.277–288, November 1984.

[6] H. Zimmermann, “OSI Reference Model-The IS0 Model of Architecture for Open Systems Interconnection,” IEEE Transactions on Communications, vol. 28, April 1980.

[7] Greenberg, et. al. “A Clean Slate 4D Approach to Network Control and Management,” ACM SIGCOMM Computer Communication Review, vol. 35, pp.41-54, October 2005.

[8] J. Khoury, C. Abdallah, and G. Heileman, “Towards Formalizing Network Architectural Descriptions,” Springer Berlin / Heidelberg, vol. 5977, pp.:132-145, 2010

[9] J.S. Gero “Design Prototypes: A Knowledge Representation Schema for Design,” AI Magazine, vol. 11, pp. 26–36, Winter,1990.

[10] Melanie Mitchell,“Complex systems: Network thinking,” Artificial Intelligence, vol 170, pp. 1194-1212, December 2006 .

[11] H. Mili, A. Elkharraz and H. Mcheick, “Understanding Separation of Concerns,” Viewed 2010 May 14 Available: http://www.latece.uqam.ca/publications/mili-kharraz-mcheick.pdf.

[12] Zundong Zhang, Limin Jia, Yuanyuan Chai1, Min Guo, “A Study on the Elementary Control Methodologies for Complex Systems,” Control and Decision Conference, pp. 4455-4460, July, 2008.

[13] E. Ben-Jacob and H.Levine, “Self-engineering Capabilities of Bacteria,” Journal of Royal Society. [Online]. Available: http://star.tau.ac.il/~eshel/papers/Interface.pdf

[14] “Software Architecture: Central Concerns, Key Decisions,” Architecture Resources For Enterprise Advantage. Viewed May 14, Available: http://www.bredemeyer.com

[15] Samir Rihani. Nonlinear Systems. Viewed May 14, Available: http://www.globalcomplexity.org/NonlinearSystems.htm

[16] K. Dooley, “A Nominal Definition of Complex Adaptive Systems,” The Chaos Network, vol. 8, pp. 2-3 1996.

[17] Internet Pioneers. Paul Baran. Viewed May 14, Available: http://www.ibiblio.org/pioneers/baran.html
[18] The Internet. Paul Baran Invents Packet Switching. Viewed May 14, Available: http://www.livinginternet.com/i/ii_rand.htm
[19] The Internet. UK National Physical Laboratory (NPL) & Donald Davies. Viewed May 14, Available: http://www.livinginternet.com/i/ii_npl.htm
[20] L. Pouzin, "Presentation and Major Design Aspects of the Cyclades Computer Network". Proceedings of the NATO Advanced Study Institute on Computer Communication Networks. Sussex, United Kingdom: Noordhoff International Publishing. pp. 415-434, 1973. [Online] Available: http://rogerdmoore.ca/PS/CYCLB.html
[21] Wikipedia. CYCLADES. Viewed May 14 Available: http://en.wikipedia.org/wiki/CYCLADES
[22] Highlights of JF Koh's Honours Programme, 1997. Internet Timeline. Viewed May 14 Available: http://wwwmcc.murdoch.edu.au/ReadingRoom/VID/jfk/timeline.htm
[23] THINK: Technical Histories of Network Protocols: A Technical History of the ARPANET- A Technical Tour Overview. Viewed May 14. Available http://userweb.cs.utexas.edu/users/chris/nph/ARPANET/ScottR/arpanet/tour/overview.htm
[24] THINK: Technical Histories of Network Protocols: Timeline. Viewed May 14. Available: http://userweb.cs.utexas.edu/users/chris/nph/ARPANET/ScottR/arpanet/timeline.htm
[25] Internet Society: Histories of the Internet- The Initial Internetting Concepts. Viewed May 14. Available: http://www.isoc.org/internet/history/brief.shtml#Initial_Concepts
[26] THINK: Technical Histories of Network Protocols:A Technical History of the ARPANET- A Technical Tour IMP-to-IMP. Viewed May 14. Available : http://userweb.cs.utexas.edu/users/chris/nph/ARPANET/ScottR/arpanet/tour/impimp.htm
[27] THINK: Technical Histories of Network Protocols: Overview of the Host-to-IMP Protocols. Viewed May 14. Available: http://userweb.cs.utexas.edu/users/chris/nph/ARPANET/ScottR/arpanet/tour/hostimp.htm
[28] THINK: Technical Histories of Network Protocols: Overview of the Host-to-Host Protocols. Viewed May 14. Available: http://userweb.cs.utexas.edu/users/chris/nph/ARPANET/ScottR/arpanet/tour/hosthost.htm
[29] L. H. Landweber "The Computer Science Network," Al Magazine Vol.3, 1982. [Online] http://www.aaai.org/ojs/index.php/aimagazine/article/viewFile/381/317
[30] CSNET - Computer Science Network. Viewed May 14. Available http://www.livinginternet.com/i/ii_csnet.htm
[31] D. Koren, The OSI Model - A Surviving Remnant of the OSI Protocol Stack. Viewed May14. Available http://www2.rad.com/networks/introductory/layers/main.htm
[32] David D. Clark, "The Design Philosophy of the DARPA Internet Protocols," in Symposium Proceedings on Communication Architecture and Protocols Review vol. 18, pp. 106–114, August 1988.

[33] D. Clark, J. Wroclawski, K. Sollins, and R. Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet," SIGCOMM Computer Communication Review, pp.347 -356, 2002.
[34] M. Blumenthal, and D. Clark, "Rethinking the design of the Internet: The end to end arguments vs. the brave new world,e ACM Transactions on Internet Technology, vol.1 pp.70–109, August 2001.
[35] D. Clark et.al., (August 2004) "New Arch: Future Generation Internet Architecture", http://www.isi.edu/newarch/iDOCS/final.finalreport.pdf
[36] J. Saltzer, D. Reed, and D. Clark, "End-To-End Arguments in System Design," ACM Transactions on Computer Systems, vol. 2, pp. 277–288, November 1984.
[37] D. Clark, K. Sollins, J, Wroclawski, and T. Faber, "Addressing Reality: An Architectural Response to Real-World Demands on the Evolving Internet," ACM SIGCOMM Workshops, Karlsruhe, Germany, August ,2003.
[38] D. Krioukov, K. Claffy, K. Fall, and A. Brady, "On Compact Routing for the Internet," ACM SIGCOMM Computer Communication Review, vol. 37, July 2007.
[39] D1.4/5/6v1: ANA Blueprint – First Version. Workpackage 1 Deliverable 1.4/5/6v1, ANA Project FP6-IST-27489, February 2007
[40] R. Jain, "Internet 3.0: Ten Problems with Current Internet Architecture and Solutions for the Next Generation," Proceedings IEEE Military Communications Conference (Milcom 2006), Washington DC, October, 2006.
[41] D. Clark and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," In Proceedings ACM SIGCOMM, Sept 1990.
[42] R. Brännström, R. Kodikara , C. Ahlund, and A. Zaslavsky, "Mobility Management for multiple diverse applications in heterogeneous wireless networks," In Proceedings of Third IEEE Consumer Communications and Networking Conference (CCNC), pp. 818-822, January 2006.
[43] W. Eberle, B. Bougard, S. Pollin, and F. Catthoor, "From Myth to Methodology: Cross-Layer Design for Energy-Efficient Wireless Communication," In Proceedings of the 42nd Annual Design Automation Conference (DAC), pp. 303-308, 2005.
[44] G. Carneiro, J. Ruela, M. Ricardo, "Cross-Layer Design in 4G Wireless Terminals," IEEE Wireless Communication Magazine. vol. 11,pp. 7-13, April 2004.
[45] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks," IEEE Communication Magazine, vol. 44, pp. 134-141, November 2006
[46] FIND – Future Internet Design (FIND) - US National Science Foundations (NSF). Viewed May 14, Available: http://www.nets-find.net/
[47] FIRE - Future Internet Research & Experimentation. Viewed May 14, Available: http://cordis.europa.eu/fp7/ict/fire/home_en.html
[48] L. Larzon, U. Bodin, and O. Schelen "Hints and Notifications", In Proceedings of IEEE Wireless Communications and Networking Conference (WCNC), Orlando, Florida, USA, March 2002.

[49] Q. Wang, and M.A. Abu-Rgheff, " Cross-layer Signaling for Next Generation Systems, " IEEE Wireless Communications and Networking Conference (WCNC), vol.2, pp.1084-1089, March 2003.
[50] V. Raisinghani and S. Iyer, "Cross-Layer Feedback Architecture for Mobile Device Protocol Stacks," IEEE Communications Magazine, vol. 44, pp. 85-92, January 2006.
[51] K. ElDafrawy, M.El Zarki, and M.Khairy, "Proposal for Cross-Layer Coordination Framework for Next Generation Wireless Systems," Proceedings of the International Conference on Wireless Communications and Mobile Computing (IWCMC'06), pp. 141-146, July 2006.
[52] V. Kawadia and P. R. Kumar, "A Cautionary Perspective on Cross Layer Design," IEEE Wireless Communication, vol. 12, pp. 3–11, February 2005.
[53] Ilia Baldine et.al., "A Unified Software Architecture to Enable Cross-Layer Design in the Future Internet," [Online], Available http://net-silos.net/joomla/index.php?
[54] J. Touch, Y. WangV. Pingali, "A Recursive Network Architecture," [Online] Available: http://www.isi.edu/touch/pubs/isi-tr-2006-626/
[55] V. Jacobson, et. al., "Networking Named Content," In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT), Rome, Italy, December , 2009.
[56] N. Shenoy, et. al " A Scalable Architecture," [Online]. Available: www.nets-find.net/Meetings/S09/MeetigTalks/nirmala.ppt
[57] Ariane Keller, et.al., "A System Architecture for Evolving Protocol Stacks," Invited Paper in IEEE Proceedings of the 17th International Conference on Computer Communications and Networks (ICCCN) pp. 144-150, August 2008.
[58] J. Scott, P. Huiy, J. Crowcrofty, and C. Diot, "Haggle: a Networking Architecture Designed Around Mobile Users," In Proceedings of the Third Annual Conference on Wireless On-demand Network Systems and Services, pp. 78-86, 2006. [Online]. Available: http://www.cl.cam.ac.uk/~ph315/publications/haggle-wons06-editforxtoff.pdf
[59] BIOlogically inspired NETwork and Services (BIONETS). Viewed May 14. Available: http://www.bionets.eu/index.php?area=11
[60] C. Tschudin. "Flexible Protocol Stacks" In Proceedings of The Conference on Communications Architecture and Protocols, pp. 197-205, Zurich, Switzerland, October 1991.
[61] M. Jung, and E. Biersack, "A Component-Based Architecture for Software Communication Systems," In Proceedings of the 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), p.36, Edinburgh, Scotland, April 2000.
[62] R. Braden, T. Faber, and M. Handley, "From Protocol Stack to Protocol Heap -- Role-Based Architecture,", HotNets-I, October 2002; also in Computer Communication Review, vol. 33, January 2003.

[63] “Reference Model for Service Oriented Architecture 1.0 ,” Committee Specification 1. August 2006. [Online]. Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm
[64] K. Park and W. Willinger, “The Internet As a Large-Scale Complex System,” Oxford University Press, Jun 2005.
[65] Y. Bar-Yam, “Dynamics of Complex Systems,” Addison-Wesley, August 1997.
[66] J. Lillibridge, “Complex System,” [Online] Available: www.hoyt.org/documents/jack-complex.pdf
[67] J. Mogul, “Emergent (Mis)behavior vs. Complex Software Systems,” In the Proceeding of the 2006 EuroSys Conference, pp. 293-304, April, 2006.
[68] D. Stirling, “ Modeling Complex Systems,” [Online]. Available http://www.learndev.org/dl/BtSM2005-Stirling-Complexity.pdf
[69] Wikipedia. Software Engineering. Viewed May 14. Available http://en.wikipedia.org/wiki/Software_engineering
[70] A. Gacemi1, A. Senail, M. Oussalah2, “Separation of Concerns in Software Architecture via a Multiviews Description,” Proceedings of the IEEE International Conference on Information Reuse and Integration, pp. 60-65, 2004.
[71] N.M. Chowdhury and R. Boutaba, “Network Virtualization: State of the Art and Research Challenges,” IEEE Communications Magazine, vol.47, pp. 20-26, July 2009.
[72] Aaron J. Burstein and Fred B. Schneider, “Trustworthiness as a Limitation on Network Neutrality,” Federal Communications Law Journal, June, 2009. [Online] Available: http://www.highbeam.com/doc/1G1-206794866.html
[73] Xue-hai Peng and Chuang Lin, “Architecture of Trustworthy Networks,” Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing, pp. 269-276, 2006.
[74] Resilience. Wikipedia. Viewed May 14. Available: https://wiki.ittc.ku.edu/resilinets/Definitions#Resilience
[75] Survivability. Wikipedia, Viewed May 14. Available: https://wiki.ittc.ku.edu/resilinets/Definitions#Survivability
[76] D. A. Menascé, “Virtualization: Concepts, Applications, and Performance Modeling,” TechRepublic. [Online]. Availabel: http://whitepapers.techrepublic.com.com/abstract.aspx?docid=344979
[77] N. Niebert et. al., “Network Virtualization: A Viable Path Towards the Future Internet,” Wireless Personal Communications, vol. 45, pp.511-520, June, 2008
[78] Philippe Kruchten, “Casting Software Design in the Function-Behavior-Structure Framework,” IEEE Software, vol.22, pp.52-58, 2005.
[79] K. Dorst P. E. Vermaas, “John Gero’s Function-Behaviour-Structure model of designing: a critical analysis,” Research in Engineering Design, vol. 16, pp. 17-26, November, 2005.
[80] H. Hassan, R. Eltarras, M. Eltoweissy, “Towards a Framework for Evolvable Network Design,” Collaborate Com 2008, Orlando, FL, USA, Nov. 2008.

[81] H. Hassan, et. al., “CellNet: A Bottom-Up Approach to Network Design,” NTMS 2009, Third International Conference on New Technologies, Mobility and Security, Cairo, Egypt, December 2009.
[82] “The Network Simulator – ns-2,” http://www.isi.edu.nsnam/ns/
[83] E. Hamadani and V. Rakocevic, “A Cross Layer Solution to Address TCP Intra-flow Performance Degradation in Multihop Ad hoc Networks,” Journal of Internet Engineering, vol. 2, pp. 146-156, June 2008
[84] Z. Fu, et. al., “The Impact of Multihop Wireless Channel on TCP Performance,”IEEE Transactions on Mobile Computing, vol. 4, pp.209-221, March/April 2005
[85] K. Nahm, A. Helmy, and C. Jay Kuo, “TCP over Multihop 802.11 Networks: Issues and Performance Enhancement,” Proceedings of the 6th ACM International symposium on Mobile ad hoc networking and computing, IL, USA, 2005
[86] S. Kurkowski, T. Camp, and M. Colagrosso, “MANET Simulation Studies: The Incredibles,” Mobile Computing and Communications Review, vol. 9,pp 50-61, 2005.
[87] “The Network Simulator ns-2: Documentation,” [Online]. Available http://www.isi.edu/nsnam/ns/ns-documentation.html
[88] J. Li, C. Blake, D.S.J. De Couto, H.I. Lee, and R. Morris, “Capacity of Ad Hoc Wireless Networks,” In the Proceedings of 7th ACM International Conference on Mobile Computing and Networking(MOBICOM), Rome, Italy, July, 2001.
[89] M. Umlauft, and P. Reichl, “Experiences with the ns-2 Network Simulator - Explicitly Setting Seeds Considered Harmful,” Wireless Telecommunications Symposium WTS 2007, Pomona, Californien, USA.
[90] S. Floyd, Ed., “Metrics for the Evaluation of Congestion Control Mechanisms,” RFC 5166. [Online] Available. http://www.ietf.org/rfc/rfc5166.txt
[91] T. A. ElBatt et al., “Power Management for Throughput Enhancement in Wireless Ad-hoc Networks,” In Proceedings of IEEE International Conference on Communications (ICC), pp. 1506–13, 2000.
[92] F. Polack et. al., “Complex Systems Models: Engineering Simulations,” in ALife XI. MIT press, 2008.
[93] K. Pawlikowski, J. Jeong, and R. Lee, “On credibility of simulation studies of telecommunication networks,” IEEE Communications Magazine, pages 132–139, 2001.pages 50–59, 2000.