Towards a Framework for Evolvable Network Design

Conference Paper · November 2008

DOI: 10.1007/978-3-642-03354-4_29 · Source: DBLP

3 authors:

Hoda Hassan
The Knowledge Hub Universities
13 PUBLICATIONS **53** CITATIONS
SEE PROFILE

Ramy Eltarras
Virginia Polytechnic Institute and State University
9 PUBLICATIONS **51** CITATIONS
SEE PROFILE

Mohamed Eltoweissy
Virginia Military Institute
152 PUBLICATIONS **2,744** CITATIONS
SEE PROFILE

Towards a Framework for Evolvable Network Design

Hoda Hassan, Ramy Eltarras, Mohamed Eltoweissy

The Bradley Department of Electrical and Computer Engineering, Virginia Tech
hmhassan@vt.edu, ramy@vt.edu, toweissy@vt.edu

Abstract. The layered Internet architecture that had long guided network design and protocol engineering was an "interconnection architecture" defining a framework for interconnecting networks rather than a model for generic network structuring and engineering. We claim that the approach of abstracting the network in terms of an internetwork hinders the thorough understanding of the network salient characteristics and emergent behavior resulting in impeding design evolution required to address extreme scale, heterogeneity, and complexity. This paper reports on our work in progress that aims to: 1) Investigate the problem space in terms of the factors and decisions that influenced the design and development of computer networks; 2) Sketch the core principles for designing complex computer networks; and 3) Propose a model and related framework for building evolvable, adaptable and self organizing networks We will adopt a bottom up strategy primarily focusing on the building unit of the network model, which we call the "network cell". The model is inspired by natural complex systems. A network cell is intrinsically capable of specialization, adaptation and evolution. Subsequently, we propose CellNet; a framework for evolvable network design. We outline scenarios for using the CellNet framework to enhance legacy Internet protocol stack.

1. Introduction

Engineering of computer networks has long been influenced by concepts developed during the inchoative stage of computer network design. This resulted in adopting a layered architecture for abstracting network functionalities as well as for engineering network protocols; a methodology that proved to be neither adaptable nor evolvable in response to the continuous change in network operational requirements and technological advancements. As a direct consequence to the layered model limitations, a myriad of alternative network architectures have been proposed targeting computer networks in general and the Internet in particular. While some ingenious proposals were developed at times, we argue that the general trend towards network science and engineering lacks a systematic formalization of core principles that need to guide the process of network design. This research work is an attempt to 1) investigate the problem space in terms of the factors and decisions that influenced the design and development of computer networks 2) extract the core principles for designing a computer network relying heavily on concepts and practices adopted in complex systems modeling, software engineering and computer communications 3) Propose a model and related framework for building evolvable, adaptable and self organizing future networks relying on studies performed on natural complex systems, the structure of their components, and their overall intrinsic behavior specifically studies conducted on primordial communities of bacteria. We will adopt a bottom up strategy primarily focusing on the building unit of the network, which we call the "network cell". We argue that specialization, adaptation and evolution should be intrinsic features of the "network cell" for the network to exhibit intelligent emergent behavior.

The remainder of this paper is organized as follows. Section 2 gives an overview of the problem space; section3 introduces the proposed "network cell"; section 4 explores the evolutionary capability of the network cell; section 5 sketches the CellNet framework within which the network cell will operate; section 6 provides a formal definition of networks and internetworks in terms of the network cell and finally section 7 provides an example for the model realizations. The paper concludes in section 8 with an outline of future work.

2. Computer Network Design

Different computer network concepts, paradigms, and projects were developed before computer internetworking came into being. The most prominent of these projects was the ARPANET. The APRANET was a packet switched store and forward network whose primary mission was to provide distributed data communication network that can withstand almost any degree of destruction to individual components without losing end-to-end communications [1]. The success of the ARPANET to connect isolated computers and the rise of other national networks motivated the idea of inter-connecting networks. It was then realized that a general internetworking protocol is required. This was the motivation for designing the TCP protocol [2]. The adoption of TCP/IP as the "transport protocol" for internetworking could be considered the initiation of the Internet as we know it today. TCP/IP suite abstracts the interconnection functionalities into five layers. The astounding success of the TCP/IP protocol suite in interconnecting desperate networks motivated the adoption of a layered architecture as a reference model for network protocol engineering and development even for protocols running within intranets. The primary motivation of our work is highlighting this misconception; the layered architecture that had long guided network design and protocol engineering was an "interconnection architecture" defining a framework for interconnecting networks rather than a model for network structuring and engineering. We claim that the prevalent approach of abstracting the network in terms of an internetwork hinders the thorough understanding of the network salient characteristics and interactions resulting in impeding design decisions. Hence we embark by clarifying our vision of the network and its relationship to an internetwork. We admit that the presented definitions are already recognized in literature but our contribution is the reasoning that follows the definitions. We define a network as a communication substrate that allows the exchange of data among two or more computers despite the possible heterogeneity in hardware, middleware and software of the attached computers. While an internetwork is defined as a communication substrate that allows the exchange of data among two or more computers by connecting the network communication substrate that connects these computers despite the possible heterogeneity in hardware, middleware and software of both the computers and the communication substratum, i.e. an internetwork is a network of networks connected by internetwork communication substratum. In that sense we reason that the definition of an internetwork is actually a recursive definition of a network with the base case being a network. Thus we reason further that to design an internetwork we need first to design the network that forms the building block for an internetwork.

To design a computer network we note that computer networks are complex by definition. Their complexity can be attributed to the heterogeneous and distributed nature of the technologies adopted, protocols employed, devices used, applications supported, variability of operating conditions and performance requirements, not to mention the unpredictable interactions of all previously mentioned aspects. Hence computer networks stand as a typical example for software intensive complex systems.
Three inherent properties shared by all complex systems are complexity, emergent behavior and composability from autonomous components [3]. Complexity refers to the immense amount of information required to depict the system profile in terms of macro and micro states. Emergent behavior in complex system refers to the ability of the components of the system 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. Emergent behavior can be classified as self-organization, adaptation or evolution. Self-organization refers to changes in component individual behavior due to inter-component communication, while 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, on the other hand, refers to a higher form of intelligent adaptation and/or self organization of components in response to changes by accounting on previously recorded knowledge form past experience(s). Evolution usually implies the presence of memory elements as well as monitoring functions in evolvable components. Finally composability from autonomous components implies the distributed structure of complex systems where different entities collaborate to perform the global system function. Although delineating the properties of complex systems are rather straightforward, yet designing a system that exhibits these properties is a challenging task. This observation led us to 1) devise a network building block that we coin as the "network cell" by which we build the network in an attempt to imitate natural complex systems' structure behavior and capabilities; and 2) Adopt

Separation of Concerns (SoC) as software engineering methodology to tackle functional decomposition in networks yielding our proposed CellNet framework that identifies Application, Communication, Resource and Federation as four functional concerns.

3. The Network Cell

Our research in natural complex systems led us to an interesting study on primordial bacteria that provided us with a vivid representation of the main features that need to be present in entities composing a complex system. In this recent study it has been noted that bacteria is capable of self engineering in the sense that it can "utilize intricate communication capabilities to cooperatively form (self-organize) complex colonies with elevated adaptability—the colonial pattern is collectively engineered according to the encountered environmental conditions." The bacterial cell is described to be "analogous to a complex man-made cybernetic system composed of information-processing systems and at least two thermodynamic machines. Their outer membranes enable them to sense the environment and to exchange energy, matter, and information with it. The internal state and stored information of the cell and the surrounding conditions regulate the membranes." [4].

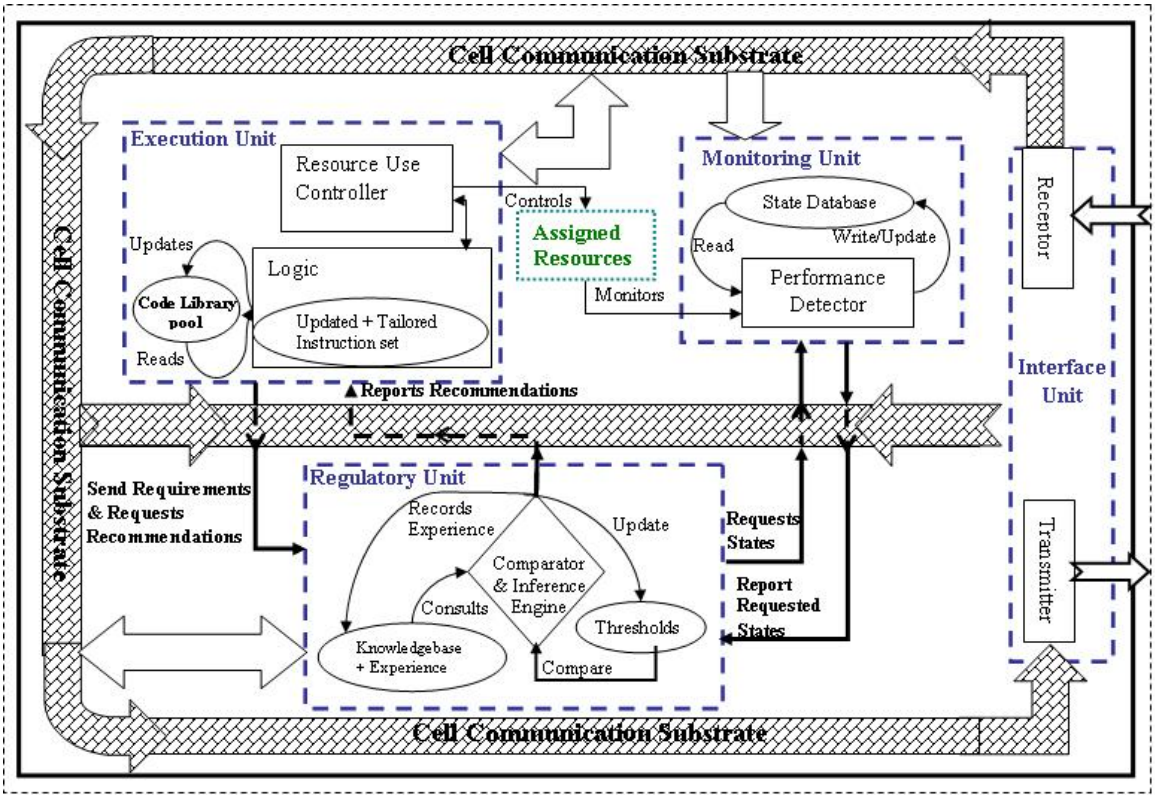

Fig. 1. Generic Network Cell structure

Guided by the previous description, we present the structure of our generic network cell (GNC) whose structure will be common to all network cells (NC) regardless of their assigned responsibilities or roles. As shown in figure 1, the GNC is composed of four units; the Interface Unit (IU), the Monitoring Unit (MU), the Regulatory Unit (RU) and the Execution Unit (EU). A 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 and 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 behavior realized and instruction set to be executed by all units. For example, the parameters to be monitored and states recorded

by the MU, the threshold values to be used by the RU, the library pool out of which the logic to be tailored and executed by the EU. Once the NC is assigned a functional role it turns to be a specialized NC. Therefore the GNC is just a template out of which all specialized NCs 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 the role(s) it is assigned. Once a NC materializes by assuming a functional role, it will be assigned, according to its functional needs, a portion of the system resources required for its proper operation. These resources include primarily memory and CPU cycles. Following is an outline for the intrinsic operation of each of the units shown in figure 1 as this will be common to all NCs

- Interface Unit (IU) is the NC boundary allowing it to communicate with the outside world (environment or peer NCs). Through the IU the NCs receive and transmit different forms/representations of data (states, instructions, control, content etc…)
- Monitoring Unit (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. The MU will extract state information and represent these states in a quantifiable format. These quantified states are then stored in the state database to be retrieved upon requests received from the RU
- Regulatory Unit (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 form the EU asking for advices and recommendations for performance enhancement or by performance deterioration. The RU starts inspecting environmental parameters to infer the reasons that accounted for performance deterioration and this step may lead to communication with neighboring peers requesting their views of the environment. The RU has the capability of gaining knowledge and learning from past experience which it records in the Knowledge/Experience database. Therefore the RU provides educated recommendation to the EU to optimize its operation. In addition, the RU may update some of the threshold levels according to its inferred decisions
- Execution Unit (EU) is responsible to execute 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. The EU is composed of two main components; the Logic component (LC) and the Resource Use Controller component (RUCC). The logic component is the part responsible for performing the NC function. The Logic component starts by creating a set of instructions that best accomplishes the required function using the library pool. It also requests the RU recommendations thus incorporating both the RU knowledge and experience as well as accounting for environmental alterations. Once the LC receives feedback recommendations form the RU it might update its tailored instruction set to fit the inferred operational status. The RUCC on the other hand is responsible for managing resources assigned to the NC. The RUCC works together with the LC to ensure optimal internal resource usage and distribution. The RUCC is also responsible for estimating the required level of resources for the NC operation as a whole and thus requests the estimated resources form the system.

4. Network Cell Evolution

The capability of network cell (NC) to record its past experience, and use it to infer the future updates required for its software at runtime, in response to internal or external triggers, imply that the NC is capable of evolution. Accordingly a network built of NCs will also be capable of evolution as the network behavior is perceived as the collective behavior of its constituents. We envision the NC experience to be recorded in terms of perceived behavioral patterns, responses to a perceived pattern or a sequence of patterns represented as sequence of updates, and the resulting outcome of each of the adopted decisions in terms of the overall gain in cell performance. The inference engine in the NC's RU will be responsible to pick up the most suitable course of action based on the recorded experience and present NC states. Since each cell can undergo changes independent of its peer cells we need a mechanism to guarantee that evolving cells will maintain communication in spite of any changes applied to their software To allow the independent evolution of cells while maintaining continuous operation of the network, a dynamic interface management technique is required to insure that progressive evolution of cells does not lead to network breakage. A cell publishes one or more local interfaces exposing its services to other cells in the network. When a cell

evolution induces changes in the cell local interface, the cell publishes a new interface to support its evolved behavior while retaining the old interfaces to allow other cells that didn't undergo a compatible evolution to continue using the cell services through the older interfaces. Dwelling further into the responsibilities of the interface management capability we foresee that the IU within a cell encompasses the following subunits dedicated for interface management

- Local Interface Manager (LIM): allows the registration/de-registration of local interfaces at runtime. It publishes the local interfaces to other cells, maintains a List of Published Interfaces (LPI), responds to queries requesting available interfaces, and proxies interface invocations to appropriate service implementation.
- Remote Interface Manager (RIM): allows the registration/de-registration of remote interfaces at runtime and maintains a List of Remote Interfaces (LRI). The LRI determines the capabilities of the cell requesting a service from another cell. It also provides stubs to the interfaces listed in LRI to allow the hosting cell to invoke services on other cells.
- Cooperation Interface Manager (CIM): Multiple cell cooperation can be done by implementing cooperation interfaces dictated by the cell managing the cooperation. These interfaces allow the cell responsible of cooperation management to communicate with the participant cells without getting into their implementation details. Like any other cell, the cooperation manager cell is capable of evolution and consequently new cooperation interfaces can be registered. The cooperation participants selects the most appropriate cooperation interface to use in the same way cells select the most appropriate remote interface to use. Accordingly, CIM allows the registration/de-registration of the cooperation interfaces at runtime. It maintains a List of Cooperation Interface when the cells request to cooperate

Using the cell paradigm, we note that the NC, similar to the biological cell, will have a life cycle where NC starts at its infancy with minimal experience, and through learning, the NC experience will mature. This gained experience can be transferred to other naïve NCs introducing the phenomenon of generational learning among NCs.

5. The CellNet Framework

Having derived the structure of the NC, which represents the basic building block out of which the network will be constructed, we embark to define the framework within which the NC will operate in an attempt to formalize a new network design methodology that we refer to as Cell Oriented Architecture (COA). One important thing to emphasize is that a computer network is not a stand alone system but rather a complex distributed system requiring a dual faceted design strategy to address both the vertical functional decomposition and the horizontal functional distribution while paying attention to the different functional dependencies and interactions that exist along these two dimensions. Relying on SE merhods in tackling complexity we adopt the principle of Separation of Concerns (SoC) to derive CellNet framework. SoC is a general problem-solving technique that addresses complexity by cutting down the problem space into smaller manageable, loosely-coupled, and easier to solve sub-problems allowing for better understanding and design. SoC addresses concerns from two different views. The first view defines two types of concerns; core concerns and cross-cutting concerns. Core concerns refer to the core functionalities of a system that can be identified with clear cutting boundaries and hence represented in separable modules. Cross-cutting concerns, on the other hand are behavioral aspects of the system functions that span over multiple modules trying to manage or optimize the core concerns, and if not carefully represented, result in scattering and tangling of system resulting behavior.

The second view of the SoC concept differentiates between concerns, whether core or cross-cutting, representing the loci of functional computations and states, and concern interactions representing the flow of information and state communication. Guided by the first view CellNet abstracts the core network concerns into application-oriented, communication-oriented and resource-oriented functions and allows for concern federation thus addressing the cross cutting functional aspects among the previously identified core concerns. Figure 2 (left) models the CellNet component and for better understanding we present it superimposed on the TCP/IP model in figure 2 (right). Yet we emphasize, as indicated in the figures, that

the CellNet framework does not impose layering among its defined abstractions allowing for diverse inter-functional communication.

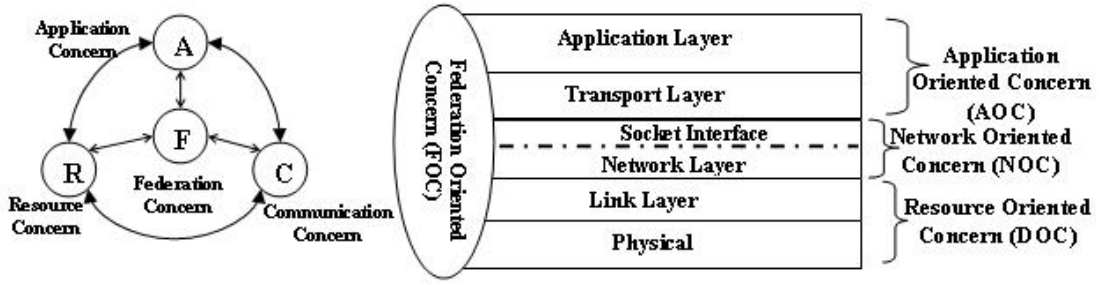

Fig. 2. The CellNet model (left). The CellNet framework superimposed on the TCP/IP model

The CellNet framework can be realized through the instantiation of the following specialized NCs (SNC): Resource Oriented Cell (ROC), Communication Oriented Cell (COC), Application Oriented Cell (AOC), and Federation Oriented Cell (FOC). Each of these NCs will have the structure of GNC, but each will have its dedicated function that will be executed by the logic in their EU. The instantiation of AOC, COC and ROC is mandatory for the correct operation within a network. In contrast the FOC instantiation is optional, and when in action it acts as a global RU that receives inquiries form and sends recommendations to local RUs within the federating NCs.

For the second view of SoC, we recall our definitions for network and internetworks. Accordingly, we identify three boundaries for NC interactions; within the same computer, on different computers communicating using a network communication substrate, and on different computers communicating using an internetwork communication substrate. NCs instantiated on the same computer form what we call a network component (NComp). The NComp can be composed of any number of specialized NCs but the minimum configuration for an NComp is one instantiation of each of the basic NCs (AOC, COC, and ROC). NComp represent the first interaction boundary. The second interaction boundary is represented by a network hosting two or more NComp communicating using the same homogenous substrate. The third interaction boundary is represented by an internetwork hosting two or more networks, which on turn host the NComps communicating using a heterogeneous internetwork substrate thus requiring an intermediate NComp to act as a translator. The formal definition for this setup is defined in the next section.

The NCs accompanied by the CellNet framework present a new paradigm for building networks. However, to be realized within the present legacy network we propose attaching specialized NCs to existing protocols according to network functional decomposition as dictated by the CellNet framework. For example, the reliable transport service provided by TCP can be encapsulated into an AOC while socket management will be controlled by a COC

6. Formal CellNet Based Network Definition

In this section we present the formal definition for network structure based on CellNet using EBNF. The definition will use the following notations:

Trailing * means repeat 0 or more times.

Trailing + means repeat 1 or more times

The following abbreviations will be used

INet = Internetwork

Net = Network

IComSub = Internetwork communication substrate

NComSub = Network communication substrate

NComp = Network component

NC = Network Cell

Network Definition

INet = NComp (IComSub NComp)+
IComSub = (NComSub NComp)+ NComSub
Net = NComp (NcomSub NComp)+
NComp = AONE CONE RONE NC*
NC = AONE | CONE | RONE | FONE

7. Network Realization Using CellNet

In this section we present an example to show how a legacy network can be enhanced using CellNet. The CellNet framework introduces monitoring adaptation and learning to enable

1. The TCP protocol to realize the contention level on the links along the path; and
2. The MAC layer to adapt power level to reduce interference, while handling adverse unintended consequences that might occur.

The scenario uses cross layer designs proposed in [5] and [6] and uses the network topology used in [5] and shown in figure 3

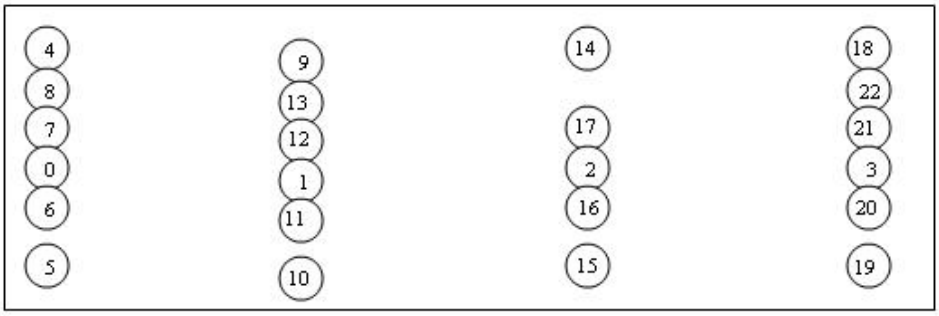

Fig. 3. Network Topology reproduced from [5]

Scenario Assumption:
The following NCs are instantiated: TCP-AC an AOC for handling TCP logic, L-RC an ROC for wireless link adaptation, P-RC, an ROC for controlling power, R-CC a COC for handling routing and forwarding, and FC a FOC for federating all previously mentioned specialized cells
The Scenario Steps
Numbers refer to steps taken by legacy protocols bullets refer to steps taken by the CellNet specialized cells instantiated on the nodes

1. Node 0 powers up

- FC goal is to provide highest throughput while minimizing power consumption. Let R be the power consumption rate of the link, I be the interference realized, E be the bit error rate attained and T be the throughput achieved, then FC optimization goal can be represented as MIN (R, I, E) and MAX (T). Both R and E can be represented as a scalar value, I can be abstracted in terms of collisions represented as the size of the MAC contention window (CW), and T needs the input of end application in terms of the goodput
- On initialization the P-RC will assign each physical resource on the device a maximum rate for power consumption Rmax which should not be exceeded. Let the rate assigned for the link be LRmax
- The L-RC is responsible to control the PHY layer operation (signal strength) such that the power rate consumptions does not exceed LRmax while achieving required threshold for BER (through adjusting SNR)
- FC requests L-RC to load code for power adaptation and report on BER and contention level
- L-RC instructs PHY to detect the minimum power required to make 1 hop connection to a maximum of 2 neighbors [6]. In case no neighbors are detected then power level is increased and the discovery

process is repeated. If power level used hits Pmax (Pmax defined in the thresholds used by the RU) then L-RC assumes that the node is isolated and reports that no connection is available

- Neighbor list is constructed and connection cost is measured in terms of average transmission power required to reach each neighbor

2. Node 0 initiates a TCP connection stating node 3 as the destination.

- TCP-AC on 0 realizes the need for a reliable transmission for the connection
- TCP-AC reports to FC on QoS requirements (reliable transmission, delay tolerant)
- FC recommends that the TCP-AC loads updated code for TCP to operate on wireless links (TCC proposed in [5]) to include consideration contention level along the path and requests TCP-AC to monitor and report on RTT, congestion window, packet error ratw and goodthroughput (these values are already calculated in the TCC protocol)
- Within TCP-AC, RU consults its own KB and communicates, with MU and EU to apply FC recommendations and any other self inferred recommendations (In case contradiction occurs FC recommendations have a higher priority than RU self recommendations)
- TCP-AC on 0 requests route discovery form R-CC to destination 3
- FC requests R-CC to gather path parameters in terms of contention and BER and report on values perceived on node 0

3. AODV initiates route discovery by sending RREQ for destination 3 (In this scenario intermediate replies are prohibited)

- R-CC uses the neighbor list created by L-RC listing 1 hop neighbors
- R-CC record path parameters in terms of contention level and BER in the data exchange unit (DXU) to be sent along the path and exchanged among R-CCs along the path to destination
- Along the path each R-CC will query the respective L-RC for contention level and BER then updates the respective values in the received DXU. The highest values for contention level and BER will be the values reported for path parameters

4. Node 3 will reply with RREP for each RREQ received

- With each returning RREP the path state in terms of highest contention value and BER will be obtained
- R-CC along the path will instruct the AODV to use the path with the least contention and BER even if it is not the shortest path (minimum power transmission is already enforced since the neighbor list is obtained from the L-RC)
- R-CC reports path parameters to FC in terms of expected contention and BER and also reports whether a federation is active on the destination node or not
- FC decides on the best way to handle the flow in terms of fragmentation, FEC etc… and sends recommendation to RUs of TCP-AC and L-RC

5. TCP flow starts (In case there is a FC on 3 steps b & c will be performed through FC to FC communication)

- TCP-AC marks the flow to require a reliable service
- TCP-AC on 0 sends the expected path quality to TCP-AC to be initiated on 3
- TCP-AC on 0 indicates the changes to be made to TCP code on 3
- L-RC along the path instructs LL not to discard packets requesting reliable service

6. First TCP segment reaches 3

- TCP-AC on 3 reads path parameters as indicated by TCP-AC on 0 and loads updates required for TCP

7. During the TCP connection

- TCP-AC on 0 reports on the RTT, congestion window, packet error frame and goodput achieved to FC
- L-RC reports on the BER and contention level seen on the node
- According to the optimization equation employed by FC the decision will be taken as whether to increase power transmission to combat BER and provide shorter routes to enhance throughput

8. Related Work

Limitations induced by the layered architecture have been the focus of several research proposals. These proposals can be classified into two main categories; the first aimed to preserve layering with some modification to allow layer interaction, while the other adopted a clean slate design. Cross Layer Designs

are the most prominent example for the former [7]. CLD were applied mainly to wireless networks to enhance network awareness of performance conditions. For clean slate design proposals, these provided radical changes to the network legacy stack. Their main feature is to allow fine grained composition of protocols. An example would be the ANA project aiming to introduce autonomic behavior to the network [8].

9. Conclusion

The design of computer networks needs to account for evolution. Inspired by natural complex systems, we proposed composing the computer network out of Network Cells (NC) capable of evolving through continuous learning and adaptation. We also proposed the CellNet framework within which the NC functions and interactions are defined. CellNet framework is derived based on the concept of separation of concerns that tackle complexity by dividing the problem space into smaller manageable, loosely-coupled, and easier to solve sub-problems allowing for better understanding and design. The CellNet framework abstracts network functions into three main core concerns and one crosscutting concern. These identified abstractions will be the reference model for specializing the NC. Our future work will encompass formalizing a new design methodology that we refer to as Cell Oriented Architecture on which the NC and the CellNet framework are based. We intend to construct a network prototype composed of NC using the CellNet framework to study the performance gains achieved when introducing learning and adaptation to networks.

Reference

1. ‘THINK’ protocols, ARPANET” http://www.cs.utexas.edu/users/chris/think/ARPANET/Timeline/#1958
2. Highlights of JF Koh's Honours Programme, 1997, http://wwwmcc.murdoch.edu.au/ReadingRoom/VID/jfk/timeline.htm
3. Melanie Mitchell, "Complex Systems: Network Thinking,"Preprint submitted to Elsevier Science, 8 September 2006 http://web.cecs.pdx.edu/~mm/AIJ2006.pdf
4. E. Ben-Jacob, "Social behavior of bacteria: from physics to complex organization,"The European Physical Journal B - Condensed Matter and Complex Systems,Volume 65, Number 3 / October, 2008
5. Ehsan Hamadani, Veselin Rakocevic, "A Cross layer Analysis of TCP Instability in Multihop Ad hoc Networks,"Journal of Internet Engineering, Vol 2, No 1 (2008). http://www.jie-online.org/ojs/index.php/jie/article/view/41
6. Vikas Kawadia,P. R. Kumar, "A Cautionary Perspective on Cross-Layer Design," IEEE Wireless Communications February 2005
7. Vineet Srivastava,Mehul Motani, "Cross-Layer Design: A Survey and the Road Ahead,"IEEE Communications Magazine December 2005
8. D1.4/5/6v1: ANA Blueprint – First Version. Workpackage 1 Deliverable 1.4/5/6v1, ANA Project FP6-IST-27489, February 2007