Chapter 1

Motivation and Related Work

1.1 Introduction

A wireless network provides mobility and cable free connectivity, which became a bottleneck in wired networks. In various scenarios, wireless network is the only choice for network connectivity. For example, installing wired connection across the river bank or free way or tactical areas is more difficult than installing wireless infrastructure. Sometimes government polices may not allow cabling across the required locations. This may create problem in providing network connectivity across the given region. Further, maintaining wired infrastructure is costly and time consuming than wireless networks. Any damage or fault in cable causes higher downtime in a wired network.

Mobility enables network users to move physically while using an appliance, such as handheld personal computers or data collectors. Many network based applications require mobility for end devices. These include applications like warehouse inventory, surveillance applications, healthcare applications, applications that deal with natural calamity, retail shopping and high speed vehicular networks. These applications became possible due to the seamless mobility provided by the wireless networks.

Using wired networks, it is difficult to collect information from places like hostile environment, enemy field, highly populated urban areas or far distant rural areas, moving vehicles, elderly care systems or volcanic areas. Wireless networks overcome the prob-
1.1 Introduction

Wireless network has many benefits for having last hop as well as multi-hop wireless connectivity. This has opened new areas and applications like Wireless Sensor Networks, Body Sensor Networks, Wireless Ad-Hoc Network, Mobile Ad Hoc Networks and Vehicular Networks etc. Closely deployed wireless sensors help in understanding the behavior of volcanic eruption, environment of tropical forests and behavior of endangered species. A widely deployed surveillance system makes safer surroundings. Vehicular network helps in solving problem of busy highway and free way. It also navigates to reach unknown destinations. A continuous health monitoring system provides crucial information about patient’s health that helps doctors to arrive at proper diagnosis. These emerging applications have increased the comfort level of human life and made our life more comfortable and healthier. By breaking barrier of wired connectivity, wireless network has impacted our day to day life to a greater extent.

Besides all its benefits, a wireless network has its own problems like low throughput, higher security risks, higher link error rate, broadcast nature of communication, higher energy utilization, licensing problems, problem of multiple operating standards etc. Normally wireless networks are built using Open Systems Interconnect(OSI) standard. These problems are further classified based on OSI layers. At physical layer, wireless network has challenges like RF spectrum issues, wireless channel characteristics, fading, power control, co-channel interference, multiple operating standards etc. At Medium Access Layer, problems like collision and inefficient use of channel are of concern. Other major problems that wireless MAC faces are energy efficiency, fairness and quality of service. A wireless network frequently faces topological changes which affects the functioning of network layer. Multicasting, scalability, security, aggregation and node cooperation are some of the problems faced by the routing layer of wireless networks. Behavior of transport layer is highly dynamic in wireless networks. Transport layer problems like maintaining end-to-end semantics, handling of handoff and distinguishing losses between congestion and link error become more critical in wireless networks.
1.1 Introduction

Normally wireless networks are built around the OSI layered architecture. This highly successful layered architecture, does not help much in overcoming many of the wireless network problems discussed above. At the time during which OSI architecture was designed, majority of the networks were deployed as wired networks and communication standards were yet to be defined. Considering this, OSI layered architecture is primarily designed focusing on modularity as its key design principle. Here, layers are communicating with each other through standard interfaces. Any changes made to one layer does not affect other layers as long as their interface remains unchanged. OSI layered architecture is highly successful architecture in wired networks where inter-dependency between layers is less compared to wireless networks.

Wireless network gives low performance in terms of throughput compared to wired network, when built as per OSI layered architecture. One of the best examples that highlights the problem of OSI layered architecture is Transmission Control Protocol (TCP) working on wireless networks. TCP protocol changes its congestion window size when it encounters packet loss. It considers packet loss as outcome of congestion, so it considers each packet loss as indicator of possible congestion. That is valid for wired networks where majority of packet loss is due to congestion rather than link errors. TCP considers channel error as indicator of congestion and reduces its window size that reduces the actual throughput of the network.

TCP over wireless network is only one of the examples that highlight the limitation of OSI layered architecture. The list of such limitations is long, but the root cause of many of the problems is lack of information at the decision making point. For an example TCP congestion window controller does not have clear information about the packet loss and that lack of information ends up in wrong decision and low network throughput. To some extent, this lack of information can be solved by bringing necessary information at the decision making point. But the fundamental design philosophy of OSI layer architecture does not allow one to do so. One can achieve this by violating the OSI layered architecture through introducing cross layer interactions.
1.2 Cross Layer Interactions

Cross layer design has emerged as a fascinating new area of research in wireless networks. Protocol designed by violating referenced layered architecture is called cross-layered design [11]. With these added details, it is now possible to make informed decisions about protocol functioning and optimization.

It has been shown that traditional legacy protocols, with disjoint layers, provide a measure of security. However, they are not suitable for the wireless networks as they do not sufficiently capture its dynamic nature. By exploiting cross-layer interactions, one can solve some of the difficulties inherent in wireless networks. This section covers the factors that motivate the cross-layer design [12] and current practices [13, 14] and problems with cross layer interactions [15] in wireless networks.

1.2.1 Current practice

Cross layer interactions are designed to achieve certain performance objectives. These performance objectives of wireless network can be categorized as follows:

- To satisfy QoS requirements.
- To improve resource efficiency.
- To optimize system connectivity to improve system scalability.

Efficient power utilization is highly required in wireless networks, as it has a direct and profound impact on the lifetime of the network [16, 17]. The key objective is to institute a power aware protocol that takes into consideration the Residual Energy Information (REI) of the node. Designing routing and data link protocols considering the Channel State Information (CSI) between nodes is one of the example of it. Another example is adjusting transmission range in response to changing network topology.

With directional antenna, the normal layered architecture gives suboptimal results and does not completely utilize the capability of directional antenna [18]. Target tracking is a known wireless sensor network application. Liang et. al. in [19] used cross layer
1.2 Cross Layer Interactions

approach in which they had excluded the transport and network layer. Their proposed cross layer approach between application and MAC layer has built efficient target tracking application which considers QoS. Here application layer directly talks to MAC and Physical layer.

Gentian et. al. in [20] shows how cross layer design helps in improving performance of virtual MISO (Multiple Input Single Output) model. Their result shows up to 60 percent improvement compared to SISO (Single Input Single Output) channel. B Hamdaoui et. al in [21] has proposed cross layer approach for admission control and flow level control for multi antenna system. Using MAC and PHY layer, author has increased end-to-end flow acceptance rate and network throughput utilization. Javier et. al in [22] have solved key distribution problem using cross layer framework. Yingqun Yu et. al in [23] have solved the problem of transport layer rate control with MAC layer channel access.

Excessive retransmission and hidden terminal problems are hard to solve in IEEE 802.11. References in literature highlight this problem. In [24] the author has addressed the excessive retransmission problem and showed that it is NP hard to solve and gave heuristic solution for it. Cross layer design helps in improving this scenario. Controlling physical layer transmission power based on constellation size, packet arrival rate, QoS requirements, power limitations and channel condition are few other well known examples of cross layer interactions.

Above discussed were few of the available cross layer protocols in the literature. Further details on cross layer protocols can be found in [13, 14]. These protocols fall in one of the cross layer interaction categories discussed in following section. These categories are designed considering the way information flows among the layers. That is how the required information is provided to the required layer.

1.2.2 Classification

Cross-layer design for wireless networks can be broadly classified based on it’s treatment to layered architecture. Some of the proposed protocols preserve underlying layered structure while others compromise with layered structure. Those protocols which preserve
1.2 Cross Layer Interactions

Figure 1.1: Information flow in Information Sharing Cross Layer Interactions

Information sharing class of cross-layer protocols preserve the underlying layered architecture. Layers in layered architecture are only allowed to communicate with layer below it or layer above it. Information sharing violates this constraint. It provides interfaces to the layers other than the neighboring layers. Here, the given layer opens interfaces for any layer of the layered architecture. Based on the type of information flow in the interface, information sharing can be divided into three categories: forward interface, backward interface and forward-backward interface as shown in figure 1.1.

If information flows only in one direction then based on its flow direction it is either considered as backward interface 1.1(a) or forward interface 1.1(b). Information flowing from upper layer to lower layer is referred as forward information flow and information flowing from lower layer to upper layer is referred as backward information flow. Interface is called forward-backward interface 1.1(c) if a layer opens information flow for a layer below and a layer above it in the layered structure. These layers may be non-neighboring layers. Information flows in both the directions from upper layer to lower layer and from
lower layer to upper layer.

These architectures are client server architectures, here server provides the information and client has cache in which it stores the information for decision making. Server only provides information to the client and does not track what information it has provided to the client. In the forward interface and backward interface protocols, flows are independent with cache placed at information receiving side. But in forward-backward interface, layers are not independent as cache is at both the sides. Content of this cache depends on the decision made by the layer. Decision of layer may change the content of its cache. Implementation details of both the layers are transparent to each other. That makes forward-backward architecture hard to handle.

**Design Coupling**

Information sharing protocols compromise the architecture only in terms of additional interfaces. By means of additional interfaces, it shares the information with non-neighboring layers. These interfaces are query type interfaces without having any side effect on referred layers. Design coupling is step forward in this direction. Here the communicating layers not only share their information but they also modify parameters of other layers. Design coupling interfaces can be divided into following categories as shown in figure 1.2: coupling without interface 1.2(a), merging of layers 1.2(b), shared database 1.2(c) and direct communication 1.2(d).

In coupling without interface, targeted layers are strictly designed to work in collaboration. At the design time, layers presume certain properties of the targeted layers and make design decisions based on those assumptions. Though their communicating interfaces remain untouched, any design modification on one layer affects the working of other layers.

To optimize the performance, layer designers have merged the working of layers and have come up with the compromised layered architecture. Merged layers provide same interface to other layers as they were providing previously. But the interface between the merged layers vanishes. Layers above or below the merged layer work with the same
1.2 Cross Layer Interactions

(a) Without Interface

(b) Merged Layer

(c) Shared Database

(d) Direct Communication

Figure 1.2: Information flow in Design Coupling Cross Layer Interaction
interface. Mainly merging of layers occurs for performance tuning and optimization. Merging of layers occurs between the neighboring layers.

In shared database, database works as common cache between the layers. Each layer that participates in this sharing can update the common cache or consume the information provided by the common cache. In direct communication a given layer can access any other layer without requiring any flow. It can treat variables of other layers as local variables for optimizing the performance of certain parameters.

1.2.3 Problems in cross layer interactions

As we saw previously that cross layer interactions are practiced mainly between PHY and MAC, PHY and NET, PHY and TRAN, APP and TRAN or between PHY, MAC and NET layers. Designing cross layer interaction is a non-trivial task. The system gets more complicated when multiple cross layer interactions take place simultaneously. Here with some care, one gets improved performance from cross layer implementation. But multiple cross layer implementations create a system which is complicated and hard to manage and track. Further uncontrolled cross layer interaction creates the problem of instability and puts additional overhead on system functioning. It suggests that cross layer interaction is not sufficient for an efficient and stable system.

Cross layer interaction on its own is not sufficient, it requires mechanism or framework which makes the stable and manageable cross layer interactions [15]. Cross Layer Architecture talks about overall system view or mechanism that makes the system adaptable. With Cross layer architecture it becomes possible to work with different cross layer implementations simultaneously.

1.3 Cross Layer Architecture

Development of a complex system is a difficult task. Here difficulty comes from the size of the system and the complex interaction that take place within the system. Certain design patterns are useful to overcome such system complexity. An architecture breaks down
1.3 Cross Layer Architecture

Cross Layer Architecture is one of the design patterns created to manage cross layer interactions. The complexity of the system is managed by applying abstractions and encapsulations.

1.3.1 Abstract view of network protocol stack

A protocol belonging to the network protocol stack can be viewed as a combination of two parts: communication part and control part. Communication part moves information from one end to another end while control part governs this information flow.

For example, routing protocol at network layer has two main functions. First it generates the routing table. Second, based on the generated routing table it forwards the packets. Here, the communication part forwards the data packets and the control part generates the routing table. Same is the case with the other layers and protocols. In general, the entire network protocol stack can be viewed as combination of control and communication parts. We call this primary control and communication part.

In layered architecture, communication parts and control parts are confined to one layer or one protocol. With Cross Layer Interaction (CLI) the scenario changes. CLI creates additional information or control flow in the architecture. Here, one layer shares its information with other layers or control part of one layer works with control part of other layers. We refer to this as secondary control part. Objective is to increase...
the performance of network protocol stack by introducing secondary controls (CLIs) in the stack. Figure 1.3 shows an abstract architecture of a network protocol stack having multiple CLIs.

With cross layer interactions the network protocol stack is divided into three parts. One is primary communication part and other two are primary and secondary control parts. Here, secondary control part contains cross layer mechanism implemented in the layered architecture. With introduction of secondary control part it raises two questions. One is about the placement of secondary control part and the other is about how to bring the required information to secondary control part.

Primary control part resides at one place and all its required information is bounded to one layer or one protocol. But as secondary control part deals with more than one protocol or layer, question arises regarding its placement. Either to put this additional control as part of one layer or in between two layer or as separate component interacting with layers. Secondly, primary control part gets its information from the layer itself. But in the case of cross layer interaction the secondary control part may require information which is not available at the current layer or one place. Here the key problem is how to bring the required information to secondary part.

Any architecture that is designed to deal with the cross layer interaction must address following two questions. A). Where to put the secondary control part? and B.) How to bring required information to secondary control part? In this light we will analyze the available cross layer architectures.

1.3.2 Literature

The objective of cross layer architecture is to accurately model, incorporate and leverage the secondary controls in the system. In literature, a majority of work done focuses on exploring the possibility of different secondary controls which enhance the system performance. Very few address the problem of incorporating secondary control in the system. Those which are available can be classified as revolutionary or evolutionary cross layer architectures.
1.3 Cross Layer Architecture

Revolutionary architectures are specifically designed targeting Cross Layer Interactions (CLI) and completely ignore the layered design. TinyCubus\[25\] is a well known example of a revolutionary architecture. An architecture which extends the layered design for CLI is called evolutionary architecture. MobileMAN\[1\], CrossTalk\[2\], ECLAIR\[3, 26\] and RCL\[4\] are well known examples of evolutionary architectures.

Shakkottai et. al \[27\] have talked about some early efforts regarding standardized cross layer implementation. Here they are mainly concerned with cross layer direction for the 3G network. This point was raised in early development of wireless networks but still no adequate solution is present in the literature for this.

Chang et al. \[28\] proposed an architecture for multiple cross layer interaction to support QoS requirements. A piggybacked signaling is used with XML for internal messaging. The paper highlights important issues that one has to deal with when working with multiple cross layer interactions. Faulty assumption, shared database, dependency loops and overhead are the key issues focused by them. Their proposed solution is centralized solution implemented between network and MAC layer as control middleware. Information is collected by piggybacking with the packet. Higher layer puts the cross layer information at the end of the packet which the centralized horizontal layer receives to implement optimized algorithm. Here protocols have only considered the top down information flow in the network. The architecture does not support the implementation of cross layer protocols that work with bottom up information flow. Further, the architecture does not guarantee for providing information to middleware. Due to buffer overflow at network layer if packet gets dropped then the centralized middleware misses the required information.

TinyCubus\[25\] is a revolutionary architecture specifically designed targeting the Tiny OS\[29, 30\] operating system. This architecture consists of three components: Cross Layer Framework (CLF), Data Management Framework (DMF) and Configuration Engine (CE). DMF maps a) applications requirements with b) system parameters considering c) optimization parameters. Configuration of these three kinds of parameters generates cubus in TinyCubus. In TinyCubus, it is difficult to perform synchronized
1.3 Cross Layer Architecture

reconfiguration of these components. Further, TinyOS does not have rich support for interfaces and callbacks. Maintaining callbacks and interfaces between TinyOS based components generates additional overhead and increases the overall system complexity in TinyCubus.

MobileMAN[1] is an evolutionary architecture. For cross layer communication it manages one vertical layer called Network Status. Layers share their information for other layers through Network Status layer. MobileMAN changes layers to implement cross layer interaction. Modified layer gets its required information from Network Status. Any update in design of Network Status triggers the changes in all the dependent layers. Here, Cross layer interactions are implemented by modifying layer itself that increases the development time and makes it difficult to introduce multiple cross layer interactions in the system. Further it is hard to recover from existing cross layer interactions.

Choi et. al [31] proposes taxonomy for cross layer architecture design and based on that they have proposed cross layer framework. Their taxonomy defines how information flows, the adaptation methods and the delivery methods. Taxonomy defines the working of cross layer implementations. It provides abstract ways to implement cross layer interaction without going into specific case. But the taxonomy and taxonomy based cross layer modules do not define how the various cross layer interactions take place with each other. It provides abstract view of cross layer interaction but it does not provide sufficient information for cross layer architecture.


Zhijiang et. al [28] proposed the cross layer architecture using the concept of middleware. To manage cross layer interactions, their architecture adds one layer between the MAC layer and network layer. Here the added middleware layer gets its required informa-
1.3 Cross Layer Architecture

...tion in the form of command and data piggybacked to packets passing across the network protocol stack. The added layer manages the interaction occurring between the different cross layers using suppress and resume signals. Their proposed architecture considers the bottom up information flow only and ignores the top down information flow. Piggybacking of cross layer information to the packet modifies the protocol message structure and creates problem of decoding and encoding of messages. Here, middleware only provides the cross layer information while the cross layer interaction is implemented in the layer itself that creates the additional overhead to the protocol stack.

Another proposed framework \cite{32} is agent based framework. Here cross layer optimization is achieved by optimized agents (OA). In their work they have classified the interaction between layers as inter-layer interaction and intra-layer interaction. Inter-layer interaction happens between non-adjacent layers and intra-layer interaction happens between adjacent layers. Using information gathered by interactions each layer implements the optimization steps using bottom-up or top-down optimization agents. Their framework preserves the layered architecture and implements optimization agents by extending layers with interaction information. The main drawback of this framework is that bottom-up agents and top-down agents work independently. This may cause stability and correctness problem in proposed solution.

ECLAIR\cite{3} is another cross layer architecture. Features of ECLAIR\cite{3} makes it an attractive choice for cross layer implementation. It has small footprint on layered architecture. ECLAIR is flexible and modular architecture in which each cross layer interaction is implemented using separate optimizer. Separate optimizer increases the messaging overhead in the system. Further multiple optimizers in ECLAIR create the problem of feedback loop and increase the possibility of conflict. Few other architectures available in literature are \cite{33, 34, 35, 36}.

TinyCubus has difficulty in managing CLIs with changing modules. MobileMAN and CrossTalk have problems of longer development time and difficulty in recovering from CLI. ECLAIR struggles with the feedback loop in cross layer implementation. To the best of our knowledge, available cross layer architectures in the literature do not provide
1.3 Cross Layer Architecture

efficient solution for the following CLI problems: accidental occurrence of feedback loops in the system, longer development time, recovery from cross layer interactions, small footprint on referred architecture and multiple cross layer interactions.

1.3.3 Design goal of cross layer architecture

Architecture plays key role in implementation of cross layer interactions. A well organized architecture should provide stable platform to experiment various cross layer interactions. Current state of the art shows that very few attempts have been made on the design of cross layer architectures. And the available architectures do not address the key issues of cross layer interactions.

In TinyCubus, MobileMAN and CrossTalk, communication and control parts are tightly coupled together. This tight binding makes the system complex and hard to track when multiple CLIs take place in the system. Once the CLI is implemented in the system any modification made to it regenerates the entire implementation cycle. This problem is there in many of the previously proposed architectures. TinyCubus, MobileMAN, CrossTalk and ECLAIR have distributed secondary control flow for different CLIs. This distributed control creates the problem of instability and feedback loop in the system. CrossTalk architecture modifies existing messaging structure by adding piggyback information. That decreases the stability of the architecture and the protocol. Restoring system to its previous state is difficult with current cross layer architectures.

Study of cross layer interaction and cross layer architecture shows that an architecture designed to handle cross layer interactions should provide the following features:

- It should provide smooth process for creating, updating and extending a cross layer interaction.
- It should handle multiple cross layer interactions.
- Easy to recover from implemented cross layer interactions with minimum effort.
- It should create small footprint on existing architecture.
1.4 Summary

- It should take care of the possible conflict in the system.
- It should take care of the accidental feedback loops in the system.

In thesis, we propose cross layer architecture CLAPDAWN which addresses the above discussed CLA problems. CLAPDAWN is an evolutionary architecture. As mentioned earlier, the evolutionary CLAs existing in literature are designed targeting layered architectures. Our modular architecture is not restricted to layered architecture and can be used to extend existing network protocol architectures. Further details about proposed architecture is given in chapter 2.

1.4 Summary

Wireless network provides a technical platform for the emerging applications. Besides its benefits, it comes with the new problems. We have highlighted benefits and problems experienced by the wireless networks in this chapter. Root cause of many of these problems is lack of information at the decision making points. Cross layer interactions helps in that. The chapter discussed importance of the cross layer interactions and problems encountered in dealing with them. Here, we discussed the needs of the cross layer architecture, limitations of existing architectures and desired features from the cross layer architecture. Based on these constructive arguments we have proposed our cross layer architecture in next chapter.