CLAPDAWN: Cross Layer Architecture for Protocol Design in A Wireless Network

by

Sunil Jardosh

(200421001)

Under the guidance of

Prof. Prabhat Ranjan

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

Doctor of Philosophy

in

Information and Communication Technology

to

Dhirubhai Ambani Institute of Information and Communication Technology

DEC 2010

Declaration

This is to certify that

1. the thesis comprises my original work towards the degree of Doctor of Philosophy in Information and Communication Technology at DA-IICT and has not been submitted elsewhere for a degree,

2. due acknowledgment has been made in the text to all other material used.

Signature of Student

Certificate

This is to certify that the thesis work entitled "CLAPDAWN: Cross Layer Architecture for Protocol Design in A Wireless Network " has been carried out by Sunil Jardosh (200421001) for the degree of Doctor of Philosophy in Information and Communication Technology at this Institute under my supervision.

Thesis Supervisor Prof. Prabhat Ranjan

Acknowledgments

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Prabhat Ranjan, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. He has always provided me with the required resources and guidance, without which this research could not have been possible. He has always supported me in various academic and administrative complexities and other difficulties that are part of student life. His experimental approach and applicable results oriented nature have created deep impression in my mind, and will continue to inspire me for the rest of the life.

I thank Prof. Sanjay Srivastava for providing excellent guidance by means of courses and research discussions on computer networks. He has devoted significant amount of his valuable time to plan and discuss my work. I thank Prof. V. Sunitha and Prof. Rahul Muthu for the technical discussions on network modeling. I thank Prof. Sanjay Chaudhary for his encouragements and motivation through out my stay at DA-IICT. I am thankful to all the DA-IICT faculty members who have guided, mentored or supported me by acting on various academic committees.

I sincerely thank the external reviewers of my thesis. In spite of busy professional schedule, they have accepted the reviewing request. I acknowledge the DA-IICT Resource Center for ensuring best library and reference services that have played vital role in my work.

I acknowledge my fellow lab mates in Modeling and Analysis Group of NeTworks (MAGNeT) Kamal Mistry, Narmawala Zunnun, Parth Rao, Sapan Shah, Brijesh Patel, Kavan Seth, Manish Chaturvedi, Rahul Shah and Pankesh Patel for stimulating discussions, for the sleepless nights we were working together before the submission deadlines and all the fun we have had at the lab. I am grateful to all PhD students at DA-IICT. Especially, I acknowledge Shubham Jain for his support. Vikram Sorathiya, Diwyang Rawal, Pratik Shah, Ratnik Gandhi, Gauri Joshi, Nilesh Banker, Purhsotaman Anatkrishanana, Ghanshaym Paramar, Jignesh Bhavshar, Bhavesh Dharmani and many other current and past PhD students of DA-IICT have made my stay a memorable experience.

Finally, I am forever indebted to my parents, Chetan and Sonal for their understanding, endless patience and encouragement when it was most required.

Abstract

Architecture plays a key role in overall success of the network protocol stack. It is essential to have robust architecture for the complex system having multiple cross protocol interactions. In the literature more emphasis is given to the cross layer protocol design than the cross layer architecture. Cross Layer Architectures [1, 2, 3, 4] available in literature are not addressing the problems of cross layer interaction at their depth.

Our work focuses on cross layer architecture design that supports multiple cross layer protocols and cross layer interactions. It addresses the problem of lack of support for multiple cross layer interactions, feedback loops, a longer development time and smooth rollover from cross layer interaction. Our proposed Cross Layer Architecture for Protocol Design in A Wireless Network (CLAPDAWN) addresses these problems faced by many of the previously proposed architectures.

We have worked on two critical problems of IEEE 802.15.4 based event driven wireless sensor networks: topology control problem and prioritized event handling problem. The IEEE 802.15.4 standard configures network nodes as Reduced Function Devices (RFD) and Full Function Devices (FFD). In randomly deployed wireless sensor networks, the problem of configuring the network nodes as RFD or FFD, maintaining the network functionality is a topology control problem. To solve this, we have proposed a multipoint relay based Connected Dominating Set (CDS) construction algorithm.

In literature the topology control problem has been solved by centralized [5, 6, 6] and distributed [7, 8, 9] approaches. Dynamic nature of wireless sensor network makes centralized approaches less applicable and distributed approaches have the problem of overlapping and redundant nodes in the generated CDS. In our work we have proposed ROOT-Initiative (ROOT-I) and ROOT-Ik topology control algorithms to construct one-connected and k-connected networks respectively.

Proposed algorithms control the overlapping and redundant nodes in the network. They reduce the number of active nodes in the network that helps in extending the network lifetime. Further, the tradeoff between energy consumption and fault tolerability is analyzed for k-connectedness. Our results show that in randomly deployed dense networks, the algorithm has better approximation ratio with acceptable time and message complexity.

The solution for the prioritized event handling is based on IEEE 802.15.4 MAC scheduling and channel access mechanism. IEEE 802.15.4 has Guaranteed Time Slot(GTS) mechanism for guaranteed data delivery. The GTS mechanism is not sufficient to provide solutions for the critical event handling problem. It has problems of reservation delay and limited number of GTS slots. To handle critical events in IEEE 802.15.4 based networks, we have proposed Explicit Prioritized Channel Access Protocol(EPCAP) and Implicit Prioritized Channel Access Protocol(IPCAP) mechanisms. The proposed mechanisms achieve higher delivery ratio for important events with smaller delay. The IPCAP and EPCAP are modeled using Markov chain and M/G/c queuing model respectively and tested through simulation.

In ROOT-I and ROOT-Ik, network layer and MAC layer communicate to decide the node type (RFD or FFD) and in EPCAP and IPCAP prioritized events are handled by MAC based on information provided by application layer. As mentioned above CLAPDAWN is designed to support cross layer interactions in the system. To provide proof of concept for the CLAPDAWN we have modeled ROOT-I, ROOT-Ik, EPCAP and IPCAP as cross layer interactions in CLAPDAWN and we discuss how various components of CLAPDAWN communicate to implement these protocols.

Furthermore to provide comparative analysis we have implemented ECLAIR [3] cross layer architecture with the CLAPDAWN. We have implemented the complex video streaming cross layer protocol [10] on both the architectures and derived the performance results. To make more rigorous comparison of both the architectures we have implemented conflicting cross layer interactions in the system. Our results show that CLAPDAWN has relatively better performance results than the ECLAIR. That shows CLAPDAWN provides better and stable platform for cross layer protocol development.

Contents

List of Figures

List of Tables

List of Abbreviations

APP	**APP**lication layer
CDS	**C**onnected **D**ominating **S**et
CLE	**C**ross **L**ayer **E**ngine
CLAPDAWN	**C**ross **L**ayer **A**rchitecture for **P**rotocol **D**esign in **A**d hoc **W**ireless **N**etworks
CLA	**C**ross **L**ayer **A**rchitecture
CLI	**C**ross **L**ayer **I**nteraction
CLKB	**C**ross **L**ayer **K**nowledge **B**ase
CP	**C**ontrol **P**lane
CNI	**C**ontrol **N**etwork **I**nterface
CSI	**C**hannel **S**tate **I**nformation
CSMA-CA	**C**arrier **S**ense **M**ultiple **A**ccess - **C**ollision **A**voidance
ECA	**E**vent **C**ontrol **A**ction
ECLAIR	**E**fficient **C**ross **L**ayer **A**rchitecture
EPCAP	**E**xplicit **P**rioritized **C**hannel **A**ccess **P**rotocol
FP	**F**unctional **P**lane
FFD	**F**ull **F**unction **D**evices
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IPCAP	**I**mplicit **P**rioritized **C**hannel **A**ccess **P**rotocol
LNI	**L**ayer **N**otification **I**nterface
NET	**NET**work layer
PHY	**PHY**sical later
REI	**R**esidual **E**nergy **I**informaiton
RFD	**R**educe **F**unction **D**evices
ROOT-I	**R**oot **I**nitiative
ROOT-Ik	**R**oot **I**nitiative **k**
MAC	**M**edium **A**ccess **C**ontrol layer
MCDS	**M**inimum **C**onnected **D**ominating **S**et
MPR	**M**ulti **P**oint **R**elay
OSI	**O**pen **S**ystem **I**ntercommunication
TRAN	**TRAN**sport layer

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-

lem of fixed wired connectivity and provide mobility and flexibility to network devices. 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.

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 unchaged. 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 it's 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

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.

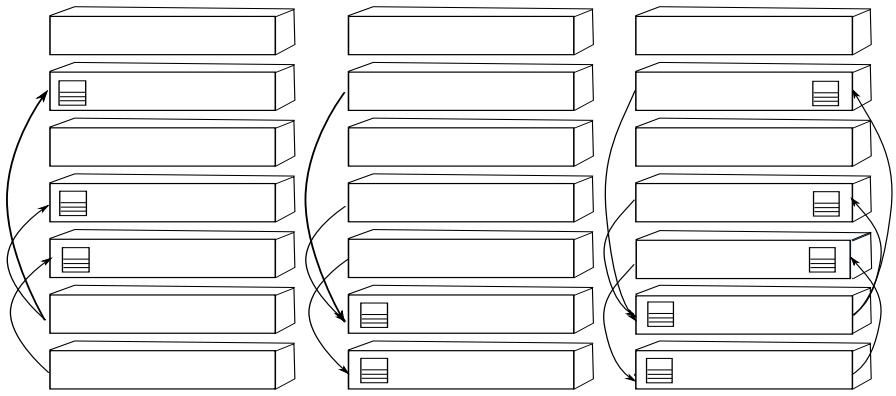
Gentian et. al. in [20] shows how cross layer design helps in improving performance of virtual MISO (Multiple Input Single Output)model. Their result shows up to 60 percent improvement compared to SISO (Single Input Single Output)channel. B Hamdaoui et. al in [21] has proposed cross layer approach for 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



(a) Backward (b) Forward (c) Forward-Backward

Figure 1.1: Information flow in Information Sharing Cross Layer Interactions

the layered structure are called information sharing cross layer interactions. Protocols which compromise with layered structure are called design coupling. Following subsection further refines this classification based on adopted techniques [11].

Information Sharing

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 in to 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.

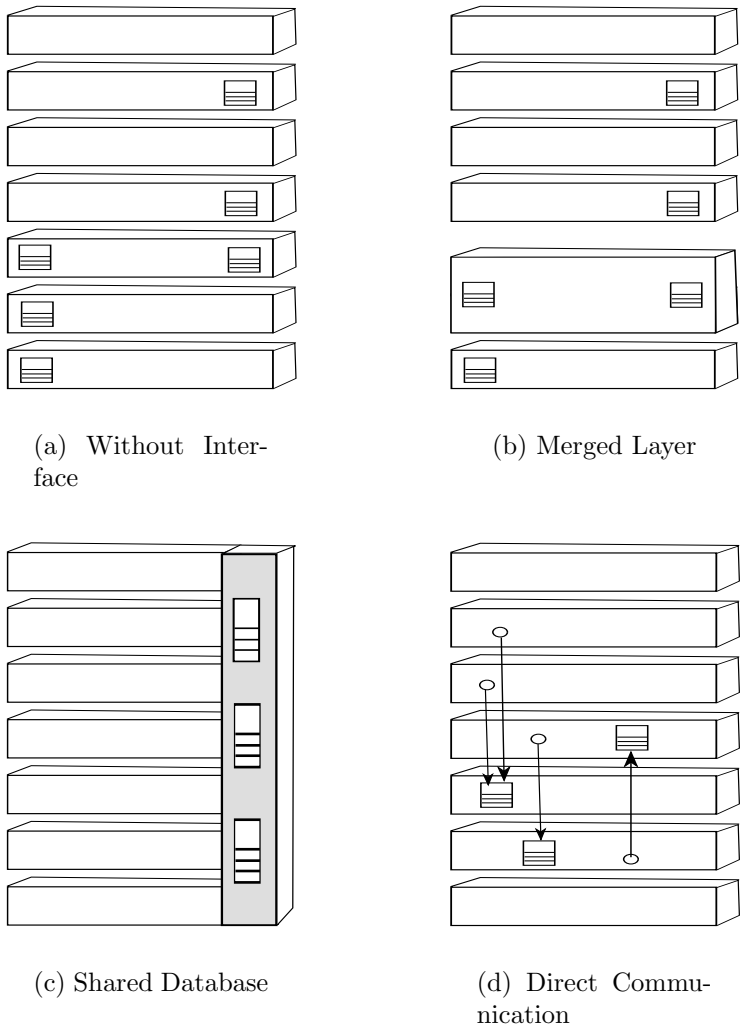
These architectures are client server architectures, here server provides the information and client has cache in which it stores the information for decision making. Server only provides information to the client and does not track what information it has provided to the client. In the forward interface 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



(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

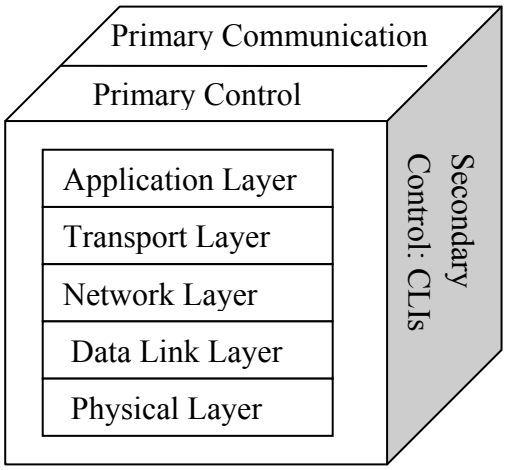


Figure 1.3: An abstract architecture of a network protocol stack with cross layer interactions

the complexity of the system by applying abstractions and encapsulations. Cross Layer Architecture is one of such design patterns created to manage cross layer interactions taking place in the system.

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 slightly. 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.

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

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.

CrossTalk follows the same design principles as MobileMAN[1]. CrossTalk [2] architecture maintains the local as well as global view for cross layer interaction. To maintain global view, CrossTalk adds global view information to the packet using piggybacking. Piggybacking modifies the messaging structure used in the network. RCL[4] uses a configuration table for cross layer interaction. Both the CrossTalk[2] and RCL[4] have same the limitations as MobileMAN[1].

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-

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 [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[3] is another cross layer architecture. Features of ECLAIR[3] makes it an attractive choice for cross layer implementation. It has small foot print 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 [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

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.

- 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.

Chapter 2

CLAPDAWN: Cross Layer Architecture for Protocol Design in A Wireless Network

2.1 Introduction

In Chapter 1 we presented a survey of existing cross layer architectures and discussed their limitations. We have also identified the design objective for a cross layer architecture, namely: feedback loops free system, smaller development time, easy to recover from cross layer interactions, small footprint on referred architecture and support for multiple cross layer interactions. In this chapter we present our cross layer architecture CLAPDAWN, which is designed based on the above design goals.

2.2 CLAPDAWN Overview

As we saw in previous chapter that it is hard to manage multiple cross layer interactions in the system. Unmanageable cross layer interactions trigger the stability problem in the system. Literature covered in section 1.3.2 shows that available architectures are not addressing these problems at their depth. To overcome the limitation of existing

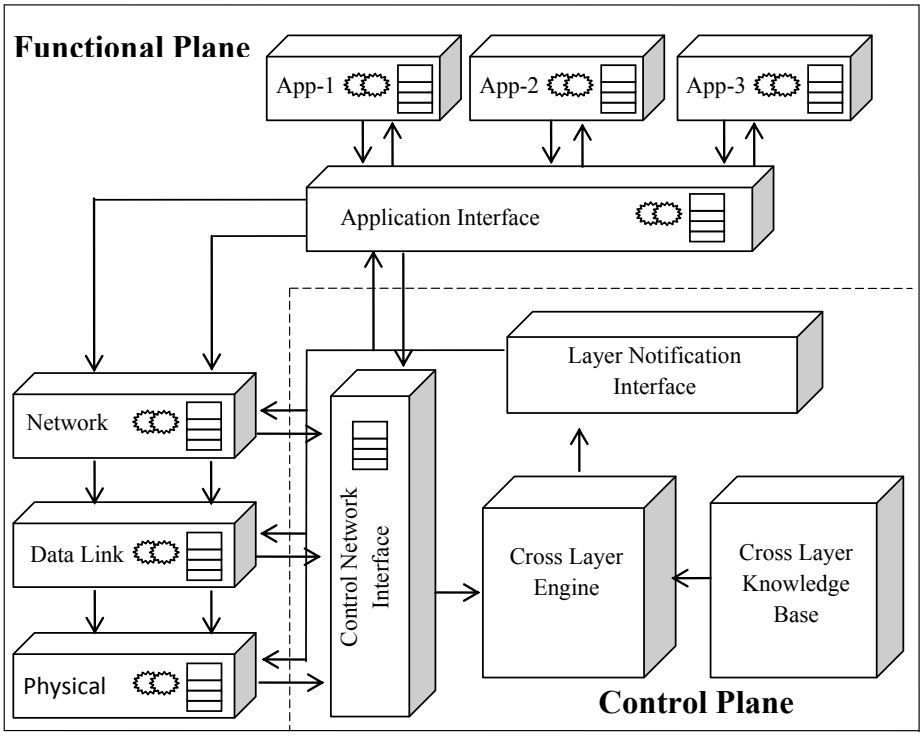


Figure 2.1: CLAPDAWN Architecture

cross layer architectures and to provide rapid implementation of cross layer interactions, we have proposed the cross layer architecture CLAPDAWN: Cross Layer Architecture for Protocol Design in A Wireless Network. CLAPDAWN is designed to provide manageable and extendable cross layer interaction in the system.

In the section 1.3.1 we have discussed the three components of network protocol architecture namely primary communication part, primary control part and secondary control part. CLAPDAWN manages these three parts using two planes, Functional Plane (FP) and Control Plane (CP) as shown in figure 9.4. In CLAPDAWN, primary communication part and primary control part create the functional body of the cross layer architecture and we grouped them as Functional Plane (FP). Control Plane works over functional plane and governs the functionality of it to implement the cross layer interactions. In CLAPDAWN Control Plane (CP) manages all cross layer interactions taking place in the system.

2.3 Functional Plane (FP)

Functional Plane contains all the functionalities required to develop network protocol stack. As we discussed in our previous chapter that with cross layer interaction network protocol stack can be viewed as combination of three parts, primary control and communication parts and secondary control part. From these three parts, FP consists of two parts: primary communication part and primary control part. FP is network protocol stack without having any cross layer interactions in it.

FP communicates with the control plane through interfaces provided by control plane discussed in next subsection. Figure 9.4 shows the FP having four layers physical layer, data link layer, network layer and application layer. Normally only one active protocol resides at network, MAC and physical layer. But there are cases in which more than one protocols run actively at the given layer. For example, there may be more than one application existing at the application layer. To take care of it, FP contains the application interface that manages different applications running at application layer. Same can be designed for other layers having multiple protocols.

In literature, existing Cross Layer Architectures(CLAs) either only focus on layered architecture or completely forget the highly successful layered architecture. Revolutionary architecture completely ignores the layered architecture and evolutionary architecture tries to maintain the layered structure. In both the cases, the architecture looses valuable quality of single solution or layered structure. CLAPDAWN separates primary control and primary communication from secondary control. That provides additional flexibility and portability to the CLAPDAWN and expands the scope of it. Because of this separation any existing architecture can be used as reference architecture in CLAPDAWN.

To make the representation simple, we have shown the FP with four layers. In general, FP may represent TCP/IP protocol stack, or Open System Interconnect (OSI) seven layer stack or any other existing architecture. We call this architecture as reference or referred architecture. In general CLAPDAWN adopts existing architecture as FP and facilitates it for cross layer implementation.

2.4 Control Plane (CP)

Control plane supports the functional plane and governs the functionality of functional plane to implement Cross Layer Interactions(CLIs). Control plane manages all the cross layer interactions or secondary controls that take place in the system. Control plane divides complex functioning of cross layer interaction into following subcomponents: Control Network Interface (CNI), Layer Notification Interface (LNI) , Cross Layer Engine (CLE) and Cross Layer Knowledge Base (CLKB). Figure 9.4 shows the organization of control plane in the CLAPDAWN. Cross layer engine and cross layer knowledge base implement the secondary control in control plane. Control plane gets its required information from functional plane using control network interface and provides the necessary updates to functional plane using layer notification interface. Control network interface and layer notification interface work as bridge between functional plane and control plane.

For simplicity here onwards we refer CLKB as knowledge base, CLE as execution engine, CNI as control interface and LNI as layer interface.

2.4.1 Cross layer knowledge base

Knowledge base stores all the secondary controls that take place in the system. CLAPDAWN separates logic of any given secondary control from its execution and stores it in the form of Event Control Action (ECA) rules in knowledge base. Different cross layer interactions may have different design objectives, but they all are modeled using standard ECA rule format.

ECA rules are triggered by an event. An event might be change in routing table, packet loss or buffer overflow. For a given event, ECA rule contains the control part. This control part checks the set of conditions before performing defined actions. Once all the conditions are satisfied it executes the action part associated with the event. Knowledge base stores different secondary controls using common ECA format as shown in figure 2.2.

Figure 2.2 shows two cross layer interactions. First interaction shows an event of

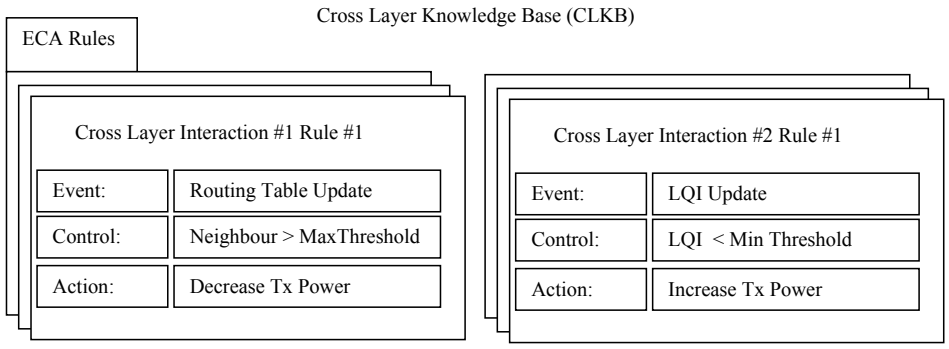


Figure 2.2: Cross Layer Knowledge Base (CLKB) with Cross Layer Interactions (CLI) stored in the form of Event Control Action (ECA) rules

network layer routing table updates. On routing table update event, it checks the control statement that whether the number of neighbors is greater than the defined maximum threshold. If number of neighbors is greater than the defined threshold then it performs the action part and decreases the physical layer transmission power. In second ECA rule, on an event of Link Quality Indicator(LQI) update, it checks whether LQI is below the allowable threshold. If LQI falls below the allowable threshold then it increases the transmission power to improve LQI.

With uniform representation of different cross layer interactions it becomes an easy task to check conflicting condition and feedback loops occurring between multiple cross layer interactions. Each event has input parameter and each action shows the affected parameters. Using group of ECA rules, CLAPDAWN generates the dependency graph. Dependency graph models the relation between different parameters and cross layer interactions.

A graph node in the dependency graph represents network parameter. The directed edge between two graph nodes represent the cross layer interaction. For the directed edge, starting graph node is an input parameter and destination node is the parameter affected by that cross layer interaction. Dependency graph makes it easy for CLAPDAWN to identify conflicting conditions and feedback loops in the system.

Multiple incoming edges from different nodes show multiple cross layer interactions affecting the same output parameter. That identifies the possible conflicts among CLIs.

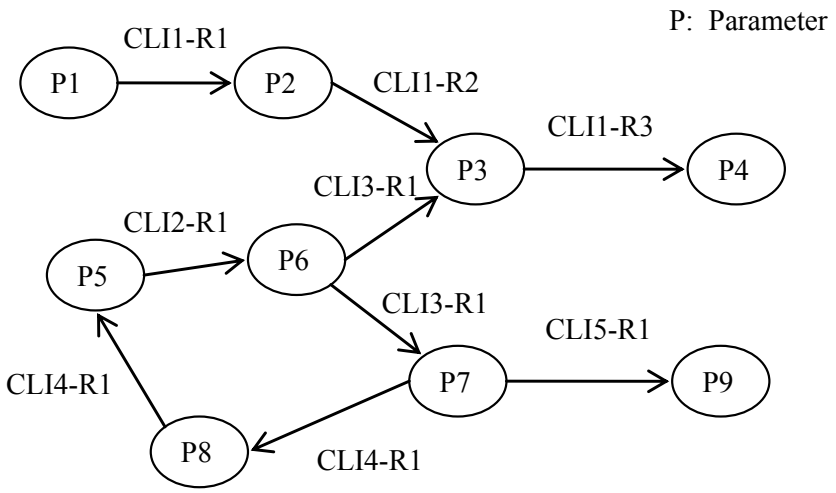


Figure 2.3: Dependency Graph of Cross Layer Interactions(CLIs) stored in Cross Layer Knowledge Base(CLKB)

Further, any loop in the graph highlights the feedback loop in the system. CLAPDAWN equipped with ECA rules in knowledge base provides a way to identify conflicting scenarios and feedback loops in the system in advance.

Figure 2.3 shows an example of a dependency graph of multiple cross layer interactions. In the given example we have nine parameters and five cross layer interactions. Each cross layer interaction may have more than one rule to execute. Here CLI 1 has three rules associated with it CLI1-R1, CLI1-R2 and CLI1-R3. CLI1-R1 is triggered by update in parameter 1 and it affects or modifies the value of parameter 2. Here a rule may update more than one parameters as shown in case of CLI3-R1. This rule updates more than one parameters, e.g. parameter 3 and parameter 7 in this example.

As mention above, in a dependency graph conflicting scenario takes place when there are multiple input edges for the given parameter. Same way feedback loops can be identified by finding loops in the dependency graph. In given example CLI 2, 3 and 4 create the feedback loop in the system and CLI 1 and 3 generate the conflict at parameter 3.

Populating ECA rules is easy compared to implementing cross layer interaction using some programming language. We have developed an editor that helps in writing ECA rules. The editor converts the rules into intermediate representation. This intermediate

representation is knowledge base programming language representation. Knowledge base programming language is easy to work with and complex interaction can directly be implemented using simple programming. Further details of ECA editor is provided in appendix A.

Here, cross layer interaction designer selects the event parameters, condition parameters and action associated with that event from the available list. The editor helps in defining the condition statements and action statements. Currently it is a menu driven editor. Editor is in its primary stage and able to craft simple cross layer interaction. Future objective is to provide graphical drag and drop editor to form complex ECA rules.

For given functional plane CLAPDAWN manages the token repository. Currently the token repository is generated manually considering the referenced FP. Based on this token repository rule, the editor provides the selection options to the CLI designer. If any change takes place in referred architecture in FP then knowledge base has to make some configuration chances for new FP. Knowledge base requires to update its repository for new FP. New cross layer interaction can be designed using new token repository.

Old cross layer interaction can work with the new FP. Here, it is the responsibility of knowledge base to maintain smooth functioning of secondary control. To use existing rules with changing FP, knowledge base requires to migrate current parameter set to new parameter set. That can be done my providing the XML mapping files. Knowledge base imports this mapping file and updates the parameter configuration accordingly.

Because of the standard ECA representation these updates are easy to perform. With ECA representation it is now easy to introduce new secondary control in the system. Uniform representation of different cross layer interactions makes execution simple. On occurrence of an event it is the responsibility of execution engine to execute the relevant ECA rules from knowledge base.

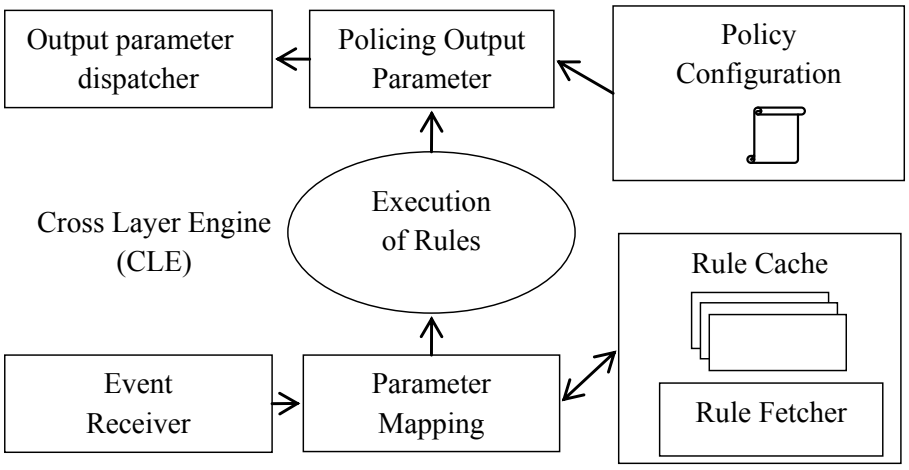


Figure 2.4: Cross Layer Engine(CLE) in CLAPDAWN with Policy configurations

2.4.2 Cross layer engine

CLAPDAWN separates cross layer interaction logic from its execution. It maintains the logic of cross layer interaction in knowledge base in the form of ECA rules. Based on event received from control interface, its the responsibility of execution engine to execute the relevant rules from knowledge base. Cross Layer Engine executes the required rules from the knowledge base and produces the updates from the FP which it conveys to layer interface.

Execution engine is the central entity in CP. It communicates with control interface by receiving events from control interface, with knowledge base for rules and with layer interface by sending messages. Execution engine performs the critical job of finding the rules, mapping them with parameters and executing them. Figure 2.4 shows the organization of execution engine.

Execution engine consists of the following components: event receiver, parameter mapper, rule fetcher, rule cache, policy configuration and parameter dispatcher. Event receiver receives events from control interface and passes the input parameter to the parameter mapper. Parameter mapper gets the relevant rules from rule cache. Rule fetcher gets required rules from knowledge base. By itself rules do not have any value associated with them. It is the responsibility of parameter mapper to attach appropriate

value to them. It stores the rule with mapped parameter values in the rule cache. Section 2.5.1 covers the detail example of it.

Execution engine executes the rules mapped with parameter and produces the results. These results are further processed by Output parameter checker using policy configuration. Following paragraph explains the policy configuration. Finally Output parameter dispatcher forwards the generated output parameters to layer interface.

Fetching rules from knowledge base and mapping it with the parameter requires configuration time. Repeating these steps for every event creates an overhead on the system. To reduce this overhead execution engine uses rule cache in which it stores the rules with the parameter mapping.

To fetch the rules from knowledge base, execution engine uses hashing on input parameters. Once execution engine gets its required rules. It checks the control statement. If it satisfies the condition then execution engine executes the action part of ECA rule. Execution of action part generates the output parameter which goes through policing.

Knowledge base identifies the possible conflicts in the system. It identifies the parameters and rules which generates conflicts and feedback loops in the system. To handle them CLAPDAWN provides policy mechanism. At the time of populating the knowledge base with ECA rules, knowledge base highlights possible conflicts and loops with CLIs with affected parameters. Further to resolve them, CP offers the option of policy configuration. Using which one can restrict the execution of the rules or change the way the final output parameter is updated. Policy configuration stores these policy options for affected parameters.

To mitigate the effect of conflict it provides the option of a) running average of output parameters, b) averaging output of all the conflicting rules, c) prioritize the conflicting rules or d) lock some of the conflicting rules. To remove the feedback loops in the system it provides option of locking. By locking one of the CLI in the loop it removes that CLI from the system which breaks the loop. Policy configuration reduces the effect of conflict and removes the feedback loops from the system. Uniform representation of rules and separating execution from logic provides conflict resistant and loop free features to

CLAPDAWN.

2.4.3 Interfaces

In its modular design CLAPDAWN separates the secondary control logic from primary communication and control part. CP manages the secondary control part and FP manages the primary control and communication part. Interfaces are part of CP and they provides the bridge between FP and CP. CP provides two interfaces control interface and layer interface using which it communicates with the FP.

Control interface and layer interface both have their own importance. Through control interface, FP propagates the events in the form of messages to the CP. Control interface receives these input messages and converts them into events and propagates them to the execution engine. Each component or layer sends its input message to their defined points in CP. Using layer interface, FP receives response of its input parameter from CP. Layer interface receives the result of ECA rule execution and updates the FP using output messages. Here, control interface sends information to the CP and layer interface sends information from the CP.

Referenced architecture in FP modifies itself to make it adjustable to control interface and layer interface. These modifications are minute compared to changes required to implement cross layer interaction within the layer. CLAPDAWN embeds the layer or components of FP to expose information to the CP and creates the catch in FP that accepts information from CP.

Mainly these interfaces are designed considering following two design aspects. First, these interfaces hide the details of cross layer implementation from the FP. Second, it also hides the changing functional plane from the CP. Both are necessary, for stable protocol stack and changing CLI requires the first abstraction. In case of evolving protocol stack like Wireless Sensor Network we require the second type of abstraction.

2.5 Working of CLAPDAWN

In CLAPDAWN FP contains the referred architecture and CP provides controlled communication between modules or layers or components of FP. CP and FP communicates using asynchronous message passing through control interface and layer interface. On update of interested parameters, FP forwards them to CP using messages through standard interface provided in control interface. Control interface gets these messages and converts them into events. With changing FP the message content may change but control interface converts these messages into events which maintains the uniformity in CP.

On event, execution engine fetches rule from knowledge base. Event in execution engine provides input parameter, based on which execution engine gets affected rules from knowledge base. For each rule, execution engine checks the control statement and if it satisfies the condition then execution engine executes the action part and generates the output parameters. These output parameters are further processed by policy configuration. Finally the output parameters are forwarded to layer interface. Layer interface updates the FP by means of output messages.

CLAPDAWN maintains the two unidirectional information flows between CP and FP. One is inflow from FP to CP using control interface and second is outflow from CP to FP using layer interface. The well organized information flow controls the problem of continuous loops in the system. In other words, currently generated information will not trigger next sequence of event until they reached the FP. Once they update the FP then only they are allowed to re-enter the CP.

In CLAPDAWN, FP sends updates to CP using messages. Control interface receives these messages and converts them into events. These events are then propagated to execution engine. This information flow from FP to CP is either synchronous or asynchronous. In asynchronous information flow, FP sends message to CP and continues its task. With synchronous message FP sends the message and waits for the reply from CP in the from of layer interface updates. In section 1.2.2 we discussed different types of information flows in the cross layer interaction. Some are information sharing typed

while others are more profound and modify layer parameters. CLAPDAWN manages them using synchronous and asynchronous information flows.

CLAPDAWN manages information updates using asynchronous messages. FP sends updates to CP using asynchronous information flow which takes care of all the information sharing type cross layer interactions. Mainly the synchronous information flow is used for packets. Each packet which requires processing from cross layer interaction will use synchronous information flow. The patch provided to referred architecture handles this. CP has rule base, using which it sends configuration request to FP. FP has standard interface which receives them and provides the required asynchronous and synchronous updates. Section 2.5.2 provides example of asynchronous and synchronous updates.

2.5.1 Cross layer interaction example

To explain design and working of CLAPDAWN with an example we have implemented a cross layer interaction involving application layer and data link layer. Application layer runs the video streaming application and data link layer (MAC) uses the IEEE 802.11e[10]. Objective of CLI is to increase the quality of video streaming by dynamically managing the prioritized queues at MAC layer.

Performance of IEEE 802.11e can be fine tuned by various mechanisms like Contention Window Size, TXOPlimit or data transmission rate. Different configurations of these parameters vary the performance of IEEE 802.11e. Required performance for the applications running at application layer can be achieved by configuring these parameters. In video streaming, different parts of encoded video data have different significance. Required performance for video streaming application can be achieved by configuring the IEEE 802.11e parameter based on significance of encoded video data.

Video streaming which uses MPEG-4 standard divides the video data into 3 frames I-Frame, P-Frame and B-Frame. Each frame has different significance. I-Frames are most critical frames for video decoding than P-Frames and B-Frames. B-Frame has lowest significance among the three frame types. IEEE 802.11e has different access queues (AC0-AC3) having separate buffer. These queues work independently from each other.

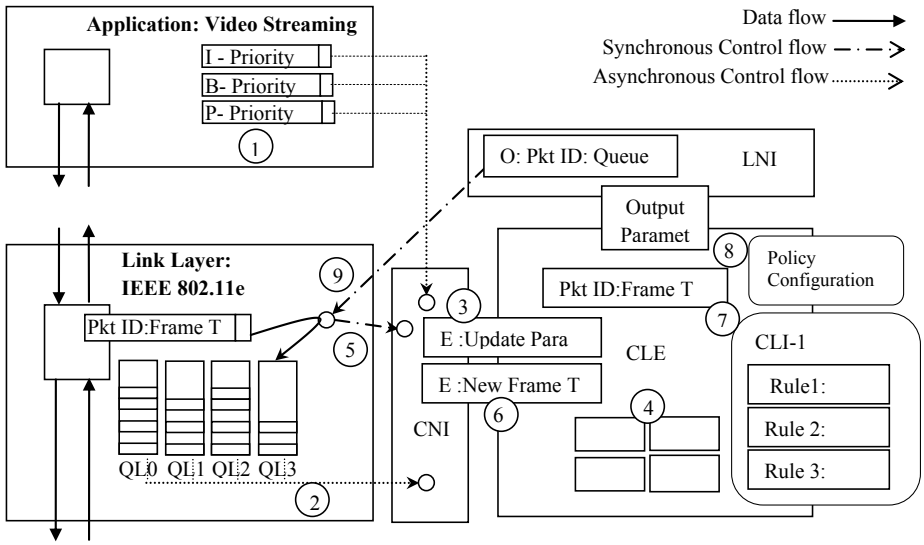


Figure 2.5: Working of CLAPDAWN with Cross Layer Interaction occurring between application layer running video streaming application and Link layer running IEEE 802.11e

Different access queues have different priority. AC0 is highest priority queue while AC3 is least priority queue among AC0-AC3. In Cross layer approach [10], MPEG-4 video packets are dynamically mapped to appropriate AC based on both the significance of the video data and the network traffic load.

2.5.2 CLAPDAWN implementation

CLAPDAWN implementation of above CLI is shown in figure 2.5. Two layers are involved in this cross layer interaction, application layer and data link layer (MAC). Figure 2.5 shows them with data flow and secondary control flow. Solid line shows the dataflow in the system. Secondary flows are of two types synchronous and asynchronous. Secondary data flow that deals with data packet are synchronous while one which updates parameter in CP are asynchronous.

Using asynchronous update events, application layer provides the significance of each frame ($Prob_{type}$) and MAC layer provides queue length to the CP, shown as step 1, 2 and 3 in figure 2.5. CP gets these events and updates its parameter, shown as step 4. If execution engine finds rules related to these parameters then it executes them. These asynchronous messages take care of the communication occurring across the layers.

Secondary control which handles per packet communication works synchronously. On arrival of packet it generates the event in CP and waits for the decision, shown as step 5 and 6. Event carries the frame type information with the packet identity. Execution engine obtains related rules from the knowledge base and executes them by applying policy configuration, step 7 and 8 in figure 2.5. Execution of these rules decide the queue number for the current frame and sends that queue number to FP using output messages. FP receives that message containing packet identity and queue number and puts the packet in the selected queue, as shown in step 9. Following are the cross layer interaction steps:

CLI-1 Rule-1

$Pr_{new} = \frac{Pr_{type} * (QL0 - Threshold_{low})}{(Threshold_{high} - Threshold_{new})}$

$rnd =$ random number between $(0, 1)$

CLI-1 Rule-2

$If(QL0 < Threshold_{low})$ {

$Quequ_{number} = 0$

}

CLI-1 Rule-3,4

$else\ if(QL0 < Threshold_{high})$ {

$if\ (rnd > Pr_{new})\ Quequ_{number} = 0$

$else\ Quequ_{number} = 1$

}

CLI-1 Rule-5,6

$elseif(QL0 > Threshold_{high})${

$if(rnd > Pr_{new})\ Quequ_{number} = 1$

$else\ Quequ_{number} = 2$

}

Above shown set of rules are executed on occurrence of events. These rules are encoded into ECA rules and stored in knowledge base. The following code snippet shows one of the ECA rule present in knowledge base.

CLI 1 - RULE 2 PARA LL frametype 2 1

CLI 1 - RULE 2 CONDITION 1 LL frametype CO 1 e

CLI 1 - RULE 2 CONDITION 2 LL QL0 LL $threshold_{low}$ l

CLI 1 - RULE 2 ACTION 1 LL $Queue_{number}$ CO 0 CO 0 d E

The above is ECA representation of rule 2 of cross layer interaction 1. It shows that the rule has two conditions to check and one action to perform. It checks for the frame type (Link Layer) and queue length (QL0) for queue zero. If frame type and queue length of queue zero satisfies the condition then it assigns the priority queue zero to the given frame. ECA rules are generated with ECA rule editor.

2.6 Summary

We have discussed our proposed architecture in this chapter and shown structure and working of it. For better handling of cross layer interaction, architecture is divided into two parts a) functional plane and b) control plane. Control plane manages all the cross layer interactions of the system through its subcomponents. Structure and working of our architecture highlights various feature of our architecture. We have compared them with the other architectures in the next chapter.

Chapter 3

CLAPDAWN: Comparative Study

In chapter 1 we had discussed available cross layer architectures in the literature and their limitations. After going through the details of CLAPDAWN in chapter 2, in this chapter we are comparing CLAPDAWN with the available cross layer architectures. Using available literature, we have identified a set of parameters based on which we have performed comparative study of cross layer architectures. Section 3.1 shows the set of evaluation parameters and section 3.2.1 covers the comparative study. Based on study did in 3.2.1 we have highlighted limitations of CLAPDAWN in section 3.3.

3.1 Evaluation Metrics

By going through work done in literature we have identified a few metrics to evaluate the performance of cross layer architectures. This section briefly explains each of the metrics. Each architecture is evaluated considering following performance measures.

3.1.1 Multiple cross layer interaction, any layer to any layer communication (A)

We check whether the architecture supports multiple cross layer interactions or not. In our study, we have considered only those architectures which are designed to support multiple cross layer interactions. In a cross layer interaction, a layer shares its information for

other layers or accesses the control mechanism of the other layers. In the case of multiple cross layer interactions, next important point is whether all the layers can access the information from all the other layers. In other words, whether the architectures supports any layer to any layer communication.

3.1.2 Rapid development time (B)

Cross layer interaction introduces the secondary controls in the system. Cross layer architecture implements this additional control as a part of layer, adding horizontal or vertical layer, or as separate component. Here, time required to implement cross layer interaction depends on the support provided by the architecture. Based on adopted method, different architecture takes different time to introduce cross layer interaction in the system. By analyzing the process of how architecture introduces cross layer interactions in the system, we have evaluated different architecture for their development time. Rapid development time study highlights the complexity involved in the architecture.

3.1.3 Rollover from cross layer interaction (C)

During design time, protocol designer experiments with different cross layer interactions for better performance. In networks like, wireless sensor networks and ad hoc networks, the working environment and network protocol experience frequent changes. For such dynamic networks, it requires that architecture must provide necessary support to such experimentation. It is required that architecture provides support for smooth addition and removal of cross layer interaction in the system. Rollover analysis is a study of how easy it is to remove the existing cross layer interaction for the system.

3.1.4 Footprint on reference architecture (D)

Cross layer interactions are introduced to the system to improve the performance of the system. In that process, how much modification is required to the referenced protocol or layer is measured by the footprint of cross layer interaction on the reference architecture.

3.1.5 Feedback loop prevention (E)

Applications like video conferencing on vehicular ad hoc network requires multiple cross layer interactions for the smooth functioning of the application. With the multiple cross layer interactions it is likely that changes made by one layer triggers a sequence of changes, that endup having a loop. Loop formed by such cross layer interactions is called a feedback loop in the system. Feedback loops are critical measures of the cross layer architecture. With the multiple cross layer interactions it is very likely that the system may face feedback loops. It is the responsibility of the architecture to take care of such feedback loops. Architectures are measured on the basis of their support provided to prevent feedback loops from the system.

3.1.6 Conflicting cross layer interactions (F)

With the multiple cross layer interaction, it may happen that more than one cross layer interaction is creating side effect on a common parameter. Cross layer interactions which updates the same parameter are considered as conflicting cross layer interactions. If that is the case then system experiences instability in its working. Here, we have analyzed different architectures to see how they are handling possible conflicts occurring between multiple cross layer interactions.

3.1.7 Time Overhead, space overhead and message overhead (G-I)

Cross layer interactions are the secondary control added to the system. They require information from other layer and they process this information for decision making. This requires computation time, memory to store cross layer interaction and messages to bring required information. Time overhead measures the additional time required to perform the cross layer interaction that includes delay in bringing necessary information to the cross layer interaction and processing time. Space overhead measures the additional memory required to store the cross layer interaction code and parameters. Message

overhead is a measure of the number of additional message exchange that take place between various components of cross layer architecture.

3.1.8 Flexibility measure (J)

Cross layer interactions are dynamic in nature. With changing environment they require some modification. Flexibility measure evaluates this process. It checks how easy it is for the cross layer architecture to make necessary correction to adapt according to the changing conditions like underlying protocol changes, change in cross layer interaction itself or change of host operating system.

3.2 Comparative Study

In our study we have compared architectures covered in chapter 1 with CLAPDAWN. For each performance measure the architecture is given points on the scale of 1 to 5. Here point 5 shows high performance and point 1 shows the low performance for that parameter. Performance parameters are as follows:

3.2.1 Multiple cross layer interaction, any layer to any layer communication (A)

This section looks at the comparative study of different architectures considering whether they provide support for multiple cross layer interactions or not. Mainly we have selected only those architectures which provide support for multiple cross layer interactions. With multiple cross layer interactions, our second objective is to check that is it possible to have any layer to any layer communication in the architecture. This study sees that some of the cross layer architectures like [28] which provide support for multiple cross layer interactions do not provide possibility of any to any layer communication.

Table 3.1: Multiple Cross Layer Interaction Support

Architecture	Support for Multiple Cross Layer Interaction	Points(A)
TinyCubus	Various cross layer interactions are managed by Tiny Cross Layer Framework. Cross Layer Framwork state repository to implement multiple cross layer interactions and any to any module communication.	5
MobileMAN	Multiple cross layer interactions are implemented with the help of additional layer referred as Network Status Layer.	5
CL interaction	Middleware is provided to support multiple cross layer interactions. Main focus is on bottom up communication only.	3
CrossTalk	Provides local and global view to implement multiple cross layer interactions.	5
ECLAIR	Using Tuning layer and Optimizer agent it provides the support for multiple cross layer interactions and any to any layer communication.	5
CLAPDAWN	Through separate control plane it provides the support for multiple cross layer interactions and any layer to any layer communication.	5

3.2.2 Rapid development time (B)

Table 3.2 provides the comparative analysis of rapid development time. They are ranked on the basis of effort required to be put to make cross layer interaction part of the system. Point 1 difficult to implement new cross layer interaction showing complex process and Point 5 shows easy process of adding cross layer interaction in the system. It shows that majority of the architectures implement the cross layer interaction in the layer itself. That increases the development and debugging time of the cross layer interaction. Here ECLAIR and CLAPDAWN implements the cross layer interaction outside the reference architecture. That helps in reducing the development time. Further, CLAPDAWN uses ECA rules in knowledge base that makes development process simpler than the ECLAIR.

3.2.3 Rollover from cross layer interaction(C)

Many experimental studies or protocol development require experimentation. In that process they frequently go for rollover and restore the system in the previous state. Table 3.3 shows the comparative study of rollover in different architectures. Because of

Table 3.2: Rapid Development Time

Architecture	Rapid Development Time	Points(B)
TinyCubus	Tiny Configuration Engine adds new cross layer interaction in the system. Cross Layer Interaction are implemented using TinyOS callback mechanism. With TinyOS callback it is difficult to wire new components in the system	2
MobileMAN	It consumes data provided by Network Status Layer and cross layer code is implemented inside the layer itself.	3
CL interaction	Cross layer interactions are implemented inside the layer by modifying layer code.	3
CrossTalk	Cross layer interactions are implemented inside the layer and required information is provided by global and local database	3
ECLAIR	Cross layer interactions are implemented as separate optimizer agents independent of network protocol stack	5
CLAPDAWN	Cross layer interaction are implemented inside the separate control plane independent from the network protocol stack	5

separation of cross layer interaction from the network protocol stack it becomes easy for ECLAIR and CLAPDAWN to perform rollover in the system.

Table 3.3: Rollover from Cross Layer Interaction

Architecture	Rollover from Cross Layer Interaction	Points(C)
TinyCubus	TinyCubus maintains wiring between the components and uses data stored in the repository to implement cross layer interaction. It is hard to recover from this wiring between components.	3
MobileMAN	Cross Layer Interaction code is implemented inside the layer itself. Requires changes in the layer code itself.	1
CL interaction	Cross Layer Interaction code is implemented inside the layer itself. Requires changes in the layer code itself.	1
CrossTalk	Cross Layer interactions are implemented inside the layer itself. Requires changes in the layer code itself.	1
ECLAIR	Cross Layer interactions are implemented as separate agents configured with tuning layer. Removing them from the system still triggers the development cycle.	4
CLAPDAWN	Cross Layer interaction are implemented inside the separate control plane with the use of cross layer knowledge base. Removal of rules from the knowledge base removes the cross layer interaction from the system.	5

3.2.4 Footprint on reference architecture(D)

To maintain stability in the system it is highly desirable that cross layer interaction creates small footprint on the referenced architecture. Table 3.4 shows the comparative study of cross layer architecture highlighting footprints on the layers. Those architectures, which implement cross layer interactions inside the layer itself, have large footprint on the layer compared to those, which implements cross layer interaction outside the layer. In that regards ECLAIR and CLAPDAWN have small footprint compared to other architectures. Here CLAPDAWN provides the standard interface that makes this process smooth compared to other architectures.

Table 3.4: Footprint on Referenced Architecture

Architecture	Footprint on referenced architecture	Points(D)
TinyCubus	Each cubus in TinyCubus is combination of application parameter, Operating system parameter and optimization parameter. Cross layer interactions modeled as cubes and maintained by framework.	3
MobileMAN	Changes Layer itself for cross layer implementation. Persistent footprint on the layers.	2
CL Interaction	Changes Layer itself for cross layer implementation. Persistent footprint on the layers	2
CrossTalk	Changes Layer itself for cross layer implementation. Persistent footprint on the layers. Modifies the inter node messaging structure to get the global view.	1
ECLAIR	Cross layer interactions are implemented outside the layer and only interfaces are created inside the layer to get required information. Each information comes with separate interface .	4
CLAPDAWN	Cross layer interactions are implemented outside the layer and only two interfaces are created inside the layer for the input and output communication. Standard interface makes footprint small and easy to manage.	5

3.2.5 Feedback loop prevention (E)

Table 3.5 shows the comparative study of possibility of feedback loop in the different cross layer architectures. It shows that no cross layer architecture provides complete solution to the feedback loop problem. Those architectures, which implement cross layer interac-

tion inside the layer, they do not have the knowledge of other cross layer interactions in the system. Here chances of feedback loops are high. In other cases architectures which manage the information about the cross layer interaction like CL interaction and CLAPDAWN reduce the chances of feedback loops with the help of additional knowledge. But still there are chances that parameters which are transparent to the cross layer interaction may create feedback loops in the system.

Table 3.5: Prevention from feedback loop

Architecture	Feedback loop prevention	Points(E)
TinyCubus	Configuration Engine using resource repository maintains information about the cross layer implementation but with dynamic components it is hard to keep track of feedback loop in the system	2
MobileMAN	Cross Layer Interactions are implemented in the layer itself and they use information provided by the network status layers and the current layer. Network status layer does not keep track of the information flow among the layers.	1
CL Interaction	Middleware layer is provided between the Network layer and MAC layer that manages the different cross layer interactions. That reduces the chance of feedback loop in the system.	4
CrossTalk	Cross Layer interaction implemented inside the layer communicates using information provided by local and global view layer. This distributed control flow increases the chance of feedback loop in the system.	1
ECLAIR	Cross layer interactions are implemented independent from each other outside the layer as optimizing agent. Distributed control increases the chance of feedback loop	2
CLAPDAWN	Cross layer interactions are implemented inside the control plane with the help of cross layer knowledgebase and cross layer engine. Available information of cross layer interaction reduces the chance of feedback loop.	4

3.2.6 Conflicting cross layer interactions(F)

Conflicts are easy to locate in the system compared to feedback loops. Table 3.6 shows the comparative study of different architectures considering whether they provide safeguards against the possible conflicts among cross layer interactions. It shows that the architecture which manages the cross layer interaction information provides better protection

against the conflict. Architectures which use central repository for cross layer interaction information have not discussed conflicting scenario among cross layer interactions. These architectures can be upgraded to provide conflict free cross layer interaction by using race condition avoidance mechanisms. CLAPDAWN uses dependancy graph which helps it to identify all possible conflicts occurring between different cross layer interactions.

Table 3.6: Preventions from conflicting cross layer interactions

Architecture	Conflicting Cross Layer Interactions prevention mechanism	Points(F)
TinyCubus	TinyCubus uses Callback mechanism of TinyOS and it has limitation with reconfiguration and that limitation makes it Conflict free.	5
MobileMAN	Cross Layer Interactions are implemented in the layer itself and they consume the information provided by layers. Network status layer does not have information about different Cross layer interactions.	2
CL Interaction	Middleware layer is provided between the Network layer and MAC layer that manages the different cross layer interaction. That reduces the chance of conflict among the cross layer interactions.	5
CrossTalk	Local and global view have no control over information consumed and produced by the layers that increase the chances of conflict in the system.	1
ECLAIR	Separate tuning layer provides the required information to each optimizer which updates them independently. This distributed update process increases the chance of conflict in the system	2
CLAPDAWN	Cross layer knowledgebase maintains all the cross layer interaction with the required input and output parameter. Using dependency graph it manages all the possible conflict in the system.	5

3.2.7 Time overhead

Table 3.7 shows the time overhead involved in different cross layer architectures. It shows that architecture, which implements the cross layer interaction inside the layer, has less time overhead compared to architecture which implements it outside. Architectures like CL interaction implements cross layer interaction inside the layer but the required information comes piggybacked inside the packet that adds additional delay to the overall

processing.

Table 3.7: Time overhead in cross layer architecture

Architecture	Time Overhead	Points(G)
TinyCubus	With dynamic modules, it requires additional time to find the correct component associated with the repository data.	3
MobileMAN	Cross layer interaction implemented inside the layer, takes information from network status layer.	4
CL Interaction	It uses piggybacked mechanism to bring required information to cross layer interaction from the middleware that adds additional delay. Network layer queue may increase this delay.	2
CrossTalk	Cross layer interactions are implemented in layer itself using information provided by local view layer and global information. Local view is updated by direct interfaces while global view is updated by information provided by piggybacking, that adds additional delay.	3
ECLAIR	Information is provided by tuning layer and each optimizer works independently. It is implemented outside the layer that increases delay.	3
CLAPDAWN	Information provided by interfaces and processed by cross layer engine. To increase the efficiency output parameters are mapped to specific engine. Processing is done outside the layer and in sequence adds delay to the system.	2

3.2.8 Space overhead

One of the important parameter is the space required to implement different cross layer interactions. In networks like sensor networks, available memory space is less compared to other networks. The table 3.8 shows the space required by different cross layer architectures. It shows that all the architectures require additional space to maintain parameters involved in cross layer interactions. For example, in ECLAIR, information exchange code and local parameters are replicated in different optimizers. In the case of CLAPDAWN, execution engine only keeps control over such replication.

Table 3.8: Space overhead in cross layer architecture architecture

Architecture	Space Overhead	Points(H)
TinyCubus	Configuration Engine, Topology Control Framework and Data Management Framework requires additional memory for each of the TinyCubus.	2
MobileMAN	Network Status Layer maintains the required information.	4
CL Interaction	Middleware layer manages information about all the cross layer interaction and parameters, with cross layer interaction implemented inside the layer.	2
CrossTalk	Cross Layer information is managed by local view and global view. Cross layer interaction are implemented inside the layer.	3
ECLAIR	Cross layer information is provided by tuning layer and interactions are implemented outside the layer, which requires additional memory.	2
CLAPDAWN	Cross layer information is provided by separate layer and each cross layer interaction is stored as ECA rule in knowledge base which requires memory.	3

3.2.9 Message overhead

Table 3.9 shows the comparative study of message overhead involved in cross layer interaction. Those architectures, which use common database to store the cross layer interaction and implement it inside the layer itself, do not have any messaging overhead. The architecture which use the piggybacked mechanism to convey required information to the cross layer interaction is free from the messaging overhead. The architecture, which implements the cross layer interaction outside the layer, have messages overhead proportional to the event frequency and the number of parameters involved in it.

3.2.10 Flexibility measure(J)

Flexibility measure is one of the important parameters involved in cross layer architecture design. It is the responsibility of cross layer architecture to provide required mechanism to adapt to changing environment as well as changing cross layer interactions.

Table 3.9: Message overhead in cross layer architecture

Architecture	Message Overhead	Points(I)
TinyCubus	Dynamic wiring between the components do not require any messaging.	4
MobileMAN	Messages required to get and put information to Network Status layer.	3
CL Interaction	Through piggybacking information forwarded to middleware.	4
CrossTalk	Using piggybacking information is provided to global view and local view.	4
ECLAIR	Cross layer interaction are implemented outside the network stack . Additional messages required for each piece of information exchanged between a) network protocol stack and tuning layer, and b) tuning layer and optimizers.	2
CLAPDAWN	Cross layer interactions are implemented outside the network stack . Additional messages required for each piece of information exchanged between Functional Plane and Control Plane.	3

Table 3.10: Flexibility with Cross Layer Interactions

Architecture	Flexibility Measure	Points(J)
TinyCubus	Due to underlying dependency of TinyOS, the architecture does not have the flexibility of adjusting different demands. Further, callback mechanism of TinyOS restricts the scope of TinyCubus.	3
MobileMAN	Each cross layer interaction implemented inside the layer, takes information from network status layer. Any modification made to network status layer affects the all the cross layer interactions.	1
CL Interaction	Middleware layer manages all the cross layer interactions implemented inside the layers.	4
CrossTalk	Cross layer interaction are implemented inside the layer. Modification made to the global view and local view triggers update in the system.	2
ECLAIR	Cross layer interactions are implemented as separate optimizer using tuning layer. System is flexible and adaptable to different changes in underlying protocol. Modification to cross layer interaction is also transparent to the rest of the system.	4
CLAPDAWN	Cross layer interactions are implemented as separate control plane. System is flexible and adaptable to different changes in the underlying protocol. Modification to cross layer interaction is also transparent to the rest of the system.	5

3.2.11 Relative ranking

We have summarized the comparative study of cross layer architectures by defining the rank for the cross layer architectures. Overall performance is calculated as average of individual performance achieved by cross layer architectures for different parameters. To decide the relative rank we have short listed our architectures based on their average value. Following table shows the average performance with the rank. It shows the CLAPDAWN has higher rank compared to other architectures.

Table 3.11: Comparative Study of different Architectures

Architecture	A	B	C	D	E	F	G	H	I	J	Average	Rank
CLAPDAWN	5	5	5	5	4	5	2	3	3	5	4.2	1
ECLAIR	5	5	4	4	2	2	3	2	2	4	3.3	2
TinyCubus	5	2	3	3	2	5	3	2	4	3	3.2	3
CL Interaction	3	3	1	2	4	5	2	2	4	4	3	4
MobileMAN	5	3	1	2	1	2	4	4	3	1	2.6	6
CrossTalk	5	3	1	1	1	1	3	3	4	2	2.4	5

3.3 CLAPDAWN Limitations

Above sections show the comparative study of different architectures with their relative ranking information. Ranking is calculated based on the average performance achieved by all the parameters. It shows that CLAPDAWN has higher rank than other architectures. Though it has higher rank, architecture performs relatively low in some of the aspects like memory overhead and running time overhead.

As all the cross layer interactions are implemented as ECA rules, it requires additional memory to store those cross layer interactions. Further architecture also has to maintain its parameters list that also requires additional memory. Here memory requirement is proportional to the number of parameters involved in cross layer interaction and number of rules affected by the give parameter. Further CLAPDAWN maps incoming and outgoing parameters with the parameters involved in ECA rules that adds initial configuration time.

CLAPDAWN uses single execution engine to execute the given rules that creates delay in overall processing. This delay can be overcome by introducing more than one execution engines. As ECA rules are independent from each other, set of rules can be mapped to different execution engines to make execution faster.

3.4 Summary

In this chapter we have highlighted evaluation metrics for the cross layer architectures. Based on identified metrics we have evaluated some of the well known cross layer architectures available in literature. Their comparison shows that CLAPDWAN provides better support for the cross layer interaction implementation than the others. Further we have ranked these architectures based on their cumulative performance. Based on these ranking we have selected ECLAIR with the CLAPDWAN and implemented it on simulator to derive experimental results for better clarity and criticism.

Chapter 4

CLAPDAWN Simulation Study

Chapter 3 shows study of different architectures and their comparative analysis. It shows that CLAPDAWN outperforms various architectures in many design features. The comparative analysis based ranking shows that CLAPDAWN has higher rank than rest of the considered architectures. Next architecture in rank is ECLAIR. To validate CLAPDAWN architecture, we have implemented first two higher ranking architecture CLAPDAWN and ECLAIR on network simulator 2. This chapter covers the implementation details of both the architectures.

Performance of both the architectures are evaluated by performing two simulation studies, discussed in section 4.2.1 and 4.2.2. We have implemented our architecture modifying Network Simulator 2 using POSIX sockets and threads. Network simulator provides the reference network protocol stack to which we embed functional plane and control plane. Combination of these two generates the CLAPDAWN. Using CLAPDAWN, the extended Network Simulator 2, we have performed two simulation studies. In our first simulation study (section 4.2.1), we implemented the CLI discussed in section 2.5. We compared it with other cross layer architecture, ECLAIR, using two performance measures: packet delivery ratio and frame loss ratio. In second simulation study (section 4.2.2) we created a complex scenario of having conflicting CLIs. First study shows the applicability of the architecture while second study shows how CLAPDAWN outperforms the other architecture in complex scenarios.

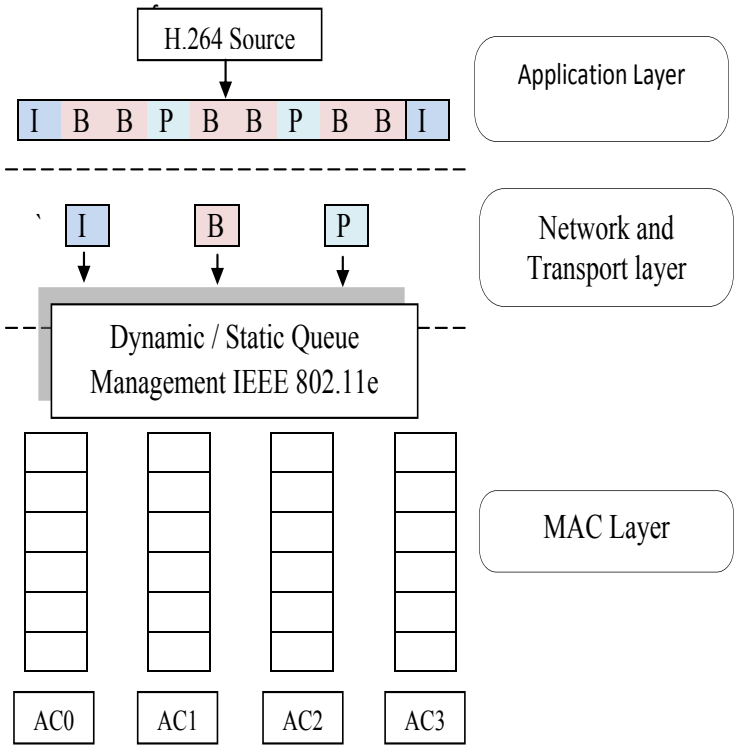


Figure 4.1: IEEE 802.11e dynamic queue management using application layer video streaming data.

4.1 Architecture Implementation

Details of CLAPDAWN architecture is given in chapter 2. As discussed there, architecture has two parts functional plane and control plane. For implementation, we have selected TCP/IP layered architecture as reference architecture. Reference architecture consist of five layers: application layer, transport layer, network layer, data link layer and physical layer. Network simulator 2 provides implementations of these layers and generates the functional plane of the CLAPDAWN.

On defined functional plane, we have implemented the cross layer interaction discussed in section 2.5.1. Two layers are involved in this cross layer interaction, application layer running H.264 video streaming application and data link layer running IEEE 802.11e protocol. Figure 4.1 shows the cross layer interaction. In cross layer interaction video packets are dynamically mapped to appropriate IEEE 802.11e queue based on the significance of the video data and the network traffic load. Cross layer interaction replaces the static queue management by dynamic queue management.

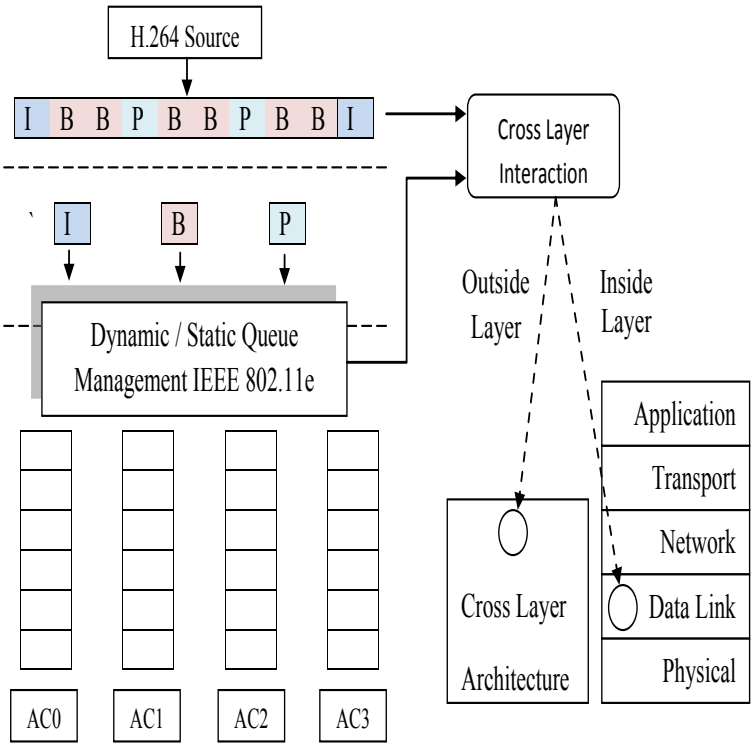


Figure 4.2: Cross layer protocol implementation

Cross layer interaction takes parameter from application layer and data link layer and can be implemented inside the layer or outside the layer as shown in figure 4.2. In our study we have implemented the cross layer interaction in three different ways a) inside the layer, and outside the layers using b) CLPADAWN architecture and c) ECLAIR architecture. Following section explains the implementation details.

4.1.1 Inside layer implementation

First implementation of our cross layer interaction is inside the data link layer. Here we have modified the IEEE 802.11e priority queue management to handle cross layer interactions as shown in figure 4.3. Figure shows the organization of cross layer interaction inside the data link layer. Left side shows the cross layer interactions and right side shows the cross layer interaction placement. It shows that MAC802_11e contains the queue management part in PriQ, which maintains the drop tail queues DTail. MAC802_11e receives frame from above layer. PriQ manages the frame using DTail queues. PriQ receives this frame and applies dynamic queue management policy discussed in section

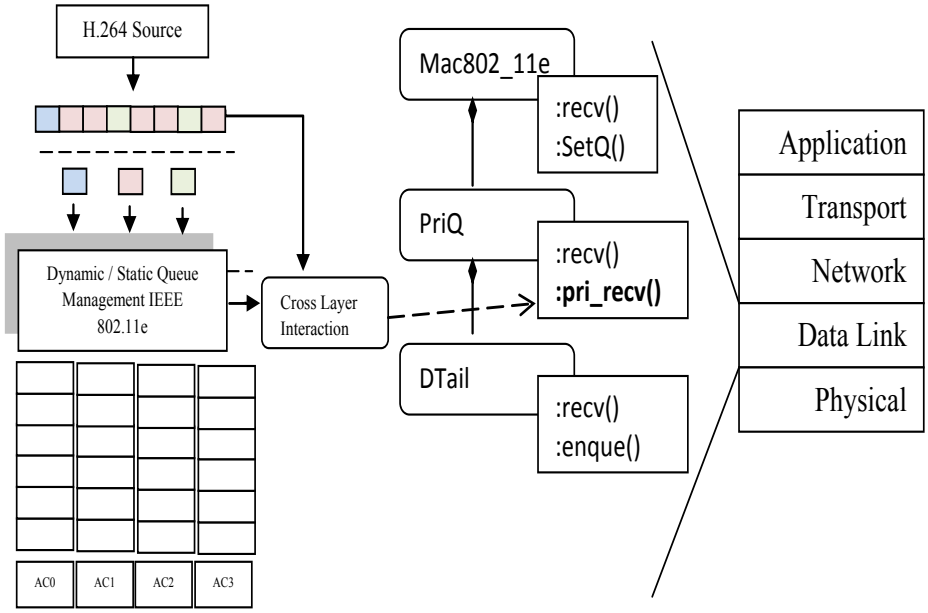


Figure 4.3: In layer cross layer protocol implementations.

2.5.1.

4.1.2 ECLAIR implementation

Figure 4.4 shows the ECLAIR implementation of cross layer protocol. As discussed in chapter 1, ECLAIR implements the cross layer interaction outside the layer architecture using tuning layer and optimizer subsystem. As shown in figure 4.4, layer interacts with its corresponding tuning layer. Cross layer interactions are implemented as optimizer in optimize subsystem. Implemented optimizer interacts with tuning layers to get the required information. Each layer provides interface using which tuning layer interacts with the layer.

We have modified the protocols in network simulator 2 by adding interfaces to get the required information. Layers are modified to provide set of interfaces for each piece of information. Tuning layers are implemented outside the regular layer structure. Tuning layer makes connection with layers and sends the data requests. Connection handler at layer receives connection request and data request. Based on data request it calls the

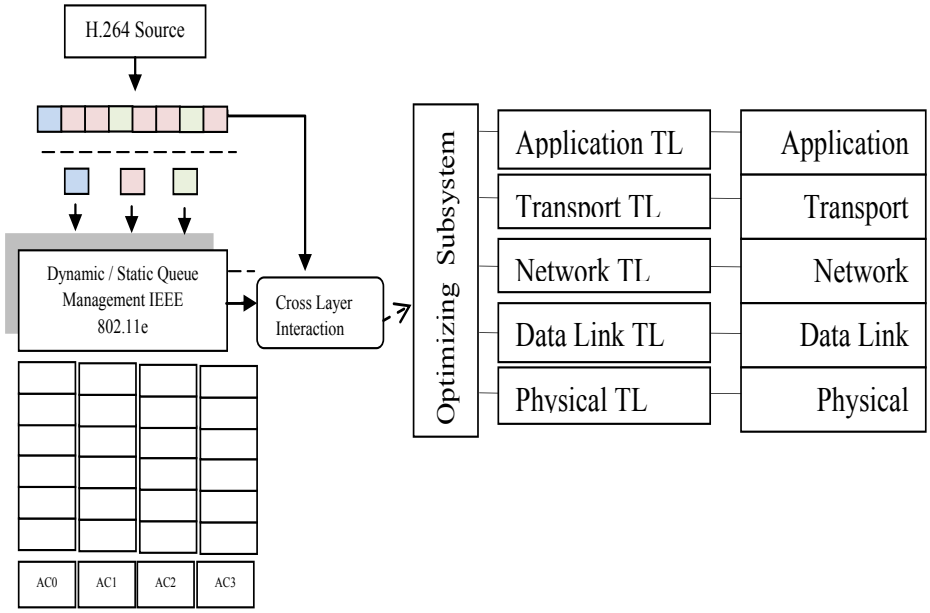


Figure 4.4: ECLAIR implementation of cross layer protocol

interface provided by layer.

Figure 4.5 shows the implementation details of ECLAIR. Each layer involved in cross layer implementation receives the connection request from its corresponding tuning layer. Example shows the application layer provides connection interface. Using that interface tuning layer makes connection with application layer at the time of initialization. Application layer provides interface like $req_{priority}()$. Tuning layer uses provided interfaces to get the required information. Tuning layer forwards the collected interface to the optimizer. Optimizer calls the tuning layer interface for the information and provides the results back to the tuning layer interface. As discussed in previous chapter, any modification to the optimizer, tuning layer or protocol triggers the development cycle.

4.1.3 CLAPDAWN implementation

Figure 4.6 shows the CLAPDAWN implementations of cross layer interaction. As discussed in chapter 2, and shown in figure, cross layer protocol is implemented into cross layer knowledge base of CLAPDAWN. Cross layer engine executes this cross layer proto-

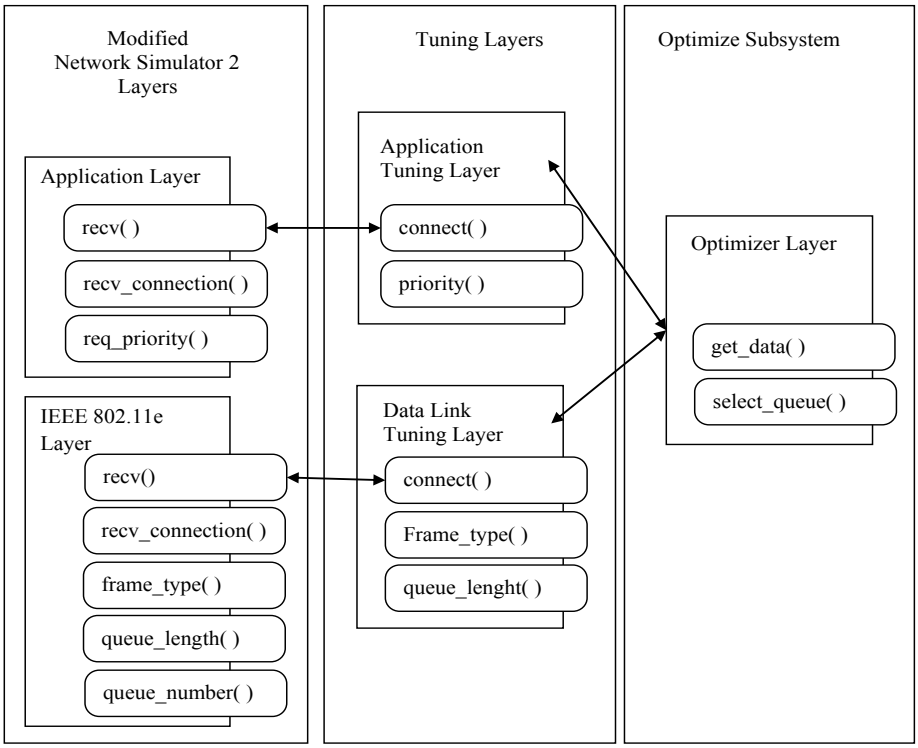


Figure 4.5: ECLAIR implementation in Network Simulator 2 with Interfaces

col using information provided by control interface and updates the referenced functional plane using layer interface.

Functional plane is implemented by modifying layers of Network Simulator 2. Protocols involved in the cross layer interaction are patched with the interface. Using this interface Control Network Interface(CNI) makes connection with the protocol. Implementation of control interface is outside the network layers. Control interface using a socket makes connection with layers of Network Simulator. Using this connection it exchanges messages with functional plane. Same way control plane forwards its information to functional plane using patched interface within functional plane.

Layer in NS2 implements two patches one for control interface and second for layer interface. Figure 4.7 shows the main components of CLAPDAWN implemented in Network Simulator 2. Layers which implement two interfaces also maintain the data structure by mapping variable name with their address. Execution engine implements the knowledge base, control interface and layer interface in it.

Here we have briefly discussed the implementation of cross layer interaction involved

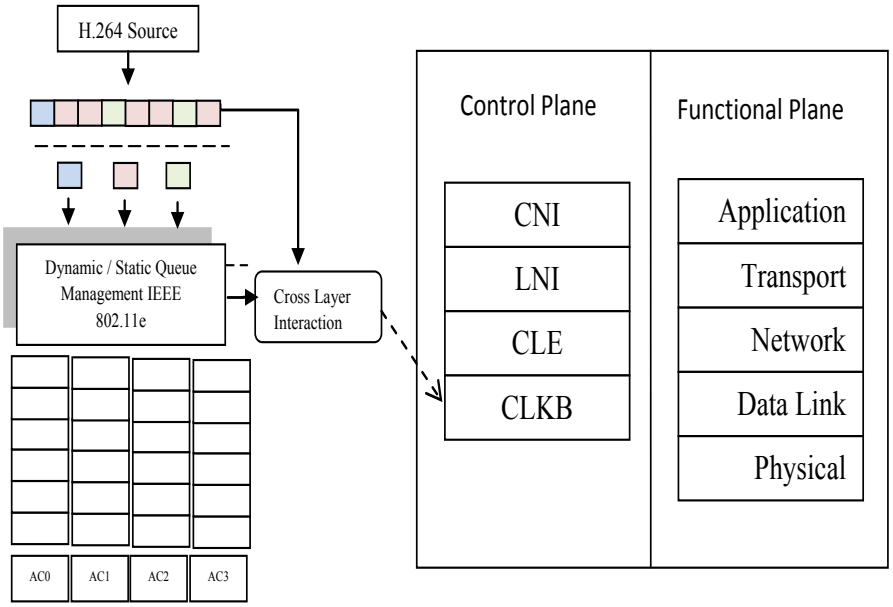


Figure 4.6: CLAPDAWN implementation of cross layer protocol

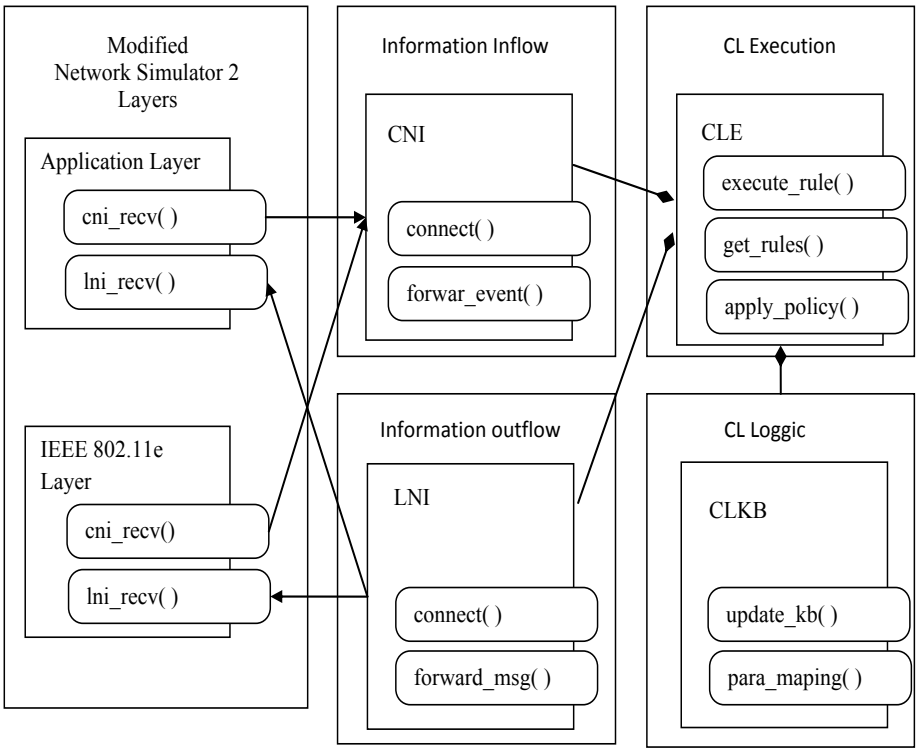


Figure 4.7: CLAPDAWN implementation of cross layer protocol

in first simulation study. Our second simulation study involves two cross layer interactions. First cross layer interaction takes place between network layer and data link layer. Second cross layer interaction takes place between network layer and physical layer. Implementation details of the second simulation study is similar to the first one. Only change is here we are dealing with two cross layer interactions simultaneously.

4.2 Simulation Study

This section discusses the simulation results of the above discussed cross layer interactions. In the section 4.2.1 we discuss the results of the first cross layer interaction and section 4.2.2 discusses the results of the second cross layer interaction.

4.2.1 Simulation study 1: complex video streaming example

Section 4 discusses the working of cross layer interaction, which dynamically adjusts the IEEE 802.11e access queues based on traffic condition and application's requirement. Dynamic queue management is improvement over static queue management. In static mechanism traffic classes are mapped to specific queues without considering the traffic load in that class.

Figure 4.8 shows behavior of four AC queues (AC0-AC3) in static mapping, having high video and audio traffic. Normally video and audio traffics are mapped to AC0. Figure 4.8 shows the utilization of queues. Here, due to high audio and video traffic, AC0 is highly utilized and reaches its maximum limit. After crossing the maximum queue limit it starts packet dropping, even though the AC1-AC3 have space to handle these additional packets.

Figure 4.9 shows that how cross layer interaction discussed in section 2.5 improves over it. Result shows how dynamic mapping changes the queue utilization scenario. Here, the load is distributed among the other queues if AC0 crosses its minimum threshold. In simulation minimum threshold is set to very low value (5 pkts). In all the cases the maximum queue limit is 50 packets.

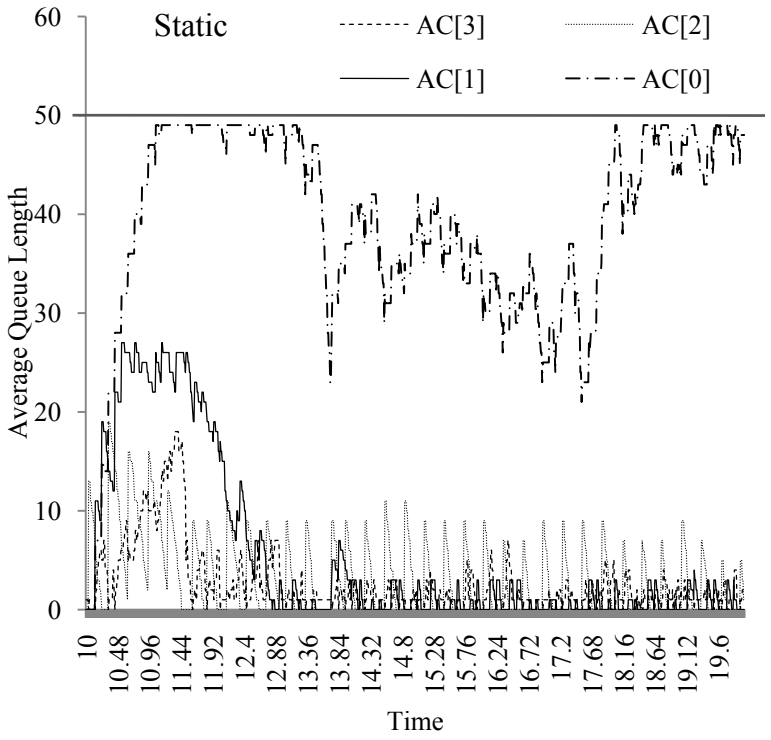


Figure 4.8: Average Queue length experienced with static method

In our simulation study, we have implemented dynamic scenario in three different ways: a) in layer implementation, b) ECLAIR and c) CLAPDAWN. On these implementations we have measured the performance of the system by varying the traffic conditions. We have compared them using packet delivery ratio and packets lost (audio and video).

Figure 4.10 compares above mentioned implementations of cross layer interaction with the static approach. It shows that dynamic approach has better delivery ratio than the static approach. Here, as the traffic increases, this gap decreases. The reason is that configured traffic is more than the capacity of underlying wireless networks. Result also shows that all the implementations of dynamic approach are almost same. In other words, separating the cross layer interaction from layered approach does not degrade the performance of the system.

Next result shown in figure 4.11, highlights effect of dynamic approach on prioritized data. Here, I-Frame information and audio information are critical information. Result shows that all dynamic approaches have similar behavior in all the cases.

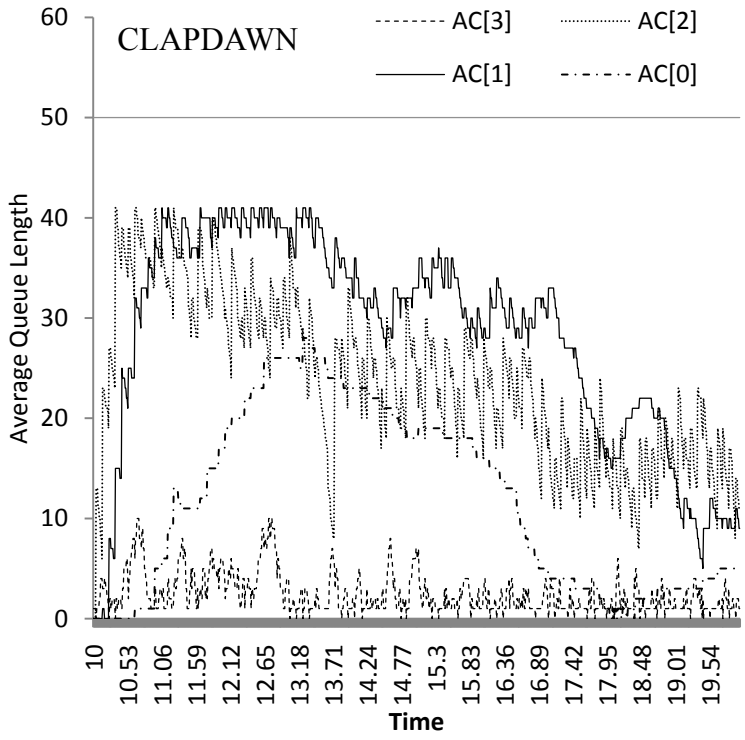


Figure 4.9: Average Queue length experienced in dynamic method with CLAPDAWN architecture

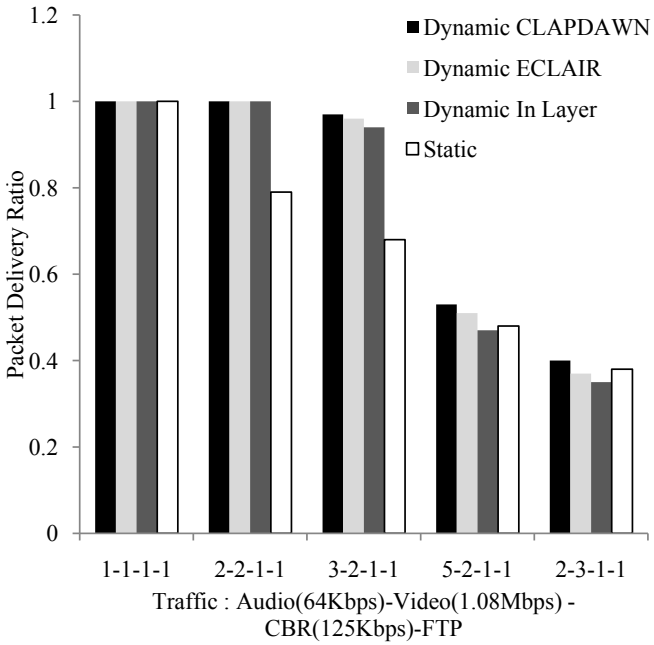


Figure 4.10: Packet delivery ratio experienced by static method and dynamic method implemented in different architectures

In summary, the first simulation study highlights the feasibility of CLAPDAWN. It shows cross layer knowledge base of CLAPDAWN can efficiently implement complex

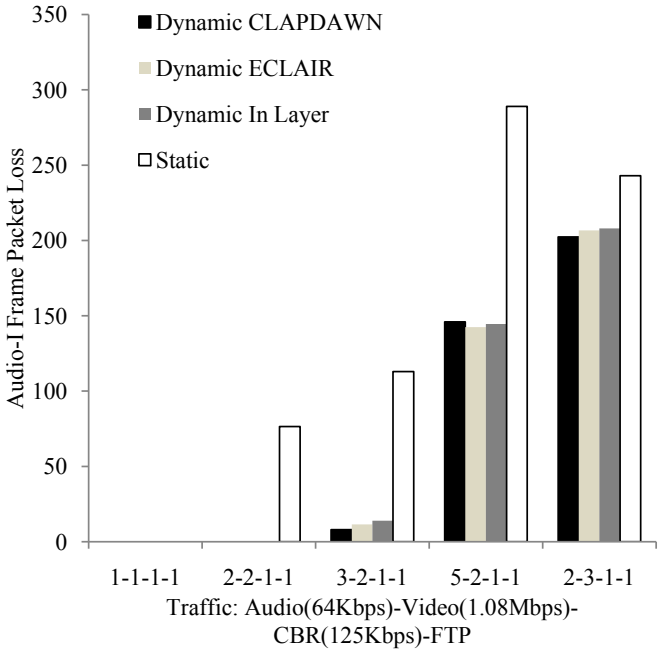


Figure 4.11: I-Frame and Audio packet loss experienced by static method and dynamic method implemented in different architectures

cross layer interaction scenarios. Next result shows the benefit of CLAPDAWN over other architectures, specifically over highly cited ECLAIR.

4.2.2 Simulation study 2 : conflicting cross layer interactions

In the next simulation study we have increased the complexity of the system by introducing multiple conflicting cross layer interactions in the system. This study highlights the importance of uniform representation of cross layer interaction in the cross layer knowledge base (ECA rules) and significance of policy configurations. Result of the study shows how these features improve the stability of the system compared to the other cross layer architecture, ECLAIR.

Our experiment introduces two conflicting cross layer interactions in the system a) network layer interacting with physical layer, in which transmission power is a function of number of neighbors and b) data link layer interacting with physical layer here, transmission power is a function of link quality. In the first cross layer interaction, if a node has higher number of neighbors then it decreases its transmission power, to decrease its interference to other nodes. If it has low number of neighbors it increases its transmission

power so that synchronized effect increases its number of neighbors. In the second case, if the node has low link quality it increases the transmission power to improve the link quality and in the case of high link quality it tries to achieve required performance using low power by reducing the transmission power.

We have performed a set of experiments with the above cross layer interaction rules configured in knowledge base. Performance of the system is analyzed by measuring how many times the monitoring parameter gets a value beyond defined threshold window. Here monitoring parameters are number of neighbor and link quality indicator.

Here, both cross layer interactions calculate value of transmission power based on current parameter values, one using number of neighbor and other using link quality indicator. CLAPDAWN provides various policy configuration to decide transmission power like, a) running average of previous results to configure current transmission power, b) the average transmission power calculated by all the cross layer interactions, c) it gives higher priority to one calculated transmission over the other. We did simulation considering these three policies.

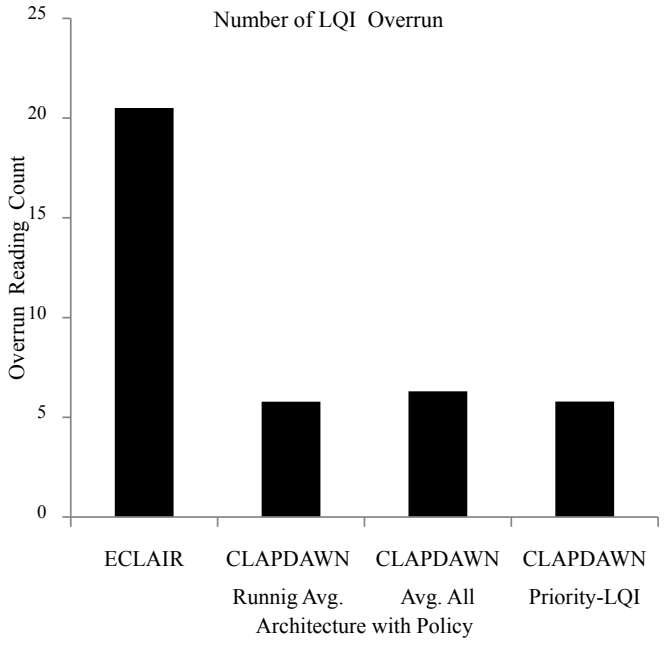


Figure 4.12: Number of Link Quality Indicator readings falling outside the threshold window in ECLAIR and CLAPDAWN with policy configuration

Figure 4.12 and 4.13 show the comparison between the ECLAIR and CLAPDAWN.

It shows that in both the cases ECLAIR reports the higher number of readings beyond the threshold boundary. In the case of CLAPDAWN different policy configurations gives different results. In that, prioritized policy configuration has better results compared to other policy configurations.

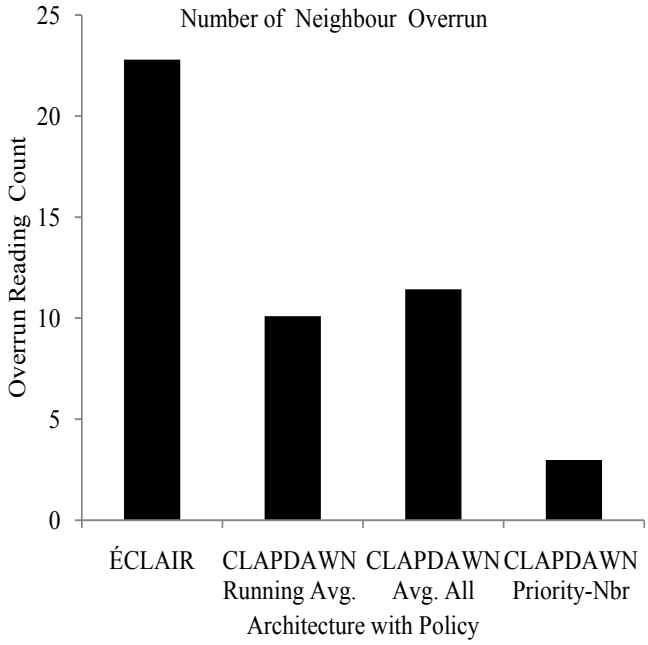


Figure 4.13: Number of neighbors falling outside the threshold window in ECLAIR and CLAPDAWN with policy configuration

In cross layer interaction, secondary control requires information from other layer. It is the responsibility of cross layer architecture to bring required information to required place efficiently. Additional information flow generates the overhead in the system, normally measured in number of messages. Next result shows the messaging overhead experienced by ECLAIR and CLAPDAWN in the above scenario. Secondary control flow has two options asynchronous and synchronous. Considering that we have divided the overhead messages into two parts, one is inflow messages and the other is outflow messages, as shown in figure 4.14 and 4.15 respectively.

Result shows that CLAPDAWN generates lesser overhead in terms of messages compared to ECLAIR. In the ECLAIR each optimizer is implemented separately and generates separate information flow. That increases the number of in and out messages in the ECLAIR and in a way generates higher overhead. CLAPDAWN has single informa-

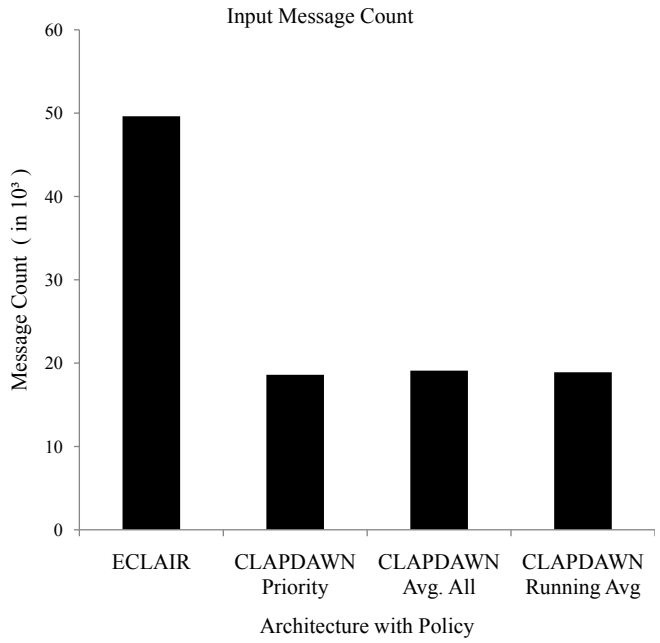


Figure 4.14: Number of input messages required by ECLAIR and CLAPDAWN with policy configuration

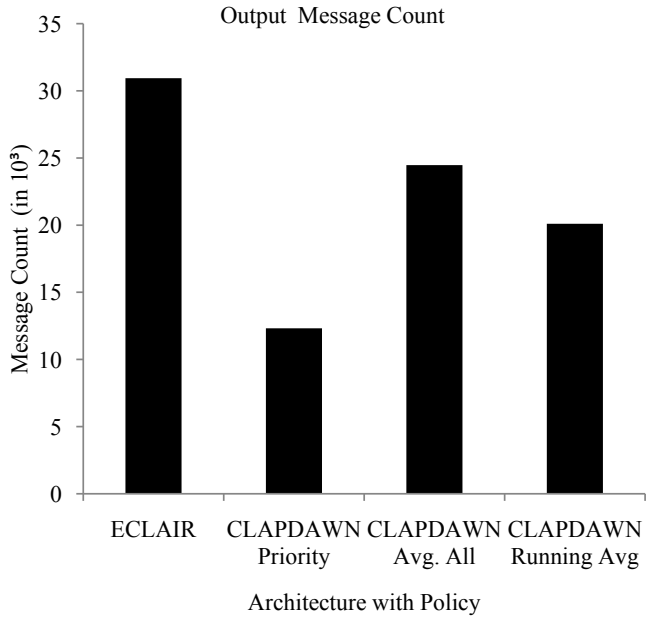


Figure 4.15: Number of output messages required by ECLAIR and CLAPDAWN with policy configuration

tion flow that reduces the number of messages in the network. CLAPDAWN has lesser message overhead compared to ECLAIR. It achieves this lesser overhead at the cost of processing time. Here, on event cross layer engine has to processes all the ECA rules.

That increases the overall response time experienced by the event. Next result shows study of response time in both the architectures, ECLAIR and CLAPDAWN.

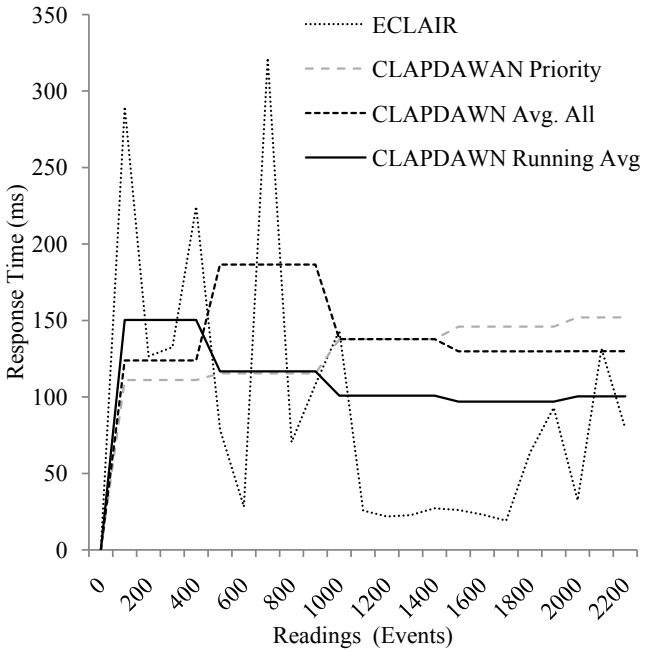


Figure 4.16: Result showing response time experienced by events in ECLAIR and CLAPDAWN with policy configuration

Figure 4.16 shows response time experienced by an individual event. In CLAPDAWN initial configuration takes some time. After initial configuration, additional delay is added by waiting time. At stable condition, after 1000 events result shows that ECLAIR has smaller response time compared to CLAPDAWN. Same can be seen in the average response time result shown in figure 4.17.

In CLAPDAWN, limitation of having high response time can be overcome by increasing the number of execution engines. Our architecture separates CLI logic from its implementation. That allows freedom of increasing number of execution engine in the CLAPDAWN and can reduce the response time.

4.3 Summary

Structure and working of CLAPDAWN highlights many advantages over existing CLA. In TinyCubus, MobileMan and CrossTalk CLAs communication and control parts are

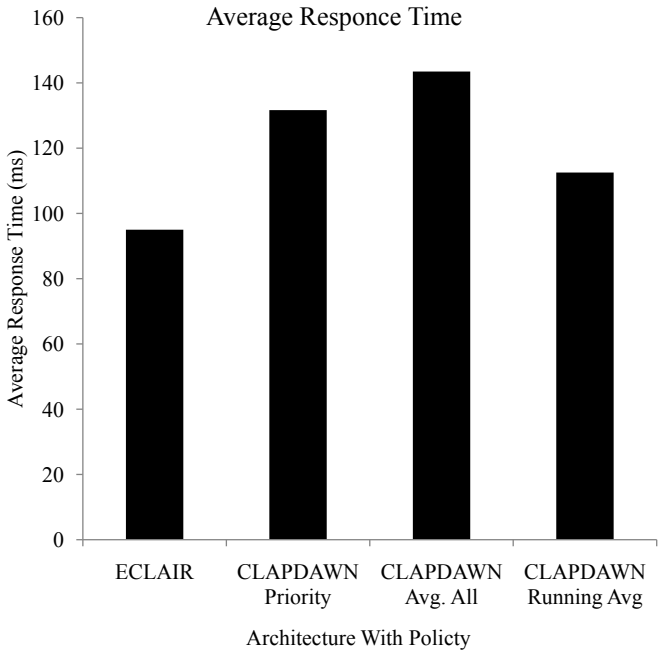


Figure 4.17: Result showing average response time experienced by an event in ECLAIR and CLAPDAWN with policy configuration

tightly coupled together. This tight coupling makes the system complex and hard to track when multiple CLIs take place in the system. CLAPDAWN separates control from communication that gives additional flexibility to the architecture. Due to this communication part remains less affected while introducing or removing cross layer interactions in the control part. That helps in working with multiple cross layer interaction 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. Our architecture tackles this problem and achieves reduced development time by separating CLI execution (execution engine) from its logic (knowledge base). Any modification that takes place in cross layer interaction logic remains transparent to the implementation of that interaction.

CLIs are implemented as ECA rules in knowledge base. Because of that, different cross layer interactions have uniform format that standardizes the process of addition, updating or removal of different CLI in knowledge base. Modularized architecture and standardized structure of CLAPDAWN reduce the development time for many of the

commercial off the shelf component based wireless networks like wireless sensor networks.

TinyCubus, MobileMan, CrossTalk and ECLAIR have distributed control flow for different CLI. This distributed control creates the problem of instability and feedback loop in the system. Knowledge base has information about all the cross layer interactions, which are there in the system. It helps in identifying the possible conflicts and feedback loops which may take place due to multiple cross layer interactions. CLAPDAWN identifies possible conflicts in the system and applies configurable policy mechanism over them to reduce their effect on overall system.

The architecture uses standard interfaces for CP and FP interaction and does not modify any existing messaging structure by adding piggyback information as done in some previous work, CrossTalk. That increases the stability of the architecture and the protocol. Restoring system to its previous state is much easier than the earlier work. Removal of ECA rule from knowledge base restores the system to its previous state and removes the unwanted cross layer interaction from the system.

With all these benefits, CLAPDAWN has higher initial processing time. At first occurrence of the event, execution engine takes some time for rule fetching and configuration. This creates high initial response time. Secondly CLAPDAWN requires more memory compared to other proposed architectures. Execution engine uses configurable policies and local set of parameters for each CLI that increases memory footprint of CLAPDAWN. These problems can be overcome by introducing more execution engines in the system with compact CLI representations.

Chapter 5

IEEE 802.15.4

5.1 Introduction

Wireless Sensor Network(WSN) is resource constrained network developed specially targeting applications having unattended network for long time. Such a network requires energy efficient and energy aware MAC and Physical layer. The previously proposed standards like IEEE 802.11 [37], IEEE 802.16 [38] or IEEE 802.15.1 [39] do not consider the node energy in their design. That makes them inappropriate for WSN and need for new protocol standard.

IEEE 802.15.4 is one of such standards specially developed for low rate, low power networks like WSN. IEEE 802.15.4 [40] is designed considering the energy concerns of sensor nodes. For efficient energy utilization, standard uses various improved techniques like, low-power transmission, small frame size, energy-efficient carrier sense multiple access with collision avoidance (CSMA-CA) algorithm and mechanism to support inactive period. As per the specification IEEE 802.15.4 comes with following features:

- 868 - 868.6 MHz (e.g., Europe)
- 902 - 928 MHz (e.g., North America)
- 2400 - 2483.5 MHz (worldwide)
- 3100 - 10 600 MHz (UWB varies by region)

- Over the air data rates of 851 kb/s, 250 kb/s, 110 kb/s, 100 kb/s, 40 kb/s, and 20 kb/s
- Star or peer-to-peer operation
- Allocated 16-bit short or 64-bit extended addresses
- Optional allocation of guaranteed time slots (GTSs)
- Carrier sense multiple access with collision avoidance (CSMA-CA) channel access
- Fully acknowledged protocol for transfer reliability
- Low power consumption and Energy detection (ED)
- Link quality indication (LQI)
- 16 channels in the 2450 MHz band, 30 channels in the 915 MHz band, and 3 channels in the 868 MHz band , 14 overlapping chirp spread spectrum (CSS) channels in the 2450 MHz band, and 16 channels in three UWB bands (500 MHz and 3.1 GHz)

As mentioned above IEEE 802.15.4 is designed for network devices with low power consumption and data low rate applications. The standard specifies bottom two layers of network protocol stack, MAC layer and physical layer. It divides network nodes into three groups, personal area coordinator (PAN coordinator), coordinator nodes and device nodes.

PAN coordinator is responsible for creating and managing the network. Network coordinators are responsible for node synchronization using beacons and providing the routing functionality to device nodes. Lastly, device nodes are responsible for collecting the information in the network and forwarding it to network coordinators or PAN coordinator .

As per the standard PAN coordinator and coordinator nodes are categorized as Full Function Device (FFD) nodes while the device nodes may be categorized as Reduced Function Device (RFD) nodes or FFD. Here FFD spends more time in active state and

consumes more energy while RFD spends more time in sleep state and consumes less energy compared to FFD.

5.2 Beacon Enabled Mode

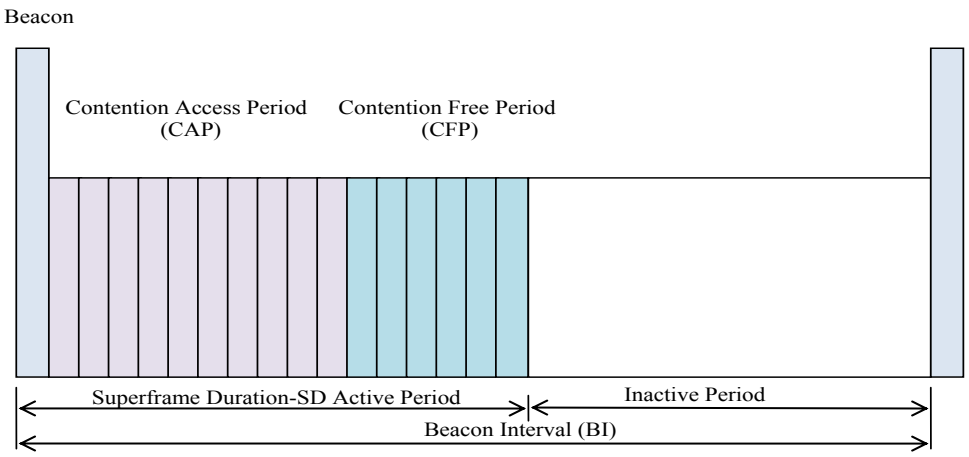


Figure 5.1: SISD Organization

IEEE 802.15.4 comes with two working modes beacon enabled mode and non-beacon mode. In beacon enabled mode of IEEE 802.15.4, Full Function Devices (FFD, PAN coordinator and coordinator nodes) periodically transmit beacons in the network. A network node uses these beacons for network synchronization and to convey the superframe structure to other nodes in the network. Beacon contains the superframe structure through which full function device informs the neighboring nodes about the active and inactive part. The time period in which, the node sends it's beacon is called Beacon Interval(BI). Standard divides the BI into two parts active part and inactive parts as shown in figure 5.1. During inactive part nodes configures themselves to be in sleep state and save their energy. During active part nodes wake up from sleep state and communicate with the coordinator node.

As per the standard, active part consists of three components, 1) beacon 2) Contention-Access Period (CAP) and 3) Contention-Free Period (CFP) . A device using beacon tracking gets active-sleep durations from coordinator, when it receives beacon for the first time. Then it synchronizes its own schedule with the coordinator's and receives

the next beacon periodically by turning on the receiver just before the expected beacon transmission time. The active part is divided into 16 slots from which maximum of 7 slots can be allocated to the CFP while other are allocated to CAP.

In CAP, those nodes which want to communicate with coordinator contend for the media using CSMA-CA algorithm explained in the next section. On successful completion of CSMA-CA algorithm the contending node communicates with the coordinator or vice versa. Beacon carries the information about assignment of guaranteed time slot and the direction of the communication in the CFP. That makes the assigned node to start communication without using CSMA-CA in CFP duration. Following are the timing specifications for the superframe in beacon enabled mode :

$$BI = aBaseSuperframeDuration \times 2^{BCO} \tag{5.1}$$

$$SD = aBaseSuperframeDuration \times 2^{SFO} \tag{5.2}$$

$$0 \leq SFO \leq BCO \leq 14 \tag{5.3}$$

$$aBaseSuperframeDuration \ = \ 960 \ symbols \ = \ 15.36 \ ms \tag{5.4}$$

$$backoff \ slot \ = \ 20 \ symbols \ = \ 320 \ \mu s \ = \ 10 \ Bytes \tag{5.5}$$

As mentioned above, in beacon enabled mode coordinator node periodically sends the beacon. The period between two consecutive beacons is configured by the Beacon Interval (BI) parameter and the active and inactive parts are configured using Superframe Duration (SD) , defined in equation 5.1 and 5.2 respectively. Further the relation between the Superframe Order (SFO) and Beacon Order(BCO) is defined by equation 5.3. These values are based on 2.45GHz specification.

In beacon enabled mode coordinator node periodically sends the beacon and syn-

chronizes the neighboring network nodes. Beacon also contains the information about Guaranteed Time Slot(GTS) , which contains the node address list for each allocated slot with the packet flow direction. Nodes in previous beacon interval reserve the slot for next Contention Free Period (CFP). That adds additional delay in packet transmission with CFP as reservation is done using CSMA-CA algorithm. Another draw back that CFP faces is of limited number of slots. In a particular beacon, maximum 7 slots can be allocated to GTS. With this a maximum of 7 nodes can send their packets in CFP. It provides a sense of priority in packet transmission, considering that in CFP the allocated slot is reserved for the allocated nodes. But it adds additional delay of slot reservation and has limited number of slots for the contending nodes.

In CAP, network nodes contend for the channel using CSMA-CA algorithm and in CFP, selected nodes goes for packet transmission without using CSMA-CA algorithm. Standard specifies that CFP is designed for applications requiring guaranteed data delivery or having specific bandwidth demand. But as GTS has its own drawback it is not serving its purpose fully.

5.3 CSMA-CA

CSMA-CA stands for the Carrier Sense Multiple Access-Collision Avoidance. IEEE 802.15.4 uses slotted CSMA-CA algorithm for channel access before sending command or data packets in CAP of beacon mode. This section covers CSMA-CA algorithm used by IEEE 802.15.4 for channel access in beacon enabled mode [41]. CSMA-CA algorithm for non-beacon mode is slightly different than the beacon enabled mode. Here we limit our scope to beacon enabled mode only and do not go into details of non-beacon mode of the standard.

On arrival of packet from higher layer or command from MAC layer, node triggers the CSMA-CA algorithm for channel access. The backoff boundaries of every device in network are aligned with superframe slot boundaries of the PAN coordinator. Each time device wishes to access the channel, it must locate the boundary for the next slot period.

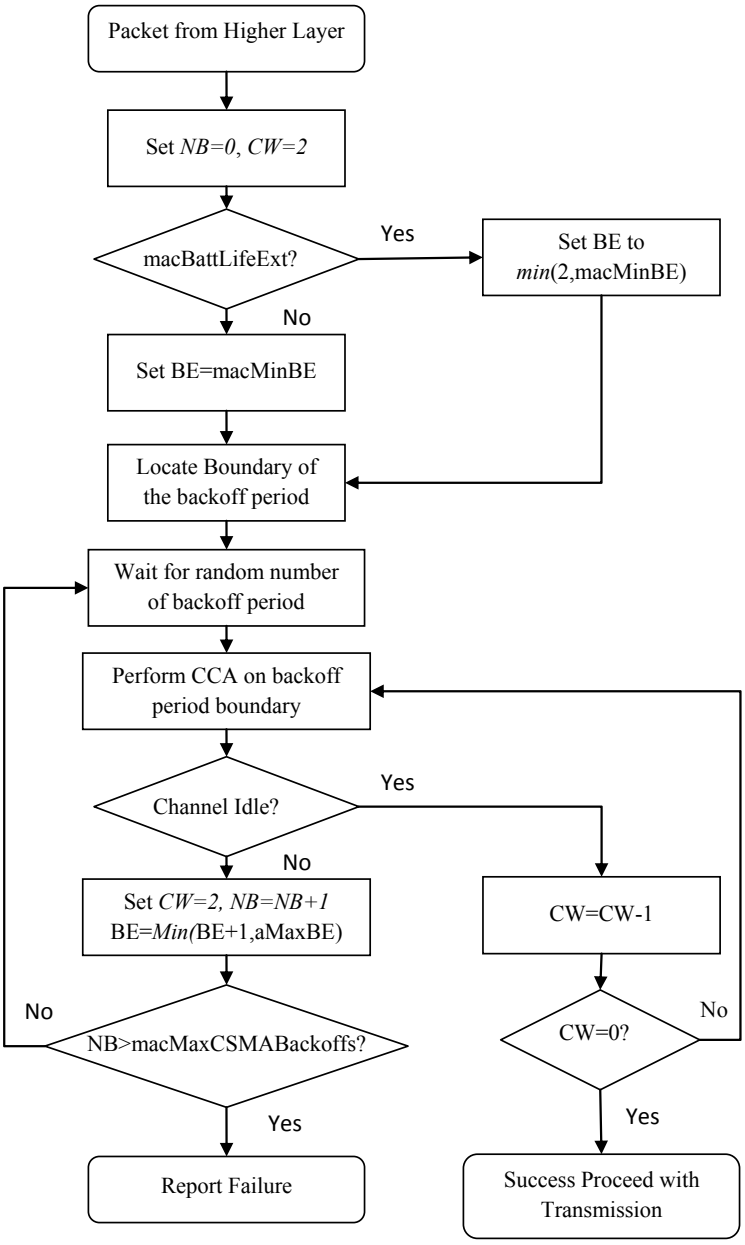


Figure 5.2: Carrier Sense Multiple Access-Collision Avoidance(CSMA-CA) algorithm for IEEE 802.15.4

Before accessing the channel, a random number of backoff slots should be waited. During this time, the device remains in idle state and does not scan the channel that helps in saving energy. After the random delay, a two slot clear channel assessment is carried out. Algorithm steps for CSMA-CA are shown in figure 5.2 and explained below.

Mainly there are three variables that every node in the network maintains for the CSMA-CA algorithm, Number of Backoffs(NB), Contention Window (CW) and Backoff exponent (BE) . NB is the number of backoff attempts that a node is allowed to get the channel for the packet transmission. CW represents the number of clear channel assessments(CCAs) node performs before going for channel access. BE is used to calculate the maximum number of slot periods a device must wait before attempting to access the channel. With slotted CSMA-CA, the MAC sublayer shall first initialize NB, CW, and BE and then locate the boundary of the next backoff period. The MAC sublayer shall delay for a random number of complete backoff periods in the range 0 to $2^{BE} - 1$ and then request PHY to perform CCA. In a slotted CSMA-CA system, the CCA shall start at backoff period boundary.

In a slotted CSMA-CA system with the BLE (Battery Life Extension) subfield set to zero, the MAC sublayer shall ensure that, after the random backoff, the remaining CSMA-CA operations can be undertaken and the entire transaction must be transmitted before the end of the CAP. If the number of backoff periods is greater than the remaining number of backoff periods in the CAP, the MAC sublayer shall pause the backoff countdown at the end of the CAP and resume it at the start of the CAP in the next superframe. If the number of backoff periods is less than or equal to the remaining number of backoff periods in the CAP, the MAC sublayer shall apply it's backoff delay and then evaluate whether it can proceed. The MAC sublayer shall proceed if the remaining CSMA-CA algorithm steps (i.e., two CCA), the frame transmission, and any acknowledgment can be completed before the end of the CAP. If the MAC sublayer can proceed, it shall request that the PHY perform the CCA in the current superframe. If the MAC sublayer cannot proceed, it shall wait until the start of the CAP in the next superframe and apply a further random backoff delay before evaluating whether it can proceed again.

In a slotted CSMA-CA system with the BLE(Battery Life Extension) subfield set to one, the MAC sublayer shall ensure that, after the random backoff, the remaining CSMA-CA operations can be undertaken and the entire transaction can be transmitted before the end of the CAP. The backoff countdown shall only occur during the first macBattLifeExtPeriods(backoff periods during which the receiver is enabled) full backoff period after the end of the IFS period following the beacon. The MAC sublayer shall proceed if the remaining CSMA-CA algorithm steps (two CCA analyses), the frame transmission, and any acknowledgment can be completed before the end of the CAP, and the frame transmission will start in one of the first macBattLifeExtPeriods full backoff periods after the IFS period following the beacon.

If the MAC sublayer can proceed, it shall request that the PHY perform the CCA in the current superframe. If the MAC sublayer cannot proceed, it shall wait until the start of the CAP in the next superframe and apply a further random backoff delay before evaluating whether it can proceed again. If the channel is assessed to be busy, the MAC sublayer shall increment both NB and BE by one, ensuring that BE shall be no more than macMaxBE(the maximum value of the backoff exponent,BE). The MAC sublayer in a slotted CSMA-CA system shall also reset CW to two. If the value of NB is less than or equal to macMaxCSMABackoffs(the maximum number of backoff attempts before declaring access failure), the CSMA-CA algorithm shall return to random backoff step. If the value of NB is greater than macMaxCSMABackoffs, the CSMA-CA algorithm shall terminate with a channel access failure status.

If the channel is assessed to be idle, the MAC sublayer in a slotted CSMA-CA system shall ensure that the contention window has expired before commencing transmission. To do this, the MAC sublayer shall first decrement CW by one and then determine whether it is equal to zero. If it is not equal to zero, the CSMA-CA algorithm shall return channel scan step. If it is equal to zero, the MAC sublayer shall begin transmission of the frame on the boundary of the next backoff period.

5.4 Issues with IEEE 802.15.4

IEEE 802.15.4 standard provides MAC and Physical layer specifications for energy efficient low power, low rate wireless networks, but there are points which remain untouched in the specification or specification does not give clear guidelines on them. In this section, we will go through those issues which are still seeking solution and not attained properly in literature and standard specification.

5.4.1 Beacon collision

IEEE 802.15.4 standard suggests two network topologies: star topology and peer-to-peer topology. As nodes are equipped with low range radio modules, network needs to run in multi-hop mode. Here, as per the standard, network should be organized as a hierarchical structure. In this hierarchical structure, PAN coordinator works as root and at each level the coordinator node collects the information and forward it towards the PAN coordinator.

With beacon enabled mode, coordinator nodes periodically broadcast their beacons. As per the IEEE 802.15.4 specification to make the network synchronized these beacons are broadcasted without using CSMA-CA algorithm. Due to this it may happen that these beacons collide with other packets in the network and get dropped. If this beacon loss occurs due to other non beacon packet then network can recover from that. But it may happen that two coordinators are within each others communication range and they broadcast the beacon which collide then it is hard to recover from such beacon loss.

As the IEEE 802.15.4 is mainly designed for the general-purpose low power networks. The designed mechanism works well where all the devices are within one hop range (star network), but by itself it is not sufficient for the multi-hop case (peer to peer network) used in low power networks. In such cases beacon loss creates serious synchronization problem in the network.

5.4.2 Boundary effect

As per the standard, in beacon enabled mode, node uses random backoff time followed by the two CCA before getting access to the channel. If it finds channel to be free in two CCA it will go to next step of packet transmission. Here node first calculates the time required to turnaround its radio, data packet transmission and time to get acknowledgement. If this time exceeds the current CAP boundary then node postpones the packet transmission to next super frame. In the next superframe at the beginning of first active slot after the beacon it goes for the packet transmission.

In such cases it may happen that more than one node goes through the same steps and ends up into the packet collision. This effect is called boundary effect in IEEE 802.15.4 protocol. A similar problem may occur when the coordinator has pending packets for two or more of the nodes. Nodes that recognize their address in the beacon frame and have no pending uplink data transmission in progress will immediately start the CSMA/CA algorithm. As the initial back off is small (0 to 7), it may happen that two or more nodes choose the same backoff and fall into collision.

5.4.3 Priority in IEEE 802.15.4

Normally protocols are designed considering that they need to maintain fairness among the network nodes. IEEE 802.15.4 is also designed based on same guidelines. But there are many cases in which the deployed sensor network needs to send critical information urgently and overrule the fairness constraint. IEEE 802.15.4 comes with GTS in beacon enabled mode which can be used to send prioritized data in the network. As GTS has its own limitations in terms of number of slots and reservation delay, GTS is not appropriate for priority implementation. Use of IEEE 802.15.4 in critical information forwarding requires additional support.

5.4.4 Turnaround time and CCA time

IEEE 802.15.4 works with slotted CSMA-CA and unslotted CSMA-CA mode. In both cases, backoff counter is decremented to zero regardless of the channel status. When the backoff counter reaches zero, a device performs CCA once for the unslotted CSMA-CA and twice for the slotted CSMA-CA in order to check whether the channel is busy or not. If it finds the channel idle during CCA time, the device transmits its data packet. When it finds channel busy during the period, it increments BE by one and repeats the random backoff procedure.

Particularly in unslotted case with acknowledgement where node only performs one CCA check, if CCA time is not set properly it creates problem with the acknowledgement packet. Normally, on receiving data packet node sends the acknowledgement packet in case the received data packet requires the acknowledgement. To do so it turns around its radio from receiving mode to transmitting mode and sends the acknowledgement packet. Meanwhile if any device is performing CCA it finds the channel idle and goes for packet transmission. If the CCA time is less than the turnaround time, then while the receiving node is turning around its radio to transmit state the sensing node sees the channel free and transmits its packet. As the ACK packet is not sent using CSMA-CA, it collides with the node transmitting the packet. To avoid this the CCA time should be configured properly so that it should be greater than the turnaround time. Another case is when the CCA start time of two nodes is less than the turn around time. In this case both the nodes see the channel idle and go for transmission.

5.4.5 Non-Beacon mode idle listening

The ideal approach to maximize battery life is to turn the transceiver on only when it transmits and receives a frame. Not knowing the exact times for frame transmission and reception, a certain amount of overhead is inevitable in a sensor network environment. The major overhead of a receiving scheme is periodic idle listening for possible frame reception. Here, wakeup interval and active duration determine consumption energy in

non-beacon mode. The former is how often and the latter is for how long the device wakes up. The overhead for a transmitting scheme is mainly due to waiting time for a receiver's wakeup schedule. The IEEE 802.15.4 does not give clear guidance on managing the resources in non-beacon mode and it is open for the final network design.

5.4.6 Use of limited GTS and use of BI, SD

As per the specification, maximum 7 Guaranteed Time Slots(GTS) can be accommodated in a given superframe. To make it more precise at most seven nodes with one slot per node can communicate with coordinator with certain guarantee. In the bursty dense network with multiple flows, it might be the case that there may be more than seven nodes contending for the guaranteed slots. Scheduling of these limited resources depends on network design. Further to make space for GTS, designer selects the BI and SD with smaller CAP time. This comes with the unfairness to the other nodes who are contending for the slot in CAP time, making CAP small makes these nodes to wait longer and increases the data delivery latency.

5.4.7 Blocking problem

With beacon enabled mode, when coordinator has packet for node it announces it in beacon with pending address. On receiving the beacon, node knows that it has pending data and it sends the data request to coordinator. On receiving data request from node, the coordinator node performs the CSMA-CA algorithm before sending the packet. During the backoff time of CSMA-CA coordinator does not receive any packet and puts the network into block state. Due to this blocking, the pending data transfers are delayed and builds up the queue at network nodes. In heavy traffic condition, this blocking may end-up in packet overflow and packet loss.

5.4.8 Topology Control Problem

Normally, IEEE 802.15.4 based wireless sensor networks are deployed randomly having homogeneous sensor nodes. At the run time as per the network's requirement these sensor nodes are configured as FFD(PAN Coordinator and Coordinator Nodes) or RFD (Device Nodes). The IEEE 802.15.4 classifies the network nodes and their functionality, but the configuration of node depends on network deployment and requirement. Configuring wireless sensor network nodes as FFD or RFD nodes based on network requirement is a topology control problem in wireless sensor networks.

5.5 Summary

In this chapter we have gone through the important features of IEEE 802.15.4 communication standards. The chapter highlights problems encountered by IEEE 802.15.4. From this set of problems we have worked on two wireless sensor network problems a) priority in IEEE 802.15.4 and b) topology control problem and modeled it into CLAPDAWN as proof of concept. Next part of the thesis discusses these problems in depth with our proposed solutions and their modeling in CLAPDAWN. Chapters 6 and 7 cover the prioritized channel access problem based on IEEE 802.15.4 event driven wireless sensor networks and chapter 8 and 9 covers the topology control problem.

Chapter 6

Event Driven Wireless Sensor Networks

6.1 Introduction

Wireless Sensor Networks are application specific, data centric networks having application span ranging from household applications to mission critical applications. Event driven application form an important class of applications of wireless sensor networks. Intrusion detection and fire detection are examples of event driven wireless sensor network applications. Here, one of the desired goals is that on occurrence of critical events, the network should notify the base station about those events with minimum delay. Secondly, wireless sensor networks are specifically designed targeting applications having unattended nodes for long time. Such a network requires energy efficient and energy aware MAC and Physical layer. It creates the tradeoff between energy consumption and delay. With this tradeoff it is hard to achieve the desired goal of event driven applications.

IEEE 802.15.4 is a communication standard specifically designed, targeting such resource constrained networks. IEEE 802.15.4 provides MAC and Physical layer specifications for low power personal area networks like wireless sensor networks. Chapter 5 covers IEEE 802.15.4 and its problems, details about IEEE 802.15.4 can be found in [42, 41]. A brief survey of well known MAC protocols designed for WSNs is available in [43].

Generally such MAC specifications are designed considering fairness as one of their design considerations. But cases like event driven applications require some biasing to handle critical events with higher consideration. To handle this, IEEE 802.15.4 provides the Guaranteed Time Slot (GTS) mechanism. Here, nodes reserve one or more GTS in the next superframe to achieve guaranteed data delivery with coordinator nodes. The GTS mechanism has its own disadvantages. First, it creates an additional delay for slot reservation. Second, the number of GTS per superfame is limited. Lastly changing the number of GTS increases active time of devices and decreases the network lifetime. Use of GTS in event driven applications will not solve the problem at its depth.

As explained in the above paragraph, GTS will not help much in handling critical events in event driven applications. Contention Access Period (CAP), the other part of a superframe also does not have any mechanism to handle critical events. Further, CAP uses the CSMA-CA algorithm which is known to be unfair in certain scenarios [44, 45]. Considering these limitations, we have worked on IEEE 802.15.4 to provide support for a prioritized channel access mechanism for wireless sensor networks.

In our work we have focused on introducing the application level priority in the IEEE 802.15.4 MAC layer so that in CAP with beacon enabled mode, nodes have higher chances of delivering urgent information to the base station. To achieve this, we have proposed explicit priority and implicit priority based channel access mechanisms for beacon enabled mode of IEEE 802.15.4.

6.2 Related Work

Several approaches based on simulation and experiments have been proposed in the literature for performance evaluation of IEEE 802.15.4. On those lines [46, 47, 48] have provided simulation based experimental studies of IEEE 802.15.4 to analyze association, delay performance and collision in the network. Lu et al. [49] performed simulation based study of throughput and energy efficiency of IEEE 802.15.4. However, instead of relying on simulation results and lengthy experiments, it is of great interest to determine the

analytical model that can help in designing the network.

In his pioneering work, Binachi [50] has proposed an accurate two dimensional Markov chain based model for the binary exponential backoff protocol of IEEE 802.11. He has analyzed and computed the IEEE 802.11 DCF saturated throughput. In modeling of IEEE 802.15.4, [51, 52, 53, 54, 55, 56] have got their initial inspiration from Binachi [50]. A very detailed analysis of beacon enabled clustered IEEE 802.15.4 is given in [54, 53]. Here, Misic et al. have modeled the working of IEEE 802.15.4 with great detail which has made the model cumbersome and difficult to follow. They have modeled the packet queues in the device data buffer and request buffer as an M/G/1 queuing system. Here each state is modeled based on the counter value as done in [50]. With this they have modeled the MAC sublayer for both downlink and uplink traffic in beacon enabled mode of IEEE 802.15.4.

Ramachandran et al. [57] presented the analysis of IEEE 802.15.4 beacon-enabled mode under non-saturation traffic. They simplified their model by using geometric distribution instead of uniform distribution for the random backoff stage. Their result shows that with no acknowledgement, initializing the contention window (CW) length to 1 helps in improving the throughput and reduces the energy consumption. Park T.R et al. [52] have analyzed IEEE 802.15.4 using Markov chains for throughput and energy consumption in saturated traffic condition. Their proposed model utilizes the probability of a device in the channel sensing state instead of the channel accessing state.

Xinhua Ling et al. [58] have proposed an accurate analytical model for the IEEE 802.15.4 MAC protocol. In their work instead of modeling the channel they have modeled behavior of an individual node based on a three-level renewal process. Chiara et. al. [56] has modeled star topology based non-beacon enabled mode of IEEE 802.15.4 and derived success probability for the transmission of a packet in a round and the mean energy consumed by a node during a round. To make the protocol simple they have assumed that all the nodes can hear each other. Kim et al. [52] have proposed simple mathematical model for non-beacon enabled mode of IEEE 802.15.4. With their model they have derived the appropriate number of nodes for required QoS constraints like

average packet delay, packet loss probability and battery life.

As discussed above several approaches based on simulations and experiments have been proposed in the literature for performance evaluation of IEEE 802.15.4. Along those lines [46, 47, 59, 49] undertake the simulation based experimental study of IEEE 802.15.4 and [51, 52, 53, 54, 55, 57] provide modeling of IEEE 802.15.4 inspired by Binachi [50]. The literature mainly discusses IEEE 802.15.4 in a normal working environment. Very few papers [60, 61, 62, 63, 64] discuss the prioritized working of IEEE 802.15.4. And those which talk about prioritized IEEE 802.15.4, only give simulation based study. In their analytical study Kima et. al. [64] assume that the priority of clear channel assessment is same in both the scans which is not a valid assumption. In ideal conditions the second scan depends on the cumulative effect of first scan perform by all the nodes in the network. To the best of our knowledge proper analytical study of prioritized IEEE 802.15.4 has not been conducted extensively.

We address the problem of critical event handling in IEEE 802.15.4 based wireless sensor networks. For this we propose two protocols; the first protocol is Explicit Prioritized Channel Access Protocol (EPCAP) and the second is Implicit Prioritized Channel Access Protocol (IPCAP). Many of the previous works like [56], assume that all the nodes are in each others communication range which makes them impractical in real world scenarios. IPCAP and EPCAP do not assume such strict conditions. Further, rather than giving the prioritized protocol and comparing it with previous work, our objective is to provide correct analytical model for prioritized event handling protocol working on IEEE 802.15.4. In our work we model the working of IPCAP as a two dimensional Markov chain and EPCAP as M/G/c queuing model.

To achieve the desired goal, we have exploited disadvantages of CSMA-CA in handling critical events in event driven applications, by designing Implicit Prioritized Channel Access Protocol (IPCAP) for beacon enabled mode of IEEE 802.15.4. IPCAP helps devices to deliver critical events prior to normal events. IPCAP works with the CSMA-CA algorithm. We have modeled the working of tampered CSMA-CA as a bidimensional Markov chain and derived equations for various network properties.

The second protocol that we propose is Explicit Prioritized Channel Access Protocol (EPCAP) for beacon enabled mode of IEEE 802.15.4. EPCAP also helps devices deliver critical events prior to normal events. EPCAP achieves this by using secondary beacon. We have modeled the working of EPCAP as M/G/c queuing model and derived equations for various network properties. Here, IPCAP and EPCAP assume rational behavior of all the nodes and do not consider the case of having malicious nodes in the network.

6.3 Implicit Prioritized Channel Access Protocol (IPCAP)

Consider a scenario having a set of coordinator nodes and a set of device nodes. Detailed working of coordinator nodes and device nodes can be found in [42, 41]. Device nodes need to convey their information to coordinator nodes. Among these, some device nodes are transmitting periodic routine information while some device nodes have urgent information that requires immediate attention from a coordinator node. It is the responsibility of IPCAP to deliver urgent events prior to normal events. In IPCAP nodes locally configure their CSMA-CA parameters such that the high priority nodes have higher chances of getting channel access compared to low priority nodes.

Table 6.1: Configuration Parameters for Priority Class

	$minBE$	$maxBE$	ρ	$maxNB$
Class1	2	5	1.5	9
Class2	2	5	1.8	7
Class3	3	5	2	5
Class4	3	5	2.3	4
Class5	4	5	2.5	4

Table 6.2: CSMA/CA Default parameter values

Parameter	Default Value
$minBE$	2
$maxBE$	5
$maxNB$	5
Increment Factor ρ	2

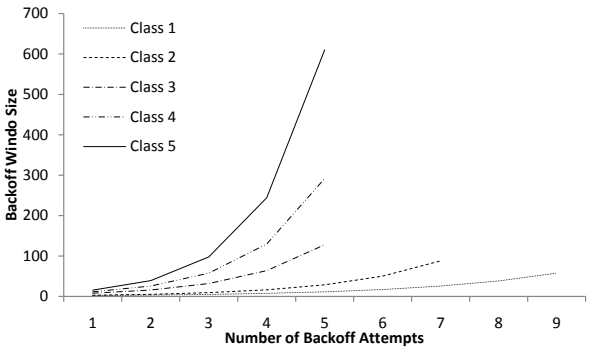


Figure 6.1: Priority Classes

In CAP period of beacon-enabled mode, whenever a node has a packet to send for coordinator it triggers the slotted CSMA-CA algorithm for channel access. In the standard CSMA-CA algorithm, the node first sets the NB to zero and BE to minBE value then it chooses a random backoff interval using the current BE value. After finishing the backoff interval, the node checks whether the channel is idle or not. If it finds the channel idle two times it will go ahead with the communication. In case it finds the channel busy it will increment the value of BE and NB and perform the same steps again until the NB reaches to its allowed limit which is by default set to five.

In normal scenarios all the nodes follow the above algorithm. To implement priority scheme, IPCAP modifies the algorithm so that a node goes for the channel access based on its priority level. High priority nodes will get more chances of channel access with configured values of BE and maxBE. Figure 6.1 shows its backoff boundary and relation between various priority classes. Figure 6.1 has five classes and maximum 9 retries for channel access for certain priority class, value for each class is given in table 6.1. Figure 6.1, shows the backoff interval of various priority classes with respect to their attempt.

Here, a high priority class quickly goes for more attempts while a low priority class delays its attempts of channel access. Based on this configuration, there are higher chances of getting channel access by high priority classes compared to low priority classes. That enables higher priority class to deliver its information more quickly than the low priority classes. And in event driven applications critical event gets delivered faster than normal events.

6.3.1 Markov model for IPCAP

Table 6.2 shows default values for some of the key parameters of IEEE 802.15.4 standard. The standard has constant BE-increasing factor, having value 2. Each time it doubles the backoff exponent. With available floating point arithmetic unit and rate at which IEEE 802.15.4 operates, use of non integer BE-increasing factor does not constitute a major hurdle. Considering this, IPCAP uses the non-integer value to fine tune the BE-increasing factor ρ.

Based on their importance, events are divided into different priority classes. Here, each priority class has a different configuration for CSMA-CA parameters. The class with higher priority has lower number and has higher chances of getting the channel. There are $N(0, 1, .., N-1)$ priority classes in the system. These priority classes are differentiated from each other by $minBE_i$, $maxBE_i$, $maxNB_i$ and BE-increasing factor ρ_i. Two classes i and j are said to be $i < j$, $class_i$ has higher priority than $class_j$, if one of the following inequalities hold: $minBE_i < minBE_j$, $maxBE_i > maxBE_j$ or $\rho_i < \rho_j$. The example shown in figure 6.1 has five priority classes indexed from 1 to 5. Here, class 1 has the highest priority and class 5 has the lowest. Table 6.1 shows a sample configuration for these priority classes and table 6.2 shows the default values for their variables.

For class i, let W_j^i be the backoff window size in the j^{th} backoff attempt, W_{max}^i be the maximum backoff window size and let $W_0^i = \rho_i^{minBE} = W_{min}^i$. The relation between W_j^i, $maxNB_i$, W_{min}^i, W_{max}^i and ρ_i is as below, where $b_i = log_{\rho_i}(W_{max}^i / W_{min}^i)$,

$$\begin{aligned} W_j^i &= (\rho_i)^j W_0^i \\ &= (\rho_i)^j W_{min}^i \end{aligned} \tag{6.1}$$

$$\text{for } j = 0, 1, .., b_i - 1 \text{ if } maxNB_i > b_i - 1.$$

$$\begin{aligned} W_j^i &= (\rho_i)^{b_i} W_0^i = (\rho_i)^{b_i} W_{min}^i \\ &= W_{max}^i \end{aligned} \tag{6.2}$$

for $j = b_i, .., maxNB^i$ if $maxNB_i > b_i - 1$.

$$\begin{aligned} W_j^i &= (\rho_i)^j W_0^i \\ &= (\rho_i)^j W_{min}^i \end{aligned} \tag{6.3}$$

for $j = 0, 1, .., b_i - 1$ if $maxNB_i \leq b_i - 1$.

In this notation the subscript represents the priority class. In cases where subscript and superscript are both present, the superscript represents the priority class.

IPCAP models the tampered CSMA-CA algorithm as a bidimensional Markov chain as shown in figure 6.2 inspired from Binachi's [50] work. For a given station in the priority class $i \in \{0, 1, .., N-1\}$, let $b(i,t)$ be a random process representing the backoff counter value at time t, and let $s(i,t)$ be a random process representing the backoff stage at time t. The value of the backoff counter $b(i,t)$ is uniformly chosen in the range $0, 1, .., W_j^i - 1$, where $W_j^i = (\rho_i)^j W_0^i$. The bidimensional random process $\{s(j,t), b(i,t)\}$ is a discrete-time Markov chain. The state of each device in priority class i is described by the triple (i, j, k), where j stands for the backoff stage taking values from $j = 0, 1, ..., maxNB_i$ and k stands for the backoff delay taking values from $0, 1, ..., W_j^i - 1$ in timeslots. Equations 6.4 to 6.15 give the state transition probabilities for the Markov chain shown in figure 6.2.

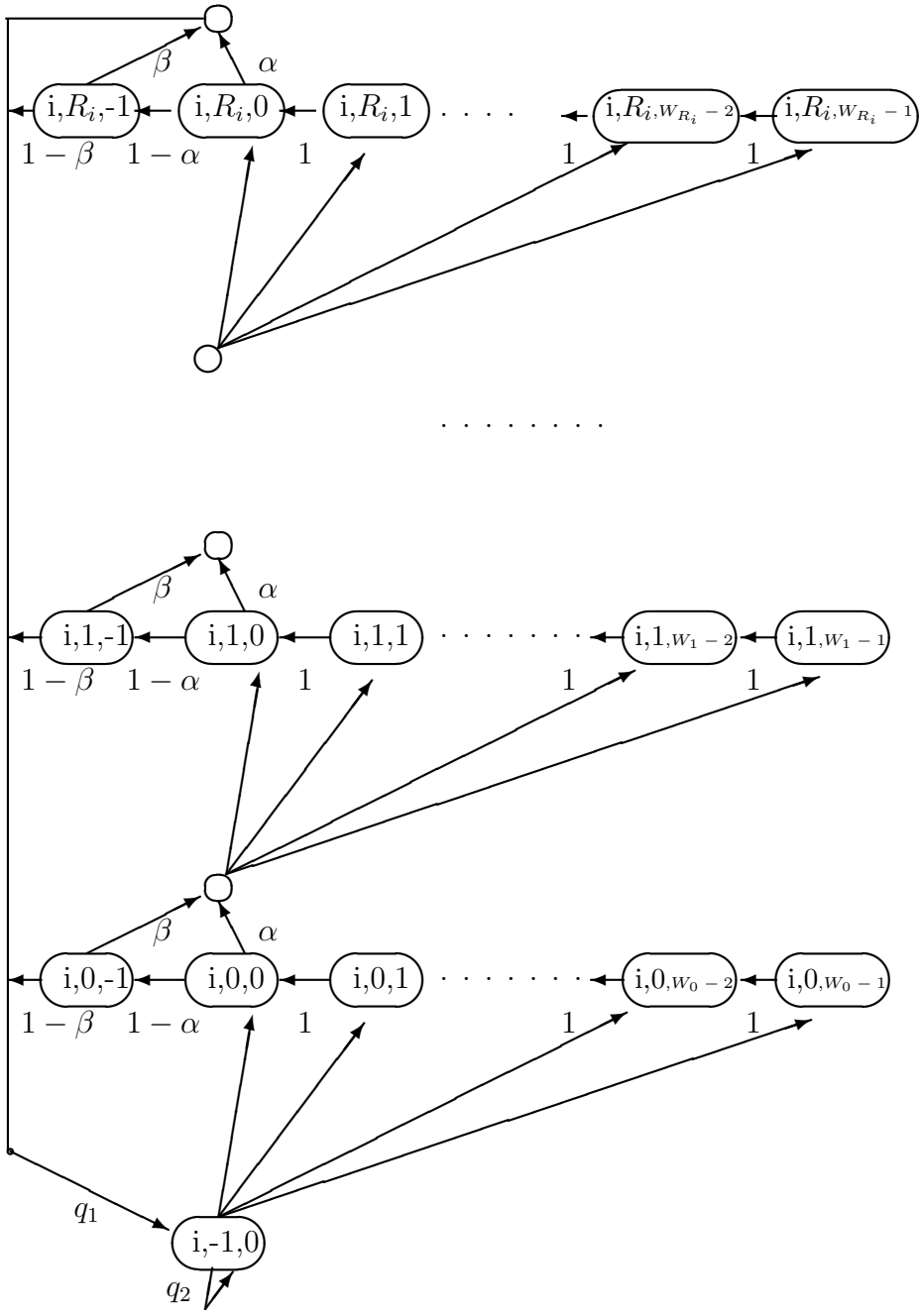


Figure 6.2: Two Dimensional Markov Model for Implicit Priority Scheme

$$P\{i, j, k/i, j, k+1\} = 1 \tag{6.4}$$

Equation 6.4 gives the state transition probability from the $(k+1)^{th}$ backoff stage to the k^{th} backoff stage at the j^{th} retry for $class_i$. On each timer tick the device moves from backoff stage $k+1$ to backoff state k with probability one.

In CSMA-CA algorithm after completing the backoff count, the node goes for the first CCA. Let, α is the probability of getting channel busy in the first CCA slot. Hence, with $1-\alpha$ probability the node finds the channel free in the first CCA slot and goes for the second CCA. Here, the node goes from stage $(i, j, 0)$ to $(i, j, -1)$ with probability $1-\alpha$ as shown in equation 6.5.

$$P\{i, j, -1/i, j, 0\} = 1 - \alpha \tag{6.5}$$

$$P\{i, j, k/i, j-1, 0\} = \alpha/W_j^i \tag{6.6}$$

Here, W_j^i is the maximum backoff window size at j^{th} retry for priority class i. The node finds the channel busy in the first CCA slot of the $(j-1)^{th}$ retry and goes for next retry with probability α/W_j^i as given in equation 6.6. Here with probability $1/W_j^i$, the node selects the next backoff value.

$$P\{i, j, k/i, j-1, -1\} = \beta/W_j^i \tag{6.7}$$

Equation 6.7, β/W_j^i gives the state transition probability to move from the $(j-1)^{th}$ retry state to one of the backoff states in the j^{th} retry after finding the channel busy in the second CCA attempt. Here β is the probability of getting the channel busy in the second CCA attempt.

$$P\{i, 0, k/i, j, -1\} = (1-\beta)(1-q_1)/W_0^i \tag{6.8}$$

Equation 6.8 is the probability of restarting the transmission process again after getting a successful transmission or getting a failure at j^{th} attempt. W_0^i is backoff window size for priority $class_i$ at the first attempt. Here, $(1-\beta)$ is the probability of getting the channel free in the second CCA attempt and $(1-q_1)$ is the probability of not going to ideal state. Probability of going to the idle state after the j^{th} retry is given in equation 6.9.

$$P\{i,-1,0/i,j,-1\} = (1-\beta)q_1 \tag{6.9}$$

$$P\{i,0,k/i,R^i,0\} = \alpha(1-q_1)/W_0^i \tag{6.10}$$

Equation 6.10, is the state transition probability of going to the first transmission attempt after getting failure at R^i retry attempt for priority class i. Here $R^i = maxNB_i$ is the maximum retry attempts allowed to class i.

$$P\{i,-1,0/i,R^i,-1\} = q_1 \tag{6.11}$$

$$P\{i,0,k/i,R^i,-1\} = (1-q_1)/W_0^i \tag{6.12}$$

Equation 6.11 is the state transition probability of finding the channel busy at the R^i retry attempt and going to the idle state from here. Equation 6.12 gives the probability of moving to the first backoff stage from finding channel busy at R^i attempt in the second CCA attempt. Equation 6.13 gives the probability of moving to the idle state after reaching the allowed maximum retry attempts.

$$P\{i,-1,0/i,R^i,-1\} = q_1 \tag{6.13}$$

$$P\{i,0,k/i,-1,0\} = (1-q_2)/W_0^i \tag{6.14}$$

Equation 6.14 gives the state transition probability of moving from the idle state to the first backoff stage before accessing the channel. Here q_2 is the state transition probability to keep the node in the idle state as given in equation 6.15.

$$P\{i,-1,0/i,-1,0\} = q_2 \tag{6.15}$$

Let $b_{(i,j,k)}$ be the study state probability for the Markov chain in backoff state k at

the j^{th} attempt in class i.

$$\begin{aligned} b_{(i,0,-1)} &= b_{(i,-1,0)}q_2 + \sum_{j=0}^{R^i-1} b_{(i,j,-1)}(1-\beta)q_1 \\ &\quad + b_{(i,R^i,-1)}q_1 + b_{(i,R^i,0)}\alpha q_1 \end{aligned} \tag{6.16}$$

$$b_{(i,j,-1)} = b_{(i,j,0)}(1-\alpha) \tag{6.17}$$

$$\begin{aligned} b_{(i,0,k)} &= b_{(i,-1,0)}\frac{(1-q_1)}{(W_0^i)} \\ &\quad + \sum_{j=0}^{(R^i-1)} b_{(i,j,-1)}(1-\beta)\frac{(1-q_1)}{(W_0^i)} \\ &\quad + b_{(i,R^i,-1)}\frac{(1-q_1)}{(W_0^i)} + b_{(i,R^i,0)}\alpha\frac{(1-q_1)}{(W_0^i)} \end{aligned} \tag{6.18}$$

$$b_{(i,j,W_j-1)} = b_{(i,j-1,-1)}\frac{\beta}{W_{j-1}^i} + b_{(i,j-1,0)}\frac{\alpha}{W_{j-1}^i} \tag{6.19}$$

$$b_{(i,j,k)} = b_{(i,j-1,-1)}\frac{\beta}{W_{j-1}^i} + b_{(i,j-1,0)}\frac{\alpha}{W_{j-1}^i} + b_{(i,j,k-1)} \tag{6.20}$$

Here, $0 \leq k \leq W_0^i - 2$, and $1 \leq j \leq R^i = maxNB_i$

By the Markov chains property the probability sum over all the states is one. Using that,

$$1 = b_{(i,-1,0)} + \sum_{j=0}^{R^i-1}\sum_{k=0}^{W_j^i-1} b_{(i,j,k)} \tag{6.21}$$

Let $\delta = (1-\alpha)\beta + \alpha, \delta_1 = 1-\delta$ and $\delta_2 = (1-2\delta)$ and putting values of each state in equation 6.18 with $k = 0$,

Next, one has to relate an individual node to the entire network. Let Φ be the probability with which a node goes for channel access. We know that α is the probability of getting the channel busy at the first attempt of channel sensing. In other words, let α be the probability that one of the remaining $N-1$ nodes senses the channel successfully and starts the transmission independent of current node. Here L be the payload size,

$$\alpha = L[1-(1-\Phi)^{N-1}](1-\alpha)(1-\beta) \tag{6.22}$$

$$\Phi = 1 - \left[1 - \frac{\alpha}{L(1-\alpha)(1-\beta)}\right]^{\frac{1}{N}} \tag{6.23}$$

Another equation for Φ can be derived using β. β is the probability that a node finds the channel busy in the second CCA. This probability is not the same as α. Here, α is independent of previous knowledge whereas β is not independent. Probability of getting channel busy in the second CCA can not be considered an independent event, as it depends on the collective knowledge the nodes have acquired in the first CCA. In unacknowledged case the device will get the channel busy only if another node in the network senses for the second time during the device's first sense and starts a new transmission in the second slot.

$$\beta = \frac{1-(1-\Phi)^N}{2-(1-\Phi)^N} \tag{6.24}$$

$$\Phi = 1 - \left[\frac{1-2\beta}{1-\beta}\right]^{\frac{1}{N-1}} \tag{6.25}$$

Another way to define Φ is,

$$\Phi = \sum_{j-0}^{R^i-1} b_{(i,j,0)} = \frac{1-\delta^{R^i}}{1-\delta} b_{(i,0,0)} \tag{6.26}$$

Finally the probability that a node will go to transmission stage is,

$$\begin{aligned} \tau_i &= \Phi(1-\alpha)(1-\beta) \\ &= \frac{1-\delta^{R^i}}{1-\delta}(1-\delta)(1-\beta)b_{(i,0,0)} \end{aligned} \tag{6.27}$$

By assuming that $q_1 = q_2$ with equation 6.23, 6.25 and 6.26 the system can be solved to get transmission probability τ_i. Let P_b be the probability that channel is busy. It happens only when at least one station transmits during a slot time.

$$P_b = 1 - \prod_{i=0}^{N-1}(1-\tau_i)^{n_i} \tag{6.28}$$

6.3.2 Throughput equation

This section derives throughput equation for priority $class_i$. Throughput calculation for priority $class_i$ requires equations for, successful transmission probability in a slot time, probability of getting the channel busy and ideal channel probability. Let, P_s^i $(0 \leq i \leq N)$ be the probability for successful transmission in a slot time for the priority $class_i$ and let P_s be the probability that successful transmission occurs in a slot time.

$$P_s^i = n_i\tau_i(1-\tau_i)^{n_i-1} \prod_{j=0, i\neq j}^{N-1}(1-\tau_j)^{n_j} P\{i,j,k/i,j,k+1\} \tag{6.29}$$

$$P_s = \sum_{i=0}^{N-1} P_s^i = (1-P_b)\sum_{i=0}^{N-1}\frac{n_i\tau_i}{1-\tau_i} \tag{6.30}$$

The probability that channel is ideal for a slot time is $(1-P_b)$ and channel collision probability is $(1-P_b-P_s)$, *i.e* channel is neither busy nor successful. Using Binachi [50], throughput for class i,

$$S_i = \frac{E[Payload\ transmission\ time\ in\ slot\ time]}{E[length\ of\ slot\ time]} \tag{6.31}$$

$$S_i = \frac{P_s^i T_{E(L)}}{(1-P_b)\theta + P_s T_s + (P_b - P_s)T_c} \tag{6.32}$$

Here θ, T_s, T_c and $T_{E(L)}$ are the empty slot time, busy channel time successful transmission, average collision time for channel and time to transmit pay load respectively.

$$T_s = T_H + T_{E(L)} + SIFS + T_{ACK} + DIFS \tag{6.33}$$

$$T_c = T_H + T_{E(L^*)} + SIFS + T_{ACK} + DIFS \tag{6.34}$$

6.3.3 Packet dropping probability

Let $Pkt^i_{drop}(0 \leq i \leq N)$ denotes the packet dropping probability for the $class_i$. Packets get dropped when they reach their maximum retry limit R^i. So probability of packet drop is calculated as,

$$Pkt^i_{drop} = P_b^{R^i+1} \tag{6.35}$$

Let $Pkt^i_{success}(0 \leq i \leq N)$ denotes the packet success probability for the $class_i$.

$$\begin{aligned} Pkt^i_{success} &= \sum_{j=0}^{R^i} P_b^j (1-P_b) \\ &= 1 - P_b^{R^i+1} \end{aligned} \tag{6.36}$$

6.3.4 Delay for priority class i

Delay for priority $class_i$, is the average delay that includes medium access delay, transmission delay and the inter-frame space. The average delay depends on stations backoff attempt and number of backoff slots experienced. Let $X_i(0 \leq i \leq N)$ denote the random

variable representing the total number of backoff slots station encounters. The probability that a frame is successfully transmitted after jth retry is given by $\frac{P_b^j(1-P_b)}{1-P_b^{(R^i+1)}}$. The average number of backoff slots that a station requires to transmit a frame successfully at the j^{th} retry is $\sum_{k=0}^{j}[(W_k^i-1)/2]$.

$$E(X_i) = \sum_{j=0}^{R^i} \frac{P_b^j(1-P_b)}{(1-P^{R^i+1})} \sum_{k=0}^{j} \frac{W_k^i - 1}{2} \tag{6.37}$$

Let $E(N_i^{retry})$ denote the average number of retries for the priority $class_i$.

$$E(N_i^{retry}) = \sum_{j=0}^{R^i} \frac{jP_b^j(1-P_b)}{1-P^{R^i+1}} \tag{6.38}$$

Let, $(0 \leq D_i \leq N)$ denote the random variable representing the frame delay for the priority $class_i$.

$$E(D_i) = E(X_i)\theta + E(N_i^{retry})(T_c + T_o) + T_s \tag{6.39}$$

$$T_o = SIFS + T_{ACK_{timeout}} \tag{6.40}$$

T_s, T_c and T_o are success time, collision time and wait time defined in equation 6.33, 6.34 and 6.40 respectively.

6.3.5 Energy equation

$$E_{avg} = \frac{E[energy\ per\ time\ slot]}{E[successful\ data\ tranmission\ per\ time\ slot]} \tag{6.41}$$

$$E_{avg} = \frac{(1-P_b)E_I + P_sE_T + (P_b - P_s)E_c}{P_s^i E_{E(L)}} \tag{6.42}$$

Here E_I, E_T , E_c and $E_{E(L)}$ are the energy values involved in idle listening, transmission energy , energy involved in collision and energy involved in data transmission, respectively. These energy values are calculated based on the time involved in each state.

6.4 Explicit Prioritized Channel Access Protocol (EPCAP)

Explicit priority scheme is desinged to work with beacon enabled mode of IEEE 802.15.4. In beacon enabled mode coordinator nodes periodically send the beacon in the network. Nodes belonging to that coordinator node synchronize with the beacon to locate the slot boundary. Nodes having packets for the coordinator contend in CAP part of the superframe using CSMA-CA algorithm. Nodes which get success in getting clear channel will transmit their packets to the coordinator.

By default all the nodes attached with a coordinator and having packets for that coordinator will go for clear channel assessment after random backoff in CAP part. Nodes having critical information will also contend with nodes having normal information for the coordinator. That delays critical information and causes problems.

First task of the problem solving is to distinguish the critical information nodes from the routine or normal information nodes. After differentiating critical information nodes from the normal information nodes, the coordinator allows only critical information nodes to contend for the channel and restricts the normal information nodes from contending for the channel.

The first part, that is, of distinguishing critical nodes from normal nodes, requires informing the coordinator that some of the nodes have critical information to send to the base station. The second part requires that the coordinator should inform the normal nodes not to contend for the channel, as the network has critical nodes, which want to convey their information to the coordinator. Here onwards we will consider critical information nodes as higher priority nodes and normal information nodes as low priority nodes. We restrict ourselves to two priority levels, but our scheme can be extended to more than two priority levels.

In the literature it is commonly assumed that network nodes are within each others' communication range. But in normal network scenarios due to limited communication range, it is not valid to assume that all the nodes are in each others communication range.

In our scheme we do not assume that all the nodes are within each others communication range and are able to hear messages transmitted by other nodes.

To solve the first part of the problem, that is of informing the coordinator node about the priority packet with minimum delay, we have used the secondary beacon mechanism. Nodes with the high priority information send a secondary beacon to the coordinator node. On getting the secondary beacon, the coordinator node comes to know that some node in the network has high priority information pending for the coordinator.

The second part of the problem is restricting low priority nodes in the network from contending for the channel with the high priority nodes. As all nodes are not in each other's communication range, low priority nodes may not be able to hear the secondary beacon. The node which is sure to be in the communication range of the low priority nodes as well as the high priority nodes is the coordinator node. In our scheme it is the responsibility of coordinator to make the low priority nodes aware that the network has high priority nodes pending with a message.

The coordinator node informs about the need for transmission by a higher priority nodes. This it achieves by sending a primary beacon which is same as the default beacon with the additional information about the priority. On getting this information from the coordinator, low priority devices do not contend for the channel in the current beacon interval. Detailed algorithms are given in following paragraphs for the coordinator node, higher priority and low priority nodes.

Algorithm 8.1 gives the steps for the device node registered with corresponding coordinator node. At the time of initialization the device node configures a set of timers one for each priority level. The node configures these timers just before the next primary beacon and in the order of their priority level. The first timer to expire is the highest priority level timer, next is second highest and so on and last timer to expire is the primary beacon timer. Time duration between consecutive timers depends on the time required for a node to send secondary beacon and the time required at the coordinator node.

To get synchronized with the coordinator, the secondary beacons are transmitted without using CSMA-CA algorithm. In this case it may happen that more than one

Algorithm 6.1: Algorithm at Device Nodes

```
1: Configure Priority timers
2: Start priority timers
3: OnTimer(Priority i)
4: if node is registered with coordinator then
5:    Check network layer for data packet
6:    if Pending packet of priority i then
7:       Send secondary beacon
8:    end if
9: end if
10:
11: OnBeacon(Primary)
12: Get the priority value
13: if any packet for same priority then
14:    Contend for channel
15: else
16:    Go to sleep state
17: end if
```

node sends the secondary beacon at a particular priority level and this leads to packet collision. As the coordinator is not interested in the content of the secondary beacon and just measures the energy level at the time of the corresponding priority level, secondary beacon collisions do not have serious consequences. A higher energy level indicates to the coordinator that some of the nodes have packets for that particular priority level. Pre-configuration step helps in finding the number of beacons colliding at a particular level.

The most important point that previous works have not highlighted is the effect of external noise and other node's constructive and destructive interference. To detect that, a pre-configuration step is required. Based on that, the coordinator maps the detected energy level with the number of contending nodes. This result is not accurate but it will provide number of contending nodes with certain probability for the next BI. Based on this information, the coordinator node configures its active and passive time for next BI.

This helps in saving energy at all the nodes. If the number of nodes is small in a given priority level then the coordinator node may decrease BI's active time and save the nodes energy. Second, if some other priority level has waiting nodes then the coordinator configures a smaller BI interval and soon sends the next beacon. But since changing

BI requires more adjustments at coordinator node and device nodes, here we have not changed the BI. In the present work we have used the detected energy level to identify the priority level and the effect of it on BI is not included.

Algorithm 6.2: Algorithm at Coordinator Nodes

```
1: Configure Priority timers
2: Start priority timers
3: OnTimer(Priority i)
4:     Go for energy detection
5:     Store the received energy and
6:     SNR for the given priority level
7: OnTimer(Primary Beacon)
8:     Get the first slot with signal energy and SNR
9:     Assign above slot level to priority level of primary beacon
10:    Set primary beacon priority to priority level
11:    Send primary beacon
```

Algorithm 6.2 is for the coordinator node. In beacon enabled mode, the coordinator node is responsible for receiving secondary beacons from registered nodes and sending of primary beacon to registered nodes. At the time of device initialization, the coordinator configures the priority timers, one for each priority level and starts them. These timers are configured according to time required to detect energy at the physical layer by calling the energy detection utility provided by IEEE 802.15.4 physical layer.

The coordinator node goes for the energy detection for each configured priority level. The coordinator node detects the energy for a configured period of time and records the reading (signal to noise ratio and energy) provided by the physical layer. Based on the detected energy at each level, the coordinator node decides the priority level for the primary beacon and superframe CAP period. It sets that priority level in the primary beacon and broadcasts it at the time of primary beacon expiry.

Both the algorithms together implement the explicit priority mechanism in beacon enabled mode of IEEE 802.15.4. The following example explains the working of explicit priority scheme in beacon enabled IEEE 802.15.4.

To explain our scheme more clearly we have crafted an example with a coordinator node and 3 device nodes. Figure 6.3 shows our example. Here, the horizontal line

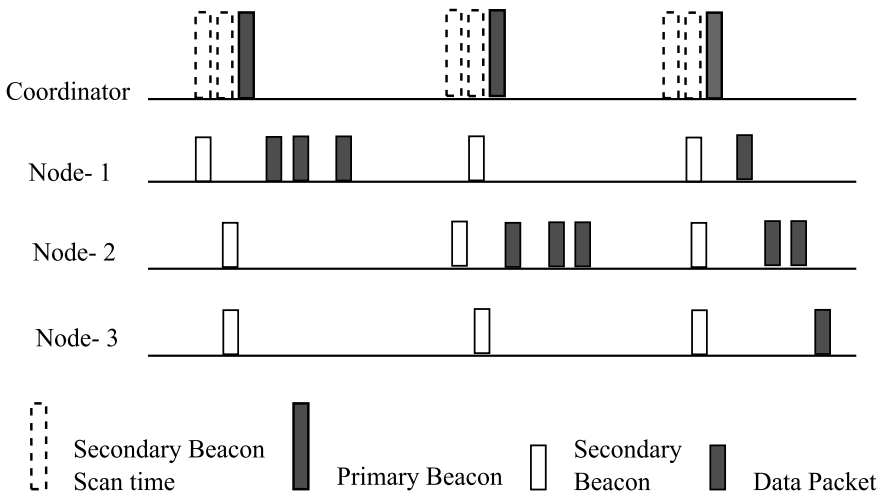


Figure 6.3: Example showing working of Explicit Priority Scheme

represents the time line for the network. The example talks about the network with two priority levels, high priority level and low priority level. The big solid rectangle shown in the figure represents the primary beacon packet, the small rectangle represents the secondary beacon packet and the small solid rectangle represents the data packets.

On the coordinator line the dotted empty rectangles are the time slots during which the coordinator node goes for the energy detection at physical layer. The coordinator node goes for the energy detection before sending the primary beacon. On the coordinator line, the first empty dotted rectangle is for the high priority secondary beacon energy scan. The second is for the low priority secondary beacon energy scan. The coordinator periodically follows this pattern of energy detection for priority levels, followed by sending of primary beacon.

In figure 6.3 node one has high priority packets for coordinator. At the timeout of high priority timer, node one having pending higher priority packet goes for the secondary beacon transmission to the coordinator. The coordinator detects this in the higher priority energy scan. On the second scan which is for low priority packets, node 2 and node 3 send their secondary beacon to the coordinator node. Based on detected energy at low level slot, coordinator comes to know that some nodes have low level packets pending for the coordinator. Based on energy reading of both the high level priority slot and low

level priority slot, the coordinator can infer that network has high level priority packets pending in the network.

The coordinator sets the priority level to high priority and sends the primary beacon to the network nodes. On getting this primary beacon, registered nodes 2 and 3 come to know that the current beacon CAP is for the high priority packets and only high priority nodes will contend for the channel. In our case only node 1 will contend for the channel. Here as shown in figure 6.3, only node 1 is sending data packets which is marked with dotted small rectangle.

In next beacon interval node 2 has higher priority packet so node 2 is sending its packet in the CAP period of the current superframe. In the last case, no node in the network has high priority packets for the coordinator. Here coordinator node detects very low energy, only noise in energy scan performed at high priority slot. Further, there are nodes having low priority packets which send the secondary beacon at the low priority energy scan. Getting both the energy levels, the coordinator sets the priority to low priority in the primary beacon. As shown in figure 6.3 all the three nodes contend for the channel and gradually node 1, node 3 and node 2 get the channel and forward their packets to the coordinator. It may happen that more than one node will contend with high priority packets. In such cases the number of contending nodes are far less than the normal scenario in which all the nodes contend for the channel.

6.4.1 M/G/c queuing model for EPCAP

$M/G/c$ is multi-server queueing system model. Explicit priority scheme is modeled as M/G/c queueing system with finite number of distinct classes $C = 1, ..., I$. To derive waiting time equation we have adopted the same methodology used in [65].

In EPCAP, each priority class has some nodes which contend for the channel. Assuming each priority class has same number of nodes c, we can consider these c nodes as c servers. Each server processes packet requests and goes for channel access. At the end the server will transmit the data with success probability defined in section 6.3. This success probability is a special case of IPCAP with all the parameters set to their default

values. Figure 6.4 shows the M/G/c modeling of EPCAP.

Here, the packet arrives at the system according to independent Poisson processes; λ_i denotes the arrival rate of class i, $i \in C$. The service time of a given class $i \in C$ is assumed to be independent and identically distributed random variable V_i with finite second moment.

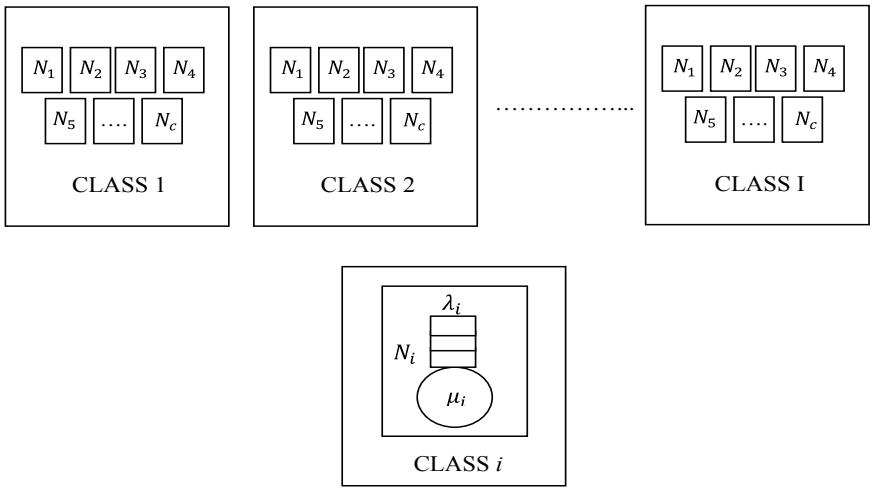


Figure 6.4: Explicit Priority Scheme M/G/c model

Following are the strong work conserving rules that EPCAP follows: a) no FFD in the current priority class is free when a packet is in the queue, b) the discipline does not affect the amount of service time given to the packet or the arrival time of any packet, c) Priories are assigned on the basis of the history of the FFD and the time elapsed since the last epoch at which the system became empty.

Assumption A**:** if $c > 1$, assume all customers have the same service time distribution.

Let $\rho_i = \lambda_i E[V_i]$, $i \in C$ and W_{ni} = delay of the nth customer of class i ($i \in C$; $n \geq 1$).

Considering following work conservation law, $\sum_{i=1}^{I} = \rho_i/c < 1$ and condition (A). With fixed priority updates , there are numbers W_j^* , $j \in E$ such that,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} W_{ni} = W_i^* \qquad (w.p.1), \qquad i = 1, ..., I. \tag{6.43}$$

More specifically, in a delay dependent priority rule, positive weights α_i $(i \in C)$ are specified such that a packet in class i, who arrives at time τ is given a dynamic priority value of $\alpha_i(t-\tau)$ at time $t > \tau$. Further dynamic priority can be extended to mixed dynamic priority.

A mixed dynamic priority rule is characterized by a partition $C_1, ..., C_J$ of C. The subsets $C_l : l = 1, ..., J$ are referred to as leagues. A customer in C_k has absolute priority over a customer in C_l if and only if $k > l$. Relative priorities within a league l (l=1,...,L) are determined according to a dynamic priority rule with weight vector $\alpha^{(l)} = (\alpha_i), i \in C_l$. Customer classes are numbered in ascending order of their attributed priority. Here we are assuming that $I = L$. That says each league has only one class in it.

As in Kleinrock [66] consider a packet in class $p \in C$. Define, N_{ip} = number of packet from class i who are in the queue when the tagged customer (in class p) arrives and who receive service before it does $(i \in C)$.

M_{ip} = number of customers from class i who arrive at the system while the tagged customer (in class p) is in queue and who receive service before it does $(i \in C)$.

Assume condition (A) holds. Let V_0 be, the initial delay an arbitrary packet experiences (if any) until the first epoch at which a server becomes available for service. The distribution of V_0 is independent of priority update rules, and $E[V_0] < \infty$.

$$E[V_0] = \frac{1}{2} \sum_{i=1}^{I} \lambda_i E[V_i^2] \tag{6.44}$$

Since Poisson arrivals see time averages,

$$W_p = E[V_0] + \frac{1}{c} \sum_{i=1}^{J} E[N_{ip} + M_{ip}]E[V_i], \quad p \in E. \tag{6.45}$$

let $S_l = \bigcup_{k=l}^{I} C_k$, $l = 1, ..., I$. Following the analysis in Kleinrock [65] we obtain:

$$E[N_{ip}] = 0 \quad for \quad i \in A' = E \setminus S_l, \tag{6.46}$$

$$E[N_{ip}] = \lambda_i W_j \alpha_i / \alpha_p \quad for \quad i \in C_l, i < p, \tag{6.47}$$

$$E[N_{ip}] = \lambda_i W_i \quad for \quad i \in S_l, i \geq p, \tag{6.48}$$

$$E[M_{ip}] = 0 \quad for \quad i \leq p \tag{6.49}$$

$$E[M_{ip}] = \lambda_i W_p (1 - \frac{\alpha_p}{\alpha_i}) \quad for \quad i \in C_l, i < p, \tag{6.50}$$

$$E[M_{ip}] = \lambda_i W_p \quad for \quad i \in S_{l+1}. \tag{6.51}$$

for $p \in C_l \ \ (l = 1, ..., I)$, using the above equation we have,

$$\begin{aligned} W_p = E(V_0) \ &+ \ \frac{1}{c} \sum_{i \in E_l, i<p} \rho_i W_i \frac{\alpha_i}{\alpha_p} \\ &+ \ \frac{1}{c} \sum_{i \in S_l, i \geq p} \rho_i W_i \\ &+ \ \frac{1}{c} \sum_{i \in E_{lp}, i<p} \rho_i W_p (1 - \frac{\alpha_p}{\alpha_i}) \\ &+ \ \frac{1}{c} \sum_{i \in S_{l+1}} \rho_i W_p \end{aligned} \tag{6.52}$$

Values of $W_p, p \in C_l \ \ (l = 1, ..., I)$, can be obtained by solving liner system of equations 6.53.

Assumption (B): Assuming that arrival rates and service rates are the same for all the classes, $\lambda_i = \lambda$, $E[V_i] = E[V] = \mu$, and $\rho_i = \rho = \lambda E[V]$, $i \in C$.

$$
\begin{aligned}
W_p = E(V_0) \quad &+ \quad \frac{1}{c} \sum_{i \in C_l, i<p} \rho W_i \frac{\alpha_i}{\alpha_p} \\
&+ \quad \frac{1}{c} \sum_{i \in S_l, i \geq p} \rho W_i \\
&+ \quad \frac{1}{c} \sum_{i \in E_l, i<p} \rho W_p (1 - \frac{\alpha_p}{\alpha_i}) \\
&+ \quad \frac{1}{c} \sum_{i \in S_{l+1}} \rho W_p
\end{aligned} \tag{6.53}
$$

Solving the above system of linear equations gives the average waiting for EPCAP model. The above analysis is based on the synthesis algorithm presented in [65].

6.5 Summary

To handle critical events in event driven wireless sensor network, we have proposed two prioritized channel access protocols IPCAP and EPCAP. Both the protocols are designed to work with networks built upon IEEE 802.15.4 communication standard. To handle critical events IPCAP modifies CSMA-CA algorithm and EPCAP uses the concept of secondary beacon. This chapter covered working and modeling of IPCAP and EPCAP. IPCAP is modeled as two dimensional Markov chain and EPCAP is modeled as M/G/c queuing model.

Chapter 7

Implementation of Prioritized MAC

7.1 Introduction

Chapter 6 covers the details of proposed IPCAP and EPCAP protocols. Both the protocols configure the MAC layer parameters based on information they receive from the application layer. To verify the working of IPCAP and EPCAP, we have implemented both the protocols in Qualnet Network simulator. This chapter covers the implementation details of our protocols. It shows the conceptual representation of both the protocols in CLAPDAWN architecture. Further, major results that we have found in the course of simulation of both the protocols are shown in this chapter.

7.2 Implementation

Figure 7.1 shows framework for our implementation. Application layer provides priority information to control interface. MAC layer provides the channel and beacon details to control interface. On arrival of events application layer generates the message and puts priority information in control interface. This will trigger the event in control interface and it propagates them to execution engine. On getting events, execution engine fetches rules from rule base and executes them. Here rule base contains rules for implicit priority mechanism and explicit priority mechanism.

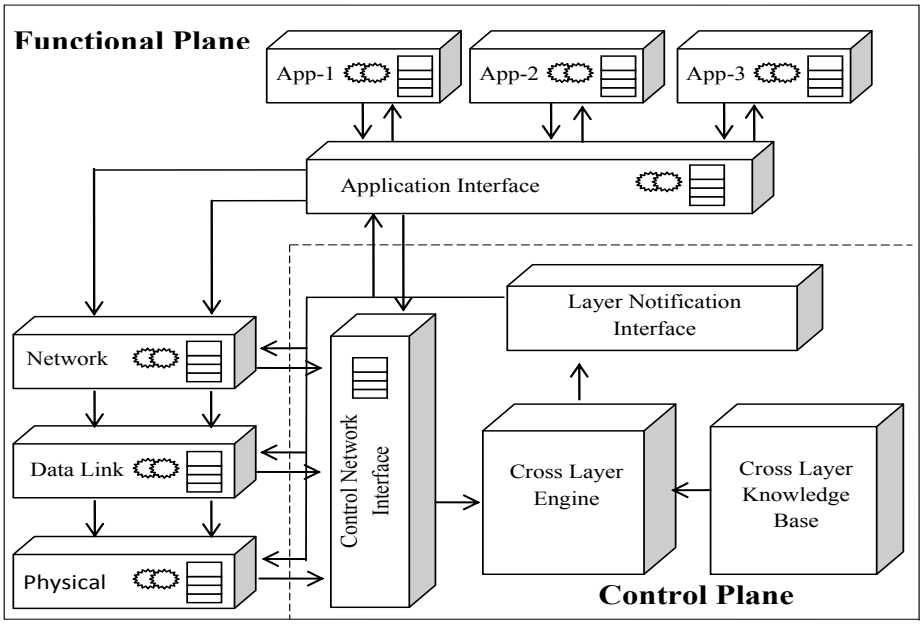


Figure 7.1: CLAPDAWN Framework

7.2.1 IPCAP implementation

Implicit priority rules contain the information about the parameters of CSMA-CA algorithm. Using this information execution engine generates the messages to set the parameter at MAC layer. It puts these messages into notification layer. Notification layer sends these messages to MAC layer, which at the end updates its parameters.

7.2.2 EPCAP implementation

Explicit priority rules contain the information about the secondary beacon. Execution engine gets this information and generates the messages to send secondary beacon. It puts these messages into notification layer. Notification layer sends these messages to MAC layer, which at the end sends the secondary beacon.

At the coordinator node priority rule generates the messages for energy scan. Execution engine puts these messages on notification layer, that triggers the MAC layer to go for energy scan for each priority slot. After energy scan the detected energy readings are updated on CNI, based on which the execution engine decides the priority level for the

next BI.

In IPCAP and EPCAP, MAC layer is itself not deciding about CSMA-CA parameter or secondary beacon. In IPCAP it configures the CSMA-CA parameter and in EPCAP it transmits the secondary beacon based on the notification messages received from notification layer. The knowledge base and the inference engine control the overall working. That makes the conventional layer mechanism untouched from the cross layer implementations.

7.3 Simulation Results

7.3.1 IPCAP

QualNet is a network simulator from scalable network technology. We have implemented IPCAP protocol in QualNet and compared it with the standard IEEE 802.15.4 MAC protocol. Both the protocols are evaluated considering different network conditions. In our simulation we have changed network traffic and one hop neighbor density. With these network conditions we have compared IPCAP and standard protocol for packet delivery, prioritized packet delivery, throughput and energy consumption. In all the cases we have considered network as star topology with percentage of nodes rounded to the nearest integer value. Following subsections highlight our major findings.

Packet delivery ratio

The proposed prioritized protocol works with application specific prioritized classes. A network node defines its class based on application data. Nodes belonging to high priority class should get higher chances of getting channel than nodes with lower priority classes. In our simulation we have considered five priority classes. Each priority class comes with the set of CSMA-CA parameters defined in chapter 6. Node selects its priority class based on application layer specification.

To analyze the packet delivery ratio of different priority classes, we have tested our network by varying the number of one hop neighboring nodes and network traffic. Each priority class generates the same amount of packets in the network. Figure 7.2 shows

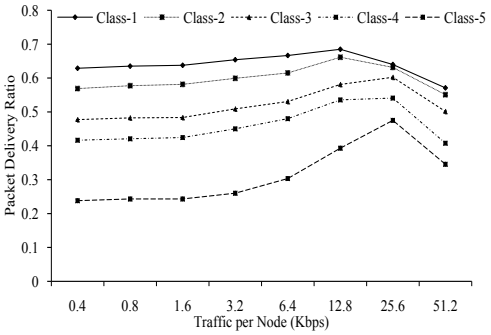
simulation results for one hop contending nodes ranging from 5 nodes to 25 nodes. In the result, X-axis represents the network traffic and Y-axis represents the packet delivery ratio. In our simulation we have increased network traffic from 0.4 Kbps to 51.2 Kbps. Here each packet consists of 50 bytes.

Figure 7.2(a) shows the simulation result for the network having 5 nodes as one hop neighbor nodes. Simulation result shows that set 1 which represents the priority class 1 has higher packet delivery ratio than the other priority classes. Priority class 1 is getting more chances of sending packets. So it has higher packet delivery than the other priority classes. Next we have priority class 2, which has packet delivery less than the priority class 1 and greater than the priority class 3, 4 and 5. As priority class 2 gets more chances of packet access than the priority class 3,4 and 5 it has higher packet delivery than the priority class 3, 4 and 5. But, it has lower chances of channel access than the priority class 1 so its packet delivery count is less than the priority class 1. Priority class 3, 4 and 5 follow the same pattern.

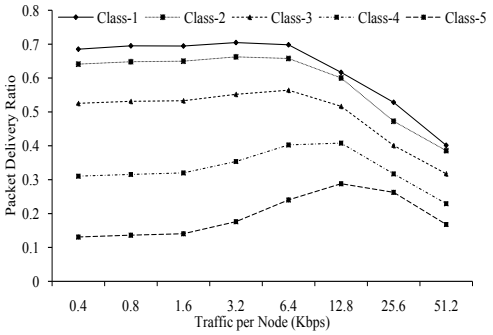
Figure 7.2(a) shows that as the network traffic increases the number of packets delivered increases for each priority class. As the traffic increases, soon the network reaches its limit (transition point) and gets congested. After that the network nodes start dropping packets. This decreases the packet delivery in the network. This particular pattern can be seen in all the simulation results from figure 7.2(a) to 7.2(e). As number of nodes increases this transition point shifts from high traffic to low traffic.

Prioritized packet delivery

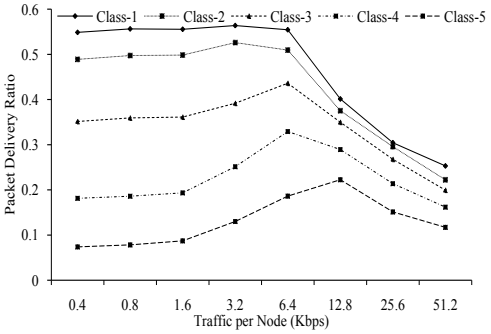
Figure 7.3 and 7.4 show the next set of simulation results. It covers the result for prioritized packet delivery in given priority class. The previous set of results shows how nodes falling in same priority class compete for the channel. There, the high priority class has more success in channel access compared to low priority class and all the priority classes have same number of nodes. Here, we analyze the packet delivery ratio of nodes for network having two priority classes. Lower priority nodes are considered as normal nodes and higher priority nodes are considered as prioritized nodes. In simulation we



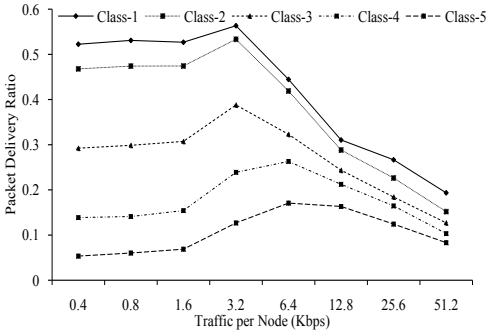
(a) 5 One Hop Neighbor Nodes



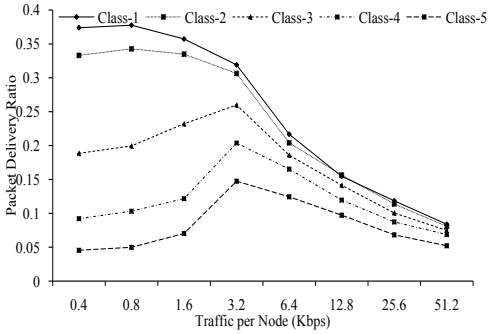
(b) 10 One Hop Neighbor Nodes



(c) 15 One Hop Neighbor Nodes



(d) 20 One Hop Neighbor Nodes



(e) 25 One Hop Neighbor Nodes

Figure 7.2: Packet Delivery Ratio for Different Priority Classes

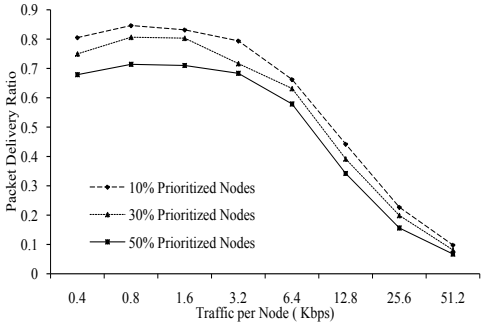
have analyzed the effect of prioritized nodes on the normal nodes. We also analyzed the effect of increasing the number of nodes in the given priority class.

To analyze the prioritized packet delivery of different priority classes, we have tested our network with varying one hop neighboring nodes and network traffic. Each node in given priority class delivers the same number of packets in the network. We have tested our network for 10 and 15 one hop neighbor nodes. With increasing traffic we have measured the packet delivery ratio on Y-axis.

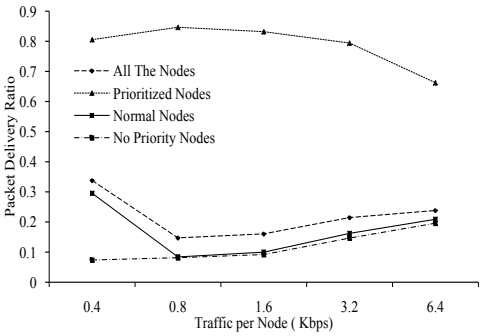
Figure 7.3 and 7.4 show the simulation results for the prioritized packet delivery for 10 and 15 one hop neighbor nodes respectively. We have increased the number of nodes in higher priority class from 10% to 50% and measured the effect of it on packet delivery ratio. Simulation result 7.3(a) shows that as the number of nodes increases in the given priority class, the packets delivered in that priority class decreases. Result shows that from given set of nodes when 10% nodes are falling in priority class, it has higher packet delivery than the case in which network has 30% priority nodes. As the number of nodes increases in given priority class the more nodes will contend for priority slot and that decreases the packet delivery of the given priority class. But these priority class nodes get more chances of packet delivery than the normal nodes. This increases overall packet delivery ratio.

For better clarity we have compared the prioritized delivery results with normal nodes. Figure 7.3(b), 7.3(c) and 7.3(d) cover the simulation results with different number of prioritized and normal nodes in 10 one hop contending nodes. As seen in the result as the number of nodes in given priority class increases, the number of packets delivered in that priority class decreases but it still remains more than the normal nodes. Further the difference between the packets delivered in prioritized class and all network nodes decreases. And this pattern repeats for 30% and 50% prioritized nodes.

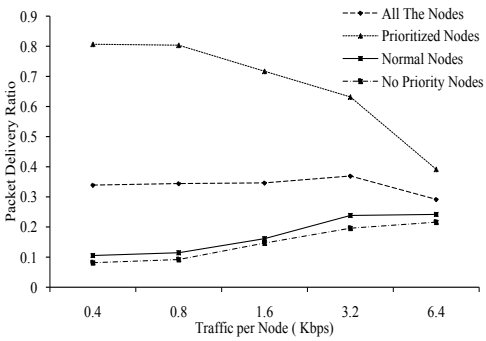
The result 7.3(b) shows the packet delivery of 10% prioritized nodes with the normal node. With 10% priority nodes, the difference between higher priority nodes and lower priority nodes is less. Here majority of the nodes work as normal nodes and the effect of it is visible in the results. Packet delivery ratio of network having no priority nodes



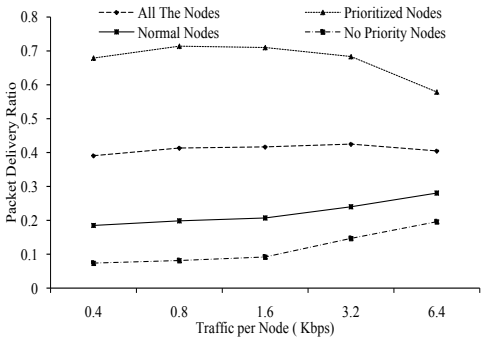
(a) Prioritized Nodes



(b) 10% Prioritized Nodes in Network

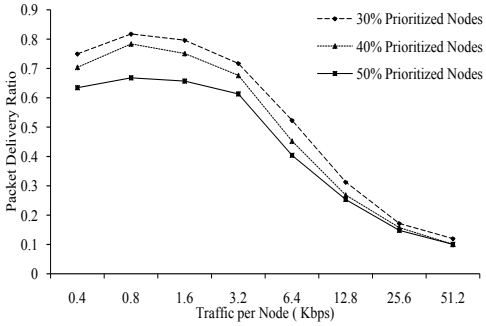


(c) 30% Prioritized Nodes in Network

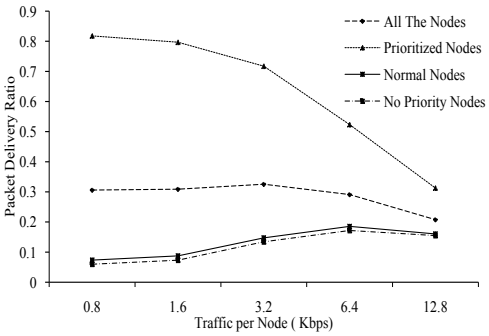


(d) 50% Prioritized Nodes in Network

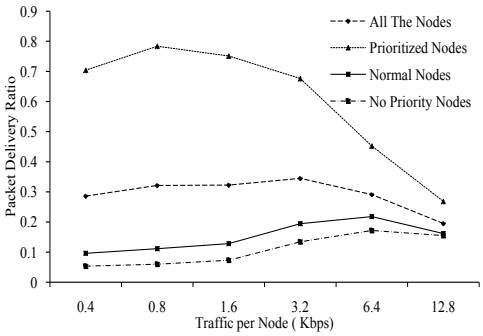
Figure 7.3: Prioritized Packet Delivery Ratio for 10 one Hop Nodes



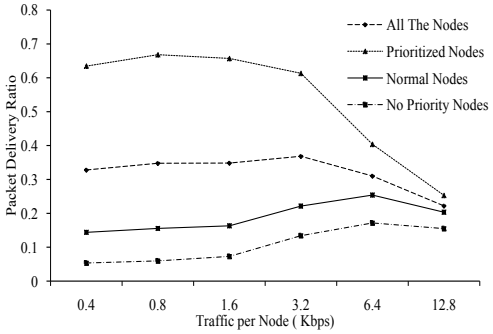
(a) Prioritized Nodes



(b) 10% Prioritized Nodes in Network



(c) 30% Prioritized Nodes in Network



(d) 50% Prioritized Nodes in Network

Figure 7.4: Prioritized Packet Delivery Ratio for 15 one Hop Nodes

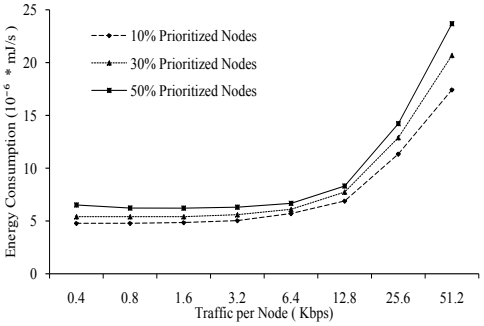
and packet delivery ratio of normal nodes for the network having 10% priority nodes are close to each other. Further less number of nodes are in higher class so it has higher packet delivery ratio. But as the number of priority nodes increases in the network, this difference increases as shown in figure 7.3(c) to 7.3(d) and 7.4(b) to 7.4(d).

Energy in priority class

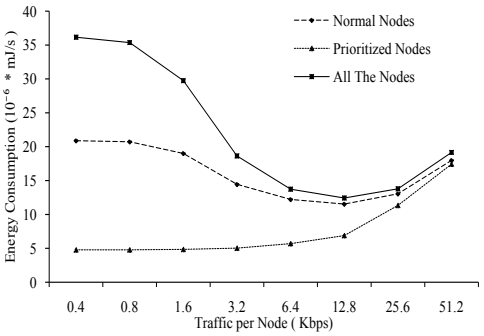
Next set of results given in 7.5 shows the per bit energy consumption in given priority class. In this set of results we have compared per bit energy consumption by a node in a given priority class. We have tested our network with 10 and 15 one hop neighbor nodes. In results, X-axis represents the network traffic and Y-axis represents per bit energy consumption in milli Joules. In our simulation, we have increased network traffic from 400 bps to 51.2 Kbps. With this increasing traffic we have measured the per bit energy consumption on Y-axis.

Figure 7.5 and 7.6 show the simulation results for the per bit energy consumption in given priority class for one hop neighbor density of 10 nodes. We have increased the number of nodes in given priority class from 10% to 50% and measured the effect of it on per bit energy consumption. Simulation result 7.5(a) shows that as the number of nodes increases in the given priority class the packet delivered in that priority class decreases. That increases the wait time and number of packet transmission attempts for the nodes, which leads to higher energy consumption for the nodes. Result shows that from given set of nodes when 10% of the nodes are higher priority nodes, network has higher packet delivery and with 30% nodes it has less packet delivery ratio. Because of it 10% nodes have lesser per bit energy consumption than the case in which network has 30% priority nodes.

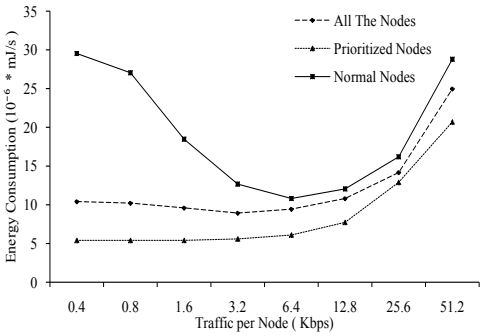
Further figures 7.5(b) 7.5(c) and 7.5(d) show the simulation results comparing energy consumption between prioritized nodes and normal nodes. Result shows that the prioritized nodes consume less energy compared to normal nodes. It also shows that normal nodes or low priority nodes have higher energy consumption than the overall energy consumption. This difference decreases as the number of priority nodes increases in



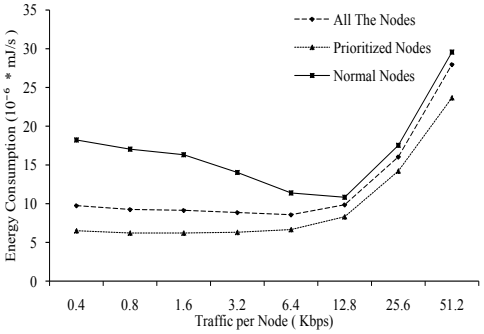
(a) Prioritized Nodes



(b) 10% Prioritized Nodes in Network

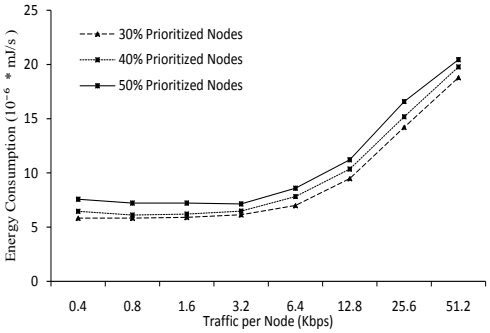


(c) 30% Prioritized Nodes in Network

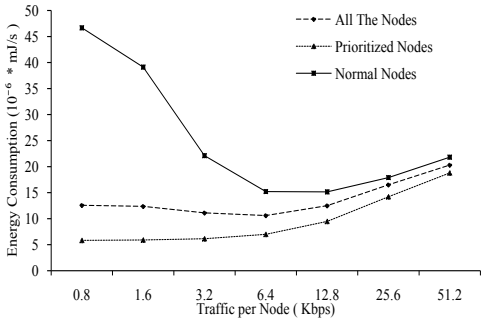


(d) 50% Prioritized Nodes in Network

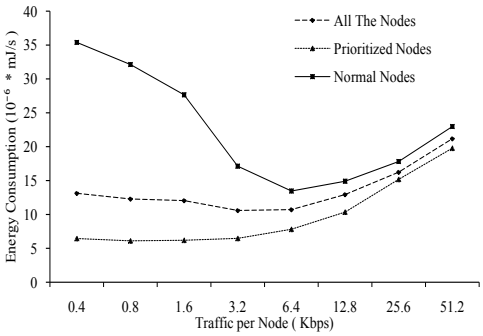
Figure 7.5: Per bit Energy Consumption by Nodes with 10 Nodes



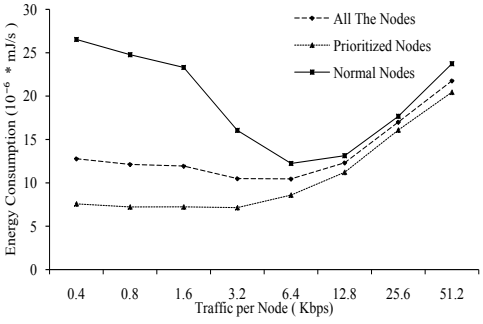
(a) Prioritized Nodes



(b) 30% Prioritized Nodes in Network



(c) 40% Prioritized Nodes in Network



(d) 50% Prioritized Nodes in Network

Figure 7.6: Per bit Energy Consumption by Nodes with 15 Nodes

the network because of low energy consumption of the higher priority nodes, the overall average energy consumption decreases.

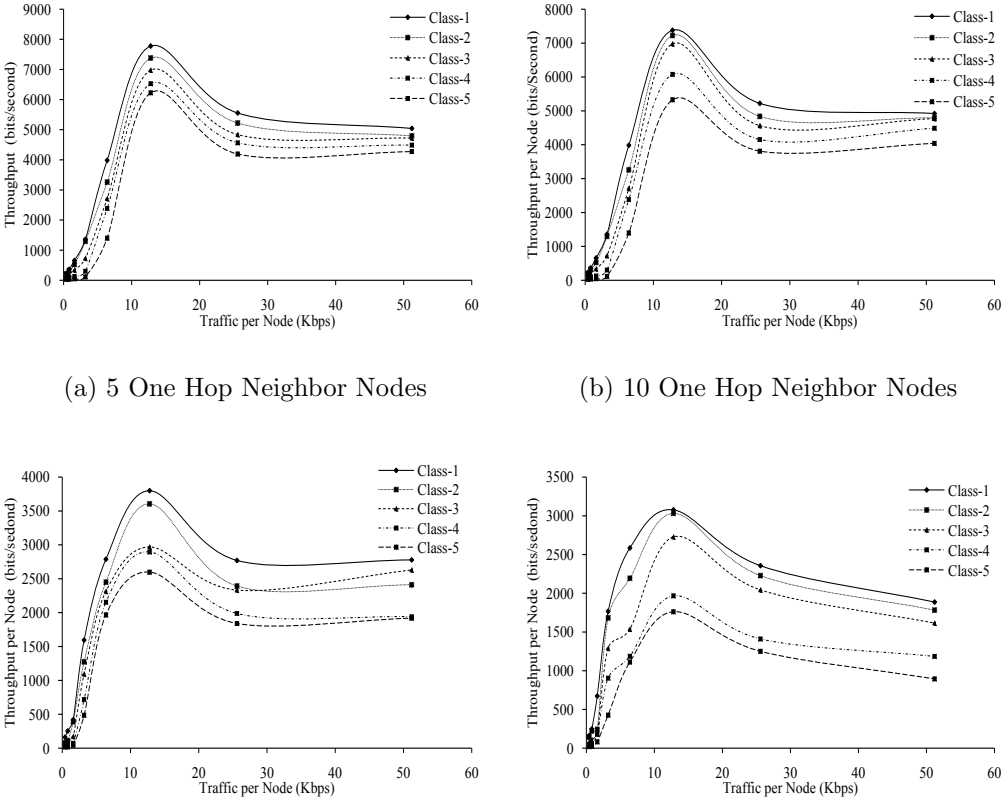
Result shows that with the increase in traffic, the overall energy consumption also increases. Overall energy consumption and prioritized energy consumption follow the same trend. As the traffic increases the energy consumption increases. As we will see in the throughput result that network throughput increases as the traffic increases. For 10 one hop nodes, throughput increases upto 12.8Kbps and then it starts decreasing. This effect is also visible in energy results. Here, network has capacity upto 12.8Kbps, and it uses it to transmit prioritized data. That increases the energy consumption of prioritized nodes and decreases the energy consumption of normal nodes. After the 12.8Kbps transition point the network has traffic more than its capacity and that increases the energy consumption for all the nodes.

Throughput

Next set of simulation results are of throughput. Figure 7.7 shows the througput results for the priority class 1 to 5. To analyze the throughput of packet delivery of a given priority class, we have tested our network with varying one hop neighboring nodes and network traffic. Each node in given priority class delivers the same number of packets in the network. We have tested our network with one hop neighbors ranging from 5 nodes to 25 nodes. In each simulation result, X-axis represents the network traffic and Y-axis represents the throughput in bytes. In our simulation, we have increased network traffic from 400 bps to 51.2 Kbps. With this increasing traffic we have measured the throughput in bits/second on Y-axis.

Figure 7.7 shows the simulation results for the throughput in bits/second for 5 to 20 one hop neighbor nodes. Result shows that class one achieves the highest throughput among all the priority classes. As result shows the throughput increases up to traffic 12.8Kbps and then it starts decreasing. Energy result also reflects the same results.

In chapter 6 we have gone through the modeling of IPCAP. We have modeled IPCAP as two dimensional Markov chain and derived equation for various parameters. In Table



(a) 5 One Hop Neighbor Nodes

(b) 10 One Hop Neighbor Nodes

(c) 15 One Hop Neighbor Nodes

(d) 20 One Hop Neighbor Nodes

Figure 7.7: Throughput for Different Priority Classes

7.1 we have compared the throughput results derived analytically with the simulation results. Table shows our simulation result varies by approximately 10% with the numerical results.

Nodes	Analytical	Simulation	diff
5	7797	7778	8.2%
10	7358	7380	9.4%
15	7049	7028	9.1%
20	3822	3799	10.0%

(a) Class 1

Nodes	Analytical	Simulation	diff
5	7405	7380	11.0%
10	7246	7225	9.1%
15	6625	6647	9.7%
20	3629	3605	10.3%

(b) Class 2

Traffic	Analytical	Simulation	diff
5	6994	6980	6.0%
10	6963	6980	7.4%
15	5812	5789	10.2%
20	2994	2969	10.7%

(c) Class 3

Traffic	Analytical	Simulation	diff
5	6543	6526	7.3%
10	6101	6078	9.8%
15	5342	5326	6.8%
20	2873	2895	9.5%

(d) Class 4

Traffic	Analytical	Simulation	diff
5	6242	6225	7.3%
10	5349	5326	9.8%
15	4896	4881	6.8%
20	2576	2598	9.5%

(e) Class 5

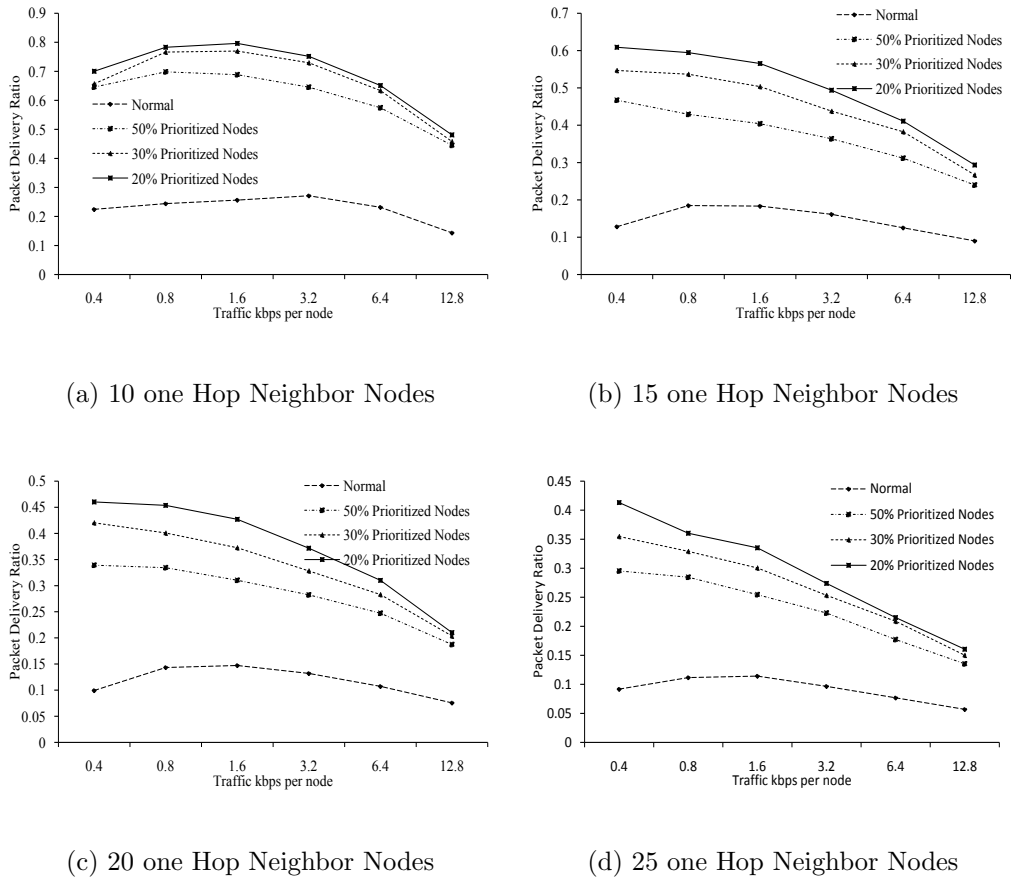
Table 7.1: Throughput Results Analytical and Simulation for Class-1

In this section we have gone through simulation results of IPCAP. Our study shows that IPCAP derives higher delivery ratio for the prioritized nodes that increases the delivery ratio of critical events. Other supporting results are also shown with the packet delivery ratio results. We have also compared the simulation results with the analytical results. Comparison shows that simulation results varies by approximate 10% with the analytical results.

7.3.2 EPCAP

Packet delivery

Figures 7.8 shows the simulation results for packet delivery ratio. In the result, X-axis shows the traffic intensity, and Y-axis shows the packet delivery ratio. We have tested our scheme from very low traffic to high traffic conditions. Here packet delivery ratio gives the fraction of packets delivered to coordinator nodes. Result shows that standard has low packet delivery ratio than the prioritized approach.



(a) 10 one Hop Neighbor Nodes

(b) 15 one Hop Neighbor Nodes

(c) 20 one Hop Neighbor Nodes

(d) 25 one Hop Neighbor Nodes

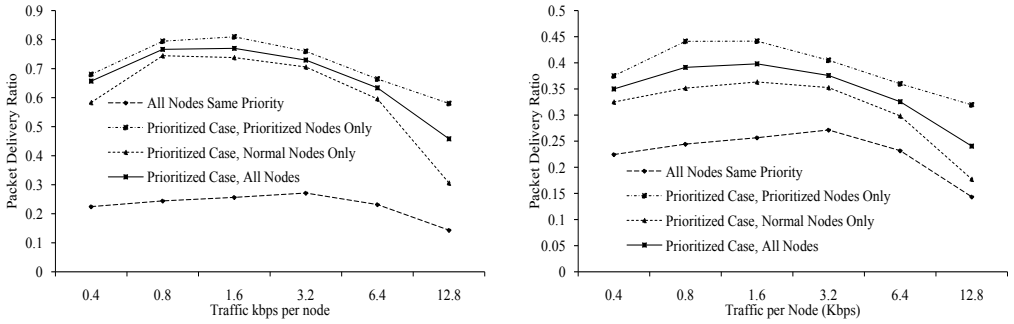
Figure 7.8: Packet Delivery Ratio

We have tested this simulation result for the networks having 10, 15, 20 and 25 one hop neighboring nodes. For each configuration we have tested the network by increasing the number of prioritized nodes in the network. Because prioritized scheme controls the number of nodes contending for the channel. It reduces the number of contenders for each slot which increases the packet delivery in prioritized mechanism.

Result shows that as the traffic increases, the delivery ratio decreases for the nodes. The delivery ratio depends on number of nodes. As the number of nodes increases in the prioritized case the delivery ratio decreases. In all the cases simulation result shows that prioritized scheme has high packet delivery count.

Prioritized packet delivery

For EPCAP, second set of results are of prioritized packet delivery in given priority class. Previous result shows the comparison between average common delivery and prioritized delivery. There, comparison was between the network having all nodes of same priority level with network having some nodes as higher priority nodes. With priority scheme, we have two types of nodes in the network one with the higher priority and the other with the normal priority or low priority. Previous result shows that the network with higher priority nodes have higher delivery ratio as the contending nodes are less in given slot. But increased delivery comes at the cost of low packet delivery of normal nodes. Following set of result shows the comparison between average delivery of prioritized nodes and average delivery of normal nodes.



(a) 30% Prioritized Nodes (b) 50% Prioritized Nodes

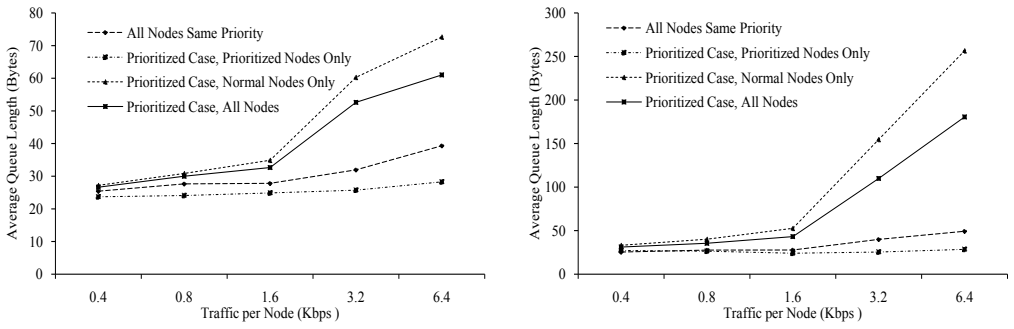
Figure 7.9: Packet Delivery Ratio for 10 one Hop Nodes with 30% and 50% nodes are prioritized nodes

Figure 7.9 shows the result for prioritized packet delivery in given priority class. These results show the comparison between average delivery of prioritized nodes and average delivery of normal nodes. Simulation result 7.9(a) shows that priority scheme (Prioritized

case, All nodes) has higher packet delivery ratio than the network having all the nodes of same priority level, same is discussed in previous subsection 7.3.2. Here, network has 30% prioritized nodes and rest of the nodes are normal nodes. As result shows that increase in delivery ratio of prioritized nodes comes at the cost of normal nodes. Same pattern is there in result 7.9(b) showing 50% prioritized nodes, here the difference between the prioritized and normal nodes is smaller compared to 30% nodes. Increase in average packet delivery of prioritized nodes helps in improving the results of overall packet delivery.

Queue length and delay

Next simulation results show the average queue length and delay experienced by the network. Figure 7.10 shows the average queue length result for the 10 one hop neighboring nodes. Figure 7.10(a) shows the queue length result with 30% prioritized nodes and figure 7.10(b) shows the queue length result for 50% prioritized nodes. The result shows that the prioritized nodes have smaller queue length compared to other scenarios. Here, the prioritized nodes are getting higher chances for their packet delivery that helps in reducing the queue length of prioritized nodes. Prioritized nodes get their chances at the cost of normal nodes. That increases the queue length of the normal nodes and the over all average queue length.

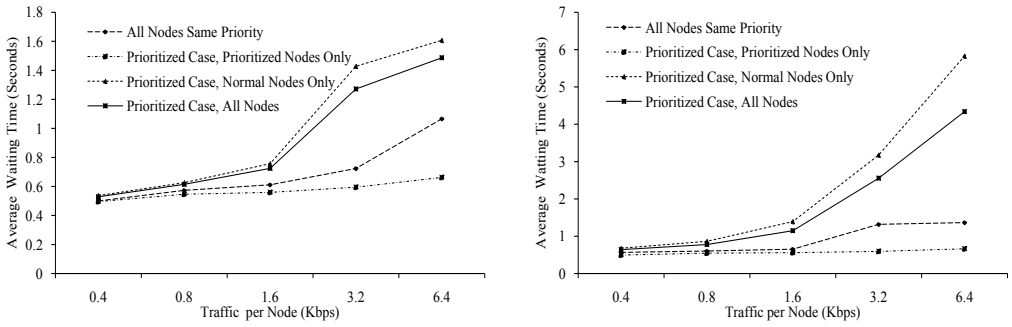


(a) 30% Prioritized Nodes (b) 50% Prioritized Nodes

Figure 7.10: Average Queue Length at Node with 10 one Hop Nodes

Figure 7.11 shows the average waiting time result for the network having 10 one hop neighbor nodes. Figure 7.11(a) shows the waiting time result for 30% prioritized nodes

and figure 7.11(b) shows the waiting time results for the 50% prioritized nodes. The result shows that prioritized nodes have smaller waiting time in the queue than the other cases. Here network achieves the smaller waiting time for the prioritized nodes on the cost of higher waiting time for the normal nodes, that increases the overall waiting time of the network. With prioritized nodes the average waiting time that the network achieves is higher than the waiting time without any prioritized nodes.



(a) 30% Prioritized Nodes (b) 50% Prioritized Nodes

Figure 7.11: Average Waiting Time at Node with 10 one Hop Nodes

In this section we have gone through the simulation results achieved for EPCAP. Next we have compared simulation results with the analytical results. In chapter 6 we have modeled EPCAP as M/G/c model. Model gives system of linear equations. Using IPCAP results and system of linear equation we have derived value of queue length for priority classes. This section shows comparisons between analytical results and the numerical results. It shows that both the results are varying upto 12%. Following table shows our comparison.

Traffic	Simulation		Analytical		Difference	
	Normal	Prioritized	Normal	Prioritized	Normal	Prioritized
0.4	33	27	37	24	10.1%	10.2%
0.8	40	26	42	25	5.1%	4.2%
1.6	52	24	58	27	10.3%	10.2%
3.2	154	26	169	28	9.2%	8.7%
6.4	257	29	280	31	9.1%	9.6%

(a) 15 Nodes

Table 7.2: Queue Length Results Analytical and Simulation for EPCPA with 50% privatized nodes

7.4 Summary

We have simulated IPCAP and EPCAP in Qualnet network simulator provided by Scalable network technology. The chapter covers our major simulation results. Here main emphasis is on critical event delivery. Results show that both the protocols deliver more number of prioritized packets to the base station than the standard. That indicates that IPCAP and EPCAP increase the delivery chances of critical events in event driven wireless sensor networks. Furthermore key performance parameters are discussed both for analytical and numerical results.

Chapter 8

Topology Control for Wireless Sensor Networks

8.1 Introduction

The underlying design philosophy of WSN is to create networks that consist of large number of small and low-end devices called sensor nodes. These sensor nodes are made of computing, storage, sensing, communication and power units. Computation and storage units are characterized by their computation speed and storage capacity respectively. Sensing unit is characterized by its sensing accuracy while communication unit is mainly characterized by its communication range and communication rate. Lastly, power unit is characterized by its power capacity.

In sensor networks, these device-level characteristics are of low ends. They are neither capable enough to handle long distance, high rate communication nor able to process high volume of data at high speed. We can call such units as resource stressed units and devices resource stressed devices. While networks that consist of such devices can be called as resource stressed networks. Resource stressed sensor network requires efficient utilization of these scarce resources, which is always the driving factor for many of the solutions designed for wireless sensor networks (WSN).

One of the ways through which sensor networks have overcome its limited resource

problem is by large number of nodes and their cooperation and coordination. Large number of nodes makes the network deployment dense. This dense deployment makes some nodes to overlap in communication and sensing range. Because of that, nodes make redundant sensing and create unnecessary data communication.

Further WSN is application specific, data centric network and it requires data aggregation or data fusion for efficient use of available resources. Dense deployment makes that task hard to achieve efficiently. Large number of nodes create problem of redundancy and resource under utilization, which reduces network's effective throughput. WSN solution providers try to optimize this tradeoff at every level of network design. Topology control is one of the ways through which researchers are trying to optimize this tradeoff in WSNs.

In large number of applications, the network is required to be randomly deployed, self-configured and battery powered. Due to limited communication capability and energy resources, the network frequently faces link failure that changes network's topology. Frequently changing network topology makes it hard to maintain topological information at one place. Hence, centralized solutions have limited scope in such a working scenario.

Further nodes in wireless network communicate with each other through direct communication or through multi-hop wireless communication. Here, transmitted packet can be heard by all the neighboring nodes in the network. Due to application's requirement or to maintain network functionality, many times it is required to flood the information in the network. Query and event flooding, data collection, data dissemination and network monitoring are few cases in which network is required to perform broadcasting and flooding.

Redundant data, contention, collision, longer on-time and energy wastage are some of the problems caused by flooding [67, 8]. This problem becomes severe as number of nodes increase in the network. As the number of nodes increases more nodes participate in data forwarding task. Normally they are referred to as active nodes. Objective of topology control mechanism is to reduce the number of active devices without compromising network functionality.

In the literature, topology control problem has been solved by probabilistic counter

based, location based, cluster based and dominating set based methods. Assumptions made in some of these techniques may not be valid for wireless networks. Like IEEE 802.15.4 does not provide full support to change transmission power. Or other schemes may require information about the entire network. So, many of these techniques may not be applicable to IEEE 802.15.4 based networks.

Considering limitations of wireless networks, we have proposed a distributed algorithm to construct a Connected Dominating Set(CDS) for the randomly deployed network nodes. CDS restricts the number of nodes involved in packet forwarding and flooding. Controlled packet transmission improves the overall networks energy utilization and extends the network life. For better explanation of the algorithm, we have discussed it using IEEE 802.15.4 terminology but the algorithm is applicable to general class of wireless networks having single sink.

In IEEE 802.15.4 based networks, as the number of FFD nodes increase more nodes will participate in packet forwarding or flooding. A brief overview of IEEE 802.15.4 standard is given in chapter 5. Objective of topology control mechanism is to reduce the number of FFD nodes without compromising network functionality.

8.2 Related Work

To reduce unwanted traffic and energy usage, various topology control techniques have been proposed in literature. Mainly these methods are classified as hierarchical, cluster based methods or dominating set based methods. In cluster based method, the network consists of cluster heads and cluster members. In the case of dominating set based method, the network creates a Connected Dominating Set (CDS) that is a subset of network nodes. CDS are helpful in various ways like reducing the routing overhead [68], for energy efficiency [69] and for better approximation of area coverage [70]. Results show that dominating set provides better fault tolerance and routing flexibility.

We use the concept of backbone structure. One can find its root in power distribution networks. Gradually, packet cellular network started using that concept and has become

one of the successful technological revolutions. It was in late eighties that the Ephremides et al. first noticed the use of backbone structure in wireless routing and introduced the concept of virtual backbone [71]. Later on, various approaches to construct CDS have been proposed in literature. Some of them are based on Dynamic Programming [5], Linear Programming [6], Linear Programming Relaxation [6], Maximum Independent Set(MIS), or Multi-Point Relay (MPR).

Linear Programming and Dynamic Programming are centralized approaches which require information about the entire network at one place which is not feasible for wireless networks. Here our focus is on distributed methods of CDS construction. So, we will mainly discuss the other two solution methods Maximum Independent Set(MIS) based methods and Multi-point Relay (MPR) based methods.

Some other topology control solutions like [72, 73, 74] are specifically designed targeting IEEE 802.15.4 based networks. Jian Ma et. al [72] proposed three centralized pruning based topology control algorithms with $O(1)$, $O(n)$ and $O(\sqrt{n})$ running time. Here n is network size. Li Deying et. al in [73], using Euclidean plane, modeled the topology control problem as power assignment problem. Using simulation results they have shown the effectiveness of the proposed scheme which creates strongly connected network consuming minimum total power. A. Haffiz et. al [74] used Super Frame resolution algorithm to target the problem of interference.

In the case of MIS based methods [68, 7, 75, 76, 77] the network first identifies the independent set based on certain node properties like identity, node degree or their location. Once the network identifies independent nodes, it connects this set to provide connected network.

Peng-Jun Wan et. al. [68, 7, 75] analyzed the complexity and approximation ratio of previous algorithms and proposed the distributed algorithm for connected dominating set generation. Many modifications of [68] have been proposed in literature. Main changes are in the comparison method used in node selection rules.

In their original work [68], node identity was the comparison parameter while others have tried with node degree, node location or available energy. A variant of it uses X

and Y coordinates in case the node degrees are same for the nodes. Yi Zou et. al. in [76] proposed coverage-centric active node selection method using the concept of connected dominating set. They showed that when communication range is more than double of the sensing range the coverage itself provides the connectivity.

Funke. S [77] proposed constant approximation based CDS generation algorithm. Their algorithm is an improvement over [7] using the relation between the MIS and Minimum Connected Dominating Set(MCDS). MIS based methods can be implemented in a distributed way and may have approximation ratio smaller than MPR based methods. The main drawback of this approach is that it increases the network diameter which leads to higher energy consumption in packet transmission and data collection. MPR based methods helps in achieving smaller network diameter.

In the case of MPR based methods [8, 9, 78, 79, 80, 81, 82], each node identifies the relay nodes from its one hop neighbors. Mainly these methods are classified as source dependent methods and source independent methods. In source dependent methods, relay nodes are identified for each node considering source nodes, while in case of source independent methods, relay nodes are selected independent of source nodes. These selected nodes then propagate the information in the network.

Amir et. al. [9] proposed multipoint relay based broadcast flood control mechanism. First, they showed that to generate MPR is an NP-Hard problem. Then they gave an approximation algorithm with $log\ n$ approximation ratio, here n is number of nodes in the network. Many solutions afterwards have extended their work to generate CDS. Later on, Cedric et. al. in [78] proposed MPR based algorithm to generate a connected dominating set. They have defined forwarding rules for pure flooding and Multi Point Relay based flooding. They have proposed algorithm to compute MPR. It selects nodes in increasing order of their identity. They compute the MPR set for current node in $O(m^3)$ time, where m is the node degree.

Je Wu [82] has enhanced relay node selection by providing new rules which they called rule 1 and rule 2. They have used extended marking process to generate the dominating set. In their analysis they have also shown the comparative analysis between cluster

based and dominating set based methods. In their process, they have marked their nodes to be a part of dominating set based on node identity and node coverage. Dia F and Je Wu in[83] gave the extended localized algorithm for connected dominating set which is an extension of their previous work [82]. Here they have extended rule 1 and rule 2, in their general form as rule k.

Das B et. al. in [80] have proposed three distributed algorithms for MCDS construction. First two algorithms are the distributed implementation of Guha and Khullers [79] algorithms and last one is bounded degree MCDS construction algorithm. K Mohammed et al. in [81] have proposed algorithm to generate the CDS considering network diameter, risk-factor and interference. It is one of the early works in which solution uses other parameters besides minimizing approximation ratio of generated CDS. Algorithm in [79] generates the tree similar to depth first way, while in [81], author has used breadth first approach to reduce the diameter of the network. Stojmenovic et. al in [8] have proposed internal node based broadcasting method and they have compared their scheme with the cluster based scheme.

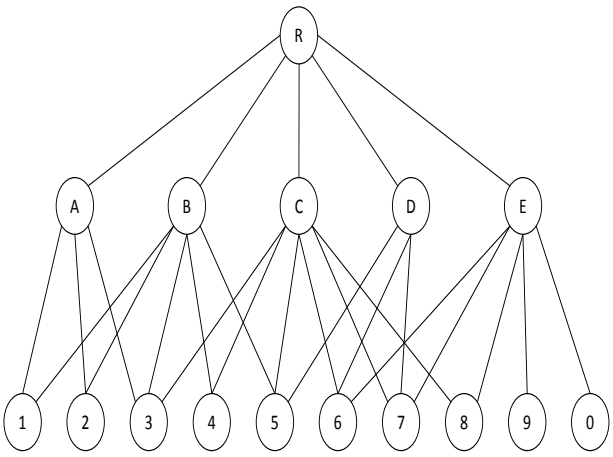


Figure 8.1: Example showing problem with greedy selection

Majority of above MPR based CDS construction schemes selects the cover nodes or

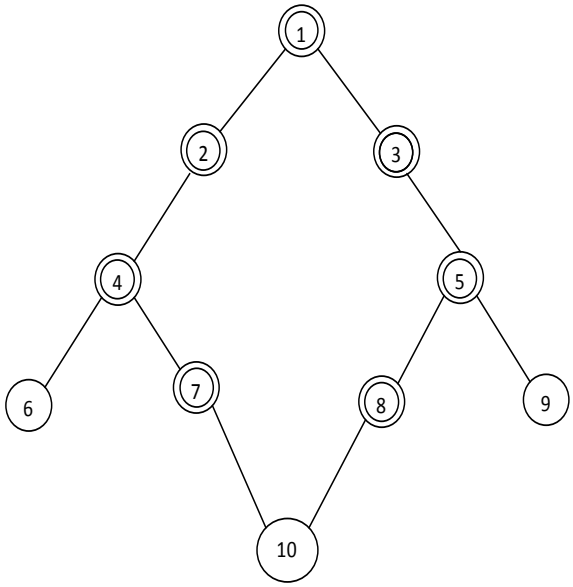


Figure 8.2: Example showing problem with limited information

relay nodes using greedy approach. In their process of relay node selection, they select the replay nodes based on node identity, node degree, number of covered nodes, location of nodes, or based on physical coverage provided by the nodes. For example [80] uses unmarked neighboring nodes, [82] and [83] uses node identity.

In general in all the cases these methods are extensions or modifications of greedy approach. As it is known that greedy approach has problem of selecting sub-optimal results. For example [9] selects relay nodes based on node degree and many times it selects those higher degree nodes which are not required as part of solution. But due to greedy approach they become a part of the solution. These are the redundant nodes in the final selection. As shown in figure 8.1 here, due to greedy selection node C is getting selected though node B and node E provides the required cover. Other approaches like Dynamic programming help in this scenario but they are mainly centralized approaches. Other solutions [82, 83], solve this problem but they require more than 2 hop information to solve it.

Secondly MPR based methods are distributed methods and they have limitation on their local knowledge. Here, due to limited information many overlapping nodes are

getting selected by the algorithm. As shown in figure 8.2 one of the nodes 7 or 8 is redundantly getting selected in CDS.

Wireless network requires distributed localized algorithm that should rely on localized information. To solve this redundant node problem and overlapping node problem we have proposed distributed localized multi-point relay based topology control algorithm ROOT-I to construct CDS for wireless sensor network. The algorithm constructs the connected dominating set that helps in reducing the broadcast flooding traffic, by controlling the broadcasting nodes participating in the flooding process.

8.3 ROOT-I Topology Control Algorithm

Proposed ROOT-Initiative(ROOT-I) algorithm is multipoint relay based topology control algorithm. ROOT-I generates the connected dominating set using two hop information. Algorithm divides the steps of CDS generation into four phases namely level assignment, ranking of node, cover selection and decision making.

First phase is of level assignment which assigns level to each node. Level is a node's shortest hop count from the base station. Second phase is to determine the rank of the node. In this phase, node decides its rank with respect to neighboring nodes. In the third phase, node collects two hop neighboring information and calculates its covering nodes. Here covering nodes are one hop neighboring nodes having two hop neighbors of a given node as neighbors. The last phase is decision making phase in which node decides whether they should work as FFD or be a neighboring node of one of FFD nodes. Following is the detailed explanation of each of the above steps.

8.3.1 Level assignment

First phase is of level assignment. Here, each node receives its shortest hop count distance from the root node (PAN coordinator) and propagates this information further in the network. Root which is normally the base station initiates this process in the network. To decide which nodes should work as root node is a problem in itself called leader election

problem.

Normally in IEEE 802.15.4 based networks PAN coordinator works as root node. Root node assigns itself level zero and broadcasts the message. On receiving level message, node calculates the new level. If it finds that the new level is smaller than node's current level it updates its current level and broadcasts this message to neighboring nodes. Nodes in the network follow the same process and process terminates at leaf node when it finds that all the neighboring nodes have decided their level.

Major advantage of using leveling in the network is that it helps to control the network diameter. Primary task of sensor network is to collect information from the network. Network node collects the required information and forwards that information towards the base station. Base station records the network wide information for application specific tasks. Small network diameter helps in reducing the cost of packet forwarding and routing updates in the network. Approaches available in literature, mainly independent set based approaches ignore this requirement that increases their overall energy consumption though they have smaller CDS size. Another advantage of assigning level to the network node is that it helps in giving unique rank to the network nodes.

8.3.2 Ranking of the node

An important component of ROOT-I algorithm is rank assignment to the nodes. Each node gets its rank from the information that it collects from the neighboring nodes. In the process, the node collects one hop neighbor information and its depth from the root node.

Rank of a node consists of three tuples $< Level, Degree, ID >$. Here level is node's shortest hop distance from the base station, degree is nodes one hop neighbor count and ID is a unique number assigned to the node. Combining these three pieces of information each node gets its unique rank in the network. Nodes are compared in order of level, degree and ID. Node N_1 is higher rank node than node N_2 based on following criteria.

$if\ N_1.Level < N_2.Level$

$elseif\ N_1.Level = N_2.Level\ AND\ N_1.Degree > N_2.Degree$

$elseif\ N_1.Level = N_2.Level\ AND\ N_1.Degree = N_2.Degree\ AND\ N_1.ID > N_2.ID$

For example consider node $1 < 1, 3, 1 >$, node $2 < 1, 4, 2 >$ and node $3 < 2, 5, 3 >$. Here node 2 has higher rank than node 1 and node 3 while node 1 has higher rank than node 3. Rank of the node highlights its relative importance among the neighboring nodes.

8.3.3 Cover selection

Next step in the process of CDS construction is the cover selection. As mentioned above, a cover is the subset of one hop neighbors whose combined one hop neighbor set contains all the two hop neighbors of a given node. To calculate cover set nodes in the network, we require their two hop neighbor information. To get the required information, the nodes broadcast their one hop neighbors. Doing that all the nodes in the network will get their two hop neighbor information. After collecting two hop information, node generates the two hop graph, considering itself as root node of that graph. Then node uses greedy approach on its one hop neighbors to find cover set.

In the process of cover set generation, node first selects the essential nodes. Here, essential nodes are those nodes which cover two hop neighbors not covered by any other nodes. After selecting essential nodes, ROOT-I algorithm marks those selected nodes as essential nodes. Next, to cover remaining two hop neighbors, node selects the covering nodes based on the node ranking, these are called greedy nodes. At the end of iteration, the selected node set contains the essential nodes and greedy nodes. Then the algorithm removes the other one hop nodes from the two hop graph and starts the next iteration. The algorithm repeats this step until only essential nodes remain as marked nodes.

A majority of the previously proposed greedy selection algorithms faces the unwanted node problem in the final solution. This step removes those unwanted nodes that appear in greedy selection. For example, node R as shown in figure 8.1 first selects node E, C and B in first iteration. Here it marks the node E as an essential node and node C and B as greedy nodes. In the next iteration, from node C and B it selects node B as essential node. In this step, node E and B together covers all the two hop nodes of node R. In other words all the two hop nodes are covered by essential nodes that ends the

iteration. By this we have overcome the problem of greedy selection and have acquired local optimal solution. Though it removes redundant nodes from the local solution but still the global solution contains overlapping nodes. Algorithm refines these nodes in final phase. These overlapping nodes are mostly ignored by most of the algorithms proposed in literature and those who have considered it require more than two hop information.

8.3.4 Decision making

With limited information, nodes only have two hop information. In many of the occasions they select unnecessary nodes as FFD. Previously proposed MPR based algorithms have not addressed this problem in their work. In our work we have used reverse notification mechanism that removes these unnecessary nodes from the CDS.

Final phase in the algorithm is of decision making. Here node makes the decision whether they should work as FFD or RFD. In the decision making, selected one hop neighbor nodes send their selection information to two hop neighbor nodes. On receiving this information all the two hop nodes know the nodes which cover them. Based on rank they choose the best among all available options. Here the cover selection process is not one way. It is a two way process.

Following example explains the point further. Here, network has 10 nodes spanning in five levels. Node 1 is the root node. Label beside the node gives the three tuple ranking information of the node.

In figure 8.3, nodes which works as FFD and part of CDS are shown in double circles, other nodes which work as RFD are shown in single circle. As shown in the figure, node 1 selects node 2 and node 3 as it's covering nodes to cover it's two hop neighbor node 4 and node 5. At level 3, node 7 and node 8 both are selected as part of CDS to cover the node 10 which is two hop neighbor node of node 4 and node 5. As node 4 and 5 do not have this information, both the nodes get selected by CDS.

To overcome this, node 10 selects one of the nodes to cover it. Here, in decision making phase, second level nodes inform the covering node about their maximum rank covering node. If one hop node does not receive any notification from two hop nodes with

Algorithm 8.1: ROOT-I Topology Control Algorithm

Phase 1: Level assignment
Root Node broadcast $level_{msg} < id, level >$
On $level_{msg} < id, level >$ calculate new level
if *new level* $<$ *current level* **then**
    update current level
    Broadcast $level_{msg} < id, newlevel >$
end if

Phase 2:Ranking of Node
All nodes send $neighbor1_{msg} < id, degree, level >$
On $neighbor1_{msg} < id, degree, level >$,
based on ranking rule categorize neighboring nodes as
sibling,parent or child list

Phase 3:Coverage Selection
Collect two hop neighbor information by $Neighbor2_{msg} < id, neighbors >$
Select essential nodes
while not covered all two hop neighbor **do**
    Select one hop neighbor with highest rank
    Update cover list
end while
Mark selected nodes as essential nodes and greedy nodes
Remove other nodes from One hop set
Repeat 17-24 until all the selected nodes are essential nodes
Send Cover information to one hop neighbor
Selected one hop nodes broadcast their covered node list

Phase 4: Decision Making
Select higher rank node from received covering nodes
Send notification msg to selected node
if no notification msg received **then**
    be non CDS node
end if

maximum nodes as itself, it will withdraw itself from CDS. It does so because nodes to which it was providing cover have selected other node to be the covering node.

In the example node 10 has two covering nodes, node 7 and node 8, for node 4 and node 5 respectively. Here node 8 has higher rank than the node 7 so node 10 selects node 8 as its covering node. Node 7 does not have any node to which it provides cover and will withdraw itself from the CDS and become RFD. Same can be seen in figures 8.4 and 8.5. Here, dots are RFD nodes and dots with circle are FFD nodes.

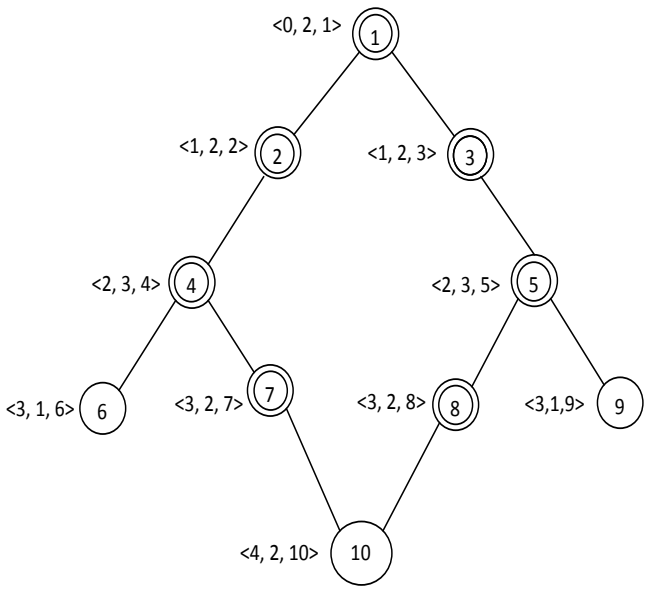


Figure 8.3: Example showing working of ROOT-I

Figure 8.4 shows that all the two hop nodes are selected as FFD nodes. Same topological configuration is there in figure 8.5 but here ROOT-I removes the overlapping nodes from the solution.

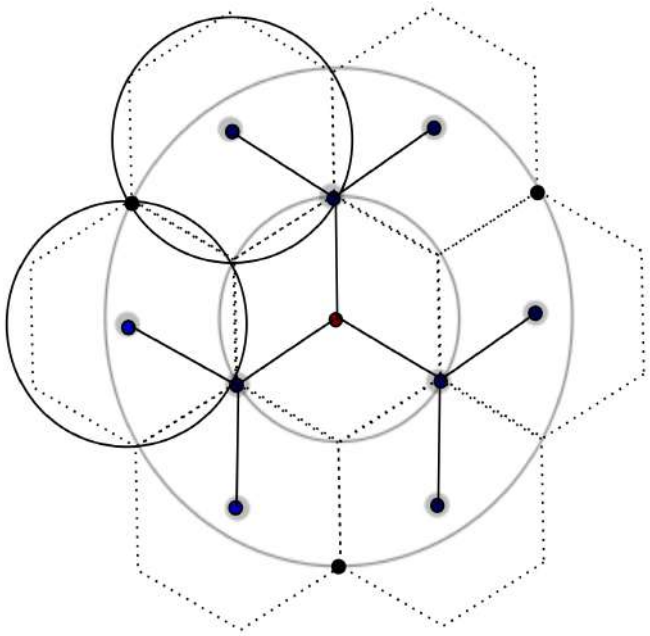


Figure 8.4: Example showing overlapping nodes

Algorithm 8.1 is our ROOT-I algorithm for the CDS generation or FFD selection.

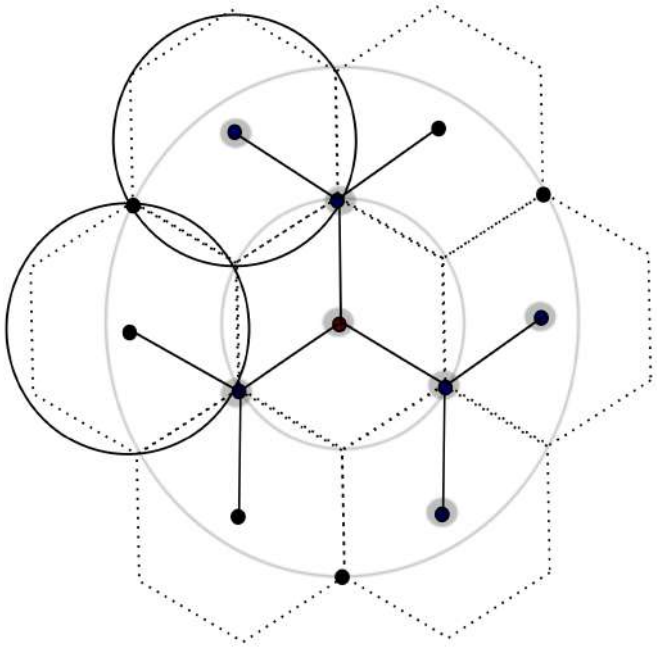


Figure 8.5: Broad view showing working of ROOT-I

In ROOT-I algorithm root initiates the process and gradually decision making phase moves towards the leaf nodes. ROOT-I algorithm uses level assignment and ranking phase for the coverage selection and decision making. Node sends the covering node list to its one hop neighbors and they send the cover by message to level two nodes. Using this information two hop neighbors send the reverse notification message to one hop neighbor of the nodes. This reverse flow of decision making helps in reducing unwanted covering nodes and gradually decreases the CDS size. Here, we have achieved the better performance within available information. In many other solutions it requires more than two hop information to achieve required results.

8.4 Fault Tolerable Topology Control Algorithm

As discussed above wireless sensor network collects the information at the base station. Sensor node senses the data and forwards it towards the base station. As information moves towards the base station, investment made by the network in terms of energy increases. It is well known that chances of packet loss increases as hop count increases. If information loss occurs near the base station it will cost more to the network. Network

should try to reduce the packet loss near the root node to get the benefit of the investment that network has put in getting information near the root.

8.4.1 K-connectedness

In case of 1-connected network, one and only one node is expected to receive the packet. Due to bad channel or node failure it may happen that the required receiver may not receive the intended packet. In 1-connected network there are high chances of packet loss. Improvement over 1-connected network is k-connected network. Here objective is to ensure that if one path in the network breaks then there must be some alternate paths through which a packet reaches to the base station.

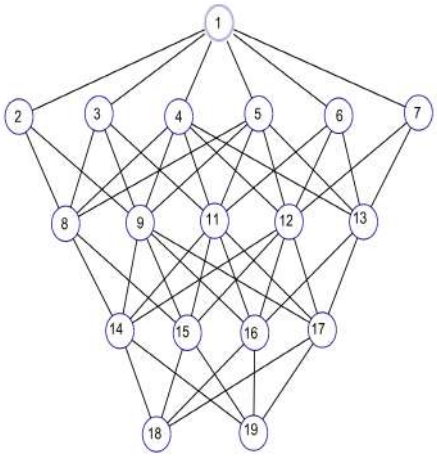


Figure 8.6: Example showing CDS-k with k set to 4

Figures 8.6 to 8.9 show the connectivity graph for various k-values. Figure 8.6, 8.7, 8.8 and 8.9 are of 4-connected, 3-connected, 2-connected and 1-connected graphs respectively. In figure 8.6 which is of 4-connectedness, each node is required to be connected to 4 upper layer nodes. Here, all the nodes are working as FFD nodes and hardly any node can configure itself as RFD node. In figure 8.7 the network is 3-connected and one node at every level can go to off mode. As shown in figure 8.8 and 8.9 number of RFD nodes increases for 2-connected and 1-connected networks.

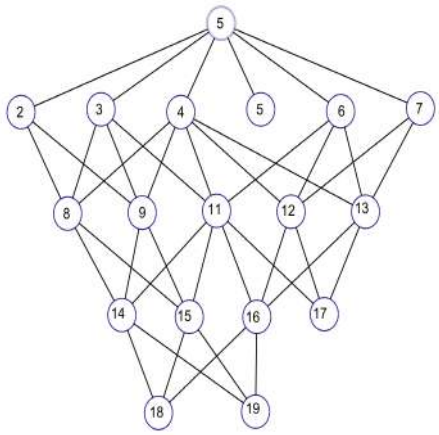


Figure 8.7: Example showing CDS-k with k set to 3

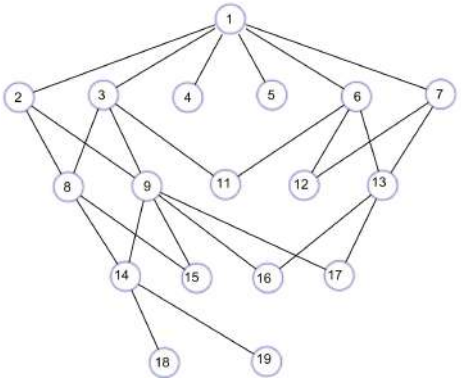


Figure 8.8: Example showing CDS-k with k set to 2

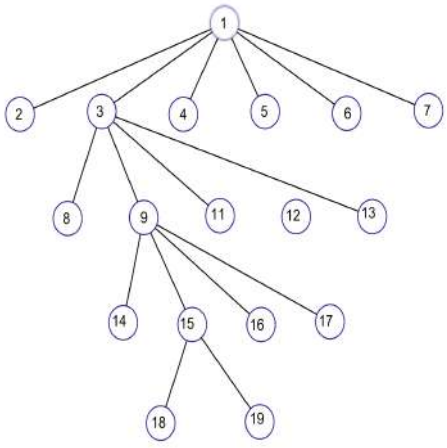


Figure 8.9: Example showing CDS-k with k set to 1

8.4.2 Problem with k-connectedness

In k-connected network, k-connectivity is achieved at every level of the network from leaf node to root node. Here, starting from the leaf nodes every node at given level is connected to k-nodes of the level above it. Each level maintains k-connectivity that provides k path between source and destination.

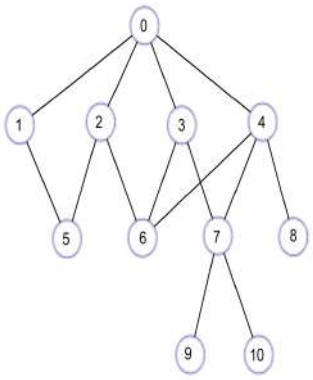


Figure 8.10: Example of CDS-k with k set to 2

Figure 8.10 shows the 2-connected network. Every node in the level is connected to at least 2 nodes of immediate upper layer. This rule is not followed at level 1, where all the nodes are connected to the base station. Node 7 is connected to two nodes of level one node 3 and node 4. Node 3 and 4 are connected to root, node 0. Here the broadcasted packet from 7 is received by node 3 and 4 and both will try to broadcast that packet to deliver it to the root node. With the higher value of k, number of rebroadcasted packets is high.

Each packet received at the base station has cost associated with it. This cost consists of two components: a) cost that the network has invested in to bring that packet at the base station and b) additional cost that the network has invested to prevent packet loss.

In k-connected case the first part increases as the information goes near the base station but the second part remains constant. At each level, network node is connected with k immediate upper layer nodes making constant provision to overcome the packet loss.

Here, broadcasted packets move simultaneously on k paths towards the destination. At each level, network invests equal amount of energy in packet forwarding. This mech-

anism tries to prevent packet loss at each level with the same priority. As packet moves towards the base station, the total energy invested by the network keeps on increasing.

Normally, it has been observed that as the packet moves near the base station its chances of getting lost increases. The constant provision to decrease the packet loss does not help much in such circumstances. It requires dynamic adaptation scheme that adapts itself based on packet loss probability. Literature available on k-connectedness has not addressed this problem at its depth.

In our work we have focused on analyzing the energy investment of the network while making decision about the fault tolerability of the network. As network invests more energy in information forwarding, network should also make sufficient provisions so that the investment made in information forwarding should not get lost.

On this line our first observation is that near the base station the chances of packet loss are higher and at the leaf the chances of packet loss are less. Second during its path towards destination network invests some energy at each level. Here near the leaf the total investment is very low and near the root investment is high. Considering that to reduce the packet loss and increase the fault tolerability of network we have extended ROOT-I to ROOT-Ik algorithm to provide k-connectivity.

8.5 Value of k in k-connectivity

In k-connectedness, at every level the node is connected to k parent nodes. At level one all the nodes are connected to only base station. As talked previously, in such a mechanism at each level network provides same amount of provision for fault tolerability. To improve over it we are dynamically configuring the k parameter of the network while maintaining the k-connectivity. At every level, node chooses its value of k, based on its current network parameter and traffic requirement with observed loss rate.

As the value of k-decreases the network has more nodes that can configure themselves as RFD nodes and as the value of k increases this number decreases. For higher value of k network has high fault tolerability but also has high energy cost. Low value of k saves

the network energy but network is less tolerable to link or node failure.

Instead of accepting one extreme or the other, we seek a method to construct a network topology that provides good trade-off of fault tolerability and energy consumption. Near the leaf where network has not invested much in the information transfer, losing information will not cost much to network, but as the information goes near to base station here network has invested more energy so requires more fault tolerability to the network. Based on that, value of k is the function of node level and fault tolerability required at each level. We have defined the value of k based on spatial and temporal network parameters. In first case, k is function of node proximity, that is node's location in the network relative to the base station. In second case, k is function of temporal information that is packet loss encountered at each level.

8.5.1 k based on node proximity

Network has two extreme values of k, one is K_{max} and other is K_{min}. K_{max} is the maximum connectivity count of the network and K_{min} is minimum connectivity count of the network. Maximum value that K_{max} can take is the minimum node degree of all the nodes and minimum value that K_{min} can take is one. In other words $K_{max}, K_{min} \in [min\ node\ degree, 1]$.

$$k = max(1, K_{max} * (\xi)) \tag{8.1}$$

Here, the value of $\xi \in [0, 1]$. Here ξ is fault tolerability parameter. If more importance is given to the fault tolerability then the network has higher number of FFD nodes and in other case network has less number of FFD nodes. In our case we require higher value of fault tolerability near the base station and relatively low fault tolerability near the leaf nodes.

The intuition is that, the further the information has to travel in the network, the more likely that it will encounter a failure. When information gets close to the base station, we want to give it the best possible chance to make it the rest of the way so

Table 8.1: Symbols used to derive value of k in k-connected network

Symbol	Comment
μ_i	Service Rate at level i
λ_i	Arrival Rate at level i
n_i	Number of Nodes at level i
T_i	Number of Active Transmission possible at level i
A^c	Interference Area of a Node
A^i	Level i Interference zone
Pr_i^{loss}	Packet loss probability at level i
D_i	Packets Dropped at level i
Pkt_i^A	Number of packets arrived at level i from level i-1
Pkt_i^D	Number of packets departed from level i
Pkt_i	Number of packets at level i

that it will be recorded at base station. Here value of ξ is the function of node level. we calculate the value of ξ at each node and for node i the value of ξ is ξ_i .

$$\xi_i = 1 - l_i/L \tag{8.2}$$

Here, L is farthest node from the base station, or in other words it is the diameter of the network with one end having base station and other end a leaf node. l_i is the level of node i. Using this value each node calculates the value of ξ_i and using that value of ξ_i the node calculates its connectivity count k. Effect of this on the CDS size, packet delivery and overhead are examined through simulation in chapter 9.

Here, value of k is derived from node's spatial information. This information is static throughout the lifetime of the network. Next we have derived the value of k based on temporal information. Here we have used traffic experienced at given level to derive the value of k.

8.5.2 k based on traffic

Next we have derived the value of k based on traffic experienced at each level. Table 8.1 explains the variables used in the derivation. Here objective is that at each level all the received packets are successfully delivered to the next level.

First of all we derive number of concurrent transmissions possible at the given level.

To make it simple we have assumed that communication and carrier sensing range are same for the nodes.

$$T_i = \frac{A^i}{A^c} \tag{8.3}$$

Using this and service rate, we can derive the value of total number of packets transmitted from a given layer.

$$Pkt_i = T_i \mu_i \tag{8.4}$$

At the given layer node generates the packet and node receives the packet from the previous layer. So total number of packets available at the layer is sum of this two.

$$Pkt_i^A = T_{i-1}\mu_{i-1}k_{i-1}(1 - Pr_i^{loss}) \tag{8.5}$$

$$Pkt_i^G = \lambda_i n_i \tag{8.6}$$

$$\begin{aligned} Pkt_i^{Total} &= Pkt_i^A + Pkt_i^G & (8.7)\\ &= T_{i-1}\mu_{i-1}k_{i-1}(1 - Pr_i^{loss}) + \lambda_i n_i & (8.8) \end{aligned}$$

This helps in deriving the number of packets loss at the given layer.

$$\begin{aligned} Pkt_i^D &= Pkt_i^{Total} - Pkt_i & (8.9)\\ &= T_{i-1}\mu_{i-1}k_{i-1}(1 - Pr_i^{loose}) + \lambda_i n_i - T_i\mu_i & (8.10) \end{aligned}$$

If we consider that packet loss is zero at each level then,

$$T_{i-1}\mu_{i-1}k_{i-1}(1 - Pr_i^{loss}) + \lambda_i n_i - T_i\mu_i = 0 \quad (8.11)$$

$$\frac{T_i\mu_i - \lambda_i n_i}{T_{i-1}\mu_{i-1}(1 - Pr_i^{loss})} = k_{i-1} \quad (8.12)$$

That gives the set of equations to derive value of k at each level $i = 1, 2..., L$. Using this value network defines the connectivity required at each level.

8.5.3 ROOT-Ik topology control algorithm

ROOT-Ik is extension of ROOT-I algorithm. ROOT-I is special case of ROOT-Ik with value of k set to one. ROOT-I algorithm for each node defines one forwarder node. Here, first two steps are same in both the algorithms ROOT-I and ROOT-Ik. In the third step the extended algorithm selects the k forwarder nodes instead of selecting one forwarder node. Same is with the final step. Here instead of sending one notification message the two hop node sends the k notification messages. Here k nodes are getting selected based on their rank. ROOT-Ik selected the value of k at each level or at each node based on the equation derived in the previous section.

8.5.4 Packet forwarding in k-connectedness

There are two places where the information of broadcasting and connectivity can be manipulated in the favor of network performance. First, as discussed above from two parts of energy consumption, one can update the second part. As done in ROOT-Ik, at each level the connectivity is modified based on node level. Here instead of making constant investment, network dynamically adjusts its investment based on the current environment. Secondly the broadcast packet count can be modified to control the broadcast flooding in the network.

For k-connected network, on hearing the packet from other node, node will broadcast

that packet if it is part of k-connected set. Each node has k-connected nodes in the above layer. Packet broadcasted by the node will be heard by all the k-nodes belonging to the above layer. On hearing packet from child nodes all the k-nodes try to broadcast that packet.

As wireless media is broadcast media and shared by all the nodes. Only one node can successfully transmit its packet while others have to wait for their chance to access the channel. Here objective with k-connectedness is to increase the chances of connectivity and make sure that the packet is successfully delivered to destination.

In our protocol we try to reduce these rebroadcasted packets probabilistically based on available topological information. Every node knows about their level, parents level and relaying mode. Based on this information nodes probabilistically decides whether to broadcast that packet or not. On hearing the packet from child, node will set the timer to broadcast the packet with probability one. In that waiting time, if it hears the same packet from nodes that belong to same level as that of the node then it decreases the probability of sending the packet after time out. In case if it hears the same packet from the parent then it reduces the probability of that broadcasted packet by higher value or makes it to zero so on timeout that packet will not be broadcasted. The probability is decreased based on the k value in k-connectivity.

As this k-connectivity based probabilistic broadcast control mechanism prunes the broadcasting nodes, this will reduce the rebroadcasted packets. With proper selection of probability decrement mechanism, the network can achieve required fault tolerability with reduced broadcast overhead. Effect of these can be seen in the packet delivery simulation results discussed in next chapter.

8.6 Summary

Topology control is the mechanism by which network manages its structure. We have proposed MPR based topology control algorithms ROOT-I and ROOT-Ik which generates 1-connected and k-connected network respectively. The chapter explained the working

of ROOT-I and ROOT-Ik algorithms. In k-connectedness tradeoff between energy consumption and fault tolerability is managed by dynamically selecting value of k based on partial and temporal information.

Chapter 9

Modeling and Simulation of Topology Control algorithms

9.1 Modeling

We have modeled network as Unit Disk Graph (UDG) $G = (V,\ E)$, where V is node set of reduced function and full function devices and set E represents bidirectional link between devices. IEEE 802.15.4 devices have 64 bit (later on mapped to 16 bit) MAC address so each device in the network has unique MAC address assigned to it. Devices are deployed in two dimensional plane with fixed communication range. Here, communication range is normalized to one unit and two nodes can communicate with each other if the distance between them is at most one unit.

Let $N_k(v)$ be the k hop neighborhood of node v. Special cases are one hop neighborhood $N(v)$ without any subscript and $N_0(v) = v$. Subgraph G_k^v of graph $G(V, E)$ is defined as $G_k^v = (V^v, E^v)$ where $V^u = \cup_{i=0}^{k} N_i(v)$ and $E^v = \{(x, y)|x \in N_{k-1}\ and\ y \in N_k, N_{k-1}, N_k \subset V^v\}$. Graph G_k^v is tree having node v as root node and rest of the nodes are union of k hop neighborhood of node v. Node which is already selected as coordinator node maintains a two hop neighbor set N_2. Using G_2^v node v selects the one hop neighbor nodes from $N(v)$ as coordinator set to cover its two hop neighbors $N_2(v)$. Selected coordinator node u remains as coordinator node if one of the $N_2(v)$ nodes selects the node u

as uplink node otherwise it works as a device node.

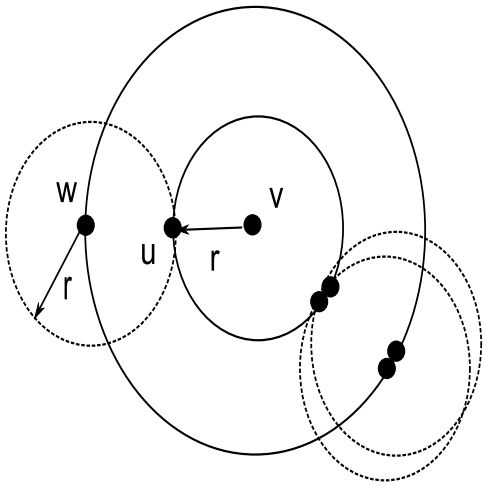


Figure 9.1: Node v local approximation ratio $O(n')$

Theorem 1. (Local Upper Bond): The ROOT-I has a $O(n')$ local approximation ratio.

ROOT-I works on two hop neighbor information. Let v be the node having transmission range r and let $n' = |N_2(v)|$, be the number of two hop neighbors of v. To show that local approximation ratio for ROOT-I is $O(n')$, let node v be at the center of $disk(v, r)$ and $disk(v, 2r)$. Here, $disk(v, r)$ covers all one hop neighbors of node v and $disk(v, 2r)$ covers all the one and two hop neighbors of node v, as shown in figure 9.1.

To analyze the worst case scenario, consider that all the one hop neighbors of node v are placed at the edge of $disk(v, r)$ and all the two hop neighbors of v are placed at the edge of $disk(v, 2r)$. To cover $N_2(v)$, node v selects nodes from $N(v)$. Selected node set is called coordinator node set and each selected node will work as coordinator based on ROOT-I algorithm. Each node in $N_2(v)$ is covered by only one node from $N(v)$. As shown in figure 9.1, node w will be covered by node u only. Based on it, as the number of nodes in $N_2(v)$ increases, corresponding coordinator set size increases. Therefor the local approximation ratio for ROOT-I is $O(n')$ $\square$.

Let c be the finite deployment region with $d = \frac{r}{\sqrt{2}}$. Union $U(c)$ and Intersection $I(c)$ region of c are union and intersection of $disk(u, r)$ of all $u \in c$, shown in figure 9.2. Region $I(c)$ contains level $i - 1$ nodes for given level i nodes of c. Here $I(c)$ is non overlapping

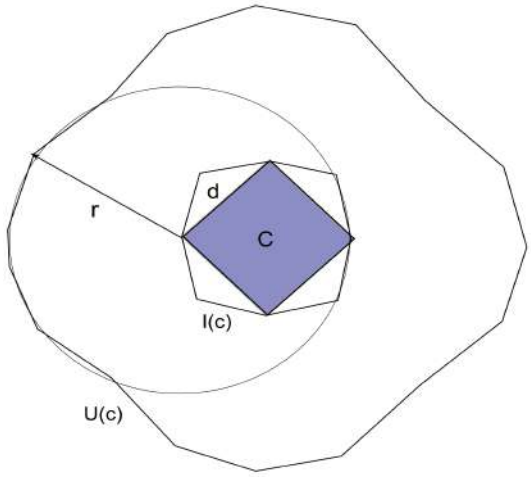


Figure 9.2: Union U(c) and Intersection I(c) region for finite region c having width $d = \frac{r}{\sqrt{2}}$

region, while $U(c)$ may overlap by neighboring region, as level $i+1$ may be overlapped by other level i nodes neighbor.

Figure 9.3, shows the three neighboring regions c_{-1}, c_0 and c_1. As the parent nodes at level $i-1$ don't know about neighboring node's $level_i$ nodes regions $I(c_{-1}), I(c_0), I(c_1)$ are independent and non-overlapping from each other. As the second level nodes may be overlapped by other node's coordinators, $U(c)$ may have an overlap from neighbor regions. The amount of overlap between neighboring regions is called overlapping factor $Of(c)$ for the region c. Overlapping factor for a given region depends on the position of neighboring regions. In other words it depends on the node density. The overlap factor for region c varies from 0 to $maxL$, as shown in figure 9.3. Here c_{-1} and c_0 with maximum overlap $maxL$ while c_0 and c_1 with minimum overlap 0. Here $ORank(c)$ represents the fraction of $U(c)$ covered by only nodes belonging to c.

$$Area\ of\ I(c) = 0.5r^2 \tag{9.1}$$

$$Area\ of\ U(c) = d^2 + 8dr + 4r^2 \tag{9.2}$$

$$ORank(c) = \frac{U(c)(1 - Of(c)) - \frac{1}{2}Of(c)U(c)}{U(c)} \tag{9.3}$$

Theorem 2. The number of coordinator nodes selected by ROOT-I in a finite

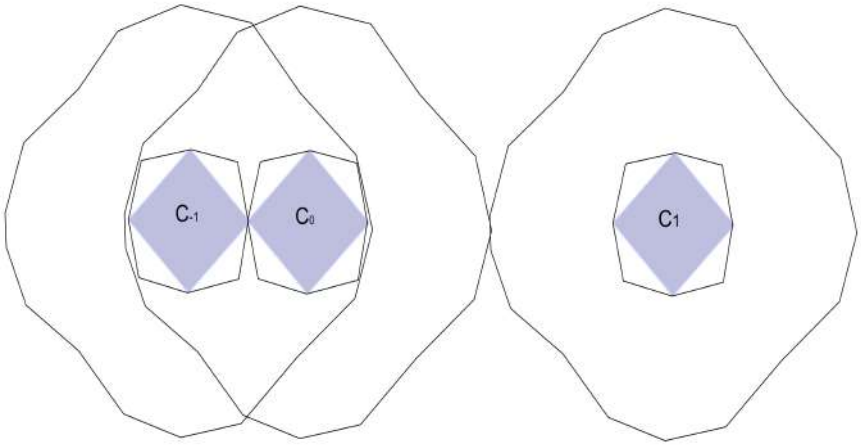


Figure 9.3: Overlapping with multiple regions

region c is bounded by $\mathbf{O(n) \cdot ORank(c)}$.

Let u be the node in c. For any node $v \in I(c)$, node u becomes the coordinator node based on following conditions, first, node u has same level with higher coverage and higher id node than any other nodes in c. In other word next highest node among the remaining nodes in c. Second condition is node $w \in U(c)$ selects node u as uplink node as shown in figure 9.2.

Condition one gives the number of coordinator nodes selected for a given region and condition two reduces the size of selected nodes based on $U(c)$ selection. Based on Theorem 1 and rank of region, total number of coordinator nodes selected after the algorithm is $O(n) \cdot ORank(c)$. □

Theorem 3. In a network with n nodes randomly and uniformly deployed in a finite region C, $\lim_{n\to\infty} E(CDS_{ROOT-I}) = O(n') \cdot ORank(C) \cdot E(CDSopt)$. Here CDS_{ROOT-I} is size of CDS that ROOT-I algorithm generates.

We assume that the finite region C is divided into N equal square regions having width $d = \frac{r}{\sqrt{2}}$. Let $CDSopt$ be the optimal dominating set .

Here each square region is covered by the coordinator node in $CDSopt$. Considering there is no overlapping and redundant nodes in $CDSopt$. Each node in $CDSopt$ can cover $O(1)$ nonempty squares. Total number of squares covered by $CDSopt$ is given by the following equation.

$$N = O(1) \cdot E(CDSopt)$$

Based on Theorem 1, expected number of coordinator nodes for given square is $O(n')$. Assuming high node density, each square region contains at least one node. Considering Theorem 1 and Theorem 2, the expected number of coordinator nodes in ROOT-I algorithm is given by,

$$\lim_{n\to\infty} E(CDS_{ROOT-I}) = O(n') \cdot ORank(c) \cdot N$$

Considering above two results we can get our required result. □

9.2 Time and Message Complexity

ROOT-I $N_0(u)$ requires $N(u)$ and $N_2(u)$ information to select coordinator nodes. $N_0(u)$ collects the $N(u)$ and $N_2(u)$ information and based on $N_2(u)$ it selects the coordinator node from $N(u)$. $N_0(u)$ selects the $N(u)$ node based on its precedence which is decided by three tuple formation. Further this primary selection is further filtered by $N_2(u)$ uplink information. $N_2(u)$ selects the uplink node from all the available $N(u)$ nodes.

To do this $N_0(u)$ requires $N(u)$ steps to find out the one hop neighbors essential nodes. In next step $N_0(u)$ selects nodes from remaining $N(u)$ nodes to cover $N_2(u)$. Calculating essential nodes and then finding the remaining $N(u)$ nodes requires $2 \cdot N(u)$ steps. Considering Δ coordinator selected total time required for each step is $2 \cdot \Delta \cdot N(u)$. In last step $N_2(u)$ node selects the uplink node from the available Δ nodes. It requires $\Delta \cdot N_2(u)$ steps. Finally total time complexity of ROOT-I is $O(2 \cdot \Delta \cdot N(u) + \Delta \cdot N_2(u) + \Delta \cdot N(u) \cdot N(u))$. Here, $N(u) \cdot N(u)$ is the time required to sort one hop nodes based on its neighbor.

ROOT-I requires two hop information at $N_0(u)$. To collect that information $N(u)$ and $N_2(u)$ send HELLO messages. Using that $N_0(u)$ collects its required information. To do these ROOT-I uses $N(u)$ and $N_2(u)$ HELLO messages. After finding Δ coordinator nodes

$N_0(u)$ propagates these information to $N_2(u)$ using $\Delta + 1$ messages. In final step $N_2(u)$ nodes send uplink message. So the total message complexity of ROOT-I is $O(N(u) + 2 \cdot N_2(u) + \Delta + 1)$.

9.3 Implementation Details

Figure 9.4 shows architecture for our implementation. Here rule base contains rules for topology control mechanism. Application layer provides the information about fault tolerability of the network to CNI. MAC layer provides the channel detail and working role of the node. At initialization time application layer calculates the required fault tolerability and updates the CNI. Network layer updates the CNI by providing information about the neighbor table. It triggers the event in CNI. CNI propagates these events to cross layer engine. Cross layer engine gets the topology control rules from rule base.

Topology control rules contain the information about the working role of the node. Cross layer engine gets this information and generates the messages to set the working mode at MAC layer. It puts these messages into notification layer. Notification layer sends these messages to MAC layer. MAC at the end updates its parameter and working role. Here MAC layer itself does not decide its working role. It sets its working role based on the notification messages received from notification layer. Knowledge base and inference engine control the over all working. That makes the conventional layer mechanism untouched from the cross layer implementations.

9.4 Simulation Results

ROOT-I and ROOT-Ik algorithms are simulated using Network Simulator 2. Section 9.3 shows CLAPDAWN representation of proposed algorithm. We have compared our ROOT-I algorithm with MPR and MIS based method. We have tested these algorithms for different network conditions by varying number of nodes in the network, changing the deployment area and varying transmission range. Performance of these algorithms

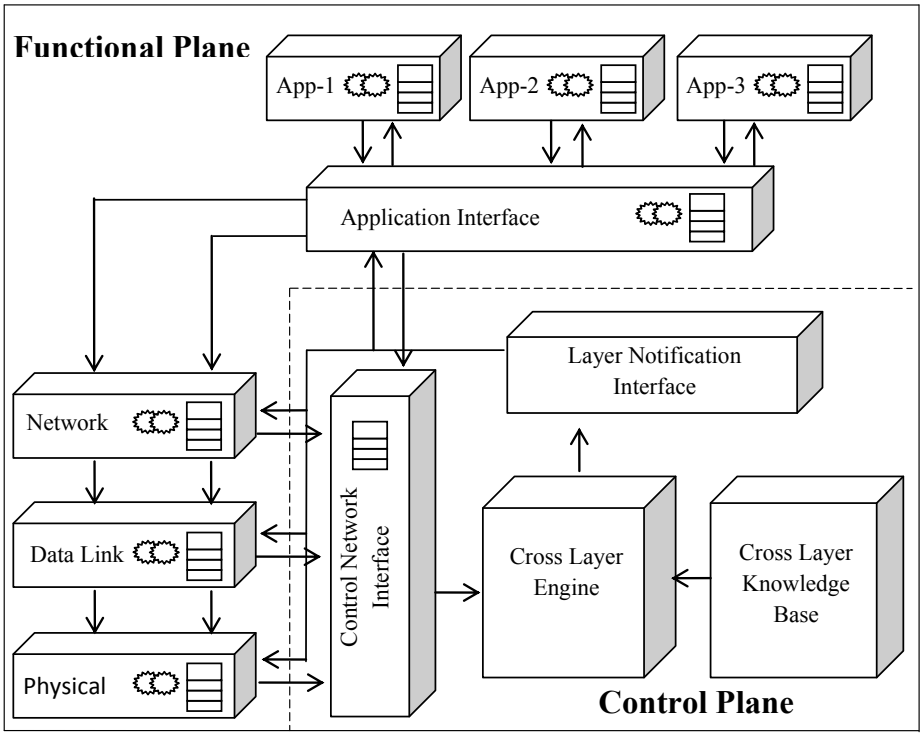


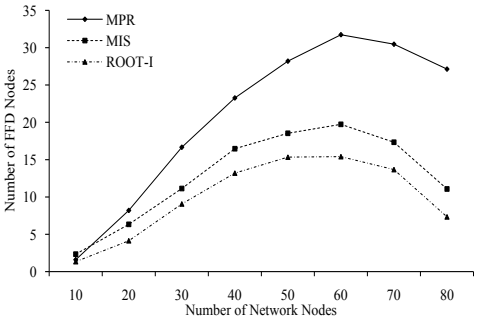
Figure 9.4: CLAPDAWN Architecture

is evaluated based on the number of FFDs, number of packets delivered and energy consumption. Following section covers simulation results showing comparative study of ROOT-I algorithm with multipoint relay based approach MPR and maximum independent set based approaches MIS.

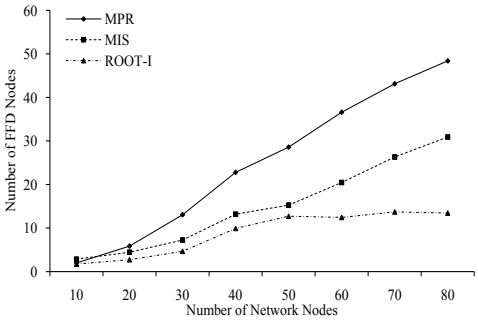
9.4.1 Connected dominating node size

First set of results is of connected dominating set size. Figure 9.5 shows the simulation results for connected dominating set. Here connected dominating set size shows number of nodes working as FFD. Simulation is carried out for two communication ranges a) 20 meter and b) 40 meters. Simulation also checks the effect of different network size on CDS size. Here, nodes are deployed randomly following uniform distribution in given field area. In result, X-axis represents the number of nodes in the network, and Y-axis represents the CDS size.

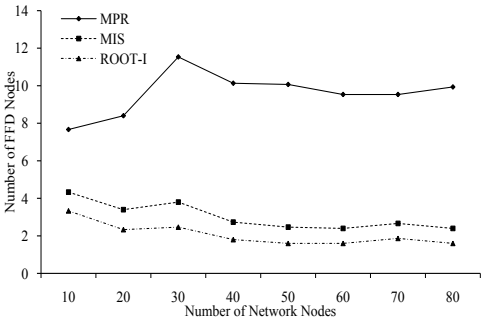
Figure 9.5(a) and 9.5(c) cover the simulation results for 20 meter transmission range.



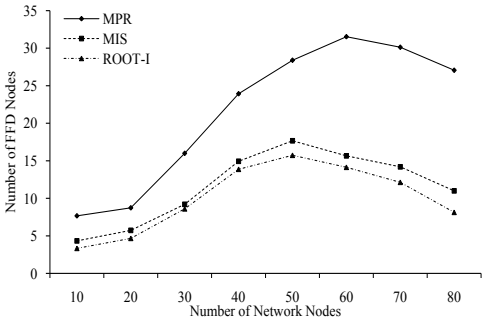
(a) With 20 meter Communication Range X area



(b) With 40 meter Communication Range X area



(c) With 20 meter Communication Range 2X area



(d) With 40 meter Communication Range 2X area

Figure 9.5: Result showing Connected Dominating Set Size

Y-axis represents number of nodes in CDS, size of CDS or number of nodes working as FFD nodes. Simulation results show that the multi-point relay has maximum number of nodes in CDS. Next, MIS has smaller number of nodes in CDS and ROOT-I has lower value of number of nodes working as FFD compare to other methods.

As ROOT-I removes the unwanted nodes in its decision making phase that helps in decreasing the size of coordinator nodes. Further, we have tested our scheme with increasing the field area that decreases the node density and increases the number of CDS nodes. Here, connectivity of sparse network remains same for all the cases. In the result we have ignored the isolated nodes for all the cases. Effect of it is visible in the results after 60 network nodes and with increased field area. Here reduced CDS size is not because of overlap or redundant nodes removal but its due to isolated nodes in the network. Results show that in sparse connectivity ROOT-I has better results than the other proposed algorithms.

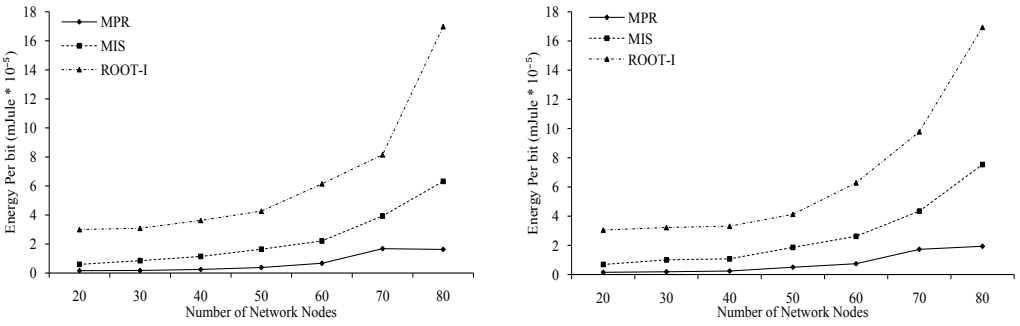
By increasing the field area, we have decreased the node density. In the next set of results we have tested the effect of increasing the node density. We increased the node's transmission range that increases the node's connectivity in the network. Figure 9.5(b) and 9.5(d) show the simulation results for 40 meter transmission range. Here also, the ROOT-I has smaller CDS than the other schemes.

Comparing these results with the 20 meter communication range shows that 40 meter communication range has higher number of neighboring nodes. Higher number of neighboring nodes also creates more overlap and that ends up having smaller CDS set. That matches with the analytical results. As $Of(c)$ increases, it decreases $ORank(C)$ and overall CDS size. In multi-point relay each node generates its cover only considering its two hop neighborhood. This process continues and that keeps on increasing the CDS size for MPR. In the case of MIS, network generates the independent set and that helps it to reduce the size of CDS node set. Improvement over greedy selection and overlapping nodes reduces the CDS size in ROOT-I.

9.4.2 Energy per bit

Next set of results show the total energy consumption that network encounters while collecting packets from the network nodes. Here, each node transmits packets to the base station and FFD node forwards those packets towards the base station. Here, energy consumption is calculated for all the packets that base station receives.

Simulation is carried out for different number of nodes. X-axis represents the number of nodes in the network and Y-axis represents the normalized energy consumption. This energy consumption is normalized to per bit energy consumption. That shows, how much energy, network has to invest to receive one bit of data. Energy per bit is calculated for two networks one with communication range set to 20 meters and other with communication range set to 40 meters. Similarly base station to node traffic flow can also be analyzed.



(a) With 20 meter Communication Range X area

(b) With 40 meter Communication Range 2X area

Figure 9.6: Overall Per Bit Energy Consumption

Figures 9.6(a) show the simulation results for the 20 meter communication range. Result shows that the ROOT-I has higher energy consumption than the MIS and MPR methods. In our previous results we have shown that the MIS and MPR have higher number of CDS nodes that generate many redundant packets in the network. This increases the total number of packets received at the base station. As the number of packets received increases it decreases the per bit energy consumption.

Next we have also checked the effect of increasing transmission range on per bit energy consumption in figure 9.6(b). It shows that as the communication range increases there

are higher chances of packet reception. This increases the packet delivery but increased transmission power increases the energy consumption per bit.

9.4.3 Energy per bit goodput

Next set of results is for per bit energy consumption for goodput. Here, by goodput we mean, number of packets delivered after removing the redundant packets. We have considered the broadcast network. Network nodes broadcast the packet and CDS nodes propagates these packets towards the base station.

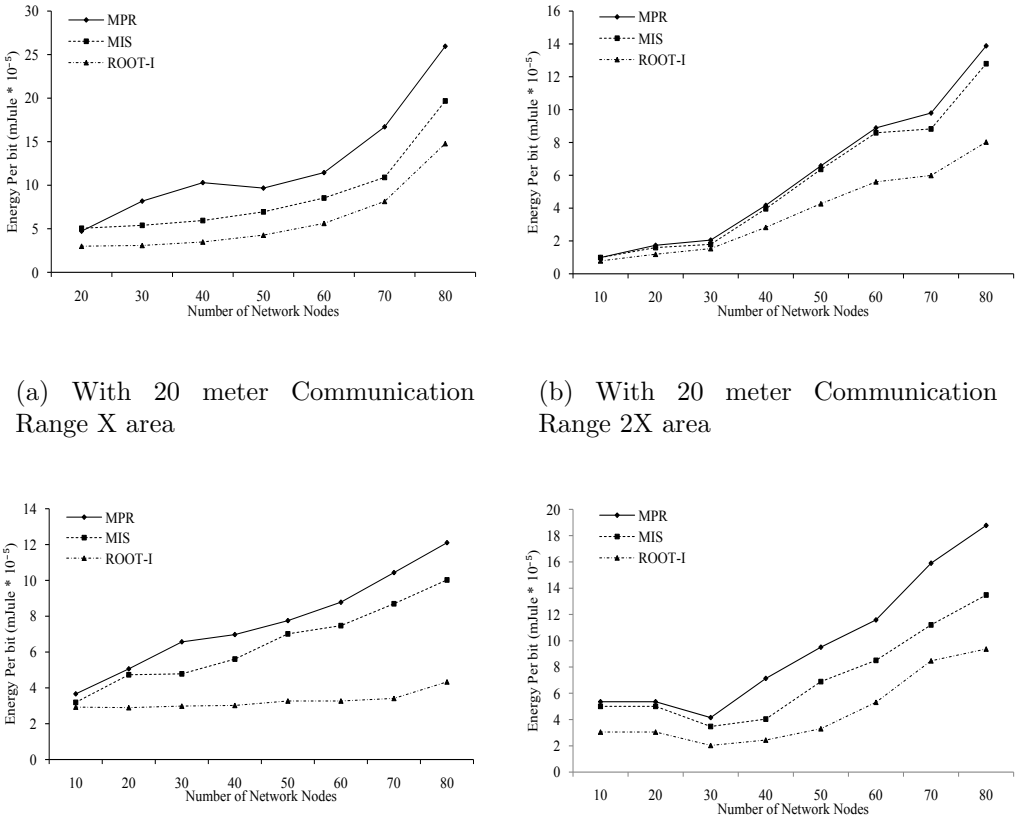
In previous result, we have considered the total received packets and per bit energy count for that. Figures 9.7 shows the simulation results for goodput. Here, results are in two sets, one is with network having communication range set to 20 meters and other is network having communication range set to 40 meters.

Figures 9.7(a) and 9.7(b) show goodput results for the network having communication range set to 20 meters. Here, X-axis represents the number of nodes in the network and Y-axis represents the normalized energy consumption per bit. Simulation results show that the MPR has the highest energy consumption compared to other algorithms.

In our previous results we saw that MPR has lesser total energy consumption than the other schemes and ROOT-I has higher energy consumption. There MRP delivers higher number of packets to the base station that decreases the per bit energy consumption. As the total received packets contain higher chunk of redundant packets, that gives the low per bit energy consumption.

Next, we have removed those redundant packets from the results to get goodput results. It shows that ROOT-I has lower energy consumption than the other algorithm. It suggests that as the total number of packets delivered is higher, the network has higher capacity. But due to redundant delivery, network is not utilized properly. That makes the network resources under utilized in other algorithms.

Figures 9.7(c) to 9.7(d) show the simulation results for the network having communication range set to 40 meters. As we saw in our previous result that higher communication range has higher packet delivery. That helps in improving the energy results and we have



(a) With 20 meter Communication Range X area

(b) With 20 meter Communication Range 2X area

(c) With 40 meter Communication Range X area

(d) With 40 meter Communication Range 2X area

Figure 9.7: Goodput Per Bit Energy Consumption

lesser energy cost than the previous results.

Here, higher communication range increases energy consumption but as seen in the previous result it also affects the number of FFD nodes. Reduced number of FFD reduces the redundant packets in the network. That decreases the energy consumption. Here increased transmission power increases the energy usage and reduced redundant packets reduces the energy consumption. Overall effect of this is visible in the result. Here MPR and MIS have higher increment in per bit energy consumption compared to 20 meter communication range. This increase in energy consumption is relatively low in ROOT-I based method due to reduced redundant packets.

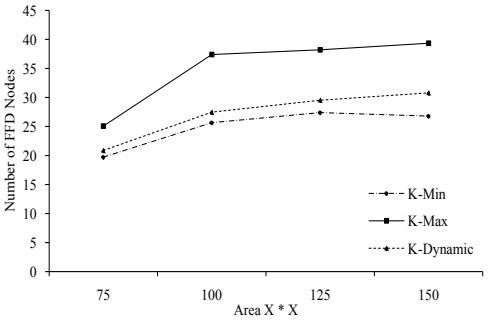
9.4.4 K-connectedness

ROOT-Ik is extension of ROOT-I and next set of results are for ROOT-Ik. Above set of results are for 1-connectedness which is special case of K-connected network with k set to one. There we saw that ROOT-I performs better than the compared algorithms. For ROOT-Ik, we performed simulations for changing the way the value of k is selected. One is by keeping the value of k fixed at all the layers as done in normal cases of k-connectedness. For that we considered two extremes k minimum and k maximum. In k minimum value of k is set to one and described it as K-Min in the results. For k maximum, value of k is set to minimum node degree of all the nodes and described it as K-Max in the results. We compared this two mechanisms with our dynamic policy in which value of k is configured based on network parameters. In the result it is shown as k-dynamic and it uses node proximity to configure the value of k as discussed in section 8.5.1.

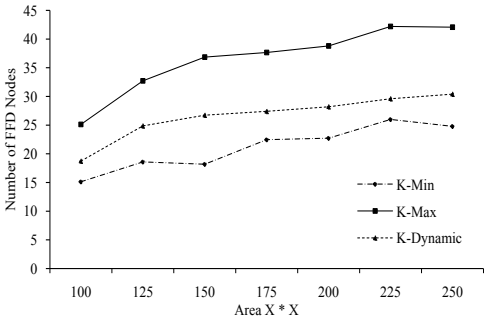
CDS size

Figure 9.4.4 shows the simulation results for the size of connected dominating set. Simulation is carried out for fixed number of network nodes that is 50 nodes. Algorithms are tested for different network scenarios like varying deployment area that changes network density. Here, X-axis represents the network deployment area and Y-axis represents the CDS size. In the results the isolated nodes are considered as FFD nodes and included into CDS size. Here, nodes having the same level are not selected as coordinator for other nodes of that level.

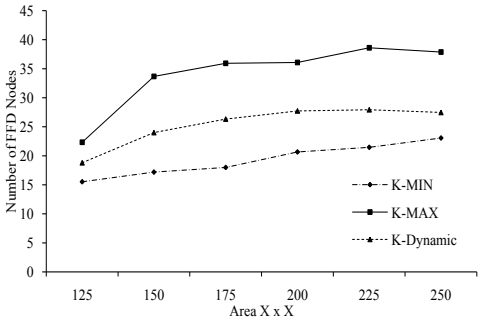
Results shows that K-Max has higher number of nodes in CDS compared to K-Dynamic and K-Min. K-Dynamic CDS has lesser number of nodes in CDS as it is not selecting same number of nodes at every level. And K-Dynamic has higher number of nodes in CDS as it selects more than minimum nodes as it moves towards the base station. Result shows that as network size increases network density decreases and more nodes contribute in CDS. As the network size increases network becomes sparse and more nodes become the isolated nodes.



(a) With 30 meter Communication Range



(b) With 40 meter Communication Range



(c) With 50 meter Communication Range

Figure 9.8: Result showing CDS size of 50 network nodes

In figure 9.8(a) as field size increases network becomes sparse and soon looses its connectivity. With this as the network size increases more and more nodes contributes in CDS. Next, as the communication range increases it increases the network connectivity. Figure 9.8(b) and 9.8(c) show results with better network connectivity, achieved by increasing the transmission range. We will see the effect of different values of k on network performance in next results.

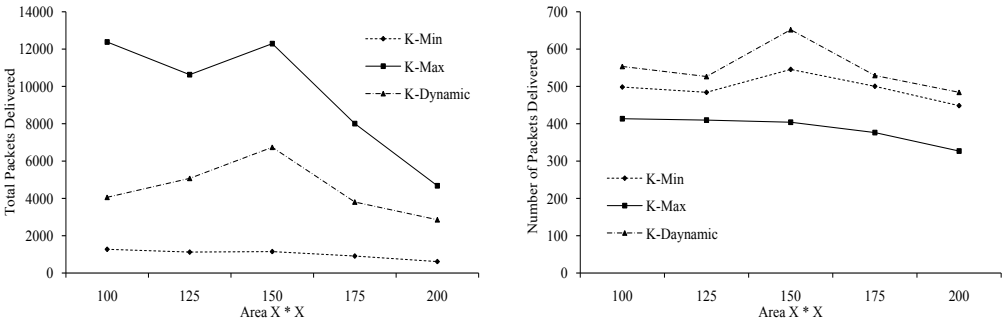
Packet received

Figure 9.9(a) covers the simulation result that gives analysis of received packets. Result shows the total received packet in the network. Results are for 50 nodes and for varying field area. Results show that K-Max receives higher number of total packets compared to other methods. As number of CDS nodes are high in K-Max, it increases the total packet delivery. In the case of K-Dynamic, it has less number of CDS nodes compared to K-Max and has lower number of total packet received compared to K-Max. Same is true with K-Min.

Results also show that as the network density decreases, it affects the connectivity that changes the packet delivery. Initially as the field area increases network gets higher chances of packet delivery because of less congestion and available connectivity. As the field area increases further, network looses its connectivity and that decreases the packet delivery. Because of this in the result we have first upward slope and then downward slope. Network achieves the maximum delivery at 150 sq.m. area.

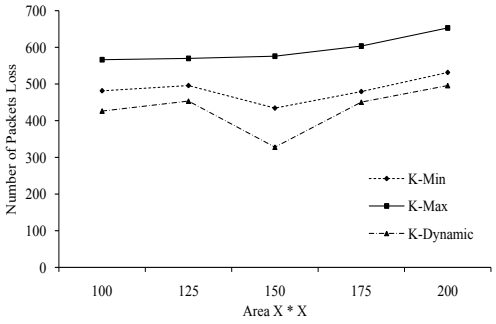
Next result shown in figure 9.9(b) talks about the actual packet delivery. As talked previously, total packets contain all the received packets which also includes the redundant packets. Result shown in figure 9.9(b) gives the packet delivery after removing redundant packets.

It shows that the network with higher number of CDS nodes deliver more packets but they also deliver more number of redundant packets. In the K-Max case network starting from leaf node starts replicating the packets. With k-Dynamic the network reduces that redundancy starting from leaf nodes. Reduced redundant packets increase chances for



(a) Total packets delivered

(b) Effective packets delivered



(c) Packet loss encountered

Figure 9.9: Result showing performance of network for 50 network nodes

other packets to get delivered to the base station.

Result shows that at 150 sq.m. area, network has higher packet delivery considering the effect of congestion and connectivity. Here K-Dynamic has higher actual packet delivery compared to other schemes.

Packet loss

In this section we discussed about the packet loss in the network. As the number of CDS nodes increases, network has higher chances of collision. Opposite to it with low connectivity network gets less chances of packet delivery and increases the packet loss. It requires a balanced mechanism which reduces the broadcast storm and also increases the delivery chance. Results show that k-Dynamic serves this purpose.

Figure 9.9(c) shows the simulation result covering packet loss in the network. Results shows that k-dynamic has lesser number of packet loss compared to other mechanism, while K-Max has higher packet loss. Normally k-connectivity is used to increase the packet delivery but results show that the higher number of CDS nodes create the congestion problem. With higher value of k, network performance degrades and reaches lower than the the one connectivity. But K-Dynamic selects the value of k based on network requirement and that controls the congestion. That increases the chances of packet delivery which improves the effective packet delivery and reduces the packet loss.

9.5 Summary

ROOT-I is multi-point relay based topology control algorithm, designed for network having single sink node, for example IEEE 802.15.4 based networks. Level assignment, ranking, cover selection and decision making are the main four phases of ROOT-I algorithm. Using $O(2 \cdot \Delta \cdot N(u) + \Delta \cdot N_2(u) + \Delta \cdot N(u) \cdot N(u))$ time complexity and $O(N(u) + 2 \cdot N_2(u) + \Delta + 1)$ message complexity ROOT-I solves the problem of greedy selection and multiple overlapping nodes, faced by majority of the greedy based MPR algorithms. ROOT-I has $O(n)$ approximation ratio and simulation results show the ef-

fect of overlapping factor $ORank(C)$ on CDS size. Simulation results show that due to ROOT-I topology control algorithm network achieves better performance in packet delivery and energy consumption. ROOT-I algorithm is designed to provide one connectivity to the nodes. For better fault tolerability ROOT-I algorithm is extended to ROOT-Ik algorithm. ROOT-Ik increases the connectivity of ROOT-I algorithm. ROOT-Ik dynamically selects the value of k based on spatial and temporal information. Dynamically selected value of k increases the network performance, simulation results support it.

Chapter 10

Future Work and Summary

10.1 Cross Layer Architecture for Protocol Design in A Wireless Network

To provide rapid development platform for cross layer implementation, we have proposed cross layer architecture "Cross Layer Architecture for Protocol Design in A Wireless Network (CLAPDAWN)". CLAPDAWN addresses the problems to which available cross layer architectures in the literature do not provide efficient solutions. These include the following problems: lack of support for multiple cross layer interactions, longer development time, accidental occurrence of feedback loops in the system and recovery from cross layer interactions.

CLAPDAWN simplifies the complex functioning of cross layer interactions into two parts: functional plane and control plane. Furthermore, to manage cross layer interactions, control plane is divided into four subcomponents a) Control Network Interface b) Execution Engine c) Knowledge base and d) Layer Notification Interface. This functional decoupling makes it easy to provide support for multiple cross layer interactions in the system.

We have implemented our architecture by extending Network Simulator 2 using POSIX sockets and threads. Here, class *Agent* is extended to provide necessary communication functionality. Majority of the protocols in NS-2 are derived from this classes so inherently

all the classes get their required support. Classes which are not derived from *Agent* class are extended by patching additional communication codes to them. This additional support works as interfaces using which various components in CLAPDAWN communicate with each other.

For comparative analysis we have also implemented ECLAIR using the same methodology. Adaptability and applicability of our architecture is verified by implementing complex cross layer video streaming protocol [10]. Furthermore, both the architectures are tested for odd CLI scenarios, like conflicting CLI and feedback loops.

CLAPDAWN manages to reduce development time by separating execution of cross layer interaction from its logic. Any modification that takes place in cross layer interaction logic remains transparent to the implementation of that interaction. Section 2.5.2 shows that compared to other architectures, it is easy to implement cross layer interactions in CLAPDAWN. Simulation study done in section 4.2.1 highlights that performance results achieved by implementing cross layer interaction in CLAPDAWN are as good as implemented in the layer.

CLAPDAWN implements cross layer logic as ECA rules in knowledge base. It helps in identifying the possible conflicts and feedback loops which may take place due to multiple cross layer interactions. CLAPDAWN identifies possible conflicts in the system and applies configurable policy mechanism over them to reduce their effect on the overall system. Simulation study done in section 4.2.2 shows that policy mechanism makes the system more stable compared to ECLAIR.

In our work we have verified the performance of CLAPDAWN through simulation. Our future objective is to port CLAPDWAN to operating system like Linux. Furthermore configuring ECA rules in knowledge base is key part of implementing cross layer interaction in CLAPDAWN. Currently, CLAPDAWN uses menu driven rule editor to configure ECA rules. Our objective is to make it more flexible with drag and drop features to configure rules in CLAPDAWN.

10.2 IPCAP and EPCAP Channel Access Protocols

Implicit and explicit prioritized channel access protocols are designed to help event driven wireless sensor networks implemented on IEEE 802.15.4 communication standards. IPCAP and EPCAP increase delivery chances of critical events in event driven application. IPCAP and EPCAP do not assume strict conditions like all the nodes being in each other's communication range.

IPCAP increases the delivery chances of critical events by modifying CSMA-CA configurations supported by IEEE 802.15.4. Same is modeled as two dimensional Markov chain. Unlike previous work, our modeling does not assume that both the Clear Channel Access have same priority. EPCAP uses secondary beacon concept to increase critical event delivery. Pre-configuration step helps here to overcome limitations faced by previous work. We have modeled EPCAP as M/G/c model and derived waiting time equation for each priority class.

Both the protocols are simulated using Qualnet Network Simulator and compared with IEEE 802.15.4 standard. Simulation result shown in chapter 7 highlights that both the protocols have higher delivery ratio for critical events compared to normal events. IPCAP achieves this by controlling the number of channel access tries and time between two accesses. In the case of EPCAP, it reduces the number of contending nodes to provide more chances to critical events.

In current implementation, EPCAP only defines the priority level in primary beacon. It can be further enhanced to provide better control over active duration of the FFD nodes. Currently we are working on the optimal assignment of superframe duration and beacon interval based on number of nodes in given priority class. IPCAP is designed to work with beacon enabled mode and can be extended to work with non-beacon enabled mode.

10.3 ROOT-I and ROOT-Ik Topology Control Mechanism

ROOT-Initiative (ROOT-I) and ROOT-Initiative k (ROOT-Ik) are multi-point relay based topology control algorithms. IEEE 802.15.4 comes with two types of device configurations: FFD and RFD nodes. ROOT-I and ROOT-Ik decide node type and configure the network topology as 1-connected and k-connected dominating sets respectively.

ROOT-I overcomes the problem of redundant and overlapping nodes faced by many of the previously proposed topology control algorithms. ROOT-I removes the unwanted nodes from the final solution using repetitive selection of essential nodes and reverse messaging as discussed in section 8.3. Here, network is modeled as unit disk graph. Using that we have derived approximation ratio for the ROOT-I algorithm in section 9.1.

To provide better fault tolerability 1-connectedness of ROOT-I is extended to k-connectedness in ROOT-Ik. Tradeoff between energy consumption and fault tolerability is adjusted by dynamically configuring the value of k based on node proximity discussed in section 8.5.1 and based on temporal traffic discussed in section 8.5.2.

ROOT-I and ROOT-Ik are implemented in Network Simulator 2. ROOT-I is compared with extended MRP and MIS based algorithms. Performance of ROOT-Ik is tested for various values of k in normal configuration and in dynamic configuration. Results discussed in chapter 9 show that due to reverse notification and repetitive essential node selection ROOT-I has better performance results than the other compared schemes.

ROOT-I and ROOT-Ik both the algorithms are initiated by the root node or the base station node, in IEEE 802.15.4 notation referred as PAN coordinator. In homogeneous wireless sensor networks selection of the PAN coordinator is a problem in itself called the leader election algorithm. Currently we are working on a leader election algorithm that decides that in a given random deployment of homogenous nodes which node will work as PAN coordinator and initiates the ROOT-I or ROOT-Ik algorithm.

10.4 Summary

Problem of cross layer architecture is not addressed adequately in the literature. In our work we have worked on cross layer architecture design and proposed "Cross Layer Architecture for Protocol Design in A Wireless Network (CLAPDAWN)". Using CLAPDAWN one can easily experiment with multiple cross layer interactions in the system. Further in our work we have designed solutions for prioritized event delivery problem as IPCAP and EPCAP, and for topology control problem as ROOT-I and ROOT-Ik algorithms.

Both the mechanisms: IPCAP and EPCAP are able to deliver more number of critical events to the base station than the standards. The IPCAP and EPCAP extend the IEEE 802.15.4 to work with event driven wireless sensor networks. ROOT-I and ROOT-Ik control the number of Full Function Devices in the networks. Dynamically configured value of k in k-connectedness provides better tradeoff between energy consumption and fault tolerability.

We have modeled these problems as cross layer protocols in CLAPDAWN, as discussed in section 7.2 and 9.3. In the above mentioned protocols decisions regarding the node priority (in IPCAP and EPCAP) and node type (FFD or RFD in ROOT-I and ROOT-Ik) are made in cross layer execution engine and not inside the conventional layers. Hence, these layers remain less affected and this provides rapid development and experimentation.

CLAPDAWN is implemented by extending Network Simulator 2 and compared with ECLAIR [3] for complex video streaming cross layer protocol [10] and multiple conflicting cross layer interactions. Result shows that CLAPDAWN outperforms the ECLAIR and provides better and stable platform for cross layer protocol development. With CLAPDAWN it is easy to add, remove or modify cross layer interactions and see the effect of it on the system. Cross layer designer can easily identify conflicts and loops among multiple cross layer interactions.

Appendix A

ECA Rule Editor

CLAPDAWN and ECLARI cross layer architectures implement cross layer interactions outside the network protocol stack. CLAPDAWN stores the cross layer interactions in its knowledge base in the form of ECA rules. ECLAIR implements cross layer interactions as optimizer. To make the development of cross layer interaction easy in both cases we have developed a rule editor. Current version of rule editor is menu driven text based editor. It generates two output files a) ".c" file which contains the ECLAIR implementation and ".kb" which contains the ECA rules for the CLAPDAWN implementation. Following screen shots show the working of our rule editor.

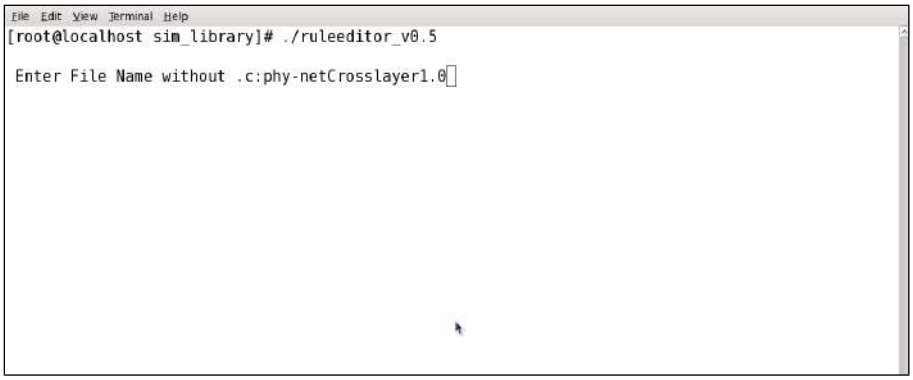


Figure A.1: Rule Editor with the file name for cross layer interaction to be stored in

Figure A.1 and A.2 show the rule editor screen. Here user gives the file name to which the generated cross layer interaction will be stored by the editor.

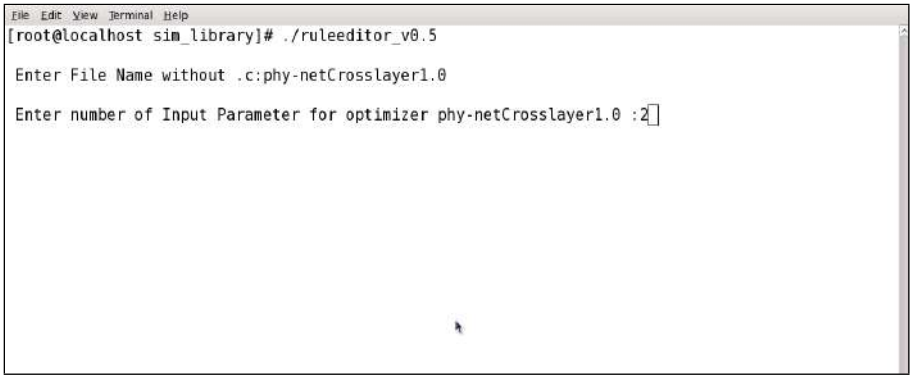


Figure A.2: Rule Editor asking for number of input parameters involved in cross layer interaction

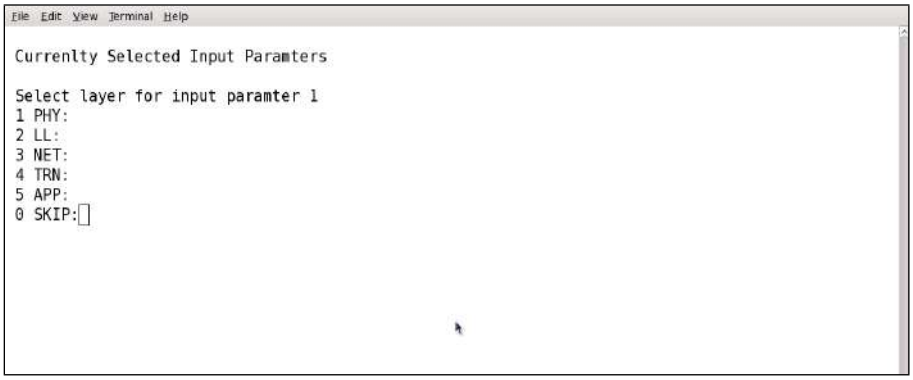


Figure A.3: Rule Editor asking for layer for first input parameter

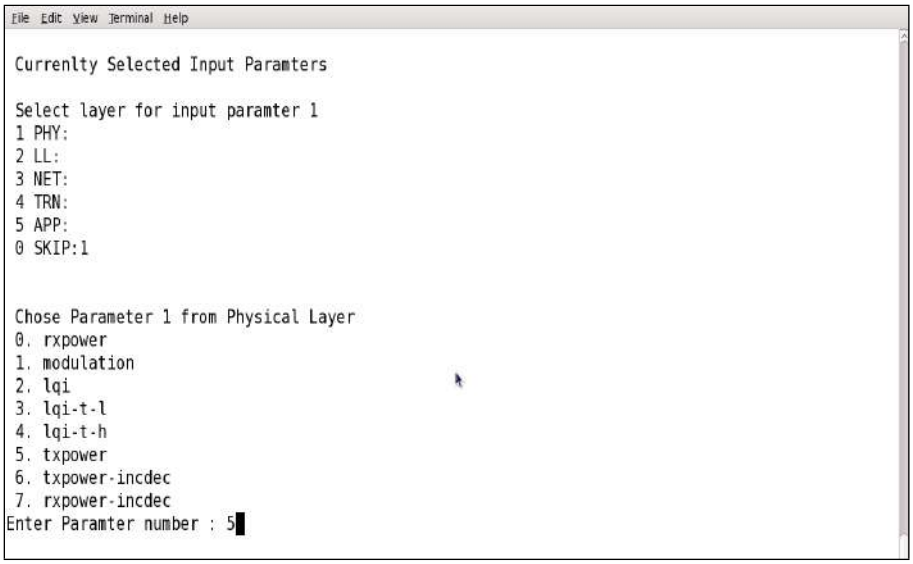


Figure A.4: Rule Editor, user entering first input parameter for physical layer

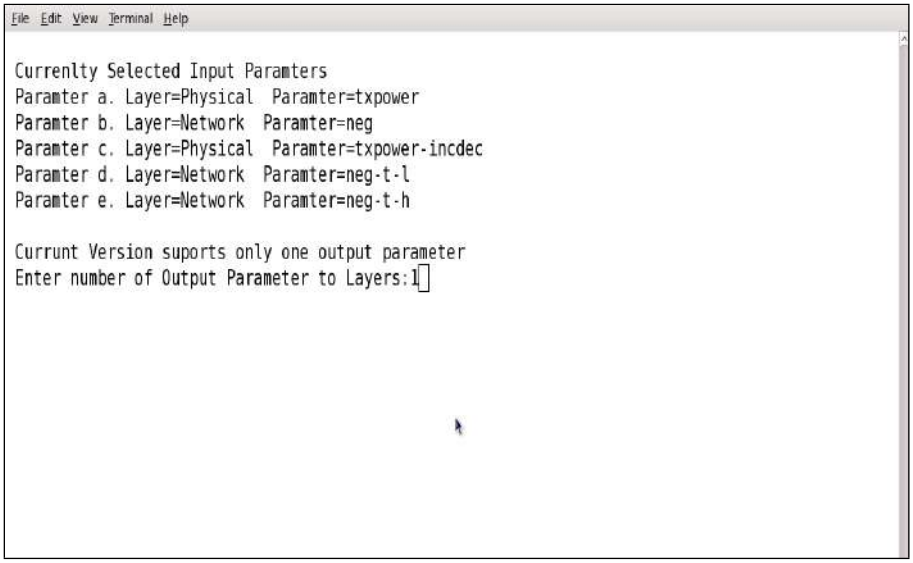


Figure A.5: Rule Editor asking for number of output parameters

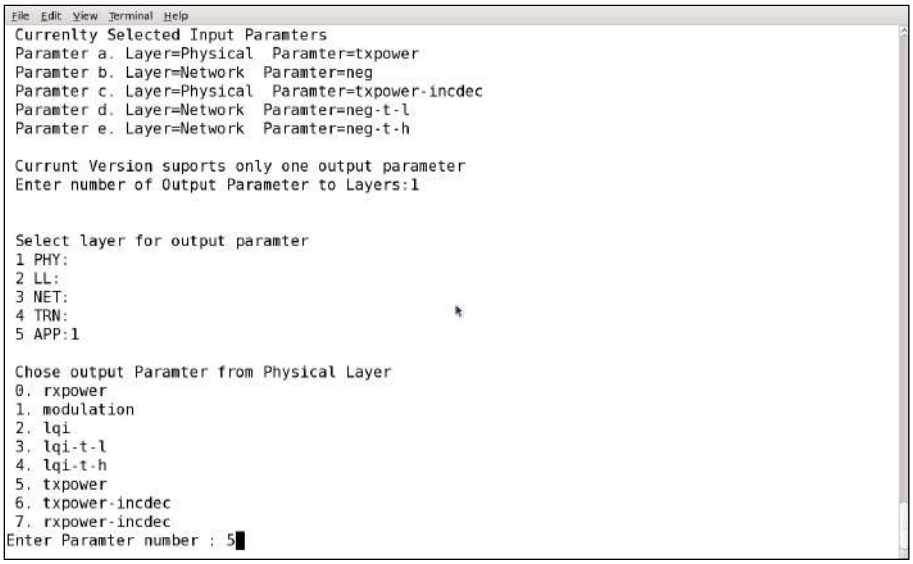


Figure A.6: Rule Editor, user selecting first output parameter

To generate it ask for the number of input parameters and output parameters. Figure A.3, A.4, A.5 and A.6 show the steps to configure input and output parameters. First it asks for the number of input parameters. Accepting number of input parameters, for each parameter it asks for layer name and parameter of that layer. This information is stored in parameter xml file. Same steps it repeats for output parameters.

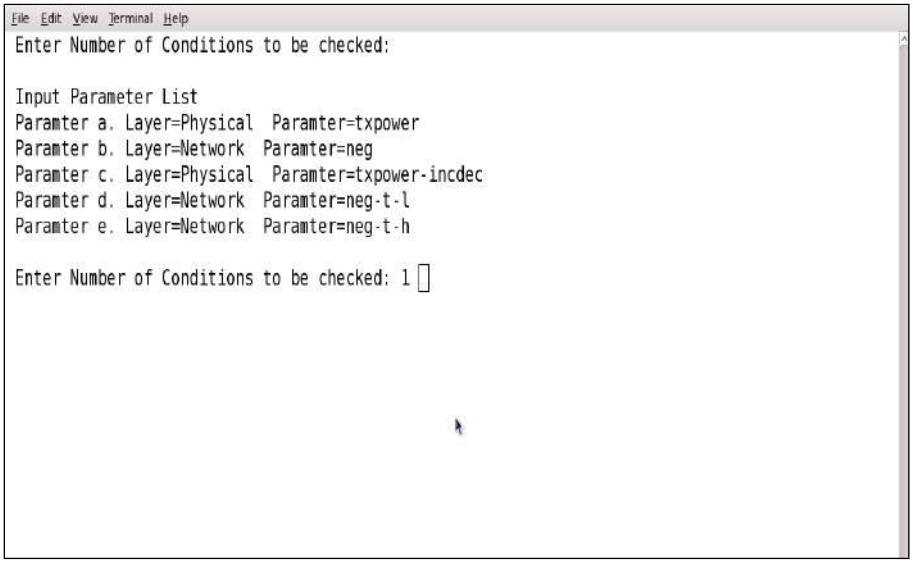


Figure A.7: Rule editor asking for number of conditions to be checked

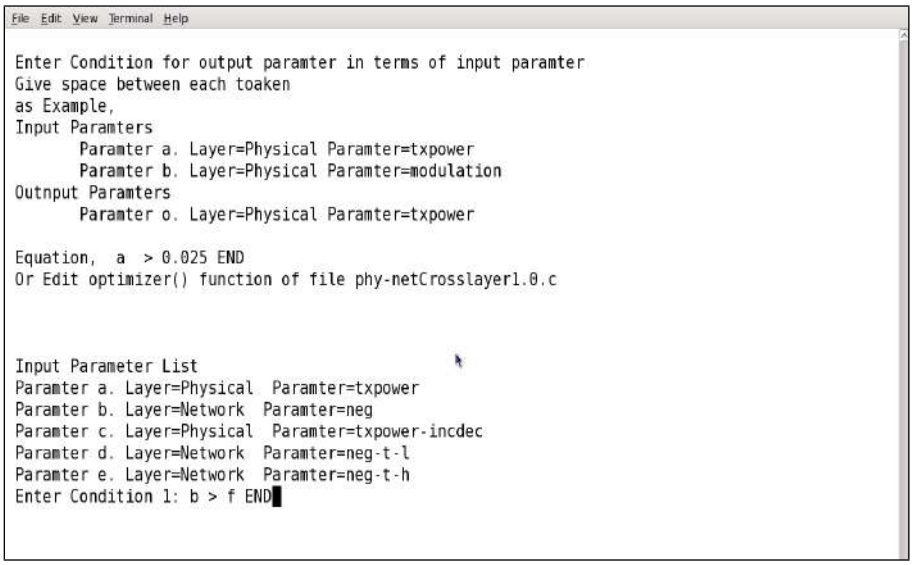


Figure A.8: Rule editor accepting first condition from user

After accepting the input and output parameters editor goes for the conditions. It first ask for the number of conditions required for this cross layer interaction and then it one by one accepts each conditions. Same steps are shown in figure A.7 and A.8.

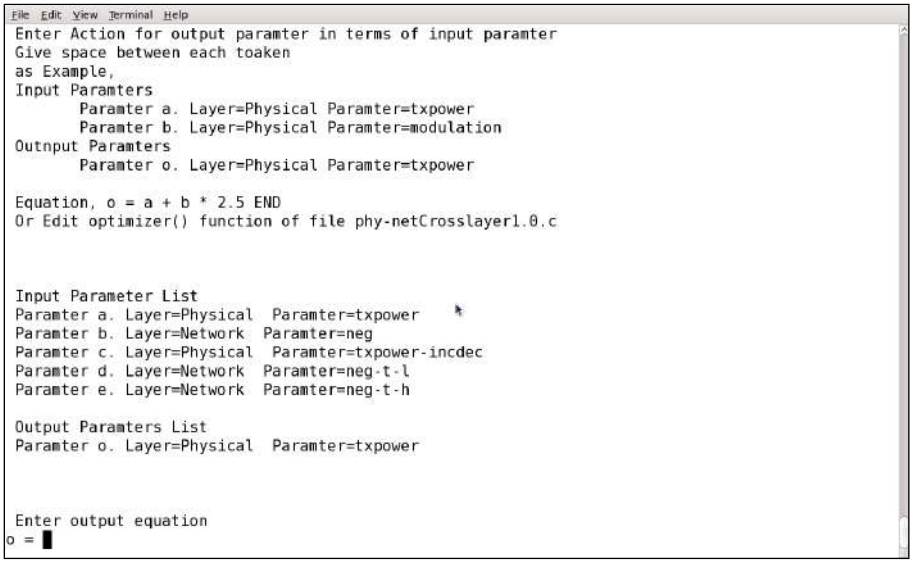


Figure A.9: Rule Editor accepting first action part from the user

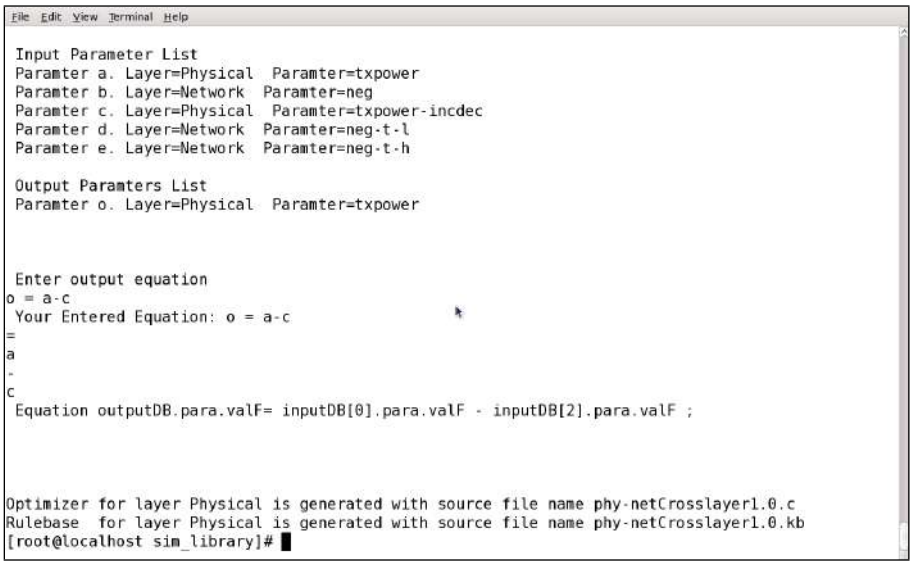


Figure A.10: Rule editor with action and generated output files

After collecting conditions it ask for the action part. In the current version action part only accepts one action. Remaining actions can be added by modifying ".c" and ".kb" files. Figure A.9 and A.10 show the final steps. Any further modification can be made by modifying the *optimization*() funciton of the generated file or the rules of the kb file. Sample optimizer file for ECLAIR and CLAPDAWN rulebase file is given in appendix B and C respectively.

Appendix B

ECLAIR Implementation of CLI

```
/* ECLAIR Implementation of cross layer interaction discussed in
   chapter 4. */

Optimizer( )
{
/*

    Code

*/

switch(frameType) {
    case 1:
       if(qlen1< threshold1)  {
           pri_i=1;
         done=1;
       }
       else if((qlen1 < threshold2))  {
            prob= rob0*(qlen1-threshold1)/(threshold2-threshold1);
            tmpx=Random();
            if(tmpx<prob)
                if((qlen2 < threshold2))
                      pri_i =2;
                else
                      pri_i =3;
            else
                pri_i = 1;
         done=1;
       }
       else if((qlen2 < threshold2))  {
```

```
            prob= prob0*(qlen2-threshold1)/(threshold2-threshold1);
            tmpx=Random();
            if(tmpx<prob)
               if((qlen3 < threshold2))
                       pri_i=3;
                  else
                       pri_i=2;
            else
                  pri_i=2;
            done=1;
   }
   else
   {
     tmpx=Random();
     if(tmpx<prob0)
         if((qlen3< threshold2))
             pri_i=3;
         else
             pri_i=2;
     else
         pri_i=2;
     done=1;
   }
break;

case 2:
     if((qlen1< threshold1))
     {
         pri_i=1;
         done=1;
     }
     else if((qlen1 < threshold2))
     {
         prob= prob1*(qlen1-threshold1)/(threshold2-threshold1);
         tmpx=Random();
         if(tmpx<prob)
             if((qlen2< threshold2))
                 pri_i=2;
             else
                 pri_i=3;
         else
         pri_i=1;
         done=1;
     }
     else if((qlen2< threshold2))
     {
         prob= prob1*(qlen2-threshold1)/(threshold2-threshold1);
```

```
            tmpx=Random();
            if(tmpx<prob)
                if((qlen3< threshold2))
                    pri_i=3;
                else
                    pri_i=2;
            else
            pri_i=2;
            done=1;
        }
        else
        {
            tmpx=Random();
            if(tmpx<prob1)
                if((qlen3< threshold2))
                    pri_i=3;
                else
                    pri_i=2;
            else
                pri_i=2;
            done=1;
        }
        break;
    case 3:
        if((qlen1< threshold1)){
            pri_i=1;
            done=1;
        }
        else if((qlen1< threshold2))
        {
            prob= prob2*(qlen1-threshold1)/(threshold2-threshold1);
            tmpx=Random();
            if(tmpx<prob)
                if((qlen2< threshold2))
                    pri_i=2;
                else
                    pri_i=3;
            else
                pri_i=1;
            done=1;
        }
        else if((qlen2< threshold2))
        {
            prob= prob2*(qlen2-threshold1)/(threshold2-threshold1);
            tmpx=Random();
            if(tmpx<prob)
                if((qlen3< threshold2))
```

```
                    pri_i=3;
                else
                    pri_i=2;
            }
            else
                pri_i=2;
            done=1;
        }
        else {
            tmpx=Random();
            if(tmpx<prob2) {
                if((qlen3< threshold2))
                    pri_i=3;
                else
                    pri_i=2;
            }
            else
                pri_i=2;
            done=1;
        }
        default:
            printf("Error in PriQ::pri_recv() frame type %d\n",frameType);
         break;
    }
/* Code */
}
```

Appendix C

CLAPDAWN Implementation of CLI

```
/* CLAPDAWN implementation of cross layer interaction discussed in chapter 4*/
RULE 1 PARA  LL frametype 2 1
RULE 1 CONDITION 1 LL frametype CO 1 e
RULE 1 CONDITION 2 LL qlen1 LL threshold1 l
RULE 1 ACTION 1 LL pri_i CO 1  CO 0 d  E

RULE 2 PARA  LL frametype 3 4
RULE 2 CONDITION 1 LL frametype CO 1 e
RULE 2 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 2 CONDITION 3 LL qlen1 LL threshold2 l
RULE 2 ACTION 1 LO l1 LL threshold2 LL threshold1 d  E
RULE 2 ACTION 2 LO l2 LL qlen1 LL threshold1 d  E
RULE 2 ACTION 3 LO l3 LL prob0 LO l2 m E
RULE 2 ACTION 4 LO prob LO l3 LO l2 d E

RULE 3 PARA  LL frametype 5 1
RULE 3 CONDITION 1 LL frametype CO 1 e
RULE 3 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 3 CONDITION 3 LL qlen1 LL threshold2 l
RULE 3 CONDITION 4 LL tmpx LO prob l
RULE 3 CONDITION 5 LL qlen2 LL threshold2 l
RULE 3 ACTION 1 LL pri_i CO 2 CO 0 d  E

RULE 4 PARA  LL frametype 5 1
RULE 4 CONDITION 1 LL frametype CO 1 e
RULE 4 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 4 CONDITION 3 LL qlen1 LL threshold2 l
```

```
RULE 4 CONDITION 4 LL tmpx LO prob l
RULE 4 CONDITION 5 LL qlen2 LL threshold2 ge
RULE 4 ACTION 1 LL pri_i CO 3 CO 0 d  E

RULE 5 PARA  LL frametype 4 1
RULE 5 CONDITION 1 LL frametype CO 1 e
RULE 5 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 5 CONDITION 3 LL qlen1 LL threshold2 l
RULE 5 CONDITION 4 LL tmpx LO prob ge
RULE 5 ACTION 1 LL pri_i CO 1 CO 0 d E

RULE 6 PARA  LL frametype 3 4
RULE 6 CONDITION 1 LL frametype CO 1 e
RULE 6 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 6 CONDITION 3 LL qlen2 LL threshold2 l
RULE 6 ACTION 1 LO l1 LL threshold2 LL threshold1 d  E
RULE 6 ACTION 2 LO l2 LL qlen2 LL threshold1 d  E
RULE 6 ACTION 3 LO l3 LL prob0 LO l2 m E
RULE 6 ACTION 4 LO prob LO l3 LO l2 d E

RULE 7 PARA  LL frametype 5 1
RULE 7 CONDITION 1 LL frametype CO 1 e
RULE 7 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 7 CONDITION 3 LL qlen2 LL threshold2 l
RULE 7 CONDITION 4 LL tmpx LO prob l
RULE 7 CONDITION 5 LL qlen3 LL threshodl2 l
RULE 7 ACTION 1 LL pri_i CO 3 CO 0 d  E

RULE 8 PARA  LL frametype 5 1
RULE 8 CONDITION 1 LL frametype CO 1 e
RULE 8 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 8 CONDITION 3 LL qlen2 LL threshold2 l
RULE 8 CONDITION 4 LL tmpx LO prob l
RULE 8 CONDITION 5 LL qlen3 LL threshodl2 ge
RULE 8 ACTION 1 LL pri_i CO 2 CO 0 d  E

RULE 9 PARA  LL frametype 5 1
RULE 9 CONDITION 1 LL frametype CO 1 e
RULE 9 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 9 CONDITION 3 LL qlen2 LL threshold2 l
RULE 9 CONDITION 4 LL tmpx LO prob l
RULE 9 CONDITION 5 LL qlen3 LL threshodl2 ge
RULE 9 ACTION 1 LL pri_i CO 2 CO 0 d E

RULE 10 PARA  LL frametype 3 1
RULE 10 CONDITION 1 LL frametype CO 1 e
RULE 10 CONDITION 2 LL qlen1 LL threshold2 ge
```

```
RULE 10 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 10 ACTION 1 LO prob LL prob0 CO 0 d  E

RULE 11 PARA  LL frametype 5 1
RULE 11 CONDITION 1 LL frametype CO 1 e
RULE 11 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 11 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 11 CONDITION 4 LL tmpx LO prob l
RULE 11 CONDITION 5 LL qlen3 LL threshodl2 l
RULE 11 ACTION 1 LL pri_i CO 3 CO 0 d E

RULE 12 PARA  LL frametype 5 1
RULE 12 CONDITION 1 LL frametype CO 1 e
RULE 12 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 12 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 12 CONDITION 4 LL tmpx LO prob l
RULE 12 CONDITION 5 LL qlen3 LL threshodl2 ge
RULE 12 ACTION 1 LL pri_i CO 2 CO 0 d E

RULE 13 PARA  LL frametype 4 1
RULE 13 CONDITION 1 LL frametype CO 1 e
RULE 13 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 13 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 13 CONDITION 4 LL tmpx LO prob ge
RULE 13 ACTION 1 LL pri_i CO 2 CO 0 d  E

RULE 14 PARA  LL frametype 2 1
RULE 14 CONDITION 1 LL frametype CO 2 e
RULE 14 CONDITION 2 LL qlen1 LL threshold1 l
RULE 14 ACTION 1 LL pri_i CO 1 CO 0 d  E

RULE 15 PARA  LL frametype 3 4
RULE 15 CONDITION 1 LL frametype CO 2 e
RULE 15 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 15 CONDITION 3 LL qlen1 LL threshold2 l
RULE 15 ACTION 1 LO l1 LL threshold2 LL threshold1 d  E
RULE 15 ACTION 2 LO l2 LL qlen1 LL threshold1 d  E
RULE 15 ACTION 3 LO l3 LL prob1 LO l2 m E
RULE 15 ACTION 4 LO prob LO l3 LO l2 d E

RULE 16 PARA  LL frametype 5 1
RULE 16 CONDITION 1 LL frametype CO 2 e
RULE 16 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 16 CONDITION 3 LL qlen1 LL threshold2 l
RULE 16 CONDITION 4 LL tmpx LO prob l
RULE 16 CONDITION 5 LL qlen2 LL threshold2 l
RULE 16 ACTION 1 LL pri_i CO 2 CO 0 d E
```

```
RULE 17 PARA  LL frametype 5 1
RULE 17 CONDITION 1 LL frametype CO 2 e
RULE 17 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 17 CONDITION 3 LL qlen1 LL threshold2 l
RULE 17 CONDITION 4 LL tmpx LO prob l
RULE 17 CONDITION 5 LL qlen2 LL threshold2 ge
RULE 17 ACTION 1 LL pri_i CO 3 CO 0 d  E

RULE 18 PARA  LL frametype 4 1
RULE 18 CONDITION 1 LL frametype CO 2 e
RULE 18 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 18 CONDITION 3 LL qlen1 LL threshold2 l
RULE 18 CONDITION 4 LL tmpx LO prob ge
RULE 18 ACTION 1 LL pri_i CO 1 CO 0 d  E

RULE 19 PARA  LL frametype 3 4
RULE 19 CONDITION 1 LL frametype CO 2 e
RULE 19 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 19 CONDITION 3 LL qlen2 LL threshold2 l
RULE 19 ACTION 1 LO l1 LL threshold2 LL threshold1 d  E
RULE 19 ACTION 2 LO l2 LL qlen2 LL threshold1 d  E
RULE 19 ACTION 3 LO l3 LL prob1 LO l2 m E
RULE 19 ACTION 4 LO prob LO l3 LO l2 d E

RULE 20 PARA  LL frametype 5 1
RULE 20 CONDITION 1 LL frametype CO 2 e
RULE 20 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 20 CONDITION 3 LL qlen2 LL threshold2 l
RULE 20 CONDITION 4 LL tmpx LO prob l
RULE 20 CONDITION 5 LL qlen3 LL threshodl2 l
RULE 20 ACTION 1 LL pri_i CO 3 CO 0 d  E

RULE 21 PARA  LL frametype 5 1
RULE 21 CONDITION 1 LL frametype CO 2 e
RULE 21 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 21 CONDITION 3 LL qlen2 LL threshold2 l
RULE 21 CONDITION 4 LL tmpx LO prob l
RULE 21 CONDITION 5 LL qlen3 LL threshodl2 ge
RULE 21 ACTION 1 LL pri_i CO 2 CO 0 d E

RULE 22 PARA  LL frametype 5 1
RULE 22 CONDITION 1 LL frametype CO 2 e
RULE 22 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 22 CONDITION 3 LL qlen2 LL threshold2 l
RULE 22 CONDITION 4 LL tmpx LO prob l
RULE 22 CONDITION 5 LL qlen3 LL threshodl2 ge
```

```
RULE 22 ACTION 1 LL pri_i CO 2 CO 0 d  E

RULE 23 PARA  LL frametype 3 1
RULE 23 CONDITION 1 LL frametype CO 2 e
RULE 23 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 23 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 23 ACTION 1 LO prob LL prob1 CO 0 d E

RULE 24 PARA  LL frametype 5 1
RULE 24 CONDITION 1 LL frametype CO 2 e
RULE 24 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 24 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 24 CONDITION 4 LL tmpx LO prob l
RULE 24 CONDITION 5 LL qlen3 LL threshodl2 l
RULE 24 ACTION 1 LL pri_i CO 3 CO 0 d  E

RULE 25 PARA  LL frametype 5 1
RULE 25 CONDITION 1 LL frametype CO 2 e
RULE 25 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 25 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 25 CONDITION 4 LL tmpx LO prob l
RULE 25 CONDITION 5 LL qlen3 LL threshodl2 ge
RULE 25 ACTION 1 LL pri_i CO 2 CO 0 d E

RULE 26 PARA  LL frametype 4 1
RULE 26 CONDITION 1 LL frametype CO 2 e
RULE 26 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 26 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 26 CONDITION 4 LL tmpx LO prob ge
RULE 26 ACTION 1 LL pri_i CO 2 CO 0 d E

RULE 27 PARA  LL frametype 2 1
RULE 27 CONDITION 1 LL frametype CO 3 e
RULE 27 CONDITION 2 LL qlen1 LL threshold1 l
RULE 27 ACTION 1 LL pri_i CO 1 CO 0 d E

RULE 28 PARA  LL frametype 3 4
RULE 28 CONDITION 1 LL frametype CO 3 e
RULE 28 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 28 CONDITION 3 LL qlen1 LL threshold2 l
RULE 28 ACTION 1 LO l1 LL threshold2 LL threshold1 d  E
RULE 28 ACTION 2 LO l2 LL qlen1 LL threshold1 d  E
RULE 28 ACTION 3 LO l3 LL prob2 LO l2 m E
RULE 28 ACTION 4 LO prob LO l3 LO l2 d E

RULE 29 PARA  LL frametype 5 1
RULE 29 CONDITION 1 LL frametype CO 3 e
```

```
RULE 29 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 29 CONDITION 3 LL qlen1 LL threshold2 l
RULE 29 CONDITION 4 LL tmpx LO prob l
RULE 29 CONDITION 5 LL qlen2 LL threshold2 l
RULE 29 ACTION 1 LL pri_i CO 2 CO 0 d  E

RULE 30 PARA  LL frametype 5 1
RULE 30 CONDITION 1 LL frametype CO 3 e
RULE 30 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 30 CONDITION 3 LL qlen1 LL threshold2 l
RULE 30 CONDITION 4 LL tmpx LO prob l
RULE 30 CONDITION 5 LL qlen2 LL threshold2 ge
RULE 30 ACTION 1 LL pri_i CO 3 CO 0 d  E

RULE 31 PARA  LL frametype 4 1
RULE 31 CONDITION 1 LL frametype CO 3 e
RULE 31 CONDITION 2 LL qlen1 LL threshold1 ge
RULE 31 CONDITION 3 LL qlen1 LL threshold2 l
RULE 31 CONDITION 4 LL tmpx LO prob ge
RULE 31 ACTION 1 LL pri_i CO 1 CO 0 d E

RULE 32 PARA  LL frametype 3 4
RULE 32 CONDITION 1 LL frametype CO 3 e
RULE 32 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 32 CONDITION 3 LL qlen2 LL threshold2 l
RULE 32 ACTION 1 LO l1 LL threshold2 LL threshold1 d  E
RULE 32 ACTION 2 LO l2 LL qlen2 LL threshold1 d  E
RULE 32 ACTION 3 LO l3 LL prob2 LO l2 m E
RULE 32 ACTION 4 LO prob LO l3 LO l2 d E

RULE 33 PARA  LL frametype 5 1
RULE 33 CONDITION 1 LL frametype CO 3 e
RULE 33 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 33 CONDITION 3 LL qlen2 LL threshold2 l
RULE 33 CONDITION 4 LL tmpx LO prob l
RULE 33 CONDITION 5 LL qlen3 LL threshodl2 l
RULE 33 ACTION 1 LL pri_i CO 3 CO 0 d E

RULE 34 PARA  LL frametype 5 1
RULE 34 CONDITION 1 LL frametype CO 3 e
RULE 34 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 34 CONDITION 3 LL qlen2 LL threshold2 l
RULE 34 CONDITION 4 LL tmpx LO prob l
RULE 34 CONDITION 5 LL qlen3 LL threshodl2 ge
RULE 34 ACTION 1 LL pri_i CO 2 CO 0 d E

RULE 35 PARA  LL frametype 5 1
```

```
RULE 35 CONDITION 1 LL frametype CO 3 e
RULE 35 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 35 CONDITION 3 LL qlen2 LL threshold2 l
RULE 35 CONDITION 4 LL tmpx LO prob l
RULE 35 CONDITION 5 LL qlen3 LL threshodl2 ge
RULE 35 ACTION 1 LL pri_i CO 2 CO 0 d E

RULE 36 PARA  LL frametype 3 1
RULE 36 CONDITION 1 LL frametype CO 3 e
RULE 36 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 36 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 36 ACTION 1 LO prob LL prob2 CO 0 d  E

RULE 37 PARA  LL frametype 5 1
RULE 37 CONDITION 1 LL frametype CO 3 e
RULE 37 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 37 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 37 CONDITION 4 LL tmpx LO prob l
RULE 37 CONDITION 5 LL qlen3 LL threshodl2 l
RULE 37 ACTION 1 LL pri_i CO 3 CO 0 d E

RULE 38 PARA  LL frametype 5 1
RULE 38 CONDITION 1 LL frametype CO 3 e
RULE 38 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 38 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 38 CONDITION 4 LL tmpx LO prob l
RULE 38 CONDITION 5 LL qlen3 LL threshodl2 ge
RULE 38 ACTION 1 LL pri_i CO 2 CO 0 d E

RULE 39 PARA  LL frametype 4 1
RULE 39 CONDITION 1 LL frametype CO 3 e
RULE 39 CONDITION 2 LL qlen1 LL threshold2 ge
RULE 39 CONDITION 3 LL qlen2 LL threshold2 ge
RULE 39 CONDITION 4 LL tmpx LO prob ge
RULE 39 ACTION 1 LL pri_i CO 2 CO 0 d E
```

Bibliography

[1] E. Borgia, M. Conti, and F. Delmastro, "Mobileman: design, integration, and experimentation of cross-layer mobile multihop ad hoc networks," *IEEE Communications Magazine*, vol. 44, no. 7, pp. 80–85, July 2006.

[2] R. Winter, J. H. Schiller, N. Nikaein, and C. Bonnet, "Crosstalk: cross-layer decision support based on global knowledge," *IEEE Communications Magazine*, vol. 44, no. 1, pp. 93–99, Jan 2006.

[3] V. T. Raisinghani and S. Iyer, "Cross-layer feedback architecture for mobile device protocol stacks," *IEEE Communications Magazine*, vol. 44, no. 1, pp. 85–92, January 2006.

[4] M. I. Tiado, M. I. Tiado, R. Dhaou, and A. l. Beylot, "Rcl: A new method for cross-layer network modeling and simulation," in *The 7th IFIP International Conference on Mobile and Wireless Communications Networks*, Marrakech, Morocco, September 2005.

[5] Y. D. Liang, "Dominations in trapezoid graphs," *Information Processing Letter*, vol. 52, no. 6, pp. 309–315, 1994.

[6] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1982.

[7] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "A simple heuristic for minimum connected dominating set in graphs," *International Journal of Foundations of Computer Science*, vol. 14, no. 2, pp. 323–333, 2003.

[8] I. Stojmenovic, M. Seddigh, and J. Zunic, "Internal nodes based broadcasting in wireless networks," in *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 9*. Washington, DC, USA: IEEE Computer Society, 2001, p. 9005.

[9] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9*. Washington, DC, USA: IEEE Computer Society, 2002, p. 298.

[10] C.-K. Shieh, C.-H. Ke, N. K. Chilamkurti, and S. Zeadallys, "An adaptive cross-layer mapping algorithm for mpeg-4 video transmission over ieee 802.11e wlan," *Telecommunication System, Special Issues on Mobility Management and Wireless Access*, vol. 42, no. 3, pp. 223–234, December 2009.

[11] V. Srivastava and M. Motani, "Cross-layer design: A survey and the road ahead," *IEEE Communications Magazine*, vol. 43, no. 12, pp. 112– 119, December 2005.

[12] R. Jurdak, *Wireless Ad Hoc and Sensor Networks: A Cross-Layer Design Perspective.*, R. Jurdak, Ed. Springer-Verlag: Signals and Communication Technology, 2007.

[13] V. Gauthier, M. Marot, and M. Becker, *Recent Adavances In Modeling and Simulation Tools for Communicaiton networks and services.* Springer, 2007, ch. Cross-Layer Interaction in Wireless Ad Hoc Networks: A Practical Example, pp. 105–120.

[14] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452 – 1463, August 2006.

[15] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross layer," *IEEE Wireless Communications Magazine*, vol. 12, no. 1, pp. 3–11, February 2005.

[16] V. Raghunathan, C. Schurgers, S. Park, M. Srivastava, and B. Shaw, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 40–50, March 2002.

[17] A. Cerpa, J. Elson, M. Hamilton, J. Zhao, D. Estrin, and L. Girod, "Habitat monitoring: application driver for wireless communications technology," in *SIGCOMM LA '01: Workshop on Data communication in Latin America and the Caribbean.* New York, NY, USA: ACM, 2001, pp. 20–41.

[18] P. He, J. Li, and W. Dong, "A novel distributed topology control algorithm for ad hoc networks," in *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops.* Washington, DC, USA: IEEE Computer Society, 2007, pp. 653–658.

[19] L. Song and D. Hatzinakos, "A cross-layer architecture of wireless sensor networks for target tracking," *IEEE/ACM Transaction on networking*, vol. 15, no. 1, pp. 145–158, 2007.

[20] G. Jakllari, S. V. Krishnamurthy, M. Faloutsos, P. V. Krishnamurthy, and O. Ercetin, "A cross-layer framework for exploiting virtual miso links in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 579–594, 2007.

[21] B. Hamdaoui and P. Ramanathan, "A cross-layer admission control framework for wireless ad-hoc networks using multiple antennas," *IEEE Transactions on Wireless Communications*, vol. 6, no. 11, pp. 4014–4024, November 2007.

[22] J. Salido, L. Lazos, and R. Poovendran, "Energy and bandwidth-efficient key distribution in wireless ad hoc networks: a cross-layer approach," *IEEE/ACM Transaction on networking*, vol. 15, no. 6, pp. 1527–1540, 2007.

[23] Y. Yu and G. B. Giannakis, "Cross-layer congestion and contention control for wireless ad hoc networks," *IEEE/ACM Transaction on networking*, vol. 7, no. 1, pp. 37–42, January 2008.

[24] C. Chun-Yuan, W. E. Hsiao-Kuang, and C. Gen-Huey, "A reliable and efficient mac layer broadcast protocol for mobile ad hoc networks," *IEEE transactions on vehicular technology*, vol. 56, no. 2, pp. 2296–2305, 2007.

[25] P. J. Marrn, D. Minder, A. Lachenmann, and K. Rothermels, "Tinycubus: An adaptive cross-layer framework for sensor networks," *it - Information Technology*, vol. 47, no. 2, pp. 87–97, 2005.

[26] V. T. Raisinghani and S. Iyer, "Eclair: An efficient cross layer architecture for wireless protocol stacks," in *World Wireless Congress*, San Francisco, USA, May 25-28 2004.

[27] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-layer design for wireless networks," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 74–80, Octomber 2003.

[28] Z. Chang, G. Gaydadjiev, and S. Vassiliadis, "Infrastructure for cross-layer designs interaction," in *Proceedings of 16th International Conference on Computer Communications and Networks, ICCCN 2007.* Honolulu, HI: IEEE, August 2007, pp. 19–25.

[29] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, and E. Brewer, *Ambient Intelliegence.* Springer, 2005, ch. TinyOS: An Operating System for Sensor Networks, pp. 114–148.

[30] [Online]. Available: http://www.tinyos.net/

[31] S.-H. Choi, D. E. Perry, and S. M. Nettles, "A software architecture for cross-layer wireless network adaptations," in *Seventh Working IEEE/IFIP Conference on Software Architecture, 2008. WICSA.* Vancouver, BC: IEEE, February 2008, pp. 281–284.

[32] W. Su and T. L. Lim, "Cross-layer design and optimization forwireless sensor networks," in *SNPD-SAWN '06: Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 278–284.

[33] S.-H. Choi, D. E. Perry, and S. M. Nettles, "A software architecture for cross-layer wireless network adaptations," in *Seventh Working IEEE/IFIP Conference on Software Architecture, WICSA 2008*, Febuary 2008, pp. 18–21.

[34] S. Mohapatra, N. Dutt, A. Nnicolau, and N. Venkatasubramanian, "Dynamo: A cross-layer framework for end-to-end qos and energy optimization in mobile handheld devices," *IEEE Journal of Selected Areas in Communication*, vol. 25, no. 4, pp. 722–737, May 2007.

[35] K. Loukil, N. B. Amor, and M. Abid, "Self adaptive reconfigurable system based on middleware cross layer adaptation model," in *6th International Multi-Conference on Systems, Signals and Devices, SSD '09.* IEEE, March 2009, pp. 1–9.

[36] M. Khalid, R. Sankar, and I. ho Ra, "Towards a cross layer framework for ad hoc wireless networks," in *14th IEEE Symposium on Computers and Communications (ISCC 2009)*. Sousse, Tunisia: IEEE, 2009, pp. 159–164.

[37] *IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Standard, Wireless LAN standard*, IEEE standard association Std., August 1999.

[38] *IEEE 802.16 Broadband Wireless Metropolitan Area Network (WirelessMAN) Medium Access Control (MAC) and Physical Layer (PHY) Specification, Standard,*, IEEE standard association Std., 2004.

[39] *IEEE 802.15.1, Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs(tm)), Standard,*, IEEE standard association Std.

[40] *IEEE 802.15.4, Part 15.4 Wireless Medum Access Control (MAC) and Physical Layer (PHY) Specification for Low-Rate Wireless Personal Area Networks (LR-WPANs), Standard,*, Std., 2006.

[41] *IEEE 802.15.4, Part 15.4 Wireless Medum Access Control (MAC) and Physical Layer (PHY) Specification for Low-Rate Wireless Personal Area Networks (LR-WPANs), Standard,*, IEEE Std., 2006.

[42] *IEEE 802.15.4, Part 15.4 Wireless Medum Access Control (MAC) and Physical Layer (PHY) Specification for Low-Rate Wireless Personal Area Networks (LR-WPANs), Standard,*, IEEE Std., 2004.

[43] I. Demirkol, C. Ersoy, and F. Alagz, "Mac protocols for wireless sensor networks: a survey," *IEEE Communication Magazine*, vol. 44, no. 4, pp. 115 – 121, 2006.

[44] R. Pries, D. Staehle, S. Oechsner, M. Menth, S. Menth, and P. Tran-Gia, "On the unfair channel access phenomenon in wireless lans," in *21st International Teletraffic Congress, 2009. ITC 21.* Paris: IEEE, September 2009, pp. 1–8.

[45] M. Durvy, O. Dousse, and P. Thiran, "Self-organization properties of csma/ca systems and their consequences on fairness," *IEEE Transactions on Information Theory*, vol. 55, no. 3, pp. 931 – 943, March 2009.

[46] J. Zheng and M. J. Lee, "Will ieee 802.15.4 make ubiquitous networking a reality?: A discussion on a potential low power, low bit rate standard," *IEEE Communication Magazine*, vol. 42, no. 6, pp. 140–146, June 2004.

[47] ——, *Sensor Network Operations*. IEEE Press, Wiley Interscience, 2006, ch. A comprehensive performance study of IEEE 802.15.4, pp. 218–237.

[48] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene, *Design, Automation, and Test in Europe*. Springer Netherlands, 2008, ch. Energy Efficiency of the IEEE 802.15.4 Standard in Dense Wireless Microsensor Networks: Modeling and Improvement Perspectives, pp. 221–234.

[49] G. Lu, B. Krisnamachari, and S. Cauligi, "Performance evaluation of the ieee 802.15.4 mac for low-rate low-power wireless networks," in *International Performance Computing and Communications Conf. (IPCCC'04)*, Phoenix, AZ, April 2004, pp. 701–706.

[50] G. Bianchi, "Performance analysis of the ieee 802.11 distributed coordinationfunction," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, 2000.

[51] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 3, pp. 493–506, 2004.

[52] T. R. Park, T. H. Kim, J. Y. Choi, S. Choi, and W. H. Kwon, "Throughput and energy consumption analysis of ieee 802.15.4 slotted csma/ca," *Electronics Letters*, vol. 41, no. 18, pp. 1017– 1019, 2005.

[53] J. Misic and V. Misic, "Access delay and throughput for uplink transmissions in ieee 802.15.4 pan," *Elsevier Computer Communications Journal*, vol. 28, no. 10, pp. 1152–1166, 2005.

[54] J. Misic, S. Shafi, and V. B. Misic, "The impact of mac parameters on the performance of 802.15.4 pan," *Elsevier Ad hoc Networks Journal*, vol. 3, no. 5, p. 509528, 2005.

[55] S. Pollin, M. Ergen, S. C. Ergen, B. Bougard, L. V. der Perre, F. Catthoor, I. Moerman, A. Bahai, and P. Varaiya, "Performance analysis of slotted carrier sense ieee 802.15.4 medium access layer," in *IEEE Global Telecommunications Conference, 2006. GLOBECOM '06.*, December 2006, pp. 1–6.

[56] C. Buratti, "A mathematical model for performance of ieee 802.15.4 beacon-enabled mode," in *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*. New York, NY, USA: ACM, 2009, pp. 1184–1190.

[57] I. Ramachandran, A. K. Das, and S. Roy, "Analysis of the contention access period of ieee 802.15.4 mac," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 1, p. 4, 2007.

[58] X. Ling, Y. Cheng, J. W. Mark, and X. Shen, "A renewal theory based analytical model for the contention access period of ieee 802.15.4 mac," *IEEE Transactions on Wireless Communications*, vol. 7, no. 6, pp. 2340–2349, 2008.

[59] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene, "Energy efficiency of the ieee 802.15.4 standard in dense wireless microsensor networks: Modeling and improvement perspectives," in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1*, ser. DATE '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 196–201. [Online]. Available: http://dx.doi.org/10.1109/DATE.2005.136

[60] D. Kipnis, A. Willig, J.-H. Hauer, and N. Karowski, "The angel ieee 802.15.4 enhancement layer: Coupling priority queueing and service differentiation," in *14th European Wireless Conference, EW 2008*, June 2008, pp. 1–7.

[61] B. C. Villaverde, S. Rea, and D. Pesch, "D-sedgam: A dynamic service differentiation based gts allocation mechanism for ieee 802.15.4 wsn," in *Seventh International Conference on Information Technology: New Generations (ITNG)*, Las Vegas, NV, April 2010, pp. 852 – 857.

[62] A. Koubaa, M. Alves, B. Nefzi, and Y.-Q. Song, "Improving the ieee 802.15.4 slotted csma/ca mac for time-critical events in wireless sensor networks," in *In Proc. of the Workshop of Real-Time Networks (RTN 2006) Satellite Workshop to (ECRTS 2006)*, 2006.

[63] T. H. Kim and S. Choi, "Priority-based delay mitigation for event-monitoring ieee 802.15.4 lr-wpans," *IEEE Communications Letters*, vol. 10, no. 3, pp. 213–215, March 2006.

[64] E.-J. Kima, M. Kimb, S.-K. Youma, S. Choia, and C.-H. Kang, "Priority-based service differentiation scheme for ieee 802.15.4 sensor networks," *International Journal of Electronics and Communications*, vol. 61, no. 2, pp. 69–81, February 2007.

[65] A. Federgruen and H. Groenevelt, "M/g/c queueing systems with multiple customer classes: characterization and control of achievable performance under nonpreemptive priority rules," *Manage. Sci.*, vol. 34, pp. 1121–1138, September 1988. [Online]. Available: http://portal.acm.org/citation.cfm?id=53137.53144

[66] L. Kleinrock, *Queueing Systems. Vol. 2.* John Wiley, 1976.

[67] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking.* New York, NY, USA: ACM, 1999, pp. 151–162.

[68] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *DIALM '99: Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications.* New York, NY, USA: ACM, 1999, pp. 7–14.

[69] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

[70] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *Computer*, vol. 37, no. 2, pp. 40–46, 2004.

[71] A. Ephremides, J. Wieselthier, and D. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," *Proceedings of the IEEE*, 1987.

[72] J. Ma, M. Gao, Q. Zhang, and L. M. Ni, "Energy-efficient localized topology control algorithms in ieee 802.15.4-based sensor networks," *IEEE Transactions On Parallel and Distributed Systems*, vol. 18, no. 5, pp. 711–720, 2007.

[73] D. Li, H. Du, and W. Chen, "Power conservation for strongly connected topology control in wireless sensor network," in *FOWANC '08: Proceeding of the 1st ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing*. New York, NY, USA: ACM, 2008, pp. 9–16.

[74] A. H. Shuaib and A. H. Aghvami, "Dynamic topology control for the ieee 802.15.4 network," *International Journal of Sensor Networks*, vol. 6, no. 3/4, pp. 212–223, 2009.

[75] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," *Mobile Networks and Applications - Discrete algorithms and methods for mobile computing and communications*, vol. 9, no. 2, pp. 141–149, 2004.

[76] Y. Zou and K. Chakrabarty, "A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 978–991, 2005.

[77] S. Funke, A. Kesselman, U. Meyer, and M. Segal, "A simple improved distributed algorithm for minimum cds in unit disk graphs," *ACM Trans. Sen. Netw.*, vol. 2, no. 3, pp. 444–453, 2006.

[78] C. Adjih, P. Jacquet, and L. Viennot, "Computing connected dominated sets with multipoint relays," *Ad Hoc & Sensor Wireless Networks*, vol. 1, no. 1-2, 2005.

[79] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.

[80] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *ICC (1)*, 1997, pp. 376–380.

[81] K. Mohammed, L. Gewali, and V. Muthukumar, "Generating quality dominating sets for sensor network," in *ICCIMA '05: Proceedings of the Sixth International Conference on Computational Intelligence and Multimedia Applications*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 204–211.

[82] J. Wu, "Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 9, pp. 866–881, 2002.

[83] F. Dai and J. Wu, "An extended localized algorithm for connected dominating set formation in ad hoc wireless networks," *IEEE Transactions On Parallel Distributed Systems*, vol. 15, no. 10, pp. 908–920, 2004.

Index