Stony Brook University

Architecture and Protocols for a High-Performance, Secure IEEE 802.11-based Wireless Mesh Network

A Dissertation Presented

by

Ashish Raniwala

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2009

Stony Brook University

The Graduate School

Ashish Raniwala

We, the dissertation committee for the above candidate for
the Doctor of Philosophy degree,
hereby recommend acceptance of this dissertation.

Professor Tzi-cker Chiueh, Dissertation Advisor
Department of Computer Science

Professor Samir Das, Chairperson of Defense
Department of Computer Science

Professor Jie Gao
Department of Computer Science

Professor Xin Wang
Department of Electrical and Computer Engineering

This dissertation is accepted by the Graduate School.

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Architecture and Protocols for a High-Performance, Secure IEEE 802.11-based Wireless Mesh Network

by
Ashish Raniwala

Doctor of Philosophy
in
Computer Science

Stony Brook University

2009

Today's wireless LANs reside only on the last hop between the end users' desktop/laptop machines and the enterprise backbone network. A comprehensive wired backbone still needs to be deployed to inter-connect these access points and the enterprise computing resources. In this project, we architected a novel wireless mesh backbone network architecture (called Hyacinth) that can eliminate most, if not all, of this wiring overhead. In a wireless mesh network (WMN), close-by access points communicate with each other using direct wireless links, while distant access points communicate using multiple wireless hops. In this dissertation, we formulate the capacity, fairness, and security issues with Hyacinth architecture and devise novel solutions to them. Our proposed architecture has three major components: Multi-channel Mesh Networking, Stateful Transport Protocol, and Secure Routing.

Limited capacity remains a pressing issue even for single-hop wireless LANs, let alone a multi-hop WMN where inter-path and intra-path interference limit the number of links that can be simultaneously active in the network. Fortunately, the IEEE 802.11b/g standards and IEEE 802.11a standard provide 3 and 12-25 non-overlapped frequency channels, respectively, which could be used simultaneously within a neighborhood. Hyacinth employs multiple radio channels in each radio neighborhood by equipping each node with multiple network interfaces. To fully utilize the performance potential of this approach, Hyacinth provides two traffic load-aware channel assignment and routing algorithms, both of which tune the network channel assignment and routing based on the network topology and the latest traffic patterns. Even with the use of just 2 radio interfaces per node, the proposed algorithms improve the network cross-section goodput by factors of up to 7 when compared with single-interface single-channel WMNs.

The next key issue with WMNs is lack of an effective transport protocol that can fairly and efficiently allocate the limited network capacity among multiple flows sharing the network. While many transport protocols have been proposed specifically for multi-hop wireless networks, most of them refrain from keeping state in the intermediate network nodes.

We study the research question of how much performance improvement is possible if intermediate network nodes could maintain as much state as is needed. In particular, we investigate how a stateful transport protocol can accurately measure the effective physical link capacity, and fairly and efficiently allocate this capacity by estimating the number of sharing flows and their individual sending rates. Additionally, we examine how leveraging the link-layer retransmission mechanism can improve the performance of reliable packet delivery. While the proposed mechanisms improve the fairness and utilization of transport flows on a WMN, they fail to address the hidden node problem that causes one wireless link's transmission to be inhibited by another link, eventually leading to unequal bandwidth allocation between the two. To address this problem, we further propose a global bandwidth allocation algorithm that can provide end-to-end flow-level max-min fairness despite weaknesses in the MAC layer.

The final concern of enterprise users about WLAN technology is its security. In the case of a WMN, the security requirement is even more stringent, because even a single compromised node has the potential of making the entire network unavailable. A compromised node can easily disrupt the network routing state by tampering with control communication or advertising crafted topology/traffic data. We develop a centralized network architecture that incorporates security as a first-class requirement at par with connectivity and performance. The architecture and its associated protocol secure all core operations in a mesh network – topology and traffic statistics collection, route and channel computation, data plane state distribution, network reconfiguration, and also packet forwarding. It can quickly detect most common misbehaviors and trace the problem down to specific nodes. The secure routing mechanisms significantly enhance the availability of a Hyacinth network when some of the WMN nodes are compromised, misconfigured, or broken.

To
Papa, Rahul,
Qi and Sophie

Contents

List of Tables

List of Figures

Acknowledgments

First and foremost, I would like to express my gratitude towards my advisor Tzi-cker Chiueh. From him, I learned how to conduct research that matters. He instilled in me the importance of critical/independent thinking and open discussions in coming up with great problems to work on, and that of building and evaluating systems in truly understanding and solving those problems. He gave me the freedom to explore and learn what interested me, the opportunities to contribute and learn from other related projects, and the guidance and encouragement at all the right times. Most of all, he served, and continues to serve, as an inspiration for me.

I also want to thank my dissertation committee members, Samir Das, Jie Gao, and Xin Wang, for their valuable feedback and guidance. A very special thank to Michael Bender who has always been there as a colleague and a friend to me, and from whom I learned so much about problem solving, presentations, collaboration, and much more.

Many thanks to my colleagues at Google who supported me in completing my dissertation. Most notably, I want to thank Craig Chambers, whose support during the last entire year has been crucial in helping me reach this point. He gave me lots of flexibility in my work hours, and also gave me the necessary 'push'. I also want to thank Theo Vassilakis, Charlie Garrett, and Frances Perry, for their ceaseless prodding and support.

Several thanks to my co-researchers at ECSL – Kartik Gopalan, Srikant Sharma, Pradipta De, and Rupa Krishnan – for all the great work that we did together, and for keeping high spirits working through all those paper deadlines and nite outs. Thanks to Ningning Zhu, Fanglu Guo, Manish Prasad, Gang Peng, Maohua Lu, Jatan Modi, Krishna Tatavarthi, and others, in making ECSL such a fun and wonderful place to work at.

I also owe a lot to my senior colleagues at ECSL – Prashant Pradhan, Srinidhi Varadarajan, Lan Huang, Anindya Neogi, Tulika Mitra, and Chuan-Kai Yang – who guided me through the foundational years of my doctoral study.

Thanks to my parents and my brother Rahul, for everything they have done and continue to do for me. Without their encouragement and support, this dissertation would not have been possible.

Last, but not the least, I want to thank my wife Qi, and my daughter Sophie, for their unwavering love and support. Qi has made countless sacrifices to let me pursue my doctorate studies – traveling alone, taking care of Sophie, staying a continent apart from me for several years, being a great listener, supporting me despite all odds and delays, and much more. A very special thank to little Sophie, who got very little of my time so far.

Chapter 1

Introduction

1.1 Motivation

Today's wireless LANs reside only on the last hop between end users' desktop/laptop machines and the enterprise backbone network. The backbone network that inter-connects these access points and the enterprise computing resources is predominantly based on wired Ethernet. The wiring requirement not only complicates deployment by limiting the potential locations where access points could be placed, but also adds to the post-deployment management overheads [2, 3]. While building architects already accommodate this need by pre-cabling the new buildings with Ethernet cables, pulling a cable to each access point may not be feasible or desirable in older architectures or in outdoor campus-wide wireless network deployments.

An attractive alternative to wired backbone is a *wireless mesh backbone network* [4, 5]. Based on the ad hoc networking paradigm, a wireless mesh network (WMN) allows one to inter-connect access points using wireless links. While closeby access points communicate with each other using direct wireless links, distant access points communicate using multiple wireless hops. The multi-hop wireless backbone network thus formed enables each access point to communicate with every other access point as well as to access the enterprise computing resources. The WMN architecture can thus eliminate most, if not all, of the wiring overhead associated with access point deployment.

Wireless mesh networks are also gaining significant momentum as an inexpensive solution to provide last-mile connectivity to the Internet [6, 7, 8]. Here, some of the nodes are provided with wired connectivity to the Internet, while the rest of the nodes access the Internet through these wire-connected nodes by forming a multi-hop wireless mesh network with them. As deployment and maintenance of wired infrastructure is a major cost component in providing ubiquitous high-speed wireless Internet access [7], use of mesh network on the last-hop brings down the overall ISP costs. For similar reasons, wireless mesh network can be an attractive alternative even to *wired* broadband technologies such as DSL/cable modem.

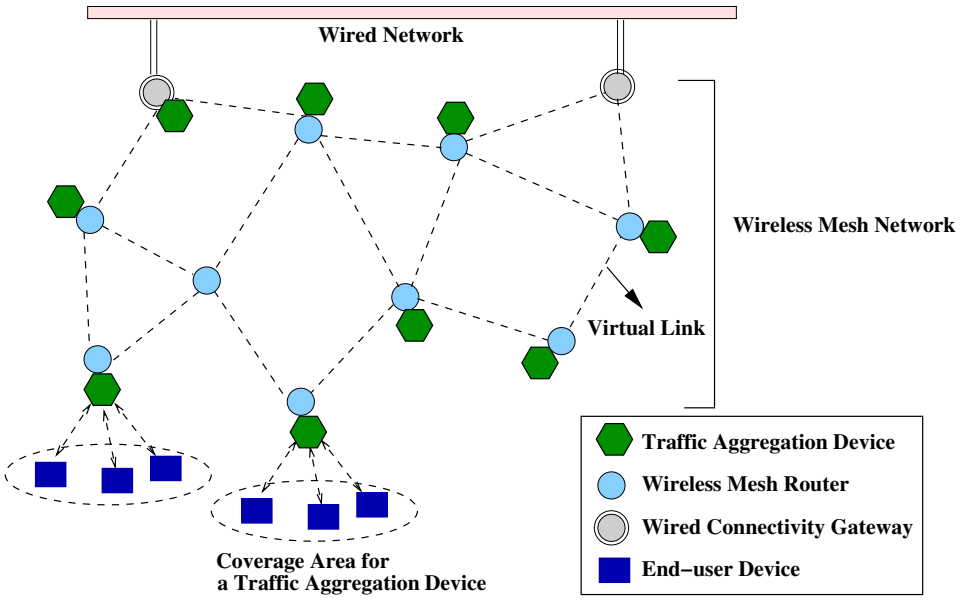


Figure 1.1: *A wireless mesh network (WMN) core, which is connected to a wired network through a set of wired connectivity gateways. Each WMN node has a radio interface that it uses to communicate with other WMN nodes over wireless links as shown. A WMN node is optionally equipped with a traffic aggregation device (similar to an 802.11 access point) that interacts with individual mobile stations. The WMN relays mobile stations' aggregated data traffic to/from the wired network.*

1.2 Wireless Mesh Network

As shown in Fig 1.1, a wireless mesh network (WMN) consists of fixed wireless routers, each of which is optionally equipped with a traffic aggregation access point that provides network connectivity to end-user mobile stations within its coverage area. In turn, the wireless routers form a multi-hop ad hoc network among themselves to relay the traffic to and from mobile stations. Some of the WMN nodes serve as gateways between the WMN and a wired network. Most of the infrastructure resources such as file servers, Internet gateways and application servers, reside on the wired network and can be accessed through any of the gateways. In the most general case, the physical links between gateways and the wired network can be a wired link, or a point-to-point 802.11 or 802.16 wireless link.

Each node in a WMN is equipped with an 802.11-compliant NIC. For direct communication, two nodes need to be within *communication range* of each other. A pair of nodes that are within *interference range* may interfere with each other's communication, even if they cannot directly communicate. The "virtual links" shown between the nodes depict direct communication between them. The interaction between mobile nodes and a traffic aggregation device is similar to the infrastructure mode operation of the IEEE 802.11 standards.

Although a wireless mesh network (WMN) is similar in concept to a mobile ad hoc

network (MANET), there are some important differences between the two:

- Firstly, nodes in a WMN are fixed, *i.e.* not mobile. Topology changes are therefore infrequent, and occur only due to occasional node failures, node shut-down for maintenance, or addition of new nodes.
- Secondly, the traffic characteristics, being aggregated from a large number of traffic flows, do not change very frequently, permitting optimization of network based on measured traffic profiles.
- Thirdly, the traffic distribution in a WMN is typically skewed, as most of the user traffic is directed to/from the gateway nodes that are connected to the wired network. This happens because users typically want to access resources on the Internet or on the enterprise servers, and both of them most likely reside on the wired infrastructure [9]. This further results in a more *predictable traffic pattern* enabling use of a traffic engineering approach.
- Finally, to serve as an effective backbone, a WMN requires *proactive* discovery of paths to reduce packet delays. In contrast, in most mobile ad hoc networks, reactive routing strategies are a normal as additional packet latency due to on-demand route discovery is acceptable and maximization of battery life is more important.

1.3 Research Issues

Substantial research has been conducted in providing basic connectivity in multi-hop WMNs. However, before the vision of a fully wireless enterprise network can be realized, several other WMN issues need to be solved. The goal of this dissertation is to formulate these issues and devise practical solutions to them. Specifically, we focus our attention on three aspects of WMNs: capacity, fairness, and security. We now discuss each of these issues briefly.

1.3.1 Limited Capacity

Despite many advances in wireless physical-layer technologies, limited capacity remains a pressing issue even for single-hop wireless LANs. The advertised 54 Mbps bandwidth for IEEE 802.11a/g based hardware is the peak *link-level* data rate. When all the overheads – MAC contention, 802.11 headers, 802.11 ACK and packet errors – are accounted for, the actual goodput available to applications is almost halved. In addition, the maximum link-layer data rate falls quickly with increasing distance between the transmitter and the receiver. The bandwidth issue is even more severe for multi-hop wireless mesh networks for the following reasons:

- *Inter-flow and Intra-flow Interference:* In order to keep the network connected, all nodes in a wireless mesh network typically operate over over the same radio channel.

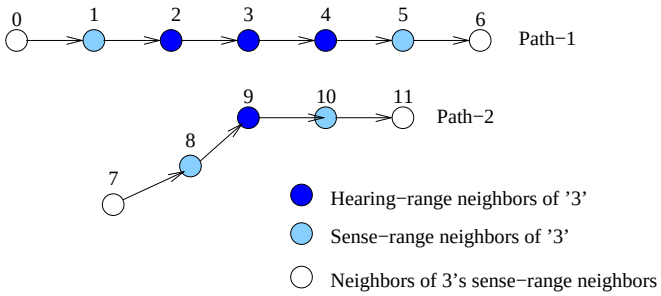


Figure 1.2: *Intra-path and Inter-path interference in a single-channel multi-hop ad hoc network. Nodes 1, 2, 4, 5 are in the interference range of Node 3, and hence can not transmit/receive when Node 3 is active. Nodes 8, 9, and 10 belonging to another node-disjoint path also fall in the interference range of Node 3. Thus, none of the wireless links shown in the figure can simultaneously operate when Node 3 is transmitting to Node 4.*

This results in substantial interference between transmissions from adjacent nodes on the same path as well as neighboring paths reducing the end-to-end capacity of the network [10, 11]. Figure 1.2 depicts an example of such interference.

- *Ineffective Route Selection:* The most popular routing metric for wireless mesh networks is the hop-count metric. However, using the hop-count metric leads to sub-optimal path selection. First, small hop-count translates into longer, and hence more error-prone, individual hops [12]. Second, use of minimum hop-count does not do anything to load-balance the traffic across the network. This reduces the effective capacity of the wireless mesh network.

- *TCP's Control Overhead:* TCP, the *de facto* transport protocol of the Internet, further compounds this problem by failing to effectively utilize the available bandwidth. Firstly, TCP's reliance on ACK clocking requires it to send an acknowledgment every packet or every other packet. These end-to-end acknowledgments consume substantial network bandwidth (upto 20%) because of high fixed per-packet overhead in 802.11 wireless networks. Secondly, when a packet is lost on an intermediate hop, TCP's end-to-end strategy requires the retransmission to traverse the entire path all over again. This leads to wastage of bandwidth on all preceding hops where the prior transmissions of the very packet being retransmitted were successful. Table 1.1 estimates the overhead imposed by TCP ACKs by comparing TCP throughput with optimal UDP throughput. The table was obtained by conducting experiments on a one-hop IEEE 802.11a-based network. In Appendix A, we derive the analytical equations for the same, and the last two columns in Table 1.1 use those equations.

- *Ineffective Congestion Control:* TCP's congestion control relies on packet drops to detect network congestion. In wireless networks, however, packets are also dropped because of bit errors. A TCP sender fails to distinguish between these frequent bit errors and true congestion, and inadvertently reduces its sending rate even in case

Link-Rate (Mbps)	TCP Thruput (Mbps)	Optimal Thruput (Mbps)	ACK Overhead %	Predicted Overhead w/ DelACK	Predicted Overhead w/o DelAck
6	4.6	5.3	13.2	6.3	12.6
12	8.4	10.3	18.4	9.3	18.6
18	12.0	14.9	19.4	11.6	23.2
24	16.5	19.3	14.5	14.1	28.2
36	22.6	26.8	15.7	17.9	35.8
48	26.9	32.9	18.2	21.0	42.0

Table 1.1: *Relative overhead of MAC contention, PLCP header, and Link-layer ACK increases as more sophisticated link-layer encoding is used. This substantially increases the relative overhead of TCP ACKs. Beyond 18 Mbps, TCP's DelACK mechanism kicks in, and hence the TCP's ACK overhead becomes closer to the predicted overhead with DelACK.*

Channel Condition	TCP Thruput (Mbps)	Optimal UDP Thruput (Mbps)	TCP Underutilization (%)
Very bad	.08	.87	90.8
Bad	3.37	6.07	44.5
Average	14.5	18.6	22.0
Very Good	26.9	32.9	18.2

Table 1.2: *TCP's congestion control performance degrades significantly in case of channel error-induced packet drops. This is because TCP's congestion control misinterprets channel error-induced packet drops as sign of network congestion and slows down the sender. The channel conditions were controlled by changing the transmission power of the NICs.*

of bit errors. Depending on the channel error conditions, this phenomenon can lead to substantial underutilization of the network [13]. Table 1.2 shows the difference between bandwidth achieved by a TCP flow and an optimal flow running on an IEEE 802.11a-based one-hop network under various channel conditions.

1.3.2 Flow Unfairness

The second class of problems associated with wireless mesh networks, specifically those based on IEEE 802.11 standards, can be categorized as fairness problems. Here we discuss some of the representative ones:

- *Hidden Terminal Problem:* It is well-known that IEEE 802.11 MAC layer exhibits the *hidden node problem* [14] that causes one wireless link's transmission to be inhibited by another link. While the RTS/CTS messages in 802.11's MAC protocol effectively stop a hidden node from interfering with an on-going communication transaction,

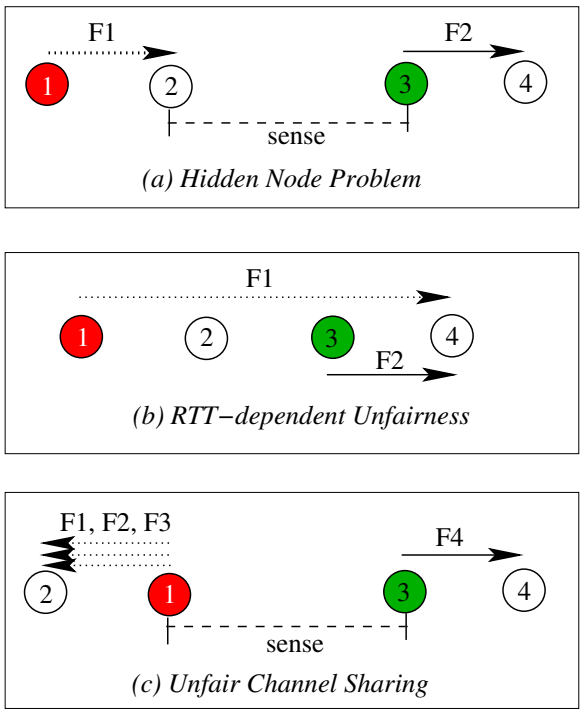


Figure 1.3: *Three scenarios in which significant unfairness among flows arises. The wireless node getting a lesser than fair share of bandwidth is marked as '1' (and colored in red or white), whereas the one getting a larger share is marked as '3' (and colored in green or black): (a) Node 1 lacks informations about Node 3's transmissions, attempts its communication at inopportune times, and eventually backs off unnecessarily; (b) Flow F1 traverses more hops than Flow F2. Some transport protocols, such as TCP, give more bandwidth to flow F2; (c) Flow F1, F2, F3, and F4 all share the same channel, but most transport protocols allocate more bandwidth to F4 than to others.*

they cannot prevent the hidden node from initiating its RTS/CTS sequence at inopportune times and subsequently suffering from long backoff delays. TCP exacerbates this unfairness problem because TCP senders further back off when their packets take a long time to get through the inhibited links. As a result, TCP flows traversing on an inhibited link could be completely suppressed in the worst case. Figure 1.3 (a) depicts this scenario, and Figure 1.4 presents the results from the corresponding experiment conducted on an IEEE 802.11a based 4-node testbed.

- *RTT-Dependent Unfairness:* TCP's fairness depends strongly on the RTT of the flows involved. Specifically, when two multi-hop TCP flows share the same wireless link, the flow traversing a fewer number of hops tends to acquire a higher share of bandwidth. While this is true even for TCP operations on the wired Internet, the problem is much more likely to occur in a WMN. In a WMN, most of the traffic is directed to/from gateway nodes that connect the WMN to the wired Internet. As different

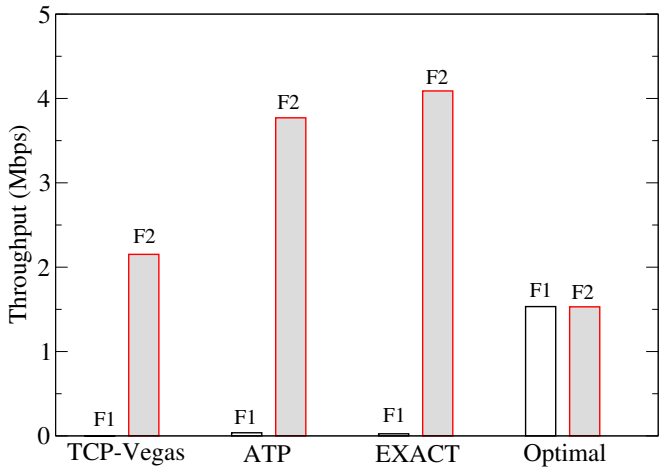


Figure 1.4: *Amplification of the hidden node problem (Fig 1.3 (a)) in existing wireless transport protocols.*

WMN nodes are bound to be different hops away from these gateway nodes, an RTT-independent fairness model is essential for effective mesh operations (Figure 1.3 (b)).

- *Channel Sharing Problem:* Existing transport protocols at best attempt to allocate a radio channel's bandwidth fairly among flows from a single node, rather than among all flows from all nodes that share the radio channel. As a result, a flow emanating from a node with fewer flows tends to get a larger than fair share of channel bandwidth. Figure 1.3 (c) depicts an example of this case, while Figure 1.5 presents the empirical results from the corresponding experiment.

- *Bad Fish Problem:* Even when two interfering links are not hidden from each other and have equal number of flows traversing them, IEEE 802.11's MAC allocates equal number of packet transmissions to each of them. However, these interfering links could be operating at vastly different link rates, e.g. 1 Mbps and 11 Mbps. In such a case, the effective throughput of 11 Mbps link becomes limited by that of 1 Mbps link. We call this problem the bad fish problem. If suppose the MAC layer instead allocates equal channel time to the two links, the 11 Mbps link would no longer be limited because of the interfering link operating at 1 Mbps link rate [1].

1.3.3 Routing Insecurity

Over-the-air communication makes IEEE 802.11 infrastructure networks inherently more vulnerable to attacks than their wired counterparts. Lack of security had probably been the

[1]One might rightly argue that if the fairness is made independent of the RTT, it should also be made independent of the channel encoding. In a sense, both RTT and channel encoding govern the end-to-end channel time consumed by every packet. However, our point is that TCP and most existing transport protocols do not have any knob to control/implement the desired fairness behavior for a given situation.

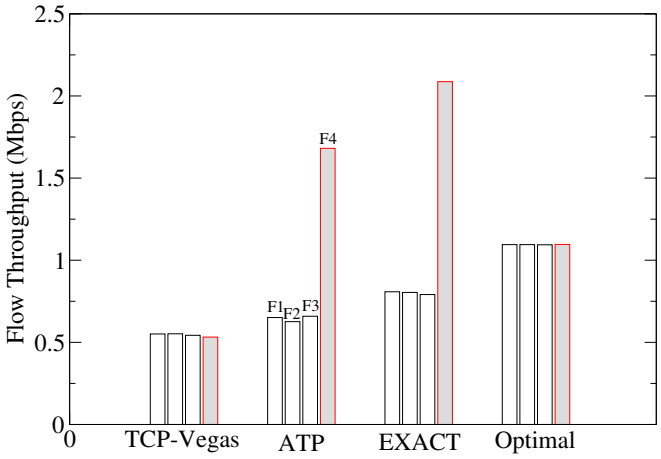


Figure 1.5: *While existing transport protocols work effectively when the participating flows share some common intermediate node, they fail to allocate bandwidth fairly when flows share a common radio channel (Fig 1.3 (c)). The optimal bandwidth allocation was achieved by exhaustively trying different sending rates. Note that although TCP-Vegas allocates bandwidth fairly among the flows, its overall channel utilization is much lower than the optimum.*

biggest roadblock to their earlier widespread deployments across enterprises. The problem is exacerbated in a wireless mesh network, where a single compromised router can potentially make the entire network unavailable. The failure probability of a WMN therefore increases exponentially with the network size. For wide acceptance, a wireless mesh network architecture needs to address the security concerns of the enterprise users.

One can classify these attacks into control plane attacks and data plane attacks. At the control plane level, a compromised node can prevent the establishment of consistent and optimal routing protocol state across network nodes. On the other hand, it is possible for a malicious node to participate faithfully in the network configuration, but later tamper with the data packets as they pass through it, thereby launching a data plane attack. Dealing with only one or the other kind of attacks is clearly insufficient from security viewpoint. For secure operations, we need to ensure that the routing protocol state is correctly established across all nodes and that all nodes indeed perform packet forwarding in a manner consistent with that state.

Let us look at these attacks in slight detail:

- *Topology and Traffic Misreporting:* The topology information and traffic statistics form the key input to any routing and channel allocation algorithm. A compromised node can falsify these inputs to adversely affect the output of the protocol computation. For instance, a compromised node could falsely advertise reachability to certain destinations, forcing data packets to pass through itself. It could similarly affect network reconfiguration by pretending to be a gateway node with connection to the wired network. A compromised node could advertise high loads on certain free

channels thereby forcing its interference neighbors to operate on other truly busy channels. Further, it could falsely claim to observe interference from any network node and similarly force them to operate on busy channels.

- *Protocol Computation:* Even if a compromised node were somehow forced to faithfully advertise the topology and traffic information, it could tamper with the portion of the protocol computation on itself. For instance, a compromised node could create load imbalance across gateways by switching to more loaded gateways. Similarly, a compromised node could create load imbalance across channels in its vicinity. It could also make the network unstable by similarly introducing artificial route flaps and 'channel flaps'. Finally, even if the channel and route computations were done in a centralized fashion, a compromised node could misconfigure itself resulting in sub-optimal network performance and/or network disconnectivity.

- *Packet Mishandling:* A compromised node can easily tamper with the packets that pass through it. It could thus launch debilitating attacks at both the control plane and the data plane. By tampering with the control packet stream, the compromised node can prevent nodes from reaching a consistent routing state. By tampering with the data packet stream, the compromised node can disrupt operations at the transport and the application layer.

  A compromised node can easily inject fabricated or duplicated packets into the network. It could also modify, drop, or re-order genuine packets for the connections going through it. While use of end-to-end encryption with per-packet sequence number can detect these attacks on end nodes, it does little in way of pinpointing and isolating such malicious nodes thus leaving the network vulnerable to a denial-of-service attacks by such malicious nodes. A compromised node could also increase end-to-end network latency by introducing artificial delays. Table 1.3 illustrates the impact of simulated packet mishandling attacks through an experiment based on ns-2 simulations.

Unfortunately, the standard 802.11 security mechanisms do not extend to a *multi-hop setting*. Moreover, the past research in providing security in multi-hop wireless networks has only provided piecemeal solutions [31, 32, 33, 34]. Many of them only deal with the control plane attacks, leaving the network susceptible to attacks during actual packet forwarding. While an end-to-end application layer security can detect when the underlying packet stream is being tampered with, they can do little to pinpoint and isolate the attacking node.

Further, it is unclear if the past research is directly applicable to securing wireless mesh networks. Most of the proposed solutions had been developed in the context of a *mobile ad hoc network* where due to lack of wired connectivity, it is impractical to assume existence of any centralized infrastructure. A wireless mesh network does not have this limitation and thus lends itself to much stronger security.

Attack Type	Network Throughput (Mbps)
No Attack	2.32
Delay	1.05
Drop	0.55
Duplication	2.24
Reorder	0.30
Congestion	0.12
Mis-forward	0.97
Fabrication	2.24
Corruption	1.29

Table 1.3: *An experiment to illustrate the impact of various packet mishandling attacks on network throughput with TCP flows. The experiments were conducted on a 7x7 grid network with 2 gateways. 7 randomly chosen nodes sent TCP traffic towards their respectively closest gateway. Each attack results in a drop in network throughput, and some attacks lead to a larger degradation in performance than others. In some of the cases, e.g. Drop, Reorder, and Congestion, the impact is further intensified due to TCP's congestion control.*

1.4 Hyacinth Proposal

In this dissertation, we propose an IEEE 802.11-based multi-hop wireless ad hoc network architecture (called Hyacinth) that incorporates several novel features to address the capacity, fairness, and security issues we just discussed. Economies of scale have motivated us to use IEEE 802.11 as the base technology for building WMNs. This also enables us to work directly with commodity 802.11-based interfaces and implement Hyacinth as a system software. Although our prototype uses 802.11 interfaces, the architecture is also applicable to the *802.16a* networks, where customer premise equipments form a mesh connectivity to reach the base station. We now discuss the key features of Hyacinth.

- **High-capacity Multi-channel Mesh Networking:** Due to limited capacity, a single-channel WMN can not support the high bandwidth requirements of an enterprise backbone or an ISP last-mile network. Fortunately, the IEEE 802.11b/g standards and IEEE 802.11a standard provide 3 and 12-25 non-overlapped frequency channels, respectively, which could be used simultaneously within a neighborhood. Ability to utilize multiple channels substantially increases the effective bandwidth available to wireless network nodes. Although there have been previous research efforts that aimed to exploit multiple radio channels in an ad hoc network, most of them were based on proprietary MAC protocols [15, 16, 17, 18], and therefore cannot be directly applied to wireless networks using commodity 802.11 interfaces. Hyacinth, on the other hand, employs multiple radio channels simultaneously by equipping each node with multiple NICs each operating on a different channel.

  To fully utilize the performance potential of this approach, we propose two traffic load-aware channel assignment and routing algorithms. The first algorithm takes

a *centralized* traffic engineering approach while the second algorithm works in a *distributed* manner. Both the centralized and distributed algorithms tune the network channel assignment and routing based on the the network topology and the latest traffic patterns. The centralized algorithm optimizes the network for any arbitrary traffic pattern, while the distributed algorithm optimizes the network for common case where most of the WMN traffic is directed to/from the wired gateway nodes.

- **Fair and Efficient Stateful Transport Protocol:** We propose *Link-Aware Reliable Transport Protocol (LRTP)* – a transport protocol specifically designed for WMNs. LRTP features an explicit rate-based congestion control mechanism, where each WMN node independently measures the amount of wireless bandwidth available to its outgoing links. After link bandwidth measurement, flows sharing the link are assigned their fair shares considering the number of flows and their current requirements. LRTP also features a hop-by-hop reliable packet delivery mechanism that utilizes the 802.11 link-layer ACK to infer delivery status on each hop and to perform local packet loss recovery. Unlike 802.11's link-layer retransmission, LRTP performs intelligent channel-aware transmission scheduling to avoid head-of-the-line blocking problem [19].

  Additionally, LRTP features an idealized congestion control mechanism named *Coordinated Congestion Control aLgorithm (C3L)*. C3L performs global bandwidth allocation and thus provides end-to-end flow-level max-min fairness despite weaknesses in the MAC layer. C3L incorporates an advanced topology discovery algorithm that is able to identify not only all the usable wireless links between nodes, but also their interference relationships, including inhibition relationships due to the hidden node problem. C3L takes a centralized traffic engineering approach, which based on the latest traffic loads continuously computes the max-min fair share of individual wireless links. Unlike previous solutions to this problem, C3L is designed to work with multi-hop flows and takes into account both inter-flow and intra-flow dependency. Furthermore, it incorporates a general collision domain capacity re-estimation algorithm that can effectively resolve the unfairness problem due to hidden nodes.

  While the WMN nodes in the proposed architecture run the LRTP protocol, the end-user mobile nodes as well as the nodes on the wired network still run the original TCP using I-TCP approach [20]. To achieve this, each ingress/egress WMN node employs a TCP-LRTP proxy that transparently converts an end-to-end TCP connection into three sub-connections: a TCP sub-connection running from end-user mobile to the ingress WMN node, an LRTP sub-connection from ingress WMN node to egress WMN node, and another TCP sub-connection from egress WMN node up to the final end-point of the original connection. This architecture is shown in Figure 1.6.

- **Secure Routing and Forwarding:** In Hyacinth, network security is considered a first-class requirement at par with connectivity and performance. Our proposed architecture and its associated protocol secure network operations at both the control plane as well as the data plane. A typical router's function is restricted to providing

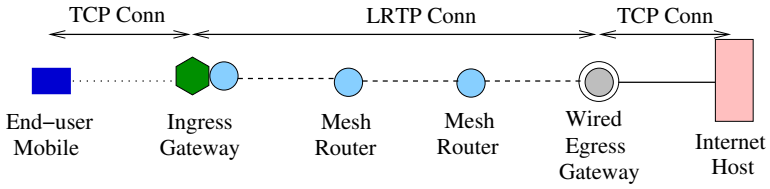


Figure 1.6: *Transparent splitting of end-user TCP connections into three sub-connections. Note that TCP ACKs never traverse the WMN.*

input to the route selection and channel allocation computation, while the actual computation is done on a centralized controller. The collection of these inputs (topology and traffic statistics) from the network nodes as well as dissemination of the reconfiguration packets (channel and route changes) to the network nodes is made secure. Moreover, a cross-check based mechanism is used to validate the inputs for any false representation.

A transparent monitoring layer runs on each node to ensure untampered forwarding of control as well as data packets in accordance with the routing and channel allocation decisions made at the central controller. Lastly, a compromised node mishandling packet communication is quickly identified to be isolated from the network. All these protocol mechanisms significantly enhance the availability of a Hyacinth network in presence of compromised, misconfigured, or broken nodes.

1.5 Contributions

This dissertation proposes a novel wireless mesh network architecture that can be readily built using 802.11-based wireless LAN hardware and is specifically tailored to multi-hop wireless access network applications. In particular, this dissertation makes the following research contributions:

- We propose a multi-channel wireless mesh network architecture in which each node is equipped with multiple IEEE 802.11 interfaces, present the research issues involved in this architecture, and demonstrate through an extensive simulation study the potential gain in aggregate bandwidth of this architecture [21, 22].

- We develop and evaluate a novel centralized channel assignment and bandwidth allocation algorithm for the proposed multi-channel wireless mesh networks. The algorithm reaps the full potential of proposed architecture by exploiting the network topology and traffic load information, and works with any given routing algorithm. Even with the use of just 2 NICs per node, the proposed algorithm improves the network cross-section goodput by factors of up to 7 when compared with single-NIC architecture [21].

- We design and implement a fully distributed channel assignment algorithm and a spanning tree-based load-balancing routing algorithm that can dynamically adapt to

traffic load changes as well as network failures automatically. Again, a comprehensive performance study shows significant bandwidth improvements of multi-channel WMNs over single-channel WMNs with use of the proposed algorithms [23].

- We develop a hop-by-hop reliable packet delivery mechanism that utilizes link-layer ACK information for packet loss detection and local retransmission. It thus largely eliminates transport-protocol ACKs that currently consume substantial bandwidth of a multi-hop wireless path [24].

- We devise an explicit rate-based congestion control mechanism that measures the effective wireless link bandwidth and allocates bandwidth shares among flows sharing a wireless link taking into account inherent bandwidth demands of end nodes [24].

- To achieve max-min fairness over an 802.11-based WMN, we design and implement a coordinated congestion control algorithm (C3L) that performs global bandwidth allocation and thus provides end-to-end flow-level max-min fairness despite weaknesses in 802.11 MAC layer. We develop an advanced topology discovery algorithm that is able to identify not only all the usable wireless links between nodes, but also their interference relationships, including inhibition relationships due to the hidden node problem. Unlike previous congestion control algorithms, C3L accounts for both inter-flow and intra-flow dependency, and incorporates a general collision domain capacity estimation algorithm that can effectively resolve the unfairness problem due to hidden nodes [25, 26].

- We design and implement security mechanisms as first-class components into the network. Our techniques secure the collection of topology and traffic statistics as well as provide checks against falsified information. The route and channel computation is done on a central controller and the resulting data plane state is securely distributed across the network. A transparent authentication and monitoring layer is then used to ensure that packets, both data and control, are forwarded in accordance with that state. To the best of our knowledge, this is one of the first attempts to comprehensively secure WMN against external attackers and internal compromised nodes.

- We develop a fine-grained packet tracing mechanism that can quickly trace routing misbehavior, including delaying of packets, down to specific nodes, so they can be isolated from the network.

- We implement one of the first Hyacinth prototypes that incorporates multi-channel networking and stateful transport protocol concepts. An empirical measurement study based on the fully working Hyacinth prototype validates our corresponding simulation results [23, 24].

1.6 Outline

The rest of the dissertation is organized as follows.

Chapter 2 formulates the capacity, fairness, and security problems associated with the Hyacinth architecture.

Chapter 3 discusses network-layer resource management techniques to address the capacity issues. Specifically, it presents the two sets of traffic load-aware channel assignment and routing algorithms. Further, it presents and analyzes the results of a comprehensive ns-2 simulation study of the proposed architecture and algorithms. The chapter ends with the evaluation of a 9-node Hyacinth prototype that we built using commodity PCs and 802.11a/b/g network cards.

Chapter 4 presents the details of the stateful transport protocol LRTP, and how it addresses the fairness and efficiency issues associated with the existing transport protocols. Specifically, it presents the hop-by-hop reliability mechanism, the explicit rate-based congestion control algorithm, and the coordinated congestion control algorithm (C3L). The chapter ends with a performance evaluation of LRTP's mechanisms using ns-2 simulations and 9-node MiNT testbed experiments.

Chapter 5 focuses on the security issues associated with a Hyacinth network. It presents a distributed secure routing approach that we explored earlier and the problems associated with that approach. It then details our proposed centralized security protocol, and provides both analytical proofs and simulation-based results to demonstrate its effectiveness.

Chapter 6 presents the Hyacinth prototype. Specifically, it details the implementation issues with the multi-channel mesh networking techniques, and the stateful transport protocol. It also speculates on the potential implementation issues with the proposed security mechanisms.

Chapter 7 compares and contrasts Hyacinth work with past as well as current research efforts in the field.

Chapter 8 concludes the dissertation with a summary of research contributions, and an outline of the future directions of this work.

Chapter 2

Problem Formulation

This chapter formulates the core research problems addressed by the dissertation. In Section 2.1, we formulate the network-layer resource management problem in a multi-channel multi-radio wireless mesh network. We then motivate the need for a transport protocol that can deal with the intricate interference relationships between wireless links, and formulate the problem of developing such a fair and efficient transport protocol in Section 2.2. In Section 2.3, we discuss why cryptography alone is not sufficient to secure a mesh network against compromised nodes, and present the problem of building resilience into the Hyacinth control protocol and data forwarding layer.

2.1 Problem 1: Network-Layer Resource Management

Conventional ad hoc networking research efforts focused mostly on providing basic connectivity over a multi-hop wireless network. These ad hoc networks are typically based on single-channel architecture, wherein to keep the network connected all nodes in the network operate over the same channel. As shown in the previous chapter, this limits the amount of capacity available for end-to-end communication and in turn the applicability of this architecture to build enterprise backbone or ISP last-mile networks where bandwidth requirements are anything but small. Fortunately, the IEEE 802.11b/g standards and IEEE 802.11a standard provide 3 and 12-25 non-overlapped frequency channels, respectively, which can be used simultaneously within a neighborhood. Ability to simultaneously utilize multiple channels substantially increases the effective bandwidth available to wireless network nodes. Although there have been previous research efforts that aimed to exploit multiple radio channels in an ad hoc network, most of them were based on proprietary MAC protocols [15, 16, 17, 18], and therefore cannot be directly applied to wireless networks using commodity 802.11 interfaces.

One of the implicit design choices for wireless ad hoc networks has been to employ a single interface on each. A single-NIC architecture inherently limits the whole network to operate in one single channel, because use of multiple channels means that the subset of nodes using one channel are disconnected from other nodes that are not using the same

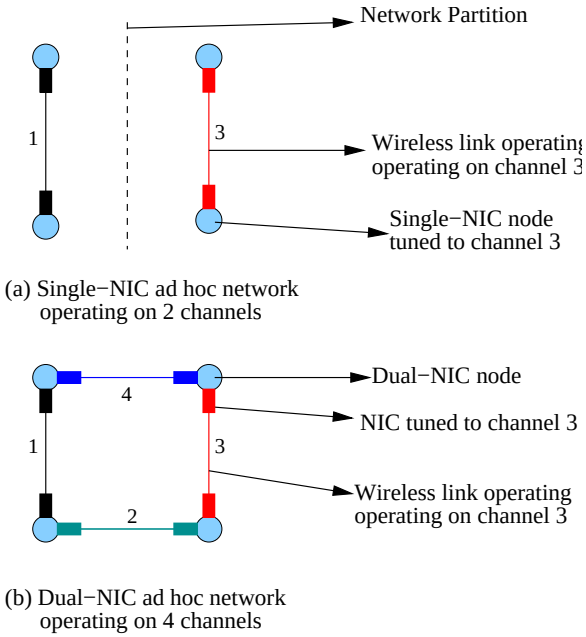


Figure 2.1: *(a) Single-NIC network gets disconnected when operating in multiple channels, (b) Multiple NICs on each network node enable forming a connected multi-channel network.*

channel (Figure 2.1). Cross-channel communication in this case requires either channel-switching capability within each node, or multiple NICs per node each tuned to operate on different channels. Channel-switching requires fine-grained synchronization among nodes as to when any node will transmit/receive over a particular channel. Such fine-grained synchronization is difficult to achieve without modifying the 802.11 MAC layer. Therefore, in Hyacinth architecture, we choose to enable cross-channel communication by using *multi-radio routers* where each router is equipped with multiple 802.11 commodity NICs each operating on a different channel.

2.1.1 Multi-Radio Mesh Network Model

As shown in Fig 2.2, the wireless mesh network (WMN) architecture that this work targets at consists of fixed wireless routers, each of which is equipped with a traffic aggregation access point that provides network connectivity to end-user mobile stations within its coverage area. In turn, the wireless routers form a multi-hop ad hoc network among themselves to relay the traffic to and from mobile stations. Some of the WMN nodes serve as gateways between the WMN and a wired network. All infrastructure resources such as file servers, Internet gateways and application servers, reside on the wired network and can be accessed through any of the gateways. In the most general case, the physical links between gateways and the wired network can be a wired link, or a point-to-point 802.11 or 802.16 wireless link.

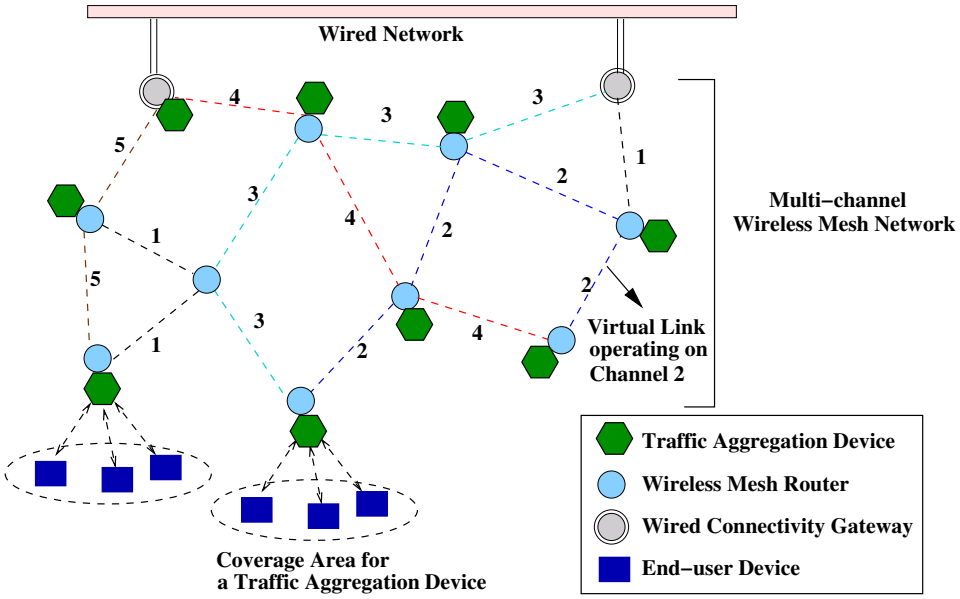


Figure 2.2: *The Hyacinth architecture consists of a multi-channel wireless mesh network (WMN) core, which is connected to a wired network through a set of wired connectivity gateways. Each WMN node has multiple interfaces, each operating at a distinct radio channel. A WMN node is equipped with a traffic aggregation device (similar to an 802.11 access point) that interacts with individual mobile stations. The multi-channel WMN relays mobile stations' aggregated data traffic to/from the wired network. The links between nodes denote direct communication over the channel indicated by the number on the link. In this example, each node is equipped with 2 wireless NICs. Therefore the number of channels any node uses simultaneously cannot be more than 2; the network as a whole uses 5 distinct channels.*

Each node in a multi-channel WMN is equipped with multiple 802.11-compliant NICs, each of which is tuned to a particular radio channel for a relatively long period of time, such as several minutes or hours. For direct communication, two nodes need to be within *communication range* of each other, and need to have a common channel assigned to their interfaces. A pair of nodes that use the same channel and are within *interference range* may interfere with each other's communication, even if they cannot directly communicate. Node pairs using different channels can transmit packets simultaneously without interference. For example, in Fig 2.2, each node is equipped with 2 NICs. The "virtual links" shown between the nodes depict direct communication between them, and the channel used by a pair of nodes is shown as the number associated with the connecting link. This example network totally uses 5 distinct channels. Note that mobile nodes have only a single NIC, and the interaction between mobile nodes and a traffic aggregation device is similar to the infrastructure mode operation of the IEEE 802.11 standard.

A multi-NIC-per-node wireless mesh network architecture raises two research questions:

1. Which of the 3 or 12-25 radio channels should be assigned to a given 802.11 interface? For two nodes to communicate with each other, their interfaces need to be assigned to a common channel. However, as more interfaces within an interference range are assigned to the same radio channel, the effective bandwidth available to each interface decreases. Therefore, a channel assignment algorithm needs to balance between the goals of maintaining connectivity and increasing aggregate bandwidth.

2. How packets should be routed through this multi-interface wireless ad hoc network? The routing strategy in the network determines the load on each 802.11 interface, and in turn affects the bandwidth requirement and thus channel assignment of each interface.

2.1.2 Channel Assignment Problem

Intuitively, the goal of channel assignment in a multi-channel WMN is to bind each network interface to a radio channel in such a way that the available bandwidth on each virtual link is proportional to the load it needs to carry. This problem is different from the channel assignment problem in cellular networks [27], because adjacent base stations in a cellular network are connected through wired networks, whereas adjacent nodes in a WMN can only communicate with each other through wireless links. Therefore, if one simply assigns a least-used channel to a WLAN interface, there is no guarantee that the resulting mesh network is even connected. A Hyacinth node *needs* to share a common channel with each of its communication-range neighbors with which it wishes to set up a virtual link or connectivity. On the other hand, to reduce interference a node should minimize the number of neighbors who share a common channel. More generally, one should break each collision domain into as many channels as possible while maintaining the required connectivity among neighboring nodes.

The channel assignment problem can actually be divided into two subproblems: (1) neighbor-to-interface binding, and (2) interface-to-channel binding. Neighbor-to-interface binding determines through which interface a node uses to communicate with each of its neighbors with whom it intends to establish a virtual link. Because the number of interfaces per node is limited, each node typically uses one interface to communicate with multiple of its neighbors. Interface-to-channel binding determines which radio channel a network interface should use.

The main constraints that a channel assignment algorithm needs to satisfy are:

- The number of distinct channels that can be assigned to a WMN node is bounded by the number of NICs it has.
- Two nodes that communicate with each other directly should share at least one common channel.
- The raw capacity of a radio channel within an interference zone is limited.
- The total number of non-overlapped radio channels is fixed.

Conceptually, links that need to support higher traffic load should be given more bandwidth than others. This means that these links should use a radio channel that is shared among a fewer number of nodes. An ideal load-aware channel assignment would distribute radio resource among links in a way that matches their expected traffic loads.

For clarity, it is useful to formulate the problem as a *constrained edge coloring problem* [28]. Here, we are given the following inputs:

- Network graph $G = (V, E)$, where each vertex v_i corresponds to a wireless mesh network node, and each edge e_{ij} represents a direct communication link from v_i to v_j if the nodes were to have a common channel between them.
- Interference matrix $I = [i_{m,n}]$, where $i_{m,n}$ is 1 if v_m and v_n interfere with each other (assuming common channel between them), and 0 otherwise.
- Traffic matrix $T = [t_{ij}]$ where t_{ij} is the amount of traffic going over link $i - j$.
- Total number of radio channels r.
- Number of radio interfaces k on each node.
- Capacity of each channel Cap.

The objective is to devise a channel allocation vector $C = [C_i]$, where C_i is the channel to be used by the virtual link i, such that:

- For any link i, $C_i <= r$,
- For any node n and all links j incident onto it $|[C_j]| <= k$, and
- For any channel p and any clique q in I_p, $\sum_{e \epsilon q} t_e <= Cap$, where I_p is the part of interference graph I that is operating over channel p.

2.1.3 Load-Balancing Routing Problem

Channel assignment depends on the load on each virtual link, which in turns depends upon routing. Unlike a conventional mobile ad hoc routing protocol, where any node could communicate with any other node, the traffic distribution of a WMN is skewed – most of the WMN nodes communicate primarily with nodes on the wired network. This is the case because most users are primarily interested in accessing the Internet or enterprise servers, both of which are likely to reside on the wired network [9]. The goal of the routing algorithm is thus to determine route(s) between each traffic aggregation device and the wired network in such a way that balances the load on the mesh network, including the links to the wired network. Load balancing helps avoid bottleneck links, and increases the network resource utilization efficiency.

Finally, routing can also increase the tolerance of network against node failures by coming up with multiple node-independent routes for each pair of end-hosts [29]. At run-time, if a node fails leading to a path failure, the affected nodes can have alternate paths to route packets to their destinations.

While it is harder to formulate the general form of network load balancing problem, it is possible to look at the gateway-level load-balancing routing as a *multi-level bin-packing problem*. Specifically, given:

- Traffic vector $T = [t_n]$, where t_n is the traffic generated by the node n,
- Neighbor set $N = [N_n]$, where N_n is the set of neighbors of the node n, and
- Set of gateway nodes $G = [G_j]$.

The goal is to find a parent $p(n)$ for each node n such that:

- $p(n) \epsilon N_n$,
- n's ancestor $p^i(n) \epsilon G$ for some i, and
- $max_j(\sum_{n \epsilon C(G_j)}(t_n))$ is minimized, where $n \epsilon C(G_j)$ iff $p^i(n) = G_j$ for some i.

2.1.4 Evaluation Metric

The ultimate goal of the channel assignment and routing algorithms is to maximize the overall network goodput, or the number of bytes it can transport between the traffic aggregation devices and the wired connectivity gateways within a unit time. To formalize this goal, we define the *cross-section goodput* of a network as

$$X = \sum_a min(\sum_i C(a, g_i), B(a)) \tag{2.1}$$

where $C(a, g_i)$ is the *useful* network bandwidth available between a traffic aggregation device a and a gateway node g_i. If the bandwidth requirement between a traffic aggregation

device a and the wired network is $B(a)$, then only up to $B(a)$ of the bandwidth between the node a and all the gateway nodes is considered useful. This criteria ensures that only the usable bandwidth of a network is counted towards its cross-section throughput, hence the term cross-section goodput. The goal of the channel assignment and routing algorithms is to maximize this cross-section goodput X.

2.1.5 Other Dimensions of Resource Management

Even with use of multiple radio interfaces, it is not always possible to operate two physically interfering links over non-interfering channels. Due to limited number of interfaces, a node invariably needs to use each of its interfaces to communicate with multiple neighbors. As a result, multiple interfering links may still end up operating on the same channel. To alleviate such interference, one can further control the topology by varying the following network parameters:

1. *Transmission Power:* Most 802.11-based interfaces support transmission power control as a mean to control the range and/or to control the interference. Lower transmission power implies smaller interference range, but simultaneously smaller number of operable links in the network. Higher transmission power effectively increases the density of the network, providing longer-range links at the cost of increased interference.

2. *Directionality of Antenna:* It is also possible to equip each interface with a directional antenna. This further shapes the interference zone of the network interface limiting the interference caused by its packet transmissions. Similar to radio channel assignment, it may not be practically possible to steer the antenna to point to a different direction for each packet exchange. Therefore, the direction of the antenna need to be fixed along with the channel used by it based on the long-term network traffic profile.

3. *Transmission Scheduling:* IEEE 802.11 uses CSMA/CA protocol to decide when to begin a packet transmission. While distributed in nature, it invariably leads to collisions at times. Furthermore, it introduces the well-known hidden terminal problem and the exposed terminal problem. Theoretically, it is possible to carefully schedule packet transmissions such that no two conflicting links that are operating on the same channel are scheduled to transmit at the same time. Such fine-grained channel scheduling can overcome the inefficiencies of the 802.11's MAC layer.

2.2 Problem 2: Fair and Efficient Mesh Transport Protocol

Use of multiple channels helps to eliminate adjacent links' interference and to improve *end-to-end* path capacity. The next research question is how to enable applications to make the

most of this raw network-layer capacity, a responsibility traditionally fulfilled by transport protocol. An effective transport protocol must fairly and efficiently allocate the network bandwidth among multiple competing flows, while minimizing its own overhead. There are several unique characteristics of wireless mesh networks that make this a hard problem to solve:

- *High Per-packet Overhead:* One characteristic of IEEE 802.11 wireless networks is high per-packet overhead. Regardless of its payload size, an 802.11 packet incurs large MAC contention, PLCP header, and link-layer ACK overhead. This makes conventional reliability mechanisms such as per-packet acknowledgement and end-to-end loss detection/retransmission unsuitable for wireless mesh networks.

- *High Error Rates:* Wireless networks commonly observe transient packet drops induced by bit errors. However, on a wired network packet drops pretty much always indicate congestion on an intermediate router. In fact, packet drops are commonly used as a mechanism to indicate router congestion and thus to request the corresponding senders to reduce their sending rate.

- *Hidden Terminals:* The MAC protocol of IEEE 802.11 introduces serious unfairness among competing nodes when used in multi-hop WMNs [14]. The well-known *hidden node problem* [30] causes one wireless link's transmission to be inhibited by another link, eventually leading to unequal bandwidth allocation between the two. More specifically, while the RTS/CTS messages in 802.11's MAC protocol effectively stop a hidden node from interfering with an on-going communication transaction, they cannot prevent the hidden node from initiating its RTS/CTS sequence at inopportune times and subsequently suffering from long backoff delays. To address these issues, the transport-layer sending rate needs to be carefully controlled.

- *Channel Space Sharing:* Unlike wired networks, nearby wireless links sharding a radio channel can interfere even if they do not share any physical node. This raises another fairness problem, as a radio channel's bandwidth needs to be fairly sharded by all flows emanating from all nodes that share the radio channel.

2.2.1 Stateful Transport Protocol

Most transport protocols proposed in the literature or in use today were originally designed to work over the wired Internet. Many of them strive to scale up to gigabits/sec links and are therefore designed to be stateless in the sense that intermediate routers never need to keep transport-layer state, e.g. unacknowledged flow packets or flow's sending rate information. While such scalability is definitely desirable for wired Internet, it is less relevant for wireless networks, because the link speed of wireless networks is much lower and the number of legacy devices are few to justify backward compatibility considerations.

In this dissertation, we argue that it is acceptable to maintain transport-layer state in intermediate wireless routers as long as it can maximize the utilization efficiency of the limited wireless link capacity. More specifically, we explore the other extreme of the transport

protocol design space: *stateful transport protocol* in the context of multi-channel wireless mesh networks. In our stateful transport protocol, the intermediate routers are aware of the transport level flows and participate in ensuring flow reliability as well as inter-flow fairness. Our stateful transport protocol comprises of the following two mechanisms: (1) Low-overhead reliability layer, and (2) Max-min fair congestion control mechanism.

2.2.2 Low-Overhead Reliability Layer

The reliable packet delivery mechanism infers the delivery status of each data packet, and retransmits packets that are determined to be lost. As discussed above, a simple end-to-end loss determination and retransmission is not suitable due to high per-packet overhead. IEEE 802.11 already employs a local retransmission mechanism, where it retransmits a link-layer packet upto a certain fixed number of times and with increasing backoff, if it does not get a link-layer acknowledgement back from the receiver. Unfortunately, this mechanism easily leads to head-of-the-line blocking problem from ineffective link scheduling. Other wireless links sharing the same interface get blocked, even if they can achieve error-free transmissions in the short-term. The question here is how to utilize intermediate nodes to devise a low-overhead reliability layer.

2.2.3 Max-Min Fair Congestion Control

The congestion control mechanism estimates the available bandwidth between the source and the destination, and allocates a fair share to each sender node. The goal of congestion control is to ensure high network utilization while avoiding congestion and maintaining inter-flow fairness. The research question here is how to devise a congestion control mechanism that can achieve end-to-end max-min flow fairness over 802.11-based WMNs that is inherently unfair. Max-min fairness, a popular notion of fairness in both wired and wireless networks, dictates that a flow is allowed to receive as much bandwidth as it can so long as other flows receiving a smaller share are not adversely affected.

Formally, we are given the following as inputs:

1. Network graph $G = (V, E)$, where each vertex v_i corresponds to a wireless mesh network node, and each edge e_{ij} represents a direct communication link from v_i to v_j.

2. Interference matrix $I = [i_{m,n}]$, where $i_{m,n}$ is 1 if v_m and v_n interfere with each other, and 0 otherwise.

3. Flow vector $F = [f_i]$, where each flow f_i is characterized by a node pair (v_m, v_n), which represents the flow's source v_m and destination v_n.

4. Routing matrix $R = [r_{m,n}]$, where $r_{m,n}$ is the ordered set of nodes that a packet from v_m to v_n passes through.

5. Maximum channel capacity C_{max}.

The goal is to come up with a bandwidth allocation vector $B = [b_i]$, where b_i is the bandwidth allocation to flow f_i, and B is max-min fair. That is, increasing any b_i to $b_i + \delta$ leads to reduction in allocation b_j of some flow, where $b_j < b_i$. For simplicity of description, we assume greedy flows with infinite bandwidth requirements. Our approach can be easily extended to flows with finite bandwidth requirements.

There are several unique characteristics o an 802.11-based WMN, that make this a hard problem to solve:

1. *Intra-flow dependency:* The bandwidth allocation to a multi-hop flow across all hops should be the same as the bandwidth assignment on its bottleneck link. Allocating more bandwidth on any other hops represents a waste of resource.

2. *Shared radio channel:* Unlike a wired network, where each link can operate independently without interfering with other links, a wireless link shares the radio channel with other links in its proximity. A wireless network is composed of multiple overlapped collision domains.

3. *MAC-dependent capacity:* While the capacity of any collision domain cannot exceed C_{max}, its effective capacity is dependent on how much time the MAC layer spends in backoffs, transmission and retransmissions, and in general cannot be known beforehand.

4. *Asymmetric MAC Contention:* Channel sharing within a collision domain could be asymmetric. Here, the inhibited sender has incomplete information about the channel status (busy or idle), and attempts its communication at inopportune times. As a result, the attempts fail more frequently and the backoff delay is increased. The end result is that transmissions on inhibited link are less likely to go through successfully than on inhibiting links.

2.3 Problem 3: Secure Routing and Forwarding

Security has remained a major roadblock to widespread deployments of IEEE 802.11-based infrastructure mode WLANs. Security is an even bigger issue for a WMN, because it serves as the backbone rather than an end-point of the network. While use of cryptographic techniques can deal with external attacks by preventing unauthorized devices from subverting the network operations, they do not directly help when one or more of the network routers are compromised. Broadly speaking, a compromised Hyacinth router can disrupt network operations in two different ways:

1. It can disrupt the routing protocol operations. For example, a compromised router can inject bogus routing packets and drop/modify legitimate routing packets, and thus prevent routers to achieve a consistent routing state.

2. It can disrupt the forwarding mechanism. For example, a compromised router may drop all payload packets passing through it essentially forming a black hole. A compromised router can also modify the payload packets or inject fabricated packets disrupting operations at transport and application layers.

Most of the past efforts in this field had been on providing security in a mobile ad hoc network setting [31, 32, 33, 34]. Due to lack of wired connectivity, it is impractical in a mobile ad hoc network to assume existence of any centralized infrastructure, e.g. a certificate authority. This makes the problem of wireless mesh network security fundamentally different from that of providing security in a mobile ad hoc network.

Further, the focus of our study is tree-based load-balancing routing protocol. Although tree-based routing form an important point in the design space for routing protocols, little attention has been paid to the problem of providing routing security in this class of networks. On the surface, a tree topology seems easier to tackle. However, unlike a general mesh network, a network with tree topology disallows use of redundant paths between nodes to secure communication.

2.3.1 Comparison With Wired Internet Security

Routing infrastructure security is an important concern even for wired Internet [35, 36, 37]. Therefore, an important question to ask is that from security standpoint, how different is a wireless mesh network from its wired counterpart. We believe that the following characteristics set apart the security problem of a WMN:

- *Control of Nodes:* The wired Internet is composed of thousands of autonomous systems (ASes) each of which controls its part of the network. The ASes are unwilling to share important information with each other. In contrast, a WMN is usually owned and controlled by a single ISP. One can thus assume a cooperating environment while designing WMN protocols.

- *Broadcast Media:* It is obvious that the broadcast nature of wireless medium makes it more vulnerable than a wire. What is not obvious is that the broadcast nature makes it possible for each network node to monitor the communication activities of all its neighbors by sniffing the “air”. This enables use of distributed watchdog solutions where each node is monitored by all its neighbors for signs of compromise. Note that this is only true for single-channel WMNs: a multi-channel WMN node needs to send multiple unicasts in order to broadcast a packet to all its neighbors.

- *Network Scale:* A protocol designed for wired Internet needs to scale up both in terms of link speeds as well as in terms of network size. For instance, it is unreasonable to expect each router to know the entire Internet topology. While such scalability is definitely desirable for wired Internet, it is less relevant for wireless networks, because the link speed of wireless networks is much lower.

- *Nascent Technology:* In a wired Internet, any new protocol deployment needs to be concerned about backward compatibility. At the moment, this is not a requirement from WMN. The number of legacy devices are few to justify any backward compatibility consideration.

2.3.2 Attack Model

The core Hyacinth protocol already deals with fail-stop failures by reconfiguring network topology to work around a failed node. From security viewpoint, we concern ourselves with Byzantine attacks. Specifically, we assume that an attacker is able to hack into some of the nodes and inject malicious code into the compromised nodes with the goal of subverting the network protocol and communication.

Let us now discuss various possible attacks that we wish to tackle in this research. Broadly, one can classify the attacks into control plane attacks and data plane attacks. At the *control plane level*, a compromised node can prevent the establishment of consistent and correct routing protocol state across network nodes. At the *data plane level*, a malicious node can prevent correct forwarding of the data packets. The attacks launched by tampering the forwarding of the control packets could be classified either way; we classify them as data plane attacks.

Dealing with only one or the other kind of attacks is clearly insufficient from security standpoint. For secure operations, we need to ensure that the routing protocol state is correctly established across all nodes and that all nodes indeed reconfigure themselves and perform packet forwarding in a manner consistent with that state.

2.3.2.1 Topology/Traffic Misreporting

The topology information and traffic statistics from other nodes form the input to the routing and channel allocation protocol. A compromised node can falsify these inputs to adversely affect the output of the protocol computation. The following representative examples illustrate various attacks that can thus be launched.

- A compromised node could pretend to be a gateway node with connection to the wired network. It could thus entice its neighbors to join its subtree forcing all their packets to pass through itself.

- A compromised node could claim to be a parent (or more generally an ancestor) of any network node. It could thus receive all packets originally destined to the victim node.

- A compromised node could advertise high loads on certain free channels thereby forcing its interference neighbors to operate on other truly busy channels. Further, it could falsely claim to observe interference from any network node and similarly force them to operate on busy channels.

2.3.2.2 Protocol Miscomputation / Node Misconfiguration

If the protocol computation is distributed across the network nodes, then it is also possible for an attacker to tamper with the protocol computation on the compromised node. For instance, a compromised node could create load imbalance across gateways by deciding to switch to more loaded gateways. A similar attack could result in load imbalance across channels in the vicinity of the compromised node. Although the network will eventually adjust to accommodate these actions of compromised nodes, the compromised node could repeatedly switch gateways and channels to create artificial route flaps and 'channel flaps' making the network unstable.

Even if the protocol computation is not distributed, it is possible for a compromised node to configure itself without regards to the output of the protocol computation. Such misconfigurations could result not only in sub-optimal network performance, but also lead to unexpected network disconnectivity.

2.3.2.3 Packet Mishandling

A compromised node can easily tamper with the packets that pass through it. It could thus launch attacks at both the control plane and the data plane. By tampering with the control packets, the compromised node can prevent nodes from reaching a consistent routing state. By tampering with the data packet stream, the compromised node can make the network unusable from the application standpoint.

More concretely, a compromised node can modify, drop, or re-order genuine packets for the connections going through it. It could also inject fabricated or duplicated packets into the network. While use of end-to-end encryption with per-packet sequence number can detect these attacks on end nodes, it does little in the way of pinpointing and isolating such malicious nodes thus leaving the network susceptible to a denial-of-service attacks by such malicious nodes.

A compromised node could also increase end-to-end network latency by introducing artificial delays. It could do so by mis-forwarding the packets adding to their path length. It could also achieve this effect by simply holding packets for some time before forwarding them to the correct next hop.

2.3.3 Research Question

The overall research question here is how to make the routing infrastructure sufficiently resilient so that it can operate correctly in presence of compromised routers. In general, it is possible for nodes to collude, but for this research we focus on the case when the compromised routers are isolated and do not collaborate among themselves. More formally:

Assume that a small number (k) of isolated routers in a network of n routers are compromised, and send carefully but independently crafted malicious packets to create inconsistent routing state in the network nodes. How can the network automatically (1) detect the

situation, (2) pinpoint/isolate these compromised nodes, and (3) maintain correct protocol operations in the rest of the network?

Chapter 3

Resource Management in Multi-channel WMNs

3.1 Introduction

In this chapter, we devise network-layer resource management techniques to reap the raw performance potential of our proposed multi-channel multi-radio wireless mesh network architecture. Recall from Chapter 2 that the first problem here is that of *load-balancing routing*: How to find a multi-hop path from each access point to one of the wired-connectivity gateways such that the aggregated traffic on the gateways is well balanced? The second problem is that of *channel assignment*: How to assign channels to individual radio interfaces such that the resulting link capacities closely match the imposed link loads?

Even with a complete knowledge of network topology and traffic matrix, both the load-balancing routing and the channel assignment problems are NP-hard. The hardness of the former can be proved by a simple reduction from the well-known bin packing problem. We demonstrate the hardness of the latter by reducing multiple subset sum problem to it.

To establish a baseline, we propose a centralized traffic engineering-based approach that jointly solves the channel assignment and routing problem. Specifically, we devise an iterative algorithm that visits the channel assignment and routing steps in each iteration. The channel assignment is done using a greedy algorithm that assigns locally optimal channel to each wireless link in the network which traversing them in the order of their criticality (measured by their expected traffic loads). The routing step executes the Bellman-Ford algorithm while taking into account the residual link capacities.

As a practical solution, we propose a tree-based distributed algorithm. In this algorithm, an 802.1D-like mechanism is used to construct spanning trees around each gateway node. The spanning trees formation not only takes into account the hopcount from the node to the gateways, but also the residual traffic capacities of the gateways. Based on the channels' traffic load information exchanged with the neighbors, each node optimizes the channel assignment to its interfaces and co-ordinates such channel assignment with its immediate neighbors.

Our evaluations suggest that even with use of just 2 interfaces on each node, the proposed algorithms improve the network throughput by a factor of 6 to 7 times when compared with single-interface architecture. The result is not surprising as even though any particular node can only operate over 2 channels simultaneously, the network as a whole can utilize several channels boosting the network capacity.

The rest of the chapter is organized as follows. We first prove the NP-hardness of the channel assignment in Section 3.2. Section 3.3 discusses a centralized traffic engineering-based approach to derive effective channel assignment and end-to-end routes. Section 3.4 devises a distributed channel assignment and routing algorithm. Section 3.5 thoroughly evaluates the performance of two sets of algorithms using ns-2 simulations, while Section 3.6 presents performance results obtained from a 9-node prototype implementation. Finally, Section 3.7 concludes the chapter with a summary of contributions.

3.2 NP-hardness of Channel Assignment

Even with a complete knowledge of network topology and traffic matrix, the channel assignment problem as well as the load-balancing routing problem are both *NP-hard*. The load-balancing routing problem can be shown to be hard by simple reduction from the knapsack problem [38]; the proof for channel assignment hardness is slightly more involved.

Given the expected load e_i on each virtual link i, the goal of channel assignment algorithm is to assign a channel c_j to each network interface j, such that the resulting available bandwidth b_i on each virtual link i is at least equal to its expected load e_i. There are m physical channels each with a total capacity of C in any given interference zone. Each node is equipped with k wireless network interfaces.

We now prove the NP-hardness of the channel assignment problem by reducing the *Multiple Subset Sum Problem* [39] to the channel assignment problem. The multiple subset sum problem can be stated as follows. We are given a set of n items with weights w_1, w_2,..w_i,..w_n, and m identical bins of capacity C each. The objective is to pack these items in the bins such that the total weight of items in the bins is maximized.

An instance of multiple subset sum problem is converted into an equivalent instance of channel assignment problem as follows. We construct a single collision domain network of $2 * n$ nodes where each node is equipped with 2 network cards. We now add a virtual link between nodes 1 and 2 with bandwidth requirement of w_1 which is the weight of the first item in the given multiple subset problem. We next add another virtual link between nodes 3 and 4 with bandwidth requirement of w_2, and so on. Next, we incorporate virtual links between nodes 2 and 3, nodes 4 and 5, and so on each with bandwidth requirement of C. We also add a link between nodes $2n$ back to 1. The construction is shown in Figure 3.1. The capacity of the channel is the same as the bin capacity C, and the number of channels is equal to $m + n$.

Let us now see what does a solution to this constructed problem looks like. First of all, each of the blue links with bandwidth requirement C has to be assigned over a dedicated

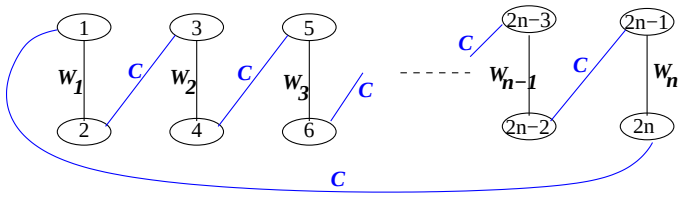


Figure 3.1: *Constructed graph for an instance of multiple subset sum problem.*

channel. Thus, the solution must use the remaining m channels to satisfy all the black links. Each blue link also uses two network interfaces – one on each of the two nodes it is incident upon. Thus, the solution must use the remaining single interface on each of the node to satisfy the black links bandwidth requirements.

Now, all the black links, say W_{x_1}, W_{x_2},.. ,W_{x_p} that the solution puts over one of the m channels must have a sum less than C. This means, that for the original multiple subset problem, all of x_1, x_2, ..., x_p can go the one bin. Similarly, all the items corresponding to virtual links scheduled over one of the m channels can go to one of the m bins. Thus, if the channel assignment problem were solvable in polynomial time, so would be the multiple subset sum problem. Since the multiple subset sum problem is NP-hard, the channel assignment problem is also NP-hard.

3.3 Centralized Channel Assignment and Routing Algorithms

To establish a baseline, we develop a greedy centralized algorithm to the channel assignment/routing problem in multi-channel WMNs [21]. The algorithm works for a general any-to-any traffic pattern. The algorithm first estimates the load imposed on each virtual link by each traffic flow in the given traffic matrix, and thus the total expected load on each virtual link. The channel assignment algorithm then visits all the virtual links in decreasing order of their expected loads. Upon visiting a particular virtual link, the algorithm greedily assigns it a channel that leads to minimum interference and contention with neighboring nodes in the interference zone whose WLAN interfaces have already been assigned to specific channels.

3.3.1 Neighbor Partitioning Scheme

One approach to the channel assignment problem is to start with one node, partition its neighbors into q groups and assign one group to each of its interfaces. Each of this node's neighbors, in turn, partitions its neighbors into q groups, while maintaining the grouping done by the first node as a constraint. This process is iteratively repeated until all nodes have partitioned their neighbors. Each group can then be bound to the least-used channel in the neighborhood. In general, this scheme requires a way to partition neighbors that results

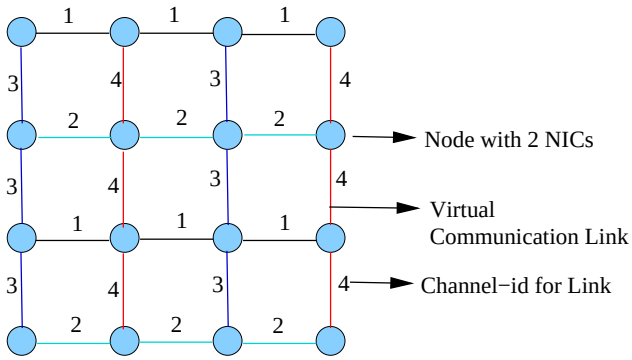


Figure 3.2: *Result of neighbor partitioning scheme for a grid-like wireless mesh network of 16 nodes. The channel assignment is based on the network topology information.*

in a uniform channel assignment across the network. For a grid network, this neighbor partitioning can be based on patterns such as shown in Figure 3.2. In this example, each node has 2 NICs, but the resulting network uses 4 channels. For a general network, partitioning of neighbors could be done more randomly.

While the above neighbor partitioning scheme indeed allows a network to use more channels than the number of interfaces per node, it does not take into account the traffic load on the virtual links between neighboring nodes. The scheme thus would work well, only if each virtual link in the network has the same traffic load. However, this does not hold true in most cases. In any given network, some of the links invariably end up carrying more traffic than the others. Conceptually links that need to support higher traffic load should be given more bandwidth than others. This means that these links should use a radio channel that is shared among a fewer number of nodes. A load-aware channel assignment distributes radio resource among nodes in a way that matches the spatial distribution of the traffic load.

3.3.2 Load-Aware Channel Assignment and Routing

We assume a virtual link exists between any pair of nodes that is within hearing range of each other. To maximize a network's overall goodput, the routing algorithm needs to route traffic to balance the load on the network's virtual links or simply links to avoid bottlenecks. However, the proposed wireless mesh network architecture offers one more degree of freedom, *modifying a virtual link's capacity* by assigning a radio channel to the link. This is possible because the capacity of a virtual link depends on the number of other links that are within its interference range and that are using the same radio channel.

Because routing depends on the virtual links' capacity, which is determined by channel assignment, and channel assignment depends on the virtual links' expected load, which is affected by routing, there is thus a circular dependency between radio channel assignment and packet routing. To break this circularity, we start with an initial estimation of the expected load on each virtual link without regard to the link capacity, and then iterate over

channel assignment and routing steps until the bandwidth allocated to each virtual link matches its expected load as closely as it can. More concretely, given a set of node pairs and the expected traffic load between each node pair, the routing algorithm devises the initial routes for the node pairs. Given these initial routes for the node pairs and thus the traffic load on each virtual link, the radio channel assignment algorithm assigns a radio channel to each interface, such that the amount of bandwidth made available to each virtual link is no less than its expected load. The new channel assignment is fed back to the routing step to arrive at more informed routing decisions, *i.e.* using actual link capacities based on current channel assignment. At the end of each iteration, if some of the link loads are more than their capacities, the algorithm goes back to find a better channel assignment using the link-loads from previous iteration, redo the routing, and compare the new link loads with new link capacities. This iterative process continues on until no further improvement is possible. Figure 3.3 depicts this process. Because it is not always possible to reach a feasible solution, our goal therefore is to reduce the difference between link capacities and their expected loads as much as possible.

In summary, the inputs to the combined channel assignment and routing algorithm are (1) an estimated traffic load for all communicating node pairs, (2) a wireless mesh network network topology, and (3) the number of 802.11 network interfaces available in each node and the number of non-overlapping radio channels. The outputs of this algorithm are (1) the channel bound to each 802.11 interface and (2) the set of paths for every pair of communicating node pairs in the wireless mesh network.

3.3.2.1 Initial Link Load Estimation

The combined channel assignment and routing algorithm, first derives a rough estimate of the expected link load. One possibility is to assume each link's capacity is an equal share of the bandwidth sum of all radio channels among all interfering links within the same neighborhood. Specifically, we assume the capacity of link l, C_l, to be

$$C_l = \frac{Q * C_Q}{L_l} \tag{3.1}$$

where Q is the number of available channels, C_Q is the capacity per channel, and L_l are the number of other virtual links within the interference range of l. The equation essentially divides the aggregated channel capacities among all interfering links, without regard to number of NICs per node. Based on these virtual link capacities, the routing algorithm determines the initial routes and thus the initial link loads.

A more accurate estimate of expected link load is based on the notion of link criticality [40]. To compute initial expected link loads, we assume perfect load balancing across all acceptable paths between each communicating node pair. Let's call the number of acceptable paths between a pair of nodes (s, d), $P(s, d)$, and the number of acceptable paths between (s, d) that passes a link l, $P_l(s, d)$. Then the *expected-load* on link l, ϕ_l, is calculated using the equation

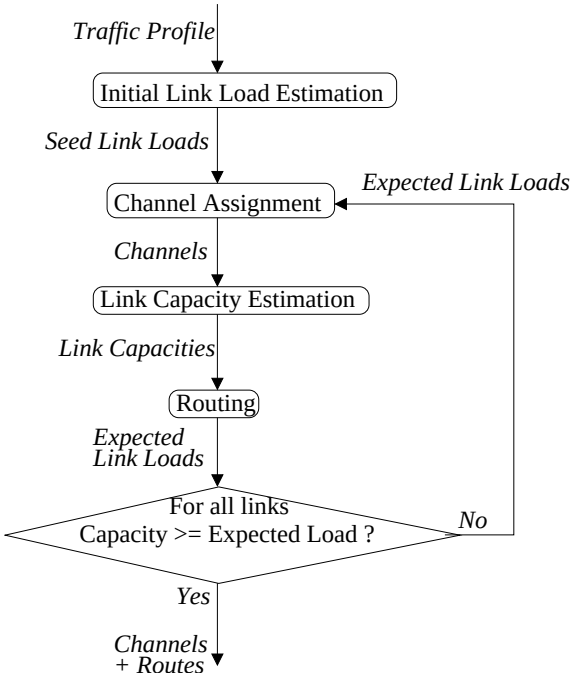


Figure 3.3: *Basic flowchart discussing the various aspects of traffic engineering in multi-channel mesh network architecture. At the beginning, a rough estimation of link load is used as the seed. The channel assignment algorithm governs the capacities of links. The routing algorithm uses these capacities to come up with routes, and in turn feeds more accurate expected loads on the links to the next iteration.*

$$\phi_l = \sum_{s,d} \frac{P_l(s,d)}{P(s,d)} * B(s,d) \tag{3.2}$$

where $B(s,d)$ is the estimated load between the node pair (s,d). This equation says that the initial expected load on a link is the sum of loads from all acceptable paths, across all possible node pairs, that pass through that link. Because of the assumption of uniform multi-path routing, the load that an acceptable path between a node pair is expected to carry is the node pair's expected load divided by the total number of acceptable paths between them. While the resulting estimates of this approach are not 100% accurate, they provide a good starting point to kick off the iterative refinement process.

3.3.2.2 Channel Assignment

Given the expected load on each virtual link, the goal of channel assignment algorithm is to assign channels to network interfaces such that the resulting available bandwidth on these interfaces is at least equal to their expected traffic load. We proved in Section 3.2 that the channel assignment problem is *NP-hard*. In this subsection, we present a greedy load-aware channel assignment algorithm, which works as follows. The virtual links in the wireless mesh network are visited in the decreasing order of link criticality, or the expected load on a link. When a virtual link is traversed, it is assigned a channel based on the current channel assignment of the incident nodes, called $node_1$ and $node_2$, respectively in the following. The channel list of a node refers to the set of channels assigned to its virtual links. Assuming there are q NICs per node, there are 3 possible cases -

1. Both $node_1$ and $node_2$ have fewer than q members in their channel list. In this case, we assign any channel that has the least degree of interference to the virtual link in question.

2. One of the nodes, say $node_1$, has q members in its channel list, and the other node's channel list has fewer than q members. In this case, we choose one of the channels in $node_1$'s channel list, assign it to the virtual link and add it to $node_2$'s channel list, if it is not already there. The channel in $node_1$'s channel list that minimizes the degree of interference for the virtual link is chosen.

3. Both $node_1$ and $node_2$ have q members in their channel list. If there are common channels shared by $node_1$ and $node_2$, we pick the common channel that minimizes the degree of interference and assign it to the virtual link. If no such common channel exists, then we pick a channel from $node_1$ and a channel from $node_2$, merge them into one channel, and assign this merged channel to the virtual link. In this case, all the other instances of the two channels being merged need to be renamed into the new channel as well, as shown in Figure 3.4.

By the *degree of interference*, we mean the sum of expected load from the virtual links in the interference region that are assigned to the same radio channel. As more virtual links

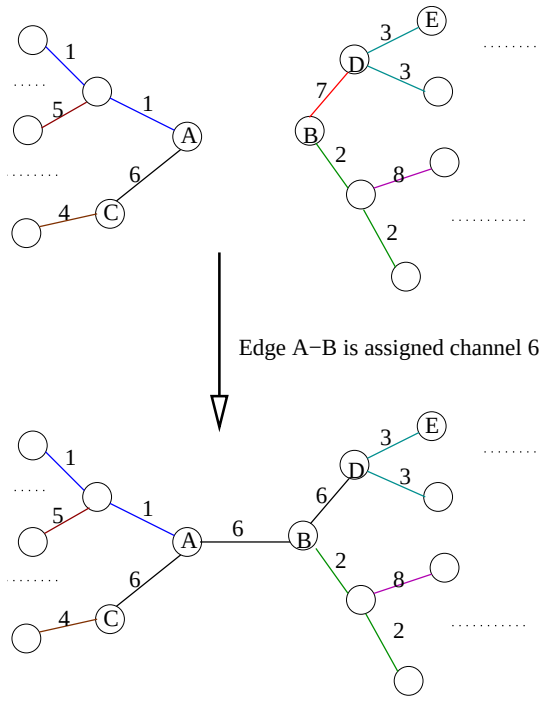


Figure 3.4: *Illustrative example to show the 3rd case of channel assignment. Node A's channel-list is [1,6], and node B's channel-list is [2,7]. Since they have non-intersecting sets of channels in use and each node has 2 NICs, link A-B needs to be assigned one of the channels from [1,2,6,7]. Based on resulting channel expected-loads, channel 6 is assigned to A-B, and channel 7 is renamed to channel 6.*

within an interference range tend to decrease the bandwidth share available to each of them, decreasing the degree of interference of a link increases its available bandwidth. By visiting the virtual link in the decreasing order of link criticality, more loaded links are more likely to be assigned to a channel with less interference, and thus given a higher capacity.

3.3.2.3 Link Capacity Estimation

To evaluate the effectiveness of a channel assignment algorithm, we need to calculate the capacity of each virtual link, and compare it against the the link's expected load. The portion of channel bandwidth available to a virtual link, or the link capacity, is determined by the number of other virtual links in its interference range that are also assigned to the same channel. Of course, the exact short-term instantaneous bandwidth available to each link depends on such complex system dynamics as capture effect, coherence period, physical obstacles, stray RF interferences, and distance. Our attempt here is to come up with an approximation of the long-term bandwidth share available to a virtual link. We approximate a virtual link's capacity by

$$bw_i = \frac{\phi_i}{\sum_{j \in Intf(i)} \phi_j} * C \tag{3.3}$$

where bw_i is the long-term bandwidth available to a virtual link i, ϕ_i is the expected load on link i, $Intf(i)$ is the set of other virtual links in the interference zone of link i, and C is the sustained radio channel capacity. The rationale of this formula is that when a channel is not overloaded, the bandwidth share available to a virtual link is approximately equal to its expected load weighted by the total expected load on the channel. The higher the expected load on a link, the more channel share it would get. The accuracy of this formula decreases as $\sum_{j \in Intf(i)} \phi_j$ approaches C.

3.3.2.4 Routing Algorithm

The load-aware channel assignment algorithm is not tied to any specific routing algorithm. It can work with different routing algorithms. For evaluation purposes, we explore two different routing algorithms – (1) shortest path routing, and (2) randomized multi-path routing. The shortest path routing is based on standard Bellman-Ford algorithm with minimum hop-count metric. The shortest path here refers to the shortest "feasible" path, *i.e.*, a path with sufficient available bandwidth and least hop-count. The multi-path routing algorithm attempts to achieve load-balancing by distributing the traffic between a pair of nodes among multiple available paths at run time. The exact set of paths between a communicating node pair are chosen randomly out of the set of available paths with sufficient bandwidth. Although traffic between a node pair is split across multiple paths in this case, packets associated with a network connection still follows a single path to avoid TCP reordering [41].

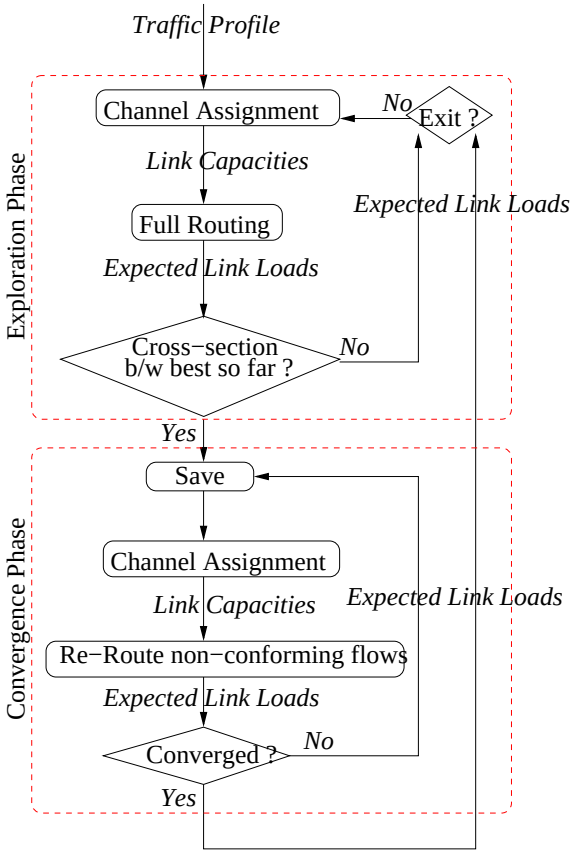


Figure 3.5: *Overall iterations. In the exploration phase, full routing is performed in every step to allow algorithm to explore new configurations. In the convergence phase, only non-conforming flows are rerouted to fine-tune specific configurations coming out of exploration phase.*

3.3.2.5 Putting It All Together

Figure 3.5 depicts the iterative process of the combined channel assignment and routing algorithm. At the very beginning, the initial link load is first estimated. Then the channel assignment step and the full routing step are iterated multiple times until the cross-section goodput of the resulting network converges. We call these iterations the *exploration phase*. Then we enter the *convergence phase*, in which channel assignment and routing are also iterated until the cross-section goodput of the resulting network converges. However, in this case only the node pairs that have not found a route with sufficient bandwidth to meet their traffic demands need to be re-routed. The *exploration phase* and *convergence phase* themselves are iterated until either all node pairs are successfully routed, or no better network configuration can be found.

3.4 Distributed Routing/Channel Assignment Algorithm

In this section, we present a distributed routing/channel assignment algorithm that utilizes only local topology and local traffic load information to perform channel assignment and route computation. This information is collected from a $(k+1)$-hop neighborhood, where k is the ratio between the interference and communication ranges, and is typically between 2 and 3.

3.4.1 Distributed Load-Balancing Routing

As most of the traffic on a WMN is directed to/from the wired network, each WMN node needs to discover a path to reach one or multiple wired gateway nodes. Logically, each wired gateway node is the root of a spanning tree, and each WMN node attempts to participate in one or multiple such spanning trees. These spanning trees are connected to each other through the wired network. When each WMN node joins multiple spanning trees, it can distribute its load among these trees and also use them as alternative routes when nodes or links fail. However, a WMN node may need additional wireless network interfaces to join multiple trees. In this dissertation, we restrict our focus on the case where each node is actively associated with only one of the trees and uses the other trees only for failure recovery. The techniques we develop are generic enough to be applied to the latter case when each node associates with multiple gateway nodes.

3.4.1.1 Routing Tree Construction

The basic tree construction process is similar to IEEE 802.1D's spanning tree formation algorithm [42] with two major differences – (a) the metric used by each WMN node to determine a parent is dynamic to achieve better load balancing, and (b) load-aware channel assignment technique is used to automatically form a fat-tree where more relay bandwidth is available on virtual links closer to the roots of the trees, i.e., wired gateways.

Assume a node X has already discovered a path to the wired network. It periodically, every T_a time units, broadcasts this reachability information to its one-hop neighbors using an ADVERTISE packet. Initially, only the gateway nodes can send out such advertisements because of direct connectivity to the wired network. Over time, intermediate WMN nodes that have a multi-hop path to one of the gateway nodes can also make such advertisements. The ADVERTISE packet that X sends out contains the "cost" of reaching the wired network through X. Upon receiving an advertisement, X's neighbor, say node Y, can decide to join X if Y does not have a path to the wired network, or the cost to reach the wired network through X is less than Y's current choice. To join node X, node Y sends a JOIN message to X. On receiving the JOIN message, X adds Y to its children list, and sends an ACCEPT message to Y with information about channel(s) and IP address to use for forwarding traffic from Y to X. In terms of the routing tree, X is now the *parent* of Y, and Y is one of the *children* of X. Finally, Y sends a LEAVE message to its previous parent node, say V. From

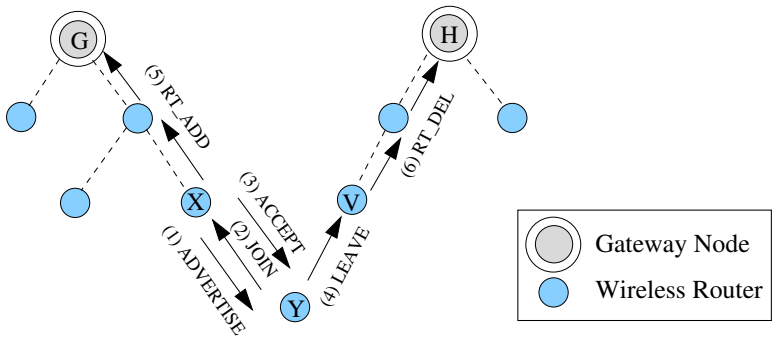


Figure 3.6: *A distributed route discovery/update protocol used to establish routes between multi-channel WMN nodes and wired gateways.*

this point on, Y also broadcasts ADVERTISE packets to its own one-hop neighbors to further extend the reachability tree. Fig 3.6 shows the message exchange sequence.

As a result of the exchange of JOIN/ACCEPT/LEAVE messages, the routing tables on the involved nodes are updated. First, the default routing entry of Y points to X as the next hop. All nodes in the tree from V upwards to the corresponding gateway node *delete* the forwarding entries pointing to Y and its children, if any. On the other hand, all nodes in the tree from X upwards to the gateway node *add* a forwarding entry for packets destined to Y and its children. To perform these route updates, the RT_ADD/RT_DEL messages are sent up to the root of the corresponding trees, as shown in Fig 3.6.

Delivery of some protocol messages such as JOIN, ACCEPT and LEAVE needs to be reliable for consistent network operations. These reliable connections are built as part of the *Neighbor Discovery Protocol*, wherein a new node broadcasts a HELLO message to its one-hop neighbors. Upon receiving this HELLO message from a new node, each of its neighbors establishes a reliable connection with the new node and also sends an ADVERTISE message to expedite the route discovery for the new node. A reliable connection is built on top of the UDP layer and is used for delivering all control messages that require reliability. The HELLO message itself can be lost, and is thus broadcasted multiple times to minimize the probability of message loss. The ADVERTISE message, in contrast, is sent as an unreliable broadcast packet for efficiency reasons.

3.4.1.2 Routing Metric

The “cost” metric carried in the ADVERTISE messages determines the final tree/forest structure. We explore three different cost metrics. First is the *hop count* between a WMN node and the gateway node associated with an ADVERTISE message. This metric enables a WMN node to reach the wired network using the minimum number of hops, but does nothing to balance network load. An advantage of using the hop-count metric is rapid convergence, as the minimum hopcount from a node to a wired network is determined by physical topology and is thus mostly static.

The second cost metric is the *gateway link capacity*, which indicates the residual capacity of the *uplink* that connects the root gateway of a tree to the wired network. Residual capacity of any link is determined by subtracting the current usage of the link from its overall capacity. In case the total bandwidth of a gateway's wireless links is smaller than its uplink, we take the wireless links' bandwidth as the gateway link capacity. The third cost metric is the *path capacity*, which represents the minimum residual bandwidth of the path that connects a WMN node to the wired network. Path capacity is more general than gateway link capacity because the former assumes that the bottleneck of a path can be any constituent link on the path, rather than always the gateway link. The *capacity of a wireless link* is approximated by subtracting the aggregate usage of the link's channel within its neighborhood from the channel's raw capacity which is assumed to be fixed within any collision domain.

The latter two metrics are dynamic, and can result in route flaps and a non-convergent network behavior. Route flaps occur when multiple nodes discover and switch to an underutilized path at the same time. Such simultaneous switching results in overloading of the originally underutilized path. This problem is similar to the route flapping problem observed on the wired Internet [43]. One can use similar measures as in BGP to dampen these route flaps reactively. We prevent route flaps by introducing a slight modification of the protocol. Since gateway is the only node aware of its latest link load, we propagate JOIN message up to the gateway. The gateway can now send an ACCEPT or a REJECT back to the newly joining node based on the gateway's latest residual link capacity. In addition, any intermediate node can also send a REJECT message to new requests if its capacity has decreased because other nodes switched to join its subtree. Additionally, the RT_ADD message also updates the current link usage on each hop of the path. This protocol modification effectively addresses the route flap problem at its source itself. As shown in the performance section, the bandwidth overhead introduced by this protocol change is fairly small.

3.4.2 Distributed Load-Aware Channel Assignment

The neighbor discovery and routing protocol in the previous subsection allows each WMN node to connect with its neighbors and identify a path to the wired network. We now discuss the mechanisms through which a WMN node can decide how to bind its interfaces to neighbors and how to assign radio channels to these interfaces without global coordination, as in the case of centralized algorithm.

3.4.2.1 Neighbor-Interface Binding

The key problem in the design of a *distributed* channel assignment algorithm is *channel dependency* among the nodes, which is illustrated in Fig 3.7. In this example, assume node D finds that the link D-E is heavily loaded and should be moved to a lightly loaded channel 7. As D only has 2 NICs, it can only operate on two channels simultaneously. To satisfy this constraint, link D-F also needs to change its channel. The same argument

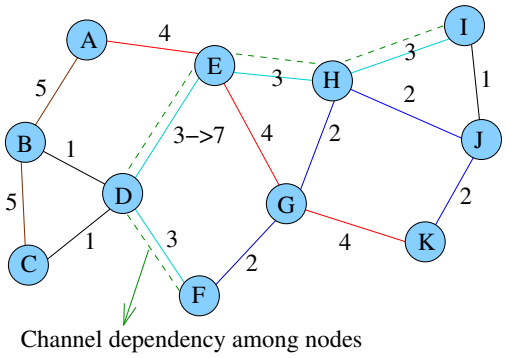


Figure 3.7: *This example shows how a change in channel assignment could lead to a series of channel re-assignments across the network because of the channel dependency problem.*

goes for node E, which needs to change the channel assignment for link E-H. This ripple effect further propagates to link H-I. Similar ripple effects would ensue if link A-E were to change its channel. In this case, link E-G and link G-K will need to change their channels as well. This channel dependency relationship among network nodes makes it difficult for an individual node to predict the effect of a local channel re-assignment decision.

To bound the impact of a change in channel assignment, we impose a restriction on the WMN nodes. Specifically, the set of NICs that a node uses to communicate with its parent node, termed *UP-NICs*, is disjoint from the set of NICs the node uses to communicate with its children nodes, called *DOWN-NICs*, as shown in Fig 3.8. Each WMN node is responsible for assigning channels to its DOWN-NICs. Each of the node's UP-NICs is associated with a unique DOWN-NIC of the parent node and is assigned the same channel as the parent's corresponding DOWN-NIC. This restriction effectively prevents channel dependencies from propagating from a node's parent to its children, and thus ensures that a node can assign/modify its DOWN-NICs' channel assignment without introducing ripple effects in the network. Because a gateway node does not have any parent, it uses all its NICs as DOWN-NICs. To increase the relay capability, each non-gateway node attempts to equally divide its NICs into UP-NICs and DOWN-NICs. However, a node farther from the gateway is assigned a lower priority in choosing channels, and thus may not get the required bandwidth on any single channel. In this case, the node can dedicate more NICs as DOWN-NICs to aggregate the leftover bandwidth from multiple channels.

3.4.2.2 Interface-Channel Assignment

Once the neighbor-to-interface mapping is determined, the final question is how to assign a channel to each of the NICs. The channel assignment of a WMN node's UP-NICs is the responsibility of its parent. To assign channels to a WMN node's DOWN-NICs, it needs to estimate the usage status of all the channels within its interference neighborhood. Each node therefore periodically exchanges its individual channel usage information as a CHNL_USAGE packet with all its $(k+1)$-hop neighbors, where k is the ratio of the interference range and the communication range. Because all the children and parent of a

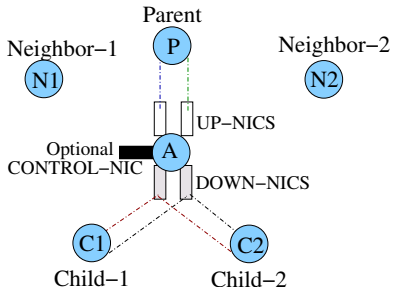


Figure 3.8: *Eliminating the channel dependency problem by separating the set of NICs used in each WMN node into UP-NICs and DOWN-NICs so that any channel assignment change in a WMN node's DOWN-NICs does not affect its UP-NICs.*

node, say A, can interfere with their own k hop neighbors, A's $(k+1)$-hop neighborhood includes all the nodes that can potentially interfere with A's communication. The aggregate traffic load of a particular channel is estimated by summing up the loads contributed by all the interfering neighbors that happen to use this channel. To account for the MAC-layer overhead such as contention, the *total load of a channel* is a weighted combination of the aggregated traffic load and the number of nodes using the channel.

Based on the per-channel total load information, a WMN node determines a set of channels that are least-used in its vicinity. As nodes higher up in the spanning trees need more relay bandwidth, they are given a higher priority in channel assignment. More specifically, the priority of a WMN node is equal to its hop distance from the gateway. When a WMN node performs channel assignment, it restricts its search to those channels that are not used by any of its interfering neighbors with a higher priority. The outcome of this priority mechanism is a *fat-tree* architecture where links higher up in the tree are given higher bandwidth.

Because traffic patterns and thus channel loads can evolve over time, the interface-to-channel mapping is adjusted periodically, every T_c time units. Within a *channel load-balancing* phase, a WMN node evaluates its current channel assignment based on the channel usage information it receives from neighboring nodes. As soon as the node finds a relatively less loaded channel after accounting for priority and its own usage of current channel, it moves one of its DOWN-NICs operating on a heavily-loaded channel to use the less-loaded channel, and sends a CHNL_CHANGE message with the new channel information to the affected child nodes, which modify the channels of their UP-NICs accordingly. The node also sends an updated channel usage map to its $(k+1)$-hop neighbors. This quick channel usage update strategy ensures that other nodes in the neighborhood do not migrate to the new channel because they assume (incorrectly) that it is still less loaded. The probability of race condition is further reduced by skewing the load-balancing phases among neighboring nodes.

In the case that a relatively less-loaded channel is unavailable for a NIC operating on a heavily-loaded channel, the WMN node performs child-interface load balancing. Here, it re-distributes its children among its DOWN-NICs such that the DOWN-NICs' channels

get more uniformly loaded.

3.4.3 Virtual Control Network

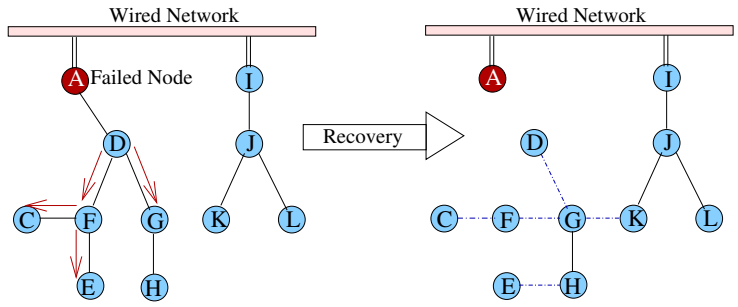
Unlike in a single-channel mesh network, nodes in a multi-channel WMN may not share any common channel with some of their physical neighbors. One simple option is to add a *CONTROL-NIC* on each node, tune it to a common channel, and route all control traffic such as ADVERTISE messages over this control network. We refer to this approach as the *physical control network* approach. The additional hardware interface can be saved by forming a *virtual control network* over the same multi-channel mesh network to exchange control packets.

A WMN node needs to communicate each control message to its c-hop *physical* neighbors, where c depends on the message's type and can range from 1 to $(k+1)$, where k is again the ratio of interference and communication ranges. Since there is always a path between a WMN node and its physical neighbors, a control message can be delivered through one or multiple hops on the mesh network. For efficiency reasons, the broadcast control messages are delivered using *IP multicast*. Essentially the idea here is to implement layer-2 communication using layer-3 routing so as to eliminate the dedicated control interface on each node. With the use of virtual control network, a *new* node needs to scan all channels for broadcasting HELLO messages during the neighbor discovery phase. Channel scanning process can be done in 5 to 10 sec. When two nodes scan the channels simultaneously, each node only uses its UP-NIC to perform the scanning, while keeping the DOWN-NIC at a fixed channel. This ensures that the neighboring nodes can eventually discover each other.

3.4.4 Failure Recovery

When a node fails, nodes in its subtree lose their connectivity to the wired network. Hyacinth reorganizes the network to bypass the failed node and restore the connectivity. To accommodate node failures, each WMN node remembers *alternative* advertisements it has received from all other potential parent nodes. Upon detection of a parent-node failure, each of its child nodes sends a JOIN message to a "backup" parent node, and re-establishes its connectivity with the wired network. This scheme allows fast recovery from a node failure without committing any additional physical radio resources.

Not all failures can be handled locally, as shown in Fig 3.9(a). Here, when node A fails, its child node D does not have a ready-backup parent that can re-connect it to the wired network. To recover from such failures, node D sends a FAILURE message to its immediate children F and G, asks them to perform their own failure recovery, and forces itself to go back to the channel-scanning mode. The FAILURE message contains the list of failed parent nodes, in this case, A and D. Each child node in turn attempts to associate with its respective backup parents. However, in this process the child nodes actively avoid known failed nodes, in this case A and D. If a child cannot find any usable backup parent, it recursively broadcasts a FAILURE message to its children after adding its own name to the failed-nodes-list included in the FAILURE message, and goes to channel-scanning



(a) Failure message after node A fails (b) New connectivity after recovery

Figure 3.9: *This example network illustrates the distributed failure recovery protocol used in Hyacinth.*

mode. In Fig 3.9(a), node G performs local recovery, while node F relays the FAILURE message to its own children C and E. Fig 3.9(b) shows the final network connectivity after the recovery process is completed. A failure recovery is eventually followed by the usual periodic traffic profiling, and channel adjustments to re-balance the network load across different channels.

Another possibility is for each WMN node to actively maintain simultaneous connectivity with multiple parent nodes. Compared with the single-parent scheme described above, this scheme incurs zero failure recovery time but requires at least two UP-NICs on each node.

3.5 Performance Evaluation

We studied the performance gains of the proposed multi-channel WMN architecture and the effectiveness of the proposed channel assignment and routing algorithms through extensive ns-2 simulations. We modified ns-2 to support multiple wireless cards on each wireless node and to support dynamic channel assignment. The evaluation metric for most experiments is *cross-section goodput*, which is defined as the sum of all useful bandwidth between traffic aggregation nodes in a WMN and their corresponding gateway nodes (eqn 2.1). The following are the default settings for the simulations. Each node is equipped with 2 NICs, and the number of physical channels is 12. For effective multi-hopping, RTS/CTS mechanism is enabled. The ratio between the interference range and the communication range is set to 2. Channel load-balancing period T_c of a node is set to 1 minute, while the channel-usage and routes advertisement frequency, is set to once every 30 seconds (T_a).

3.5.1 Improvements due to Multi-channel Mesh Networking

3.5.1.1 Spanning Tree Topology

We measured the throughput improvements achieved by Hyacinth's multi-channel mesh networking architecture using different channel assignment algorithms. Ten different 60-node network topologies are generated, each randomly sampled from a 9x9 square grid network. Based on the topology and location, each node could communicate with up to 4 neighbors. Four of these 60 nodes were designated as the gateway nodes and connected to the wired network. For each topology, 30 nodes were chosen at random to generate traffic flows. Each traffic flow represents an aggregate of traffic streams from multiple users. The average bandwidth for each flow is chosen at random between 0 and 3 Mbps.

To drive the network to saturation, the bandwidth of all the flows is proportionally varied until the network can only route 80% of the aggregate input traffic. The *relative* performance of different algorithms does not change for other values of saturation threshold, e.g. 100% at which we ensure that each flow has to be assigned its full required bandwidth. The same is true when the saturation threshold is made per-flow to ensure fairness across different flows. For example, one can ensure that each flow has to be assigned at least a certain percentage of its traffic requirement. For brevity, we only show the overall cross-section goodput for all the graphs.

The results in Fig 3.10 show that even with identical channel assignment scheme [44], deploying 2 NICs on each node improves the network goodput by a factor of 2 compared with conventional single-channel network. With the proposed distributed channel assignment algorithm the network throughput becomes 6 to 7 times that of single-channel network. Intuitively, Hyacinth's channel assignment algorithm breaks a collision domain in a single-channel network into multiple collision domains each operating in a different frequency range. This division of collision domain across different frequency channels is the key reason for the nonlinear goodput improvement (6-7 times) with respect to the increase in the number of NICs (from 1 to 2). Moreover, the interference among adjacent hops of an individual path and among neighboring paths is greatly reduced.

The first distributed channel assignment scheme, called *physical control network*, uses a dedicated control channel for communicating all control traffic. This requires an additional WLAN interface on each node specifically for control traffic. The second distributed channel assignment scheme, called *virtual control network*, multiplexes the control traffic over data NICs thereby reducing the per-node hardware cost. The fact that these two schemes have comparable performance suggests that control traffic introduces minimal overhead when multiplexed on the main data channels. Finally, the centralized channel assignment/routing algorithm does not perform much better than the distributed versions; this shows that the performance loss due to distribution of intelligence is very small.

An alternate design for a multi-channel mesh networking [45] is to equip each node with a single interface and operate the sub-network rooted at each gateway at a different channel [45]. Logically, this should reduce the contention among nodes and thus improve

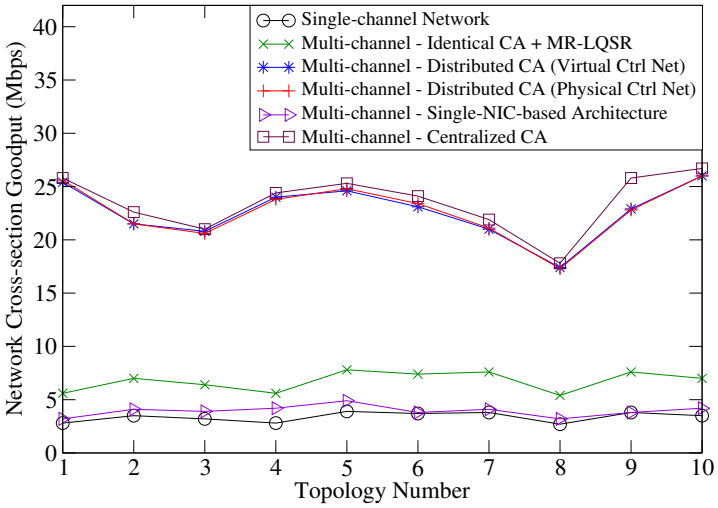


Figure 3.10: *Network cross-section goodput of 10 different 60-node topologies randomly sampled from a 9x9 square grid. With just 2 interfaces on each node, the distributed channel assignment improves the network throughput 6 to 7 times as compared with a single-channel network of the same topology.*

the network goodput. Surprisingly, this scheme does not give much throughput improvement over a single-channel mesh network as shown in Fig 3.10. The fact that only a single channel is used *within* a tree means that there is still heavy collision and interference on the wireless links around each gateway, which is most likely where the bottleneck is. In contrast, a true multi-channel mesh network architecture can split the wireless links around each gateway into as many collision domains as possible, thus delivering higher effective bandwidth.

We next simulated a 64-node network placed in an 8x8 grid with 4 uniformly distributed gateway nodes connected to the wired network. The remaining 60 nodes were (randomly) divided into 5 different popularity sets of equal sizes. A node's popularity determines the size of its user base and in turn the number of new HTTP connections generated every second from the node. The simulations were done using the PackMime extension of ns-2 that decides the request arrival pattern and response sizes using models discussed in [46]. Based on the node popularity, the average rate of new HTTP requests for the node was 0, x, 2x, 3x, or 4x, where x is the traffic intensity for the experiment. Fig 3.11 shows the response time observed by web users on this simulated network. The observed delay in the single-channel mesh network increases rapidly with the traffic intensity and eventually limits the number of users it can support. With just 2 NICs on each node, the multi-channel mesh network reduces the HTTP response time substantially. Additionally, at saturation the multi-channel WMN can support over 4 times as much web traffic as compared with the single-channel WMN, and consequently a much larger user base.

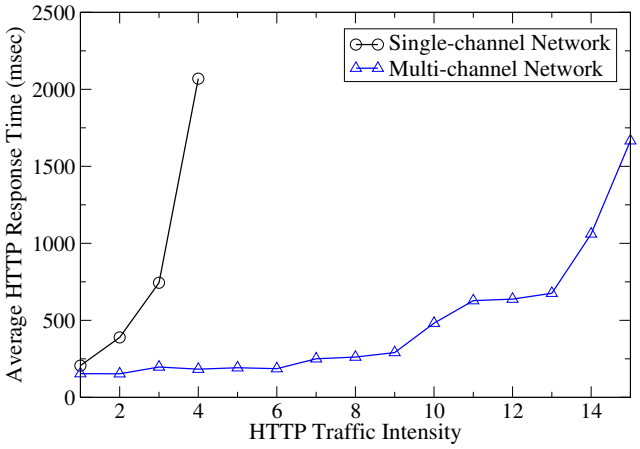


Figure 3.11: *Web browsing (HTTP) response time as seen by users on a 64-node WMN. The multi-channel WMN architecture can drastically reduce the HTTP response time, as well as increase the number of users a network can support.*

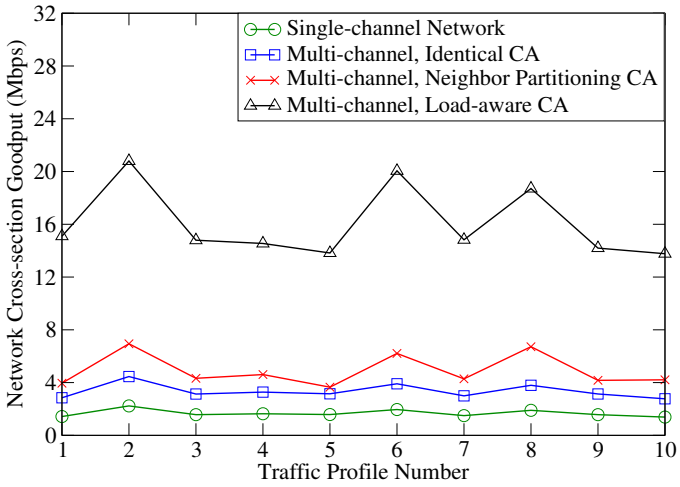


Figure 3.12: *The network cross-sectional goodput for 20 randomly chosen pairs of ingress-egress nodes with shortest-path routing. The figures show that even with the simple neighbor partitioning approach, there is substantial improvement in network cross-sectional goodput by use of just 2 NICs per node. Using the Load-aware channel assignment, however, yield the full potential of multi-channel wireless networks. The channel assignment from neighbor partitioning algorithm is the one corresponding to Figure 3.2.*

3.5.1.2 General Topology

The distributed algorithm optimizes the network for the common case when all traffic is directed to/from the gateway nodes. The centralized algorithm however works for arbitrary any-to-any traffic. Figure 3.12 presents the cross-section goodput of a 100-node square-grid

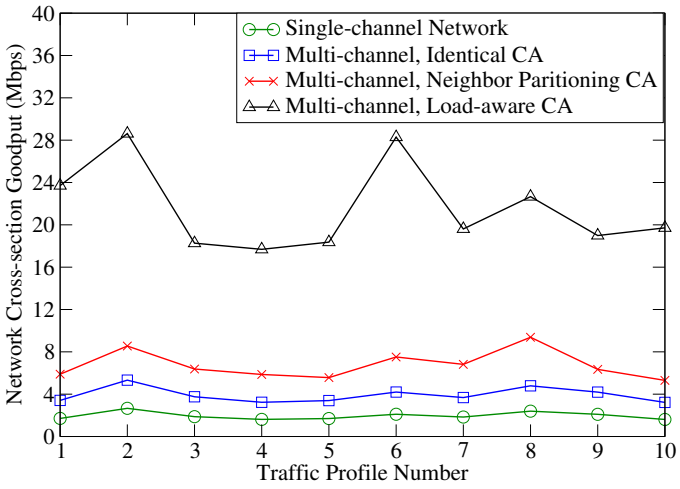


Figure 3.13: *Network cross-sectional goodput with randomized load-balanced routing. This figure demonstrates the adaptability of channel assignment to link loads imposed by different routing schemes.*

network for various traffic profiles each containing 20 pairs of randomly chosen ingress-egress nodes. The cross-section goodput here is defined as the sum of useful bandwidth assigned between all communicating ingress-egress node pairs. For each profile, the amount of traffic between each ingress-egress node pair was chosen at random between 0 and 3 Mbps. Depending on its position, each node could communicate with up to 4 neighbors. The routing algorithm for these experiments was the shortest-path routing.

The graphs show the cross-section goodput made available for single-channel network and for 12-channel/2-nic-per-node network with different channel assignment schemes. Compared with conventional single-channel wireless mesh network architecture, the neighbor partitioning scheme (as shown in Figure 3.2) achieves between 2.5 and 3.5 times as much improvement in cross-section goodput. In contrast, load-aware channel assignment can achieve over 8 times improvement in cross-section goodput with just 2 NICs per node. Intuitively, equipping each wireless mesh network node with multiple interfaces allows the network to use several radio channels simultaneously. This breaks each collision domain into several collision domains operating in a different frequency range. A collision domain is further sub-divided spatially when a given ingress-egress node pair takes a different path to route the traffic. Again, this division of collision domain across frequency and spatial domain is the key reason for the nonlinear goodput improvement with respect to the increase in the number of NICs.

Figure 3.13 shows the same performance comparison when the routing algorithm is changed to randomized multi-path routing. Because we do not perform any explicit load balancing in multi-path routing scheme, the performance improvement when going from single-path routing to multiple-path routing is only marginal. This is true for both the

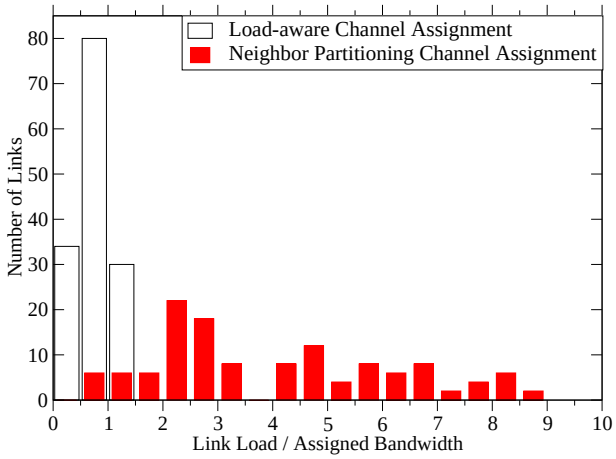


Figure 3.14: *Ratio of load imposed by routing algorithm and bandwidth assigned by the channel assignment algorithm for all links in the network. A ratio close to 1 for all the links in load-aware channel assignment case, implies assigned bandwidth closely matches the imposed load.*

single-channel case and multiple-channel case. However, the goodput gain of the multi-channel network architecture with proper channel assignment algorithms over the conventional single-channel architecture does not seem to depend on a particular routing algorithm. This adaptability of the channel assignment algorithm enables one to choose a routing scheme appropriate to the deployment scenario. The small improvement achieved with use of randomization-based multi-path routing is because of better load-balancing of the network. With the use of a more explicit load-balanced routing, the network performance should improve even further. We verified the cross-section goodput assigned by the various algorithms using ns-2 simulations. For brevity, we only show the overall cross-section goodput for all the graphs.

Figure 3.14 demonstrates the effectiveness of the channel assignment done by load-aware channel assignment scheme. For each link in the network, the ratio of load imposed by the routing algorithm and the bandwidth assigned by the channel assignment algorithm was measured. A ratio close to 1 indicates that more bandwidth is allocated to links that require more bandwidth. We observe that although the link load imposed by routing varied anywhere from 0 to 3.9 Mbps across network links, the ratio is close to (or less than) 1 for the load-aware channel assignment scheme. Achieving this distribution of channel resource among the nodes to match the spatial distribution of traffic load is the key to good performance of the scheme. For the neighbor partitioning scheme, most of the links are overloaded resulting in the variation of ratio from 0.5 to 8.9, the reason is that the latter performs a load-insensitive assignment of channels. The histogram for Identical channel assignment scheme (not shown) is similar in nature to the Neighbor partitioning approach.

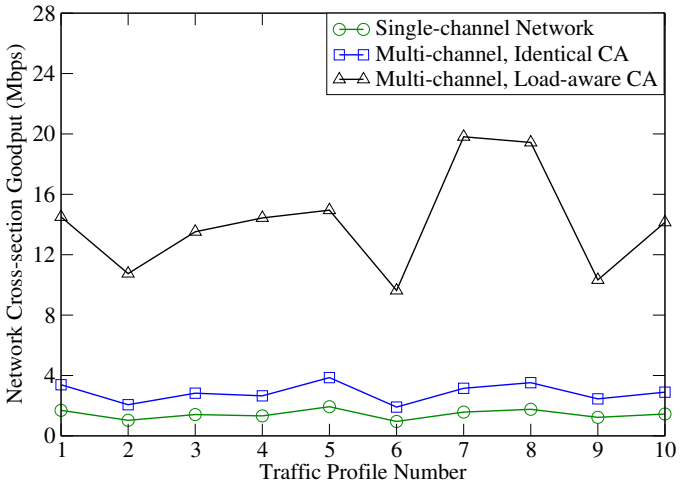


Figure 3.15: *Comparison of multi-channel network against single-channel network for MIT Roofnet topology [1].*

We also experimented with other network topologies. Figure 3.15 shows the performance comparison of the 29-node MIT Roofnet network [1] simulated in ns-2. The data for graph connectivity is based on signal-strength numbers from the testbed. Each point in the graph corresponds to a randomly generated traffic profile of 15 ingress-egress node pairs. The 8+ times improvement in network performance demonstrates the usefulness of multi-channel architecture for real networks. We observed similar improvements for other topologies - hexagonal grid, and incomplete mesh. The performance improvement using neighbor partitioning scheme, however, depends on the topology. A more generic way to partition the neighbors is needed in the latter scheme to handle general mesh networks.

In Figure 3.16, we varied the number of ingress-egress pairs in the 10x10 network (each node equipped with 2 NICs) while keeping the aggregated offered load to be the same. As more ingress-egress pairs are introduced, the traffic requirement is more distributed across the network leading to an overall increase in network utilization. The load-aware scheme adapts the channel assignment to these different sets of traffic requirements maintaining the performance improvements over single-channel network.

3.5.2 Impact of System Parameters

3.5.2.1 Number of WLAN Interfaces and Radio Channels

The number of non-overlapped radio channels is 3 for 802.11b/g and 12-25 for 802.11a. Fig 3.17 shows the effects of varying the number of radio channels on the network goodput. These experiments were conducted on a 64-node grid network with 4 gateway nodes uniformly placed across the network. The traffic was generated by 30 randomly chosen wireless mesh nodes, at an average rate chosen randomly between 0 and 3 Mbps for each node. We also vary the number of NICs per node to evaluate the impact of number of NICs

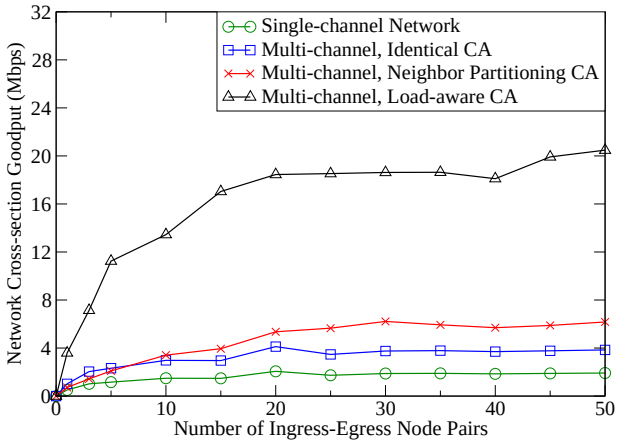


Figure 3.16: *Impact of varying the number of ingress-egress pairs on goodput improvements.*

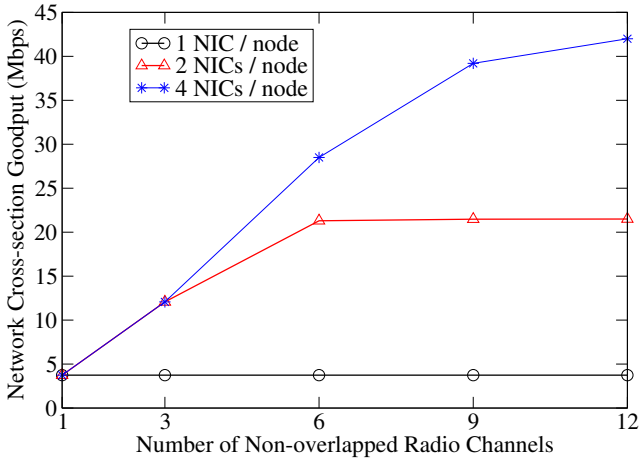


Figure 3.17: *Effects of varying the number of WLAN interfaces per node and number of non-overlapped radio channels.*

on the utilization efficiency of the channels. The network goodput increases monotonically with the number of non-overlapped channels when the the number of NICs per node is kept constant, because a collision domain can be broken into more non-interfering collision domains. When each node has 2 NICs, the network goodput saturates at about 6 channels. When the number of NICs on each node is increased to 4, the network can use up to 12 channels before its performance starts to saturate.

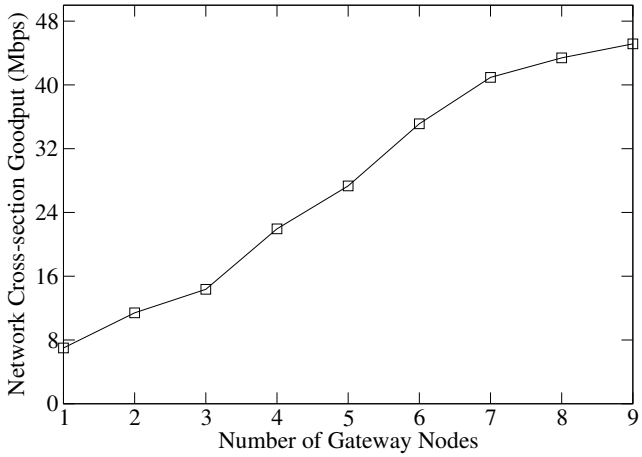


Figure 3.18: *The network goodput increases with the number of gateway nodes in the network. This means that the proposed channel assignment and routing algorithm can effectively reconfigure the WMN to leverage additional bandwidth made available by additional gateways.*

3.5.2.2 Number of Gateway Nodes

A major cost component of a WMN is its gateway nodes to the wired Internet [7]. The proposed algorithms attempt to make the best of the available gateways to increase the network goodput, as shown in Fig 3.18. In these experiments, the number of gateway nodes was increased from one to nine in a 9x9 grid network. In the final configuration, the 9 gateway nodes were uniformly distributed across the network. Each additional gateway node improves the network goodput because it adds more capacity to relay traffic from the user nodes to the wired network, and the proposed channel assignment and routing algorithm can effectively reconfigure the WMN to leverage additional bandwidth made available by additional gateways.

3.5.2.3 Placement of Gateway Nodes

Fig 3.19 shows the effect of placement of gateway nodes on the network goodput. As expected, the best performing configuration is the uniform placement of gateways as it makes it easier to spread the traffic load among the gateways. However, even when gateway nodes are concentrated in one place, the proposed channel assignment and routing algorithm can still reap most of the performance benefits. Concentration of gateway nodes is desirable because it simplifies installation (wiring) and management of gateway nodes, reducing the overall cost of WMN.

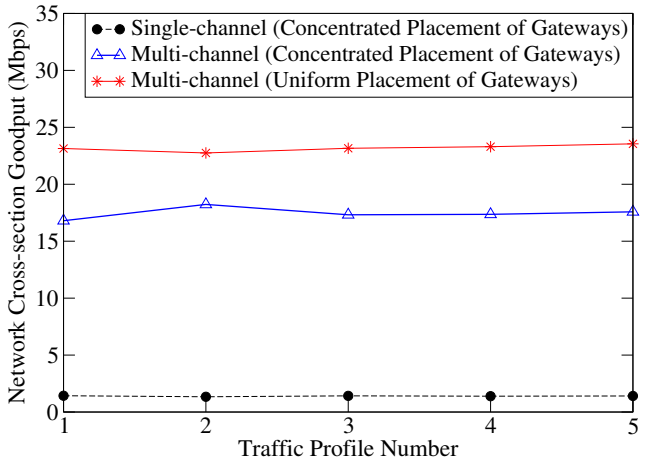


Figure 3.19: *Effect of placement of gateway nodes on the network goodput. Even when gateway nodes are concentrated, the proposed algorithms can still adapt the channel assignment and routes to maximize the network goodput.*

3.5.3 Impact of Algorithm Parameters

3.5.3.1 Channel Selection Criterion

To select a channel for an interface, a node needs to estimate the load on each of the available physical channels. We compare three different criterion used in selecting a channel. The first criterion uses the number of interfering nodes sharing a channel. The second criterion estimates a channel's load by adding up the channel usage of each of the interfering nodes. The third criterion takes into account a channel's load as well as whether it is being used by nodes closer to some gateways. By reducing the extent of congestion and collision for channels used by nodes closer to a gateway, this metric provides higher capacity to links closer to the root of a tree, thus enabling a fat tree architecture.

As shown in Fig 3.20, using the measured channel usage as the channel selection criterion gives substantial performance improvement over the criterion based only on the number of nodes sharing a channel, because measured channel usage is a more accurate way to reflect the load of a channel. Surprisingly, adding channel prioritization does not help at all. The reason is that channels used by links closer to the gateway nodes are typically more loaded, and as a result tend to be avoided by nearby descendant nodes that select channels based only on the measured channel usage criterion. This automatically assigns more bandwidth to links closer to the gateway nodes, thus forming a fat-tree.

Fig 3.21 shows the effectiveness of using measured channel usage as the channel selection criterion to allocate capacities to links in proportion of their bandwidth requirements. Before channel load balancing, a large fraction of network links have a high packet drop rate in either the interface queue or over the air. Channel load balancing puts heavily loaded links onto different channels, substantially reducing these packet-drops.

The current Hyacinth prototype uses both measured channel usage and contention

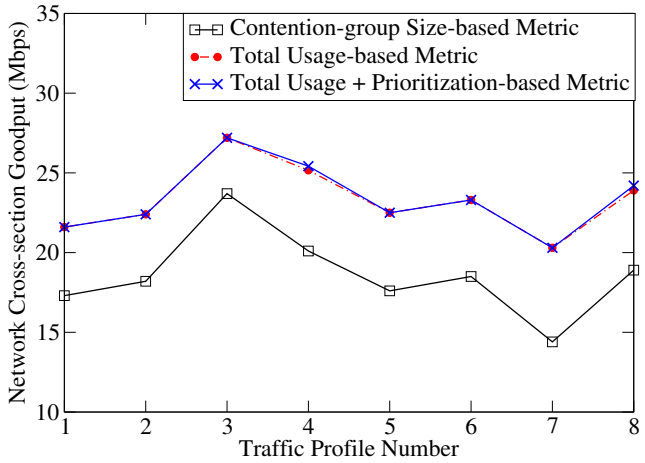


Figure 3.20: *Performance comparison among three channel selection criteria. Measured channel usage is a more accurate way to estimate the load of a channel.*

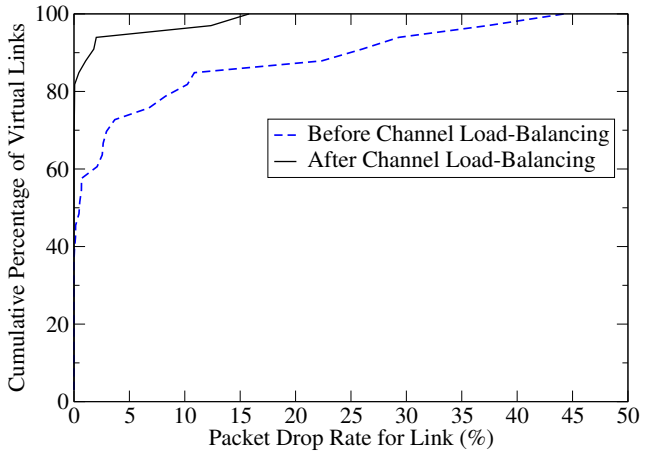


Figure 3.21: *Effectiveness of usage-based channel selection in dividing the interference zone across different frequencies. Packet drop rates are decreased because of reduced contention among links as well as allocation of higher capacity to more loaded links.*

group size in channel selection: If the channel usages of two channels differ by more than 10%, the lightly loaded one is chosen; otherwise, the channel with a smaller contention group size is chosen.

3.5.3.2 Routing Metric

We compare the impact of various routing metrics on the overall network performance for skewed traffic profiles. These experiments were conducted over a 64-node grid network with 4 uniformly placed gateway nodes. For each traffic profile, 20 different traffic aggregation devices were chosen in a *skewed* manner, specifically closer to two of the gateway nodes. Fig 3.22 shows the results. As expected, *shortest path routing* does not utilize the

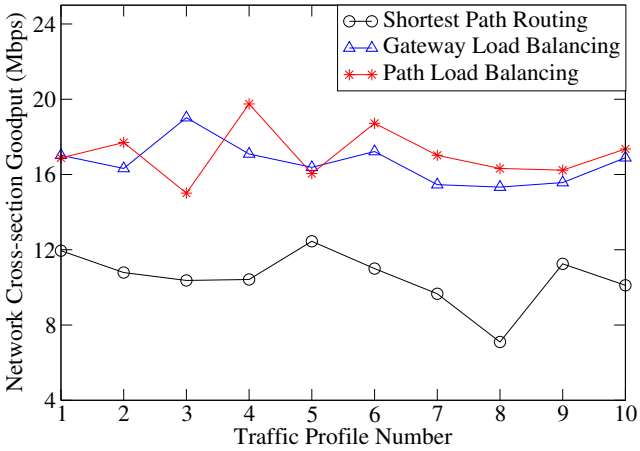


Figure 3.22: *Performance comparison among load balancing routing metrics in the presence of skewed traffic. Gateway load balancing only looks at the available bandwidth on the gateway nodes, while path load balancing considers end-to-end available bandwidth between a WMN node and a gateway node.*

gateways' bandwidth effectively. *Gateway load balancing* considers the available bandwidth on each of the gateway nodes while choosing paths, and *path load balancing* estimates the end-to-end available bandwidth between a WMN node and a gateway node when choosing the routing path. Performance of path load balancing is only slightly better than that of gateway load balancing, suggesting that gateways are the main bottlenecks. In a real-world network, even intermediate wireless links could form bottlenecks due to interference from other radio sources, and path load balancing should find more optimal paths than gateway load-balancing.

3.5.4 Traffic Adaptation and Protocol Complexity

In Hyacinth, the network nodes continuously monitor the loads on their interfaces, exchange channel usage information every 30 sec, and make load-balancing decisions every 60 sec. The network goodput should improve as nodes modify their channel assignments and routing. Fig 3.23 shows how a 64-node Hyacinth network adapts to a change in traffic loads. In this case, one traffic profile of 30 aggregated flows changes to another at time 600 sec, and the network converges to a new configuration within 2 load-balancing periods at time 723 sec.

Fig 3.24 shows the impact on the convergence time of the load-balancing period. For lower load-balancing periods, the network converges within 2 load-balancing periods. For higher load-balancing periods, it takes 1.5 rounds on the average to converge. Different load-balancing periods also incur different bandwidth overheads. This overhead is less than 15 Kbps even with very frequent load-balancing, At the default 60-second load-balancing period for our experiments, the protocol overhead is 5 Kbps, which is about 0.025 % of the maximum network cross-section goodput.

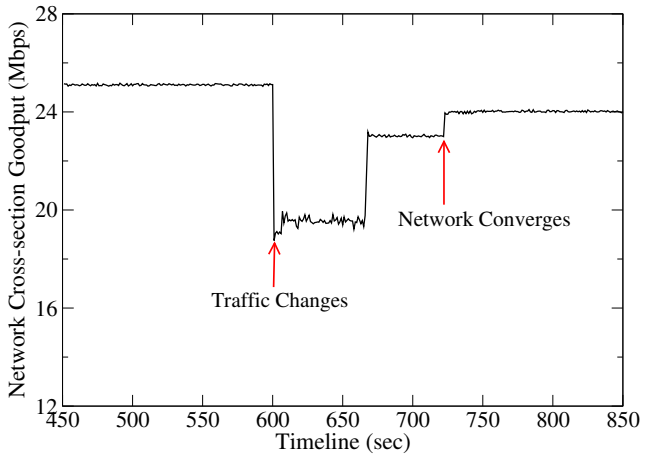


Figure 3.23: *Responsiveness of a Hyacinth network's adaptation to changes in traffic load distribution. The traffic profile switches from one set of 30 aggregated flows to another at time 600 seconds. The network converges to a new configuration in 2 load-balancing periods, each of which is 60 seconds.*

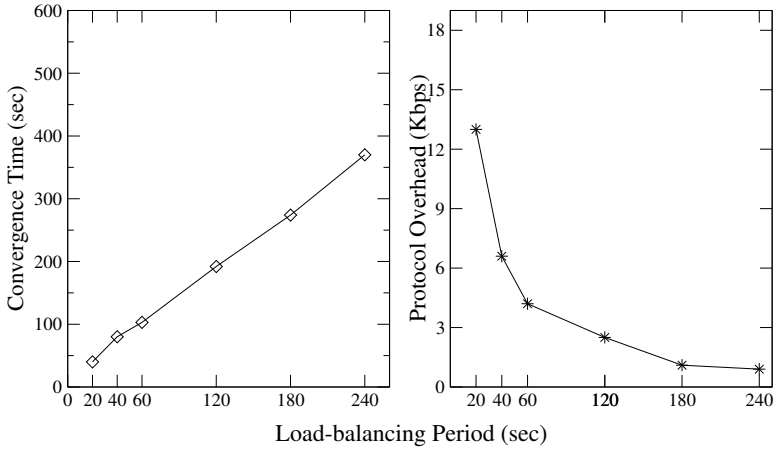


Figure 3.24: *Impact of load-balancing period on network convergence time against change in traffic load, and the corresponding bandwidth overhead. The network typically converges in less than 2 load-balancing periods, because the load-balancing periods for different nodes are de-synchronized. The bandwidth overhead due to load balancing actions is relatively low even for a small load-balancing period of 20 seconds, and reduces logarithmically with increasing load-balancing period.*

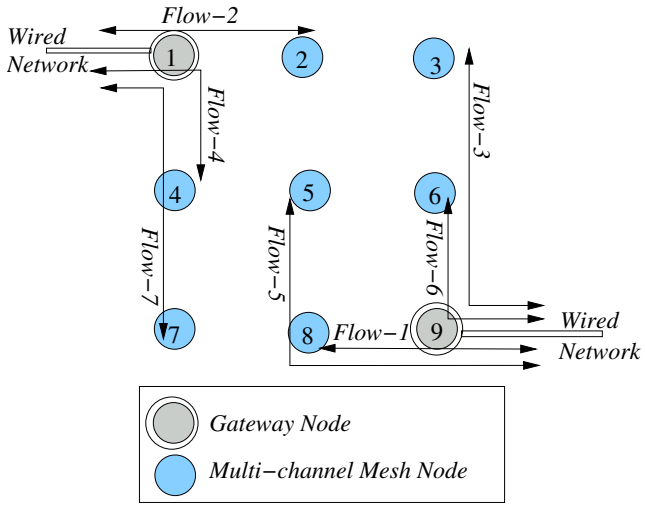


Figure 3.25: *Physical topology of the 9-node Hyacinth prototype. Each node is equipped with 2 802.11a PCI NICs whose channels are tuned dynamically by the channel/route daemon. Node 1 and Node 9 are connected to the wired network providing access to departmental servers and the Internet.*

3.6 Prototype Performance Evaluation

To demonstrate the feasibility of the proposed multi-channel WMN architecture, we built a 9-node Hyacinth prototype. The details of the prototype are discussed in Chapter 6. In this section, we report the results obtained from a performance study conducted on it.

Each node in the current Hyacinth prototype is a standard desktop PC running Linux 2.4.26 equipped with two different network interfaces – an Orinoco 802.11a/b/g PCI card and a Netgear 802.11a/b/g PCI card. Both cards operate in the 802.11a ad-hoc mode. Although the 802.11 interfaces mounted on the same machine operate on non-overlapped channels, they still interfere with one another [44]. We believe this interference arises from radiation leakage because of imperfect channel-filter hardware in commodity cards. We reduce this interference by using PCI cards equipped with external antennas (separated by 2 feet distance) and no internal antenna. Additionally, the channels assigned to the two cards mounted on the same machine are at least one channel apart from each other.

Fig 3.25 shows the topology of the 9-node Hyacinth prototype we built. The nine nodes are placed in an area of size approximately 20m x 10m spanning two lab rooms with two gateway nodes, Node 1 and Node 9, connected to the department wired network. The transmit power on each node is reduced to 1 mW to limit interference zones of individual nodes. For evaluation purposes, the prototype can be run in two different modes – single-channel mode, and multi-channel mode. The single-channel mode only uses one of the cards to form the mesh network.

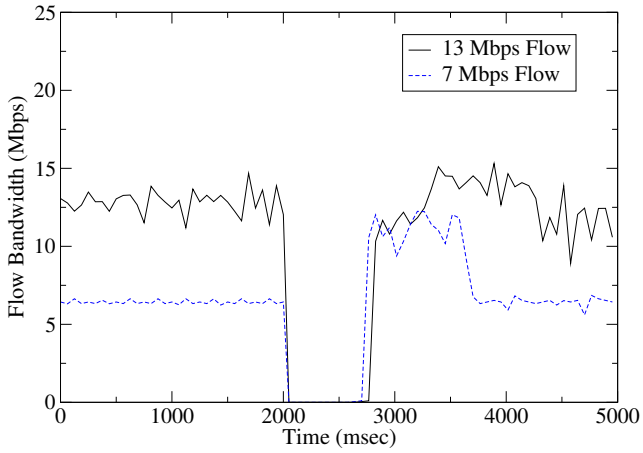


Figure 3.26: *Failure recovery time for Node 3 in Fig 3.25, when Node 6 fails. Failure occurs at time 2000 msec, and network recovers within 700 msec.*

3.6.1 FTP bandwidth

We measured the performance of FTP flows (shown in Fig. 3.25) that download data from the department server simultaneously. The aggregate performance of the flows in the multi-channel operation mode is 55.58 Mbps, which is about 5 times the aggregate throughput in the single-channel operation mode (11.32 Mbps). Measurements of upload FTP traffic showed similar performance gains. These results match closely with those obtained from an ns-2 simulation of the prototype, and thus validate our previous simulation results. The throughput improvement over single-channel case is limited to 5 times as opposed to 6 or 7 times, because of the small size of the prototype. A larger network should provide higher throughput gains with multi-channel mesh networking.

3.6.2 Failover Latency

We evaluated the failover aspect of the proposed architecture using the prototype. Fig 3.26 shows how the bandwidth of network flows evolves over time when a node in the Hyacinth prototype fails. In both the experiments, Node 6 was made to fail, and Node 3 failed over to Node 2 as its new parent. The period of time during which the network flows' bandwidth drops to zero represents the failure recovery time. The failure recovery time is between 600 to 700 msec. Out of these 150 msec is the failure detection time, which was done by exchanging HELLO packets between Node 3 and Node 6. The propagation time for the route-change request is about 1 msec. Most of the remaining time goes into changing the routing tables.

For failover, Node 3 has to establish a new link-layer connection with Node 2. Due to driver implementation problem, the channel switching latency was extremely high. To remove this overhead from the measurement, we kept an additional interface of Node 3 tuned on the same channel as Node 2. With an appropriate driver implementation, channel

switching should only add about 50-100 msec to overall failover latency [47].

3.7 Conclusions

Despite many technological advances at the physical layer, limited bandwidth remains a pressing issue for wireless networks, especially when compared with their wired counterpart. The bandwidth problem is even more serious for multi-hop wireless mesh networks (WMNs) due to interference between successive hops on the same path as well as that between neighboring paths. As a result, conventional single-channel WMNs cannot adequately support the bandwidth requirements of last-mile wireless broadband access networks, let alone a campus backbone that completely replaces the wired Ethernet. In this chapter, we described a novel multi-radio WMN architecture that effectively addresses this bandwidth problem by fully exploiting non-overlapped radio channels that the IEEE 802.11 standards make available.

In particular, we discussed two fundamental design issues in the proposed WMN architecture where each network is equipped with multiple radio interfaces. First, which of the available non-overlapped radio channels should be assigned to each 802.11 interface in the WMN? For two nodes to communicate with each other, their interfaces need to be assigned to a common channel. However, as more interfaces within an interference range are assigned to the same radio channel, the effective bandwidth available to each interface decreases. Therefore, a channel assignment algorithm needs to balance between maintaining network connectivity and increasing aggregate bandwidth. Second, how packets should be routed through a multi-channel wireless mesh network? The routing strategy in the network determines the load on each 802.11 interface, and in turn affects the bandwidth requirement and thus the channel assignment decision for each interface.

We developed two sets of algorithms that tune the routes and radio channels used by the network based on its topology and latest traffic profiles. Through a detailed simulation study, we showed that by deploying just 2 NICs per node, the algorithms we developed for the proposed multi-channel WMN architecture can achieve a factor of 6 to 7 throughput improvement compared to the conventional single-channel WMN architecture.In addition, to demonstrate the feasibility of the proposed multi-channel WMN architecture, we successfully built a 9-node Hyacinth prototype that consists of PCs each equipped with two commodity 802.11a interfaces, and empirically showed that it can indeed improve the aggregate throughput of multiple FTP sessions by a factor of 5. These results convincingly prove that with proper channel assignment and routing algorithms the proposed multi-channel wireless mesh network architecture can indeed provide the capacity required for campus-wide wireless backbones.

Chapter 4

Stateful Transport Protocol

4.1 Introduction

The resource management techniques discussed in previous chapter substantially improve end-to-end path capacity of a wireless mesh network. The next question is how to enable applications to make the most of this network-layer capacity, a role traditionally fulfilled by transport protocol. In Chapter 2, we discussed the necessity to devise a custom transport protocol that can fairly and effectively allocate the end-to-end network-layer bandwidth to competing transport flows. We further argued that to improve network utilization efficiently, it is worthwhile to even make individual network routers aware of the transport layer flows and for them to participate in flow-level bandwidth allocation decisions. In this chapter, we develop such a stateful transport protocol, termed *Link-aware Reliable Transport Protocol (LRTP)*, to achieve this goal by providing low-overhead packet reliability and max-min flow fairness.

To achieve reliability, LRTP takes a cross-layer approach. Specifically, it monitors 802.11 link-layer ACKs, or lack thereof, to determine the status of last transmitted DATA packet, and proactively performs local retransmission of the lost packet. This mechanism not only obviates an end-to-end transport layer ACK for each data packet, but also performs quicker and efficient recovery of lost packets.

From congestion control standpoint, LRTP has two different mechanisms. The first mechanism, termed Explicit Rate-based Congestion Control, achieves intra-node max-min fairness. With this mechanism, each LRTP node estimates the available capacity of its outgoing links and fairly divides it across all flows that are traversing these links.

While the rate-based congestion control improves the network utilization and flow fairness when compared with state-of-the-art transport protocols, it does not achieve network-wide fairness. This is because 802.11 MAC layer performs a flow-unaware distribution of bandwidth across different nodes sharing the channel space. The second congestion control mechanism of LRTP, termed Coordinated Congestion Control (C3L), remedies this problem and indeed achieves max-min fairness over the unmodified 802.11 MAC layer. It takes a traffic engineering approach, and continuously adapts the bandwidth allocation

for individual wireless links to the latest traffic profile. The per-link bandwidth is further divided among flows passing through it by the link's sender. Unlike some of the previous proposals, C3L ensures end-to-end flow fairness by taking into account the intra-flow and inter-flow dependencies. Further, it incorporates a general capacity re-estimation algorithm that can effectively resolve the unfairness among wireless links as created by the 802.11 MAC protocol.

The rest of the chapter is organized as follows. Section 4.2 discusses the LRTP's hop-by-hop reliable packet delivery mechanism. Section 4.3 presents the distributed intra-node congestion control mechanism, while Section 4.4 presents the Coordinated Congestion Control mechanism that achieves max-min fairness on top of 802.11-based WMNs. Section 4.5 evaluates LRTP's mechanisms over 9-node miniaturized wireless network testbed MiNT as well as in ns-2 simulations. Finally, Section 4.6 concludes the chapter with a summary of contributions.

4.2 Hop-by-Hop Reliable Packet Delivery

Wireless links suffer from high packet loss rate due to bit corruption. LRTP addresses the packet loss problem using a hop-by-hop retransmission strategy. When an intermediate node detects a packet is lost at a particular hop, it immediately retransmits the packet. This saves unnecessary retransmissions on previous hops over which the packet has already successfully traversed. Moreover, the packet loss recovery procedure is initiated immediately without waiting for end-to-end feedbacks, which typically incur a round-trip delay.

4.2.1 Hop-by-hop Retransmission

To detect packet loss as quickly as possible, LRTP leverages link-layer ACKs in IEEE 802.11 networks, rather than using transport layer ACKs. More concretely, 802.11 already implements an ARQ-based local retransmission scheme to improve the reliability of wireless links. Each frame transmission is explicitly acknowledged by the receiver with a link-layer ACK. LRTP monitors the link-layer ACK of each transmitted packet to determine if it is successfully received. If the link-layer ACK for a transmitted packet is not received in time, LRTP schedules the packet for retransmission. This cross-layer optimization obviates per-packet transport-layer ACKs such as those used in TCP, and saves up to 20% bandwidth, as shown in Table 1.1. In [24], we additionally provide an analytical derivation for this ACK overhead.

The key insight behind LRTP's reliable packet delivery mechanism is that if link-layer ACK is successfully received at each intermediate hop, there is no need for end-to-end transport-layer ACKs. However, packets may be dropped by intermediate nodes due to buffer overrun or route changes. LRTP cannot use link-layer ACKs to detect such congestion related losses. To recover from these losses LRTP resorts to an end-to-end *negative* acknowledgment and retransmission scheme. That is, periodically a flow's receiver sends back a negative ACK that lists all the packets that it has not received and the last packet

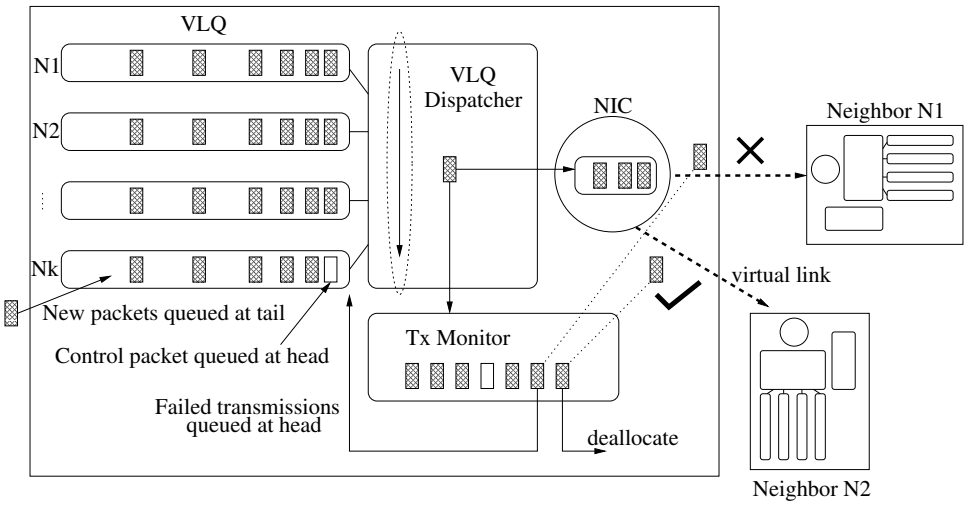


Figure 4.1: *Each outgoing packet is enqueued at the tail of the corresponding virtual link queue (VLQ). The VLQ dispatcher scans through VLQs in a weighted round robin fashion and schedules the packets for transmission on the wireless NIC. The weight assigned to a VLQ is proportional to the sum of channel time assigned to flows going over it. LRTP's transmission monitor keeps a copy of every transmitted packet. When the transmission monitor detects a failure, it enqueues the packet back at the head of the corresponding VLQ. Upon successful transmission, corresponding buffers are deallocated.*

it successfully receives. The sender upon receiving a negative ACK retransmits the corresponding packets. To save on bandwidth, the negative ACK is piggybacked with the rate information packets that are also sent periodically.

4.2.2 Virtual Link Scheduling

In the IEEE 802.11 standard, upon detection of packet loss, the link layer performs its own retransmissions, until either an ACK is received or the number of retries exceeds a predetermined maximum threshold (4 to 7). Although this appears the same as LRTP's reliable packet delivery mechanism, there are some important differences between the two. First, because the quality of different virtual links on a router node may vary significantly at any point in time, using a large retry threshold runs the risk that a router's wireless NIC is unproductively tied up with a poor-quality virtual link when it can be used to transmit frames on other better quality virtual links. This *head-of-line blocking* could also indefinitely delay transport-layer control packets from the receiver and thus disturb the congestion control operation. In contrast, LRTP uses a software-based frame retransmission scheme that can interleave retransmissions on different virtual links so as to maximize the overall throughput of a wireless NIC and to prioritize transmission of control packets. Second, LRTP's software-based retransmission also helps in case of route changes: Instead of trying on the broken link/path, LRTP performs packet retransmissions on the newly discovered path.

To fairly allocate the usage of a wireless NIC among a router node's virtual links, LRTP maintains a separate virtual link queue (VLQs) for each of its neighboring nodes. These VLQs are serviced in a round robin fashion. All *data* packets being transmitted to a particular node are enqueued at the tail of the corresponding VLQ. Only the *control* packets are queued at the head of the VLQ for prioritization. Whenever the wireless NIC becomes available, the VLQ dispatcher dequeues packets from the head of the next eligible VLQ and schedules its transmission on the NIC. A copy of each of these packets is buffered in the transmission monitor, which monitors if a packet is successfully transmitted. Upon success, the corresponding buffer is deallocated; otherwise, the monitor enqueues the packet at the head of the corresponding VLQ for retransmission. Re-queueing a lost packet on head of VLQ as done by LRTP, instead of the head of network card queue as done by 802.11 ARQ mechanism, prevents the head-of-line-blocking problem mentioned earlier. The transmission scheduling mechanism in LRTP is depicted in Figure 4.1.

For efficient operation, packet transmission from a wireless NIC should be *pipelined*. That is, the VLQ dispatcher should maintain multiple packets in the physical transmission queue inside the wireless NIC whenever possible. However, when the physical transmission queue contains too many packets, it becomes difficult to prioritize control packets, and software packet retransmission could lead to excessive packet re-ordering. Therefore, LRTP keeps the number of packets in the physical transmission queue just large enough to enable efficient pipelining, typically around 4.

4.3 Intra-node Fairness: Explicit Rate-based Congestion Control

To eliminate per-packet ACK overhead and achieve faster convergence, LRTP uses a rate-based congestion control scheme. Each intermediate node on a flow's path allocates a bandwidth share to this flow, and the data sending rate of this flow is the minimum of these per-hop bandwidth estimates. The rate-based congestion control scheme consists of two mechanisms: one to estimate each virtual link's effective capacity; and other to fairly allocate a virtual link's effective capacity among flows sharing it.

4.3.1 Virtual Link's Available Capacity Estimation

Each intermediate router node measures the effective capacity from itself to each of its active neighbors whenever it sends data packets to them. This passive monitoring based approach does not incur any additional network traffic. Ideally, for each packet LRTP measures the interval between the time when the packet is at the head of the transmission queue (and is ready for MAC contention) and the time when the packet's link-layer ACK is received. The measured interval is the packet's service time and includes both MAC-layer contention delay and transmission delay. In each router, LRTP maintains a separate data structure for packets going to a different neighbor, and uses a moving average scheme to

calculate the average service time and thus the effective capacity of each virtual link. A virtual link is a directed wireless link from a node to its direct neighbor. Keeping a separate estimate for each virtual link's effective capacity is necessary because the radio conditions for different virtual links may be quite different.

Measuring a packet's service time is non-trivial. If the transmission queue is empty, the packet service time is simply the interval between the time when it is put into the transmission queue and when its link-layer ACK is received. However, to pipeline packet transmissions, the next packet to go should already be in the transmission queue when the previous packet is processed. Therefore, one cannot use the time at which a packet is put into the transmission queue as the beginning of its service time. Instead, we use the time at which the *previous* packet's link-layer ACK is received as the beginning of the current packet's service time. Therefore, there are two possible cases:

1. When there are back-to-back packets in the network card buffers, the service time for n^{th} packet is the duration from the time $(n-1)^{th}$ packet's ACK is seen to the time the ACK for n^{th} packet is seen (i.e. $t_{ACK_n} - t_{ACK_{n-1}}$). This captures the MAC contention delay that the n^{th} packet experiences as well as its transmission delay.

2. If the transmission queue is empty when a packet is queued, the queue insertion time is taken as the beginning of the packet's service time.

4.3.2 Fair and Efficient Bandwidth Allocation

When multiple flows share a virtual link, the effective capacity of the virtual link should be fairly allocated among these flows (independent of source's distance from gateways) on one hand (fairness), and completely used up by these flows on the other (efficiency). To efficiently and fairly allocate the channel bandwidth among flows, each node measures (1) the capacity of each outgoing virtual link, (2) the number of flows going over each virtual link, and (3) input rate of each flow. The per-flow information maintained on intermediate routers makes LRTP stateful. The per-flow information is however maintained as soft-state: A new flow is automatically added to the router's flow-list when its first packet is seen by the router, and an existing flow is automatically expired after certain period of inactivity.

Consider the general case where a node has k outgoing virtual links. The measured capacity of the i^{th} virtual link is C_i, and the number of flows going over it is N_i. Further, let the measured input rate of the j^{th} flow be f_j, and its estimated bandwidth requirement be r_j. The j^{th} flow's requirement r_j is estimated by comparing its input rate f_j with its current bandwidth share a_j. If $f_j < a_j$, then r_j is estimated to be the same as f_j. If, on the other hand, $f_j = a_j$, then r_j is estimated to be ∞, as the flow's source might indeed be able to pump packets at some rate more than f_j, but cannot because it is only assigned a bandwidth share of a_j.

Intuitively, each flow gets an equal share of the link's effective capacity, or each flow's bandwidth share is $\frac{C_i}{N_i}$. If each and every one of the flows sharing a virtual link can use up its bandwidth share, this simple scheme fairly and efficiently divides the link's bandwidth among them. However, there are two issues with this simple scheme:

- If some flows do not really use their share, i.e. $r_j < \frac{C_i}{N_i}$, then it is necessary to further re-distribute the surplus from these flows equally among these flows whose bandwidth requirement exceeds $\frac{C_i}{N_i}$.
- Multiple virtual links emanating from a node may share the same network card (NIC). In this case, the measured C_i for each link is only valid if that particular link is active all the time. However, only one of these multiple virtual links emanating from a NIC can be active at a time. But, two virtual links emanating from two different NICs can indeed be active simultaneously. Therefore, the allocation algorithm needs to work at NIC level instead of virtual link level or node level.

To optimally allocate the link bandwidth C_i across N_i flows in the general case, an LRTP router runs a *max-min allocation algorithm* for each network card as shown in Algorithm 1. To account for differences in link transmission rates and link error rate (and hence C_i) of the associated virtual links, the j^{th} flow's requirement r_j is normalized to $(r_j * C_{max}/C_i)$ where C_{max} is the maximum of the capacities of the virtual links, while C_i is the capacity of the virtual link over which flow j is traversing. $(r_j * C_{max}/C_i)$ in some sense measures the amount of channel time that the flow j needs. The max-min allocation algorithm visits all the flows in an increasing order of their normalized requirement $((r_j + \delta) * C_{max}/C_i)$. Upon visiting a flow j, the algorithm allocates the flow the minimum of its fair share $C_{unalloc}/N_{unalloc}$ and its requirement $((r_j + \delta) * C_{max}/C_i)$. Here, $C_{unalloc}$ is the current unallocated channel capacity for the NIC, and $N_{unalloc}$ is the number of unallocated flows on the NIC. The δ is added to the flow's allocation so that the flow has a chance to grow. Specifically, without extra δ allocation, the flow can only continue to send at its current sending rate f_j. The value of δ should be large enough so as to make a measurable difference when the flow's requirement increases, but small enough to reduce the wastage. We use a value of δ equal to 35Kbps that equals 3 packets/sec for 1500 bytes sized packets.

The bandwidth allocation algorithm is run every time (1) a new flow enters the router, (2) an existing flow expires, (3) the bandwidth requirement of a flow changes, or (4) the available virtual link capacity changes. For each run of the algorithm the unassigned flows are maintained as a *priority queue*; this enables finding the flow with next smallest requirement in $O(log(N))$ time, where N is the number of flows in the router. The overall computational complexity of each algorithm run is therefore $O(N * log(N))$.

The above algorithm does the bandwidth allocation to a flow on each individual hop. Every packet of the flow carries a bandwidth stamp which is examined by every intermediate node on its path. If an intermediate node decides to allocate a smaller bandwidth to a packet's associated flow, the packet's bandwidth stamp is overwritten with the new smaller bandwidth value. The receiver of a flow collects the bandwidth stamps carried by the flow's packets, and communicates their exponential average to the flow's sender through periodic (or change-driven) feedback mechanism. The sender then adjusts its data sending rate to match the feedback bandwidth share.

Algorithm 1: LRTP's NIC-Level Max-Min Fair Allocation Algorithm

Let C_i be the capacity of the i^{th} virtual link of the network card under consideration, and N_i be the total number of flows going through the link.
Let r_j be the estimated requirement of flow j, and A_j its new bandwidth allocation. Flow j goes over i^{th} virtual link.
$C_{max} \leftarrow \text{maximum}_i(C_i)$
$C_{unalloc} \leftarrow C_{max}$
$N_{unalloc} \leftarrow \sum_i N_i$
For all j, $A_j \leftarrow 0$
while ($\exists$ unassigned flow j on the card (i.e. $A_j = 0$) **do**
  $r_{min} \leftarrow \text{minimum}_j((r_j + \delta) * C_{max}/C_i)$
  $A_{min} \leftarrow \text{minimum}(r_{min}, C_{unalloc}/N_{unalloc})$
  $C_{unalloc} \leftarrow C_{unalloc} - A_{min}$
  $N_{unalloc} \leftarrow N_{unalloc} - 1$
end while
for (all flows j) **do**
  $A_j \leftarrow A_j * (C_i/C_{max})$
end for

4.4 Inter-node Fairness: Coordinated Congestion Control

Although LRTP's explicit-rate based congestion control performs much better than TCP's and ATP's congestion control [48], it has its limitations. Specifically, like ATP's, TCP's and EXACT's [49] congestion control mechanisms, it does not do an explicit allocation of bandwidth among neighboring nodes sharing the same channel. Although the network is multi-channel, due to limited number of interfaces on each node, multiple links in a neighborhood may indeed share the same channel. This leads to (1) the hidden terminal problem, and (2) the unfair channel sharing problem. The rate-based congestion control algorithm can thus only achieve fairness at individual node level, rather than at network level. An ideal solution to overcome these problems and achieve inter-node fairness is to explicitly divide the radio channel bandwidth among nodes in a neighborhood, rather than leaving this division to 802.11 MAC layer. This bandwidth division should take into account not only the interference relationship between nodes, but also the number of flows going over them. Specifically, each node should determine the other nodes that lie in its interference arena, and determine the fair channel share it should get. Such a mechanism can improve the fairness for cases shown in Figure 1.3 (reproduced here as Figure 4.2 for ease of exposition).

The objective of this section is to devise a congestion control mechanism that can achieve end-to-end max-min flow fairness over 802.11-based WMNs. The max-min fairness model dictates that a flow is allowed to receive as much bandwidth as it can so long as other flows receiving a smaller share are not adversely affected.

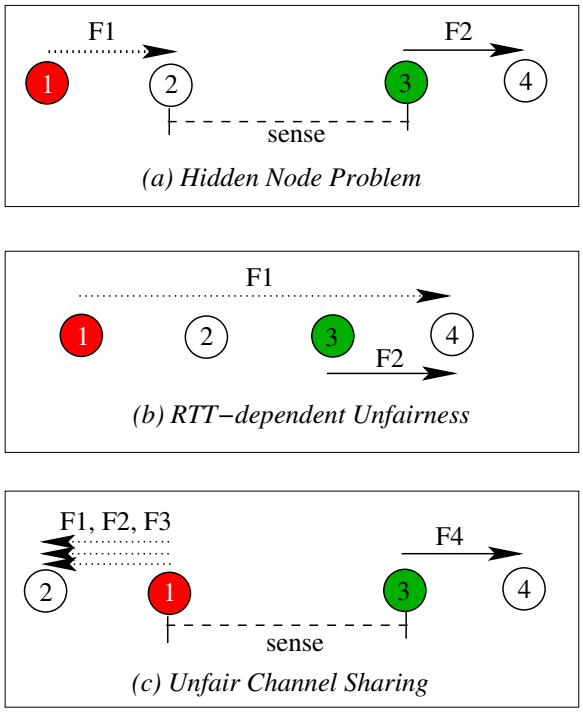


Figure 4.2: *Three scenarios in which significant unfairness among flows arises. The wireless node getting a lesser than fair share of bandwidth is marked as '1' (and colored in red or white), whereas the one getting a larger share is marked as '3' (and colored in green or black). (a) Node 1 lacks informations about Node 3's transmissions, attempts its communication at inopportune times, and eventually backs off unnecessarily. (b) Flow F1 traverses more hops than Flow F2. Some transport protocols, such as TCP, give more bandwidth to flow F2. (c) Flow F1, F2, F3, and F4 all share the same channel, but most transport protocols allocate more bandwidth to F4 than to others.*

4.4.1 Definitions

Before presenting the our solution, here are some definitions we need in our description:

- A bandwidth allocation is *max-min fair* if one can not increase the bandwidth allocation of any flow without reducing the bandwidth allocation of another flow with an already smaller share [50].
- A *collision domain* is defined as a maximal set of directed wireless links all of which interfere with each other pairwise. There is one sender associated with each directed wireless link.
- A *symmetric collision domain* is one where transmission on each link is visible to all the other senders, and therefore each contending sender gets an equal opportunity of accessing the common channel.
- An *asymmetric collision domain* is one where transmissions on some link (inhibitor link) are not visible to other senders (inhibited links), and consequently senders associated with inhibited links are at a disadvantaged position as compared with senders of inhibiting links.

4.4.2 Intuition

4.4.2.1 Single Collision Domain Network

In the simplest case, all links interfere with one another, and hence belong to a single collision domain. The bandwidth share of each link is then proportional to the the number of flows going over it. If n_i is the number of flows going over the i^{th} link, then the *link's* fair share should be $C_{max} * n_i / \sum_i n_i$.

The above allocation works if the effective channel capacity C_{eff} equals C_{max}. Even in a symmetric collision domain, C_{eff} could be less than C_{max} because the MAC-layer overheads, such as backoffs and packet errors, reduce the effective channel capacity. The situation is even more complicated for an asymmetric collision domain, where senders associated with inhibiting links may need to slow down intentionally so that senders of inhibited links can compete on a more equal footing. This slow-down leads to a reduction in the effective channel capacity.

To estimate C_{eff}, we first assume that C_{est}, the estimated overall capacity of the collision domain, equals C_{max}. If $C_{est} > C_{eff}$, then some of the links in the collision domain would not be able to support their incoming traffic load, and their queues would get built up. From the build-up we can infer that the current C_{est} is higher than the collision domain's effective capacity. The queue build up should be sufficient before we infer $C_{est} > C_{eff}$, as short queue build up may occur even because of bursty traffic. If, on the other hand, none of the links in the collision domain has its queue built up despite all nodes sending at their assigned rates, then we may have underestimated the collision domain's capacity.

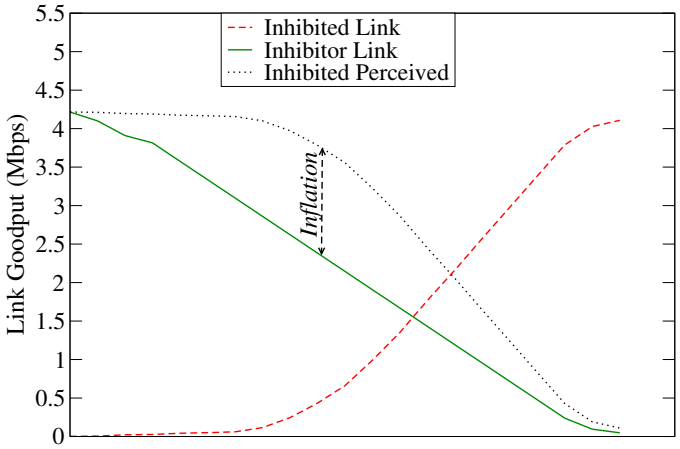


Figure 4.3: *Reducing the rate allocation to inhibiting link gives a fairer ground for the inhibited sender to contend for the channel.*

The same logic can be used to detect capacity mis-estimation in an asymmetric collision domain. Consider the simplest asymmetric collision domain consisting of two links with one flow each as in Figure 4.2(a). Unlike in a symmetric collision domain, where allocating $C_{max}/2$ to each sender would result in queue build-up at both senders, here only the inhibited sender's queue is built up. The inhibiting link's sender, on the other hand, can transmit data at a rate up to C_{max} without experiencing any queue build-up. Figure 4.3 shows this effect. In essence, the inhibited sender perceives an inflated picture of the inhibitor's traffic. Therefore, simply decreasing the inhibitor's rate to $C_{max}/2$ does not help the inhibited sender.

One way to resolve the starvation issue above is to adjust the capacity estimate for a collision domain. If at least one of the senders in a collision domain sees its queue built up, the channel capacity estimate is decreased to $C_{est} - \delta$. However, if none of the senders in a collision domain sees its queue built up, the capacity estimate is increased to $C_{est} + \delta$. Unlike TCP, where each flow does this probing, this algorithm probes a collision domain's capacity for all flows associated with that domain.

4.4.2.2 Multi-Collision Domain Network

With multiple collision domains, each link could participate in multiple overlapped collision domains, and thus receives a bandwidth allocation from each domain it participates in. The most constrained collision domain is the one that assigns the smallest bandwidth allocation and hence decides the allocation to the link.

Additionally, the flows could be multi-hop bringing the intra-flow dependency. Once a flow is allocated the bandwidth at any of its hops, that allocation needs to be propagated on all the hops. Again, the hop that participates in the most constraining collision domain is the one that ends up deciding the allocation to the flow.

4.4.2.3 Two-Level Allocation

While the above algorithm can already allocate share to individual flows, it is costlier to run the above algorithm upon every flow joining/leaving. We therefore perform a two-level allocation: the above algorithm is used to assign bandwidth to individual wireless links. At run-time, the per-link bandwidth is allocated to all the flows going over a link by the link's sender node. The sender on each link ensures that the transmitted traffic on it does not go beyond the first-level assignment. The two-level allocation enables us to incorporate fluctuations in number of flows passing through a link. At run-time, if more flows pass through a link, they can be assigned bandwidth using intra-link bandwidth assignment. Similarly, extra bandwidth gets used if there are lesser number of flows than expected.

4.4.3 Overall Algorithm

Figure 4.4 depicts the various components of the proposed Coordinated Congestion Control Algorithm (C3L), and the data flow among them. Topology discovery algorithm is used by each node to detect its (1) direct communication neighbors, (2) symmetric interference neighbors, and (3) inhibitor links. Based on the result of topology discovery, the central controller constructs a conflict graph of the network [51], and finds all cliques (fully connected maximal subgraphs) in the conflict graph. Each clique in the conflict graph corresponds to a collision domain in the physical network.

The network nodes continuously profile the traffic passing through them, and report to the central controller a list of end-to-end flows as well as the routes taken by them. Based on the traffic profile, the central controller runs the max-min fair allocation algorithm. This algorithm figures out the bandwidth allocation to individual links, and distributes this allocation information to the network nodes. The sender of each link takes the per-link allocation and divides it fairly across all the flows passing through the link. It additionally monitors the corresponding wireless link queue. If the link's queue builds up, its rate estimate is locally modified to prevent further queue build-up. The overloaded links are also reported to the central controller, which uses this information in the next run to adjust the capacity estimate of various cliques.

The central controller could reside either on the gateway nodes that are connected to the wired network or on some administrative location. The WMN nodes use a *gateway discovery protocol* similar to the one described in Chapter 3 to discover a path to the central controller.

4.4.4 Advanced Topology Discovery

The goal of a typical topology discovery process is to determine the usable communication links in the network. Our topology discovery in addition determines the interference relationship between these links, including finding out which links inhibit any particular link. This information is essential to the construction of the conflict graph and to find all collision domains in the network.

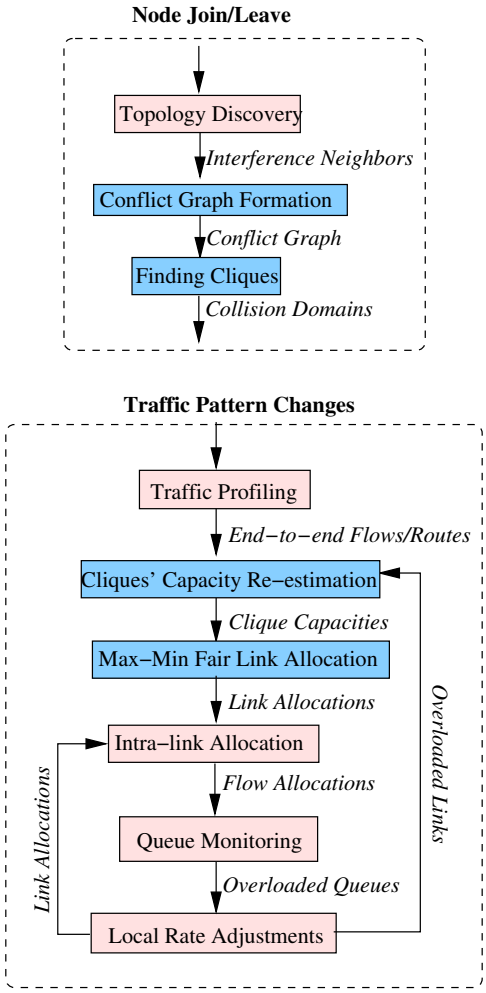


Figure 4.4: *The overall flowchart describing various control mechanisms in the C3L algorithm. The blocks in blue (colored dark) are executed on the central controller, while the ones in pink (colored light) are executed on individual nodes.*

Each node periodically broadcasts a HELLO message. The HELLO messages are received by its communication neighbors, thereby allowing the neighbors to discover the node. Furthermore, the fraction of HELLO messages received by each neighbor is used to determine the delivery ratio of the corresponding wireless link, and infer whether the link is usable or not.

The discovery of interfering neighbors is slightly more involved, as it may not be possible to directly communicate with an interfering neighbor at default settings. Our experiments with a real wireless testbed show that it is indeed possible to communicate with an interfering neighbor if (1) the link-layer encoding is switched to lowest possible (1 Mbps for 802.11b and 6 Mbps for 802.11a), and (2) the transmit power is increased by 3 dB. This is because the lowest link-layer encoding is resilient enough to communicate packets with distant neighbors as long as the received signal quality is above 3 dB. Therefore, changing the link-layer encoding and increasing the transmit power by 3 dB ensures that a node can communicate directly with interfering neighbors.

A more comprehensive approach is described in [52]. In this approach, each pair of nodes (say n_i and n_j) is scheduled to send packets simultaneously, and while the remaining $N-2$ nodes measure the delivery rate from n_i and n_j. Each observer node (say n_k) then uses the measured delivery rate to estimate interference between links from n_i and n_j to itself (in this example between links $n_i - n_k$ and $n_j - n_k$. This approach gives a more accurate estimate of the interference relationship between each N^4 possible *link* pairs in the network using N^2 experiments.

Based on the data collected during topology discovery, the central controller forms the conflict graph for the network. In a conflict graph, every usable directed communication link is represented by a vertex, and there is an edge between two vertices if the corresponding communication links interfere with each other (can not do successful simultaneous transmission). Further, the controller finds all the cliques in the conflict graph using the algorithm proposed by Bron and Kerbosch [53]. A clique is a maximal fully connected subgraph. Physically, a clique represents a collision domain: a set of links, only one of which should be active at a time. Although this algorithm finds all the cliques, it can be modified to do incremental clique finding each time the network topology changes.

Finally, for each communication link, we detect all the inhibitor links. An inhibitor link is one whose sender lies in the interference range of the inhibited link's receiver, but whose sender and receiver lie outside the interference range of the inhibited link's transmitter (Figure 4.5). In this case, the inhibited link's transmitter does not have a complete picture of the transmissions from the inhibitor node, and hence may send RTS at inopportune times, eventually leading to long backoffs.

4.4.5 Max-Min Fair Link Allocation

Based on the traffic profile obtained from the network, the controller periodically performs a max-min fair bandwidth allocation among all wireless links in the network. Listing 2 presents this bandwidth allocation algorithm.

The algorithm goes through each clique in the conflict graph in the decreasing order

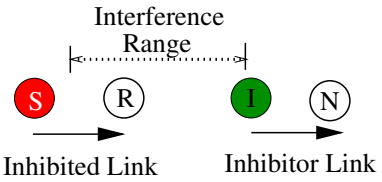


Figure 4.5: *Node S falls outside the interference range of node I, while node R falls within its interference range. Link I-N therefore inhibits link S-R.*

of constraints imposed on the participating flows in the clique. The measure of constraint is the ratio of a clique's residual capacity and the number of unassigned one-hop subflows traversing the clique's links. The most constraining clique (the one with the lowest ratio) is visited first. Upon visiting any clique, the residual capacity is equally divided among all the unassigned subflows. The bandwidth assignment to each subflow is propagated to all its upstream and downstream links. This propagation modifies the residual capacity of all the cliques these newly assigned flows pass through. The algorithm terminates once all the flows have been assigned their fair share. Note that the order of visiting cliques cannot be pre-determined, as the constraint of a clique is a dynamic measure, and may change after some of its subflows get their bandwidth assignment from other more constrained cliques.

4.4.6 Clique Capacity Re-estimation Algorithm

The capacity of a clique is dependent upon the presence of hidden nodes, exact packet transmission scheduling at MAC layer, amount of time nodes spend in backoff, and packet errors. Because of these factors, it is hard to determine a clique's capacity analytically. We therefore take a probing approach to estimate the clique's capacity.

For any given clique, the central controller starts with a capacity estimate C_{est} equal to C_{max}. If a clique's effective capacity $C_{eff} < C_{est}$, some of the senders in the clique are unable to forward their incoming traffic, and hence their queues build up. The build-up of a wireless link's queue beyond a certain threshold serves as an indication that the controller has overestimated the capacity of one or more of the cliques that the wireless link participates in. Short-term queues may also build up because of traffic burstiness: A high threshold is therefore used to identify long-term queue build-up. To pinpoint the clique with capacity overestimate, we estimate the amount of channel time usage of all the cliques that the link participates in. The clique with the maximum channel usage is the one experiencing maximal contention with respect to the link in consideration. An inhibited link perceives an inflated picture of the transmissions occurring on inhibitor links. To account for such inhibition, the inhibitor links' usage is multiplied by a factor, called W_{FACTOR}, before being added to the clique usage. The value of W_{FACTOR} is determined empirically. Once the clique with capacity overestimate is pinpointed, its capacity estimate is decremented. Figure 4.6 shows the variation of W_{FACTOR} based on graphs in Figure 4.3. W_{FACTOR} varies mostly between 1 and 2, with a typical value around 1.7

If, on the other hand, none of the senders in the clique experiences any queue build-up

Algorithm 2: Max-Min Fair Allocation Algorithm

Let C_{cid} be the clique cid, B_{cid} its residual capacity, and N_{cid} the number of unassigned subflows in it.
Let X_{fid} be the bandwidth assigned to flow fid, and O_{nid} be the bandwidth assigned to directed wireless link nid.
Let $[C]$ be the set of all cliques, and $[O]$ the set of all directed wireless links in the network.

```
for j = 1 to |[C]| do
    B_j ← Capacity estimate of clique j
    N_j ← Number of subflows in clique j
end for
while (∃ unassigned flow in the network) do
    min ← Clique with minimum(B_i/N_i)
    for each unassigned flow fid in clique min do
        X_fid ← B_min/N_min
        for each subflow sfid of flow fid do
            for each clique cid that sfid participates in do
                B_cid ← B_cid − B_min/N_min
                N_cid ← N_cid − 1
            end for
        end for
    end for
end while
for nid = 1 to |[O]| do
    O_nid = ∑_s X_s, where subflow s passes through the link nid
end for
```

despite all the senders sending at their assigned rate, then we may have underestimated the clique's capacity. If so, the capacity estimate of the clique is incremented.

The increments to clique's capacity are done in small steps of δ_i. And the decrements are done with exponentially increasing δ_r. Every time a clique's capacity is incremented, δ_r is initialized to δ_i. Each time the previous decrement is insufficient and the queues still build up, the δ_r is multiplied by 2. The rationale of such exponentially increasing decrement step is that the previous estimate of C_{est} did not result in queue build-up, while the $C_{est} + \delta_i$ did. Therefore, the clique's effective channel capacity lies within the previous estimate and current estimate: $C_{est} <= C_{eff} <= C_{est} + \delta_i$. However, sometimes the channel capacity may decrease substantially. If so, we increment the δ_r exponentially to arrive at the correct bandwidth estimate. This clique capacity estimation algorithm is shown in Listing 2.

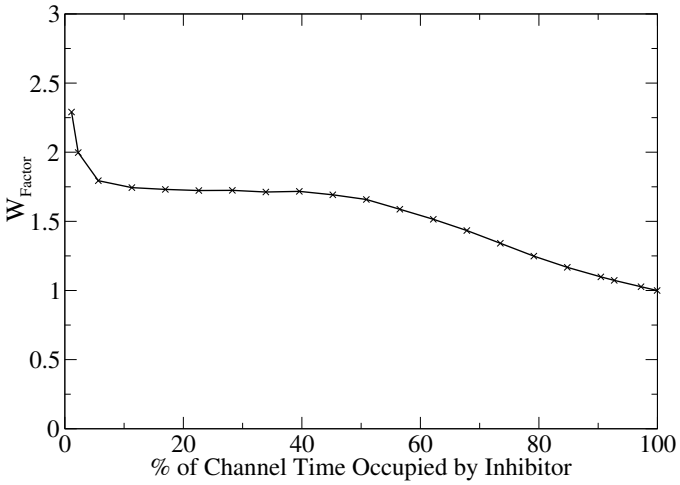


Figure 4.6: W_{FACTOR}: *The ratio of perceived inhibitor's channel usage to the actual inhibitor's load on channel.* W_{FACTOR} *varies between 1 and 2, with a typical value around 1.7*

4.4.7 Per-Link Flow Bandwidth Allocation

Given the bandwidth allocation to all the links, each link's sender runs a second-level local max-min fair allocation over all the flows traversing the link. The sender ensures that the sum of bandwidths allocated to flows on a link never exceeds the link's first-level allocation. Each of the flow's data packet header carries the minimum allocation to the flow on any of its hops. Each time a packet reaches a hop that assigns a smaller bandwidth allocation than is stamped on the packet, the stamp is changed to the local bandwidth assignment on the hop. This information is relayed via the flow's receiver back to the sender in form of periodic control packets sent every *epoch*. These control packets are prioritized on every node to provide timely feedback to the senders.

The intra-link bandwidth allocation algorithm works as follows. Flows associated with a link are first sorted in an increasing order of their sending rate. Flows with the smallest sending rate are considered first. Let C_{alloc} be the first-level allocation to the link, and n be the number of flows passing through the link. Initially, the residual capacity $C_{residual}$ equals C_{alloc}, and number of unassigned flows n_{unasgn} equals n. If a flow's sending rate f_s is smaller than $C_{residual}/n_{unasgn} - \delta$, then it is allocated $f_s + \delta$, and the residual capacity is reduced by that amount. If, on the other hand, f_s is greater than $C_{residual}/n_{unasgn} - \delta$, then the flow is allocated $C_{residual}/n_{unasgn}$. This algorithm ensures that flows with smaller requirements are assigned based on their requirement, while the remaining bandwidth is divided equally among the remaining flows. The extra δ allocation ensures that a finite-requirement flow has a chance to grow. As long as a flow uses δ less than its actual assignment, it is considered to be a finite-requirement flow with a requirement of f_s. If the flow indeed consumes the extra δ bandwidth, the estimated requirement of the flow is reset to infinity. Listing 3 lists the algorithm for intra-link bandwidth allocation.

Algorithm 3: Clique Capacity Re-estimation Algorithm

Let $Overloaded$ be the list of overloaded links. r_{nid} is the average number of retransmissions on the link nid.
Let Cap_{cid}^{prev} and Cap_{cid}^{new} are the previous and new estimated capacity of clique cid.
Let O_{nid}^{prev} be the previous bandwidth assigned to link nid.
Let δ_i^{cid} and δ_r^{cid} be the increment and decrement terms respectively for clique cid's capacity.

for each link nid in $Overloaded$ **do**
  for each clique cid that nid participates in **do**
    $Chnl_{nid}^{cid} \leftarrow$ Estimate_Relative_Channel_Usage(cid,nid)
  end for
  Find clique-id $mcid$ that has $max_{cid}(Chnl_{nid}^{cid})$
  $Cap_{mcid}^{new} \leftarrow Cap_{mcid}^{prev} - \delta_r^{mcid}$
end for
for each clique cid **do**
  if no link nid in clique is overloaded **then**
    $Cap_{cid}^{new} \leftarrow min(C_{max}, Cap_{cid}^{prev} + \delta_i^{cid})$
    $\delta_r^{cid} = \delta_i^{cid}$
  else if clique's capacity is decreased **then**
    $\delta_r^{mcid} = \delta_r^{mcid} * 2$
  end if
end for

Procedure: Estimate_Relative_Channel_Usage(cid,nid)
$usage \leftarrow 0$
for each link lid in clique cid **do**
  if Hidden(nid, lid) **then**
    $usage \leftarrow usage + O_{lid}^{prev} * r_{lid} * W_{FACTOR}$
  else
    $usage \leftarrow usage + O_{lid}^{prev} * r_{lid}$
  end if
end for
return $usage$

4.4.8 Max-Min Fairness Proof

Theorem 2: Given an accurate capacity estimate of each collision domain, the allocation done by C3L is max-min fair.

Proof: Suppose the converse is true. Then, there must be a flow, say Flow i, that can get a higher rate allocation without adversely impacting the allocation of any other flow with

Algorithm 4: Intra-Link Fair Allocation

Let C_{alloc} be the first-level allocation to the link, S_f be the estimated sending rate of flow f, A_f its fair allocation, and n the total number of flows passing through the link,
Sort flows in increasing order of their sending rate S_f.
$C_{residual} \leftarrow C_{alloc}$
$n_{unasgn} \leftarrow n$
for $f = 1$ to n **do**
  $A_f \leftarrow min(C_{residual}/n_{unasgn}, S_f + \delta)$
  $C_{alloc} \leftarrow C_{residual} - A_f$
  $n_{unasgn} \leftarrow n_{unasgn} - 1$
end for

lesser or equal allocation. According to the algorithm, Flow i must have received its allocation based on a bottleneck clique say C_b. By design, Flow i receives the highest allocation on C_b among all the flows passing through C_b. Therefore, increasing its allocation further would imply reducing some flow whose allocation is lesser or equal, which is a contradiction. Therefore, the allocation of Flow i must be highest possible. Consequently, the overall allocation done by C3L must be max-min fair.

4.5 Performance Evaluation

We compared the performance of LRTP with *TCP-Reno* (SACK and delayed-ACK options enabled) because it is currently the most widely used transport protocol over WMNs, and with *ATP* and *EXACT* which represent the state-of-the-art MANET transport protocol and congestion control mechanism.

Whenever possible, the experiments were conducted on a 9-node miniaturized multi-channel multi-hop testbed MiNT [54]. The testbed consisted of 9 nodes each equipped with four Atheros-based Netgate 5004 MP 802.11a/b/g miniPCI cards, each of which was connected to a 4dBi omni-directional antenna through a 25 dB attenuator and operated in 802.11a mode. Use of radio signal attenuators made it possible to form multiple collision domains within a small physical space and try different network topologies by physically re-arranging the nodes. Two of the cards on each node were used for communication on two non-overlapping channels, while the remaining two cards were used for monitoring the communicating cards. Unless otherwise specified, all nodes used a physical transmission rate of 6 Mbps. The link-layer retry threshold was set to 4 by default, but was set to 0 when LRTP's software-based retransmission mechanism was enabled. The routes were pre-discovered and kept fixed during the course of any experiment.

Experiments with larger topologies were conducted using ns-2 simulator extended to implement our proposed protocol.

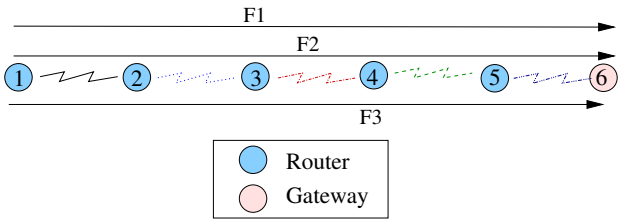


Figure 4.7: *The network topology and set of flows used in the evaluation study of different transport protocols under normal and erroneous channel conditions.*

Flow Id	TCP Thruput (Mbps)	ATP Thruput (Mbps)	LRTP Thruput (Mbps)	Optimal UDP Thruput (Mbps)
1	1.03	0.70	1.20	1.30
2	1.13	0.78	1.24	1.30
3	1.22	0.84	1.29	1.30

(a) Average Channel Conditions.

Flow Id	TCP Thruput (Mbps)	ATP Thruput (Mbps)	LRTP Thruput (Mbps)	Optimal UDP Thruput (Mbps)
1	0.27	0.34	0.48	0.54
2	0.29	0.38	0.52	0.54
3	0.36	0.50	0.54	0.54

(b) Erroneous Channel Conditions.

Table 4.1: *(a) Throughput comparison among different transport protocols in normal radio conditions. ATP and LRTP improve over TCP by eliminating per-packet ACKs. The difference between ATP and LRTP is mainly due to LRTP's better bandwidth share estimation algorithm. LRTP achieves close to the ideal behavior in terms of both overall throughput and inter-flow fairness. (b) Throughput comparison among different transport protocols in noisy radio conditions. TCP incorrectly interprets bursty bit errors as a sign of congestion, and slows down the sending rate unnecessarily. ATP and LRTP do not suffer from the same problem that TCP has because they don't rely on ACKs for congestion identification. LRTP performs better than ATP because of its localized retransmission scheme and better bandwidth estimation.*

4.5.1 LRTP with Intra-node Fairness

We first evaluated the effectiveness of LRTP with the Explicit Rate-based Congestion Control proposed in Section 4.3. Recall that each node here independently estimates the bandwidth of its virtual links and divides it among the outgoing flows.

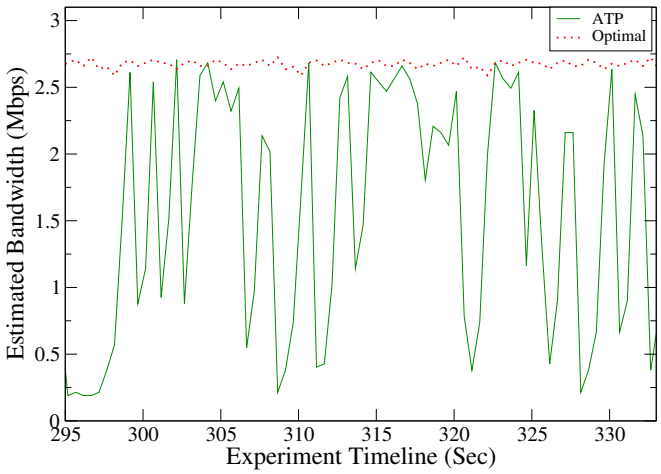


Figure 4.8: *Despite relatively low fluctuations in channel bandwidth, coupling queue-size into service-time leads to fluctuations.*

4.5.1.1 Throughput

We first compared the overall throughputs of different transport protocols under two wireless link conditions: (1) normal radio channel conditions without errors, and (2) noisy radio channel conditions where errors were introduced by reducing the transmission power of nodes. We used a linear chain topology as shown in Figure 4.7, and introduced 3 flows through the network, each of which originated from Node 1, and terminated at the same wired gateway–Node 6.

Table 4.1 (a) compares the throughputs of the three flows using the chosen transport protocols under normal radio channel condition. Under normal channel conditions, LRTP's performance was closest to the optimum, which was estimated by trying to send UDP streams at different rates until reaching the maximum. Surprisingly, ATP's performance was even worse than that of TCP.

The problem lies in ATP's use of overall-service-time to estimate network bandwidth. This approach couples the queue-size management with rate estimation, which leads to traffic fluctuations and in turn non-optimal estimation of channel bandwidth [55]. More concretely, it is very hard to maintain exactly one data packet from each flow on every router. If there is even a slight change in transmission time of a single packet, a queue (say of two packets) builds up on the router for the rest of the epoch. Clearly, an extra packet in the queue does not indicate any change in the network bandwidth. However, in the next epoch ATP sender proportionally reduces its sending rate (to half in this example) to bring the queue down to one packet. It is in these epochs, that an ATP sender underestimates the path bandwidth. Figure 4.8 shows substantial fluctuations in the bandwidth achieved by a single ATP flow traversing 5-hop linear chain network. Here each hop operates in a different channel. A UDP stream that was sent over the same network suffered much less fluctuation which suggests that variations in radio channel conditions are relatively small.

Wireless channels suffer from frequent channel errors [56]. To study the impact of

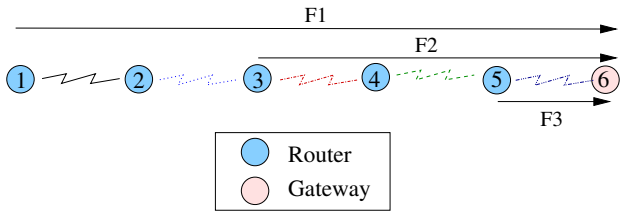


Figure 4.9: *The network topology used in the evaluation study and the set of flows going through it in fairness experiments.*

Flow Origin	TCP Thruput (Mbps)	ATP Thruput (Mbps)	LRTP Thruput (Mbps)	Optimal Thruput (Mbps)
1	1.27	0.94	1.75	1.74
3	1.35	0.94	1.63	1.74
5	1.96	1.16	1.85	1.74

Table 4.2: *LRTP removes RTT-related unfairness.*

channel errors on congestion control mechanism, we experimented with the same chain topology, but reduced the transmission power of nodes. Table 4.1 (b) shows the throughputs of the three flows under different transport protocols. In this case, TCP's performance dropped substantially because its congestion control mechanism was inappropriately triggered whenever three or more consecutive packets were dropped due to burst errors. ATP and LRTP rely on explicit rate measurements, and do not mistake bit errors for congestion. LRTP's improvement over ATP came from its hop-by-hop retransmission strategy that avoids the bandwidth waste due to unnecessary end-to-end retransmissions, as well as from more accurate bandwidth estimation.

4.5.1.2 Fairness

A WMN typically has one or multiple gateways that connect the WMN to the wired Internet. Because different nodes in a WMN have different hop distances to these gateways, it is important that the transport protocol ensures network flows with different hopcounts get an equal share of the network resource. Although most transport protocols can fairly allocate the network resource among flows that share the same set of hops, they cannot handle flows with different numbers of hops very well. We used the network topology and the three flows shown in Figure 4.9 in this experiment, where each flow was doing an FTP upload. Table 4.2 shows the FTP throughputs of the three flows under TCP, ATP, and LRTP. TCP showed maximum unfairness by penalizing the flow with the largest hopcount, F_1, and giving more of the available bandwidth to the shortest flow, F_3. Because bandwidth allocation in ATP and LRTP is independent of the flows' RTT, both of them can provide reasonable fairness. Between them, LRTP offered the most fair bandwidth allocation because of its direct bandwidth allocation.

Protocol	Base Tx Time (sec)	Tx Time with Extra Bandwidth (sec)	% Improvement
TCP	27.17	24.90	8.4
ATP	28.63	28.11	1.8
LRTP	24.03	20.52	14.6
Optimal	23.52	20.00	17.6

Table 4.3: *Transfer time for a 10MB file when the available channel bandwidth was varied by introducing an ON-OFF 2 Mbps UDP stream. The base case corresponds to the experiment where the UDP stream remained ON for the whole experiment duration. The extra bandwidth case corresponds to the experiment where the UDP stream was turned OFF three times for 2 sec each time.*

4.5.1.3 Network Adaptation

Unlike wired networks, the raw capacity of wireless links can vary widely because of physical transmission rate adaptation, fluctuations in injected workload and radio channel characteristics, and route changes. To evaluate LRTP's responsiveness to variations in available bandwidth, we took a 4-hop linear chain multi-channel network. First, a constant-rate 2 Mbps UDP stream was sent on the channel used by one of the intermediate hops, and the total elapsed time for transferring a 10 MB file over this network was measured. The experiment was then repeated but this time the 2 Mbps UDP stream was suspended three times (for 2 sec each time) during the experiment. This created the effect of changing network bandwidth. Table 4.3 shows the transfer time that each of the three transport protocols required with and without network bandwidth changes. Among these three protocols, LRTP was best capable of exploiting the transient bandwidth improvements by 2 Mbps and turning it into overall throughput gain. Further, our calculation shows that the reduction in file transmission time achieved by LRTP is close to optimal. This shows that LRTP is more adaptive to raw bandwidth variation than the other two transport protocols.

4.5.1.4 Throughput and Fairness in Tree-based Topology

Next, we evaluated the performance of different transport protocols for a general tree topology network as shown in Figure 4.10. All traffic was either destined towards or sourced from the root gateway (Node 3) to create the effect of FTP uploads and FTP downloads respectively. Figure 4.11 plots the throughputs achieved by individual flows. Note that the average as well as minimum allocation for LRTP are the highest suggesting that LRTP can better utilize the available network resources. ATP, on the other hand, is the worst performer because of fluctuations in the bandwidth determined by its sender. Figure 4.12 shows similar results for the case of upstream flows.

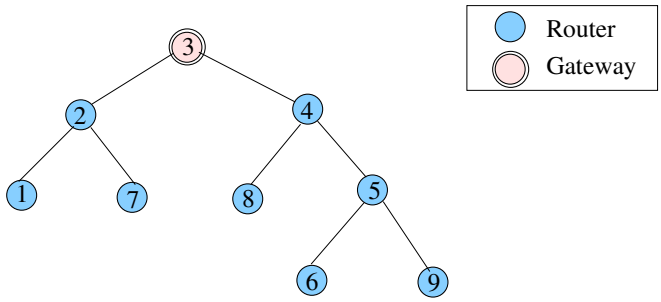


Figure 4.10: *Tree-based testbed topology: Node 3 is the gateway node connected to the wired network.*

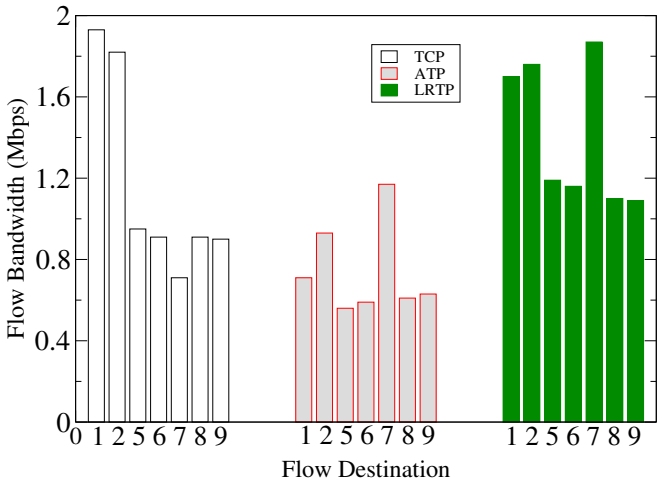


Figure 4.11: *Throughput of individual downstream FTP flows for topology in Figure 3.25. The minimum allocations to any flow for TCP, ATP, and LRTP are 0.71 Mbps, 0.56 Mbps, and 1.09 Mbps respectively. Their average allocations across all flows are 1.16 Mbps, 0.74 Mbps, and 1.41 Mbps.*

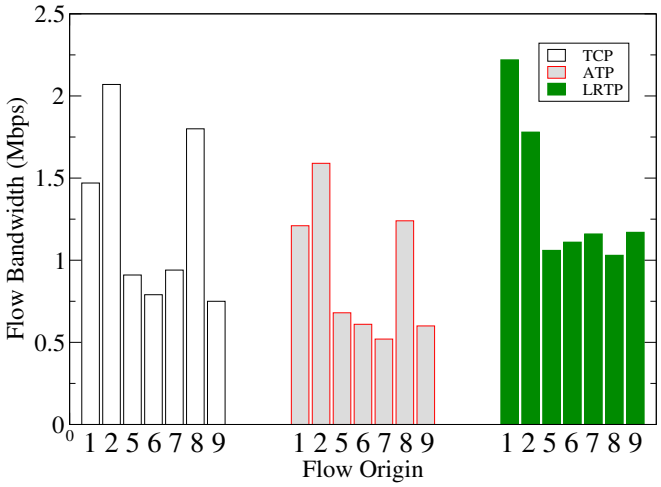


Figure 4.12: *Throughput of individual upstream FTP flows for topology in Figure 3.25. The minimum allocations for TCP, ATP, and LRTP are 0.75 Mbps, 0.60 Mbps, and 1.02 Mbps respectively, while their average allocations are 1.25 Mbps, 0.92 Mbps, and 1.36 Mbps respectively.*

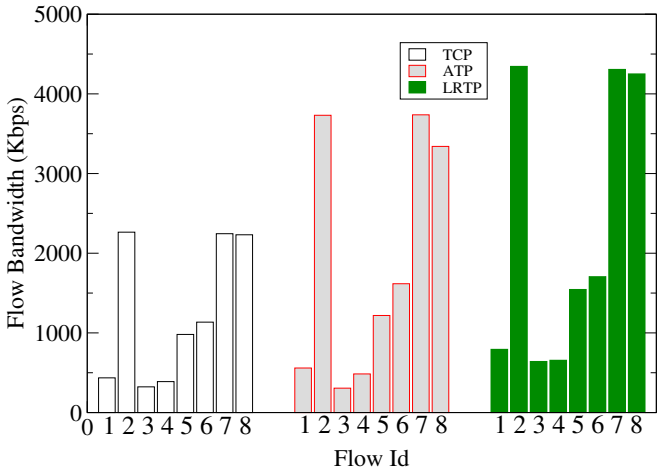


Figure 4.13: *40-node random topology simulated in ns-2. While TCP and ATP allocate a minimum of 323 Kbps and 303 Kbps respectively, the minimum allocation of LRTP does not go below 641 Kbps.*

4.5.1.5 Throughput and Fairness in Larger Topologies

To evaluate different transport protocols for larger topologies, we decided to use ns-2 simulations. We first simulated a 40-node *random topology* with 8 traffic flows between randomly chosen sources and destinations. The results are shown in Figure 4.13. LRTP not only has the highest minimum allocation, but also the highest average allocation. LRTP thus improves the flow fairness as well as overall network utilization.

Next, we simulated a 100-node network arranged in a *10X10 grid topology*. We ran

	TCP		ATP		LRTP	
Flow	**Avg**	**Std Dev**	**Avg**	**Std Dev**	**Avg**	**Std Dev**
Min-1	423	104	677	305	822	71
Min-2	453	17	687	330	919	201
Average	1057	579	1621	875	1826	1031

Table 4.4: *Minimum and average bandwidth allocations (in Kbps) for 10 flows going over a 100-node grid network. TCP is the worst performer for large topologies. LRTP still has the highest minimum allocations as well as average allocation.*

10 flows between randomly chosen sources and destinations. Table 4.4 summarizes the results. Here, TCP does the worst minimum and average allocations. Despite fluctuations in bandwidth allocation of individual flows (as can be seen from standard deviation of bandwidth achieved by minimum flows), the overall ATP performance is better than TCP for large topologies. This is because, in larger topologies due to limited number of NICs on each node, multiple neighboring links could end up using the same channel. The resulting self-interference leads to worse performance of TCP. Again, LRTP performs best both in terms of minimum allocation and average allocation.

4.5.2 LRTP with Inter-node Fairness

We next studied the effectiveness of LRTP with the Coordinated Congestion Control Algorithm (C3L) proposed in Section 4.4. Unless otherwise specified, the default settings for these experiments were as follows. All simulated networks were operating in single channel, as that is where the fairness problems are most prevalent. The RTS/CTS mechanism was enabled. δ_i was set to 5% of C_{max}. W_{FACTOR} was set to 1.7. The cycle time for running the centralized max-min fair allocation was 8 seconds, while the $epoch$ for sending response from a flow's receiver to its sender was 1 second. Again, the routes were pre-determined and kept fixed during the course of any experiment.

4.5.2.1 Fairness in Specific Network Configurations

We compared the performance of C3L with the optimum in three different scenarios from Figure 4.2. The experiments were conducted using ns-2 simulations. The optimum in each case identified was by exhaustively trying out various sending rate combinations in the network. Figure 4.14 shows the results.

Scenario (a) corresponds to the hidden node case, where Flow F1's link is inhibited by Flow F2's link. As shown in Figure 4.14(a), despite this inhibition C3L achieves an almost equal allocation to F1 and F2. Further, the allocation is close to optimal. The fair allocation comes from explicit rate control of inhibiting link in C3L. This is in contrast to the extremely unfair allocation done by TCP, ATP and EXACT in the same scenarios (Figure 1.4).

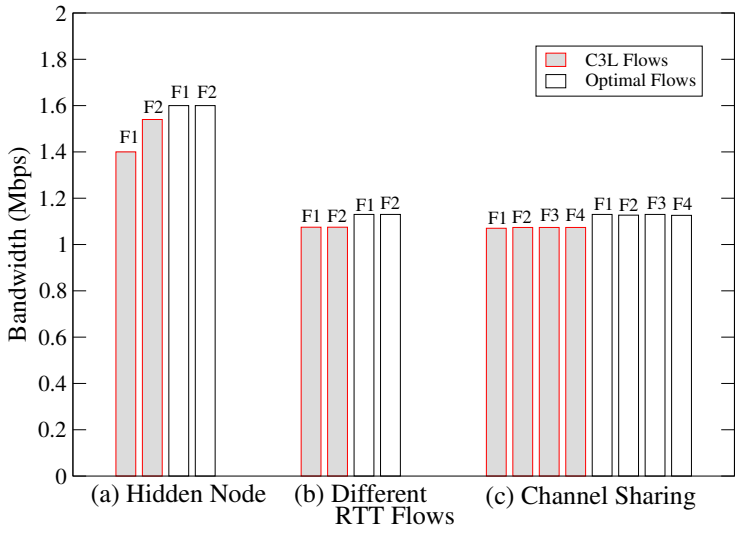


Figure 4.14: *Performance of C3L in different scenarios of Figure 4.2: (a) C3L accounts for inhibition relationship in case of hidden nodes to performs fair allocation of bandwidth; (b) C3L's fairness is end-to-end and does not depend upon the number of hops a flow traverses; (c) C3L takes spatial sharing into account and fairly allocates radio resource among interfering flows, even when they do not share a common node.*

Scenario (b) corresponds to the case, where a 3-hop flow F1 shares a hop with a 1-hop flow F2. While TCP is known to be unfair to flows traversing more hops, C3L's fairness is end-to-end and is independent of the number of hops a flow traverses, as shown in Figure 4.14(b).

Finally, scenario (c) corresponds to the case of flows sharing a common radio channel, but not a common wireless node. Most existing transport protocols give more bandwidth to a flow emanating from node with fewer flows (Figure 1.5). In contrast, C3L recognizes this special channel sharing pattern and appropriately allocates bandwidth to competing links in proportion to the number of flows passing through them. This mechanism results in a fairer bandwidth allocation among flows that share the same radio channel, as shown in Figure 4.14(c).

4.5.2.2 Empirical Results

We conducted similar experiments over MiNT, the miniaturized 802.11a testbed [54]. We began by reproducing the hidden terminal problem (Figure 4.2 (a)) in the testbed and evaluated its impact on the fairness properties of different congestion control mechanisms. Figure 4.15 shows the results. In contrast with simulation results (Figure 1.4), the inhibited flow did not suffer from starvation with any congestion mechanism. This is because of the differences in the NIC implementation of the backoff algorithm. Specifically, our experiments suggest that in commercial 802.11a/b/g NICs used in our experimentation, the maximum backoff window is set to a small value in order to achieve better performance.

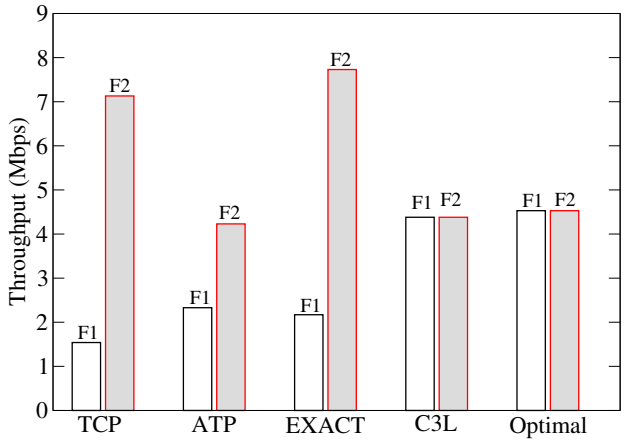


Figure 4.15: *Impact of hidden node problem on different congestion control mechanisms as observed on the testbed. The testbed topology is similar to the one shown in Figure 4.2 (a).*

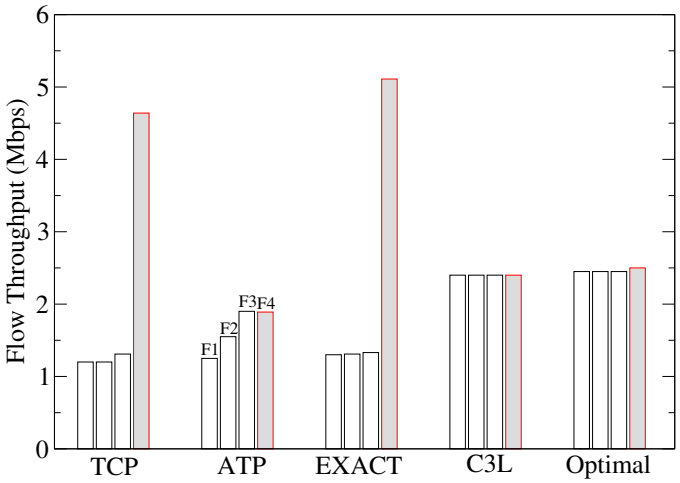


Figure 4.16: *Fairness of different congestion control mechanisms for channel space sharing scenario as observed on the testbed. The topology of the testbed was set in accordance with Figure 4.2(c).*

Similar to simulations, existing transport protocols demonstrated substantial unfairness. C3L not only remedies this unfairness, but achieves a performance close to optimal.

We similarly reproduced general channel space sharing scenario (Figure 4.2 (c)) in the testbed. Figure 4.16 shows the fairness of different congestion control mechanisms in this setup. Although all flows are sharing the same channel space, existing transport protocols allocate less bandwidth to flows F1, F2 and F3 than flow F4. C3L not only achieves fairness in this scenario, but also achieves close to optimum fairness.

4.5.2.3 Fairness in General Network Configurations

The ability to effectively deal with the above three specific scenarios renders C3L a fairer bandwidth allocation protocol for general networks. For this subsection, we evaluated the effectiveness of C3L for several large mesh networks. Here we present the results of two instances. For each protocol, we show the fairness index, the minimum-allocation flow's end-to-end bandwidth, and the average end-to-end flow bandwidth.

The fairness index indicates the degree of fairness in bandwidth allocation across flows. If X_i is the end-to-end bandwidth achieved by flow i and n is the number of flows in the network, then the fairness index is computed using the following formula [57] –

$$FairnessIndex = \frac{(\sum_i X_i)^2}{n * \sum_i (X_i)^2} \tag{4.1}$$

The closer the fairness index is to 1, the fairer the associated bandwidth allocation. The max-min fairness does not mean exactly equal allocation, and even the optimal fairness index may not reach 1 in many cases.

Figure 4.17 shows the fairness achieved by different congestion control mechanisms in a 64-node *8x8 grid network* with 20 randomly chosen flows. The fairness index for C3L is much better than other congestion control mechanisms, suggesting a more uniform allocation of bandwidth across flows by C3L. The second histogram shows the bandwidth achieved by minimum allocation flow in all the cases. C3L again does a much fairer allocation of bandwidth when compared with other protocols that end up starving some flows. Finally, the last histogram shows the average allocation across flows. The comparable average allocation of C3L suggests that C3L achieves fairness while also maximizing the utilization of the network. C3L is thus not just fair but also efficient. The average bandwidth is somewhat smaller for C3L because it gives more bandwidth to some of the flows with a larger hop count. Allocating bandwidth to a flow with a larger hop count consumes more radio resource, and therefore decreases the average flow bandwidth.

Histograms in Figure 4.18 show the same results for a 64-node *random mesh network* with 4 gateway nodes distributed across the network. The traffic profile comprises 15 flows originating from randomly chosen nodes, and ending at the closest gateway. Again, C3L performs a fairer and efficient allocation of bandwidth across all the flows. Figure 4.19 shows the actual distribution of end-to-end bandwidth achieved by different congestion control mechanisms. TCP, ATP, and EXACT give much higher bandwidth to some of the flows, while starving others. The allocation of C3L is most uniform with no obvious starvation.

4.5.3 Algorithm Analysis

In this section, we present further analysis of the algorithm. Specifically, we dissect the performance improvements due to different mechanisms of C3L, its convergence properties, and computation and communication overheads. Lastly, we discuss a limitation of C3L in real network through an experiment on the MiNT testbed.

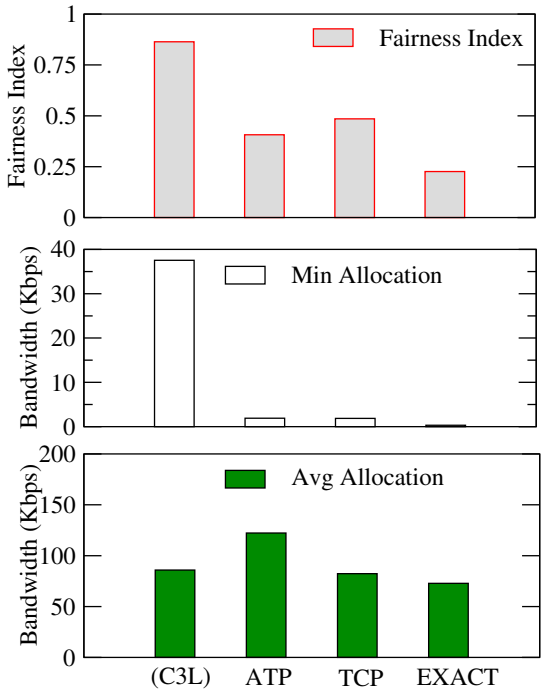


Figure 4.17: *Performance of C3L in a 64-node grid network with 20 end-to-end flows. The fairness index of C3L flows is closest to 1 suggesting a fair allocation of bandwidth with use of C3L. This result is also strengthened by the second histogram that shows that unlike ATP, TCP, and EXACT, no C3L flow is starved. Finally, C3L achieves this fairness without sacrificing the efficiency, as seen from its comparable performance in terms of average bandwidth allocation to each flow.*

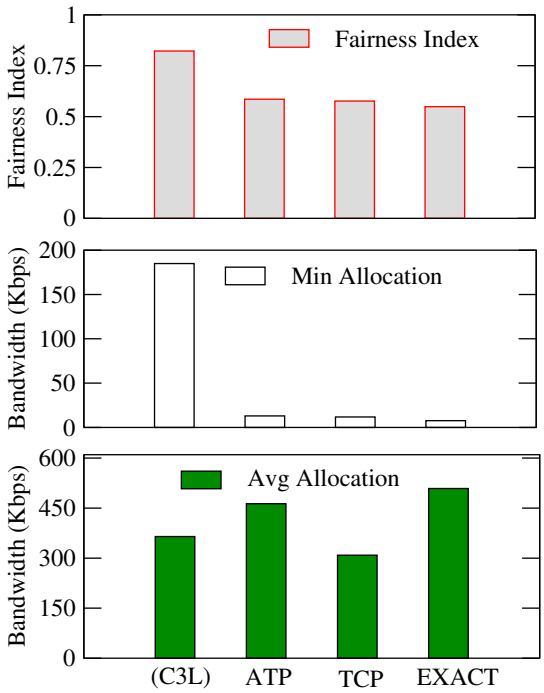


Figure 4.18: *Performance of C3L in a 64-node random mesh network with 4 gateways and 15 flows each destined to the closest gateway. The bandwidth distribution of C3L flows is most fair with no starvation.*

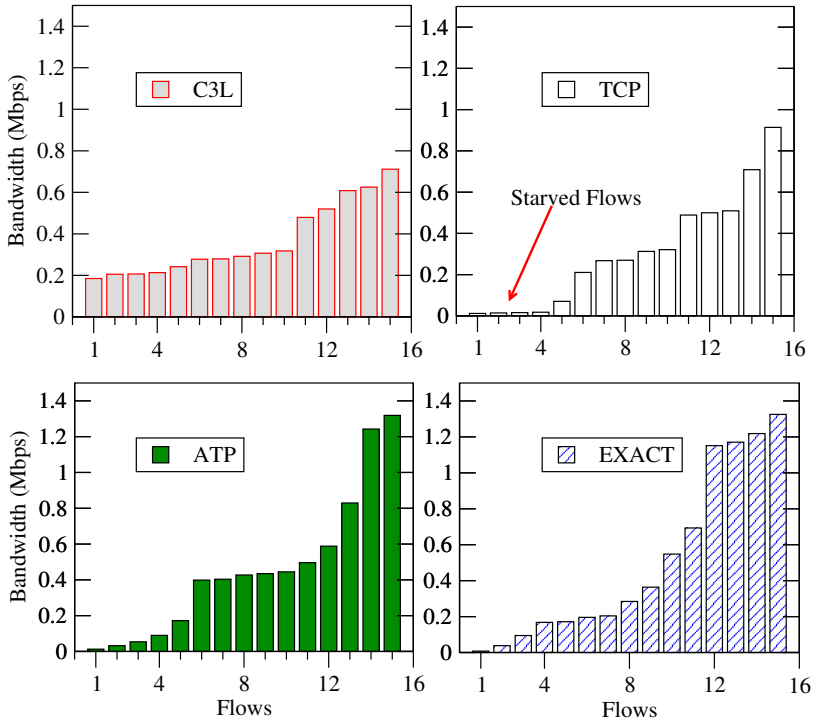


Figure 4.19: *Actual end-to-end bandwidth distribution across flows for results in Figure 4.18. C3L has the most uniform distribution of bandwidth across flows. All TCP, ATP, and EXACT lead to starvation of some of the flows.*

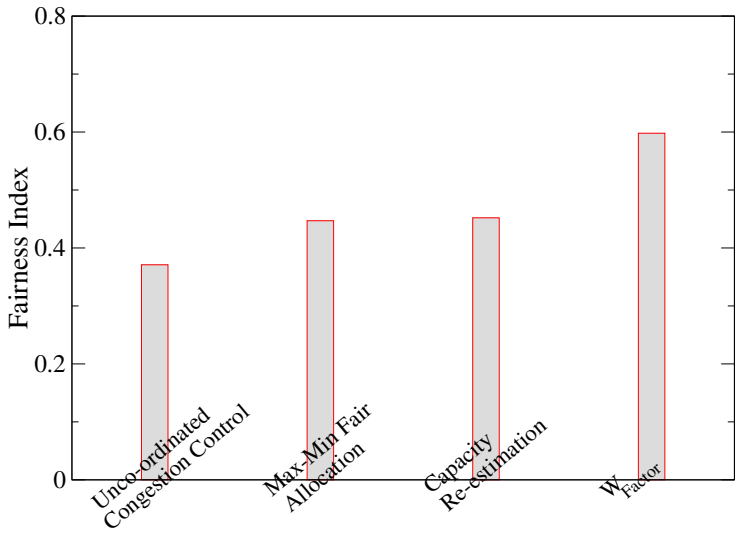


Figure 4.20: *Contribution of different control mechanisms in C3L to inter-flow fairness in a 28-node random mesh network with 5 end-to-end flows. The control mechanisms are introduced in the following order: Max-min fair allocation, Capacity re-estimation, and* W_{FACTOR}.

4.5.3.1 Contribution of Individual Mechanisms

To dissect the performance improvements due to different mechanisms of C3L, we took a 28-node random mesh network and ran 5 flows through the network. Figure 4.20 shows the improvement in fairness index as we turn on each control mechanism of C3L. The leftmost bar corresponds to the case where each node estimates the bandwidth of its links locally in a way similar to EXACT. This causes some flows to starve leading to a small value of fairness index. In the second bar, we introduce max-min fairness but not the capacity re-estimation algorithm. The fairness index indeed improves with the introduction of the max-min fairness algorithm. Introducing capacity re-estimation algorithm alone does not help much. However, the introduction of capacity re-estimation along with W_{FACTOR} leads to a substantial improvement in fairness index. 802.11 MAC layer introduces inefficiencies and unfairness that lead to capacity reduction of some of the collision domains. Accurate estimation of each collision domain's capacity is therefore an important component of fair bandwidth allocation.

4.5.3.2 Convergence Speed

We next evaluated the convergence properties of the C3L algorithm. We took a 64-node grid network and two randomly generated traffic profiles T_1 and T_2 of 20 flows each. For the first half of each experiment, the traffic load was exerted based on the traffic profile T_1. Then the traffic load was switched to the traffic profile T_2 in the second half and the convergence time measured. The network was assumed to be converged when 90% of the

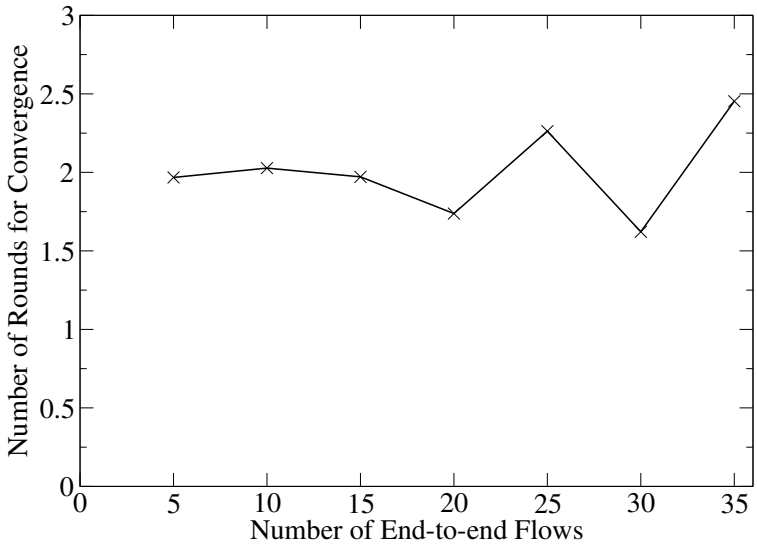


Figure 4.21: *Algorithm convergence speed vs number of flows. Most of the flows converge within 2 rounds of the proposed algorithm.*

flows reach 90% of their eventual average bandwidth. The experiment was repeated with different number of flows in T_1 and T_2. The graph in Figure 4.21 shows that most flows converge within 2 rounds of the proposed algorithm.

One way to reduce the convergence time is to reduce the cycle time, i.e. the periodicity at which the proposed algorithm is run. Table 4.5 shows the average number of rounds it takes when different cycle times are used. At very small cycle time, the intra-link bandwidth allocation does not converge completely, hence the number of rounds required for convergence increases. In the end we choose 8 seconds, which optimizes the convergence time as well as the communication overhead, as the cycle time for all our experiments.

4.5.3.3 Protocol Overheads

The proposed algorithm requires additional computation for topology discovery and max-min bandwidth allocation. Topology discovery requires incrementally finding all the cliques and is only executed when a node is added or deleted from the network. Because topology changes in a WMN are fairly infrequent, the computational cost of the incremental clique finding algorithm is less of a concern. Further, because of the physical proximity of radio interference, the degree of each conflict graph node is limited, and the number of cliques in a network tends to be linear with respect to the number of its nodes (Figure 4.22).

The run-time computation overhead of the algorithm that determines max-min fair allocation of flows is more critical to the overall network performance because it needs to be invoked frequently. We measure the computation time of each algorithm run with respect to the changing number of flows (Figure 4.23). Even with 100 flows on a 64-node grid network, this computation only takes about 1.3 msec on a standard 1.7 GHz CPU.

Cycle Time	Convergence (Number of Rounds)
2	3.31
4	3.70
8	1.34
16	1.49

Table 4.5: *Number of rounds for convergence versus cycle time for executing the proposed centralized max-min fair allocation algorithm.*

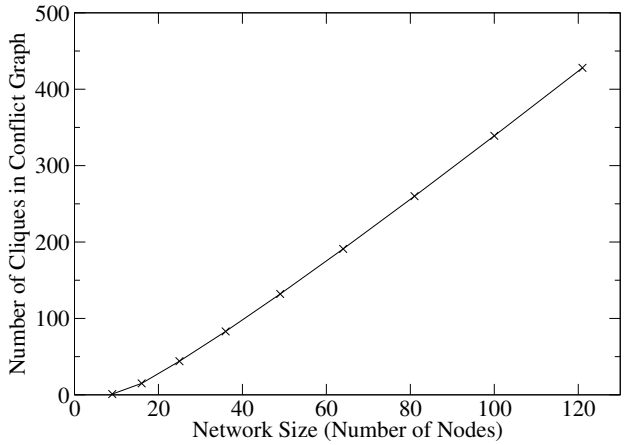


Figure 4.22: *Number of cliques increases linearly with the network size.*

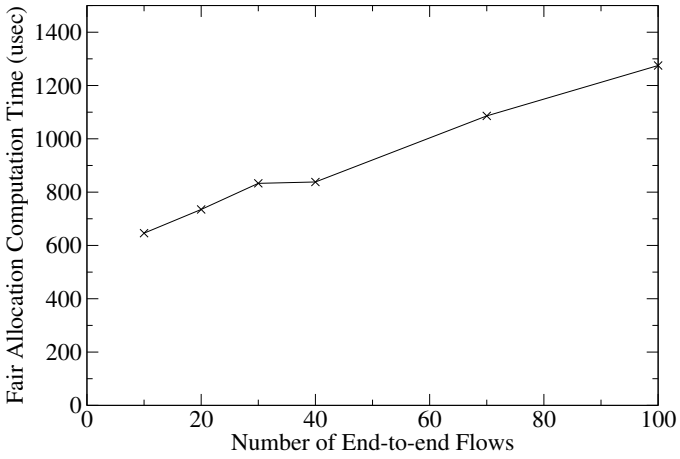


Figure 4.23: *Fair allocation computation time increases linearly with number of flows, but stays below 1.5 msec even with 100 flows.*

The other performance overhead associated with the proposed algorithm is the communication cost required to collect the traffic profile. This needs to be done once in every round of the centralized algorithm. In our simulations, each node sends one packet containing information about its links, flows, and routes, to the central controller. As the diameter of the network increases as $\sqrt{N}$ times the network size (N is the number of nodes), the overall communication overhead increases as $N * \sqrt{N}$. Our measurements show that with 100 flows on a 64-node grid network, the communication traffic imposed by C3L is less than 0.5% of network bandwidth.

4.5.3.4 C3L Limitations

A limitation of C3L is that it assumes two links are either interfering all the time or not interfering at all. While this 0-1 interference assumption works in simulations, in reality the interference between two links could be partial. Specifically, temporal variations in channels could lead to time-dependent interference, where two links sometimes interfere and sometimes not. To evaluate the impact of this 0-1 interference assumption on C3L performance, we placed 4 nodes such that link N1-N2 is on the boundary of interference zone of link N3-N4 leading to partial interference scenario, and sent two flows Flow-1 and Flow-2 over them . Figure 4.24 demonstrates the temporal variations by plotting the throughput of Flow-1 for the optimal case and the C3L case. In all, the optimal algorithm which instantly adapts the sending rates of two flows based on changes in interference between them could achieve 6.43 Mbps and 6.52 Mbps for Flow-1 and Flow-2 respectively. In contrast, C3L made a conservative assumption that the two links always interfere, and on an average achieved 4.85 Mbps and 4.90 Mbps respectively. This result does not conflict with Theorem 2, as here the capacity of the collision domain is indeed changing over time.

4.6 Conclusions

Wireless mesh network is touted as the next frontier for wireless technology revolution. To effectively bring the raw link capacity of a wireless mesh network to end user applications, an efficient and fair transport protocol is absolutely critical. To maximize the utilization efficiency of the precious wireless link resources, we advocate a stateful approach to transport protocol design so as to trade additional router complexity for higher network throughput. In fact, we believe wireless mesh network routers could be made even application-aware as long as it can improve the overall system performance.

Another major concern about using IEEE 802.11 in WMNs is its inherent unfairness at MAC layer when used in multi-hop networks [14]. Existing solutions to this problem either do not effectively resolve this unfairness, or require modifications to the MAC protocol. At the same time, the congestion control schemes in state-of-the-art transport protocols for WMNs, such as TCP, ATP and EXACT, actually exacerbate this MAC unfairness problem and end up performing a highly unfair end-to-end bandwidth allocation across flows.

In this chapter, we presented the design and evaluation of a stateful transport protocol,

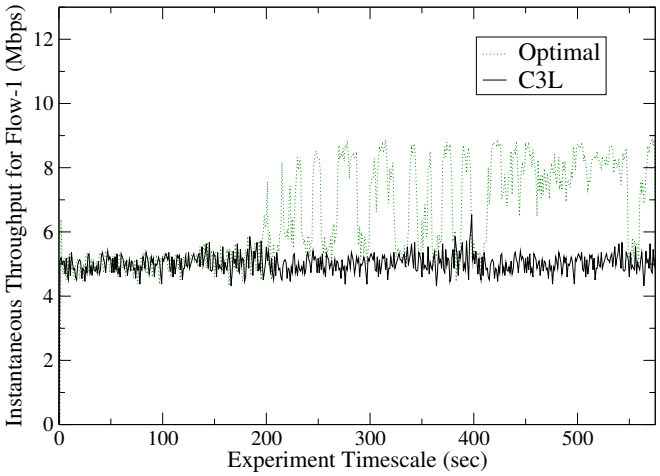


Figure 4.24: *Temporal variations in link interference results in throughout variations. Here, two flows Flow-1 and Flow-2 were sent over two interfering hops N1-N2 and N3-N4 respectively. The graph shows the instantaneous bandwidth achieved by Flow-1 for both cases. The average bandwidth achieved by Flow-1 and Flow-2 was 4.85 Mbps and 4.90 Mbps respectively using C3L, and 6.43 Mbps and 6.52 Mbps respectively using the optimal algorithm.*

called *Link layer-aware Reliable Transport Protocol (LRTP)*. Firstly, LRTP features an efficient hop-by-hop reliability mechanism that leverages 802.11 link-layer ACK information to detect packet losses and perform local retransmissions.

Secondly, we presented two different congestion control mechanisms for LRTP. The first mechanism, called Explicit Rate-based Congestion Control, measures the effective capacity of a router's virtual links and allocates bandwidth among flows that share the same virtual link or the same card. It does so taking into account the flows' inherent bandwidth demands.

We proposed a second mechanism, called *Coordinated Congestion Control (C3L)*, that further achieves max-min fairness across the network over the unmodified 802.11 MAC layer. This serves as a proof that it is indeed possible to achieve application-layer flow fairness over the unfair 802.11 MAC protocol through purely transport-layer mechanisms. To the best of our knowledge, this is the first research effort towards providing max-min fairness over unmodified IEEE 802.11-based WMNs. We take a traffic engineering approach, and continuously adapt the bandwidth allocation for individual wireless links to the latest traffic profile. The per-link bandwidth is further divided among flows passing through it by the link's sender. Unlike some of the previous proposals, our algorithm ensures end-to-end flow fairness by taking into account the intra-flow and inter-flow dependencies. Further, we proposed a general capacity re-estimation algorithm that can effectively resolve the unfairness among wireless links as created by the 802.11 MAC protocol.

Chapter 5

Secure Routing Protocol

5.1 Introduction

The architecture and protocols discussed in the previous chapters significantly improve the performance of wireless mesh networks, both in terms of aggregate capacity and per-flow fairness. The third problem to tackle before wireless mesh networks can be embraced by enterprise users is *securing network operations*. We formulated this problem in Chapter 2, and showed why it is different from the problem of securing a mobile ad hoc network. We argued that standard cryptographic techniques do not directly help when one or more of the network routers are compromised, as the compromised routers have the cryptographic keys. We further argued that for meaningful security, both the control plane and the data plane of the network need to be secured. That is, we need to ensure that a consistent and correct routing protocol state is established across all nodes and that all nodes indeed perform packet forwarding in a manner consistent with that state.

The core Hyacinth protocol already deals with fail-stop failures by reconfiguring network topology to work around a failed node. From security viewpoint, we concern ourselves with Byzantine attacks. Specifically, we assume that an attacker is able to hack into some of the nodes and inject malicious code into the compromised nodes with the goal of subverting the network protocol and communication.

The focus of our study is a centralized network architecture, where the individual mesh nodes only implement the forwarding mechanism, while all routing algorithmic machinery is implemented on one or more centralized controller units. Given that a similar architecture is being discussed even in the context of Internet-scale routing [58], it is worthwhile to explore this centralized network architecture in the WMN context for two reasons: First, most real-world WMNs are less than 5 hops in length (measured from the wired network), and hence the infinite scalability is not required. In fact, our experiments suggest that the computation easily scales to 1000+ nodes even with a single PC-based controller. Second, efficient wireless link resource utilization is much more important and centralization can make better use of wireless links and reduce routing overhead, especially when routing security is required. Based on our research, we believe this is in fact the most practical way

to solve the secure routing problem in WMN.

Centralization simplifies the security management by limiting a typical router's function, and performing the data plane state computation on a secure centralized controller. Surprisingly, centralizing the computation only solves a part of the problem, each of the following network operations needs to be secured as well:

- *Topology and Traffic Statistics Collection:* These form the input to the routing and channel allocation algorithm running on the central controller, and need to be checked for tampering from the originating nodes.
- *Data Plane State Dissemination:* Although the computation of data plane state is centralized, the results of the computation need to be securely distributed across the network.
- *Node Reconfiguration:* Compromised nodes can disregard the reconfiguration instructions they receive from the central controller, and configure themselves arbitrarily. Such mis-configurations need to be detected.
- *Packet Forwarding:* The control communication as well as the data communication needs to be protected against tampering, including reordering and delaying. Specifically, the misbehavior needs to be detected as well as the misbehaving nodes need to be identified and isolated.

Here are some assumptions we make in this chapter:

- *Tree-based Routing:* The focus of this study is tree-based load-balancing routing protocol which form an important point in the design space for routing protocols. Lack of route redundancy in tree-based topology complicates the security problem, as it is no longer possible to ensure packet integrity by simply sending it over multiple independent paths.
- *No Collusion:* In general it is possible for nodes to collude, but the primary focus of this research is ensuring correct network operations in presence of multiple independent compromised nodes.
- *Non-Zero Attack Time:* We assume that an attacker takes non-zero time to compromise a brand new node. More specifically, we assume that a node is not compromised in the first few minutes of its operation until it is able to establish communication with a centralized server on the wired network. Of course, the initialization process, including the setup of communication with the centralized controller, is itself secured.

The chapter is organized as follows. Section 5.2 briefly presents some of our past efforts in introducing security in the context of distributed routing, and illustrates the complexities of that approach. Section 5.3 develops the centralized secure resource management protocol. Section 5.4 discusses how we secure packet communication in the network. Section 5.6 evaluates the proposed techniques using ns-2 simulations. Section 5.5.3 discusses the limitations of the proposed solution.

5.2 Replication-based Secure Routing

Given the crippling impact compromised nodes can have on the network operations, it is clear that security needs to be considered a first-class requirement at par with connectivity and performance. In this section, we discuss our previous attempt to secure our distributed protocol by introducing redundant messages and state computation. Although we believe that centralized secure routing is the most practical way to solve this problem, this section illustrate the difficulties in securing a distributed solution, and provides background for our centralized security proposal.

5.2.1 Routing State Replication

The idea behind this approach is to replicate each node X's routing state across k (say 2) of its one-hop neighbors. The neighbors replicating X's state are termed as its watchdogs W_X^i. The replication of X's routing state enables W_X^i to validate all routing protocol packets that X sends to any node in the network.

In a single-channel network, replication of state can be easily achieved by requiring X to re-broadcast all routing protocol packets it receives from the network to its watchdogs W_X^i. Each watchdog node W_X^i can then re-run the same computation as X thereby reconstructing X's routing state. Note that the re-broadcast is also received by the neighbor Y who originally sent/forwarded the packet to X. Y can therefore ensure that X faithfully (without modification, fabrication, or deletion) replicates the packets that Y sends to X. To deal with lost re-broadcast packets, X could require Y and W_X^i's to acknowledge the re-broadcast and resend the packet if such an acknowledgement is not received.

Unfortunately, the problem gets far more complicated in a multi-channel WMN. Because of multi-radio configuration, a node may not be able to communicate with all its reachable neighbor through a single link-layer broadcast packet. As each broadcast needs to be carried on multiple channels, it is easier for a node to send different messages to its different neighbors. There is no direct way for the source of the packet Y to check whether X reliably re-broadcasted each packet to all of its neighbors.

There are two possible ways to achieve reliable replication in this context –

- *Receiver-initiated Replication:* In this approach, X (the node being watched) is required to forward each control packet it receives from other network nodes to all its watchdogs. To deal with transmission losses, X uses an acknowledgement/retransmission mechanism to ensure reliability of replication packets. To prevent X from modifying, fabricating, or deleting these packets in the replication process, the sender Y stamps each packet it sends with a sequence number as well as uses its private key to digitally signs the packet. This approach clearly introduces substantial overhead as each routing packet now needs to be authenticated by both the sender and the receivers.

- *Sender-initiated Replication:* Another option is to require the sender Y to independently copy each packet it sends to destination X's watchdog nodes W_X^i. As some

of these replication packets still need to go through X, Y still needs to ensure their integrity. However, it is now possible to use symmetric key cryptography using pairwise keys established among mesh nodes. Unfortunately, this brings up the problem of ensuring that the sender Y correctly replicates the packets it sends to X across its watchdogs W_X^i.

5.2.2 Routing Packet Validation

The second question is how to use this replicated state of monitored node X to validate control packets sent by X itself. To achieve this, all of X's messages should also be received by W_X^i's. In a single-channel network, the problem could be addressed by use of RF monitoring mode on W_X^i's or requiring X to broadcast all its packets. This enables W_X^i's to compare the packets sent by X against their replica of X's routing state and raise an alarm if packets sent by X mismatch the protocol.

Again the problem is far more complex in a multi-channel setting. The modified broadcast mechanics prevent its use to validate packets sent by a the monitored node X. In a proactive approach, X needs to get each packet validated by one of the watchdog nodes W_X^i before sending it. This approach has the obvious drawback of increased traffic. One could require all radio neighbors of a node to be its watchdog and replicate its routing state. As every packet needs to go through at least one of the neighbors, it is possible to validate all transmitted packets on their first hop. However, this modification increases the state replication overhead both in terms of traffic and computation.

5.2.3 Discussion

The validation requires that each routing packet be verified and authenticated by a sender's watchdog. Further, the replication requires the packet receiver's watchdog to again verify the integrity of the packet. The computational overhead of per-packet replication and validation traffic could easily become prohibitive due to use of asymmetric key cryptography.

An attempt to remove authentication requirement from each packet leaves the packets susceptible to tampering by the sender, by an intermediate node, and by the receiver. One can somewhat alleviate this computation overhead by computing the verification bits every few packets. Although this would reduce the computation overhead, it does little to simplify the protocol.

5.3 Centralized Secure Routing

Several of the problems discussed in previous section arise because the original protocol enables individual nodes to make decisions. Our proposed architecture addresses this problem at the core by taking the decision making power away from individual nodes. Specifically, the individual nodes now only report the status of the network to a centralized controller unit (abbreviated as CU), while the controller makes the actual route selection

and channel assignment decisions. One might argue for implementing protocol computation on the gateways, as the number of gateways scale linearly with the network size in a well-designed network. However, the gateways are equally hard to secure as they need to be physically distributed across the network.

We now discuss various aspects of this centralized secure routing protocol, how it detects node compromises, and how it accurately identifies compromised nodes, and prevents them from inducing any non-local impact. We begin our discussion by presenting the basic building blocks of this protocol.

5.3.1 PKI Infrastructure

To secure data communication, a verification hash code is attached to every packet as soon as it enters the mesh network (on the ingress node). This code is used to verify the packet for any signs of tampering before it leaves the mesh network (on the egress node). This verification hash code is transparent to the end hosts, e.g. the mobile clients and the wired Internet hosts they communicate with. The verification code is computed using a session key known only to each ingress-egress mesh node pair. This in turns requires each mesh node to have the following:

- A unique public/private key pair for itself, and
- Public keys of other ingress/egress nodes it communicates with.

One seemingly simple way to establish these keys is to configure each node with this information at deployment time. However, requiring a different configuration for each node would be expensive and undesirable from management standpoint, especially for larger WMN deployments. In Hyacinth, each new node instead obtains the above configuration parameters at initialization time.

To facilitate this process, the network runs a certificate authority on the controller unit (CU) that is responsible for assigning IP addresses and certificates to new nodes. Each node is pre-configured with the public key of the CU: K_{CU}^{Public}. As this value is the same for all nodes, Hyacinth does not require a separate configuration for each node.

The basic initialization protocol works as follows. Each new node upon boot up scans all the channels to discover neighbors operating on each of them. The new node then sends a message, via one of its discovered neighbors, to CU requesting assignment of a certificate (public/private key pair). The request also contains a randomly generated session key $K_{Node}^{Session}$. To prevent eavesdropping and malicious modification, the message is encrypted using CU's public key K_{CU}^{Public}. Upon receipt of this request, the CU then issues a certificate to the new node and sends it to the node along with the corresponding private key $K_{Node}^{Private}$. The return message is encrypted using the session-key $K_{Node}^{Session}$ from the original request.

In case of a node compromise, the public key K_{CU}^{Public} of the CU would be known to the attacker. To prevent an attacker from adding more attack nodes into the network, a new node upon obtaining its certificate and the private key, erases the K_{CU}^{Public} permanently and

stores its own private key $K_{Node}^{Private}$ into permanent local store. The permanent local storage ensures that if the node is rebooted, it would not need to obtain the certificate again from the CU. An existing compromised node attempting to disrupt the PKI infrastructure can neither fabricate nor modify PKI-related initialization packets. This is because the initial packet is encrypted using the controller's public key K_{CU}^{Public}, while the subsequent packets are encrypted using the session key $K_{Node}^{Session}$, both of which are only known to the new node and the CU [2]. The only harm an existing compromised node can do is drop the PKI related packets, but in general it is not possible for the compromised node to detect and selectively drop PKI packets because of encryption. Further, it is easy for a new node to detect such a situation and switch to another neighbor for establishing communication with the CU. Thus as long as the new node has one uncompromised neighbor with an established path to a gateway, it is able to establish communication with the CU.

5.3.2 Secure Topology Collection

Accurate knowledge of network topology is not only crucial to the base protocol, but it is also useful in validating several protocol packets. Hyacinth CU is kept abreast of the entire network topology: Any changes to the topology are promptly communicated to the CU. The topology here refers to the radio topology without regards to any communication constraints resulting from the channel allocation done by the CU.

Each node periodically discovers its communication range as well as interference range neighbors, and sends this information to the CU. For faster updates, a node also sends its latest topology to the CU if detects a local topology change in between regular discovery. Local topology changes could occur if a new neighbor joins the network, an existing neighbor dies, or local radio propagation characteristics change. A packet sent by a node simply contains two lists: the first enlisting the communication range neighbors, and the second enlisting the interference range neighbors.

5.3.2.1 Communication-Range Neighbors

We modify the topology discovery protocol discussed in Chapters 3 and 4 a little by applying the technique of anonymous messages earlier proposed in [59]. Recall that to discover its communication range neighbors, a node scans all the channels. Upon scanning a particular channel, the node broadcasts HELLO packets. To ensure that its neighbors faithfully participate in the topology discovery, the HELLO packets are *anonymized* by scrubbing the source MAC address field. A neighbor hearing a HELLO packet responds with a broadcast HELLO_BACK packet. This modified neighbor discovery protocol effectively forces all neighbors, including any compromised neighbors, to reveal their true radio connectivity to

[2] As an added safeguard, the CU checks the number of nodes that actually join the network against the number of nodes that were physically deployed. Moreover, if the K_{CU}^{Public} is indeed compromised, future deployments can be configured to use a newly generated pair of public-private key limiting the impact of the compromise.

the initiating node. The reason is simple – a compromised neighbor does not know anything about the initiating node and not responding to the packet could potentially reveal its misbehavior to all its neighbors over time.

A node can still add or remove other nodes from its neighbor list. Hence, the topology information is further verified for consistency on the CU –

1. The CU marks two nodes X and Y as communication range neighbors only if X declares Y to be its communication range neighbor and vice versa. This prevents a compromised node from declaring any arbitrary node as its neighbor.

2. The CU also checks for the opposite scenario: A compromised node removing several of its neighbors from its neighbor list. This is easy to spot as all those neighbors would still report this compromised node as being one of their neighbors.

Based on the PKI infrastructure we built, it is not possible for an attacker to use a compromised node's credentials to introduce fake nodes with new identities into the network. However, an attacker could still introduce multiple clones of a compromised node into the network. Although all these fake nodes have the same identity as the original compromised node, they can have different code running on them. This *node cloning attack* can thus be used to tamper with the final topology as perceived by the CU. In Hyacinth, we use a simple threshold-based scheme where if a node has too many neighbors, then it is considered to be potentially compromised. This approach works because a typical mesh node X only has a small number of neighbors. For a compromised node C, each clone C_i ends up revealing all its radio neighbors C_{ij}. This makes it easy to spot a compromised node that has been cloned.

5.3.2.2 Interference-Range Neighbors

A similar solution, however, does not work for interference range neighbors. In general, X may interfere with Y without Y interfering with X. In case of compromise, Y can use this fact to declare a large number of nodes to interfere with itself. This could impact the channel assignment that attempts to assign interference-free channel to each communicating node pair. One possibility is to use a simple threshold based mechanism. If a node Y declares that it is observing asymmetric interference from too many other nodes, then Y could be declared as compromised.

Unfortunately, asymmetric interference is not uncommon and could arise based on the node placements and settings of transmit power and receive sensitivity. For example, if the receive sensitivity is set high, then the node could observe interference even from distant nodes. If simultaneously the transmit power is set low, then the node could only communicate with nearby neighbors. In Hyacinth, we solve this problem by using fractional values of interference. Specifically, we no longer treat interference as a binary phenomenon. If two nodes X and Y both observe interference from each other, then this interference contributes a value of 1 to the total interference on the channel. However, if only X reports

interference from Y, and Y does not report any interference from X, then this interference value only contributes 1/2 to the total interference on the channel. This fractional counting limits the impact of a compromised node reporting false interference from its non-interfering nodes or not reporting interference from its interfering nodes.

5.3.2.3 Gateway Nodes

Finally, a compromised node could pretend to be a gateway node enticing other nodes to connect to the wired network through it. To prevent this, we require that each gateway node carries a proof that it is indeed a gateway. The proof is embedded as part of the node's certificate and thus needs to be obtained at initialization time. When a node sends request for certificate, it also tells the controller unit (CU) by looking at its configuration if it is a gateway, and the CU prepares the certificate accordingly. Note that it is not possible for a compromised node to pretend to be a gateway as it cannot modify (or request modification of) its certificate once it is issued (See discussion on K_{CU}^{Public} in Subsection 5.3.1). Since it is not possible for an attacker to introduce fake nodes into the network, it cannot introduce fake gateway nodes either.

To prevent malicious modifications and eavesdropping, all control packets exchanged between the CU and a node X are encrypted using the session key $K_X^{Session}$ chosen by the node. Control packets are subject to drops by the network nodes. Reliable in-order delivery is ensured through an acknowledgement and retransmission mechanism. In the rest of this section, we assume that the control packet exchange between CU and any given node is reliable and secure. We will revisit the issue of reliable packet communication in much more detail in the next section.

5.3.3 Secure Traffic Statistics Collection

The second piece of information that CU needs for route/channel computation is the WMN traffic reports from all nodes in the network. Each node reports to the CU (1) the traffic that it (or more specifically the end users connected to it) generates in either direction, and (2) the traffic that it routes for other nodes again in either direction.

To disrupt network operations, a compromised node can send falsified traffic reports to the CU. However, it is possible to correlate traffic reports from different nodes to detect inconsistencies across them and pinpoint such a node. If the measurement intervals of various nodes are unaligned, then performing such correlation becomes a non-trivial problem. To simplify the problem, we synchronize the clocks on all the network nodes to the clock on the CU using an existing clock synchronization algorithm.

Next, each node measures its traffic values at pre-defined intervals and reports them back to the CU. The CU uses a simple logic to detect inconsistencies among these reports. Specifically, the following equalities capturing conservation of traffic should hold on every node –

$$Load_{Out}^{Up} = \sum_{i} Load_{In_i}^{Up} + Load_{Gen}^{Up} - Load_{Con}^{Up} \quad (5.1)$$

Here, $Load_{Out}^{Up}$ is the aggregated upstream traffic load going out of the node towards its parent, and $Load_{In_i}^{Up}$ is the upstream load coming into the node through its i^{th} down-link. $Load_{Gen}^{Up}$ and $Load_{Con}^{Up}$ are the aggregated upstream load generated and consumed respectively by the wireless clients' associated with the node. Similarly for downstream load,

$$Load_{In}^{Down} = \sum_{j} Load_{Out_j}^{Down} + Load_{Con}^{Down} - Load_{Gen}^{Down} \quad (5.2)$$

where $Load_{In}^{Down}$ is the downstream load coming into the node through its up-link, and $Load_{Out_j}^{Down}$ is the downstream load going out through its j^{th} down-link. $Load_{Con}^{Down}$ and $Load_{Gen}^{Down}$ are the downstream load consumed and generated respectively by the node's wireless clients.

Assuming no collaboration between compromised nodes, the quantities $Load_{Out}^{Up}$, $Load_{In_i}^{Up}$, $Load_{In}^{Down}$, and $Load_{Out_j}^{Down}$ reported by any node can be easily cross-checked by comparing them with values reported by the node on the other sides of the corresponding links. This makes it hard for an individual compromised node to falsify these values, and in turn even the values of $Load_{Gen}^{Up}$, $Load_{Gen}^{Down}$, $Load_{Con}^{Up}$, and $Load_{Con}^{Down}$.

5.3.4 The Routing And Channel Optimization Algorithm

The actual channel allocation and the route selection are done by an incremental greedy algorithm that emulates the multi-spanning tree distributed algorithm. The centralized architecture also enables application of other more optimal algorithms, e.g. the one proposed in Chapter 3. Our current algorithm visits all the *trees* in the decreasing order of aggregated traffic loads they impose on their respective gateways. Upon visiting a tree, the algorithm offloads part of its traffic by moving one of its subtrees to a neighboring tree with lesser load. It then similarly optimizes the traffic load across links with each gateway. Within each subtree rooted at a gateway's link, the algorithm simply attempts to minimize the hopcount upto the gateway. For optimizing channel allocations, the algorithm estimates the channel contention around each node, and switches nodes facing maximum channel contention to other locally idle channels. The nodes higher up in the trees are given higher priority as compared to the ones lower down, resulting in a fat-tree based network.

This CU-based algorithm is detailed below.

1. Among all the trees under consideration, find the tree T^I whose aggregated traffic load is maximum (*i.e.* $\forall_i, Load(T^i) \leq Load(T^I)$).

2. Determine all of T^I's neighboring trees T_j^I that can potentially take over some of its traffic load. Two trees are considered neighbors if there is at least one subtree

(possibly composed of just a single node) that can migrate from one tree to the other without involving any other tree.

3. For each candidate neighboring tree T_j^I, determine all subtrees T_{jk}^I that can migrate from T^I to T_j^I without increasing T_j^I's final aggregated load beyond T^I's final aggregated load. In other words, the following inequality should hold for all T_{jk}^I.

$$\forall_j \forall_k, Load(T^I) - Load(T_{jk}^I) \geq Load(T_j^I) + Load(T_{jk}^I) \tag{5.3}$$

4. For each neighboring tree T_j^I, pick the subtree T_{jK}^I such that $Load(T_{jk}^I)$ is maximized over all k (*i.e.* $\forall_k, Load(T_{jk}^I) \leq Load(T_{jK}^I)$).

5. Now pick one neighboring tree T_J^I out of all the candidates such that the $Load(T_{JK}^I)$ is maximized over all j (*i.e.* $\forall_j, Load(T_{jK}^I) \leq Load(T_{JK}^I)$).

6. As the migration results in increased load on the neighboring tree, recompute the imposed load on the gateway for the neighboring tree.

7. Add $[T_{JK}^I : T^I -> T_J^I]$ to the list of route changes to be performed next.

8. Remove the tree T^I and T_J^I from consideration, and go back to step 1. While one can consider the two trees for further optimization in the subsequent iterations of the loop, the actual effects of moving T_{JK}^I from T^I to T_J^I are hard to estimate due to statistical multiplexing of traffic.

After the algorithm has gone through all the trees once, it has a list of routing changes of the form $[T_{JK}^I : T^I -> T_J^I]$ that need to be executed. The same algorithm is applied at two different levels: first across gateways to balance the inter-gateway traffic load, and next across individual links of each gateway to balance intra-gateway inter-link traffic load. The subtrees rooted at individual links of each gateway are internally re-balanced for hopcount by using a slight variation of Bellman Ford algorithm.

Next, another algorithm is run to optimize channel assignment in the network. The route optimization related changes from previous step will impact the traffic load across different channels. However, due to statistical multiplexing of traffic, the resulting traffic load on individual channels is hard to estimate. The channel optimization step therefore excludes any node that might see a change in channel traffic in its vicinity. This set is approximated as $\cup_i Intf(N_i)$, where N_i is a node whose routing table will change as a result of route optimization, and $Intf(N_i)$ is the set of nodes that interfere with N_i. Note that many of the nodes in the set can still undergo constrained channel optimization. However, excluding such nodes altogether from the channel optimization computation simplifies the algorithm.

The actual channel assignment algorithm works as follows –

1. Let U be the set of candidate nodes for channel assignment optimization (*i.e.* nodes $\notin \cup_i Intf(N_i)$). Then, for all nodes $N_j \in U$, determine $CC(N_j, i)$ which is the aggregated load seen by node N_j on channel i including the load N_j imposes on channel i.

2. Determine the node N_J and channel I such that $CC(N_J, I)$ is the maximum contention seen by any candidate node on any channel. Formally, $\forall_j \forall_i, CC(N_J, I) \leq CC(N_j, i)$.

3. Determine the least-loaded channel K from the perspective of node N_J, *i.e.* $\forall_k, CC(N_J, K) \leq CC(N_J, k)$.

4. Determine $CC(N_l, K)$ which is the maximum load seen on channel K by any node in vicinity of node N_J.

5. If $CC(N_J, I) - L(N_J) > CC(N_l, K)$, then moving node N_J from channel I to channel K would result in an overall balancing of traffic across channel I and channel K. If so, add $[N_J : I- > K]$ to the list of channel changes to be done next.

6. Remove $Intf(N_J)$ from the the set U and go back to step 1.

After the algorithm has gone through all the candidate nodes, it has another list of channel changes of the form $[N_J : I- > K]$ that need to be executed.

5.3.5 Network Re-configuration

Once the controller has determined a set of route and channel changes for the next round of network re-configuration, it communicates with individual nodes to execute these changes. Specifically, the controller sends each node the following info –

1. The time at which the next reconfiguration is to be performed ($t_{now} + \delta$).

2. A list of routing table changes that the node needs to make.

3. Any channel change that the node needs to do.

4. Any change in the UP-NIC association.

To deal with packets losses, say due to bit errors, the delivery of each packet is made reliable by using an explicit acknowledgement and retransmission mechanism. To avoid situation when some of the nodes have received their reconfiguration packets and performed reconfiguration, while the reconfiguration packets to the remaining nodes are still being retransmitted, the δ is kept sufficiently large.

Network reconfiguration inevitably results in a transient increase in packet delays as well as few packet losses. Hyacinth synchronizes the network reconfiguration to minimize such delays and losses. All network synchronize their clocks with the CU using a simple time synchronization algorithm based on average RTT difference, which is fully piggy-backed on regular communication between each node and the CU (e.g. traffic updates). It is possible to use any other more sophisticated algorithm that synchronizes the the local clocks of nodes across the network. The only constraint is that the each node needs to

synchronize directly with the CU, rather than with another node to prevent clock synchronization attacks. While it is impossible to perfectly synchronize the time, this technique minimizes the transient interval when nodes might have packet losses. Furthermore, the link layer is able to tackle the retransmissions of most of these lost data packets making the channel switching almost transparent to the higher layers.

Ideally, we would like to detect if each node indeed *executes* the reconfiguration commands that the CU sends it. If a malicious node does not perform reconfiguration, one or more connections passing through the node would observe packet losses. However, it is always possible for a malicious node to faithfully perform the reconfiguration and *still* mishandle the packet forwarding. We therefore deal with all routing misbehavior by observing the packet forwarding behavior of nodes. Routing misbehavior could be in form of excessive packet drops, in-transit modification of packets, injection of fabricated or duplicated packets, artificial delaying or re-ordering of packets, or mis-routing of packets. The next section discusses how we detect such routing misbehavior, pinpoint the misbehaving nodes, and isolate them from rest of the network.

5.4 Securing Packet Communication

All through the discussions in the previous section, we assumed that the control packet exchange is both reliable and secure. We now turn our attention to this very problem of securing end-to-end packet communication over multiple hops. Secure packet communication is one of the central primitives needed for each of the following functions –

1. Reporting of topology and traffic statistics to CU.
2. Distribution of routing and channel decisions across the network.
3. Network time synchronization.
4. Data exchange over the mesh network.

Note that the last point covers both kinds of packet exchanges – *control packets* and *data packets*. We have already discussed the kinds of attacks that we concern ourselves with in this chapter. For ease of exposition, here are the attacks we would like to deal with as far as end-to-end communication is concerned -

- Excessive packet drops.
- In-transit modification of packets.
- Injection of fabricated packets.
- Injection of duplicate packets.
- Artificial delaying of packets.

- Packet re-ordering.
- Misrouting of packets.

We tackle the problem of securing end-to-end packet communication in 3 steps: detect attack, identify attacker, and isolate attacker. It is insufficient to simply detect the attack as the attacker can degrade the network performance and worse yet make the network unusable if not identified and isolated from the rest of the network.

5.4.1 Attack Monitoring

Hyacinth incorporates various mechanisms to enable ingress/egress node pairs to detect any misbehavior on part of compromised nodes along the path. As mentioned earlier, the mechanisms are transparent to the end hosts exchanging the data packets, and are independent of any end-to-end encryption that the end hosts employ.

5.4.1.1 Monitoring Packet Forwarding

One way a compromised node can make the network unusable is by resorting to packet misrouting (or more precisely mis-forwarding). One could detect packet mis-forwarding by providing each node with a copy of its neighbors' routing tables and channel assignments. This would make it possible for each network node to look at the (src, dest) pair in a packet header and determine if it should indeed have received this packet from its previous hop.

In Hyacinth, this problem is much simplified because of the tree structure. Specifically, when a malicious node M mis-forwards a packet, the next hop node N would attempt to send the packet back to M. Thus, a mis-forwarding attack is easily detected by comparing the previous hop of each packet with its next hop. If they are the same, then a potential mis-forwarding attack is signalled.

5.4.1.2 Transparent Packet Authentication

As most sensitive traffic traversing even the wired network is protected against tampering and eavesdropping by end-to-end encryption, we expect the same to be true for wireless mesh networks. One could simply use this end-host encryption to also monitor signs of packet tampering by compromised mesh routers. While clearly useful for detecting signs of packet tampering, reliance on end-host encryption alone to pinpoint compromised routers requires excessive changes to the end-host networking stack. Hyacinth adopts a more end-host transparent monitoring and detection of compromised nodes. Each packet exchanged over the Hyacinth network contains a sequence number and a comprehensive packet signature computed using a well-known hash function but encrypted using a session key. The sequence number and the packet signature are computed/inserted at the ingress of the wireless mesh and verified/removed at the egress of the wireless mesh, all transparent to the communicating applications and the end hosts. The session key itself needs to be unique

only to a pair of ingress/egress mesh nodes. Multiple end-host connections passing through the same ingress/egress node pairs thus use the same session key.

Key Establishment: To reduce the initialization time of a node, the session key is established in a lazy manner. Establishment of session key is relatively straightforward because of the PKI infrastructure we established. Suppose, node X and node Y need to establish a session key $K_{X,Y}^{Session}$. The node initiating the communication (say X) picks a random key to be $K_{X,Y}^{Session}$, encrypts it with the K_Y^{Public} of Y , and sends it to Y. Only the receiving node Y can decrypt the packet to obtain the $K_{X,Y}^{Session}$. As such this scheme would allow a compromised node M to launch a man-in-the-middle attack by establishing a separate session key $K_{X,M}^{Session}$ with X pretending to be Y, and $K_{Y,M}^{Session}$ with Y pretending to be X. To prevent such an attack, the initial packet sent by X also contains a message of the form $(MSG.Y)$ first encrypted with the $K_X^{Private}$ and then encrypted with the K_Y^{Public}. The second level encryption using K_Y^{Public} ensures that the message can only be decrypted by Y. The first level encryption of MSG provides a proof to Y that X indeed generated this packet. Finally, appending Y to the message MSG ensures that Y cannot use it to establish a session key with another node on behalf of X.

No Fixing: Use of ingress-egress sequence numbers and packet signatures enables the egress to detect duplicated, fabricated, modified, dropped, and re-ordered packets. Duplicated, fabricated, and modified packets are dropped on the egress. It is possible to trigger retransmission to fill missing sequence numbers, and also to fix any re-ordering of the packets before releasing them out of mesh egress. However, the packet stream would typically contain packets from multiple end-user connections. Reordering of packets on ingress-egress packet stream might still maintain correct packet ordering *within* individual connections. Similarly, requesting retransmission of dropped packets might be unnecessary for multimedia applications that can tolerate missing packets but not late packets. Further, even for non-latency sensitive applications, the end host-level or application-level retransmission might triggered at the same time as we request mesh-level retransmission. To simplify the ingress/egress mesh node logic, we limit the use of sequence numbers to simply detect and record packet reordering and packet losses.

Inferencing: Packet modifications almost always signal an attack, as genuine modifications in form of packet corruption should be detected at hardware layer itself leading to packet drop. The same goes for packet fabrications, which should only happen if a node fails or is otherwise compromised. However, packet drops can also happen because of the underlying network properties. Bad channel quality at one of the hops could lead to packet drops, or simply the queue build up on an intermediate hop could force it to drop packets. With the stock 802.11 link-layer retransmission, duplicate packets resulting from 802.11 ACK drops are discarded. Further, there is no possibility of packet re-ordering at the link-layer. However, local software retransmissions scheduled by link-layer aware transport protocol (LRTP) could result in some packet duplication and re-ordering. Therefore, only when the packet drops, duplications, and re-ordering go beyond certain reasonable thresholds, does the receiver signal an attack necessitating detailed monitoring. Attacks within the thresholds would go undetected, but they are also not serious enough to have any significant impact on the network.

Due to differences among various attacks and the probabilities of raising false alarms in each case, Hyacinth uses a different *detection threshold* for each kind of attack. All the detection thresholds are relative to a *monitoring window*. More precisely, this is the window of recently received packet (from an ingress node) for which an egress node maintains metadata info and which it looks at for signs of attacks.

Impact of Network Reconfiguration: The packet authentication and monitoring are carried independently of any reconfiguration of the network. The session key and the increments of sequence number is between an ingress-egress node pair, and is independent of the actual path taken by packets between them. Even when the route between that ingress-egress node pair changes as a result of reconfiguration, the session key stays the same and the sequence number does not get reset. Any impact on the packet stream, e.g. packet drops, duplications, reordering, and delays, as a result of network reconfiguration are accommodated by the corresponding detection thresholds.

5.4.1.3 Path Delay Measurement

A similar logic applies to packet delays. It is possible for packets to be genuinely delayed because of MAC contention, local retransmissions after multiple errors, or queueing. It is fundamentally hard to distinguish such delays from artificially introduced delays by malicious nodes to impact network performance. However, it is fairly straightforward to measure the RTT of packets (including any introduced delay by malicious nodes), and signal an attack when this RTT goes above certain threshold. To measure the RTT, the receiver (the egress node) periodically sends a SYNC packet with local timestamp (T_{SYNC}^{egress}) to the sender (the ingress node). The sender records the local timestamp when it is receives this SYNC packet ($T_{SYNC}^{ingress}$). From then on, every data packet sent by the sender includes (1) local sending time for the packet $T_i^{ingress}$, (2) $T_{SYNC}^{ingress}$, and (3) T_{SYNC}^{egress}.

The receiver records the local timestamp every time it receives the data packet (T_i^{egress}). Using these numbers, the egress can determine the actual RTT of the i^{th} packet (RTT_i) using the following equation.

$$RTT_i = T_i^{egress} - T_{SYNC}^{egress} - (T_i^{ingress} - T_{SYNC}^{ingress}) \tag{5.4}$$

The above equation can be rewritten as:

$$RTT_i = T_i^{egress} - T_i^{ingress} - (T_{SYNC}^{egress} - T_{SYNC}^{ingress}) \tag{5.5}$$

As an optimization the ingress only maintains and sends the 'reverse one-way delay' $T_{SYNC}^{egress} - T_{SYNC}^{ingress}$ and the local sending timestamp $T_i^{ingress}$. The egress combines those with the T_i^{egress} to obtain the RTT for the packet. The SYNC packets are piggybacked with regular packets, and thus typically do not incur any additional overhead. This also enables us to keep the SYNC period sufficiently small, and thus ignore any clock drift incurred during this time [T_{SYNC}^{egress}, T_i^{egress}].

An important concern in path delay measurement is that the local clocks on individual nodes can skip forward/backward when it undergoes re-synchronization with the clock

on CU. To accommodate these periodic clock re-synchronization, each node maintains a *rolling synchronization counter* that is incremented every time the clock is re-synchronized. Any time a node puts its local timestamp on a packet, it also puts its current synchronization counter on that packet. In the above case, the egress sends its synchronization counter C_{SYNC}^{egress} along with the T_{SYNC}^{egress} to the ingress node. The ingress node, records the synchronization counter C_{SYNC}^{egress} along with the reverse delay $T_{SYNC}^{egress} - T_{SYNC}^{ingress}$ for the ingress-egress session.

Any time the local clock on the ingress is reset, it also adjusts the value of reverse delay (for all its sessions) accordingly. Further, when the ingress sends a packet back to the egress, it sends C_{SYNC}^{egress} along with the reverse delay stored with the session. Upon receiving this packet, the egress uses the reverse delay stamped on the packet to compute RTT only if the C_i^{egress} matches the C_{SYNC}^{egress}. It is also possible to optimize this further by maintaining the adjustment done to the local clock on egress node upon every clock resynchronization, and accounting for these while calculating RTT if the C_i^{egress} does not match C_{SYNC}^{egress}.

5.4.1.4 Splitting Gateway Functionality

The above-discussed packet authentication and delay estimation rely upon mesh ingress and egress not getting compromised. A compromised ingress could tamper with the packets before inserting the packet signature. Similarly, a compromised egress could tamper with the packets after removing the packet signature. Thus, if a mesh node gets compromised, it can tamper with all the user connections starting from its associated access point. The problem is much worse if a gateway node is compromised instead. A compromised gateway node could tamper with all the connections starting from any of the access points associated with any of the mesh nodes in its subtree. One might argue that similar to the CU, gateways are special nodes and should be physically secured. However, unlike CU that resides solely on the wired network (and hence can be physically secured), gateway nodes need to connect wirelessly to other mesh nodes, and hence need to be placed in more open space.

To deal with gateway compromises, we move the security-related functionalities out of the gateway nodes into a set of physically-secured verifier nodes deployed on the wired network. Figure 5.1 depicts this architecture. Each gateway node is associated with one verifier node, that takes over all the security-related functionalities. The purpose is to make each gateway just another node in the network, such that its compromise only has local effect.

Let us work through an example from Figure 5.1. Suppose the client C_2^1 is communicating with some host H on the wired network. A downstream packet destined to C_2^1 is first routed to V_A. V_A attaches a verification code to the packet before forwarding it to G_A. The packet then gets routed over the mesh network all the way to M_2. After verifying the packet, M_2 strips off the verification code and hands over the packet to the associated access point AP_2, who forwards it to the client C_2^1. An upstream packet sent by C_2^1 gets the verification code attached at M_2 and striped off at V_A. If G_A is now compromised, it cannot affect the $C_2^1 - H$ communication without getting detected by V_A or M_2. Note that although G_A has a connection to the wired network, it still cannot bypass V_A's checks (by

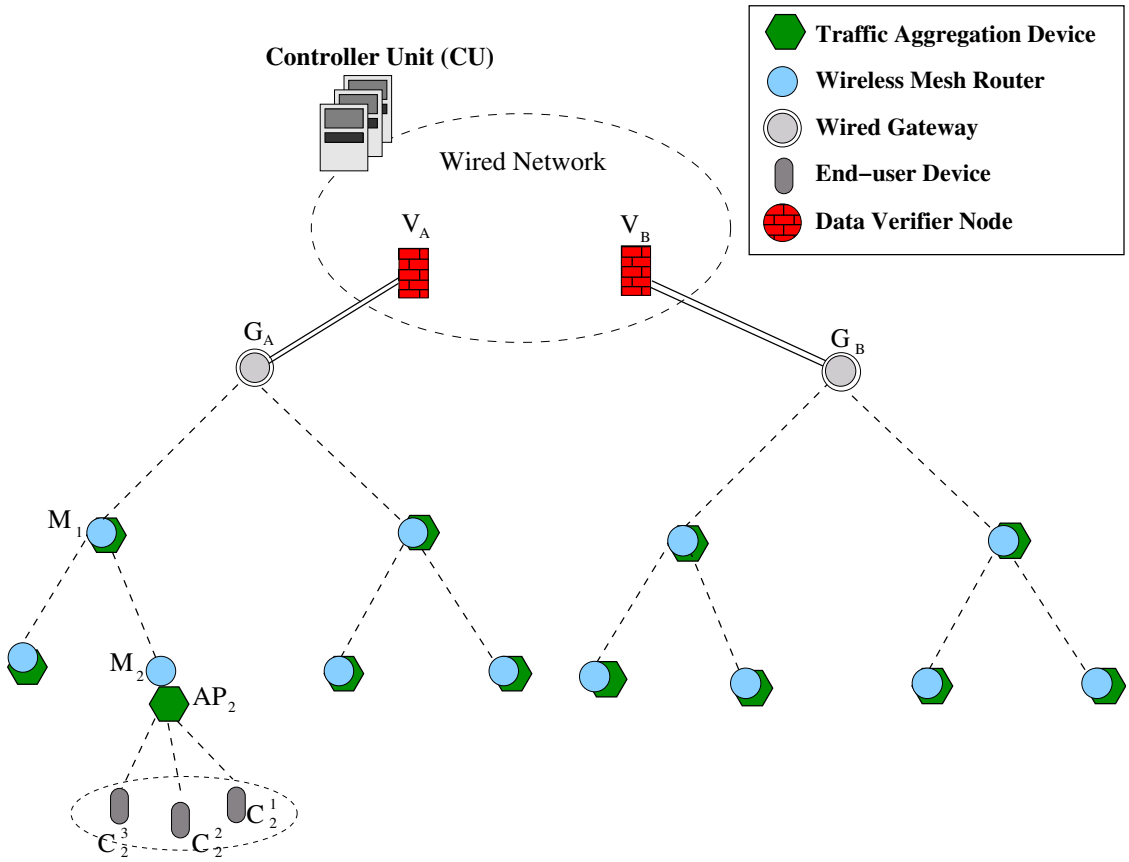


Figure 5.1: *The security-related functions are moved from the gateway nodes onto a set of verifier nodes. As verifier nodes can be physically secured, this architecture significantly limits the impact of compromised gateway nodes.*

directly sending the packets to the Internet host without going through V_A), without getting detected.

The architecture has additional hardware and networking cost due to introduction of verifier nodes. The number of verifier nodes is proportional to, but smaller than, the number of gateway nodes. As the number of gateways nodes itself is much smaller compared to the number of mesh nodes, this added hardware and networking cost is justifiable.

5.4.1.5 Client-based Reporting of Ingress/Egress Compromise

Addition of verifier nodes ensures that a compromised gateway cannot tamper with packets without getting noticed. As shown earlier, a compromised node, say M_1 cannot tamper with packets passing through it without getting detected. However, a compromised node, say M_2, can still tamper with the connections *originating from its associated access point* and not get noticed by the network. Note that the impact of such attack is limited to the clients connected to the associated access point, in this case AP_2. To determine such attacks, we fallback on clients to report such issues to the CU. More concretely, as most real-world

communication employ an ACK mechanism at application, session, or transport layers, the end clients could still detect these attacks. The CU runs a service where clients can report problems with the network connectivity. Upon receiving these reports, CU can initiate manual or automated troubleshooting, and ingress compromise is considered as one of the potential causes.

5.4.2 Malicious Node Identification

While the mechanism discussed in the previous subsection can detect tampering of packet stream by a malicious node, it does little to pinpoint such malicious nodes or isolate them. A malicious node can still impact the network routing performance by repeatedly tampering the packet stream of a connection. To pinpoint such nodes, Hyacinth incorporates a *fine-grained packet tracing* mechanism. To minimize performance impact, this fine-grained packet tracing is only turned on when a pair of communicating mesh nodes (typically a mesh node and a verifier node) observe mishandling of their exchanged packets. The core idea is that the receiver node searches for the first victim node V along the path that observes mishandling of packets. As shown later in this subsection, the compromised node is either the previous hop of this victim node V or the victim node V itself.

5.4.2.1 Fine-grained Packet Tracing

Let us work with the example of a packet modification attack shown in Fig 5.2. Suppose the receiver node R detects that the packet stream it is receiving from the sender node S is being tampering with. R gets the information about the route to S from the CU, and then performs the search for the malicious node by picking a different intermediate node to snoop on the packets in each iteration. Upon picking an intermediate node I, R requests I to maintain a running history of the last several packets it forwards for connections between the sender S and the receiver R. This window of packets cached by I is called the *Snoop Window*. If a particular sequence number is repeated, the intermediate node I only stores first copy of the packet it receives. Next time, the receiver R detects malicious modifications of a packet, it asks the node I to send back its stored copy of the packet [3]. Node R matches this copy against the copy of the packet it received. If the copies match, then the node V lies between S and I both inclusive. On the other hand, if the copies mismatch then the node V lies between I and R both inclusive.

To pinpoint node V, the receiver node R performs a binary search along the path from S to R. If $[S, R]$ represents the path from S to R, then the receiver S maintains $[P, Q]$ a sub-path of $[S, R]$, such that node V lies along the sub-path $[P, Q]$. In each step of the search, R picks a new node I that lies mid-way along the path $[P, Q]$ for its packet tracing. At the end of the step, S would narrow the sub-path where V lies to either $[P, I]$ or $[I, Q]$.

Claim 1: Either node V or its previous hop is malicious.

[3] One subtlety here is that the receiver R needs to wait for for some time, termed *Snoop Wait Time*, before it *can* ask the snooper for packets. The snoop wait time is measured from the time R asks I to start snooping. It cannot be 0, because of the communication and setup delays incurred for snooping.

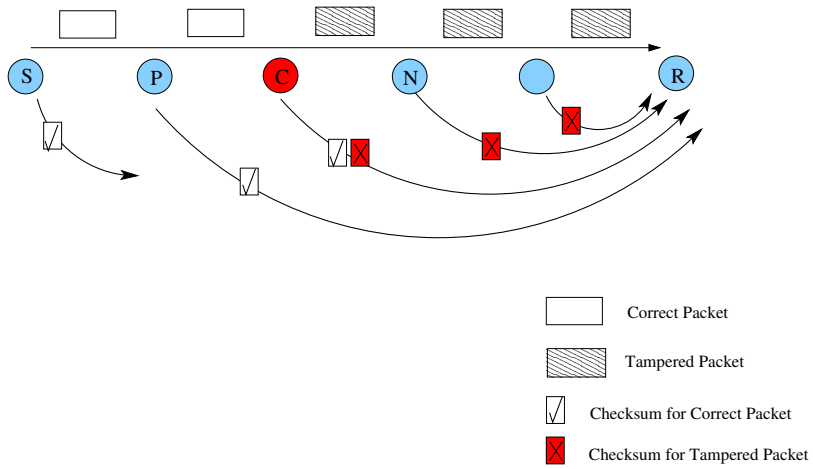


Figure 5.2: *Without correlating inferences from different paths, it is only possible to pinpoint the misbehavior to a link rather than to a node. This is because the compromised node C has the choice of returning the checksum from the original correct packet or the checksum from the tampered packet. In the former case C and its next node N are labeled as malicious, while in the latter case C and its previous node P are labeled as malicious.*

If node V's claim is true, then the previous hop of V, say U, is the malicious node. However, since a malicious node can always claim to have received the tampered copy from U, it is also possible that V itself is the malicious node. It is easy to show that either U or V is definitely malicious. If both U and V were good nodes, they should either both report tampered copy or both report untampered copy of the packet. Hence at least one of them is malicious.

Claim 2: It is not possible to detect a malicious node along a path with more precision without correlating inferences about malicious node from different paths. This is true because with any scheme it is impossible to distinguish between the two cases – the case where the node claiming to be the victim is indeed the victim and is receiving corrupted packets from its previous hop; and the case where the node claiming to be the victim is the malicious node itself and is corrupting the original packets it receives from its previous hop. The two scenarios are depicted in Figure 5.2.

Note that it is also possible for sender to do the same search. However, that would require the receiver to inform sender every time it observes tampering of packets. We therefore implement the tracing mechanism on the receiver.

5.4.2.2 Detecting Node Delaying Packets

The above technique works not only for packet modification, but also for packet fabrication, packet drops, packet duplication. A more sophisticated malicious node could faithfully forward the packets, but add excessive delay thereby impacting end-to-end network performance. However, it is easy to extend the proposed mechanism to detect artificial delays that go beyond certain normal threshold. The extension works as follows.

Each packet is stamped with origin timestamp by the ingress node (as discussed previously). The egress node looks at the timestamp and determines the end-to-end delay for the packet. If the egress node observes abnormally high delay for several or all of its packets, it and the sender initiate a binary search to determine the first node V that claims to receive an affected packet at a delayed time. The mechanism to determine partial route delay is almost the same. The receiver R runs a binary search for first node V that claims to have seen the delayed packet. The receiver R maintains the subpath $[P, Q]$ such that node V lies on this subpath (endpoints inclusive). In every iteration, it narrows this subpath by (1) randomly choosing a node I lying on the subpath, (2) sending it a SYNC packet, and (3) requesting node I to maintain receive timestamps for the recent window of S-R packets it stores. Every time, node I observes a packet whose end-to-end delay crosses the preset threshold, it asks the receive timestamp for the packet from node I. Node R then uses a similar equation as above to determine whether the delay seen by the packet can be attributed to $[S, I]$ or $[I, R]$. Based on the result, it narrows its search for V to either the subpath $[P, I]$ or the subpath $[I, Q]$. Finally, based on the same logic as packet tampering, the node V and its previous hop are reported to the CU.

It should be noted that the clock itself can skip forward/backward (due to periodic resynchronization) on the snooper node I or on the egress node R. The same mechanism of rolling synchronization counters, as discussed in Subsection 5.4.1.3, is used to account for these clock adjustments.

5.4.2.3 Impact of Network Reconfiguration

When the egress node R is running the search for the malicious node, it is possible for the network route from R to S (the ingress node) to change. When this route change happens, R's fine-grained packet tracing is restarted. Based on the explicit route information request CU receives at the beginning of any fine-grained packet tracing, the CU sends a route-invalidate packet to the appropriate egress node. More specifically, the CU maintains a list of any egress node to which it has sent route information, and during network reconfiguration uses this list to determine if any egress node needs to have its cached route information invalidated. Upon receiving a route-reset request from CU, the egress node (in this case R) explicitly requests the route again and continues the search if necessary. It utilizes any inferencing done before route invalidation to make the search over the new route faster.

5.4.2.4 Correlating Tampering Data from Multiple Sources

If a compromised node is tampering with packets along multiple paths, it is also possible for CU to correlate the inferences from each of these paths and accurately pinpoint the compromised node.

The fine-grained tracing does not get triggered if the ingress or the egress itself is tampering with the packets. The ingress (or egress) could tamper with the packets before inserting (or after removing) the packet signature. As mentioned earlier, the scope of such attacks is limited to the clients of the access point directly connected to the ingress (or egress), and we rely upon clients to inform us of such tampering. If enough clients inform about tampering that is otherwise undetected by Hyacinth's security mechanisms, then we pinpoint it to the mesh node directly connected to the access point.

5.4.3 Malicious Node Isolation

To isolate a malicious node, say M, the CU revokes its certificate by informing all nodes in the network. Upon receiving this certificate revocation, nodes in the network break their communication with the malicious node. As these packets are encrypted using the corresponding session key that is established between the CU and the recipient, it is not possible for a malicious node to fake/tamper these control packets without getting detected. Because of tree structure several network nodes could be connecting through M, and M might prevent delivery of such packets. However, such condition will get detected by the downstream nodes as their packets will not be forwarded. If so, they could start searching for another parent who has a path to the wired network without going through M.

5.4.4 Performance and Cost Impact

The proposed mechanism to detect payload tampering imposes the overhead of computing and verifying signature on a per-packet basis. On the other hand, the fine-grained packet tracing is only done when a compromised node is suspected. It is possible for the malicious node to stop tampering with the packets when it knows that the packet tracing is ON. However, this is fine as the fine-grained packet tracing only imposes modest storage requirements on the intermediate node. Moreover, as the control communication is encrypted, it is hard for the compromised node to figure out if the tracing is indeed ON at certain point in time. The malicious node thus either needs to stop packet tampering or get detected by the network.

From the cost viewpoint, this architecture introduces verifier nodes in number smaller than, but proportional to, the number of gateway nodes. As the number of gateways nodes itself is small compared to the number of mesh nodes, this added hardware and wired networking cost is justifiable.

5.5 Security Analysis

We now analyze how our proposed security architecture addresses each of the individual attacks.

5.5.1 Data Plane Attacks

To recapitulate, in a data plane attack a malicious node tampers with the packets (data packets and/or control packets) at the time of forwarding.

- **Packet Fabrication:** Every packet traversing a Hyacinth network bears a hash computed based on the *entire* packet content and encrypted using a session key only known to the ingress-egress node pair. Therefore a fabricated packet inserted by an intermediate node is easily detected at the egress. Further, as the packet hash is computed based on the entire packet content including the sequence number, it is not possible for a compromised node to fabricate a packet by utilizing 'parts' of an actual exchanged packet.

- **Packet Duplication/Replay:** Every packet sent from an ingress mesh node to an egress mesh node also bears a sequence number that increments uniformly. A packet replay by an intermediate node would thus get detected. It is possible for a packet sent by mesh ingress to get *genuinely* duplicated due to loss of an 802.11 ACK and a corresponding hop-level retransmission. If the retransmission is carried by 802.11 itself, the duplicate packet is automatically detected and suppressed by 802.11 using its own sequence numbering. It is straightforward to do similar suppression of duplicated packet if the hop-level retransmissions are instead triggered at software layer by LRTP.

- **Packet Modification:** As discussed above, the packet hash helps detect any malicious modifications done by an intermediate node. It is possible for a packet to get corrupted due to wireless bit errors. However, such corruptions are almost always detected and dropped by 802.11 layer using its own checksum mechanism.

- **Packet Drops:** The sequence number mechanism employed by the ingress-egress node pair also helps detect malicious packet drops by an intermediate node. Packets can also be dropped due to repeated wireless bit errors or queue overflow on intermediate routers or route changes. The egress node accounts for such errors by tracking their duration and frequency of re-occurrence. If they are transient and do not occur too frequently, they are considered acceptable. On the other hand, if such genuine packet drops last for long and/or occur too frequently, Hyacinth's detection mechanism treats them similar to ones intentionally induced by a malicious node.

- **Black Hole Attack:** It is possible for an attacker to just drop majority or all of the packets making it impossible for the ingress-egress node pair to communicate. In

such case, Hyacinth relies on the (absence of) periodic SYNC packet from the egress node to the ingress node, and triggers the attacker identification mechanism on the ingress node instead.

- **Packet Reordering:** A compromised node can impact network performance by reordering the packets. The sequence numbers associated with the packets are used to determine the extent of reordering. The *extent* of reordering for a given window of packets is measured by summing the differences between actual and expected packet sequence number for each position in the window. Some reordering of packets can occur due to LRTP software retransmissions and its eager 802.11 queueing needed to maintain optimal link performance. Similarly, packets can get reordered during route changes. However, such reordering is infrequent and/or small in extent. Hence, when the extent of reordering goes above certain minimum threshold and repeats frequently, it is considered to be malicious.

- **Packet Mis-forwarding:** A Hyacinth node tracks the incoming link for the packet. Due to the tree topology, there is exactly one path to reach a particular egress node starting from a certain node. Thus if the packet was mis-routed the incoming and the outgoing links are the same. This is the property that Hyacinth uses to detect mis-forwarding attacks. Note that a similar situation can occur even when paths are re-routed. However, in that case the involved nodes are aware of the routing changes and hence do not trigger false alarms.

- **Packet Delay:** Each Hyacinth packet carries information that is used to determine RTT on a per-packet basis. If the delay goes beyond certain threshold, the egress asks the CU the number of nodes along the path and multiplies it with certain maximum tolerable delay per hop to determine if the observed delay indicates an attack. It is also possible for CU to gather information from individual nodes about their measured RTT and derive the average/standard deviation of network-wide per-hop delay. This information can then be used to derive a reasonable value for maximum tolerable delay per hop.

As pointed in the above analysis, some of these errors, *e.g.*, packet duplication, drops, delays, and reorder, could also happen in normal operating conditions of a wireless mesh network. It is not possible to reliably distinguish such ‘normal’ errors from those resulting from ‘malicious’ attacks without introducing prohibitive complexity into the protocol. However, it is also not necessary to always distinguish between them. Our approach is to use detection thresholds that can be set by manually or auto-determined by the network using always-on measurements. If a non-compromised node repeatedly crosses the threshold, it is isolated from the network. This is arguably the right course of action, because even though this node is not compromised/malicious, it is still hurting the network performance due to hardware/software issue, channel conditions in its vicinity, placement, or a combination thereof.

5.5.2 Control Plane Attacks

To recapitulate, we define a control plane attack in one in which a malicious node sends carefully crafted control packets so as to prevent the network from reaching a correct routing state. In the preceding subsection, we already showed that the malicious node cannot assume the identity of another node and/or send packets on its behalf. For discussion in this subsection, we assume that the control packet bear the address of the malicious node itself. It is the payload of the packet, e.g. traffic statistics, topology information, that is fabricated to launch the attack.

- **Topology Misreporting:** The first form of control plane attack is launched by misreporting topology information. Remember that each node reports its communication-range neighbors and the interference-range neighbors. An attacker node can falsify either of these. It can also not faithfully participate in topology discovery process, skewing the topology reported by its neighbors. Assuming A is the attacker node, there are four possibilities -

  1. An interfering node I is not reported: If node I reports node A as interfering, the algorithm still would account for partial (50%) interference. Even if node A does not participate in interference determination (leading to node I not reporting A as interfering either), in the worst case it could cause most of its interference links to be removed from consideration [4]. Still the effect of this attack is limited to the radio vicinity of A.
  2. A non-interfering node I_f is reported as an interfering node: This attack is similarly limited for two reasons. First, if I_f does not report A as interfering, then we only account for 50% of interference. Second, if A reports too many such nodes as interfering (leading to partial interference), node A could be marked as potentially compromised.
  3. A communication neighbor C is not reported: This attack only has limited effect that links between A and some of its neighbors are not used for communication. If A does not report majority of its neighbors, it gets detected as its neighbors all report A in their neighbor list.
  4. A non-neighbor C_f is reported as a communication neighbor: Since A and C_f are considered neighbors only if each of them reports the other as a hearing-range neighbor, this case has no impact the working of the protocol.

- **Traffic Misreporting:** It is also possible for an attacker to misreport the actual amount of traffic it sees (routes, consumes, or generates). Hyacinth employs a cross-checking mechanism to detect any inconsistency in the traffic reported by various network nodes. Since the clocks of various nodes are approximately synchronized, it is easy to synchronize the traffic measurement cycles of the network nodes. This

[4] A's hearing range neighbors can hear A and would still report A as interfering.

simplifies the cross-checking and makes it relatively independent of the the actual measurement window size (defaults to 16 sec for of all our experiments).

The only parameter that really matter is how out of sync are the windows of measurements for neighboring nodes. We empirically show in Section 5.6 ahead that the cross-checking can easily tolerate reasonable misalignments in traffic measurement windows.

- **Node Misconfiguration:** A compromised node could misconfigure its channels and/or routes. Such misconfiguration would lead to the compromised node getting disconnected, and would easily get detected by its communication neighbors. Further, such misconfiguration could just get treated as normal node failures, and worked around by reconfiguring the network in that radio vicinity.

- **Attack Nodes Addition:** An attacker can attempt to introduce new attack nodes into the network. As discussed in Section 5.3.1, it is not possible for an attacker to bring in attack nodes with new identity. This is because each Hyacinth node needs a certificate signed by CA in order to operate. However, the only way to obtain a certificate is to use a specific key available on a node only during the short initialization phase and erased afterwards.

  Another possibility is for the attacker to introduce clones of a compromised node, all with the same identity. The clones could, however, run different software. This cloning attack is tackled with a threshold-based mechanism. As a normally operating mesh node is only expected to have a limited number of neighbors, cloning a compromised node and placing them at different points in the network bumps up the number of neighbors. The topology discovery is itself strengthened using anonymous HELLO messages and hence the attacker cannot prevent neighbors of all the clones from reporting these clones to the CU. The attacker could also co-locate these clones with the original compromised node, but that does not help the attacker beyond having control over the compromised node.

5.5.3 Limitations

Although the proposed security solution can detect and recover the network from several attacks launched by compromised nodes, it has the following limitations -

- **Occasional Tampering:** The node isolation time increases if the malicious node only tampers with packets occasionally. If the frequency of tampering falls below a certain threshold, it would go undetected by the proposed mechanism. However, this is fine as occasional packet drops/delays can also happen due to wireless channel errors. Furthermore, the actual impact of such occasional tampering is also small.

- **New Node Compromise:** A new node joining the network should not be initially compromised. This is necessary because a compromised new node can reveal the public key K_{CU}^{Public} allowing more attack nodes to join the network.

A new node joins the network quickly after it can communicate with one more node that is already part of the network. We expect that a new node should be able to communicate with the network, i.e. at least one network node, within some short time of its physical deployment. This makes it highly unlikely that the new node can be compromised. Moreover, the CU tracks the number of nodes joining the network against the number of physically deployed nodes, helping detect this rare situation if it does happen.

- **Colluding Attackers:** Our proposed protocol works well even if a certain fraction of network nodes are compromised and are used to launch an uncoordinated attack. However, it is possible for compromised nodes to collude and launch attacks that can not be detected by our proposed mechanism. For instance, two colluding non-neighboring nodes can send falsified neighborhood information to the CU by pretending to be neighbors. This collaborative control plane attack would go undetected, lead to non-optimal route/channel selection.

  The same is true for collaborative data plane attacks. The actual attack detection is done purely at the egress node by detecting anomalies in the packet stream, and hence collaboration does not affect the ability to detect the final attack. However, our attacker identification mechanism searches for 'transition points' along the path, *i.e.* links at which a packet is impacted. For instance, it is possible for two collaborating malicious nodes, say A and B along the path to evade identification by intelligently orchestrating the attack. At first, B can tamper with the packet stream. But when the binary search for the transition point probes a node between A and B, A can take over the attack launch. This way the binary search would fail to identify the attacker.

  One can devise more stringent checks to detect falsified topology/traffic information despite collaboration among compromised nodes. Similarly, it is possible to use a more sophisticated search that simultaneously probes multiple nodes along the path, and thus be robust in presence of collaborative data plane attacks. That discussion is beyond the scope of this dissertation.

- **Congestion Attacks:** One of the attacks that is not tackled by the discussed protocol is the congestion attack. It is possible for a node to simply congest the network by pumping in packets. However, it is relatively easy to detect such attacks using the C3L technique discussed in the previous chapter. Specifically, a misbehaving node's neighbor could detect if the node is trying to congest the network by going beyond its allocated bandwidth.

- **RF Attacks:** A compromised node can be forced to create RF interference. However, this issue is not related to the protocol itself – an attacker does not even need to compromise a network node to introduce RF interference.

5.6 Performance Evaluation

We implemented the proposed security mechanism in ns-2 simulator. In this section, we present the results of a comprehensive performance study conducted using that implementation.

Unless otherwise specified, each experiment was conducted on a 49-node grid network with 2 gateways located at diagonally opposite ends of the network. Each node was equipped with 4 network interfaces tuned to one of the 25 available channels (similar to IEEE 802.11a). The actual route and channel computations were done on a wired node. This wired node gathered the traffic and topology information from all the network nodes, and sent the computed channel and route modification commands to the respective nodes. Depending on its position, a node could directly communicate with upto 4 neighboring nodes. The sensing range was set to double the hearing range, making a node interfere with upto 12 neighboring nodes depending upon its position.

By default, all communication was simulated with CBR flows (without application-level retransmission) making it easier to analyze the results. One of the randomly chosen nodes was designated as the attacker and it indiscriminately impacted the control and the data packets (as they are indistinguishable from an intermediate node's viewpoint). The attack itself was launched in 5 second cycles on a fixed number of initial packets (per cycle) between a pair of communicating nodes again chosen at random. In contrast to an always-ON attack, these ON-OFF attacks are harder to detect, and therefore better test the working of the protocol.

5.6.1 Effectiveness Analysis

We first evaluated the effectiveness of the overall security protocol under various attacks. In each set of following experiments, we varied one of the network, traffic, or attack-related parameters, e.g. communication speed, attacker position, attack intensity, route length, and measured the time it takes for our protocol to detect the attack and pinpoint the attacker. These result serve the dual purpose of demonstrating the resilience of the protocol under various scenarios, and also showing the impact of these variations on the detection/identification time/accuracy.

5.6.1.1 Communication Speed

In this experiment, we set up 7 aggregated CBR flows (each representing an aggregate of a set of user flows), and simultaneously scaled the data speed for all the flows. The scaling of 4 (the rightmost bar in each graph) corresponds to the maximum traffic possible through the network without congesting the network. The same experiment was repeated with 4 different kinds of packet mishandling attacks – delay, drop, dup, and reorder. We do not show the results for packet fabrication and packet modification as they are relatively easier to detect, and their results are somewhat similar to packet duplication attacks.

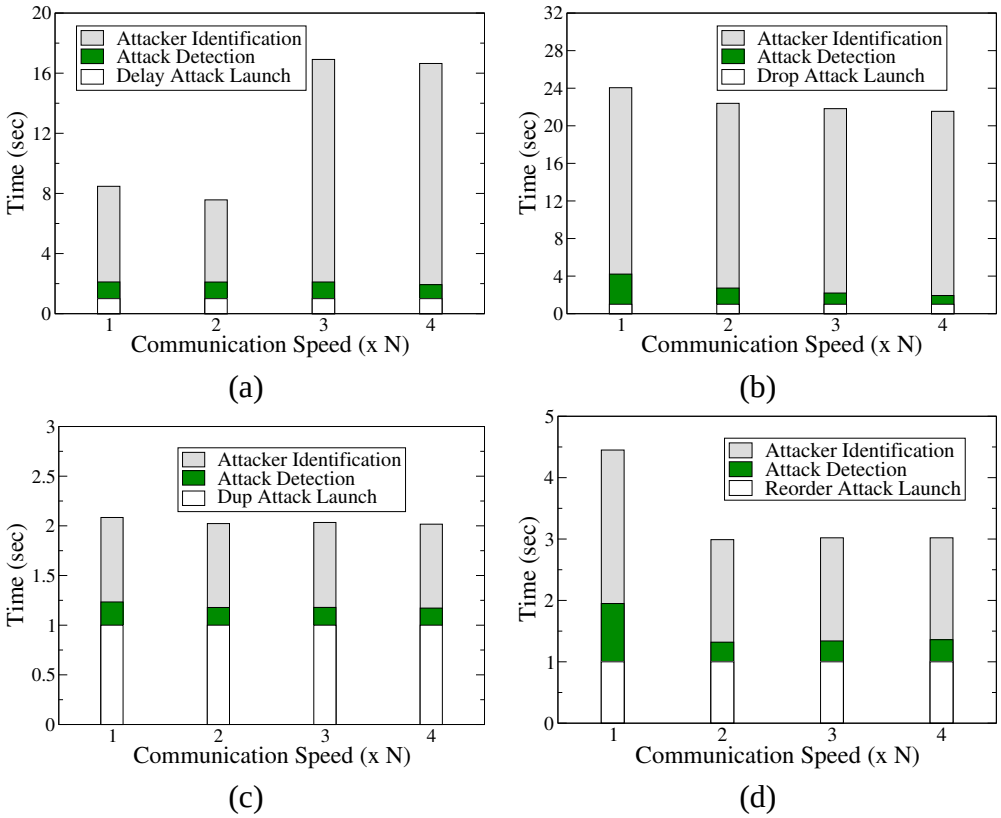


Figure 5.3: *Impact of communication speed on the protocol effectiveness. Each graph corresponds to a different packet mishandling attack – (a) delay attack, (b) drop attack, (c) dup attack, and (d) reorder attack. The protocol works effectively across different communication speeds.*

Figure 5.3 shows the results. Each point on the X axis corresponds to a certain scaling of the traffic, and the corresponding bars plot (1) the attack launch time, (2) the attack detection time, and (3) the attacker identification time. The different graphs correspond to different attack scenarios. The graphs provide several insights into the protocol -

- The dup and reorder attacks are much faster to detect than the delay and drop attacks. This is because a multi-hop wireless network does not by itself introduce duplicate packets; nor does it reorder packets. In contrast, packet delays and packet drops are a common occurrence in wireless networks. Packets could be delayed due to network buffering or local retransmissions resulting from intermittent transmission errors. The same effects could also result in packet drops. Therefore, the detection thresholds for delay and drop attacks need to be much higher.

- There are no false positives or false negatives in any of the scenarios, suggesting the robustness of the protocol to various communication speeds.

- Within each graph, the attack detection time reduces with the communication speed.

Traffic Multiplier	False +ve (count/experiment)
$<= 4.05$	0
$>= 4.10$	1+

Table 5.1: *Probability of false alarms shoots up as the network is overloaded.*

This is because higher the communication speed, smaller the time the attacker takes to impact the fixed number of packets required to cross the detection threshold.

- The same is not true for attacker identification time. The actual attacker identification requires a binary search through the path. For dup and reorder attacks, the set up and the waiting time for each step of the binary search dominates that time. For delay and drop attacks, the identification actually ran across multiple attack cycles, and thus faster communication rates did not help.

- In case of delay attacks, there is an increase in the identification time with communication speed. This is an artifact of the way the attack was simulated. Specifically, the delay attack simulation ensured that it did not lead to reordering of packets. Therefore, a packet could only be delayed to the extent allowed by its subsequent packet. This effect reduced the amount of introduced delay at higher communication speed, leading to an increase in attacker identification time.

Table 5.1 summarizes the measurements for false positives (in absence of any simulated attacks) as the network traffic was increased. As long as the traffic was below 4.05 x base-traffic, there were no false positives. Once the traffic went above this point, we consistently observed false alarms. As we expected, this tipping point was the same as the network saturation point. Below this point, there were few drops due to buffer overflows on intermediate routers. Beyond this point, there were several clear instances of intermediate routers dropping packets as they could not forward those in time.

In the above simulations, we did not use any congestion control. In a real deployment, we propose using Coordinated Congestion Control (C3L) to prevent congestion by capping the rate at which individual ingress nodes inject packets into the network. Alternatively, we expect individual application flows to use transport-layer congestion control which gets triggered using RED (Random Early Drop) mechanism implemented on mesh routers.

5.6.1.2 Path Length

We next varied the communication path length and measured its impact on the protocol efficacy. Figure 5.4 shows the results. The increased distance between the sender and receiver does not impact the detection time, but increases the attacker identification time. This is expected because the detection itself is performed only on the receiver node without any aid from intermediate nodes. Thus, there is no observable change in detection

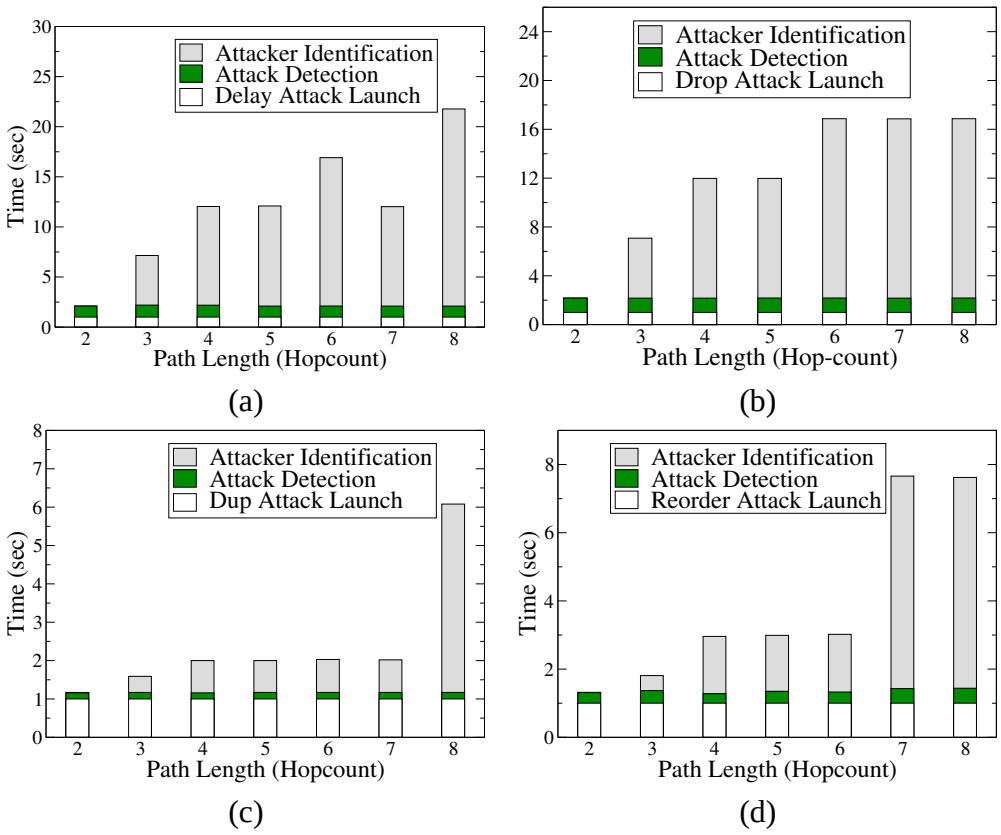


Figure 5.4: *The detection time is not impacted by the path length. The identification time increases logarithmically with the path length.*

performance. On the other hand, the attacker identification involves intermediate nodes - specifically $O(log(P))$ nodes due to the binary search involved. As a result, we observe a logarithmic increase in the attacker identification time. Since the attacks were launched in 5-second cycles, we see jumps in the identification time through all the graphs.

We also conducted another set of experiments in which we varied the path length, but did not simulate any attacks. Those experiments confirmed that the path length also does not have any direct impact on the occurrences of false positives.

5.6.1.3 Attacker Position

In this experiment, we varied the position of the attacker along the communication path. Figure 5.5 shows the results from the experiment. While the protocol works effectively regardless of the relative position of the attacker, there is no clear relationship between the position of the attacker and the time it took to identify it. One might expect that the binary search should finish earlier if the receiver happen to pick the right node in the first step itself. However, this is not the case, as attacker identification in our protocol requires finding a *pair* of adjacent nodes with differences in the monitored packet log. However, the binary search looks at only one node at a time. Even if the search starts with one of those two adjacent nodes, it takes $O(log(P))$ steps to get to the adjacent node (P is the length of the path).

5.6.1.4 Attack Intensity

In this experiment, we varied the intensity of the attack in a couple of different ways. We first varied the attack rate measured in terms of number of packets impacted during each attack cycle. Figure 5.6 shows the results from the corresponding experiment. We see false negatives or increased identification time at very low attack rates. This is to be expected as attacks fall below detection threshold. At the same time, such low-rate attacks do not cause much harm to the network either.

In most cases, an increase in the attack rate leads to faster identification of the attacker, as it is easier to observe an attack. Simultaneously though, the probability of control packets getting impacted also increases with attack rate. For example, in case of packet drop attack, there is a a higher chance of an encrypted control packet (carrying the snoop results) getting dropped and requiring retransmissions. This effect shows up most prominently in drop attack and delay attacks.

At very high rate, the drop attack converts to a black hole attack preventing any communication, and we switch to the alternate sender-side drop detection mechanism.

We next varied the impact of attack on individual packets while keeping the attack rate (number of impacted packets per attack cycle) constant. Figure 5.7 shows these results.

- For delay attack, we varied the amount of delay introduced by the attacker (Figure 5.7(a)). Intuitively, an increase in introduced delay should make it harder to detect the attack, resulting in increased detection and identification time. This is indeed true for the initial increase in delay from 1 sec to 3 sec. The step seen for the

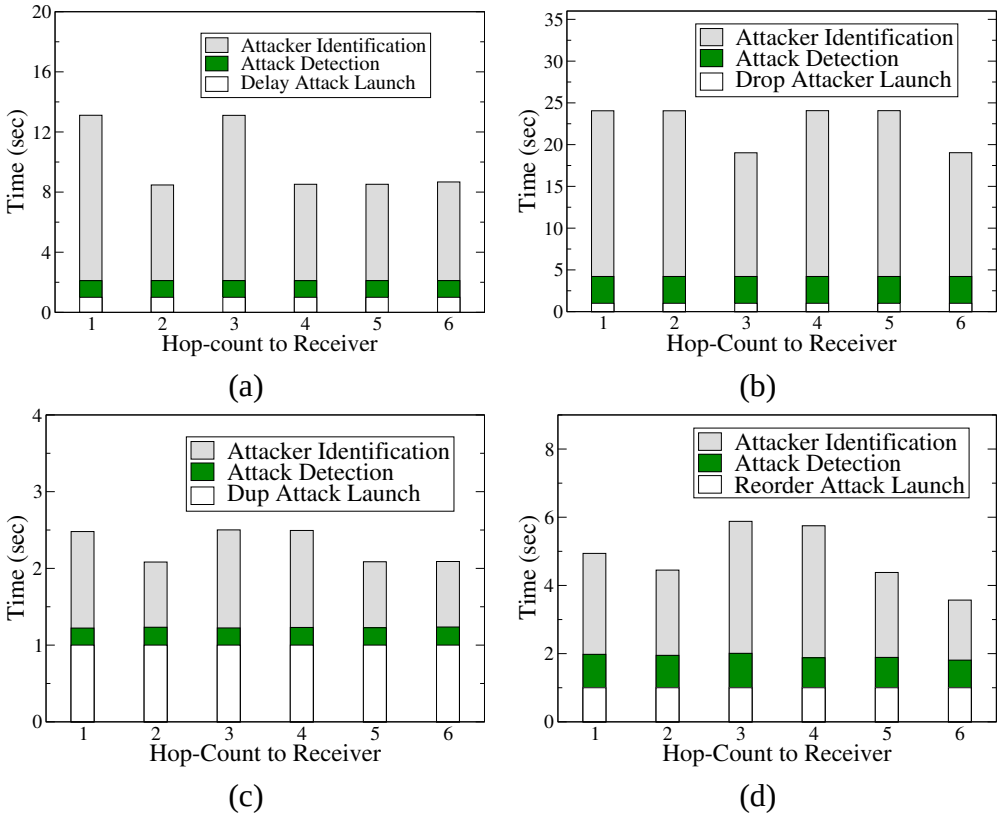


Figure 5.5: *Changing the position of the attacker does not impact the accuracy of the protocol.*

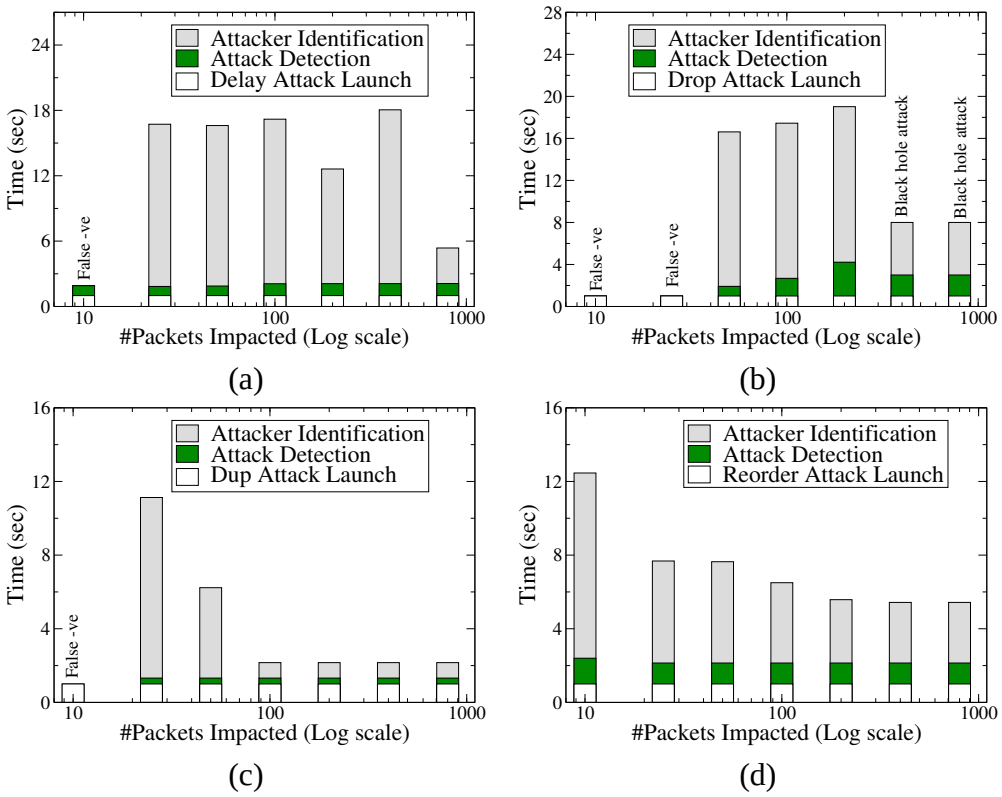


Figure 5.6: *Very low rate attacks are harder to detect, but also cause negligible harm. At high attack rate, it becomes easier to observe an attack, but simultaneously harder to communicate control packets.*

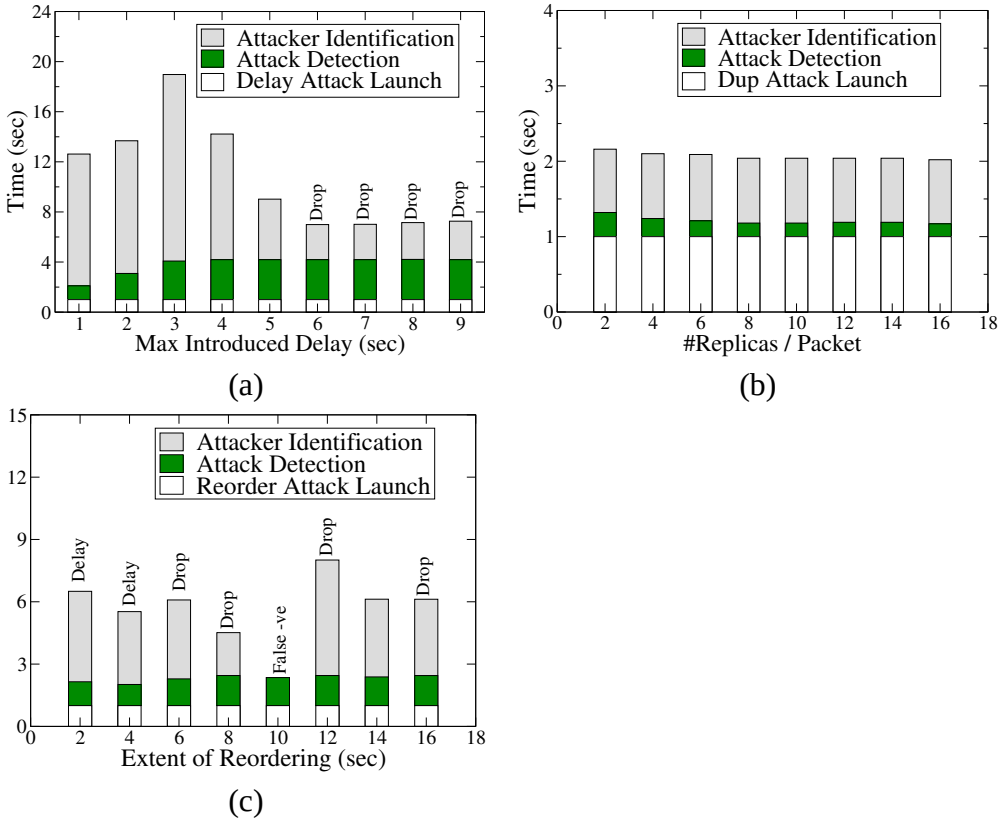


Figure 5.7: *Impact of attack intensity on protocol effectiveness. The label on a bar indicates the type of the detected attack whenever it is different from the actual attack type.*

Periodicity	1 sec	2 sec	5 sec
Delay	2.89	5.79	9.82
Drop	2.89	7.73	19.66
Duplication	1.93	3.87	4.91
Reorder	2.03	1.93	1.91

Table 5.2: *Attacker identification time measured in seconds past the attack detection, at different attack periodicity. A lower periodicity value makes the attacks continuous leading to faster identification.*

attack with 3 sec delay is due to 5-second attack cycles. However, beyond this point, it actually becomes easier to detect delayed packets as there is no need to wait for next 5-second attack cycle. Instead, in the first part of each attack cycle one can observe a silent period, and in the latter part one can observe the delayed packets arriving in bunch. This essentially makes it easier to detect the attack when a larger delay is introduced.

The right part of the graph indicates that at much higher delays, the delayed packets essentially gets treated as dropped. As a result the attacks get detected as drop attacks, as indicated in the graph by labels on the bars. For the same reason, the detection and identification time in this operating range are constant.

- For the dup attack, we varied the number of replicas per impacted packet introduced by the attacker. Figure 5.7(b) shows that this variation has negligible, though positive, effect on the protocol effectiveness. This is because with more copies per packet there is a chance of sensing the attack (during both the detection phase and the identification phase) a little earlier.

- Finally, for the reorder attack we varied the amount of time a packet was delayed before being sent, in effect increasing the amount of reorder in the packet stream. Figure 5.7(c) shows the results from this experiment. At lower attack intensity (controlled here by amount of delay introduced), the receiver actually infers a delay attack. At higher attack intensity, the receiver infers a drop attack.

5.6.1.5 Attack Pattern

In the preceding experiments, the attack cycle was set to 5 sec, leading to an ON-OFF attack. In this experiment, we wanted to see the impact of different attack patterns. We achieved this by modifying the periodicity value as shown in Table 5.2. As the periodicity value is reduced, the attacker identification time also decreases. This is because the lower periodicity values, e.g. 1 sec, make the attack lot more continuous, making it easier to detect the attack without waiting for next attack cycle.

Attack Type	Attack Launch Time (sec)	Attack Detection Time (sec)	Attacker Identification Time (sec)
Delay	1.00	2.14	7.94
Drop	1.00	-	-
Duplication	1.00	1.34	2.92
Reorder	1.00	1.86	5.41

Table 5.3: *Attacker identification time (in sec) for various attacks with presence of TCP flows. TCP congestion control gets triggered in presence of drop attacks, making it impossible to detect the attack.*

5.6.1.6 Traffic Pattern

We conducted various experiments to check the impact of different traffic patterns. We experimented with FTP (running over TCP) application traffic. We ran 7 FTP upload flows each starting from a randomly chosen node, and destined towards a node on the wired network. Table 5.3 shows the results. The proposed security protocol works equally well with TCP traffic. Although not shown in the Table, there were no false positives in absence of any attacks. In case of drop attacks, TCP congestion control gets triggered and the traffic source dramatically reduces the sending rate. This makes it impossible to detect the attack. In this way, TCP actually ends up helping the attacker.

To experiment further with variable bit rate (VBR) traffic, we simulated exponential traffic sources. We varied the fraction of time each source was ON during 500 msec cycles. The average sending rate within the ON time was fixed across all experiments. Thus, the overall average sending rate was proportional to the fraction of ON time. Figure 5.8 shows the results. As one would expect, it is easier to detect an attack when the packets are continuously being exchanged (larger ON time fraction) than when the packets are exchanged infrequently (smaller ON time fraction). This can be seen from reduction in the detection time with increasing ON time fraction. However, the attacker identification time does not show any clear correlation with the ON period. The reason is that as the fraction of ON time increases, the overall sending rate also goes high. As only a fixed number of packets are impacted in every 5 seconds attack cycle, increased sending rate makes it more likely that the identification takes multiple attack cycles.

Finally, we experimented with changing the traffic direction to see its impact on the proposed protocol. Again, we picked a set of 7 random nodes each of which communicated with a wired node. We first introduced uni-directional traffic streams, and measured the detection/identification times for each of the attacks. Next we replaced these uni-directional traffic streams with bi-directional traffic streams, with aggregate sending rates and attack rates twice as much as earlier. Table 5.4 shows the results from this experiment. For simplicity, we only show the difference of the identification time and the detection time. The protocol works well in presence of bi-directional traffic.

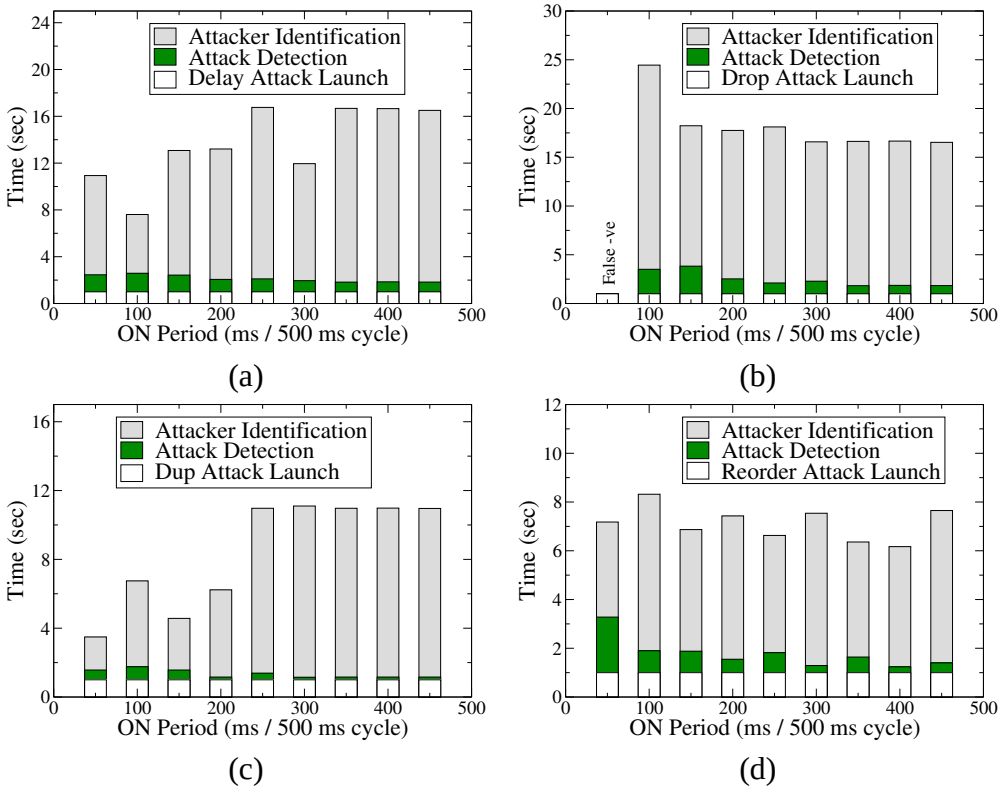


Figure 5.8: *The proposed protocol works well in presence of ON-OFF exponential traffic. Not shown in the graphs are the experimental runs without any attacks, which revealed no false positives with change in traffic pattern.*

Attack Type	**Uni-dir Attacker Identification Time (sec)**	**Bi-dir Attacker Identification Time (sec)**
Delay	15.42	15.09
Drop	14.91	13.91
Duplication	4.91	4.91
Reorder	5.95	5.80

Table 5.4: *The proposed mechanism works well in presence of bi-directional traffic including no false positives (not shown).*

Delay	Drop	Dup	Reorder	Attack Launch Time (sec)	Attack Detection Time (sec)	Attacker Identification Time (sec)
√				1.00	2.10	12.62
	√			1.00	4.21	19.02
		√		1.00	1.32	2.20
			√	1.00	2.13	6.11
√	√			1.00	2.11	9.21
√		√		1.00	1.39	2.28
√			√	1.00	2.11	7.89
	√	√		1.00	1.39	2.29
	√		√	1.00	2.11	3.12
		√	√	1.00	1.60	2.50
√	√	√		1.00	1.63	MisID
√	√		√	1.00	2.04	6.05
√		√	√	1.00	1.63	4.67
	√	√	√	1.00	1.63	2.56
√	√	√	√	1.00	1.85	6.00

Table 5.5: *Multiple simultaneous attacks launched from the same attacker. The attack rate is proportional to the size of the chosen combination. The protocol works well even in presence of mixed attacks.*

5.6.1.7 Multiple Attacks

We also experimented with multiple simultaneous attacks. We first launched multiple attacks from a single attacker. Specifically, we took each possible combination of the four attack types – delay, drop, dup and reorder – and experimented with each such combination. For each incoming packet, the attacker would pick one attack type out of the chosen combination at random and simulate the attack. Table 5.5 lists the results from this set of experiments.

Overall, the protocol works effectively even in detecting mixed attacks. This is because the actual logic to detect anomaly in packet stream does not require the attack type to stay constant during the detection and identification. The identification time is smaller for combination attacks, due to increase in the overall attack rate.

We then experimented with multiple attackers launching simultaneous, but independent, attack along the path. We started with two attackers and simulated different attack types for each of the two attackers. The results shown in Table 5.6 demonstrate that the protocol is effective in most cases. With the two attackers simultaneously launching the drop attack, the drop rate exceeded the threshold for 'black hole' attack due to the way the attack bursts were aligned on timescale. This triggered the sender-side detection.

The bar graph in Figure 5.9 shows results from another experiment where we placed an increasing number of independent attackers along a path each operating independently.

	Delay/A_s	**Drop**/A_s	**Dup**/A_s	**Reorder**/A_s
Delay/A_r	3.10 / 13.95	5.21 / 9.02	2.10 / 7.55	2.11 / 8.40
Drop/A_r	4.21 / 19.03	5.76 / 9.04*	2.18 / 7.56	4.22 / 19.07
Dup/A_r	2.10 / 7.83	4.22 / 5.48	1.19 / 2.04	1.32 / 2.18
Reorder/A_r	2.11 / 9.45	4.21 / 5.19	1.52 / 2.39	2.09 / MisID

Table 5.6: *Two independent attackers launching simultaneous attacks. A_r is the attacker closer to the receiver, while A_s is closer to the sender. The two numbers in each cell of the table are the attack detection time and the attacker identification time respectively. Both numbers are measured in seconds elapsed from the launch of the attack. The number marked * corresponds to a gray hole attack with too many packets getting dropped; the attack gets detected on the sender instead of the receiver.*

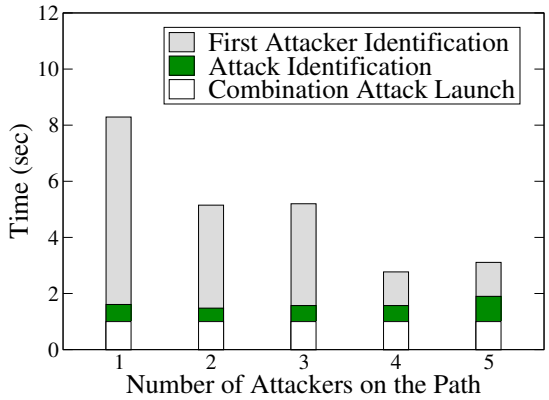


Figure 5.9: *Effectiveness of protocol in presence of multiple independent attackers along a path with 11 hops.*

Each attacker launched a combination attack by picking one attack out of delay, drop, dup, and reorder every time it impacted a attack. To increase the path length, we removed one of the gateways for this experiment. The graphs further substantiate that Hyacinth detection and identification mechanism works well in presence of multiple attackers as long as they are not collaborating. The detection/identification time reduces with more attackers due to an overall increase in the attack intensity (proportional to the number of attackers).

5.6.1.8 Network-induced Packet Errors

As discussed in Subsection 5.5, packets routed over a wireless mesh network can be duplicated, dropped, delayed, or reordered even during normal network operations. These errors could result from wireless bit errors, 802.11 backoffs/retransmissions, router buffer overflow, *etc.* We ensure that our protocol does not raise false alarms due to these *expected* errors primarily with use of appropriate detection thresholds. It is relatively easy to set these values based on running measurements from the network that can be collected as part

Per-Link Error Rate	False +ve (count/experiment)
$<= 1.25$	0
$>= 1.50$	1+

Table 5.7: *False positives resulting from* induced *wireless channel errors. Note that (1) the errors shown are post-retransmissions, and (2) the errors are accumulative over multi-hop path.*

of regular traffic statistics collection. To prevent tampering of the resulting data, we can take, say, 90th percentile of the reported values and thus remove outliers.

Table 5.7 shows the impact of increased per-link error rate on the false positive rate of the protocol. As expected, if the channel errors go above certain point, the proposed protocol treats them in the same manner as it would treat a malicious attack.

Network reconfiguration can also result in transient communication errors. The number of packets affected during a given reconfiguration depends upon the packets in transit and the extent to which the clocks on the nodes are de-synchronized. The number of packets in transit in turn depends upon the flow rate, instantaneous buffers, and also the path length. These errors are dealt with using three different mechanisms -

1. 802.11/LRTP hop-by-hop retransmissions fixes some of these errors, especially those resulting from channel reassignment;

2. The end-to-end packet losses/reordering resulting from route changes are inevitable. But due to their transient nature, they do not necessarily trigger a false alarm;

3. If a false alarm (initiating an attacker identification) might still triggered, the CU informs the egress node about the relevant route change. The details of this mechanism are discussed in Subsection 5.4.2.3.

5.6.1.9 Traffic Misreporting Attacks

It is also possible for an attacker to misreport the actual amount of traffic it sees (routes, consumes, or generates). Hyacinth employs a cross-checking mechanism to detect any inconsistency in the traffic reported by various network nodes. Since the clocks of various nodes are approximately synchronized, it is easy to synchronize the traffic measurement cycles of the network nodes. This simplifies the cross-checking and makes it relatively independent of the the actual measurement window size (defaults to 16 sec for all our experiments).

We conducted experiments with an increasing number of malicious nodes (ranging between 1 and 5) independently misreporting traffic statistics and our cross-checking mechanism indeed could identify those malicious nodes. We also evaluated the possibility of false alarms resulting from this mechanism. The only parameter than really matters is how

Measurement Window Misalignment	% of Measurement Window (16 sec)	False +ve
1.0	6.25%	0
2.0	12.5%	0
3.0	18.75%	2
4.0	25%	3

Table 5.8: *The accuracy of traffic cross-check algorithm depends upon the misalignment of traffic measurement windows across network nodes.*

out-of-sync the clocks across mesh nodes are. To measure its effect, we intentionally misaligned the traffic measurement windows across nodes. Table 5.8 shows the corresponding results. Even at 16 seconds measurement windows and 10+% window misalignment (2 sec), there are no false positives. Given the relatively high accuracy of our clock synchronization, it is possible to detect traffic misreporting even at such fine granularity. In real networks, the traffic reporting window would probably be larger making it even easier to detect traffic misreporting attacks.

5.6.2 Sensitivity Analysis

In this subsection, we analyze the sensitivity of the algorithm to various parameters.

5.6.2.1 Attack Detection Thresholds

Hyacinth uses various thresholds to detect attacks. The relationship between these detection thresholds and the accuracy of the algorithm is fairly obvious: a lower threshold enables detection of low-rate attacks, but also increases the possibility of raising false positive alarms. On the other hand, a higher threshold makes the algorithm robust by reducing false positives, but at the cost of more false negatives and increased identification time for actual attacks.

Table 5.9 shows the relationship between the threshold for the delay attack detection and the accuracy of the algorithm. A similar relationship holds for thresholds concerning other attacks – packet dropping, duplication, reordering, modification, and mis-forwarding. The key difference is that duplication, modification and mis-forwarding have much lower thresholds than others, as these are not normal occurrences in a wireless mesh network. In contrast, packets get delayed, dropped, and reordered under normal communication over a wireless mesh network.

All the detection thresholds are relative to a monitoring window (See Subsection 5.4.1.1). This is the window of recently received packet (from an ingress node) that a Hyacinth node monitors for signs of attacks. There is only a small amount of metadata kept for each packet in the window (the actual payload is not stored). Further, the computation done upon addition or removal of a packet from the monitoring window is incremental.

Delay Attack Detection Threshold	Identification Time (sec)
0.001	False +ve
0.01	7.57
0.10	7.57
0.15	17.24
0.20	17.24
0.25	17.24
0.30	17.24
0.35	22.30
0.40	False -ve

Table 5.9: *Impact of delay attack detection threshold on the effectiveness of the protocol.*

Snoop Window Size	Identification Time (sec)
3	MisID
10	MisID
20	6.09
40	6.09
80	6.09
160	6.09

Table 5.10: *Impact of snoop window size on detection of packet duplication attack.*

This makes the packet monitoring fairly light-weight. This also means that the monitoring window size and the attack detection thresholds can be flexibly set based on other criteria.

5.6.2.2 Snoop Window

Another parameter of interest is the size of the snoop window. Snoop window stores the set of recently snooped packets (potentially with their payloads if payload is being corrupted by the attacker) on a snooper node (See Subsection 5.4.2.1 for further context). A snooper node needs to store packets long enough so that the receiver has a chance to look at them and potentially ask for them from the snooper. It also needs to account for possible communication delays along the path.

We experimented with different snoop window sizes and its impact on attacker identification time. Table 5.10 shows the results. A very small snoop window leads to inaccurate attacker identification. This is to be expected as at small snoop window, the receiver cannot get the snooper node's view of the the impacted packets. However, the identification time stabilizes at relatively moderate sizes of snoop window. Furthermore, the amount of data stored per packet is less than 2 KB with the payload. Even if the snoop window was sized at 1000 packets, the storage cost per attack being detected is fairly modest (2 MB).

Snoop Wait Time / RTT	Identification Time (sec)
1.0	MisID
2	2.99
4	3.61
8	4.78
16	7.21

Table 5.11: *Impact of snoop wait time on detection of packet duplication attack.*

5.6.2.3 Snoop Wait Time

The final parameter of interest is the snoop wait time. This is the amount of time that the receiver needs to wait before it *can* ask the snooper for packets (See Subsection 5.4.2.1). The snoop wait time cannot be 0, because of the communication and setup delays incurred for snooping. Table 5.11 shows the effect of varying the snoop wait time (in multiples of RTT), on the attacker identification time. At very small values, the snooper does not have the chance to actually get any packets that the receiver is interested in. However, making the snoop wait time too large unnecessarily increases the attacker identification time. As the results highlight, any reasonable value of snoop wait time works well.

5.6.3 Protocol Analysis

In this subsection, we evaluate the computation and communication costs of the proposed protocol.

5.6.3.1 Communication Cost

- **Centralization:** The base protocol requires topology and traffic updates to be sent from each node to the centralized controller, and network reconfiguration packets sent on the reverse route. The *topology updates* are incremental and hence infrequent. On the other hand, the *traffic updates* are periodic, but result in negligible traffic. For a node with 4 interfaces, each traffic update requires 32 bytes.

  A reconfiguration packet has 8 bytes of metadata *e.g.* time for reconfiguration. Each route change is represented with 9 bytes of data, while each channel change requires 2 bytes of data. A reconfiguration can therefore be represented in less than 200 bytes, imposing minimal communication overhead.

  Centralization also requires synchronization of clocks across network nodes. Hyacinth piggybacks clock synchronization data on regular packets, imposing negligible additional overhead during normal network operations.

- **Security:** The main security-related overhead introduced by the protocol is due to the security header that is associated with every packet. Table 5.12 shows the details

Header Field	Bytes	Purpose
Flags	2	Multiple
Src Clock Count	1	Delay Measurement
Dst Clock Count	1	Delay Measurement
Reverse Delay	1	Delay Measurement
Send Time	4	Delay Measurement
Security Seq Num	4	Attack Detection
Checksum	4	Attack Detection
Total	17	-

Table 5.12: *Various fields of the Hyacinth security header.*

of the headers. The total header size is 17 bytes, or about 1.13% of 1500 bytes Ethernet packet size. This overhead is quite acceptable in itself. The header size can be further reduced through various optimizations.

Attack monitoring does not impose any communication overhead. There is some communication overhead associated with snooping which is only done on demand, *i.e.* if an attack is suspected. One iteration of snooping involves egress node to reliably request an intermediate node to start snooping. Once the anomaly is observed again, the egress node need to get the metadata or the payload for the tampered packet from the intermediate node. Even with explicit acknowledgement, each iteration only takes 5 packet exchanges plus any retransmissions. Even with a path composing of 10 nodes, the overall communication cost is $5 * log(10)$ or about 20 packets.

5.6.3.2 Per-Node Computation Cost

The main per-packet computation overhead is comparable to that introduced by transport protocol such as TCP. Each packet has an associated checksum that is encrypted with the session key and is used to detect tampering of the packet stream. Encryption is only needed for the control packets, and uses lower-overhead symmetric key encryption because of the session keys established between each pair of communicating ingress and egress nodes. The actual detection of abnormalities in the packet stream is done incrementally upon every packet arrival, and has low computation requirements. On the other hand, the snooping and attacker identification are only done when an attack is suspected.

5.6.3.3 Algorithm Scalability

The other cost of centralization is the network reconfiguration computation that is done on the centralized controller. The reconfiguration algorithm we implemented is incremental. Each run of the algorithm improves the channel and gateway load-balancing of the network. We ran several experiments to evaluate the computation time it takes for one run

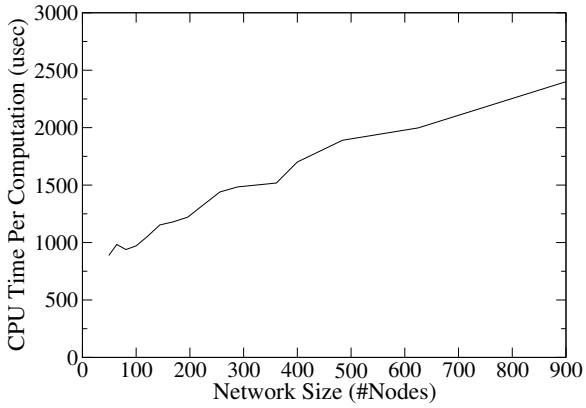


Figure 5.10: *The computation requirements for one round of algorithm with increasing network size.*

of reconfiguration, and how it scales with the network size. These experiments were conducted on a dual-core AMD Opteron Processor 8214 HE with 1 MB L2 cache and 1 GB RAM. Figure 5.10 shows the results from these experiments. With N nodes, a network had $\lceil N/50 \rceil$ gateways uniformly distributed across the physical edge of the network, and a traffic profile comprising of $N/2$ flows from randomly chosen nodes to a node on the wired network. Even with 800 nodes, the computation takes less than 3 msec. Furthermore, the memory consumption was less than 20% (*i.e.* 200 MB). The algorithm should, therefore, easily scale to a few thousand nodes even on a single compute node. As most real-world wireless mesh networks are less than 5-6 hops away from the wired network, this scalability is fairly good. If needed, the computation can be further scaled by partitioning the network into islands or performing the computation on a cluster.

5.6.3.4 Clock Synchronization

One of the important requirements for a centralized protocol is synchronization of time across network nodes. Without clock synchronization, different nodes might reconfigure their route tables and network interfaces at different times leading to transient network disconnectivity and thus packet drops. To show this effect, we conducted another experiment where we intentionally set the clocks across nodes at different phases and *disabled* our clock synchronization mechanism. Table 5.13 shows the resulting packet losses from one round of reconfiguration. In each run of the experiment, we picked the round of reconfiguration that resulted in maximum network-wide packet loss. The packet losses are small as long as the clocks are relatively synchronized, and there are no false alarms raised in these cases either. However, once the clocks go of synchronization beyond a couple of seconds, the packet losses shoot up, and our security mechanism starts raising alarms.

For secure clock synchronization, Hyacinth requires that every node synchronizes its

Clock Out-of-Sync Window (msec)	Max Packet Loss Count	False +ve's
0	0	N
200	12	N
400	13	N
1000	27	N
2000	60	N
3000	739	Y

Table 5.13: *Impact of clock desynchronization on reconfiguration-induced packet losses.*

clock directly with the central controller. This however does not impose additional communication overhead as the synchronization data is piggybacked on normal control communication.

Figure 5.11 shows the effectiveness of the clock synchronization protocol over a 7x7 node network with some nodes upto 8 wireless hops away from the wired network. In the first experiment (Figure (a)), each node had an initial phase difference (from the clock on central controller) in the range -10 to +10 sec, but the clock drift was 0. In the second experiment (Figure (b)), each node had a constant drift from the central controller clock in the range -1 to +1 msec/sec, and no initial phase difference. In the third experiment (Figure (c)), each node had both a phase difference of upto 10 sec (+ or -) and a constant drift of upto 1 msec/sec (+ or -). In each case, we plot the actual phase difference of the nodes after two rounds of clock synchronization. The phase difference values are sorted within each graph.

As expected, without a clock drift, we are able to synchronize all clocks within 2 msec of the central controller clock. In the other two cases, we are still able to achieve synchronization within 20 msec of the central controller clock. The maximum difference between any two nodes is thus 40 msec, which would in the worst case lead to a few packet drops during any reconfiguration. Further, most of these packets would get locally retransmitted by IEEE 802.11 MAC layer itself.

5.7 Conclusions

Today's wireless mesh networks are vulnerable to a variety of attacks. Application of conventional cryptographic techniques only addresses part of the problem. Specifically, those techniques do nothing to keep the network operational even if just a single node gets compromised. Before enterprises can truly embrace wireless mesh networks, the associated security issues need to be addressed in depth.

In this chapter, we argue that it is typically insufficient to patch security on top of an existing protocol. Instead, security needs to be considered a first-class requirement and addressed during the protocol design itself. We also argue for a centralized network

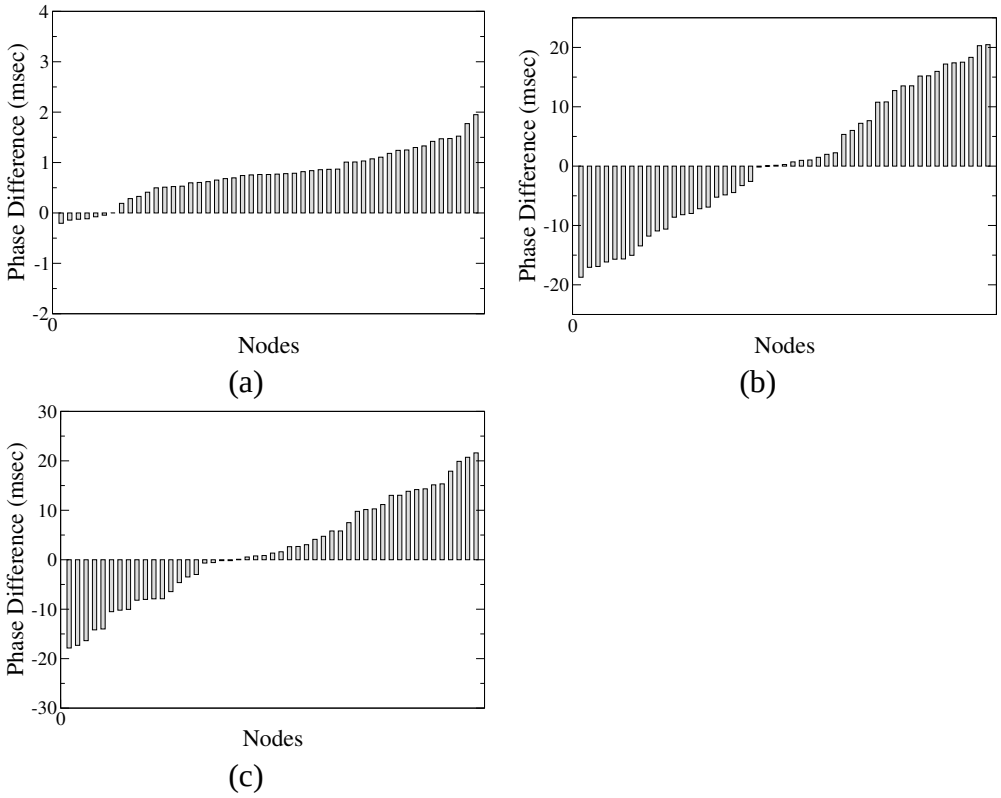


Figure 5.11: *The phase difference after two rounds of clock synchronization. Figure (a) corresponds to a network with initial phase difference across nodes; Figure (b) to a network with constant clock drift across nodes; Figure (c) to a network with both initial phase difference and constant clock drift.*

management architecture where the roles of individual nodes is reduced to reporting of topology and traffic information to a centralized controller, reconfiguring the interfaces and routing tables based on the decisions made on the centralized controller, and forwarding the packets in concert with their local network configuration. We believe this is a practical solution to securing a wireless mesh network and really making it enterprise-ready.

We designed a secure resource management protocol that performs centralized routing and channel assignment. Although many of our fundamental techniques are applicable to a network using a general mesh-based radio topology, we develop our protocol in the context of a spanning-tree based radio connectivity. For most practical enterprise deployments, we believe this is the common mode of operation. Apart from performing the computation on a centralized controller, our protocol uses a built-in PKI infrastructure. We secure the topology/traffic collection using novel cross-checking techniques. The dissemination of reconfiguration info is also fully secured. We secure the actual packet forwarding by (1) providing transparent packet authentication; (2) performing continual low-overhead in-mesh attack detection; and (3) providing a fine-grained packet tracing mechanism that can track down malicious nodes with high accuracy. Finally, we demonstrate effectiveness of our proposed protocol through analytical arguments as well as comprehensive ns-2 based simulations.

Chapter 6

Hyacinth Prototype Implementation

To demonstrate the feasibility of the proposed WMN architecture, we built a 9-node Hyacinth prototype using desktop PCs equipped with commodity 802.11 interfaces. The prototype features most of the techniques we proposed in this dissertation – multi-channel mesh networking, network-layer resource management, and stateful transport protocol with the explicit rate-based congestion control. The implementation of coordinated congestion control and centralized secure routing is left as part of future work. In this chapter, we discuss the implementation details of the current prototype, as well as the challenges we foresee in the implementation of coordinated congestion control and secure routing protocol.

6.1 Multi-Channel Mesh Network

6.1.1 Hardware Setup

The current Hyacinth prototype has 9 mesh nodes, two of which serve as gateway nodes that connect the prototype to our department's wired network. Each node in the current prototype is a standard desktop PC (Dell PowerEdge 600SC) running Linux 2.4.26 equipped with 2-3 802.11a/b/g PCI network interfaces. Fig 6.1 shows the software/hardware components in a node, while Fig 6.2 shows the picture of an actual node from the prototype. The testbed has a mix of interfaces from Orinoco and Netgear both of which are based on Atheros chipsets and are driven using the MadWifi driver [60]. Two of the cards operate in 802.11a ad-hoc mode and are used for mesh connectivity. The third card card is optional (marked with * in the Fig 6.1) and operates in 802.11g HostAP mode [61] providing infrastructure connectivity to mesh clients. For experimentation purposes, we set up the traffic sources/sinks on the mesh nodes themselves, and do not use the third optional card.

Although the 802.11 interfaces mounted on the same machine operate on non-overlapped channels, they still interfere with one another [44]. We believe this interference arises from radiation leakage because of imperfect channel-filter hardware in commodity cards. We reduce this interference by using PCI cards equipped with external antennas

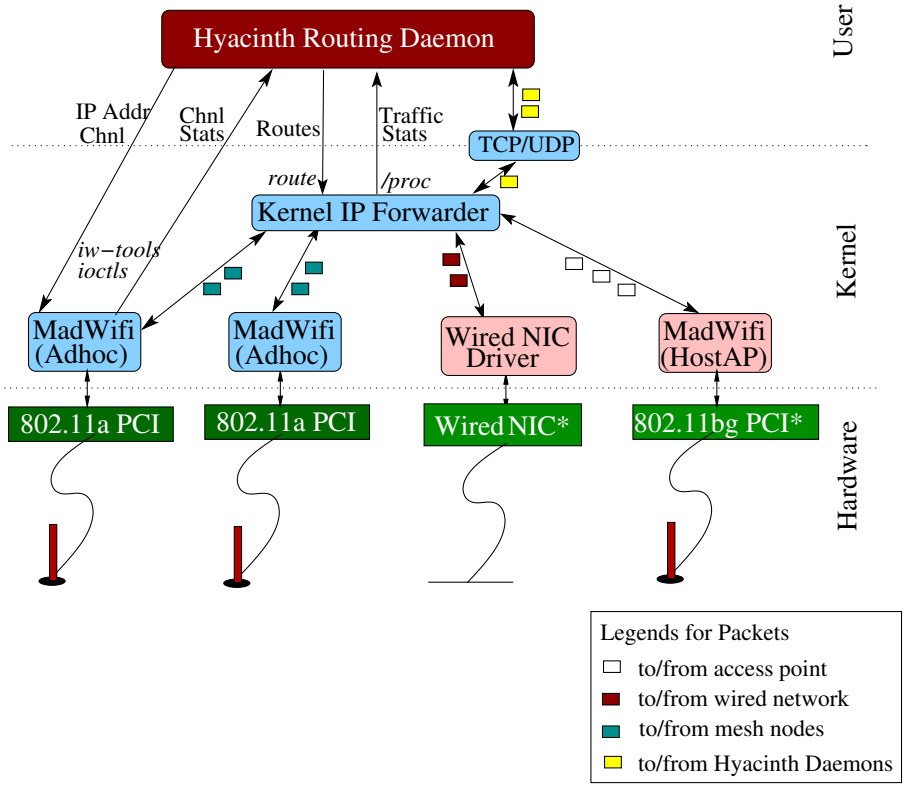


Figure 6.1: *The hardware/software architecture of an individual multi-channel WMN node in the Hyacinth prototype.*

(separated by 2 feet distance) and no internal antenna. Additionally, the channels assigned to the two cards mounted on the same machine are at least one channel apart from each other. We explore this issue of inter-NIC interference further later in the Chapter.

6.1.2 Software Architecture

Fig 6.1 shows the overall software architecture of a WMN node. To enable fast packet forwarding, Hyacinth leverages the kernel's IP forwarding mechanism. The routing/channel assignment protocol discussed in Chapter 3 is implemented by a user-level daemon process. This daemon interacts with similar daemons running on other mesh nodes using the TCP [5] and UDP socket library. The daemon uses standard driver/OS interfaces to collect channel and traffic statistics and to modify the NICs' radio channels, their IP addresses, and the kernel routing table.

[5]This can be easily replaced with a more WMN-friendly reliable transport protocol such as LRTP.

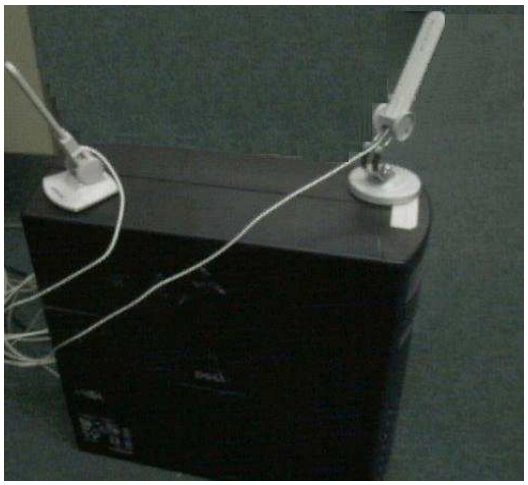


Figure 6.2: *A Hyacinth mesh node built using standard Dell desktop. The two antennas are separated by 2 feet to reduce inter-NIC interference.*

6.1.2.1 Statistics Collection

As Hyacinth daemon is not involved in actual forwarding of packets, it utilizes the standard */proc* filesystem interface provided by Linux to periodically, *i.e.* once every second, collect various traffic related statistics. Specifically, */proc/net/dev* reports statistics on the number of packets sent/received, the channel encoding used, and the number of transmission/reception errors for each interface. These statistics are used to compute normalized channel usage, which is then aggregated using exponential smoothing and periodically exchanged with neighboring mesh nodes. Each node aggregates the channel usage statistics it receives from its neighbors and those it collects locally to estimate the relative usage of various channels in its radio neighborhood. The estimation of the actual available bandwidth is a fairly complex problem in itself; it depends upon the overall available channel bandwidth and how it gets shared among interfering nodes/links. As we are only interested in relative usage of the channels, this crude aggregation provides a reasonable proxy to compare different channels.

MadWifi can report per-packet RSSI (Received Signal Strength Indicator) on a per-packet basis. We use this information as a measure of the channel quality. The aggregated RSSI is used in conjunction with packet transmission/reception error counts to infer broken/failed links and trigger network reconfiguration.

6.1.2.2 Node Reconfiguration

The IP address assignment to the mesh nodes is done through a *multi-hop DHCP protocol*. A node first assigns a temporary IP address from a small address space 192.168.254.1 to 192.168.254.254, and uses this address to connect to a global DHCP server sitting on the wired network. Upon contacting the global DHCP server, the node receives a set of new IP addresses (one for each NIC) that are unique within the entire mesh network. As IP

addresses are allocated from the same subnet, Linux IP forwarding module initiates ICMP redirects to incorrectly optimize the routes. We therefore disable all ICMP redirects in the prototype.

The actual configuration of node is done using standard Linux command tools [62] –

1. IP Address: */sbin/ifconfig*
2. Routes: */sbin/route*
3. Channel/ESSID: */sbin/iwconfig*

6.1.2.3 Access Network Integration

A WMN node optionally comprises a third wireless NIC that operates in 802.11g HostAP mode [61]. With this mode, the interface provides 802.11g infrastructure connectivity to the Hyacinth mesh network and in turn the department wired network. Each such mesh/AP node is responsible for assigning IP addresses to mobile stations that connect to it by running a local DHCP server. The IP addresses are allocated from a range that is obtained from the global DHCP server and are unique across the mesh. Whenever a mobile client moves from one access point/mesh node to another, the corresponding mapping from IP_{client} to IP_{mesh} is updated on the gateways. The gateways can thus transparently redirect packets destined to the client, just like in iMesh [63]. Further discussion of mobility support is beyond the scope of this dissertation.

6.1.2.4 Administrative Console

Each WMN node reports its current configuration and traffic statistics to a central server residing on the wired network. This central server combines the information from all the WMN nodes to provide a single point of visibility into the configuration and operation of the entire network. Specifically, the admin console provides high-level information such as the channel assignments for the nodes, the interference neighborhood of each node, the routes taken by each node to reach the wired network, the average traffic on various links, and bottlenecked nodes and channels across the network.

A more advanced console interface should provide two different views of the network - a physical view and a topological view. Physical view requires knowledge of node locations that can be acquired through locationing techniques [64, 65], or just manually input by the administrator. Topological view should depict the spanning tree formed by network routes. This is similar to what we implemented for MiNT [66].

Further, the interface could provide various control features to the administrator. For instance, the administrator could define *event monitors* and place them at different points across the network. The monitoring points could be chosen by the administrator or automatically decided by the network itself. As an example, an event monitor could look at the configuration and statistics of a particular node, a link or a channel, and trigger events if the values lie outside of certain reasonable/specified bounds. The administrative console could also allow *provisioning* of bandwidth based on node-id, or flow types, or both.

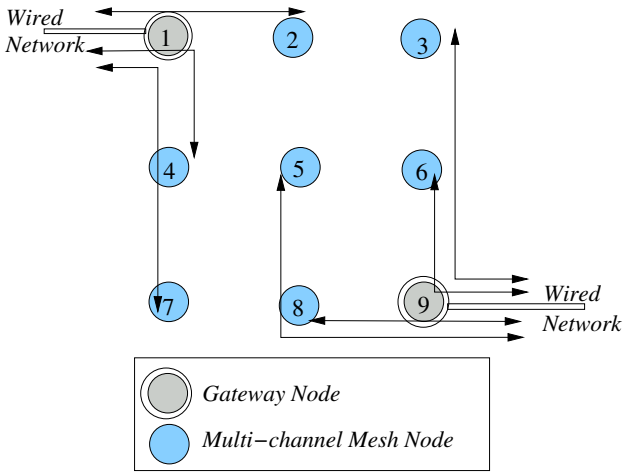


Figure 6.3: *Physical topology of the 9-node Hyacinth prototype. Each node is equipped with 2 802.11a PCI NICs whose channels are tuned dynamically by the channel/route daemon. Node 1 and Node 9 are connected to the wired network providing access to departmental servers and the Internet.*

6.1.3 Physical Topology

Fig 6.3 shows the topology of the 9-node Hyacinth prototype we built. The nine nodes are placed in an area of size approximately 20m x 10m spanning two lab rooms with two gateway nodes Node-1 and Node-9 connected to the department wired network. The transmit power on each node is reduced to 1 mW to limit interference zones of individual nodes. For evaluation purposes, the prototype can be run in two different modes – single-channel mode, and multi-channel mode. The single-channel mode only uses one of the cards to form the mesh network.

6.1.4 Inter-channel Interference

ns-2 simulator makes the assumption that there is no interference between non-overlapping channels. This assumption, however, is not entirely true in practice. In our experiments with real 802.11b PCMCIA cards, we observed various interference patterns depending on the relative positions of the cards. If the cards are placed right on top of each other, the interference is maximum, leading to only a maximum 20% gain in aggregate goodput over the single channel case (shown in table 6.1). If the cards are placed horizontally next to each other, as in Orinoco AP-1000 access points, the interference is minimum leading to almost 100% gain in aggregate goodput. In addition, the degradation due to inter-channel interference was found independent of the guard band, *i.e.* the degradation was almost the

NIC-1 Action	NIC-2 Action	NIC-1 Goodput	NIC-2 Goodput	% Maximum Goodput
send	silent	5.52	-	-
recv	silent	5.23	-	-
silent	send	-	5.46	-
silent	recv	-	5.37	-
send	send	2.44	2.77	47.6%
recv	send	2.21	4.02	58.3%
send	recv	4.22	2.42	61.0%
recv	recv	4.02	1.89	55.8%

Table 6.1: *Interference between 2 802.11b cards equipped with internal antennas placed on the same machine. The cards were operating in channels 1 and 11.*

same when channel 1 and 6 were used as compared to the case when channel 1 and 11 were used. We suspect this interference arises because of radiation leakage from cards before the channel-filter is applied.

This result has an implication over the placement of multiple cards over the same machine. The electromagnetic leakage from the cards need to be taken into account, and one card should not be placed in the zone where the strength of the leakage radiations by the other card is high. One possible way to achieve this is to use USB cards instead of PCI/PCMCIA cards and place them side-by-side in similar configuration as in Orinoco AP-1000 access points.

Another possibility is to equip cards with external antennas and place the external antennas away from each other. Using external antennas alone may not suffice; it is also necessary that the internal antenna of the card is disabled. We used Orinoco Gold PCI adapters that come with external antennas that enabled us to build multi-channel wireless mesh network using commodity PCs. Table 6.2 shows the results. The exact interference depends on the placement and card actions (send/receive). The use of external antennas is able to handle most of the interference effects as shown by table 6.2; the remaining interference is because of RF leakage from cables and from card's internal components.

Yet another option is to solve the interference problem at RF-level itself, an approach pioneered by Engim (now defunct). Engim chipsets received the complete spectrum, digitized it and processed it to compensate for inter-channel interference. This wideband spectral processing capability can help build single NIC with multi-channel communication capability while introducing minimal inter-channel interference.

6.2 Stateful Transport Protocol

As discussed in Chapter 4, we implemented and evaluated the stateful transport protocol as part of Hyacinth prototype. Our experiences working with the 9-node desktop-based

NIC-1 Action	NIC-2 Action	NIC-1 Goodput	NIC-2 Goodput	% Maximum Goodput
send	silent	5.93	-	-
recv	silent	5.75	-	-
silent	send	-	5.96	-
silent	recv	-	5.78	-
send	send	5.52	5.96	96.6%
recv	send	5.37	5.89	96.2%
send	recv	5.42	5.41	92.5%
recv	recv	5.66	5.17	93.9%

Table 6.2: *Reduced interference with the use of external antennas. Here, the cards were equipped with external antennas and operated on channels 1 and 6.*

Hyacinth prototype showed us the difficulties of using a full-scale network testbed during the development phase of the protocols and prompted us to develop a more flexible platform for wireless experimentation. We therefore built a miniaturized reconfigurable wireless network testbed called MiNT [54, 66]. In this section, we summarize the hardware setup of MiNT focusing on aspects that are relevant to our protocol implementation. We then present the details on the implementation of LRTP based on Linux 2.4.26.

6.2.1 MiNT Hardware Setup

MiNT uses small form factor PC and signal attenuators to significantly reduce the space requirement for the testbed, while providing multiple collision domains for high-fidelity experimentation. MiNT also uses off-the-shelf robot to enable quick reconfiguration of testbed. The testbed also incorporates other functionalities to aid experimentation, such as extensive monitoring, application debugging, and hybrid ns-2 experimentation.

Figure 6.4 shows the 9-node MiNT testbed that we used for LRTP evaluation. Each MiNT node comprises of a RouterBOARD 230 [67] which provides the computing platform mounted on a mobile robot that aided testbed reconfiguration through physical movements. Each RouterBOARD has 4 mini-PCI IEEE 802.11 a/b/g cards to support multi-radio experiments. A radio signal attenuator is inserted between a wireless interface and its antenna to shrink the signal coverage and thus the physical space requirement. The mobile robot is built by modifying an inexpensive off-the-shelf robotic vacuum cleaner from iRobot, called Roomba. We modified Roomba to enable control of Roomba movements from the RouterBOARD and to enable automated 24x7 operations through automated battery recharging.

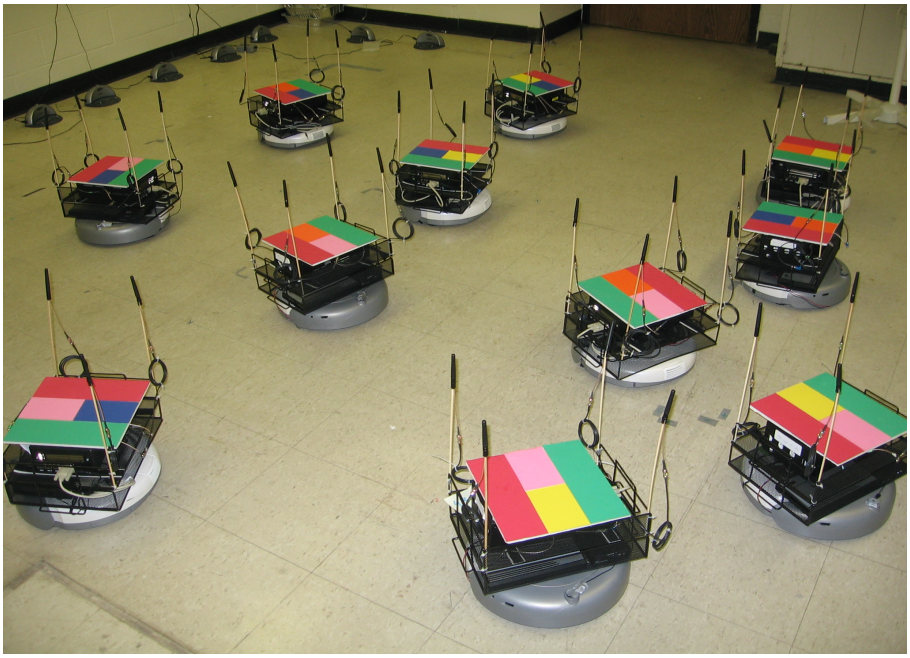


Figure 6.4: *Miniaturized wireless network testbed (MiNT) used for LRTP evaluation. MiNT enabled easier management and reconfiguration for LRTP experiments through use of radio signal attenuation and controlled physical mobility.*

6.2.2 LRTP Virtual Device Driver

As mentioned earlier, the forwarding of packets is done purely inside the kernel to reduce data copies and achieve higher performance. We therefore implemented LRTP as a pluggable Linux module. The module appears to the kernel as a virtual device driver that sits between the kernel's IP layer and the MadWifi wireless NIC driver (Fig 6.5). It enables reading of various statistics (e.g. flows being routed, virtual link lengths) as well as configuration of various parameters (e.g. exponential smoothing ratio) through the standard */proc* filesystem. We used a similar setup to also implement ATP [48] for performance comparison.

LRTP device driver works quite similarly to a standard network device driver. It registers the following functions to handle sending/receiving of packets:

- $lrtp\_hard\_header()$: This function is invoked by the kernel to attach the 'hardware header'. It adds the LRTP header that contains the maximum packet queueing time as well as the maximum packet transmission time for the flow along the path up to, and including, the next outgoing link from the current mesh node. After adding the LRTP header, $lrtp\_hard\_header()$ invokes the $hard\_header()$ function of the MadWifi device driver that in turn adds the actual Ethernet header. The LRTP header thus gets positioned right after the Ethernet header.

- $lrtp\_xmit()$: This function is invoked by kernel when it needs to transmit a packet.

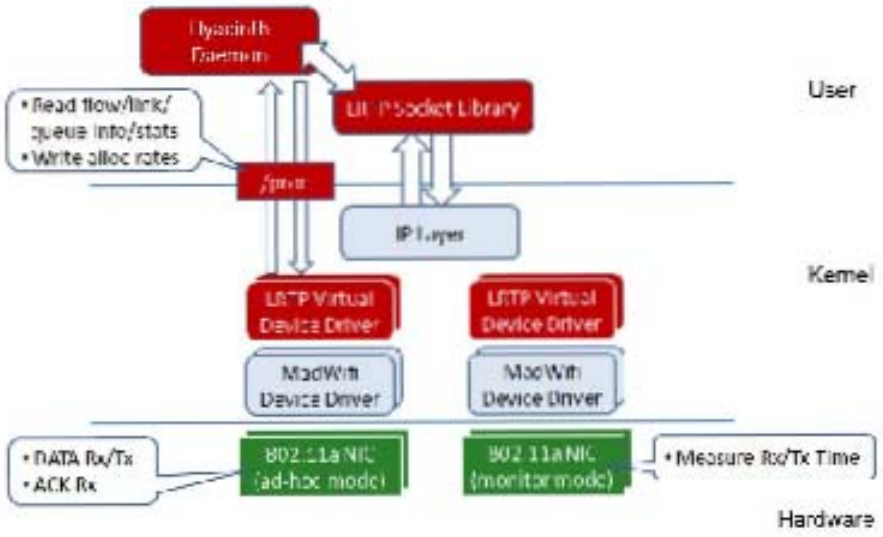


Figure 6.5: *Software architecture of LRTP. The link rate estimation, interface level max-min fair allocation, and transmission scheduling are done by the LRTP virtual device driver residing just below the IP layer. The sender rate adjustment is done by the LRTP socket library. To implement C3L, the Hyacinth daemon can perform the topology discovery, read the local traffic statistics from the driver, and report them back to the C3L controller (over an LRTP connection). The feedback on the available link bandwidth can be written back to the LRTP device driver's data structures through the driver's existing* /proc *filesystem interface.*

It updates the estimated arrival rate for the flow and also updates the allocated bandwidth if there is a significant change in the arrival rate for the flow. The function then adds the packet to the corresponding virtual link queue, prioritizing control/feedback packets over data packets by queueing them at the head of the virtual link queue instead of the tail. The actual sending of packet could happen asynchronously. $lrtp\_xmit()$ finally triggers the $lrtp\_schedule\_xmit()$ function discussed below.

- $lrtp\_schedule\_xmit()$: This function is directly invoked by the $lrtp\_xmit()$ function as well as triggered through use of kernel timers. It handles the weighted round robin transmission scheduling of all the virtual links associated with the NIC and queues the next packet from the appropriate virtual link queue to be transmitted to the MadWifi device driver's queue. The queueing is done only when the queue length of the MadWifi device driver falls below the pipelining threshold of 4 (See Chapter 4 for a discussion on the pipelining threshold). The function also always schedules a timer to reinvoke itself.

- $lrtp\_recv()$: This function is invoked by kernel to perform the bottom half processing of the received packets. It records the maximum queueing and transmission times stamped on the packet before stripping off the LRTP header and invoking the $netif\_rx()$ for further packet processing.

- $lrtp\_intr()$: This function is registered with the MadWifi device driver. It is invoked inside the interrupt handling routine of the MadWifi device driver and is used to record the system time at which a DATA packet or an ACK packet is seen on the air. This is also the point at which the ACK packets are matched against the DATA packets and the DATA packets with missing ACKs are scheduled for local software retransmissions.

6.2.3 Measuring Precise Packet Reception Timings

LRTP's hop-by-hop retransmission scheme requires the knowledge that the link-layer ACK for a transmitted packet is received. The same feedbacks are needed for computing the effective capacity of each virtual link. The latter also needs the exact time when the link layer ACK for each transmitted packet is received. While the 802.11 WLAN NICs we use can provide the link-layer ACKs while operating in normal mode, due to a firmware bug the time stamped on the received ACK packets is not accurate. To get around this problem, the current LRTP prototype exploits the RF Monitoring mode supported by most WLAN interfaces. In RF Monitoring mode, a WLAN interface can see all data, control (including ACK), and management frames transmitted in the monitored channels. With interrupt coalescing disabled, a wireless NIC interrupts the processor as soon as a frame is observed in the monitored channel. LRTP leverages RF Monitoring and instantaneous interrupt processing to obtain fairly accurate per-frame transmission completion time, and to infer the transmission status of each transmitted frame by checking the presence of the corresponding link-layer ACK.

Unfortunately, the 802.11 WLAN NICs cannot send packets when operating in the RF monitor mode. Therefore the current LRTP prototype deploys four NICs on every node, two of which were used for communication and the other two for monitoring them. The need for monitoring NICs should be obviated as the timestamping bug is resolved with newer firmware.

6.2.4 LRTP Library Implementation

Although local retransmission and rate allocation feedback mechanisms are implemented inside the kernel, the sending and receiving functionalities are implemented as socket libraries on the end hosts. The LRTP socket library looks similar to the TCP and UDP in terms of APIs. The $send()$ API implements (1) core send functionality (*e.g.* blocking and non-blocking send), (2) packet buffering and retransmissions based on negative ACKs, and (3) send rate adjustments based on the receiver feedback. The $receive()$ API implements (1) the core receive functionality (*e.g.* packet buffering, blocking receive, and select), (2) detection and notification of missing packets based on sequence numbers, (3) packet reordering, (4) explicit communication of rate feedback to the sender based on the network feedback (as stamped on LRTP header).

Note that our prototype assumes that both ends of the communication are LRTP end points. However, as discussed in Chapter 1 this is not a fundamental requirement and can be addressed with use of transparent connection splitting (See Fig 1.6).

6.2.5 Coordinated Congestion Control

The current Hyacinth prototype does not implement the coordinated congestion control. In this subsection, we present a blueprint for its implementation and discuss how this functionality should integrate with the existing prototype components.

The controller that executes the coordinated congestion control algorithm, is run as a user-level daemon process on a centralized administrator node/cluster. This centralized controller unit (CU) is physically connected to the wired network and can thus communicate with all the nodes via the gateways.

The Hyacinth daemon running on each mesh node (refer to Subsection 6.1.2) reports its topology information, including a list of the interference range neighbors for the node, to the centralized controller. This mesh node daemon also reads information on the LRTP flows being routed through the node along with their estimated bandwidth requirements, using the */proc* filesystem interface exposed by the LRTP device driver. These traffic statistics about the flows are also reported to the centralized controller.

The controller collects this topology and traffic information coming from all the mesh nodes in the network. It uses the aggregated topology information to compute/update collision domains in the network. The topology graph, the collision domains list, and the flow statistics are fed to the max-min fair allocation algorithm which then computes the fair bandwidth share of individual virtual links. The per-link allocation information is sent back to the Hyacinth daemon running on each mesh node. The daemon uses the */proc* filesystem

interface to feed this allocation information back to the LRTP device driver. Finally, the LRTP device driver uses this allocation to (1) compute and stamp per-flow allocation, and (2) control the overall sending rate for each outgoing virtual link. This final piece is already implemented by the current LRTP virtual device driver.

6.3 Centralized Secure Routing

The centralized secure routing protocol we presented in Chapter 5 is implemented inside ns-2 simulator, and is not yet implemented by our prototype. In this section, we discuss an architectural plan for a real Linux-based implementation of the protocol and the challenges we foresee in that implementation.

6.3.1 Hyacinth Controller Unit

The controller unit (CU) discussed in Subsection 6.2.5, also implements the centralized functionalities required by our secure routing protocol. Specifically, the CU is now implemented by a set of user-level daemon processes each of which provides unique functionalities/services, *e.g.* public-key infrastructure (PKI), ingress misbehavior reporting service, and DHCP service.

The Hyacinth daemon running on mesh nodes performs the topology discovery and retrieves local traffic statistics. Instead of running the route/channel computation as in the current prototype, it now reports them back to the CU. The CU aggregates the topology reports and traffic statistics from across the network nodes, and runs the cross-checking algorithm to detect any signs of false representation. It then runs the channel/route assignment computation using the reported topology and traffic statistics as inputs. The computed channels and routes, or more precisely modifications to the channels and routes, are then securely distributed to Hyacinth daemons running on the corresponding mesh nodes. Individual Hyacinth daemons implement these decisions by reconfiguring their local network settings using standard Linux tools mentioned in Subsection 6.1.2.2.

6.3.2 Hyacinth Virtual Device Driver

Most of the operations along the forwarding path of data packets are implemented by an extension of the LRTP virtual device driver implemented by the current prototype. The device driver establishes secure shared keys with its peer mesh nodes, (optionally) encrypts the outgoing packets, computes and inserts the Hyacinth security header, (optionally) decrypts the incoming packets, processes and removes the security header, monitors the incoming packet streams for signs of attacks, and continually estimates the path delays. All of this is done by the Hyacinth virtual device driver (that sits right above the 802.11 WLAN NIC device driver) completely transparent to the higher layers of the networking stack.

These functions need to be implemented over the aggregated packet streams between each pair of communicating ingress/egress mesh nodes. To enable sharing of network-level

state across otherwise independent applications, it is not possible to implement any of these functions as part of a library.

6.3.3 Hyacinth Mesh Daemon

As discussed earlier in Subsection 6.1.2.1 and 6.1.2.2, the actual topology discovery, traffic statistics collection, and node reconfiguration are still done by the mesh daemons running on each node.

In case of suspected attacks, the snooping of packets is done by the same mesh daemon. The Hyacinth device driver provides */proc* based interface to set simple packet filters (source/destination IP addresses) and thus enable/disable snooping of the forwarded packets. Once the packet filter is set, a running window of last N packets matching the filter is stored, and can be read by the mesh daemon using $ioctl()$ calls.

6.4 Conclusions

In this chapter, we presented an implementation of our proposed WMN architecture through an operational 9-node Hyacinth prototype built using commodity desktop PC and multiple 802.11 PCI NICs. Motivated in part by our implementation experiences, we also built a miniaturized reconfigurable network testbed, MiNT, which (1) uses small form factor PC and signal attenuators to significantly shrinks the space needed for a realistic experimental multi-wireless network, (2) employs off-the-shelf robots with auto-recharging to enable quick remote reconfiguration of testbed, and (3) features hybrid ns-2 simulations to provide an intermediate step going from ns-2 simulations to real implementations.

The Hyacinth prototype implements most of the ideas we have presented in this dissertation. For centralized secure routing, we presented a software architecture to serve as a blueprint for a future implementation. The prototype further strengthens our argument that commodity 802.11 hardware can be the core platform for building a high-performance and secure wireless enterprise backbone.

Chapter 7

Related Works

In this chapter, we contrast the proposed Hyacinth architecture with other related research efforts. The chapter is organized into 5 core sections. Section 7.1 discusses various channel assignment techniques proposed to effectively utilize multiple channels in a WMN. Section 7.2 compares different routing techniques aimed to improve the end-to-end throughput in a WMN. Section 7.3 studies the previous mechanisms proposed to improve the effectiveness of transport protocols. Section 7.4 focuses on various mechanisms geared towards improving the transport-layer flow fairness. Lastly, Section 7.5 discusses related mechanisms to strengthen the security of both wireless and wired multi-hop networks in presence of compromised nodes.

7.1 Multi-channel Mesh Networking

Although, the IEEE 802.11b/g standards and IEEE 802.11a standard provide 3 and 12-25 non-overlapped frequency channels, respectively, which could be used simultaneously within a neighborhood, conventional WMN architecture equips each node with a single interface. In order to preserve connectivity, this interface is kept tuned to a network-wide unique channel. To utilize multiple channels within the same network, each node either needs channel-switching capability [68, 15, 16, 69] or needs multiple interfaces each tuned to operate on a different channel.

7.1.1 Multi-channel MAC

Several proposals [15, 16, 17, 18] have been made to modify the MAC layer to support multi-channel networks. The approach taken by most of this body of research is to switch to an optimal channel for each packet transmission, essentially avoiding interference and enabling multiple parallel transmissions in a radio neighborhood. Unlike all these proposals, our architecture does not perform channel switching on a packet-by-packet basis; our channel assignment lasts for a longer duration, such as several minutes or hours, and hence

does not require re-synchronization of communicating network cards on a different channel for every packet. This property makes it feasible to implement our architecture using commodity 802.11 hardware. Further, almost all multi-channel MAC protocols can only perform local optimizations. On the other hand, Hyacinth takes a more global approach by adjusting channel assignments and routes based on the overall network traffic patterns.

There have been subsequent research proposals to use channel switching techniques on top of unmodified 802.11 protocol [70, 71]. However, none of these proposals implemented any real prototype to suggest that the approach is indeed feasible without physical hardware/firmware modifications. We demonstrated the feasibility of our architecture using commodity PCs equipped with 802.11a/b/g cards. For the same reason, our approach is more easily extensible to other low-level wireless networking protocol, e.g. 802.16. We believe that channel switching might be a more suitable approach for mobile ad hoc networks where battery life concerns might prompt use of single interface, a multi-radio architecture is more suitable for wireless mesh networks where bandwidth, and not power savings, is the primary concern.

7.1.2 Multi-radio Research

Use of multiple NICs had been previously discussed in [44] and [7]. However, Hyacinth was the first work to harness the performance potential of this approach and demonstrate the importance of intelligent channel assignment in realizing it. For instance, an identical channel assignment to all nodes as used in [44] artificially limits the throughput improvement possible over single-radio architecture.

A channel allocation problem also occurs in *cellular networks*. From the viewpoint of a cellular network provider, the number of available channels are limited, and the channels need to be re-used from cell-to-cell while maintaining the minimum re-use distance. The problem, also called channel allocation, is to assign certain channels to each cell based on its traffic and channels used in the near-by cells. Various static and dynamic techniques have been proposed and used to solve the problem [27]. However, in a cellular network, all mobile devices communicate with their corresponding base stations, while the base-station to base-station communication is carried over a separate network and the cellular channel allocation problem does not address that issue. This makes the cellular channel allocation solutions more directly useful to devising access point channel allocation [2] rather than channel allocation for a multi-radio ad hoc network.

We now discuss various channel assignment approaches, most of them proposed after the publication of Hyacinth.

7.1.2.1 Topology-Based Channel Assignment

Channel assignment can be done purely based on the network topology with the goal of minimizing the interference on any link. In that case, the channel assignment reduces to a constrained graph coloring problem where links are colored by the channels constrained

by the number of interfaces on a node. The problem is known to be computationally hard, therefore the proposed solutions only approximate the optimal.

[72] takes this topology-control approach to solve the channel assignment problem. The authors propose a greedy algorithm, termed Connected Low Interference Channel Assignment (CLICA), that visits all nodes in the dynamic order of their channel constraints; more constrained nodes are visited first. Upon visiting a node, the node picks the locally optimal channels for each of its communication edges with the goal of minimizing the maximum interference faced by any link. The authors prove that once a link is assigned certain channel, it does not need to be re-adjusted in later steps.

[73] proposes a search-based approximation solution, where the solution space is searched for a better solution until none is found for some number of iterations. It first uses a TABU search-based approximation algorithm to color each edge in the network graph with one of the K available colors such that the total number of conflicting edge pairs are minimized. In the next phase, edge-connected components are re-merged as needed to satisfy the interface constraint on each node.

[74] solves the channel assignment problem by formulating it as a linear program. Their objective is to find a channel assignment that maximizes the number of links that can be active simultaneously. The constraint of this linear program includes the number of interfaces on the node, and the fact that an interface cannot be assigned multiple channels. This problem turns out to be an integer linear program, since all the variables (channels, interfaces, interference) are all integer values. Because of its exponential complexity, the authors propose greedy heuristic algorithms to solve the problem.

Another recent work [75] decouples the problem of channel assignment from that of routing. The channel is done purely based on physical network topology and is static, while the routing is done based on dynamic network conditions. [76] additionally accounts for dynamic interference conditions and adapts the channels to minimize interference.

A topology-based channel assignment is based on the premise that all network links are equally loaded. This premise does not hold true for real-life traffic distribution, as some links are bound to carry more traffic than others. Intuitively, the goal of channel assignment in a multi-channel WMN should be to bind each network interface to a radio channel in such a way that the available bandwidth on each virtual link is proportional to the load it needs to carry. As the link load is determined by routing algorithm, ideal solution should perform channel assignment in conjunction with routing.

7.1.2.2 Traffic-Aware Channel Assignment

Most theoretical work on traffic-aware channel assignment require use of omnipotent scheduler to achieve optimal MAC scheduling [77, 78]. For instance, [77] solves the joint problem of channel assignment and routing and devise a constant-factor approximation of the optimal. The authors follow a multi-step procedure: The algorithm begins with an LP formulation for the routing problem with interference-free schedule based on the packing lemma. The channel assignment algorithm runs over this solution to achieve bandwidth equal to factor k-times required load at all the nodes. Next, an LP is formulated to reduce

the interference over the channel assignment and thus maximize the flow. It can be proven that this algorithm is within a constant factor of the optimal. The proof, however, relies on the assumption that the scheduling can be done by an omnipotent entity. The impact of using IEEE 802.11 scheduling in place of this optimal scheduler is not discussed.

Unlike these works, our proposal is substantiated using comprehensive ns-2 simulations as well as a prototype implementation using commodity 802.11 hardware.

7.2 High-Performance Routing

Routing governs how packets packets through a WMN. It is one of the most well-researched problems not only in the context of wireless ad hoc networks [79, 80], but also in the context of wired Internet. The simplest and the most widely-used routing optimizes the path length in the hope of minimizing the amount of network bandwidth used to transfer packets. Unfortunately, the shortest-path routing considers all links to be equal in importance. More broadly, it does not consider any of the important factors such as link errors, link criticality, or channel diversity during path selection. Intelligent selection of paths based on these factors can not only improve the quality of path chosen for current stream of packets, but also potentially enable network to admit more packet load in future. A fully comprehensive survey of routing protocols and their variations is beyond the scope of this dissertation. We instead discuss several representative routing techniques proposed in the literature, that aim to improve the end-to-end network throughput.

7.2.1 Link Quality-Aware Routing

In a wired network, it is usually safe to assume that a link either works well or does not work at all. The assumption does not hold for wireless networks, where a majority of the links have intermediate loss ratios [1]. Simply minimizing the hop count typically maximizes the physical length of each hop; this in turn minimizes the signal strength and maximizes the loss ratio of each hop.

Expected transmission count (ETX) is one of the first metrics that explicitly accounts for link quality during path selection [1]. The ETX of an individual link is defined as the expected number of transmissions (including retransmissions) it takes for a single packet to be successfully transmitted over that link. It is computed by the following equation:

$$ETX = \frac{1}{d_f * d_r} \tag{7.1}$$

where d_f is the forward delivery ratio, i.e. the probability of successful reception of a single packet sent by the transmitter, and d_r is the reverse delivery ratio. The ETX of a route is defined as the sum of ETX of individual links composing the path. The ETX metric not only avoids low-quality (high ETX) links, but also prefers shorter (low aggregated ETX) paths to minimize network resource usage. The delivery ratios d_f and d_r are estimated through periodic exchange of HELLO packets among neighbors.

ETX ignores the fact that different links in the network could operate over different link rates and hence consume different amounts of channel time. [44] overcomes this limitation by using expected transmission time (ETT), which is computed as:

$$ETT = ETX * \frac{S}{B} \tag{7.2}$$

where S is the average size of the packet, and B is the raw bandwidth of the link. While the true transmission time of a packet depends upon several factors such as back-off delay, the authors found that it is sufficient to use the raw bandwidth of the link to compute the expected transmission time.

Both ETX and ETT perform much better than shortest path routing, but do not capture the channel load along the path. As a result, these metrics could easily end up favoring congested paths even if a lightly loaded path is available.

7.2.2 Interference-Aware Routing

Various routing protocols have been proposed that consider the traffic load during the path selection. Load-Balanced Ad hoc Routing (LBAR) is one such early protocol [81]. In LBAR the route discovery is done in a reactive manner, where the destination chooses the least loaded path. The path load is captured by aggregating degree of nodal activity on the nodes composing the path. The degree of nodal activity for a node is in turn defined as the sum of number of active paths passing through the node and its interfering neighbors. This is given by the expression:

$$PathCost_p = \sum_{i \in p} (A_i + \sum_{j \in N(i)} A_j) \tag{7.3}$$

where A_i is the number of active flows going through node i and $N(i)$ is the set of neighbors of i.

Dynamic Load-Aware Routing (DLAR) similarly uses the the number of packets queued in a node's interface queue to estimate its load [82]. The interface queue size in some sense measures how overloaded the channel of the outgoing link is. Compared to these proposals, Hyacinth takes a more direct and proactive approach: it approximates the channel load observed by a link by explicitly summing the traffic load (bytes per second or packets per second) imposed on that channel by all its interfering links. This enables us to better estimate residual channel capacity on other channels without explicitly switching to them. In realistic deployments, it is possible for other wireless networks and other RF sources to pose interference on certain channels. A practical solution should combine the two approaches – (1) aggregate channel load from individual links for initial estimation, and (2) monitor interface queue sizes to further ensure that the initial estimates were accurate, and adjust the channel assignment if the actual capacity is much different.

Hyacinth's load-balancing routing is closest to [9]. Like Hyacinth, the proposed algorithm in [9] load balances the wireless links of a gateway node that connects a wireless

access network to the wired network. The gateway node co-ordinates the movement of all the nodes across the tree to achieve load-balancing.

7.2.3 Multi-Path Routing

Most routing protocols only utilize a single path to route packets between any communicating node pair. Another possibility is to discover and utilize multiple paths between a source and a destination node. The availability of redundant paths provides a degree of fault tolerance in case a node on the primary route fails. Further, the traffic can be distributed across multiple non-interfering paths to achieve better network load balancing and higher aggregate throughput.

Use of node disjoint paths to improve load-balancing of the network has been discussed in [83]. However, node disjointness of routes does not necessarily ensure that the paths do not interfere. This limits the amount of spatial load balancing such schemes can achieve in the context of WMNs. Much of the other work on ad hoc multi-path routing has been done with the goal of improving fault tolerance. For instance, in [29], the authors perform explicit selection of multiple node-disjoint paths at the receiver of route request. Specifically, the receiver compares the path stamped on different route requests before selecting the ones to be advertised back to the sender. Similarly in [84], authors propose a multipath variant of AODV called AOMDV. Here, multiple disjoint paths are maintained for each destination. When the primary path fails an alternate path is available instantaneously. This approach hides the route rediscovery overhead.

Although we only explored multi-path routing to a limited extent, our centralized architecture can be adapted to use other multi-path routing protocols.

7.2.4 Diversity-Aware Routing

In context of multi-channel wireless mesh networks, channel diversity of the paths becomes an important concern during the path selection. Specifically, to minimize intra-flow interference, adjacent hops of a route should operate over different channels. One such metric that accounts for channel diversity in path selection is Weighted Cumulative ETT (WCETT) [44]. Formally, WCETT is defined as –

$$WCETT = (1 - \beta) * \sum_{i=1}^{n} ETT_i + \beta * \max_{j=1}^{k} X_j \tag{7.4}$$

where ETT_i is the expected transmission time of a packet over i^{th} hop, X_j is the sum of transmission times over hops operating on channel j, and β is a tunable parameter between 0 and 1. The first term is used to limit the end-to-end delay incurred or the total network bandwidth consumed by each packet. The second term limits the number of hops that operate over the same channel. The authors' experiments suggest that WCETT is able to select channel-diverse routes.

WCETT suffers from two limitations. First, it does not take inter-flow interference into account. Paths whose nodes have more interfering neighbors would have worse performance. Second, WCETT is is non-isotonic that makes it unsuitable for use in link-state algorithms. A metric is said to be isotonic if it preserves the relative weight of two paths when both are prepended by a common path. Hyacinth's load balancing routing does not face either of these limitations.

7.2.5 Opportunistic Routing

Traditional routing protocols decide a sequence of nodes between the source and the destination, and route every packet through that sequence. This routing framework fails to leverage the broadcast nature of wireless transmissions. Specifically, when an intermediate receiver fails to receive a packet, the packet has to be retransmitted by the immediate transmitter even if another potential next hop successfully received the packet. ExOR is an opportunistic routing mechanism that makes delayed forwarding decisions [85]. Specifically, ExOR broadcasts each packet on each hop, determines the set of nodes that actually received the packet, and then chooses the best receiver (that is closest to the final destination) to forward the packet. Since choosing the best receiver incurs communication overhead, delayed decisions are made for batch of packets. The advantage of delayed forwarding decisions is that ExOR can try multiple long but radio lossy links concurrently, resulting in high expected progress per transmission.

ExOR attempts to decrease the total number of transmissions, but does not explicitly leverage the multirate option at the physical layer that leads to transient variations in transmission rates. ROMER [86] another routing protocol based on this framework leverages such transient variations to select the highest throughput path instead of the closest receiver to the destination. Specifically, ROMER forms an opportunistic, forwarding mesh that is centered around the long-term stable, minimum-cost path (e.g., the shortest path or long-term minimum-delay path), but opportunistically expands or shrinks at the runtime to exploit the highest-quality, best-rate links.

Although the opportunistic routing framework is quite promising for a single-channel mesh network, effective application of its ideas to a multi-channel mesh network is an open problem.

7.3 Effective Network Utilization

Transport protocol plays an important role in making the raw network bandwidth available to the applications. In this section, we discuss various mechanisms designed to increase the efficiency of the transport layer, by reducing overheads and accurately estimating the bandwidth available at network layer.

7.3.1 Reducing Acknowledgment Overhead

One of the characteristics of IEEE 802.11 wireless networks is high per-packet overhead. Each packet incurs large MAC contention, PLCP header, and link-layer ACK overhead. Any transport protocol that uses per-packet end-to-end acknowledgment therefore incurs substantial control overhead. Original TCP design required per-packet end-to-end acknowledgment; the delayed-ACK scheme reduces this to one ACK for every two segments. To maintain TCP ACK clocking, delayed-ACK cannot indefinitely wait for a second segment to arrive. Nevertheless, Delayed-ACK helps at higher transmission rates, as segments arrive with time gap less than delayed-ACK timeout. Further reduction in ACK traffic is achieved using ACK-filtering technique [87]. Here intermediate nodes suppress ACKs for previous packet in a connection if the ACK for a later packet is pending.

Our experiments suggest that even with delayed-ACK enabled, TCP's ACK traffic can impose upto 20% overhead on connection throughput. In Hyacinth we take a step further and design a customized transport protocol for WMN, called LRTP, that altogether eliminates the need for per-packet transport layer ACKs.

7.3.2 Reducing Retransmission Overhead

Most transport protocols employ an end-to-end loss detection and retransmission technique where the receiver identifies missing sequence numbers and requests retransmissions of lost packets from the original sender. While simple to implement due to its stateless network core, the end-to-end approach is unsuitable for wireless networks as they suffer from significant packet loss due to bit corruption. One mechanism to reduce retransmission overhead is Distributed TCP Caching (DTC) [88] that was originally proposed for wireless sensor networks. In DTC, intermediate nodes monitor network transmissions and cache data packets locally. All transport layer (TCP) ACKs are intercepted and upon reception of duplicate ACKs, cached data is retransmitted and duplicate ACKs are suppressed. DTC further monitors link-layer ACKs to determine optimal TCP segments to cache at individual nodes.

The IEEE 802.11 standard supports lost packet retransmissions at link-layer to limit the impact of bit errors on higher layers. A similar mechanism, termed PSFQ, has been proposed for wireless sensor networks [89]. In PSFQ, each hop retransmits lost packets proactively. However, 802.11 and PSFQ retransmissions suffer from head-of-the-line blocking. When transmissions to one of the node's neighbors fail repeatedly, even the packets destined to other neighbors get blocked in the interface queue. LRTP addresses this issue through intelligent scheduling of packets being sent on different virtual links (similar to the proposal in [90]).

7.3.3 Accurate Network Bandwidth Estimation

The other aspect of improving network utilization is deriving an accurate estimate of the available network bandwidth. Underestimating the available bandwidth clearly results in an

underutilized network. On the other hand, overestimating the available bandwidth results in network congestion and subsequent packet drops on intermediate routers.

1. **TCP's AIMD:** TCP employs AIMD probing to fill the network pipeline while avoiding congestion. The behavior of TCP and its variants over multi-hop ad hoc networks has been extensively studied [91, 92, 93, 48]. From congestion control standpoint, TCP exhibits several undesirable effects when operating on wireless multi-hop networks [48]. Firstly, TCP is less adaptive to fluctuating wireless channel conditions due to AIMD congestion control [48]. Secondly, TCP's ACK clocking does not work well due to ACK bunching. ACK bunching occurs because packets in a connection tend to be accumulated into chunks along a multi-hop path due to batched transmission at the link layer. It leads to further increase in traffic burstiness by skewing the RTT estimation upwards and thereby defeats TCP's self clocking strategy [48, 94]. Thirdly, TCP confuses wireless channel errors, a frequent occurrence in wireless networks, with congestion related losses and triggers its congestion control. As the channel conditions get worse, a TCP sender spends more of its time in slow-start, thereby underutilizing the network.

   Many implicit [13] and explicit [95] mechanisms have been proposed to enable TCP to discriminate between the two types of losses and to trigger its congestion control more intelligently. Similarly, proposals have been made to smoothen the traffic on the sender-side to alleviate the problems resulting from burstiness of traffic [87]. Although these proposals can improve TCP performance to some extent, one quickly hits the TCP's glass ceiling. We argue that TCP may not be the right protocol to begin with. As wireless mesh networks are still in a relatively nascent phase, we argue that instead of fixing TCP, the right approach might be to design a mesh-friendly transport protocol from ground up.

2. **Implicit Rate-based Congestion Control:** Packet dispersion is a popular technique used to measure wired network bandwidth. Many wireless protocols, such as WTCP [96], also use packet dispersion techniques. In its simplest form, a pair of back-to-back packets is sent over the network, and its inter-packet arrival delay is observed at the receiver. One would therefore expect the instantaneous service rate of the bottleneck link along the path to be reflected in the inter-packet delay at the receiver. However, this approach relies on the assumption that all flows in the network are serviced in a strict round robin fashion at the bottleneck links. This constraint is very difficult to be satisfied in 802.11-based WMNs. Since packet transmissions could be scheduled in a bursty manner, traffic at the bottleneck link may arrive in bursts, and the entire burst may be serviced without any interleaving, invalidating the strict round robin assumption.

3. **Explicit Rate-based Congestion Control:** ATP [48] takes a much more explicit approach and requires every intermediate node to measure queueing and transmission delays for packets passing through it. The sum of exponentially averaged queueing

and transmission delays yields the average packet service-time experienced by all the flows going through the intermediate node. In the equilibrium condition, each ATP flow attempts to maintain exactly one data packet on every router along the path. The service-time therefore reflects the ideal dispatch interval for all the flows competing over the bottleneck link. Every packet bears the maximum service-time encountered on any of the intermediate hops. The bottleneck service-time is communicated back to the sender, which adjusts its packet dispatch interval to match this service-time.

The problem with overall-service-time approach is that it couples the queue-size management with rate estimation, which leads to traffic fluctuations and in turn non-optimal estimation of channel bandwidth [57]. More concretely, it is very hard to maintain exactly one data packet from each flow on every router. If there is even a slight change in transmission time of a single packet, a queue (say of two packets) builds up on the router for the rest of the epoch. Clearly, an extra packet in the queue does not indicate any change in the network bandwidth. However, in the next epoch the ATP sender proportionally reduces its sending rate (to half in this example) to bring the queue down to one packet. It is in these epochs, that an ATP sender underestimates the path bandwidth.

LRTP also takes an explicit rate-based approach towards congestion control, but decouples the queue size management from rate estimation. As shown in Chapter 4, LRTP can thus achieve much better utilization of available network bandwidth.

7.4 Transport Layer Fairness

Improving network fairness has been the focus of many research projects. We discuss the representative works from the ad hoc networking literature, and compare them with Hyacinth's approach.

7.4.1 Ingress Flow Throttling

A promising approach to achieve flow fairness is to throttle flows at their ingress nodes to their network-wide fair shares. [30] provides a rough sketch of such an algorithm. Here, each WMN router measures the average offered load arriving from/to its own mobile users and the average capacity of the links to each of its adjacent routers. The offered loads and link capacities are periodically communicated to other WMN routers. For each link, a router then computes the aggregate time shares in each of the link's contention neighborhoods and chooses the minimum value. The minimum time-share is then converted into rate and transmitted to other routers. Finally, each ingress router determines its end-to-end flow rate, and throttles its flows to this system-wide fair rate. The authors propose to apply this algorithm at Layer 2 of the each router, as that keeps the TCP intact and works even in presence of uncontrolled transport flows such as UDP. Hyacinth's LRTP develops

on this approach, and provides a full solution that can provide max-min fairness on top of unmodified 802.11 MAC layer.

7.4.2 Neighborhood RED

[97] improves TCP fairness by extending the idea of Random Early Detection (RED) to multi-hop wireless networks. Simply applying RED to individual node's packet queue does not work, because of the shared channel space. Therefore, the authors view the queues on a node and its interfering neighbors as a single distributed queue and apply RED to this distributed queue. The node infers the neighborhood queue size by monitoring the channel utilization. When the utilization goes above a certain threshold, neighborhood congestion is detected. The node then computes its own drop probability, and broadcasts this information to its neighbors. Based on this information, the nodes with highest contribution to the neighborhood channel congestion drop packets from their local queues. In contrast of neighborhood-RED, LRTP routers take a much more direct approach by explicitly marking the allocation of a flow on its data packets.

7.4.3 Overlay MAC Layer

The unfairness problems of IEEE 802.11 MAC could be addressed by making it aware of application's priorities. [98] proposes another approach of forming an overlay MAC layer on top of 802.11 MAC. Based on loose time synchronization and application-assigned weights, the overlay MAC runs a distributed TDMA algorithm and allocates time slots to competing nodes. Weighted fair queueing is used to assign more time slots to nodes having greater number of flows. Every node in the network decides a set of time slots for all other nodes in the network using a random number generator with the same seed, thus enabling a distributed implementation. The packets are then trickled from the overlay MAC to 802.11 MAC based on this time slot allocation. The advantage of this approach is that it can perform fair bandwidth allocation on top of standard 802.11 MAC. However this scheme faces a common disadvantage faced by most TDMA schemes. It incurs overhead when the owner of a slot has nothing to send. They attempt to reduce the overhead using a timer after which the slot is taken over by the next node. Currently, Hyacinth does not attempt to finely control the packet flow, but only the rate at which they are entering the channel. If the approach works in practice, it could be used to control packet scheduling on individual channels, further reducing collisions and improving Hyacinth network goodput.

It should be pointed out that overlay MAC layer does *not* make it possible to switch the interface channel on a packet-by-packet basis. Therefore, one still needs to use multiple radios per mesh node to effectively utilize multiple wireless channels. One can view overlay MAC layer as a fine-grained extension of ingress flow throttling approach proposed in [30]. While [30] works at individual flow level, the overlay MAC layer works at individual hop level.

7.4.4 MAC-layer Fairness

Luo et al. [99] pointed out the trade-off between maintaining strict fairness among flows and maximizing the spectrum reuse efficiency. They presented a service model that guarantees a minimum bandwidth allocation to each participating flow while maximizing the aggregate radio resource usage. Huang and Bensaou proposed algorithms to compute the max-min fair share to each wireless link in a multi-cell ad hoc network [100]. Nandagopal et al. [101] designed an analytical framework that given an arbitrary fairness model specified as a utility function, can generate a contention resolution algorithm to achieve the specified fairness requirement. All these efforts consider only single-hop flows, and treat each multi-hop flow as multiple independent single-hop flows. As shown in [102], single-hop fairness does not translate into multi-hop fairness because of the traffic dependency among different hops of a multi-hop flow. The LRTP's C3L algorithm presented in this dissertation provides end-to-end max-min fairness to multi-hop flows.

7.4.5 Explicit Fairness Among Multi-hop Flows

Tassiulas and Sarkar [50] proposed algorithms to achieve max-min fairness across multi-hop flows. In their interference model, transmissions for multiple flows can proceed simultaneously as long as they do not share any common node. Our interference model is much more general, and better matches the real-world interference among nodes with omni-directional antennas.

Li [102] presented centralized and distributed algorithms to achieve end-to-end flow fairness in multi-hop ad hoc networks. He first defined the notion of a contending flow group that includes all flows which contend with one another directly or indirectly, and then ensured fairness by equally dividing the channel capacity among members of each contending flow group. However, their definition of contending flow group is transitive. For instance, consider 3 flows F1, F2, and F3, where F1 and F2 are in one collision domain, and F2 and F3 are in another collision domain. Li's algorithm groups these three flows into the same contending flow group, and assigns each of them B/3, where B is the channel capacity of a single collision domain. In contrast, we are able to achieve much stronger max-min fairness among multi-hop flows and assign B/2 to each flow in this case.

Finally, Yi and Shakkottai [103] present a hop-by-hop congestion control mechanism over wireless multi-hop networks, which imposes a channel access time constraint on each flow. They use an optimization algorithm that takes into account all the flows and their time constraints over their individual links to derive a global time constraint function for the entire network. Further, based on this global time constraint function, an optimal operating point that maximizes the utilization of the network resources in a fair manner is computed for each flow.

All the solutions described above require MAC layer modifications, to aid contention resolution or packet scheduling, and thus cannot be directly applied to IEEE 802.11-based WMNs. In contrast, LRTP is specifically designed to work with unmodified 802.11 MAC.

7.5 Routing Protocol Security

Securing of routing infrastructure has been a problem of research not just for ad hoc networks, but also for wired Internet. Here, we study the proposed solutions in both the fields.

7.5.1 Ad hoc Routing Security

Researchers have proposed secure variants of most of the earlier ad hoc routing protocols. [104] provides a comprehensive survey of these variants. The same problem has also been studied in the context of wireless sensor networks, and [105] surveys the various techniques proposed so far.

Authenticated Routing for Ad-Hoc Networks (ARAN) [31] is an on-demand, ad-hoc routing protocol that uses public key cryptography to ensure authentication, message integrity, and non-repudiation of routing messages. It does not use any mutable field in the route request or reply, thereby eliminating the need to protect the message from malicious modifications along the path. Further, to prevent spoofing, messages are authenticated (using certificates) at every hop along the path.

The Secure Ad hoc On-Demand Distance Vector (SAODV) [32] similarly uses digital signatures to authenticate the non-mutable fields of the messages, and hash chains to secure the hop count information (the only mutable information in the messages). A hash chain is a chain of keys $< H_1, H_2, .., H_n >$ formed by repeated application of a hash function H to an initial key k: $H_i(k) = H(H_{i-1}(k))$. To use hash chain, first the hash chain is established on a node, and then its $H_n(k)$ is provided to other nodes as part of initialization. At runtime, the keys in the hash chain are used one at a time in the reverse order to encrypt the message. The key is released in a delayed manner after the message has reached all the intended receivers. Upon release, the key is first verified through repeated application of the hash function, and then the message is authenticated using the key.

Secure Efficient Distance Vector Routing (SEAD) [33] similarly uses one-way hash chain to protect sequence number and metric in a DSDV routing update. Each router knows one value in the hash chain of every other router. Each message contains a key from the hash chain chosen based on the sequence number and the metric value itself. The key can be verified by the destination using repeated application of the hash function.

Ariadne [34] is a secure version of DSR, that protects the router list in the REQUEST from malicious deletion of hosts. It uses TESLA broadcast authentication [106], which in turns is based on hash chains, with delayed key release. Each hop authenticates new information in the REQUEST. The target buffers the REPLY until intermediate nodes can release the corresponding TESLA keys.

Most of these protocols were designed with the constraints of a mobile ad hoc network in mind; some of which do not apply to a wireless mesh network. For instance, several of them assume absence of any wired infrastructure connectivity which is not a constraint for a wireless mesh network whose very purpose is to provide access to wired infrastructure. In contrast, Hyacinth's protocol are designed with the goal of securing an enterprise-class wireless mesh network.

7.5.2 Internet Routing Security

The problem of faulty or malicious routers advertising false routes or refusing to serve advertised routes has been studied in the context of Wired Internet [107].

Secure BGP (S-BGP) [108] represents one kind of proposals. Specifically, S-BGP requires routing information to be authenticated so that routers advertising incorrect routing information can be held accountable if and when detected. Due to the large scale of the network, S-BGP simply uses the public key cryptographic signature to sign all route advertisements.

Routing Arbiter [109] established a centralized database of 'routing assets' which includes information such as the routing policy and route announcements from different ASes. This information can be queried by a router to perform some basic validity checks on the advertised routes it receives. Similarly in soBGP [37], each AS distributes its local topology which is used to form the global topology database. The topology database is then used to validate received routes.

Interdomain Routing Validation (IRV) [35] takes a similar approach but distributes the function of providing this routing assets information to individual ASes. Each AS contains an IRV server that receives a copy of each route update sent from any router in the AS. Upon reception of a route update, a BGP router queries its local IRV server, which in turn queries the relevant IRV servers to confirm the route.

Listen and Whisper [36] is another pair of mechanisms. *Listen* passively probes the data plane to ensure existence of advertised routes. Specifically, it listens to TCP exchanges to infer if there are hosts advertised by an AS, but not reachable through that AS. *Whisper* introduces signature field in every BGP update, which is updated by every AS along a path. Upon receiving updates from multiple paths, the routes are cross-checked for consistency using their signature fields. Additionally, to contain the effect of malicious routers, each router counts how often an AS appears on invalid routes, and assigns this count as a penalty value to the AS. The aggregate penalty along the routes is then minimized while making routing decisions.

7.5.3 Centralized Internet Control Plane

The 100x100 Clean Slate Project [58] is investigating the question: Assuming we were not bound by the design and implementation of the current Internet, what should this *clean slate design* be? A major design put forward by the project is to move towards a really thin control plane on individual routers that is focused on information dissemination and response to to explicit configuration instructions [110]. This refactoring of the IP control plane can significantly help with the performance, reliability, and security concerns that plague the current Internet. We make a similar argument in the context of WMN, where efficient wireless link resource utilization and secure operations are equally critical. Moreover, unlike the Internet WMN technology is still in a relatively nascent phase, and a *clean slate approach* is much more practical.

7.5.4 Secure Data Communication

One way to secure communication is to utilize multiple paths. SMT/SRP [111, 112] discovers and uses multiple disjoint paths between every source and destination. Each packet is dispersed across all these paths with additional redundancy. As long as M out of N parts reach the destination, the message can be reconstructed.

Hyacinth secure data communication by constantly monitoring the incoming packet stream on the receiver for signs of attacks, and initiating fine-grained packet tracing. Our fine-grained packet tracing builds upon ideas proposed by past researchers [113, 114]. In [113], authors proposed performing binary search to locate a faulty or malicious link along a path. However, it is relatively easy for an attacker to defeat the scheme as it can know whether the packet forwarding is being monitored or not. Hyacinth's design makes it virtually impossible for an attacker to detect if the packet forwarding is being monitored.

Secure Traceroute [114] is a refinement of the proposal in [113]. Probe packets in secure traceroute are indistinguishable from ordinary packets. This makes it hard for attacker to forward packets correctly only when probing is in progress. However, to make secure traceroute robust against multiple attackers on the route, authors use linear search instead of binary search. Similar to secure traceroute, Hyacinth makes it impossible to distinguish whether an attacker identification is in progress or not. However, Hyacinth's design makes several advancements over secure traceroute. Hyacinth uses binary search and works in presence of multiple independent attackers along a path. Hyacinth also enables identification of a malicious node launching a delay attack. Moreover, Hyacinth is adopted to the context of WMN and we make its packet tracing really lightweight: Turning it ON/OFF only requires one packet being sent from an end host to the prober/snooper node. There is some constant storage overhead, but no communication overhead until the attack is observed the next time. It can thus detect occasional attacks that are hard to detect otherwise. Finally, Hyacinth integrates this mechanism well into the core protocol, and provides an always-ON security against malicious nodes.

Authors in [115] proposed a watchdog technique to enable a node to check that a neighboring node do forward packets onward without tampering. It relies on the broadcast nature of the wireless media on each node to listen to the packets sent by its next hop and checks their headers and payloads for any signs of tampering. Catch Protocol [59] enhances this watchdog technique further through use of anonymous messages. The main problem solved by the Catch protocol is that of preventing free riding behavior in a multi-hop wireless network. In Catch protocol, each node periodically sends anonymous messages to its neighbors and require each neighbor to rebroadcast the message. This simple idea forces a would-be free rider to either drop all anonymous messages and get disconnected from the network, or forward all anonymous messages and reveal its true radio connectivity with its neighbors. In Hyacinth, we apply this technique to make the topology discovery secure and robust.

As we showed in Chapter 5, one cannot rely on watchdog technique in a multi-radio WMN to secure the routing protocol computation. Moreover, applying watchdog technique to secure packet forwarding in a multi-radio WMN would also incur significantly more

hardware, as a packet broadcast by a node can no longer be heard by all its radio neighbors.

7.5.5 PKI for Ad Hoc Networks

Much research to provide security in the context of ad hoc networks *avoids* use of any centralized PKI infrastructure and instead rely upon distributed mechanisms. Establishing a PGP Web-of-Trust (originally put forth by PGP creator Phil Zimmermann in 1992) is one such approach. In this model, any node in the network can issue certificate for another node that it trusts [116]. A node N1 can then authenticate another node N2 as long as N1 can find a chain of certificates leading upto N2. [117] discusses mechanisms to discover such certificate chains in the context of ad hoc networks.

Another approach to distribute the responsibility of a CA is based on the idea of Threshold Cryptography [118]. MOCA (Mobile Certificate Authority) uses this approach [119]. In MOCA, the task of giving out certificates is delegated to n MOCA nodes out of total N network nodes. Each of these n MOCA nodes can provide partial signature for the certificate. As long as a node can contact k out of n MOCA nodes, it can reconstruct the full signature and hence the certificate.

These approaches are more useful in the context of ad hoc networks, where the connectivity to some infrastructure is not always possible. On the other hand, any enterprise-class wireless mesh network is connected to the wired network via gateways. It is therefore more practical to just set up a CA as part of the infrastructure.

A related problem is that of compromised node cloning [120]. [121] uses locationing information to probabilistically detect cloning attack. [122] detects presence of clones using statistics on the frequency with which a set of keys is used for authentication. Hyacinth's detection of node cloning attack works by (1) making the topology discovery more accurate using anonymized messages [59], and (2) using a threshold-based mechanism.

7.5.6 Secure Time Synchronization

Time synchronization has been studied extensively in the overall context of distributed systems and computer networks. Time synchronization is also a requirement for wireless sensor networks if the data collected from various sensors is to be aggregated. The techniques proposed there range from synchronizing using reference broadcasts and flooding to a tree-based synchronization [123]. As pointed in [124], many of these protocols do not work well in presence of compromised nodes. Several techniques have also been proposed to counter these attacks [124].

Hyacinth secures the synchronization of time by always synchronizing each node to the CU. Although the communication happens over multiple hops, the actual synchronization protocol is not aware of that. This is similar to the basic Secure Opportunistic Multi-hop synchronization used in [125]. By itself, this is insufficient in presence of attackers along the path launching packet delay attacks. We therefore have another mechanism to detect such delay attacks. This is done by constantly monitoring the RTT for signs of malicious

delays. The idea of monitoring RTT for outliers is not novel and has been proposed earlier in [126].

Chapter 8

Conclusions And Future Directions

Today's enterprise wireless LAN deployments require substantial human expertise to perform several static optimizations such as access point's radio channel assignment and transmit power level setting [3]. These optimizations are not only cumbersome to perform, but are also quickly outdated as the enterprise RF environment changes. Ideally, a wireless LAN deployment should only involve ensuring blanket coverage of the entire enterprise using access points, while the access points should automatically and continuously configure the optimal channels and transmit power levels based on the latest user locations and traffic demands. Ironically, a major obstacle in realizing this vision of self-deployed wireless LANs is that each access point needs to be connected to the enterprise computing resources using a wired backbone [3]. Wiring is an even bigger issue for outdoor campus-wide wireless LAN deployment and ISP last-mile networks.

In this doctoral research, we architected a wireless mesh network that can eliminate most of this wiring overhead. In a wireless mesh network (WMN), neighboring access points communicate with each other using direct wireless links, while distant access points communicate using multiple wireless hops. Our research tackled the following challenges in forming an effective enterprise backbone using a WMN: (1) providing high capacity, (2) supporting flow-level fairness, and (3) ensuring secure protocol operations. Specifically, we proposed three novel implementation techniques: (1) multi- channel mesh networking, (2) stateful transport protocol, and (3) secure routing protocol. To demonstrate the practicality of these techniques, we implemented and evaluated a 9-node IEEE 802.11a-based mesh prototype. Many of the techniques are equally applicable to other wireless network technologies such as IEEE 802.11n- and WiMax-based mesh networks.

- Multi-channel Wireless Mesh Networking: Limited capacity remains a pressing issue even for single-hop wireless LANs, let alone a multi-hop WMN where inter-path and intra-path interference limit the number of links that can be simultaneously active in the network. To address the capacity issue, the Hyacinth architecture we proposed aggregates the bandwidth available across different radio channels within a single WMN. Specifically, we employ multiple radio channels in each radio neighborhood by equipping each node with multiple network interfaces. To fully utilize

the performance potential of this approach, we devised two traffic load-aware channel assignment and routing algorithms, each of which tunes the network channel assignment and routing based on the network topology and the latest traffic patterns. Even with the use of just 2 NICs per node, the proposed algorithms improve the network cross-section goodput by over 6 times when compared with single-channel WMNs. The two papers [23, 21] based on this work have initiated lot of research in multi-radio wireless mesh networks and are widely cited for their original contributions.

- Stateful Transport Protocol: TCP is the de facto transport protocol of the Internet. However, TCP demonstrates several undesirable behaviors from both utilization and fairness standpoints, when operating in multi-hop wireless networks. First, TCP ACK traffic consumes up to 20% of the wireless channel bandwidth. Second, TCP's congestion control does not adapt quickly to changing wireless link bandwidth. Third, a TCP sender slows down in presence of bit-error related packet losses. Each of these issues leads to reduction in effective bandwidth available to the application layers. To address these issues, we proposed a stateful transport protocol, called link-aware reliable transport protocol (LRTP) [24]. The key feature of LRTP is an explicit rate-based congestion control mechanism in which each intermediate wireless router measures the effective bandwidth of each of its wireless links, fairly allocates it among those flows traversing the wireless link, and then informs the corresponding senders to adjust their sending rates. To address the hidden terminal problem, LRTP also incorporates a coordinated congestion control mechanism [25, 26] that first determines the inhibition relationship between different routers, and performs global bandwidth allocation in a manner, thereby providing end-to-end max-min flow fairness on top of unfair IEEE 802.11 MAC protocol. Our evaluations of LRTP on the mesh prototype show that it substantially improves the network utilization and flow fairness in multi-channel WMNs.

- Secure Routing Protocol: Security remains a major roadblock to widespread deployment of 802.11 one-hop infrastructure networks. The problem is exacerbated in a wireless mesh network, where a single compromised router can easily disrupt the network routing state by tampering with control communication or advertising crafted topology/traffic data. The failure probability of a WMN therefore increases exponentially with the network size. We developed a centralized network architecture that incorporates security as a first-class requirement at par with connectivity and performance. The protocol secures all core aspects of the mesh network – topology and traffic statistics collection, route and channel computation, route decision dissemination, network reconfiguration, and also packet forwarding. It can quickly detect most common misbehaviors and trace the problem down to specific nodes. The secure routing mechanisms significantly enhance the availability of a Hyacinth network when a WMN node is compromised, misconfigured, or broken.

8.1 Future Directions

Wireless mesh networks are now being deployed at city-wide scale to provide ubiquitous wireless broadband connectivity [7, 127]. Additionally, the trend towards cross-layer optimizations [128] and use of multiple radios with smart antennas [129] will make the wireless routers more and more complex. Based on our experiences, *comprehensive network management* is a key element to the success of such large-scale wireless mesh networks. Network management is the broad term that covers network monitoring, configuration management, fault management, performance management, and protocol security. The research question here is: What is the ideal architecture and set of protocols needed to make large-scale wireless mesh networks manageable? Below are some of the specific research problems worth pursuing in this direction.

8.1.1 RF Sensor Networking

Based on our survey of enterprise wireless LAN deployment [3], the most fundamental element of any network management is the ability to visualize the utilization of network resources (in this case wireless channels) at each location. This is especially important as a large-scale wireless mesh network needs to co-exist with, rather than compete with, other residential and office wireless LANs. The research questions here are: How to produce a comprehensive RF usage map of the network using the inherent RF-sensing capability of the individual mesh node? And how to utilize multiple radios (and potentially smart antennas) equipped on each mesh node to perform accurate multi-hop locationing of the entire network? Apart from network visualization, RF usage map is also useful for network optimizations such as deciding which channels to use on which router.

8.1.2 Hybrid Re-configuration

Most researchers frown upon the mention of centralized algorithms. Experiences with Internet have led to increasing realization that the control plane of large-scale networks needs to be centralized. Centralized algorithms have the benefit of starting with the global state information about the network. Our experience with devising channel assignment algorithms for a general multi-radio wireless mesh network suggests that the intricate dependency between configurations of different routers may itself sometimes warrant a centralized approach. We believe that one needs to take an intermediate approach, where a centralized algorithm performs longer-term optimizations of the network, while a distributed algorithm fine-tunes those decisions based on more frequently changing state of the network. For instance, the centralized algorithm can decide long-term routes based on global information, while the distributed algorithm can perform local repair of network routes in case of a traffic shift or a wireless router failure. Such hybrid approach can yield novel and practical network optimization algorithms.

8.1.3 Protocol Resilience

There are bound to be attacks on externally deployed wireless mesh networks. Use of cryptography alone does not solve the problem if the attacker has been able to compromise some of the wireless mesh nodes themselves, as they now have access to the keys and the software of the wireless router. The research questions here is: How to devise mesh optimization protocols that demonstrate graceful degradation in presence of compromised routers? Further, what intelligence needs to be built into the protocol so as to pinpoint the compromised routers? We already present a security architecture that can addresses these issues and keep a wireless mesh network operational in presence of independent compromised routers. A valid question to ask is: How can one deal with compromised nodes when they can collude?

8.1.4 Stateful Protocol Design

The stateful transport protocol we developed makes the routers aware of the transport protocol flows and substantially improves both the network utilization and flow fairness. This work can be generalized to implement application-aware wireless routers that improve network efficiency over existing wireless routers. Specifically, one can raise the following research questions: How to utilize the knowledge of application requirements for more optimal packet forwarding? How to perform inter-router co-ordination to utilize such application awareness? For instance, media traffic can tolerate certain amount of bit errors, while other applications may not. Similarly, different applications have different delay and jitter requirements that can be used to prioritize among flows/packets.

8.1.5 Quality Of Service

Many popular applications such as voice-over-IP and video-over-IP have specific end-to-end delay and bandwidth requirements that can not be satisfied by best-effort service. Unlike wired networks, WMN links in a radio neighborhood share the same channel and hence it is not sufficient to just differentiate and prioritize traffic at per-router level. A mechanism such as [130] is needed to prioritize traffic across nodes. However, current proposals require modifications to the IEEE 802.11 protocol to incorporate these techniques. We believe that it should be possible to implement these schemes, or modifications thereof, on top of unmodified 802.11 using a mechanism similar to overlay MAC layer [98]. Such a combined solution that can provide differentiated service to multimedia flows traversing an 802.11-based WMN has not yet been proposed.

8.1.6 Topology Planning

While the position of nodes in a WMN could be chosen randomly and the network would adapt to that given topology, a more planned approach could use the flexibility in initial positioning of nodes to optimize the network performance. Specifically, there are two

variables here – the position of regular nodes and the placement of the gateway nodes. The gateway placement problem has been explored in [131, 132]. The goal is to place a minimum number of gateways in these locations such that the bandwidth requirement of every ingress is met. The node placement problem is explored in [133], where network performance is optimized by modifying the existing topology incrementally. Each of these solutions only address parts of the general problem which still stands as a research question: How to come up with an optimal placement for a fraction of regular and gateway nodes, while the placement of remaining ones is given as input?

8.1.7 Directional Antenna-based Mesh Networks

The use of IEEE 802.11 has even been extended to build long-distance wireless mesh networks that can be used to provide Internet-connectivity to rural areas [8]. Digital Gangetic Plains [8] already deals with many of the IEEE 802.11 assumptions that break down in such settings. The problem of using directional antennas in a general IEEE 802.11 WMN remains unanswered [134, 135]. Specifically, given a WMN where each node is equipped with multiple interfaces each attached to a steerable directional antenna, a topology determination algorithm that determines the direction and channel of each interface is not yet known.

8.1.8 Fault Diagnosis

Automated fault detection and diagnosis has been the holy grail of network management. The problem is complicated in wireless mesh network as it is non-trivial to distinguish whether the poor performance is because of bad channel conditions, interference, software bugs, faulty hardware, or compromised nodes. [136] is one such proposal that works by running fault simulations and matching the observed and the expected performance. Our proposal in Chapter 5 of fine-grained packet tracing to pinpoint malicious or faulty node provides another example of works in this direction.

Bibliography

[1] D. Couto, D. Aguayo, J. Bicket, and R. Morris; "A High-Throughput Path Metric for Multi-Hop Wireless Routing," *Proceedings of the 9th ACM MobiCom, 2003.*

[2] Ashish Raniwala, and Tzi-cker Chiueh; "Coverage and Capacity Issues in Enterprise Wireless LAN Deployment" *ECSL Technical Report 166, Nov 2004. Available at http://www.ecsl.cs.sunysb.edu/tr/TR166.pdf*

[3] Ashish Raniwala, Tzi-cker Chiueh; "Deployment Issues in Enterprise Wireless LANs," *ECSL Technical Report 145, Sept 2004. Available at http://www.ecsl.cs.sunysb.edu/tr/wlandeployment.pdf*

[4] Strix Networks; *http://www.strixsystems.com*

[5] FireTide Inc; *http://www.firetide.com*

[6] Mesh Networks Inc; *http://www.meshnetworks.com*

[7] R. Karrer, A. Sabharwal, E. Knightly; "Enabling Large-scale Wireless Broadband: The Case for TAPs," *Proceedings of the ACM Workshop on HotNets, 2003.*

[8] P. Bhagwaty, B. Ramanz, D. Sanghi; "Turning 802.11 Inside-Out," *Proceedings of the ACM Workshop on HotNets, 2003.*

[9] P. Hsiao, A. Hwang, H. Kung, D. Vlah; "Load-Balancing Routing for Wireless Access Networks," *Proceedings of the IEEE INFOCOM, 2001.*

[10] P. Gupta, P. R. Kumar; "The capacity of wireless networks," *IEEE Transactions on Information Theory, Vol. 46, No. 2. (2000), pp. 388-404.*

[11] J. Jun, M. L. Sichitiu; "The nominal capacity of wireless mesh networks" *IEEE Wireless Communications, Vol. 10, No. 5. (2003), pp. 8-14.*

[12] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers, and Robert Morris; "Performance of Multihop Wireless Networks: Shortest Path is Not Enough," *Proceedings of the ACM Workshop on HotNets, 2002.*

[13] Saad Biaz, Nitin H. Vaidya; "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver," *Proceedings of the IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET), 1999.*

[14] S. Xu, T. Saadawi "Does IEEE 802.11 Work Well in Multi-hop Wireless Network?" *IEEE Communications Magazine, Vol. 39, No. 6. (2001), pages 130-137.*

[15] Y. Liu, E. Knightly; "Opportunistic Fair Scheduling over Multiple Wireless Channels," *Proceedings of the IEEE INFOCOM, 2003.*

[16] J. So, N. Vaidya; "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using A Single Transceiver," *Proceedings of the ACM MobiHoc, 2004.*

[17] A. Tzamaloukas, J. J. Garcia-Luna-Aceves; "A Receiver-Initiated Collision-Avoidance Protocol for Multi-channel Networks," *Proceedings of the IEEE INFOCOM, 2001.*

[18] A. Nasipuri, S. Das; "A Multichannel CSMA MAC Protocol for Mobile Multihop Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 1999.*

[19] Jean Cao, Yujing Wu, and Carey Williamson; "A Station-Based Adaptation Algorithm to Improve Robustness of IEEE 802.11," *Proceedings of the Int'l Workshop on Wireless Mobile Multimedia, 2006.*

[20] A. Bakre and B. R. Badrinath; "I-TCP: Indirect TCP for Mobile Hosts," *Proceedings of the Int'l Conf on Distributed Computing Systems (ICDCS), 1995.*

[21] A. Raniwala, K. Gopalan, T. Chiueh; "Centralized Algorithms for Multi-channel Wireless Mesh Networks," *ACM Mobile Computing and Communications Review (MC2R), April 2004.*

[22] Ashish Raniwala, Tzi-cker Chiueh; "Evaluation of a Wireless Enterprise Backbone Network Architecture," *Proceedings of the 12th IEEE Hot Interconnects, 2004.*

[23] Ashish Raniwala, Tzi-cker Chiueh; "Architecture and Algorithms for an IEEE 802.11-based Wireless Mesh Network," *Proceedings of the IEEE INFOCOM, 2005.*

[24] Ashish Raniwala, Srikant Sharma, Pradipta De, Rupa Krishnan, Tzi-cker Chiueh; "Evaluation of a Stateful Transport Protocol for Multi-channel Wireless Mesh Networks," *Proceedings of the Int'l Workshop on Quality of Service (IWQoS), 2007.*

[25] Ashish Raniwala, Pradipta De, Srikant Sharma, Rupa Krishnan, Tzi-cker Chiueh; "End-to-End Flow Fairness over IEEE 802.11-based Wireless Mesh Networks," *Proceedings of the IEEE INFOCOM Mini-Sympsium on 802.11 Wireless LANs, 2007.*

[26] Ashish Raniwala, Pradipta De, Srikant Sharma, Rupa Krishnan, Tzi-cker Chiueh; "Globally Fair Radio Resource Allocation for Wireless Mesh Networks," *Proceedings of the the 17th IEEE Int'l Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2009.*

[27] I. Katzela, M. Naghshineh; "Channel assignment schemes for cellular mobile telecommunication systems: a comprehensive survey," *IEEE Personal Communications, June 1996.*

[28] T. R. Jensen, B. Toft; "Graph Coloring Problems," *Wiley Interscience, New York, 1995.*

[29] S. Lee and M. Gerla; "Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks," *Proceedings of the IEEE Int'l Conference on Communications (ICC), 2001.*

[30] V. Gambiroza, B. Sadeghi and E. Knightly; "End-to-end performance and fairness in multihop wireless backhaul networks," *Proceedings of the ACM MobiCom, 2004.*

[31] Kimaya Sanzgiri, Daniel LaFlamme, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer; "Authenticated Routing for Ad hoc Networks," *IEEE Journal on Selected Areas in Communications (JSAC) 23.3 (2005): 598-610.*

[32] Manel Guerrero Zapata: "Secure Ad hoc On-Demand Distance Vector (SAODV) Routing," *INTERNET-DRAFT draft-guerrero-manet-saodv-04.txt. September 2005.*

[33] Yih-Chun Hu, David B. Johnson, and Adrian Perrig; "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," *Ad Hoc Networks Journal, (2003), pages 175-192.*

[34] Yih-Chun Hu, Adrian Perrig, David B. Johnson; "Ariadne: A secure On-Demand Routing Protocol for Ad hoc Networks," *Proceedings of the ACM MobiCom, 2002.*

[35] Goodell, G., Aiello, W., Griffin, T., Ioannidis, J., McDaniel, P., and Rubin, A.; "Working around BGP: An incremental approach to improving security and accuracy of interdomain routing," *Proceedings of the Internet Society Network and Distributed Systems Security, 2003.*

[36] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, Randy H. Katz ; "Listen and Whisper: Security Mechanisms for BGP," *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2004.*

[37] J. Ng "Extensions to BGP to Support Secure Origin BGP (soBGP)," *June 2003, INTERNET DRAFT, http://www.watersprings. org/pub/id/draft-ng-sobgp-bgp-extensions-01.txt*

[38] Jie Gao, Li Zhang; "Load Balanced Short Path Routing in Wireless Networks," *Proceedings of the IEEE INFOCOM, 2004.*

[39] A. Caprara, H. Kellerer, U. Pferschy; "The Multiple Subset Sum Problem," *SIAM Journal on Optimization, Vol 11, 2000.*

[40] K. Gopalan; "Efficient Network Resource Allocation with QoS Guarantees," *ECSL Technical Report 133. Available at http://www.ecsl.cs.sunysb.edu/tr/TR133.pdf*

[41] Subhash Suri, Marcel Waldvogel, Priyank Ramesh Warkhede. "Profile-Based Routing: A New Framework for MPLS Traffic Engineering," *Proceedings of the Int'l Workshop on Quality of future Internet Services, 2001.*

[42] "IEEE 802.1D - MAC Bridges," *http://standards.ieee.org/getieee802/download/802.1D-1998.pdf*

[43] C. Villamizar, R. Chandra, R. Govindan; "Internet RFC 2439 - BGP Route Flap Damping"

[44] R. Draves, J. Padhye, and B. Zill; "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," *Proceedings of the ACM MobiCom, 2004.*

[45] Tropos Networks; *http://www.tropos.com*

[46] J. Cao, *et. al.*; "Stochastic Models for Generating Synthetic HTTP Source Traffic" *Proceedings of the IEEE INFOCOM, 2004.*

[47] S. Sharma, N. Zhu, T. Chiueh; "Low-Latency Handoffs for Infrastructure mode Wireless LANs," *IEEE Journal on Selected Areas in Communications (JSAC), 2004.*

[48] K. Sundaresan, V. Anantharaman, H. Y. Hsieh, and R Sivakumar; "ATP: A Reliable Transport Protocol for Ad Hoc Networks," *Proceedings of the ACM MobiHoc, 2003.*

[49] K. Chen, K. Nahrstedt, N. Vaidya; "The Utility of Explicit Rate-Based Flow Control in Mobile Ad Hoc Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 2004.*

[50] L. Tassiulas and S. Sarkar. "Maxmin fair scheduling in wireless networks," *Proceedings of the IEEE INFOCOM, 2002.*

[51] K. Jain, J. Padhye, V. N. Padmanabhan, L. Qiu; "Impact of interference on multi-hop wireless network performance," *Proceedings of the ACM MobiCom, 2003.*

[52] Jitendra Padhye, Sharad Agarwal, Venkata N. Padmanabhan, Lili Qiu, Ananth Rao, and Brian Zill; "Estimation of link interference in static multi-hop wireless networks" *Proceedings of the Internet Measurement Conference (IMC), 2005.*

[53] C. Bron, J. Kerbosch; "Algorithm 457: finding all cliques of an undirected graph," *ACM Communications, Volume 16, Issue 9, September 1973.*

[54] P. De, A. Raniwala, S. Sharma, and T. Chiueh; "MiNT: A Miniaturized Network Testbed for Mobile Wireless Research," *Proceedings of the IEEE INFOCOM, 2005.*

[55] Raj Jain, Shiv Kalyanaraman and Ram Viswanatha; "Rate Based Schemes: Mistakes to Avoid," *ATM Forum/94-0882, September 1994.*

[56] A. Miu, G. Tan, H. Balakrishnan, and J. Apostolopoulos; "Divert: Fine-grained Path Selection for Wireless LANs" *Proceedings of the 2nd ACM MobiSys, 2004.*

[57] R. Jain, A. Durresi, G. Babic; "Throughput Fairness Index: An Explanation," *ATM Forum/99-0045, February 1999.*

[58] "100x100 Clean Slate Project," *http://www.100x100network.org*

[59] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan; "Sustaining cooperation in multi-hop wireless networks" *Proceedings of the 2nd USENIX NSDI, 2005.*

[60] "The MadWifi Project," *http://madwifi-project.org*

[61] "Using HostAP with MadWifi," *http://madwifi-project.org/wiki/UserDocs/HostAP*

[62] Jean Tourrilhes; "Wireless Tools for Linux," *http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html*

[63] V. Navda, A. Kashyap and S. R. Das; "Design and Evaluation of iMesh: an Infrastructure-mode Wireless Mesh Network," *Proceedings of the 6th IEEE Int'l Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM), 2005.*

[64] Andreas Haeberlen, Eliot Flannery, Andrew Ladd, Algis Rudys, Dan S. Wallach and Lydia E. Kavraki; "Practical Robust Localization over Large-Scale IEEE 802.11 Wireless Networks," *Proceedings of the ACM MobiCom, 2004.*

[65] D Niculescu, B Nath; "Error characteristics of ad hoc positioning systems (APS)," *Proceedings of the ACM MobiHoc, 2004.*

[66] Pradipta De, Rupa Krishnan, Ashish Raniwala, Krishna Tatavarthi, Nadeem Syed, Srikant Sharma, and Tzi-cker Chiueh; "MiNT-m: An Autonomous Mobile Wireless Experimentation Platform," *Proceedings of the Int'l Conference on Mobile Systems, Applications, and Services (Mobisys), 2006.*

[67] "RouterBOARD 230," *http://www.routerboard.com/rb200.html*

[68] A. Muir and J.J. Garcia-Luna-Aceves; "A Channel Access Protocol for Multihop Wireless Networks with Multiple Channels," *Proceedings of the IEEE Int'l Conference on Communications (ICC), 1998.*

[69] Asis Nasipuri and Samir R. Das; "A Multichannel CSMA MAC Protocol for Mobile Multihop Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 1999.*

[70] Paramvir Bahl, Ranveer Chandra, and John Dunagan; "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," *Proceedings of the ACM MobiCom, 2004.*

[71] Pradeep Kyasanur and Nitin H. Vaidya; "Routing and Interface Assignment in Multi-Channel Multi-Interface Wireless Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 2005.*

[72] Mahesh Marina, Samir Das; "A Topology Control Approach to Channel Assignment in Multi-Radio Wireless Mesh Networks," *Proceedings of the IEEE Broadnets, 2005.*

[73] Anand Subramanian, Rupa Krishnan, Samir Das, Himanshu Gupta; Anand Prabhu Subramanian, Himanshu Gupta, Samir R. Das, Jing Cao; "Minimum Interference Channel Assignment in Multiradio Wireless Mesh Networks," *IEEE Transactions on Mobile Computing, vol. 7, no. 12, pp. 1459-1473, Apr. 2008.*

[74] Arindam K. Das, Hamed M. K. Alazemi, Rajiv Vijayakumar, Sumit Roy; "Optimization Models for Fixed Channel Assignment in Wireless Mesh Networks with Multiple Radios," *Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005.*

[75] Bong-Jun Ko, V. Misra, J. Padhye, D. Rubenstein; "Distributed Channel Assignment in Multi-Radio 802.11 Mesh Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 2007.*

[76] Krishna N. Ramachandran, Elizabeth M. Belding, Kevin C. Almeroth, and Milind M. Buddhikot; "Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks," *Proceedings of the IEEE INFOCOM, 2006.*

[77] Mansoor Alicherry, Randeep Bhatia, Li (Erran) Li; "Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Networks," *Proceedings of the ACM MobiCom, 2005.*

[78] X. Lin, S. Rasool; "A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad-hoc Wireless Networks," *Proceedings of IEEE INFOCOM, 2007.*

[79] E. Royer, C. Toh; "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications, April 1999.*

[80] A. Iwata, C. Chiang, G. Pei, M. Gerla, T. Chen. "Scalable Routing Strategies for Ad-hoc Wireless Networks.," *IEEE Journal on Selected Areas in Communications (JSAC), 1999.*

[81] H. Hassanein, A. Zhou; "Routing with load balancing in wireless Ad hoc networks," *Proceedings of the ACM Int'l Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), 2001.*

[82] S.J. Lee and M. Gerla; "Dynamic Load-Aware Routing in Ad hoc Networks," *Proceedings of the Int'l Conference on Communications (ICC), 2001.*

[83] Kui Wu, Janelle Harms; "Performance Study of a Multipath Routing Method for Wireless Mobile Ad Hoc Networks," *Proceedings of the Ninth Int'l Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2001.*

[84] M. K. Marina and S. R. Das; "Ad hoc On-demand Multipath Distance Vector Routing," *Proceedings of the IEEE Internaional Conference ICNP, 2001.*

[85] Sanjit Biswas and Robert Morris. "Opportunistic Routing in Multi-Hop Wireless Networks" *Proceedings of the ACM SIGCOMM, 2005.*

[86] Yuan Yuan, Hao Yang, Starsky H.Y. Wong, Songwu Lu, and William Arbaugh; "ROMER: Resilient Opportunistic Mesh Routing for Wireless Mesh Networks," *Proceedings of the IEEE Workshop on Wireless Mesh Netowrks (WiMesh), 2005.*

[87] Hari Balakrishnan, Venkata N. Padmanabhan, Randy H. Katz; "The Effects of Asymmetry on TCP Performance," *Proceedings of the ACM Mobile Computing and Networking, 1997.*

[88] Thiemo Voigt, Adam Dunkels, and Juan Alonso; "Reliability in Distributed TCP Caching," *Proceedings of the Sensor Networks Workshop at Informatik, 2004.*

[89] C.Y. Wan and A.T. Campbell; "PSFQ: A Reliable Transpor Protocol For Wireless Sensor Networks" *Proceedings of the ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA), 2002.*

[90] Pravin Bhagwat, Partha Bhattacharya, Arvind Krishma and Satish K. Tripathi; "Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs," *Wireless Network, 1997, Vol 3, No 1, Kluwer Academic Publishers.*

[91] H. Balakrishnan, V. N. Padmanabhan, S. Seshan and R. H. Katz; "A Comparison of Mechanisms for Improving TCP performance over Wireless Links," *Proceedings of the ACM SIGCOMM, 1996.*

[92] Gavin Holland, Nitin Vaidya; "Analysis of TCP performance over mobile ad hoc networks," *Proceedings of the ACM MobiCom, 1999.*

[93] Zhenghua Fu, Petros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang and Mario Gerla,; "On TCP performance in multihop wireless networks," *UCLA WiNG Technical Report, 2002.*

[94] R. Chakravorty, A. Clark, and I. Pratt; "GPRSWeb: Optimizing the Web for GPRS Links," *Proceedings of the USENIX Mobisys, 2003.*

[95] H. Balakrishnan and R. H. Katz; "Explicit Loss Notification and Wireless Web Performance," *Proceedings of the IEEE Globecom, 1998.*

[96] P. Sinha, N. Venkitaraman, R. Sivakumar and V. Bharghavan; "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks," *Proceedings of the ACM MobiCom, 1999.*

[97] Kaixin Xu, Mario Gerla, Lantao Qi, Yantai Shu; "Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED," *Proceedings of the ACM MobiCom, 2003.*

[98] Ananth Rao and Ion Stoica; "An Overlay MAC Layer for 802.11 Networks," *Proceedings of the USENIX Mobisys, 2005.*

[99] H. Luo, S. Lu, V. Bharghavan "A New Model for Packet Scheduling in Multihop Wireless Networks," *Proceedings of the ACM MobiCom, 2000.*

[100] X. Huang, B. Bensaou; "On Max-min Fairness and Scheduling in Wireless Ad-Hoc Networks: Analytical Framework and Implementation," *Proceedings of the ACM MobiHoc, 2001.*

[101] Thyagarajan Nandagopal, Tae-Eun Kim, Xia Gao, and Vaduvur Bharghavan; "Achieving MAC Layer Fairness in Wireless Packet Networks," *Proceedings of the ACM/IEEE MobiCom, 2000.*

[102] B. Li; "End-to-End Fair Bandwidth Allocation in Multi-hop Wireless Ad Hoc Networks," *Proceedings of the Int'l Conference on Distributed Computing Systems (ICDCS), 2005.*

[103] Y. Yi and S. Shakkottai; "Hop-by-hop congestion control over a wireless multi-hop network," *Proceedings of the IEEE INFOCOM, 2004.*

[104] Yih-Chun Hu, A. Perrig; "A survey of secure wireless ad hoc routing," *IEEE Security and Privacy, Volume 2, Issue 3, May-June 2004.*

[105] C. Karlof, D. Wagner; "Secure routing in wireless sensor networks: attacks and countermeasures," *Proceedings of the First IEEE Int'l Workshop on Sensor Network Protocols and Applications, 2003.*

[106] A. Perrig, *et.al.*; "RFC 4082 - Timed Efficient Stream Loss-Tolerant Authentication," *http://www.ietf.org/rfc/rfc4082.txt*

[107] S. Murphy; "RFC 4272 - BGP Security Vulnerabilities Analysis," *http://www.ietf.org/rfc/rfc4272.txt*

[108] S. Kent, C. Lynn, and K. Seo; "Secure Border Gateway Protocol (Secure-BGP)," *IEEE Journal on Selected Areas in Communications (JSAC), Vol. 18, No. 4, April 2000.*

[109] "The Routing Arbiter Project," *http://www.merit.edu/networkresearch/proje cthistory/routingartiber*

[110] J. Rexford, A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, G. Xie, J. Zhan, H. Zhang; "Network-Wide Decision Making: Toward A Wafer-Thin Control Plane," *Proceedings of the ACM Workshop on HotNets, 2004.*

[111] Panagiotis Papadimitratos and Zygmunt Haas ; "Secure Data Transmission in Mobile Ad Hoc Networks", *Proceedings of the ACM Workshop on Wireless Security (WiSe), 2003.*

[112] Panagiotis Papadimitratos and Zygmunt J. Haas; "Secure routing for mobile ad hoc networks," *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS), 2002.*

[113] B. Awerbuch, D. Holmer, C. Nita-Rotaru, H. Rubens; "An on-demand secure routing protocol resilient to byzantine failures," *Proceedings of the 1st ACM workshop on Wireless security (WiSE), 2002.*

[114] Venkata N. Padmanabhan, Daniel R. Simon; "Secure traceroute to detect faulty or malicious routing," *ACM SIGCOMM Computer Communication Review, Vol 33, Issue 1, Jan 2003.*

[115] S. Marti, T. J. Giuli, K. Lai, M. Baker; "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proceedings of the ACM MobiCom, 2000.*

[116] Jean-Pierre Hubaux, Levente Buttyán, Srdan Capkun; "The Quest for Security in Mobile Ad Hoc Networks," *Proceedings of the 2nd ACM MobiHoc, 2001.*

[117] H. Mohri, I. Yasuda, Y. Takata, H. Seki; "Certificate Chain Discovery inWeb of Trust for Ad Hoc Networks," *Proceedings of the 21st Int'l Conference on Advanced Information Networking and Applications Workshops (AINAW), 2007.*

[118] A. Shamir; "How to Share a Secret?" *Communications of the ACM, 1979.*

[119] S. Yi, R. Kravets; "MOCA : Mobile Certificate Authority for Wireless Ad Hoc Networks," *Proceedings of the Second Annual PKI Research Workshop (PKI), 2003.*

[120] J. Newsome, E. Shi, D. Song, A. Perrig; "The Sybil Attack in Sensor Networks: Analysis & Defenses," *Proceedings of Third Int'l Symposium on Information Processing in Sensor Networks (IPSN), 2004.*

[121] B. Parno, A. Perrig, V. Gligor; "Distributed detection of node replication attacks in sensor networks," *Proceedings of IEEE Symposium on Security and Privacy, 2005.*

[122] R. Brooks, P. Y. Govindaraju, M. Pirretti, N. Vijaykrishnan, M. T. Kandemir; "On the Detection of Clones in Sensor Networks Using Random Key Predistribution," *IEEE Transactions on Systems, Man, and Cybernetics, Vol 37 Issue 6, Nov 2007.*

[123] F. Sivrikaya, B. Yener; "Time Synchronization in Sensor Networks: A Survey," *IEEE Network, Vol 18 Issue 4, July-Aug 2004.*

[124] M. Manzo, T. Roosta, S. Sastry; "Time synchronization attacks in sensor networks," *Proceedings of the ACM Workshop on Security of Ad hoc and Sensor Networks, 2005.*

[125] Hui Song, Sencun Zhu, Guohong Cao; "Attack-Resilient Time Synchronization for Wireless Sensor Networks," *Proceedings of the IEEE Int'l Conference on Mobile Ad-hoc and Sensor Systems, 2005.*

[126] S. Ganeriwal, Ch. Popper, S. Capkun, M. B. Srivastava; "Secure Time Synchronization in Sensor Networks," *ACM Transactions on Information and System Security, July 2008.*

[127] M. Audeh; "Metropolitan-Scale Wi-Fi Mesh Networks," *IEEE Computer Vol 37, No 12, Dec 2004.*

[128] S. Shakkottai, T. S. Rappaport, P. C. Karlsson; "Cross-layer design for wireless networks," *IEEE Communications Magazine, 2003.*

[129] J. A. Stine; "Exploiting smart antennas in wireless mesh networks using contention access," *IEEE Wireless Communications, Vol 13 Issue 2, April 2006.*

[130] Vikram Kanodia, Chengzhi Li, Ashutosh Sabharwal, Bahareh Sadegh, and Edward Knightly; "Distributed Multi-Hop Scheduling and Medium Access with Delay and Throughput Constraints," *Proceedings of the ACM MobiCom, 2001*.

[131] Yigal Bejerano; "Efficient Integration of Multi-hop Wireless and Wired Networks with QoS Constraints," *Proceeding of the ACM MobiCom, 2002*.

[132] Ranveer Chandra, Lili Qiu, Kamal Jain and Mohammad Mahdian; "Optimizing the Placement of Integration Points in Multi-hop Wireless Networks," *Proceedings of the Int'l Conference on Network Protocols (ICNP), 2004.*

[133] Anindya Basu, Brian Boshes, Sayandev Mukherjee, Sharad Ramanathan; " Network Deformation: Traffic-Aware Algorithms for Dynamically Reducing End-to-end Delay in Multihop Wireless Networks," *Proceeding of the ACM MobiCom, 2004.*

[134] Gupqing Li, Lily Yang, W. Steven Conner, Bahar Sadeghi; "Opportunities and Challenges in Mesh Networks Using Directional Anntenas," *Proceedings of the IEEE Workshop on Wireless Mesh Netowrks (WiMesh), 2005.*

[135] Robert Vilzmann, Christian Bettstetter, Christian Hartmann; "On the Impact of Beamforming on Interference in Wireless Mesh Networks," *Proceedings of the IEEE Workshop on Wireless Mesh Netowrks (WiMesh), 2005.*

[136] A. Adya, P. Bahl, R. Chandra, L. Qiu; "Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks," *Proceedings of the ACM MobiCom, 2004.*

[137] H. Balakrishnan, S. Seshan and R. H. Katz; "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," *ACM Wireless Networks, volume 1, Dec, 1995.*

[138] J. Strauss, D. Katabi, and F. Kaashoek; "A measurement study of available bandwidth estimation tools," *Proceedings of the ACM Internet Measurement Conference (IMC), 2003.*

[139] Pradeep Kyasanur and Nitin H. Vaidya; "Routing and Interface Assignment in Multi-Channel Multi-Interface Wireless Networks," *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), 2005.*

[140] A. Brzezinski, G. Zussman, E. Modiano; "Enabling distributed throughput maximization in wireless mesh networks: a partitioning approach," *Proceedings of the ACM MobiCom, 2006.*

[141] M. Kodialam, T. Nandagopal; "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," *Proceedings of ACM MobiCom, 2006.*

[142] P. Papadimitratos, Z. J. Haas; "Secure Link State Routing for Mobile Ad Hoc Networks," *Proceedings of the IEEE Workshop on Security and Assurance in Ad hoc Networks, 2003.*

[143] K. Claffy, H. Braun, G. Polyzos; "A parameterizable methodology for Internet traffic flow profiling," *IEEE Journal on Selected Areas in Communications (JSAC), 1995.*

[144] Y. J. Lin, and M. C. Chan; "A Scalable monitoring approach based on aggregation and refinement," *IEEE Journal on Selected Areas in Communications (JSAC), 2002.*

Appendix A

Analytical Calculation of TCP's ACK Overhead

Here we derive the equation for the percentage ACK overhead in TCP with and without delayed-ACK option. Let $T_{Headers}$ be the time spent in all the protocol headers on each DATA packet. This includes the PLCP header, 802.11a header, and TCP/IP header, out of which PLCP header is transmitted at 6 Mbps while rest are transmitted at r Mbps (e.g. 54 Mbps).

$$\begin{aligned} T_{Headers} &= T_{PLCP} + T_{802.11a} + T_{IP} + T_{TCP} \\ &= 20 + 36 * 8/r + 20 * 8/r + 20 * 8/r \\ &= 20 + 608/r \end{aligned}$$

The time spent by each maximum sized data packet T_{DATA} is the sum of DIFS period and station's backoff, followed by headers transmission, payload transmission, and link-layer ACK transmission preceded by an SIFS duration.

$$\begin{aligned} T_{DATA} &= DIFS + T_{backoff} + T_{Headers} + T_{L-DATA} + SIFS + T_{PLCP} + T_{L-ACK} \\ &= 34 + (16/2) * 9 + (20 + 608/r) + (1460 * 8/r) + 16 + 20 + (14 * 8/r) \\ &= 162 + 12400/r \end{aligned}$$

T_{ACK}, the time consumed on each TCP ACK is similar except the length of the data payload.

$$\begin{aligned} T_{ACK} &= DIFS + T_{backoff} + T_{Headers} + SIFS + T_{PLCP} + T_{L-ACK} \\ &= 34 + (16/2) * 9 + (20 + 608/r) + 16 + 20 + (14 * 8/r) \\ &= 162 + 720/r \end{aligned}$$

The best case of delayed-ACK is when there is approximately one TCP ACK for every two TCP segments. If O_{ACK} is the percentage overhead of TCP ACKs, then it follows.

$$1/2 * (162 + 720/r) * 100/(162 + 12400/r) <= O_{ACK}$$

In the worst case, delayed-ACK degenerates to per-packet ACK.

$$O_{ACK} <= (162 + 720/r) * 100/(162 + 12400/r)$$