CERIAS Tech Report 2007-107

A multi-layer approach towards high-performance wireless mesh networks

by Das, Saumitra

Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis Acceptance

This is to certify that the thesis prepared

By Saumitra Mohan Das

Entitled

A Multi-Layer Approach towards High-Performance Wireless
Mesh Networks

Complies with University regulations and meets the standards of the Graduate School for originality and quality

For the degree of Doctor of Philosophy

Final examining committee members

Y. C. Hu

S. Fahmy

A. Ghaffor

N. B. Shroff

Approved by Major Professor(s): Y. C. Hu

Approved by Head of Graduate Program: M. J. T. Smith

Date of Graduate Program Head's Approval: 12/6/07

A MULTI-LAYER APPROACH TOWARDS HIGH-PERFORMANCE WIRELESS MESH NETWORKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Saumitra Mohan Das

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2007

Purdue University

West Lafayette, Indiana

UMI Number: 3307519

UMI®

UMI Microform 3307519

Copyright 2008 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Dedicated to my parents Munindra Mohan Das and Rama Das for their support, encouragement, and love. To the memory of my sister Sanghamitra Das.

ACKNOWLEDGMENTS

I would like to thank my advisor Y. Charlie Hu for his guidance and mentorship in my research and for instilling the drive to pursue new ideas in me. By allowing me to be independent while steering work on the right course, he has made me confident in my abilities to succeed in research and handle new challenges in the future. I am also grateful to the members of my Advisory Committee; Sonia Fahmy, Ness Shroff, and Arif Ghafoor, for providing guidance and taking time out to oversee my research.

I would like to express special thanks to Yunnan Wu and Ranveer Chandra for helping me learn new things and become a better researcher during my time at Microsoft Research with them. Working with theoretical and systems people together gave me valuable training for future endeavors. I would also like to thank everyone at the Networking and CCS groups at Microsoft Research for their insights and valuable discussions on my work. I would also like to thank Konstantina Papagiannaki for her mentorship, support and guidance during my time at Intel Research and later. She has been a tremendous source of encouragement during difficult times and I have learnt a lot from her about being a successful researcher.

This work has benefited greatly from interaction with fellow researchers, both in the Distributed Systems and Networking Lab and outside – Chris Gniady, Rongmei Zhang, Zheng Zhang, Sabyasachi Roy, Dimitrios Koutsonikolas, Ali Jafri, Ali Butt, Abhinav Pathak, Yuan Yuan, Suman Banerjee, Y.C. Tay and others. These interactions included both technical discussions and equally important goofing off sessions. I especially enjoyed working closely with Dimitrios those long days and nights debugging stuff in Qualnet and fighting with the wireless testbed. May the testbed be with you and serve you well in the future.

I owe special gratitude to Himabindu Pucha for her emotional and intellectual support throughout the years this dissertation took. Without her, this journey to

the PhD would have been significantly more difficult to complete. She has helped me make many tight deadlines with her great input on organizing papers as well as lightning fast scripting skills for conducting last minute experiments. Thanks Bindu, you are the best.

I am also grateful to other family members and people who have become like family that have played a direct or indirect role in supporting my education whether it was through financial help, moral support or simply providing nice home-cooked meals. So, a lot of thanks is due among others to Diptendra Das, Neeru Das, Dharmeshwar Das and family, Digam, Kulen, Madhavi Deka and family and Naba.

Finally, thanks is due to the people of the Electrical and Computer Engineering Graduate Office, especially Matt Golden, who helped and guided me through all the steps and requirements of the Purdue ECE PhD program. I would also like to thank all those faculty and staff members who agreed to house one of my testbed machines in their offices. Without their help, my testbed and thesis would have been impossible. Thanks also go out to all the students who put up with my weird looking wireless nodes in their classrooms and contained their urge to fiddle with the antennas.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Das, Saumitra M. Ph.D., Purdue University, December, 2007. A Multi-Layer Approach Towards High-Performance Wireless Mesh Networks. Major Professor: Y.Charlie Hu.

Wireless mesh networks are characterized by static mesh routers connected by wireless links to each other providing Internet connectivity via multi-hop access to sparsely deployed gateways. Such networks have important applications ranging from providing Internet access to rural, suburban and under-privileged communities to enterprise wireless offices. However, there are serious challenges in provisioning broadband connectivity using such networks due to the limited throughput caused by interference between mesh routers and coping with the dynamics of the wireless channel.

In this thesis I first deploy a large operational wireless mesh network at Purdue University to understand the behavior of such networks as well as performance problems with currently available state-of-the-art protocols and hardware. This thesis then proposes novel link and network layer protocols that can harness new hardware and technologies such as multiple radios, network coding, directional antennas and cheap storage to significantly improve the performance of wireless mesh networks. Finally, to enable actual deployment of these enhanced high-performance wireless meshes, the thesis proposes a novel method for provisioning service plans for users of such networks. All the research proposals have been implemented and evaluated on the testbed and advance the state-of-the-art in wireless mesh networking thus reducing barriers to their widespread adoption.

1. INTRODUCTION

The advent of the Internet has resulted in enormous changes in our society and has had a direct and indirect positive impact on trade and economy, education, collaboration, human rights and freedom and health. Access to the Internet is seen as a vehicle towards socio-economic change in underdeveloped and underprivileged areas of the world. For example, widespread Internet connectivity can provide access to world class education, promote farming and small business and allow for delivery of telemedicine. While Internet availability has seen tremendous growth in the past decade, a significant part of the world population still does not have access to the Internet. For example, as of September 30th 2007, only 1.2 billion of 6.5 billion people have some form of Internet access [37].

Figure 1.1 shows that this problem is especially acute in developing regions of the world like Africa and Asia where only 4.7% and 12.4% of the population have Internet access. Even in developed regions like the United State, 30% of the population does not have Internet access.

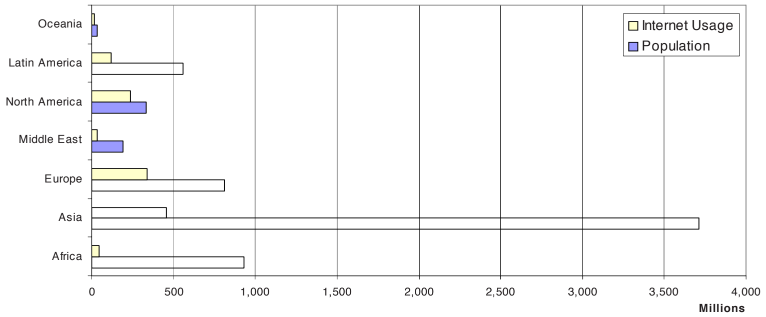


Fig. 1.1. Internet usage across the world. Figure based on data from [37].

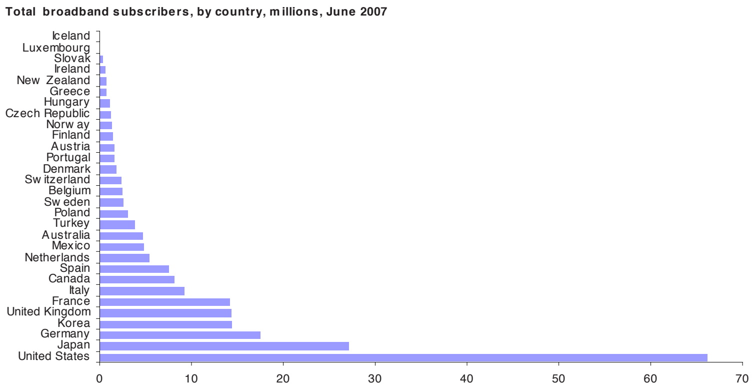


Fig. 1.2. Broadband Internet users. Figure reproduced from [67].

The picture becomes even more dismal when we look at the availability of *broadband* Internet access. Broadband Internet access is particularly critical to enabling new applications such as remote education, telephony and video conferencing, social networking, IPTV and telemedicine which can drive socio-economic development. Figure 1.2 shows the number of broadband users in specific countries. The data shows that even in developed countries like the United States, only 65 million have broadband access out of the total 234 million with some form of Internet access. These numbers are significantly lower for developing regions.

Thus, there is a need to quicken the pace of availability of Internet access and specifically broadband Internet access to reach more and more people around the globe. Unfortunately, there are several hurdles that limit the pace of broadband Internet penetration using existing last-mile technology such as Cable, Digital Subscriber Line (DSL) and Fiber.

- Cost of Deployment

  Cost of deployment is a major factor in broadband penetration. The existing POTS network is very old and upgrading it to support broadband is costly. In addition, many areas specially in the developing world do not yet have any wires reaching them and laying out this infrastructure is costly as well.

- Cost of Access

  Even when wired Internet access is available, underprivileged populations may not have the means to afford it. Reducing the cost of access is not feasible due to either the limited competition in this industry or the costs to defray the deployment. In addition, wired networks also need teams for maintenance and repair which increases access costs.

- Economic Imperative

  In many areas, it may not be economically profitable for service providers to actually deploy broadband connections if they cannot benefit from economies of scale. This hampers broadband penetrations in rural and poorer neighborhoods.

- Accessibility

  Finally, some areas may be just too remote to actually lay down a higher bandwidth pipe and other infrastructure required for broadband access. This is especially true in developing countries with large rural populations and underdeveloped transportation infrastructure.

Thus, there is a need for an alternate means of quickly spreading broadband Internet access at the last-mile. To be successful at reaching those that need it most, the alternate needs to be *cheap to deploy, cheap to maintain* as well as be easy to deploy and have low cost access.

1.1 Wireless Mesh Networks: A Path to Fast and Cheap Broadband Access

Recently a new architecture has been proposed as an attractive option for increasing broadband Internet penetration. These new networks are called wireless mesh networks [5] (WMNs) and are characterized by static mesh routers connected by wireless links to each other and to a few gateway nodes. These few gateway

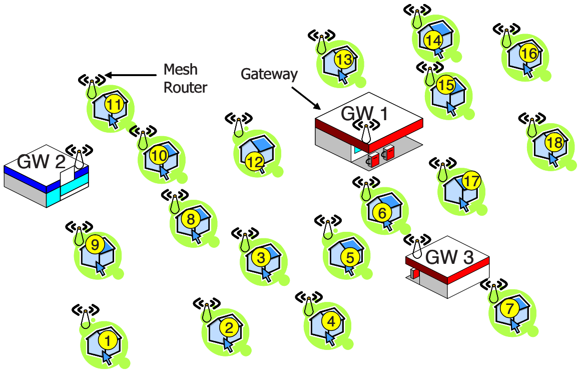


Fig. 1.3. Schematic of a typical mesh network.

nodes have wired connectivity to the Internet. A typical mesh network is depicted in Figure 1.3 in which mesh routers are deployed on top inside or on rooftops of homes.

These mesh routers connect with each other using wireless so that each home can reach one of the three Internet gateways either directly or via multi-hop packet forwarding among the mesh routers. Mesh routers are similar to cheap commodity access point devices that are widely available. In fact, several such access points can be modified in software to perform as mesh routers instead of access points. Typically home users connect to the mesh router on a separate radio interface or via wired Ethernet to the mesh routers.

Wireless mesh networks has several attractive properties that make them useful in deploying broadband Internet access to the next few billion people.

- They are low cost to deploy since no digging, permits and wire layout is required. The devices used (mesh routers) are cheap widely available commodity devices whose costs keep decreasing.

- Such networks are cheap to maintain since they have inherent robustness built in. If a mesh router goes down, the network routing protocols can figure out alternate paths till that router is rebooted or repaired or replaced. They are also not prone to faults such as wires getting cut.

- It is easy to deploy such networks in remote areas since they can cover long distances using better antennas quickly. They can also be easily expanded using user cooperation, i.e. people volunteering to put up routers in their homes to expand connectivity.
- The overall access costs are low since costs of deployment and maintenance are small. In addition, since there are only a few wired Internet subscriptions shared over a wider population, subscriptions can also be lower.

Because of these good properties, the deployment and use of WMNs has recently increased significantly and several cities have planned and/or deployed WMNs ([106, 107, 109, 112–114]). In addition to broadband Internet access, WMNs are also an attractive option for enterprise wireless offices in which the entire office communications (printing, servers, voice communication etc.) takes place over wireless links.

However, despite the attractive properties of WMNs, they face a fundamental challenge to be successful: **performance**. Wireless mesh networks operate over unlicensed spectrum (802.11) and suffer from many performance problems due to external, internal and self interference, multi-hop communication etc. Thus, the main hurdle to their adoption is the need to improve the performance one can achieve using such networks. This is essential to supporting broadband services and applications as well as providing access to a large number of users.

1.2 Performance Challenges in Wireless Mesh Networks

The wireless channel dynamics as well as interference relationships are highly complex and difficult to model or simulate very accurately. Frequently, assumptions made in modeling and simulations can change the expected behavior of the system. Thus, using a testbed to measure and understand performance of a wireless mesh network is a basic tool used in this thesis. A testbed also allows us to study many practical issues in deploying such networks as well as to serve as a vehicle to test and refine theoretical ideas to improve their practical applicability.

1.2.1 Testbed

As part of this thesis, we deployed the first wireless mesh network testbed at Purdue University, MAP [111], shown in Figure 1.4. MAP consists of 32 mesh routers spread out across four academic buildings on the Purdue University campus. The mesh routers are small form factor desktops. Each router has two radios mounted on PCI slots on the motherboard: an Atheros 5212 based 802.11a/b/g wireless card and a Senao 802.11b card based on the Prism2 chipset. Each radio is attached to a 2dBi rubber duck omnidirectional antenna with a low loss pigtail to provide flexibility in antenna placement. Each mesh router runs Mandrake Linux 10.1 and the open-source madwifi and hostap drivers. IP addresses are statically assigned. The testbed deployment environment is not wireless friendly, having floor-to-ceiling office walls instead of cubicles as well as some laboratories with structures that limit the propagation of wireless signals. Apart from structural impediments, interference exists in our deployment from other 802.11b networks (the Purdue Airlink network). We used channel 11 of 802.11b to operate our network since it was the band furthest away from those being already used in the deployment environment. We performed our measurements only during the night to minimize interference from the other 802.11b networks and other sources such as microwaves.

We now present a measurement study of this testbed to highlight performance challenges in wireless mesh networks. Specifically, we study link characteristics, multi-hop data transfer, interference, link dynamics and fairness performance of the testbed using *state-of-the-art* hardware, drivers and routing protocols. This allows us to get a baseline viewpoint of the performance that can be achieved using wireless mesh networks currently and allow us to determine ways towards improving the performance of these networks.

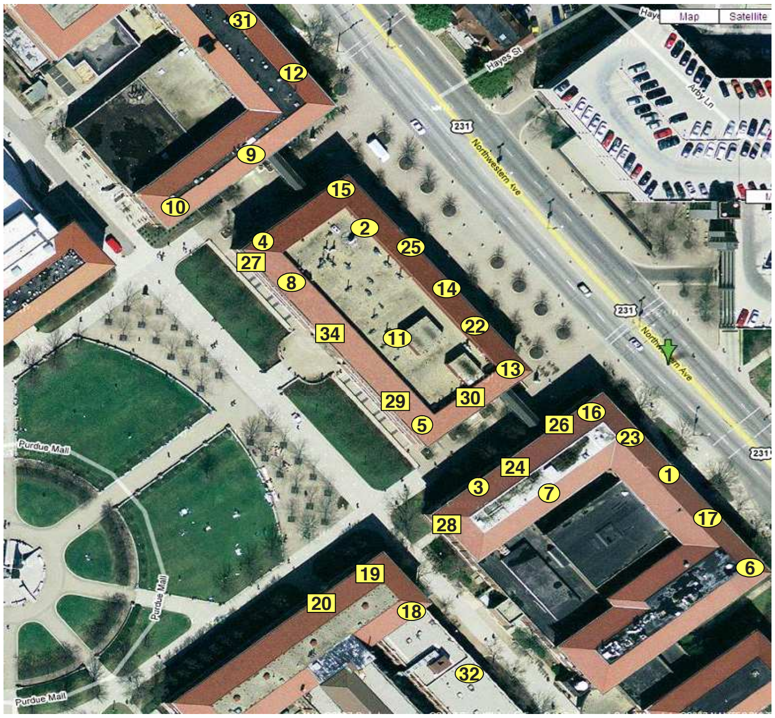


Fig. 1.4. Topology of the MAP network.

1.2.2 Link Characteristics

We first measure the link characteristics of our wireless mesh network testbed. The most important quantities of interest are loss rates and latencies observed in the network. For this set of measurements the OLSR routing protocol [20] was used in MAP, enhanced with ETX routing metric. However, we did not want our measurements to interfere with the control packets of OLSR. Hence we ran OLSR daemon for enough time to ensure that the best paths based on ETX have been set up and stabilized between all pairs of nodes and then we stopped it from sending control messages. Hence, the same routing paths have been used for all the experiments. Our methodology for measuring packet latencies was as follows: Each node in turn sent 100 1470-byte ping packets to each other node. In this way, we obtained the RTTs between any pair of nodes for the paths used by the OLSR protocol. The packet size used is typical for many Internet applications. The packet interarrival time was 0.01

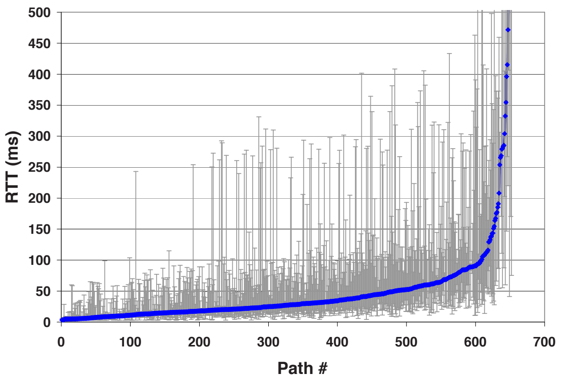


Fig. 1.5. Delay characteristics of the MAP network.

seconds which is equivalent to a sending rate of about 1.1 Mbps. Note that although many measurement papers prefer broadcast packets (e.g., [19]), we preferred to use ping packets. Since ping packets are unicast, our measured latencies better represent latencies experienced by unicast traffic, which we believe will be the common case in WMNs. For example, the RTTs in our experiment take into account retransmissions at the MAC layer, which are not used for broadcast packets.

From Figure 1.5 we observe that RTTs span a large range of values between 3.45 and 1374 msec (we kept the maximum value of the y-axis in the graph at 500 msec, for better visibility). Also, the variations for many paths are very large, and this is mainly true for the maximum values. For example, paths with average RTT below 50 msec have maximum values larger than 250 msec. This shows that network conditions experience steep changes over time and there exist periods of very poor connectivity. This can be a significant problem for applications such as VoIP or video streaming which require constant jitter. On the other hand, minimum values (the true RTT) are not in general much smaller than the average ones, which shows that the quality of the paths is not very unstable. One more encouraging observation is that for the majority of the paths (600 out of 654 paths, about 91% of all paths), the average RTTs are lower than 100 msec which is tolerable.

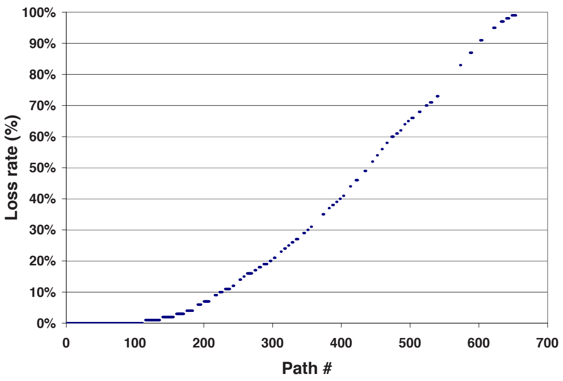


Fig. 1.6. Loss characteristics of the MAP network.

For the loss measurements, we used the same ping messages used in the previous section and we measured the average loss rate over 100 packets for each path. The results are shown in Figure 1.6 which shows the average loss rate for each of the 654 paths. From Figure 1.6 we observe that loss rates cover the whole range between 0 and 100%. The results are not as encouraging as in the case of RTT measurements. More specifically, only 300 out of 654 paths (46%) have loss rates lower than 20%. On the other hand, for 33% of the paths loss rates are higher than 50%.

1.2.3 Multi-Hop Data Transfer

We now measure the performance impact of multi-hop data transfer. This is a unique aspect of wireless mesh networks. Our methodology is as follows: Each node in turn initiated a TCP session to each other node, one at a time to measure all pair-wise paths. We used a 5-sec idle interval between any two sessions to make sure they do not interfere. The `netperf` tool was used. We found that for some of the paths, netperf was unable to initiate a connection. We did not include these paths in the results shown below and the measurements are performed on the 654 working paths.

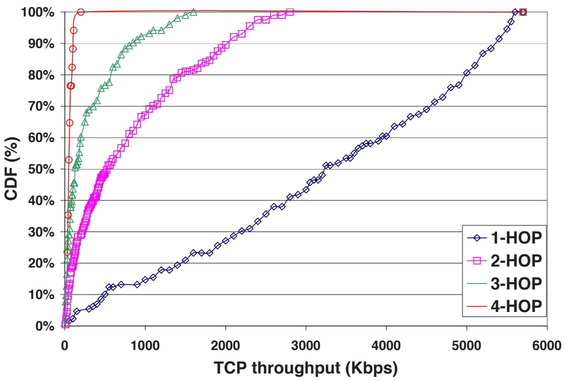


Fig. 1.7. Multi-hop transfers in the MAP network.

The results are shown in Figure 1.7 which depicts a CDF of the TCP throughput obtained over all 1, 2, 3 and 4 hop paths in the network. The results show that there are significant performance differences in multi-hop paths. The TCP throughput falls rapidly as the hop counts increase across a wide range of paths.

1.2.4 Interference

Interference is a major factor in performance obtained using wireless mesh networks. To measure interference, we executed the following experiment. We measured the pairwise interference among all nodes in our testbed, following a methodology similar to [68]. For this experiment, we had each pair of nodes broadcast 1500-byte packets together for 30 seconds. In each 30 second interval, all other nodes except for the two senders measure the throughput from the two senders. In the end of this experiment, for each pair of nodes, one viewed as the sender S and the other the interferer I (and vice versa), we calculated the receiver throughput ratio (RTR) at each one of the remaining nodes (R) as follows: $RTR_R^{S,I} = Throughput_R^{S,I}/Throughput_R^S$, where $RTR_R^{S,I}$ is the receiver throughput ratio at receiver R when S is the sender and I the interferer node, $Throughput_R^S$ is the throughput at R from node S when only S transmits and $Throughput_R^{S,I}$ is the throughput at R from node S when both S and I

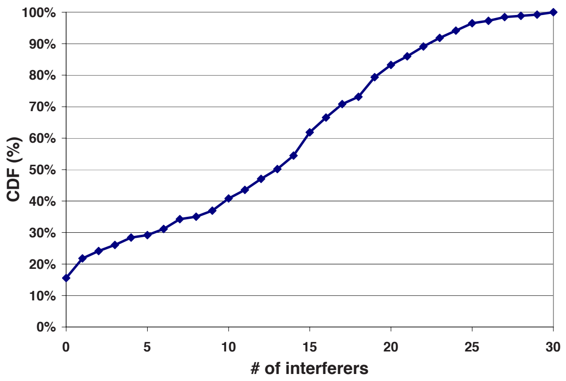


Fig. 1.8. Interference characteristics of the MAP network.

transmit simultaneously. If $RTR_R^{S,I} < 0.9$, we consider that node I is an interferer for link $S \rightarrow R$, else it is not. In this way we found out all nodes that are not interferers for each particular link $S \rightarrow R$ according to the pairwise interferer model.

Figure 1.8 shows the CDF of the number of interferer nodes for the 257 existing links. We observe that the number of interferers varies but, in general, pairwise interference is a widespread phenomenon in our testbed. About 15% of the links have no interferers. On the other hand, 60% of the links have 10 or more interferers and 50% have 13 or more interferers. Finally, the last 17% of the links have more than 20 interferers, and, as we mentioned before, we removed these heavily affected links from the rest of our experiments.

1.2.5 Link Dynamics

Due to the complexity and inaccuracy of propagation and interference models, a link's performance in wireless mesh networks is in practice typically characterized via some periodically measured link metric such as ETT (expected transmission time), ETX (expected transmission count), WCETT (weighted cumulative ETT), RTT (routing trip time), loss rate etc. While a lot of effort has been put into the

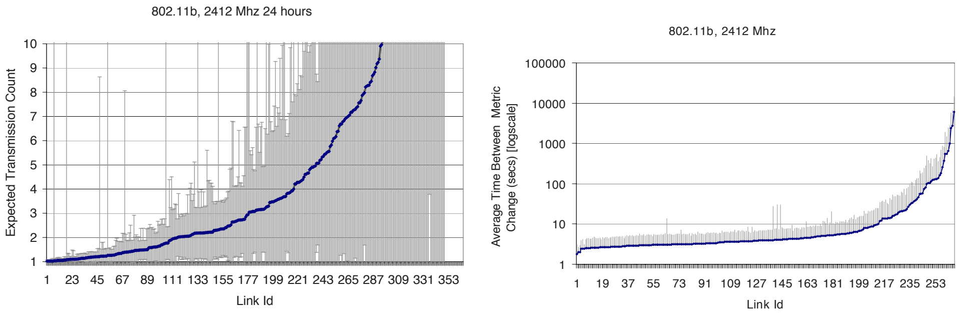


Fig. 1.9. Link dynamics of the MAP network.

design of link metrics ([21, 30, 102]); there has been little focus on the dynamics of these metrics.

To measure the dynamics of link metrics, we computed the average and standard deviation of all link metrics measured for each link in MAP for 24 hrs. The link metric was sampled every 1 second. The results in Figure 1.9 show that the link metric varies significantly in MAP over a 24 hour period. An interesting finding is that while there is correlation between link quality and link metric variability, this may not always be true and a small fraction of good quality links can also exhibit significant link variation. For each link, we also identify time intervals between significant metric changes (i.e. a change of +/- 20%). We then calculate the average and standard deviation of the duration between metric changes for each link. The time scales of link dynamics for are also shown in Figure 1.9. The results show that link metrics change frequently with over 80% of the links exhibiting variation in less than 10 seconds.

1.2.6 Fairness

Finally, the MAC layer currently used in wireless mesh networks does not have a notion of per client fairness or fairness related to a service plan. Such a notion is important if we would like to deploy wireless mesh networks as valid alternatives to current last-mile technologies. The 802.11 MAC layer assigns transmission opportunities based on a notion of single-hop fairness which does not translate into a good model to provision services such as a 128kbps download plan for x dollars.

1.2.7 Lessons Learned

The performance measurements and observations from our wireless testbed provide us with some important lessons and directions to concentrate research on. The first important lesson is that multi-hop paths significantly degrade performance specially when using the widely deployed TCP protocol. This was attributed to frequent loss events and subsequent reduction of window sizes. In addition, the RTT experienced by the TCP flow increased with congestion. Finally, TCP throughput reduces drastically with hop count. Thus, we need approaches that can reduce the hop count traveled by flows or reduce self-interference among wireless hops.

The second important lesson is that significant interference exists in mesh networks, even from nodes that are out of transmission range. More the 70% of nodes had more than 5 interferers. We also found that pair-wise interference measurements are not sufficient and multi-way interference can occur when two or more nodes transmit simultaneously [24]. Thus, schemes that assign channels or perform scheduling need to be careful in deciding which transmissions are non-interfering. In addition, we must spend effort in reducing the number of transmissions to minimize interference or improve the separation between signals.

The third important lesson is that mesh network properties are highly dynamic. Despite the small scale of such networks, latency observed by the applications can range from miniscule to 100ms with longer paths having longer latency. Interestingly, large latency variations can occur across a large number of paths. Multicast and streaming applications will need to be robust to the scale of these latencies and jitter. While around 100 (mostly one-hop) paths in our testbed have almost no loss, the remaining 554 paths have loss rates almost uniformly varying between 0 and 100%. Longer paths have significantly larger loss rates. Note that these are loss rates observed on actual paths selected by the routing protocol that applications will use (not random links) and occurred despite MAC layer retransmissions. This indicates that despite the selection of paths that minimize the expected transmission count, higher-layer protocols are still likely to experience losses and need to have mechanisms to deal with them efficiently.

Finally, the fourth important lesson learned is that we need some protocol modifications or external control to be able to support useful services in wireless mesh networks.

1.3 Thesis Contributions

This thesis proposes a multi-layer approach towards architecting high-performance wireless mesh networks. It takes into account the lessons learned from the testbed performance measurement and looks at advancing the state-of-the-art in wireless mesh networking to improve the performance and usability of such networks. The thesis focuses on two pillars: (1) Improve the **performance** of wireless mesh networks, and (2) Improve the **usability** of wireless mesh networks.

We improve the performance of wireless mesh networks by utilizing lower layer innovations to enhance mesh routers. Several such hardware/software innovations can be added to existing mesh routers. Current mesh routers are simply embedded devices similar to access points.

A schematic of a typical mesh router is depicted in Figure 1.10. It is a simple network layer device since it performs routing in contrast to just a link layer bridging function like access points in WLANs might. As shown through our performance measurements, wireless mesh networks constructed using such devices suffer from several performance issues. We propose to enhance mesh routers with several innovations such as a network coding layer in the software stack, the use of multiple radios on each routers, the use of cheap practical directional antennas and the use of cheap storage devices. Our proposed high-performance router (the $\mathbf{X} - router$) is shown

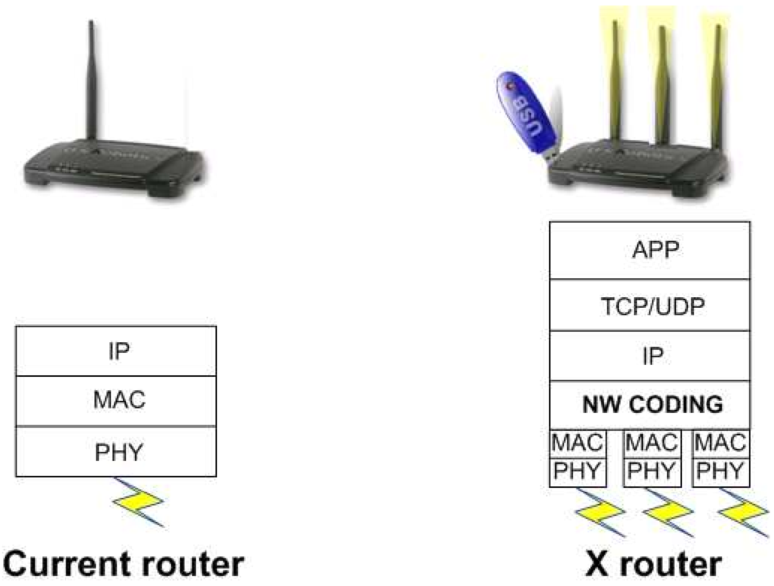


Fig. 1.10. X-router. A proposed high-performance mesh router.

in Figure 1.10. The software stack now contains the transport layer and application layer to leverage cheap storage as well as a network coding layer below the IP layer to mix packets. Finally, the device has multiple physical adapters. Thus, in essence the hardware innovations added to this device consist of multiple radio interfaces, storage devices and directional antennas, and the software innovations added to this device consists of a network coding layer and lightweight operating system to make it an intelligent layer-7 instead of a layer-3 device. Note that these upgrades do not significantly increase cost but can potentially significantly increase performance when coupled with intelligent protocols and algorithms.

1.3.1 Protocols to Leverage Lower Layer Innovations

A key contribution of this thesis is the design of novel protocols and algorithms that can systematically and efficiently take advantage of the lower layer innovations in the high-performance mesh router we propose be used. This is an important problem since the lower layer innovations by themselves cannot provide significant gains without support from such intelligent protocols. For example, network coding may not mix enough packets if flows don't get routed in certain ways. Additionally, taking advantage of multiple radios involves careful examination of channel performance on each radio which may be different in different regions of the network. A bad choice of route could mean no interference reduction. Similarly, adding directional antennas which have imperfect directivity patterns and sidelobes and backlobes requires intelligence to make best use of spatial reuse. Finally, simply adding storage to mesh routers may not improve performance since upper layer protocols need to intelligently exploit these devices. This thesis shows how all these lower layer enhancements in the high-performance ”X-router” can be utilized to their full potential to construct a high performance wireless mesh network.

1.3.2 Service Provisioning

Orthogonal to the performance of the wireless mesh network, an important issue to address is how to use a wireless mesh network to provision services to provide to consumers. In contrast to other multi-hop wireless networks that aim to provide connectivity in the face of constrained energy (sensor networks) or mobility (mobile ad hoc networks), the focus of a WMN (wireless mesh network) is quite different: A WMN aims to be a last-mile technology and thus it must compete with existing last-mile technologies. An important feature of other last-mile technologies (such as cable and DSL) is a “bitrate-for-bucks” service model, i.e., a client is typically promised a bandwidth she pays for, such as 128Kbps for $24.99/month. Thus, a fundamental requirement for WMNs is to provide clients with similar service options A high performance mesh constructed using our proposed ”X-routers” is not suitable by itself for this purpose. In this thesis, we propose the **APOLLO** system that leverages off-the-shelf hardware with software modifications to provision the “bitrate-for-bucks” service. APOLLO seamlessly integrates three synergistic components: (1) A theory-guided service planning and subscription technique to decide how to admit new subscribers. (2) A rate-based admission control to restrict the upload/download traffic from clients in accordance with their service plans. Importantly, APOLLO uses immediate admission control at the access mesh router of the client such that all packets that enter the mesh backbone network are "good" packets that should receive service. This ensures that the traffic in the network is below the capacity of the network. (3) A novel distributed light-weight fair scheduling scheme to deliver the admitted traffic that is robust to unfairness that can arise from interference, variability in wireless channel quality, collisions, etc.

1.3.3 Deployment and Measurement

A key contribution of this thesis is the emphasis on *real* experimentation. The problems tackled in this thesis are motivated through live measurements. The solutions proposed are practical and possible with currently available hardware and software technology. Finally, all solutions towards improving the performance and usability of wireless mesh networks proposed in this thesis are evaluated via a real implementation thus proving the real-world benefits of the approach. This is critical in wireless networks as real deployment environments can be significantly different from those chosen in modeling or simulation.

Bibliographic Note : The material presented in this thesis is based on the following publications: [24], [28], [23], [26], [27], and [98].

1.4 Organization of the Dissertation

This thesis is organized in 6 chapters. Chapters 2, 3, 4 and 5 propose and evaluate advanced protocols that can leverage hardware and software innovations systematically to significantly improve the performance bottleneck that plagues wireless mesh networks. Specifically, Chapter 2 proposes a routing technique to leverage underlying network coding functionality in mesh routers such that coding performance is improved. Chapter 3 proposes a routing technique to leverage multiple radios on mesh routers. Chapter 4 proposes a routing technique to leverage multiple radios coupled with directional antennas on mesh routers while Chapter 5 proposes a routing technique to leverage cheap storage devices on mesh routers. Finally, Chapter 6 proposes a scheme that can make use of such a high-performance wireless mesh network to provision viable user service plans.

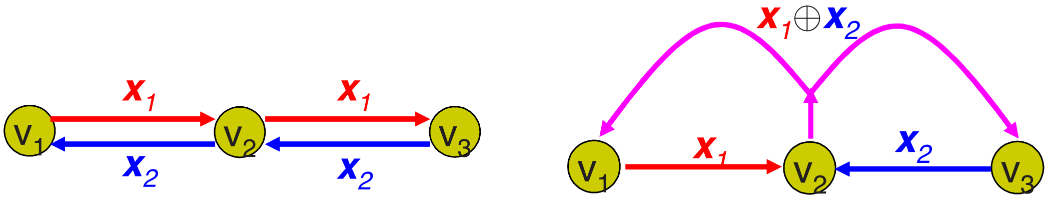
2. PROTOCOL AND ALGORITHM FOR LEVERAGING NETWORK CODING

2.1 Introduction

Network coding refers to a scheme where a node is allowed to generate output data by mixing (i.e., computing certain functions of) its received data. The broadcast property of the wireless medium renders network coding particularly useful. Consider nodes v_1, v_2, v_3 on a line, as illustrated in Figure 2.1. Suppose v_1 wants to send packet $\boldsymbol{x}_1$ to v_3 via v_2 and v_3 wants to send packet $\boldsymbol{x}_2$ to v_1 via v_2. A conventional solution would require 4 transmissions in the air (Figure 2.1(a)); using network coding, this can be done using 3 transmissions (Figure 2.1(b)). The key here is that a single broadcast transmission of $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$ (the bitwise XOR of the two packets) presents $\boldsymbol{x}_2$ to node v_1 who knows $\boldsymbol{x}_1$, and $\boldsymbol{x}_1$ to node v_3 who knows $\boldsymbol{x}_2$. This technique was termed *physical piggybacking* by Wu et al. [97] because the two packets are combined into one, without even increasing the size of the packet. It looks as if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are getting a shared ride in the air.

It is not hard to generalize Figure 2.1 to a chain of nodes. For packet exchanges between two wireless nodes along a line, the consumed channel resource could potentially be halved with physical piggybacking. Wu et al. further showed a simple distributed implementation that can realize such advantages in practice. Specifically, each wireless router can examine its local buffer and mix a left-bound packet with a right-bound packet (here “left” and “right” are in the relative sense). Such a mixture packet can be demixed by the left and right neighbors.

Generalizing [97], Katti et al. [45] recently presented a framework for taking advantage of physical piggybacking to improve the efficiency of unicasting in multi-hop wireless networks. In their approach, each node snoops on the medium and buffers



(a) (b)

Fig. 2.1. (a) The conventional solution requires 4 transmissions to exchange two packets between v_1 and v_3 via a relay node v_2. (b) Using network coding, two packets can be exchanged in 3 transmissions [97].

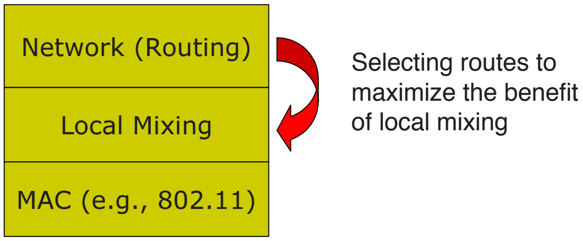


Fig. 2.2. The big picture. The local mixing engine sits between the network layer and the MAC layer and thus presents an enhanced link layer to the network layer. This chapter develops routing solutions that can better take advantage of the local mixing engine.

packets it heard. A node also informs its neighbors which packets it has overheard. This allows nodes to know roughly what packets are available at each neighbor (i.e., "who has what?"). Knowing "who has what" in the neighborhood, a node examines its pending outgoing packets and decides how to form output mixture packets, with the objective of most efficiently utilizing the medium.

These prior studies result in a link layer enhancement scheme in the networking stack. As illustrated in Figure 2.2, the local mixing engine sits above the MAC layer (e.g., 802.11) and below the network layer. Given the routing decisions, the local mixing engine tries to identify opportunities for physical piggybacking. Experimental results in [45] demonstrate the usefulness of local mixing in improving the link layer efficiency. The gain of this technique, however, critically depends on the traffic pattern in the network. This motivates the following question: Can we make intelligent routing decisions that maximize the benefits offered by the local mixing engine?

Our proposed solution builds on top of previous work on routing metric design. State-of-the-art routing protocols for wireless mesh networks have traditionally been based on finding shortest paths under certain cost metrics. The simplest path metric is the hop count along the path. Later on, various link quality metrics have been proposed for static wireless mesh networks. These metrics include for example, the per-hop round-trip time (RTT), the expected transmission count (ETX) [30], and the expected transmission time (ETT) [29].

A natural thought is to modify the link metrics to take into account the effect of the local mixing engine in reducing the transmissions over the air. This, however, is not straightforward. Consider the example setting illustrated in Figure 2.3. There are two long-term flows in the network, $v_3 \to v_2 \to v_1$ and $v_1 \to v_4 \to v_7$. We want to find a good routing path from v_1 to v_9. Due to the existence of the local mixing engine, the route $v_1 \to v_2 \to v_3 \to v_6 \to v_9$ is a good solution because the packets belonging to this new flow can be mixed with the packets belonging to the opposite flow $v_3 \to v_2 \to v_1$, resulting in improved resource efficiency. To encourage using such a route, can link $v_2 \to v_3$ announce a lower cost? There are some issues in doing so, because a packet from v_5 that traverses $v_2 \to v_3$ may not share a ride with a packet from v_3 that traverses $v_2 \to v_1$, although a packet from v_1 that traverses $v_2 \to v_3$ can.

We see from this example that in the presence of the local mixing engine, assessing the channel resource incurred by a packet transmission requires some context information about where the packet arrives from. For example, we can say that given the current traffic condition, the cost for sending a packet from v_2 to v_3 that previously arrives from v_1, is smaller. The key observation here is the need to define link cost based on some context information. More generally, this motivates the concept of a *Markovian metric.*

A Markovian metric introduces context information into the cost modelling. The cost of sending a packet (or a stream of packets) across a link is now allowed to depend

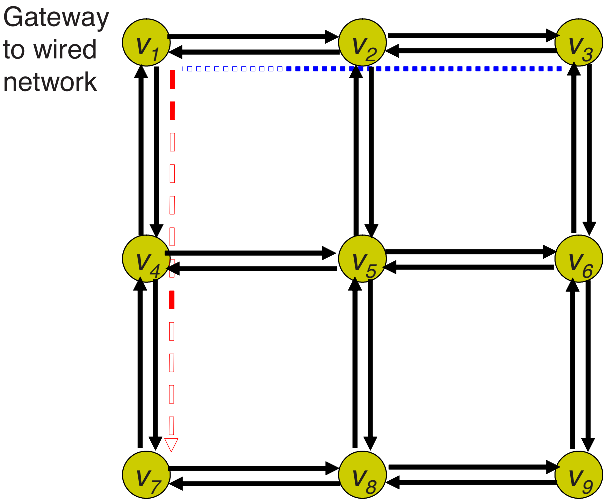


Fig. 2.3. An example mesh networking scenario. There are 9 mesh access points and v_1 is a gateway to the wired network. The connectivity graph is shown in the figure. Assume currently there are two long-term background flows, $v_3 \to v_2 \to v_1$ and $v_1 \to v_4 \to v_7$. Suppose we want to find a good routing path from v_1 to v_9.

on where the packet (or the stream) arrived from. The cost of a path is modelled as the cost of the first hop plus the cost of the second hop conditioned on the first hop, and so on. Due to this decomposition structure, the dynamic programming principle still applies and thus finding the shortest path with a Markovian metric can still be done in polynomial time. In a practical network, support for the Markovian metric can be added easily into an existing routing framework that uses a conventional routing metric.

The concept of the Markovian metric is explained in a general context in Section 2.2. After reviewing local mixing in Section 2.3, in Section 2.4 we examine how to design a specific Markovian metric to maximize the benefit of local mixing. Section 2.5 presents the experimental results.

2.2 Markovian Metric

A conventional routing metric models the cost of a path as the sum of the costs on the individual links. A Markovian metric introduces context information into the cost modelling. The cost of sending a packet (or a stream of packets) across a link is now allowed to depend on where the packet (or the stream) arrived from.

Definition 2.2.1 (Markovian Metric)
Consider a path $\mathcal{P} = v_0 \to v_1 \to \ldots \to v_k$. A Markovian metric *models the cost of a path as the sum of the conditional costs on the links:*

$$\begin{aligned}\mathsf{cost}(\mathcal{P}) \overset{\Delta}{=} & \mathsf{cost}(v_0 \to v_1) + \mathsf{cost}(v_1 \to v_2 | v_0 \to v_1) + \ldots \\ & + \mathsf{cost}(v_{k-1} \to v_k | v_{k-2} \to v_{k-1}).\end{aligned} \tag{2.1}$$

Here $\mathsf{cost}(b \to c|a \to b)$ denotes the cost of sending a packet from b to c, conditioned on that the packet arrived at b via a.

The conventional routing metric can be viewed as a special case of the Markovian metric where all the conditional link costs are equal to their unconditional counterparts. The decomposition relation (2.1) is reminiscent of the decomposition of the joint probability distribution of random variables forming a Markov chain into a product of the conditional probabilities. Thus, a Markovian metric to an unconditional metric is like a Markov chain to a memoryless sequence of random variables.

2.2.1 The Dot Graph Representation

Suppose we are given a set of unconditional link costs W_{uncon} and a set of conditional link costs W_{con}. For ease in notation, we use $w_{i,j}$ to denote an unconditional link cost $\mathsf{cost}(v_i \to v_j)$ and $w_{i,j,k}$ to denote a conditional link cost $\mathsf{cost}(v_j \to v_k|v_i \to v_j)$. We now discuss a graphical representation of these costs, which we call the *dot graph.*[1]

[1] A dot graph is a generalization of the *line graph*, a well known representation in graph theory. Given a graph $G = (V, E)$, the line graph $L(G)$ is a graph whose node set is E and whose edge set

Denote the original graph by G and the resulting dot graph by $\dot{G}$. For the example network in Figure 2.3, the graphical representation of the link costs is illustrated in Figure 2.4. In this example, we assume each unconditional link cost is 1 and there are two conditional costs, $w_{1,2,3} = 0.5$ and $w_{7,4,1} = 0.5$. For instance, here $\mathsf{cost}(v_2 \to v_3 | v_1 \to v_2) < \mathsf{cost}(v_2 \to v_3)$ because a packet from v_2 to v_3 that arrived from v_1 can be mixed with the existing traffic in the flow $v_3 \to v_2 \to v_1$.

First, we introduce a dot for each directed link in the original graph, which "splits" the original link into two halves. Note that there is a one-to-one correspondence between the links in the original graph G and the dots in $\dot{G}$. With slight abuse of notation, we refer to these dots as the names for the links in G; for example, the dot that splits the link from v_1 to v_2 is referred to as $e_{1,2}$. Therefore, $\dot{G}$ has $|V(G)|+|E(G)|$ nodes.

Second, for each conditional link cost $\mathsf{cost}(v_j \to v_k | v_i \to v_j)$ in the given set W_{con}, we draw an edge from the dot e_{ij} to the dot e_{jk}. These edges, together with the edges generated by splitting the original links, constitute the edge set of the dot graph. To distinguish from the edges in the original graph, we call an edge in the dot graph a *wire*. Therefore, $\dot{G}$ has $2|E(G)| + |W_{\text{con}}|$ wires.

Third, we associate a cost label with each wire in $\dot{G}$. The cost of a wire from a physical node $v_i \in V(G)$ to a dot $e_{i,j} \in V(\dot{G})$ is the given unconditional cost of the link, w_{ij}. The cost of a wire from a dot $e_{i,j} \in V(\dot{G})$ to a physical node $v_j \in V(G)$ is 0. The cost of a wire from a dot $e_{i,j} \in V(\dot{G})$ to another dot $e_{j,k} \in V(\dot{G})$ is $w_{i,j,k}$, the given conditional cost of the link.

In general, we allow the coexistence of a conditional cost and unconditional cost for the same link, e.g., $\mathsf{cost}(b \to c | a \to b)$ and $\mathsf{cost}(b \to c)$. We assume the conditional link cost is always less than or equal to its corresponding unconditional link cost. This is without loss of generality because we can always define the unconditional cost on a link as the maximum of the corresponding conditional link costs. The meaning is

comprises the set of all ordered edge pairs (e, e') such that e's end point is equal to e''s start point. The dot graph, however, may contain significantly less edges than the line graph, if there are only a few conditional link costs.

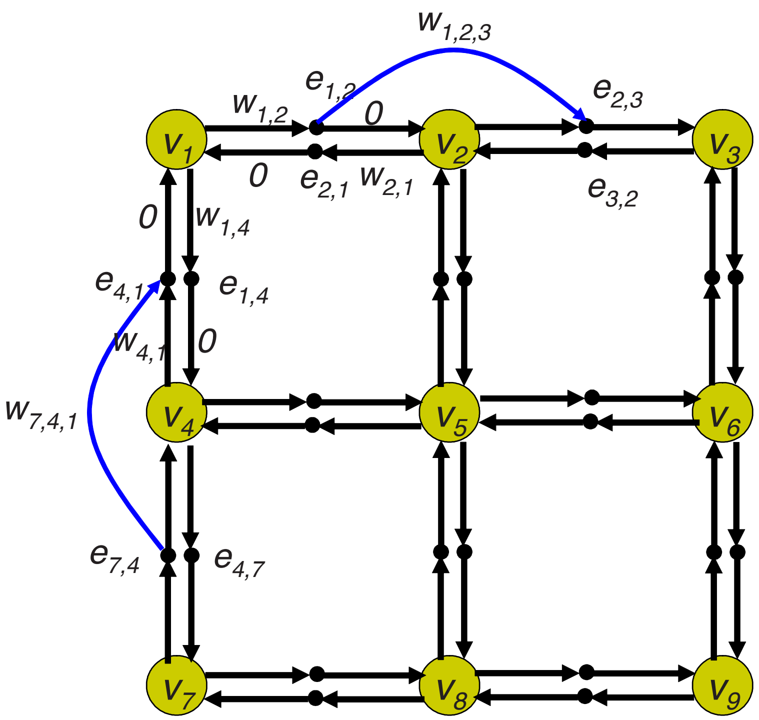


Fig. 2.4. The dot graph representation of a collection of conditional and unconditional link costs, for the example graph in Figure 2.3.

intuitive: The unconditional link cost represents a conservative estimate of the cost incurred; given further context information, the cost may be lower. For example, in Figure 2.4, $w_{1,2,3} = 0.5 < w_{2,3} = 1$; intuitively, there is a "short cut" from $e_{1,2}$ to $e_{2,3}$.

2.2.2 Minimum Cost Routing Using a Markovian Metric

Intuitively, the dot graph models the existence of short cuts at various places in the network. It is easy to see that a path in the dot graph $\dot{G}$ maps into a route in the original network and the cost of the route is just the total cost along the path in $\dot{G}$. For instance, consider a path $\mathcal{P}_1$ from v_1 to v_9 in $\dot{G}$:

$$v_1 \rightarrow e_{1,2} \rightarrow e_{2,3} \rightarrow v_3 \rightarrow e_{3,6} \rightarrow v_6 \rightarrow e_{6,9} \rightarrow v_9.$$

This corresponds to the physical route $v_1 \to v_2 \to v_3 \to v_6 \to v_9$. The cost of the path is: $w_{1,2} + w_{1,2,3} + w_{3,6} + w_{6,9} = 3.5$. In comparison, consider the path $\mathcal{P}_2$:

$$v_1 \to e_{1,4} \to v_4 \to e_{4,7} \to v_7 \to e_{7,8} \to v_8 \to e_{8,9} \to v_9.$$

This corresponds to the physical route $v_1 \to v_4 \to v_7 \to v_8 \to v_9$, which has a cost of 4. Therefore, $\mathcal{P}_1$ is better than $\mathcal{P}_2$.

More generally, to find the minimum cost route between two physical nodes, we just need to apply a shortest path algorithm over the dot graph. For example, with Dijkstra's algorithm, the complexity is $O(|V(\dot{G})|^2)$. Note that we can remove the unnecessary dots in $\dot{G}$; specifically, we can introduce a dot $e_{i,j}$ only if there is a need to express a related conditional link cost, i.e., when W_{con} includes a cost $w_{i,j,*}$ or $w_{*,i,j}$. (Here we use the symbol $*$ as a wildcard.) By doing so, the number of vertices in the dot graph can be reduced to $O(|V(G)| + \min\{|E(G)|, 2|W_{\text{con}}|\})$.

Proposition 2.2.1 (Min-Cost Routing w/ Markovian Metric) *Given a set of unconditional link costs W_{uncon} and a set of conditional link costs W_{con}, the minimum cost routing from a source node s to a sink node t can be found by running a shortest path algorithm over the dot graph. This can be done in complexity $O\left((|V(G)| + \min\{|E(G)|, 2|W_{con}|\})^2\right)$.*

2.2.3 Adaptive Routing in a Practical Network

In a practical network, routing decisions are made to reflect the changes in the topology and sometimes the traffic as well. The dot graph representation makes it particularly easy to see how to modify the existing routing protocols for a Markovian metric system. Essentially, a physical node v_i needs to play several characters in a distributed routing algorithm, including those of its neighboring dots who do not physically exist. In particular, we could divide the computation responsibility as follows. Let each physical node v_i be responsible for the outgoing wires of v_i and its incoming dots $e_{*,i}$. For example, in Figure 2.4, physical node v_2 implements the computation involving wires $e_{*,2} \to v_2$, $v_2 \to e_{2,*}$, $e_{1,2} \to e_{2,3}$.

Routing protocols can be classified as either proactive or reactive. Proactive protocols attempt to maintain up-to-date routes within the network, so that a route is readily available when a packet needs to be forwarded. Reactive protocols, on the other hand, do not maintain up-to-date topological information about the network. When a source needs to find a route to a destination node, the source initiates a route discovery procedure, typically done by flooding. In the following we examine how to support a Markovian metric in representative routing algorithms, starting with the proactive protocols.

Link State Routing

Example protocols in this category include OLSR [19] and LQSR [30]. In link state routing, each router measures the cost to each of its neighbors, constructs a packet including these measurements, sends it to all other routers, and computes the shortest paths locally. In essence, the complete topology and the link costs are experimentally measured and distributed to each router. Then each router can locally run a shortest path algorithm to decide the routes.

To support a Markovian metric, minimal changes are needed in a link state routing system. Each router can just measure the unconditional and conditional costs for the links/wires it is responsible for and broadcast the measurements to all other routers. Take node v_2 in Figure 2.4 as an example. Here v_2 needs to measure the unconditional costs to each neighbor, as well as the conditional costs of the form $\mathsf{cost}(e_{i,2} \rightarrow e_{2,j})$.

Distance Vector Routing

An example protocol in this category is DSDV [72]. In distance vector routing (also known as the Bellman-Ford algorithm), each router maintains a table (i.e., a vector) giving the best known cost to reach each destination and which interface to use to get there. There tables are updated by exchanging information with the neighbors.

Let us start with a first-cut solution. Once every T milliseconds each router sends each neighbor a list of its estimated minimum cost to reach each destination. Since each physical node v_i is also playing the roles of its incoming dots, a first-cut implementation will aggregate the tables from v_i and its incoming dots. For example, consider node v_2 in Figure 2.4. It is also responsible for playing the roles of $e_{1,2}$, $e_{3,2}$, $e_{5,2}$. Thus the aggregated table will include one minimum cost from $e_{i,2}$ to v_j, for $i = 1, 3, 5$ and all other destinations. Denote $\mathsf{cost}(e_{i,2} \rightsquigarrow v_j)$ the estimated minimum cost to reach destination v_j. Note that in this case,

$$\mathsf{cost}(e_{3,2} \rightsquigarrow v_j) = \mathsf{cost}(e_{5,2} \rightsquigarrow v_j) = \mathsf{cost}(v_2 \rightsquigarrow v_j).$$

Similar scenarios may happen whenever the conditional cost cannot lead to a lower route to the given v_j. In these scenarios, sending both $\mathsf{cost}(e_{3,2} \rightsquigarrow v_j)$ and $\mathsf{cost}(e_{5,2} \rightsquigarrow v_j)$ is wasteful. To remove such redundancy, we include $\mathsf{cost}(v_2 \rightsquigarrow v_j)$, and $\mathsf{cost}(e_{*,2} \rightsquigarrow v_j)$ only if it is lower than $\mathsf{cost}(v_2 \rightsquigarrow v_j)$. This results in an improved implementation.

Source-Routed On-Demand Route Discovery

Example protocols in this category include DSR [40] and DSR with ETX [21]. Here a source node that wishes to discover a route to a destination broadcasts a route request packet. This route request contains the address of the destination, the source node's address, and a unique identifier. When a node forwards a route request, it appends not only its own address, but also information about the related link costs. Specifically, suppose the route discovery packet has traversed a path $v_0 \to v_1 \to \ldots \to v_k$ to v_k. Then v_k sends a route request that contains:

$$\begin{aligned} &\mathsf{cost}(v_0 \to v_1),\ \mathsf{cost}(v_1 \to v_2 | v_0 \to v_1),\ \ldots, \\ &\qquad\qquad \mathsf{cost}(v_{k-1} \to v_k | v_{k-2} \to v_{k-1}), \end{aligned} \tag{2.2}$$

and also $\mathsf{cost}(v_k \to v_j | v_{k-1} \to v_k)$ for neighbor v_j (because v_j may not know how to compute this). If the physical medium is broadcast medium, then v_k can send a single broadcast packet that contains (2.2) and $\mathsf{cost}(v_k \to v_j | v_{k-1} \to v_k)$ for all neighbors v_j.

When a node v_k receives a request it has already forwarded, it forwards it again only if the accumulated cost to a neighbor v_j is better than the best which it has already forwarded with the request ID. The link metrics are included in the route replies sent back to the source.

Hop-by-Hop On-Demand Route Discovery

An example protocol in this category is AODV [71]. Similar to the source-routed on-demand discovery, here the route request initiated by the source is flooded through the network. However, each route request no longer contains the entire route. Instead, each route request received by v contains only the cumulative cost from s to v. Each node v maintains for each incoming neighbor u the minimum accumulated cost from s via u to v, denoted by $\mathsf{cost}(s \rightsquigarrow u \to v)$. When a node v receives a request, it forwards the request to a neighbor v' only if the total cost to v' is reduced. The routing table entries to the destination will be created after the destination sends back a route reply to the source, following the sequence of best previous hops.

2.3 Local Mixing: A Review

The gist of the packet exchange example in Figure 2.1 is as follows: At certain moment, v_1 has $\boldsymbol{x}_1$; v_2 has $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$; v_3 has $\boldsymbol{x}_2$. Thus a mixture packet $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$ can be demixed into $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ respectively at v_3 and v_1. In the following we use the name *source packet* to refer to a packet such as $\boldsymbol{x}_1$ which was originally generated by a source node, and the name *mixture packet* to refer to a packet such as $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$.

More generally, we may have a situation illustrated in Figure 2.5. A wireless router knows the source packets each neighbor has (i.e., “who has what”). It also knows “who wants what” because these are the packets in its output queue that it is supposed to forward to the neighbors. Then it can decide locally how to optimize the formation of mixture packets. A heuristical approach for generating the mixture packets is used in [45], which takes the packet at the head of the output packet queue, and steps

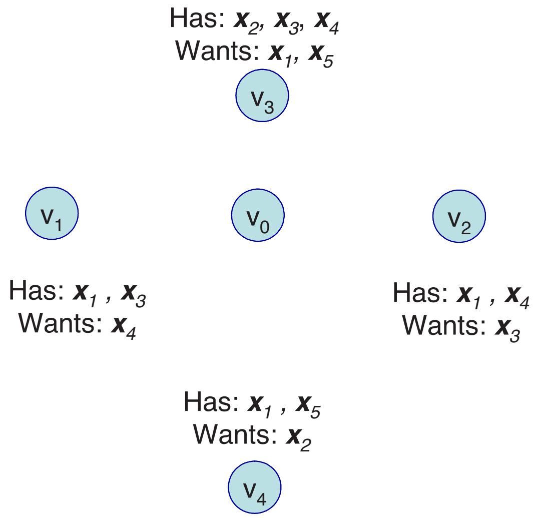


Fig. 2.5. The local mixing problem is about optimizing the formation of mixture packets at a local wireless router, knowing "who has what" and "who wants what" in a neighborhood.

through the packet queue to greedily add packets to the mixture, while ensuring the neighbors can successfully demix. For example, in Figure 2.5, there are five packets in the output queue, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5$; assume a lower indexed packet is an earlier packet. Then the greedy procedure will use three transmissions: $\boldsymbol{x}_1 \oplus \boldsymbol{x}_2$, $\boldsymbol{x}_3 \oplus \boldsymbol{x}_4$, $\boldsymbol{x}_5$. However, there is a better solution: $\boldsymbol{x}_1 \oplus \boldsymbol{x}_3 \oplus \boldsymbol{x}_4$, $\boldsymbol{x}_2 \oplus \boldsymbol{x}_5$. A mathematical abstraction of the optimized formation of the mixture packets – the *local mixing problem* – is studied by Wu et al. [99] from an information theoretic point of view. Under the assumption that each neighbor discards the received packets that are polluted by sources it does not have or want, the optimal mixing is characterized.

Why would a node have packets meant for others? In Figure 2.1, node v_1 has packet $\boldsymbol{x}_1$ because it is the previous hop of $\boldsymbol{x}_1$. More generally, due to the broadcast nature of the wireless medium, neighboring nodes may overhear packets. For example, in Figure 2.5, packet $\boldsymbol{x}_1$ may follow a path $\ldots v_4 \rightarrow v_0 \rightarrow v_3 \ldots$; v_1 and v_2 may have overheard $\boldsymbol{x}_1$ when v_4 sent it to v_0.

How does a node get to know "who has what"? First, note that a node can obtain some partial information about its neighbors' data availability in a passive fashion. For example, node v_2 may infer that node v_1 holds packet $\boldsymbol{x}_1$ if v_1 recently received packet $\boldsymbol{x}_1$ or a mixture packet involving $\boldsymbol{x}_1$ from v_1, or if v_2 recently heard v_1 acknowledging the receipt of packet $\boldsymbol{x}_1$. This suffices for packet exchanges such as Figure 2.1.

Passive inference does not incur any additional overhead. However, using passive inference alone, a node may only obtain a limited view of the neighbors' data availability. Katti et al. [45] extended this by proposing two techniques to obtain more information about local data availability: (i) Let each node explicitly announce the packets it currently has to its neighbors; (ii) let a node guess whether a neighbor has overheard a packet using information about the channel reception. In the former, each node can periodically compose *reception reports* to announce the packets it has overheard. The reception reports may also be piggybacked with ordinary packets. To implement guessing, nodes conduct measurement about the packet success probabilities to its neighbors and exchange the measurement results in the neighborhood. Such measurement and report functionality may already be needed by a routing protocol based on the expected transmission count (ETX) [21]. The guessing technique of [45] can be explained via Figure 2.5. Suppose v_4 sends a source packet $\boldsymbol{x}_1$ to v_0 without mixing; suppose v_0 knows that v_1 can receive a packet from v_4 with probability 0.8. When v_0 received $\boldsymbol{x}_1$ sent by v_4, v_0 can infer that v_1 has overheard the packet with probability 0.8. Guessing may result in a more up-to-date knowledge about "who has what"; however, if the guess is wrong, the neighbor may fail to demix a packet intended for it.

A mixture packet may be intended for more than one receiver. Due to the limited collision avoidance mechanism for broadcast in 802.11, the mixture packet is sent as a unicast packet addressed to one of the receivers [45]. A consequence of this is that the sender cannot be sure whether the other intended receivers received the packet reliably. We call such an issue the "missing ACKs" problem. The missing ACKs problem can be addressed by explicitly generating ACKs in addition to the ACK in 802.11 MAC [45]. Nodes can keep track of the packets that were sent but have not yet been acknowledged and retransmit packets after time-out. Next, we briefly review the key data structures and operations in implementation.

Each packet has a variable length header that includes: (i) the IDs of the source packets being mixed and their respective receivers, (ii) some piggybacked ACKs, (iii) some piggybacked reception reports. If no data packets were sent after a certain amount of time, then a dedicated control packet containing ACKs and reception reports is broadcast.

Each node maintains three separate buffers, `OverheardBuffer`, `ReceivedBuffer`, `SentBuffer`, holding respectively the source packets that the node overheard, received, or sent. Upon receiving a packet, the packets in these three buffers are used for demixing. Reception reports describe new content in the `OverheardBuffer`. ACKs describe new content in the `ReceivedBuffer`.

Each node maintains a `WhoHasWhatTable` whose entries are of the form "node v_i has source packet $\boldsymbol{x}_j$ with probability p". Upon receiving a packet, the `WhoHasWhatTable` is updated according to the local mixing header. If the received packet is a source packet, guessing is also performed based on the measured channel reception probabilities.

After a packet is sent, the ingredient source packets are moved from the output queue into the `SentBuffer`. In addition, timer events are inserted so that the sent packets will be moved back to the output queue for retransmission if the ACKs does not arrive after a certain time threshold.

2.4 Markovian Metric for Local Mixing

In this section we examine how to use a Markovian metric to maximize the benefit of local mixing. The central issue here is to properly define the link costs and compute them. Let us begin with the unconditional link metrics. A popular link quality metric in the literature is the expected transmission count (ETX) [21]. This metric estimates the number of transmissions, including retransmissions, needed to send a unicast packet across a link. It is obtained by measuring the loss probabilities of broadcast packets between pairs of neighboring nodes.

The ETX metric can be viewed as a characterization of the amount of resource consumed by a packet transmission. With the local mixing engine, several packets may share a ride in the air. Naturally, the passengers can share the airfare. In effect, each participating source packet is getting a discount. Such discount, however, cannot be accurately modelled by an unconditional metric such as ETX, because the applicability of the discount depends on the previous hop of the packet. We propose a conditional link metric called the *expected resource consumption* (ERC), which models the cost saving due to local mixing. Consider a packet sent in the air. If it is a mixture of k source packets, then each ingredient source packet is charged $\frac{1}{k}$ the resource consumed by the packet transmission. The resource consumed by the transmission could be measured in terms of, e.g., air time, or consumed energy.

2.4.1 Computation of Expected Resource Consumption (ERC)

We now explain how to compute the ERC. Each node maintains a `WireInfoTable`. Each row of the table contains the measured statistics about a wire, say $e_{i,j} \rightarrow e_{j,k}$, which crosses the current node v_j. The packets forwarded by the current node can be classified into categories associated with the wires. For each wire category, we collect the total number of packets sent and the total resource consumed in a sliding time window. The total resource consumption is obtained by adding the resource consumption for each sent packet. A simple charging model is used in our current implementation. For example, if a source packet across wire $e_{i,j} \rightarrow e_{j,k}$ is sent in a mixture of 3 packets, we set the resource consumption of this source packet as 1/3 of the ETX of link $e_{j,k}$.[2]

[2] We could also use ETT [29] in lieu of ETX. The ETT metric is equal to the ETX metric divided by the raw link rate. When the radios operate at the same physical link rate, the two metrics are essentially equivalent.

To implement the sliding window computation efficiently, we quantize the time axis into discrete slots of equal length. We use a sliding window of N slots. For each wire, we maintain a circular buffer of N bins; at any time, one of the N bins is active. At the start of each slot, we shift the active bin pointer once and clear the statistics in the new active bin. Each time a packet is transmitted in the air, we update the statistics in the current active bin accordingly.

To evaluate the conditional link metric for a certain wire $e_{i,j} \rightarrow e_{j,k}$, we first obtain the ERC for each slot, say n, as:

$$\mathsf{erc}_n := \frac{\text{Resource consumed by pkts sent in slot } n}{\#\text{ of packets sent in slot } n}. \tag{2.3}$$

Then we compute the ERC for the wire as the weighted average of the ERCs for the slots:

$$\mathsf{ERC} := \sum_{n=0}^{N-1} \alpha_n \mathsf{erc}_n; \quad \alpha_n = \alpha^{N-1-n} \left(\frac{1-\alpha}{1-\alpha^N} \right). \tag{2.4}$$

Here the parameter α is the forgetting factor for old observations. Old observations receive lower weights.

What if few or no packets were sent during a certain slot across the wire? We propose to conduct experiments using probing packets. The experiments are done via simulation, without disturbing the existing traffic. Specifically, we maintain a virtual output queue in addition to the output queue. Whenever we insert an actual packet into the output queue, we insert it into the virtual output queue as well. Whenever the MAC layer asks for an output packet, we also invoke the mixing algorithm on the virtual output queue to generate a mixture packet. (In fact, for the virtual output queue, we just need to decide on how to mix; no actual mixing is needed.) The result of the mixing decision is only used to update the statistics for the wires being tested.

2.4.2 Interpretation of the ERC Metric

The ERC link metric estimates the local resource consumption while considering the effects of local mixing. The resulting Markovian path metric thus estimates the total resource consumed by the route. Resource efficiency is an important performance metric for resource-constrained wireless networks, e.g., sensor networks. It is also well aligned with the overall system throughput in interference limited wireless networks, because resource saved in one flow may be used to improve the throughput for other flows.

The ERC metric emphasizes the maximization of network-wide utility more than the maximization of the individual flow utility (e.g., latency). By using a weighted combination of two Markovian metrics, one capturing the network-wide utility and the other the individual flow utility, we can adjust the balance between the two objectives. For example, occasionally there may be a longer route with a lower resource consumption than a shorter route; in this case, we may define a Markovian metric as the weighted combination of the ERC metric and the ETT (expected transmission time) metric:

$$\mathsf{cost}(e_{jk}|e_{ij}) \triangleq \beta\mathsf{ERC}(e_{jk}|e_{ij}) + (1-\beta)\mathsf{ETT}(e_{jk}), \tag{2.5}$$

where the ETT metric is computed by dividing the ETX metric by the raw link data rate (see [29]). It is up to the applications or certain network rule-makers to decide how to balance these two considerations.

2.4.3 Route Stabilization via Randomized Route Holding

In order to model the resource reduction due to local mixing, the ERC takes the traffic load into account. Could this cause oscillation in the routing decisions? We provide some intuitive reasoning below. Recall that the discounts offered by the local mixing engine exist only when the flows cross in certain ways. Stated alternatively, the advertised discounts have restrictions and hence only a few qualifying flows may find them attractive. Since the discounts benefit all the flows whose packets are being mixed, there is incentive for flows to route in a certain cooperative manner that are mutually beneficial. Presumably, if the flows try such a mutually beneficial arrangement for some time, they will confirm the discounts and tend to stay in the arrangement. Such an arrangement is analogous to the Nash equilibrium in game theory, where no player wants to deviate from its current strategy given all other players' strategies. However, a complication is that there can be more than one equilibrium. We want the flows to make dynamic decisions that eventually settle down to one equilibrium. To facilitate this, we propose the following strategy. To prevent potential route oscillations, we require each flow to stay for at least T_{hold} duration after each route change, where T_{hold} is a random variable. The randomization of the mandatory route holding time T_{hold} is used to avoid flows from changing routes at the same time. In addition, after the mandatory route holding duration, the node switches to a new route only if the new route offers a noticeably smaller total cost.

2.5 Evaluation of Markovian Metric Routing

We implemented the local mixing engine, a link-state source routing protocol, the support for Markovian metric routing, and the support for the ERC link metric in Qualnet 3.9.5, a widely used event-driven simulator for wireless networks. The resulting integrated system will be referred to as Markovian Metric Source Routing (MMSR) in the following. We implemented local mixing for MMSR as a shim layer between the routing and MAC (802.11) layers. The routing layer exposes the packet delivery probabilities to the local mixing layer (for guessing "who has what"). The local mixing layer exposes the ERC statistics to the routing layer.

The basic routing framework can be viewed as a simplified version of LQSR [30], a source-routed link-state protocol that supports link quality (unconditional) metrics. Similar to LQSR, every 2 seconds, each node sends a message that carries information about the links from this node; this message floods throughout the network. Such a message, called the `WireInfo` message, describes the unconditional and conditional link costs. Here ETX is used as the unconditional link cost, and ERC is used as the conditional link cost. To compute ERC, we use $N = 10$ slots, each of length 0.5s. If there are 25 or more packets crossing a wire, then the ERC of the slot is computed as the average according to (2.3). Otherwise, a simplified rule (instead of maintaining the virtual output queue) is applied in our current implementation. Specifically, for a wire $e_{ij} \to e_{jk}$, if there are less than 25 packets crossing, then we examine the number of unmixed packets y in the reverse wire $e_{kj} \to e_{ji}$. If $y \geq 25$, then we set the ERC as 0.5 of the ETX (50% discount); otherwise, we set the ERC as the ETX (no discount). The ERCs of the slots are combined with exponentially delaying weights, using $\alpha = 0.8$ in (2.4).

A conditional link cost, say $\texttt{cost}(e_{jk}|e_{ij})$, is included in the message only if it is at least 5% less than the ETX for e_{jk}. Each node maintains a `LinkCache` that stores its current picture of the dot graph. The `LinkCache` is updated upon receiving each `WireInfo` message. Since the conditional link costs are load-dependent and may not be periodically reported, stale conditional link cost entries are removed after time-out.

The route stabilization techniques of Section 2.4.3 are used. The mandatory holding time T_{hold} is drawn uniformly from $[1, 3]$ (sec). When a source packet is just generated, if the current mandatory holding time has elapsed, then the node looks for the optimal route in the dot graph. If the cost of the new route is at least 5% lower than that of the old route, then the new route is adopted. Otherwise, the old route is still used.

The simulations use the 802.11a MAC and a realistic signal propagation model. All radios operate at a nominal physical layer rate of 54Mbps. We use CBR flows in the evaluation.

2.5.1 Impact on Local Mixing: MMSR vs. LQSR+LM

We first demonstrate the performance of local mixing in MMSR compared to LQSR+LM, which uses the unconditional ETX metric for routing. This evaluation is performed in a 9-node grid network topology.

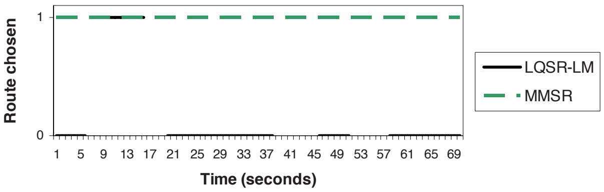


Fig. 2.6. Scenario 1: Mixing performance. State 1 denotes an optimal mixing route whereas 0 denotes a suboptimal mixing route.

Table 2.1
Gain from MMSR compared to LQSR+LM.

Scenario	Mixed (MMSR)	Mixed (LQSR+LM)
S1	20,366	1,593
S2	39,576	24,197

Mixing performance evaluation

Consider the network shown in Figure 2.3 with an existing flow $v_3 \rightsquigarrow v_1$. After 3 seconds, v_1 initiates a flow to v_9. There are many possible routes that this flow can take, but only one ($v_1 - v_2 - v_3 - v_6 - v_9$) is optimal in terms of the resource consumption. Figure 2.6 shows how often this optimal route is chosen by LQSR+LM and MMSR, respectively. As the results show, MMSR causes the flow $v_1 \rightsquigarrow v_9$ to choose the mutually beneficial route $v_1 - v_2 - v_3 - v_6 - v_9$ and this route is chosen 100% of the time, resulting in maximized mixing. Intuitively, the existing flow $v_3 \rightsquigarrow v_1$ creates a discount in terms of the conditional ERC metric in the opposite direction, which attracts v_1 to choose route $v_1 - v_2 - v_3 - v_6 - v_9$. Once the flows start in both directions, they stay together and mix because both see discounts. As shown in Table 2.5.1, MMSR increases the number of mixed packets in this scenario by 12× in comparison to LQSR+LM.

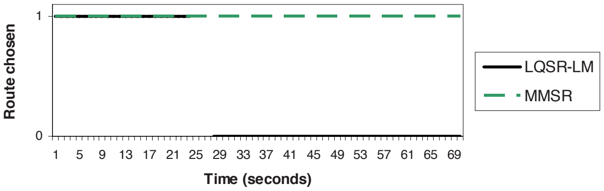


Fig. 2.7. Scenario 2: Mixing performance. State 1 denotes an optimal mixing route whereas 0 denotes a suboptimal mixing route.

Next, we examine whether mixing can take place effectively over longer routes. We thus now consider an existing flow $v_9 \rightsquigarrow v_1$ and examine the choice made by a new flow $v_1 \rightsquigarrow v_9$ originating 3 seconds later. The results are shown in Figure 2.7. Notice that LQSR+LM only chooses the optimal route for less than half of the time, whereas MMSR quickly locks on to the opposite route due to the conditional discounts. In this case, mixing is possible at 3 separate nodes and thus the number of mixed packets increase.

Note that MMSR is crucial when specific small opportunities exist for mixing (e.g. at one node in scenario S1). In this case it is unlikely that LQSR+LM will be able to exploit such opportunities. In scenario S2, mixing is possible at many hops and by pure random choice, LQSR+LM can also locate a non-trivial number of mixing opportunities. However, MMSR still provide a 63% increase in mixed packets over LQSR+LM.

Stabilizing routes

As mentioned in Section 2.4.3, an important but subtle requirement for a Markovian metric is to avoid potential route oscillations. Occasionally, if two flows that can potentially mix (such as $v_9 \rightsquigarrow v_1$ and $v_1 \rightsquigarrow v_9$) start right at the same time and choose different routes, they could be attracted to each other repeatedly and potentially oscillate. As described in Section 2.4.3, we deal with this by holding each route for a random amount of time. Figure 2.8 shows the route evolution when both flows

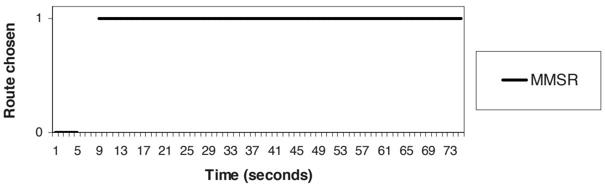


Fig. 2.8. MMSR randomization provides robustness to oscillations. State 1 denotes an optimal mixing route whereas 0 denotes a suboptimal mixing route.

start exactly together. Due to the randomized route holding strategy, the oscillation is resolved quickly (in 7 seconds in this case), allowing the two flows to mix effectively. Note that such pathological cases are unlikely to occur frequently; nonetheless, the randomized route holding strategy ensures correct operation even in such pathological cases.

Summary

In summary, we find that MMSR reliably chooses the optimal mixing route and the ERC discounting system, together with the randomized route holding strategy, facilitate flows to settle down into a mutually beneficial equilibrium in a distributed and dynamic manner.

While these results serve to illustrate how MMSR improves the mixing opportunities and thus reduces the channel usage, it is important to see what impact this can have on the overall performance of the network, i.e., what is the end benefit to the clients of the wireless network? We study this issue in the next section using the 9-node grid topology and a wireless office topology.

2.5.2 Impact on Overall Performance

We next study how MMSR compares to LQSR+LM as well as basic LQSR in terms of the overall performance.

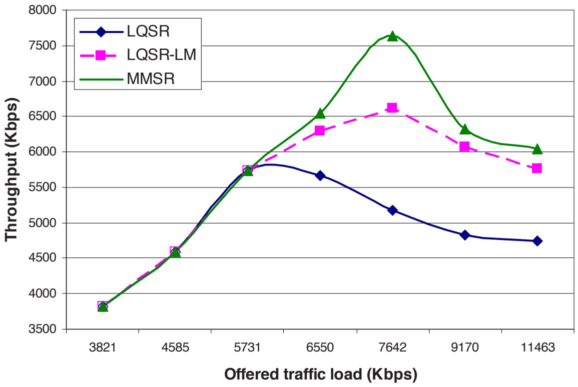


Fig. 2.9. Throughput comparison of MMSR, LQSR+LM and LQSR.

Grid Network Scenario

We continue with the 9-node grid network scenario and evaluate the performance with three flows: (1)$v_9 \rightsquigarrow v_1$, (2) $v_1 \rightsquigarrow v_9$, and (3) $v_3 \rightsquigarrow v_1$. Each flow begins randomly between 50–60 seconds into the simulation. We evaluate the performance of LQSR, LQSR+LM and MMSR for this scenario for different input loads. The results are depicted in Figure 2.9. It is observed that LQSR cannot sustain the throughput imposed by the input flows to the network as the load increases. MMSR provides significant throughput gains compared to LQSR (up to 47%) and LQSR+LM (up to 15%). This is because not only does MMSR allow subsequent flows to mix with existing flows, it explicitly tries to maximize mixing. In contrast, without the Markovian metric routing, flows mix essentially by chance. In the example, the flow 1-2-3-6-9 is mixed with 9-6-3-2-1 with MMSR, due to the mutually beneficial discounts enjoyed by both flows.

Figure 2.10 gives the amount of *resource* saved by using MMSR. MMSR consistently provides reduction of packet transmissions of over 10,000 packets across a wide variety of traffic demands. Indeed, as discussed in Section 2.4.2, MMSR functions by directly trying to minimize the resource usage. Another observation from Figure 2.10 is that the saved transmissions reduce as network load increases. This

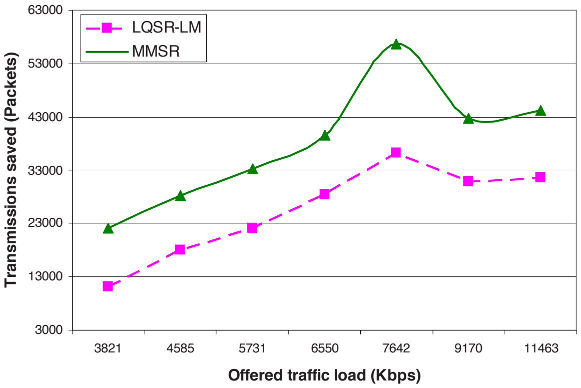


Fig. 2.10. Transmissions saved through mixing in MMSR and LQSR+LM.

is counter-intuitive since more packets should indicate more mixing opportunities. However, this occurs because of the *capture* effect in the 802.11 MAC layer which is amplified at high loads. Due to this, packets from only one node (the capturing node) fill queues for large durations of time without allowing other traffic to come in. This reduces mixing opportunities at high load. This problem can potentially be addressed through a better MAC layer design that avoids capture.

We also considered a pathological case to stress MMSR, where all three flows start exactly at the same time. We found that even in this case, throughput gains were still significant, although lower. The convergence time of MMSR is quite fast (on the order of 4–7 seconds); as a result, there is some time for long-lived flows to fully benefit from mixing.

Our results exemplify that local mixing itself cannot provide the best achievable performance and a protocol such as MMSR can help local mixing achieve its potential.

Office Mesh Scenario

We now consider how MMSR performs in a wireless office scenario [31]. Figure 2.11 gives the topology, which is the actual topology of the mesh testbed at Microsoft Research (with 42 nodes).

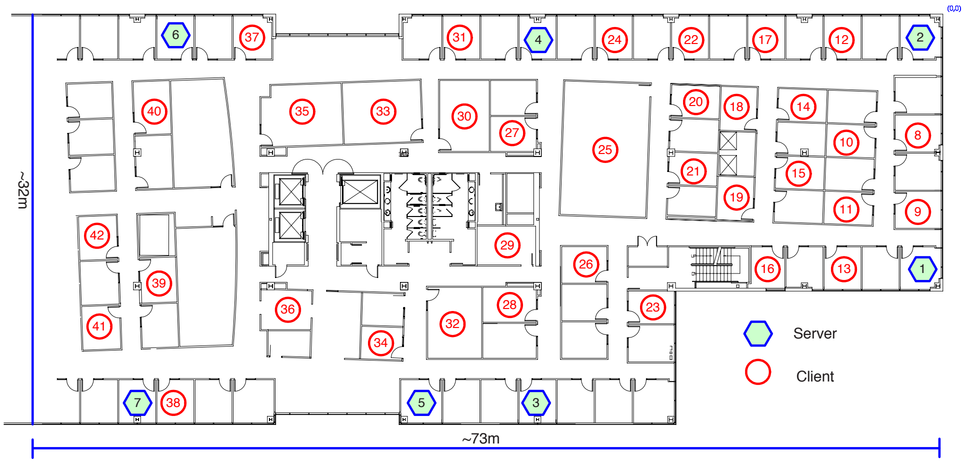


Fig. 2.11. The topology of the mesh testbed.

Table 2.2
MMSR performance in an office mesh scenario.

Scenario	Avg. Client Throughput	% Transmissions Saved
LQSR	865 Kbps	-
LQSR+LM	877 Kbps	3%
MMSR	1014 Kbps	15%

We set nodes 1–7 as servers offering different services and the remaining 35 nodes as clients. Note that a client may contact servers other than the closest one, because many servers provide a specialized service (e.g., Email/Source Control/Domain Controller). Eriksson et al. [31] observed that the traffic in an office network is predominantly download (i.e., server ⇝ client). Taking this observation into account, we simulated a traffic pattern consisting of 7 download connections, one from each server. Each connection starts and ends at random times in the simulation.

The results are depicted in Table 2.2. The results show that MMSR outperforms LQSR+LM by 16% and LQSR by 17% in throughput. MMSR also saved 15% of the transmissions (i.e., MMSR sent m source packets using 85%m transmissions). Note that the extent of gain depends on the traffic patterns and the scale of the network. A larger network with more hops would provide enhanced mixing and limit interference between flows. Additionally, if clients generate both uploads and downloads this can also provide additional gains from local mixing. At any rate, MMSR can exploit mixing opportunities if and when they occur.

2.5.3 Summary of Results

In summary, our evaluation of MMSR has shown that Markovian metrics are useful and practically applicable. The results showed that a Markovian metric can significantly improve the benefit of an advanced link-layer technique: local mixing. We found that MMSR can tolerate pathological cases (that can potentially cause oscillations) with a good convergence time, while in typical scenarios adapting flows quickly to routes that have good mixing opportunities. Finally, evaluations in an office mesh scenario also confirmed the benefit offered by MMSR and hence the Markovian metric.

2.5.4 Implementation Evaluation

MMSR was implemented in Linux by extending the SrcRR routing protocol from the RoofNet project. We also used the publicly available COPE implementation from the authors of the protocol to implement network coding. This implementation was available only for Linux.

The MMSR protocol was deployed on a subset of the MAP testbed that were available for use. These 20 nodes of the MAP testbed chosen are shown in Figure 2.12. We used the Atheros 802.11a/b/g radio on each machine for all experiments.

We now perform experiments with our Linux MMSR implementation on top of COPE in MAP. We take multiple examples of the Alice-and-Bob topologies from our testbed and demonstrate how much gain is provided by simple using SrcRR+COPE versus using MMSR+COPE. This testbed has only one radio on each node we simply use the ERC metric in MMSR. Table 2.3 shows the gains achieved using MMSR for

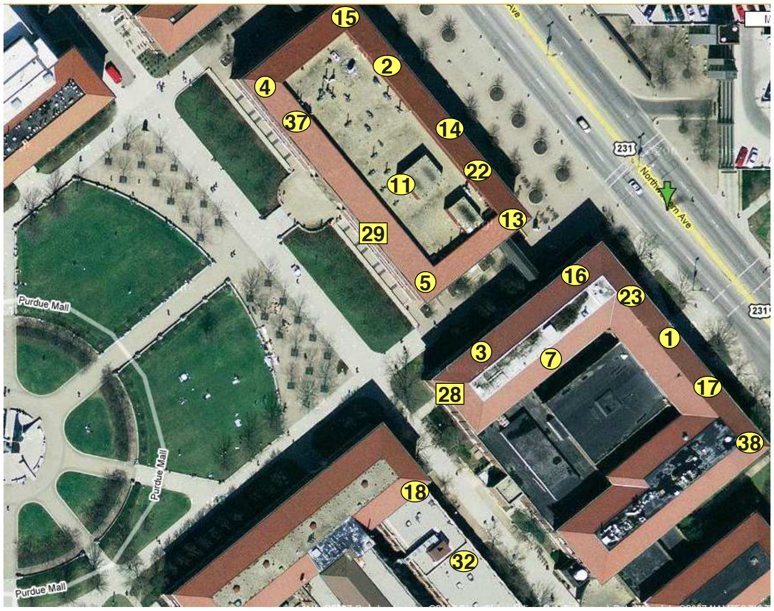


Fig. 2.12. The topology of the wireless testbed used for MMSR evaluation. A 20 node subset of the actual 32 node network was used.

Table 2.3
Median throughput gain from MMSR+COPE compared to LQSR+COPE for both UDP and TCP. For each scenario we initiated the flows 15 times.

Scenarios	UDP Gain	TCP Gain
22→3, 3→22	1.66×	1.29×
18→5, 5→18	1.53×	1.24×
11→7, 7→11	1.78×	1.31×
37→13, 2→5	1.71×	1.24×

different topologies tested in the testbed. The left hand column lists the two flows initiated in the network in each case and the node numbers refer to Figure 2.12.

The table shows significant gains from MMSR+COPE over SrcRR+COPE in the range of 1.7× for UDP and 1.27× for TCP. In the first scenario, SrcRR chooses the routes 22-13-3 and 3-5-22 which does not provide gain from network coding while causing interference among the two flows. MMSR entices flow 3-22 onto the same route due to the ERC discounts and allows network coding to occur at node 13 for the duration of the transfer. In the second scenario under SrcRR, flow 18-5 uses the route 18-28-5 while flow 5-18 uses the low quality direct route 5-18. MMSR advertises discounts on the 28-18 link for traffic from 5 and this causes the two flows to mix at node 28 resulting in gain. In the third scenario, SrcRR choose the routes 11-5-3-7 and 7-16-13-11/7-16-5-11 for the two flows resulting in no coding of packets while MMSR brings the two flows onto a mutually beneficial route traversing nodes 5 and 3 in both directions. Finally in the fourth scenario, MMSR allows nodes 37 and 2 to choose 11 as a next hop once ERC discounts become visible. Node 5 and 13 can overhear the packets from 37 and 2 respectively giving rise to network coding gain. Without MMSR the two flows tend to take the routes 37-11-5 and 2-14-22 resulting in no coding gain. Thus, we can see the benefits that MMSR provides to the COPE system in allows the network coding engine more opportunities to code packets. Note that if a lot of flows are present in the network (such as the scenarios evaluated in [45]) the benefits from MMSR would be less significant since a large number of packets present in the network can provide enough coding opportunities without intelligent route selection. However, MMSR can provide COPE with coding opportunities even in more sparse or lightly loaded networks.

2.6 Summary

In this chapter we introduced the notion of *Markovian metric*. A Markovian metric models the cost of a path as the sum of the individual conditional link costs, in a way analogous to the decomposition of the joint probability of a Markov chain into a product of conditional probabilities. We introduced the *dot graph* to represent the conditional and unconditional link costs. We showed that minimum cost routing with a Markovian metric can be done by finding a shortest path in the dot graph; hence it has polynomial complexity. We showed how to support a Markovian metric in representative routing protocols.

As a case study, we demonstrated how to use a Markovian metric to make routing decisions that better take advantage of local mixing. We proposed the expected resource consumption (ERC) as a conditional link metric that models the cost reduction due to local mixing. Markovian metric routing using the ERC link metric maximizes the total resource consumption of a path, leading to improved system efficiency. With such a Markovian metric, flows tend to self-organize themselves toward an equilibrium arrangement that can benefit each other. We proposed techniques that can facilitate flows to settle down into an equilibrium.

The local mixing engine, on its own, can improve the link layer efficiency; it identifies mixing opportunities on the fly and takes advantage of them if they are present. Routing with a Markovian metric makes local mixing more useful as it creates more mixing opportunities. This can translate to notable resource saving and throughput gain, as confirmed by simulations.

3. PROTOCOL AND ALGORITHM FOR LEVERAGING MULTIPLE RADIOS

3.1 Introduction

A promising technique for improving the capacity of wireless mesh networks is to use multiple radios. In fact, several vendors already sell routers with up to 6 radios [10, 60]. With multiple radios, a device can transmit and receive at the same time, on different radio interfaces. The radio spectrum can also be utilized more efficiently since more concurrent communications can be packed in spectrum, space and time. However, maximizing the benefit of multiple radios calls for intelligent routing decisions that are aware of multiple radios and best take advantage of them. In particular, routes that contain low interference among the constituting links should be chosen.

Routing protocols in multi-hop wireless networks traditionally find lowest cost paths, where the path cost is the sum cost for the constituting links, and the link cost reflects link quality in one way or another (e.g., the per-hop round-trip time (RTT) [2], the expected transmission count (ETX) [21], and the expected transmission time (ETT) [29]). This conventional paradigm, however, suffers from a fundamental limitation in that the links are examined in isolation of each other, whereas interference-aware routing requires proper modelling of the interdependencies of the links on a route.

Recently, progress has been made in characterizing link interdependencies and selecting interference aware routes. Draves et al. [30] proposed the WCETT routing metric for multi-radio networks and developed a link-state routing protocol called MR-LQSR. The WCETT metric penalizes a path that uses a channel multiple times. However, one issue with WCETT is that it is based on a pessimistic interference model: all links on a route that use the same channel interfere. This model fails to incorporate the possibility of channel reuse along a path. As a result, it may miss some high-throughput routes that reuse channels carefully at links sufficiently apart (see Figure 3.2). Moreover, the route selection algorithm in MR-LQSR leaves room for improvement. In [30], Dijkstra's shortest path algorithm was used for route optimization with WCETT. Although Dijkstra's algorithm is optimal for the conventional path metric, i.e. a sum of link costs, it is not optimal for WCETT.

Another proposal for interference aware routing is the MIC metric by Yang et al. [102, 103], which penalizes consecutive use of the same channel. However, an issue with the MIC metric is that it has a fixed limited memory span. A consequence of this approach is that it can no longer guarantee that a subset of a path always has a lower cost than the original path. This may result in some suboptimal paths being chosen (see Figure 3.3).

In this chapter we develop a *Context-based* Routing Protocol (CRP) for interference-aware routing in multi-radio networks, which addresses some open issues and challenges on this problem. CRP has two key components: (1) a new path metric called SIM (Self Interference aware Metric), and (2) a novel context-based path selection technique.

A key aspect of the first component, the SIM metric, is that it is context-dependent; i.e., it assesses the cost of each link in the context of previous hops for the path in consideration. The SIM metric is a weighted sum of the estimated transmission time (ETT) and the maximum expected service interval (ESI) at the bottleneck link. Similar to how the long term throughput of a pipelining system is determined by the bottleneck component, the long term throughput of a wireless route is determined by the bottleneck link. We show that the maximum ESI term can be interpreted as an ideally achievable throughput assuming perfect scheduling.

Since the SIM metric is context-dependent, a direct application of the Dijkstra's shortest path algorithm may lead to rather suboptimal paths, as also observed by Draves et al. [29] for WCETT. The reason is that Dijkstra's algorithm operates on an optimality principle: The shortest path from s to t via v is the shortest path from s to v concatenated with the shortest path from v to t. Such optimality principle no longer holds for a context-based metric. Thus, the second component of CRP is a general context-based path pruning method (CPP), for discovering good paths using a context-based metric. To model the context of a partial path while avoiding the exponential growth in the number of partial paths, CPP maintains a set of paths that reach each node v, corresponding to different *local contexts*. Each local context can be viewed as a concise summary of the dominating history of all the partial paths that end with that local context, based on the observation that older history usually has less impact on the future. The best path under each local context is maintained and considered further for possible expansion into an s–t path.

The essence of the proposed interference-aware routing solution lies in (i) properly modelling the link interdependencies (via the SIM metric) and (ii) handling the ensuing algorithmic challenges in route optimization (via the CPP method). The use of local contexts in pruning is synergistic with the use of contexts in cost modelling. Together they present a general context-based routing paradigm, which is applicable in scenarios where modelling link-interdependencies is critical. Section 3.5 discusses some other potential applications that may benefit from a context-based solution.

In summary, our contributions are: (1) We advocate that interdependencies among links require a new approach to cost modelling and propose a novel context-based path metric, SIM, which can effectively account for such interdependencies. We also provide a theoretical justification of the proposed metric. (2) We propose a novel context-based path selection method, CPP, which finds better paths than a direct application of Dijkstra's algorithm for context-based metrics. Our test-bed implementation shows that CPP is computationally practical. (3) We thoroughly evaluate the performance of CRP through Qualnet simulations and a testbed deployment and find that it provides TCP throughput gains up to 130% and on average around 50% over state-of-the-art approaches even in a small 14-node testbed. (4) Our proposal of using context and finding paths with context information is a generally applicable technique whenever link interdependencies exist and potentially has further applications than multi-radio networks. We discuss some such applications.

The rest of the chapter is organized as follows. Section 3.2 and Section 3.3 describe the SIM metric and CPP optimization method respectively. Performance results of our routing method are presented in Section 3.4. Finally, Section 3.6 concludes the chapter and Section 3.5 talks about other applications of our proposed context-based routing technique.

3.2 Self-Interference Aware Routing Metric

In this section, we describe the first component of CRP: the SIM metric. We consider a multi-hop wireless network equipped with multiple radios. Similar to [30], we assume each radio is tuned to a fixed channel for an extended duration; a route specifies the interfaces to be traversed.

We define the SIM (self-interference aware) metric, as a weighted sum of two terms:

$$\mathsf{SIM}(\mathcal{P}) \triangleq (1-\beta)\sum_k \mathsf{ETT}(e_k) + \beta \max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1}).$$

3.2.1 A Review of ETT

The first term is the sum of the expected transmission time along the route. The ETT [30] metric aims at estimating the average transmission time for sending a unit amount of data. It is defined as:

$$\mathsf{ETT} \triangleq \frac{\mathsf{ETX}}{\text{Link Bit-Rate}}. \tag{3.1}$$

Here the ETX [21] metric estimates the number of transmissions, including retransmissions, needed to send a unicast packet across a link. It is computed as $\mathsf{ETX} \triangleq 1/(1-p)$, where p is the probability that a single packet transmission over link e is not successful. The link bit-rate in (3.1) can be measured by the technique of packet pairs and this method can produce sufficiently accurate estimates of link bandwidths [30].

3.2.2 ESI of the Bottleneck Link

The second term is the estimated service interval (ESI) at the bottleneck link, which reflects how fast the route can send packets in the absence of contending traffic. Note that the max operation is used here instead of the sum operation. This can be explained by a pipeline analogy. In a pipeline system consisting of several processing stages, the long term throughput is determined by that of the bottleneck stage, rather than the average throughput of the stages. Sending packets along a wireless route is similar. The long term throughput of a flow is determined by that of the bottleneck link.

The ESI of a link is defined as:

$$\mathsf{ESI}(e_k|\mathcal{P}_{k-1}) \triangleq \mathsf{ETT}(e_k) + \sum_{j<k} p_{jk}\mathsf{ETT}(e_j). \tag{3.2}$$

Here p_{jk} is a binary number that reflects whether e_j and e_k interfere. Characterizing the interference relations among the links is itself a research challenge. For instance, one method is to make use of actual interference measurement; see, e.g., Padhye et al. [69] and the references therein. Consider two links, $A \rightarrow B$ and $C \rightarrow D$, using the same channel. As suggested in [69], for 802.11 networks, the primary forms of interference are four cases: (i) C can sense A's transmission via physical or virtual carrier sensing, and hence refrains from accessing the medium, (ii) A can sense C's transmission, (iii) transmissions of A and C collide in D, (iv) transmissions of A and C collide in B. Based on this, we adopt a simplified approach in the experiments of this chapter. We treat the two links as interfering if A has a link to C or D with sufficiently good quality, or C has a link to A or B with sufficiently good quality.

The ESI expression (3.2) leaves out the interference caused by other contending traffic. This is a simplification in modeling. The ESI expression (3.2) considers the self-interference from the previous hops of the route, by adding up the expected transmission times of the previous links. The intuition is that the packets at the link needs to share the channel with the interfering links on the route. One might ask why we do not add the ETTs from the subsequent links on the path, even though both previous and subsequent links can create interference. The following theorem provides an answer.

Theorem 3.2.1 (Interpretation of Bottleneck ESI) *Assuming ideal scheduling, sufficiently long flow, absence of contending traffic, and an ideal binary interference model dictated by a conflict graph, the end-to-end throughput of* $1/\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$ *is achievable.*

Proof: Under the assumptions in the claim, finding the optimal end-to-end throughput essentially amounts to finding an optimal interference-free scheduling of the uses of the constituting links. If we can schedule each link to transfer B bits in T seconds, then the throughput B/T can be achieved. It is well known that this problem can be viewed as a continuous version of the graph coloring problem on the conflict graph.

In greedy coloring algorithms, nodes in a graph are visited one by one. Each node tries to reuse some existing colors if possible. If not, the node selects a new color. With this procedure, it is easy to see that the graph can be colored in $\Delta(U) + 1$ colors, where $\Delta(U)$ is the maximum degree of a vertex. Notice that the greedy coloring algorithm always look at the already colored nodes, but not future nodes. Hence in fact the upper-bound can be tightened to one plus the maximum number of already-colored neighbors for the nodes.

We now apply a greedy-coloring like algorithm for scheduling the links on a route. This is illustrated in Figure 3.1 for a path with 6 links. We visit the links on the route sequentially, from the first hop to the last hop. For each link e_k, find one or more intervals with a total length of $\mathsf{ETT}(e_k)$. Similar to greedy coloring, when assigning the intervals to a link, we only need to examine the previous links, but not future links. With this greedy scheduling process, we can finish the assignment in a total duration of $\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$. If we repeat this scheduling pattern for a sufficiently long time, then we can deliver one packet end to end every $\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$ (sec). Hence the throughput is achievable. ■

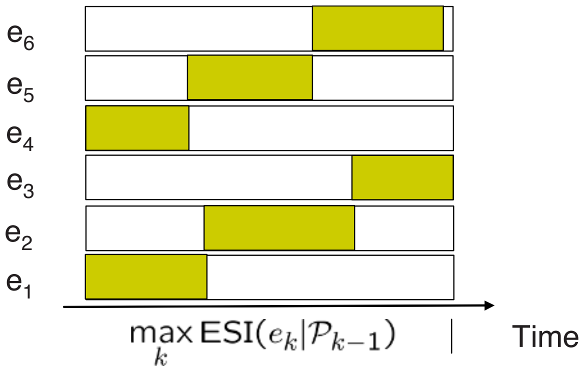


Fig. 3.1. An interference-free scheduling of links. Assume e_k interferes with e_{k-1},e_{k-2}, e_{k+1}, e_{k+2}. A shaded region for a link e_k indicates that e_k is using the medium.

The above theorem shows that the bottleneck ESI corresponds to a theoretically achievable throughput. Conversely, if a link e_k interferes with a set $\mathcal{F}$ of previous links, then typically links in $\mathcal{F} \cup \{e_k\}$ would be expected to mutually interfere (hence forming a clique in the conflict graph). If that indeed is the case, then we cannot deliver more than one packet end to end every $\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$ (sec). This non-rigorous argument shows that the maximum throughput is roughly around $\max_k \mathsf{ESI}(e_k|\mathcal{P}_{k-1})$.

3.2.3 An Alternative System Model

Earlier we assumed each radio interface is tuned to a fixed channel for an extended time duration. We call this a “static” channel assignment.

Kyasanur et al. [50] proposed an alternative system model for using multiple radios, when each node has at least two radios. In their model, for each node, some of its radio interfaces are *fixed* at certain channels and the remaining are *tunable*. To send a packet to a node v, a node tunes one of its tunable interfaces to one of v’s fixed interfaces. We call this model a “semi-static” channel assignment. Under the semi-static model, to specify a route, we can specify the sequence of receiving interfaces (e.g., via their IP addresses). The ESI metric can be applied essentially with no change to the semi-static model as well.

3.2.4 Comparison with WCETT and MIC

The bottleneck ESI models self-interference. The previously introduced metrics, WCETT and MIC, also consider self-interference. We now compare the proposed SIM metric with WCETT and MIC.

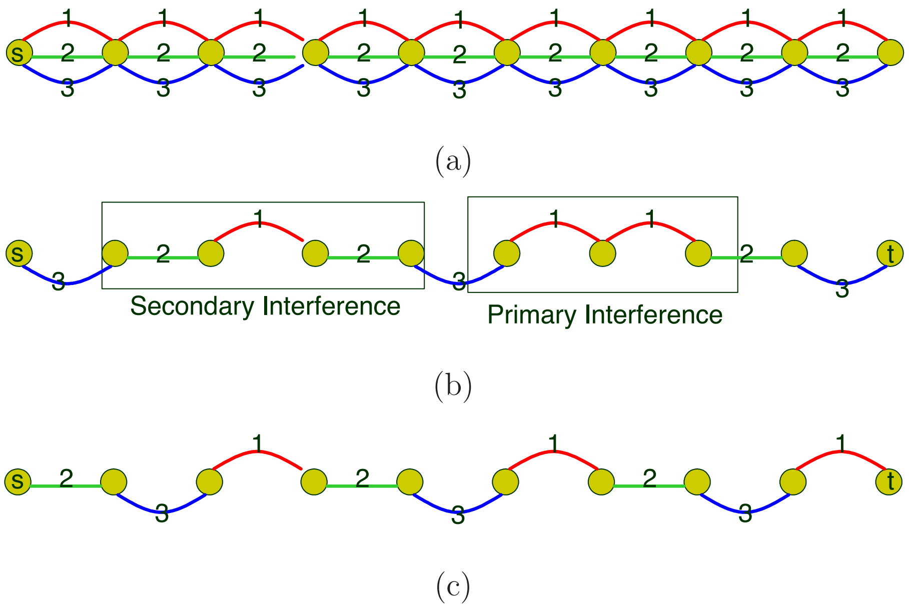
The WCETT metric proposed by Draves et al. [30] is defined as:

$$\begin{aligned} \mathsf{WCETT}(\mathcal{P}) &\overset{\Delta}{=}(1-\beta)\cdot\sum_{e_k\in\mathcal{P}}\mathsf{ETT}(e_k)+\ \beta\cdot\max_{\text{Channel } j} X_j, \\ X_j &\overset{\Delta}{=} \sum_{e_k \text{ is on channel } j} \mathsf{ETT}(e_k). \end{aligned} \tag{3.3}$$

Here e_k denotes the k-th hop on the path $\mathcal{P}$ and β is a weighting factor between 0 and 1. The WCETT metric is a weighted sum of two terms. The first term is the sum of the ETT along the path. The second term aims at capturing the effect of self-interference. A path that uses a certain channel multiple times will be penalized.

Although WCETT considers self-interference, it has some drawbacks. Consider the following example. Figure 3.2(a) shows a chain topology, where each node has three radios tuned to orthogonal channels 1,2,3. We assume two links within 2 hops interfere with each other if they are assigned the same channel. We further assume all links have similar quality. An ideal route in this setting is then a route that alternates among the three channels, such as the one illustrated by Figure 3.2(c), because it can completely avoid self-interference. Figure 3.2(b) shows a possible solution returned by WCETT. This route suffers from primary interference as well as secondary interference. This can be explained by the way WCETT models self-interference. From (3.3), it is seen that WCETT views a path as a set of links. In this example, it tries to maximize channel diversity by balancing the number of channels used on the path. Thus the path in Figure 3.2(b) and the path in Figure 3.2(c) appear equally good under WCETT.

In essence, WCETT takes a pessimistic interference model that all links on the same channel in the route interfere with each other. Indeed, if we use this interference model, then SIM reduces to WCETT. The proposed SIM metric models self-



(a)

(b)

(c)

Fig. 3.2. (a) A chain topology, where each node has three radios tuned to orthogonal channels 1,2,3. (b) An example route selected by using the WCETT metric. (c) An example route selected by using the SIM metric.

interference can differentiate different ordering of the links in the path and find the optimal path for this example. The benefit is that SIM potentially allows better channel reuse.

Another related work that considers self-interference is by Yang et al. [102, 103], who proposed the metric of interference and channel-switching (MIC). The metric is defined as:

$$\mathsf{MIC}(\mathcal{P}) \triangleq \alpha \sum_{e_k \in \mathcal{P}} \mathsf{IRU}(e_k) + \sum_{\text{Node}i \in \mathcal{P}} \mathsf{CSC}_i, \tag{3.4}$$

$$\mathsf{IRU}(e_k) \triangleq \mathsf{ETT}(e_k) \times N_k, \tag{3.5}$$

$$\mathsf{CSC}_i \triangleq \begin{cases} w_1 & \text{if previous hop is on a different channel,} \\ w_2 & \text{otherwise.} \end{cases},$$

$$0 \leq w_1 < w_2. \tag{3.6}$$

Here $\alpha > 0$ is a weighting factor to balance the two terms. In (3.5), N_k is the number of neighbors that interferes with the transmission of link e_k. The IRU (interference-aware resource usage) term aims at reflecting the inter-flow interference. A link that interferes with more neighbors will be penalized. The CSC (channel-switching cost) aims at reflecting the self-interference, since it penalizes consecutive uses of the same channel.

The total CSC term (3.6) in the MIC metric models the self-interference by considering the immediate previous hop. As an extension, the authors also considered the extension of the MIC metric to model self-interference for more than one hops [103]. However, a potential limitation with the MIC metric and its multi-hop extension is that it has a fixed limited memory span. Consider the example in Figure 3.3. Suppose the links between nodes a and b have very low costs. Under the MIC metric, it is possible that the route $s \rightarrow a \rightarrow b \rightarrow a \rightarrow t$ has a lower cost than the route $s \rightarrow a \rightarrow t$. Since the path is selected by optimizing the metric, it is unclear whether the MIC metric can rule out the possibility of selecting a pathological path, which has self-interference that are not modelled in the MIC expression.

Compared with the MIC metric, the SIM metric considers the possible interference with all previous hops. This avoids the pathological case as shown in Figure 3.3. In addition, whereas the link CSCs are summed up in (3.4), SIM uses the maximum

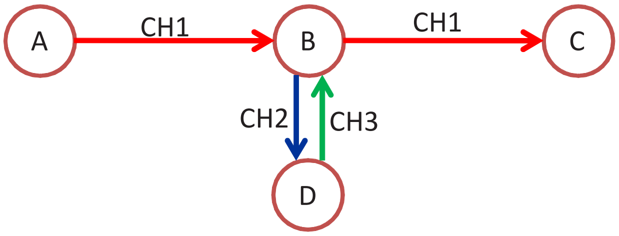


Fig. 3.3. Under the MIC metric, it is possible that the route $s \rightarrow a \rightarrow b \rightarrow a \rightarrow t$ has a lower cost than the route $s \rightarrow a \rightarrow t$. Edges labeled with channel.

ESI. We feel that the use of the maximum operation can better reflect the fact that the throughput is determined by the bottleneck link.

3.3 Context-Based Path Pruning

In this section, we describe the second component of CRP: the CPP path selection algorithm. In addition to defining a good path metric, another challenge is to find the optimal (or near-optimal) route under the path metric. As a starting point, we first consider a link state routing framework. In link state routing, each router measures the cost to each of its neighbors, constructs a packet including these measurements, sends it to all other routers, and computes the shortest paths locally. Hence for link state routing, what's needed is a centralized algorithm for computing the shortest paths. The CPP method can also be applied in some other distributed settings; this is discussed in Section 3.3.3.

Let us begin by reviewing the (simpler) problem of finding the optimal path under a decomposable path metric, where each link has a nonnegative cost and the cost of a path is the sum of the costs of the constituting links. This problem is well understood. For example, the classical shortest path algorithm by Dijkstra can be applied to find the optimal path with complexity $O(|V|^2)$, where $|V|$ denotes the number of nodes in the network. Dijkstra's algorithm maintains upper-bound labels $f(v)$ on the lengths of minimum-cost s–v paths for all $v \in V$. The label of each node is either *temporary* or

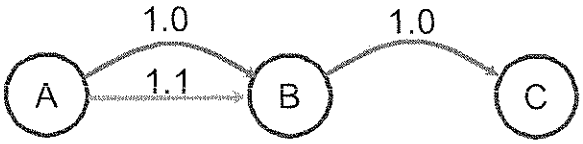


Fig. 3.4. Example of Dijkstra's problem with non-decomposable metrics. CH1 and CH2 are 2 orthogonal channels, from top to bottom. Edges labeled with ETT.

permanent, throughput the execution of the algorithm. At each iteration, a temporary label with least total cost is made permanent and the remaining temporary labels are updated. Specifically, if v^* has the minimum total cost among the temporary nodes, then we update the cost of every other temporary node w as:

$$f(w) := \min \{f(w), f(v^*) + c(v^*w)\}. \qquad (3.7)$$

The SIM metric is history-dependent; as a result, finding the shortest path under the SIM metric is a new challenge. For example, a direct application of the Dijkstra's shortest path algorithm may lead to highly suboptimal paths. Consider the example shown in Figure 3.4. Here there are two orthogonal links from A to B, and one link from B to C. The ETT metrics for the links are shown on the links. Then Dijkstra's algorithm (using (3.8)) will return a route that uses $A \xrightarrow{\text{CH1}} B \xrightarrow{\text{CH1}} C$. (The channel number is written above the arrow.) However, the better route is $A \xrightarrow{\text{CH2}} B \xrightarrow{\text{CH1}} C$.

The reason is that Dijkstra's algorithm operates on an optimality principle: The shortest path from s to t via v is the shortest path from s to v concatenated with the shortest path from v to t. Thus, each node only needs to remember the cost of the best s–v path. Such optimality principle no longer holds for a metric such as SIM or WCETT; instead, an s–v path $\mathcal{P}_1$ may have a larger cost than an s–v path $\mathcal{P}_2$ but may eventually lead to a lower cost toward the final destination node t.

We propose a context-based path pruning (CPP) technique, as a (heuristic) method for optimizing a context-based metric. To model the potential impact of past hops on future hops, we maintain a *set* of paths that reach each node v, instead of a single s–v

path with minimum cost. A natural question is: How many paths should we store at each node as we search for a good s–t path? Storing all paths would apparently result in an exponential complexity. To keep the complexity manageable, we must constrain the amount of memory at each node. Our key observation is that the effect of self-interference has a localized nature. Normally, given what happened in the near past (i.e., first few previous hops), the older history is unlikely to have a significant impact on the future. This observation motivates us to organize the memory at each node according to several *local contexts*. Each local context can be viewed as a concise summary of the dominating history of all the partial paths that end with that local context, based on the observation that older history usually has less impact on the future. During the execution of the proposed algorithm, each node stores the best path under each possible local context. The best paths stored are considered further for potential expansion into an s–t path.

Degrees of freedom exist in defining the local contexts. For example, we can define the local context of a path as the sequence of links in the last l hops. As another example, we can define the local context of a path as the sequence of channels taken by the links in the last l hops. These would both serve as reasonable summaries of the past that would impact the future.

3.3.1 Dijkstra-style Realization of CPP

Figure 3.5 shows a Dijkstra-style instantiation of the CPP method. We maintain a set T of temporary paths and a set P of permanent paths. In each step, we choose the temporary path with minimum cost. Such a path, say $\mathcal{P}^*$, is made permanent. Then we consider the possible ways of extending $\mathcal{P}^*$ toward an s–t path. For each extension $\mathcal{P} = \mathcal{P}^* + e$, we determine its local context and search for a path with the same local context in T and P. If a path with the same local context already exists, then such existing path is compared with $\mathcal{P}$ and the winner is retained. If a path with the same local context does not exist, then $\mathcal{P}$ is added to T.

```
INPUT: A function that can evaluate the cost of a path.
$T := \{s\}$; /* The set of temporary paths. */
$P := \emptyset$; /* The set of permanent paths. */
while $T \neq \emptyset$ do
  choose the path $\mathcal{P}^*$ from $T$ with the minimum cost;
  $T := T - \mathcal{P}^*$; $P := P + \mathcal{P}^*$;
  for each valid extension of $\mathcal{P}^*$, say $\mathcal{P} = \mathcal{P}^* + e$, do
    $c :=$ LocalContext($\mathcal{P}$);
    if $T \cup P$ contains a path $\mathcal{Q}$ with local context $c$ then
      replace $\mathcal{Q}$ by $\mathcal{P}$ if it has a lower cost than $\mathcal{P}$;
    else
      $T := T + \mathcal{P}$;
    end if
  end for
end while
OUTPUT: For each node $v$, find the best local context $c^*(v)$ resulting in minimum
cost. For each context $c$ of each node $v$, store the best link reaching it with minimum
cost. To recover a route from $v$ to $s$, back-track from $c^*(v)$ along the best links
toward $s$.
```

Fig. 3.5. A Context-based Path Pruning Method

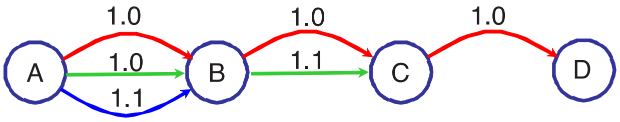


Fig. 3.6. An example graph. There are three orthogonal channels, CH1, CH2, CH3, from top to bottom.

There is an alternative way to understand the CPP method. For each physical node v, introduce one vertex v_c for each local context c applicable to v and interconnect the vertices according to original connectivity. Denote such a *context-expanded graph* by G_c. Algorithm 3.5 can be understood as applying the Dijkstra's algorithm to the expanded graph G_c to find a shortest path tree in G_c.

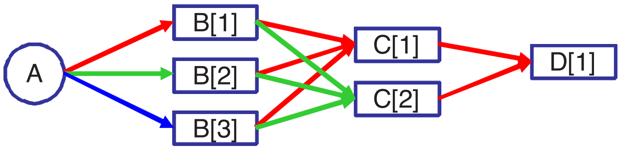


Fig. 3.7. The expanded graph for the case the local contexts are defined by the previous hop's channel ID. Here $X[i]$ corresponds to the local context where the incoming link is on channel i.

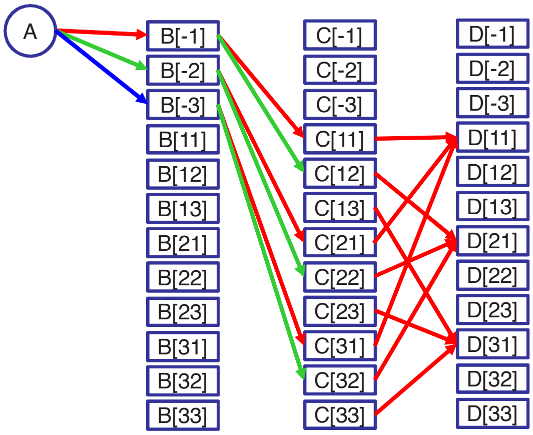


Fig. 3.8. The expanded graph for the case the local contexts are defined by the previous two hops' channel IDs. Here $X[ij]$ corresponds to the context where this node is reached via a link in channel i, followed by a link in channel j. $X[-j]$ refers to the local context where this node is directed reached from the source via a link in channel j.

Since here the path metric is not decomposable, the cost update step (3.7) needs to be revised. Instead of using (3.7), node v^* first reconstructs the best path from the source, say $s \overset{\mathcal{P}}{\leadsto} v^*$. Then each neighbor node w of v is updated as

$$f(w) := \min\left\{f(w), \mathsf{cost}\left(s \overset{\mathcal{P}}{\leadsto} v^* \to w\right)\right\}, \tag{3.8}$$

where $\mathsf{cost}(\cdot)$ returns the path metric.

We now illustrate this context-expanded graph using an example shown in Figure 3.6. Here there are there orthogonal links from A to B, two links from B to C, and one link from B to C. The ETT metrics for the links are shown on the links. Consider $\beta = 0.5$ and the simplified interference model mentioned in Section 3.2. In this example, Dijkstra's algorithm with (3.8) will return $A \xrightarrow{\text{CH1}} B \xrightarrow{\text{CH2}} C \xrightarrow{\text{CH1}} D$, with a total cost of 2.55.

Figure 3.7 shows the expanded graph for the case the local contexts are defined by the previous hop's channel ID. Taking the original node B as an example, there are three nodes in G_c, which are associated with three channels to reach B. To connect Algorithm 3.5 with running Dijkstra's algorithm (with (3.8)) over the expanded graph, we can view each node as storing the optimal path reach it from the source; the optimal path can be obtained by backtracking along the best links that reach each node.

In this case, the optimal route found is $A \xrightarrow{\text{CH2}} B \xrightarrow{\text{CH1}} C \xrightarrow{\text{CH1}} D$, with a total cost of 2.5.

If the local contexts are defined by the previous two hops' channel IDs, then the resulting expanded graph would be the one shown in Figure 3.8(c). This will yield the optimal route $A \xrightarrow{\text{CH3}} B \xrightarrow{\text{CH2}} C \xrightarrow{\text{CH1}} D$, with a total cost of 2.1.

Optimality

If the path metric indeed has a fixed memory span (say, l hops), then CPP with the local context defined by the l-hop links is guaranteed to find the optimal solution (because no pruning step is suboptimal).

In our case, the SIM path metric has a memory span that could potentially involve the entire path history. Even if the path metric has a longer memory span than the length of the local contexts, the CPP method can still be applied as an effective heuristic method.

Would CPP with a limited local context length also suffer the problem in Figure 3.3? The answer is no. This is because although the local context used in CPP only has limited information about the path history, the path metric of SIM has the information of the complete path. In particular, the route $s \rightarrow a \rightarrow b \rightarrow a \rightarrow t$ is still penalized.

Complexity

As we mentioned earlier, Algorithm 3.5 can be essentially viewed as applying Dijkstra's algorithm with (3.8) over the expanded graph. Note though that in Algorithm 3.5, the involved vertices and links are constructed on the fly, without explicitly maintaining the expanded graph. Due to the connection between Algorithm 3.5 and Dijkstra's algorithm, we can easily conclude that the time complexity of the Algorithm 3.5 is $O(C^2)$, where C is the total number of local contexts at all nodes. If we define the local context of a path as the sequence of channels taken by the links in the last l hops, then C is upper-bounded by $(K + K^2 + \ldots + K^l) * |V(G)|$, where K is the number of channels in the system and $V(G)$ is the set of nodes in the original network. For practical purposes, we specifically propose to use $l = 2$; see Figure 3.6 for an example definition of the local contexts. This would lead to a specific complexity of:

$$O \quad \left((K + K^2) \cdot |V|\right)^2 \quad . \tag{3.9}$$

3.3.2 Related Work: Comparison with the Route Selection Method in [102, 103]

Yang *et al.* [102, 103] proposed a method for finding the optimal route under the MIC metric (3.4) and its multi-hop generalization in polynomial time complexity. The method hinges upon the fact that the path metric is decomposable into a sum of link costs, where the cost of a link depends only a fixed number of previous hops; in other words, the path metric is additive and has a fixed memory span. In [102, 103], a virtual graph is constructed, by introducing virtual vertices and edges to represent different states; each edge has an associated cost. Then finding an optimal route under the decomposable, finite-memory metric in the original problem becomes the problem of finding an optimal path in the virtual graph under a memoryless metric, where the cost of a path is simply the sum of the costs of the constituting edges. Therefore, for the additive path metric with a fixed memory span, the optimal solution can be found in polynomial time.

In comparison, the CPP method is applicable for all path metrics, not only when the path metric is decomposable and has finite-memory. When the path metric is decomposable and has finite-memory, CPP also finds the optimal answer. When the path metric is not decomposable, CPP can still be used, as a heuristic method. This comes from the fact that CPP operates by always examining the partial-paths from s and applying the path metric to evaluate their costs, which is apparent from Algorithm 3.5, as well as the path-based update rule in (3.8). Observe also that the edges in Figures 3.7 and 3.8 do not have associated costs, in contrast to the virtual graph in [102, 103]. Although the local context used in CPP only has limited information about the path history, the path metric has the information of the complete path and can properly take into account any local or global link interdependency. This mix of "local" memory and "global" path evaluation metric is a distinct feature of the CPP method, rendering it generally applicable and efficient.

3.3.3 Applying CPP to Other Protocols

The CPP method is a general method to efficiently explore the path space and can also be applied to other settings such as distance vector protocols and on-demand route discovery.

First, let us examine how CPP can be implemented in a framework based on the Bellman–Ford shortest path algorithm. The Bellman–Ford algorithm maintains for each node v first the minimum cost s–v path with 0 interior node, then the minimum cost s–v path with ≤ 1 interior node, then the minimum cost s–v path with ≤ 2 interior nodes, and so on. Similar to what we did for Dijkstra's method, to use the CPP method, we essentially can run the Bellman–Ford algorithm over the context-expanded graph G_c. Specifically, we maintain a set Q_k of paths. For $k = 1$ to $|V(G)| - 2$, we consider the paths in the Q_k one by one. For each path $\mathcal{P}^*$ in

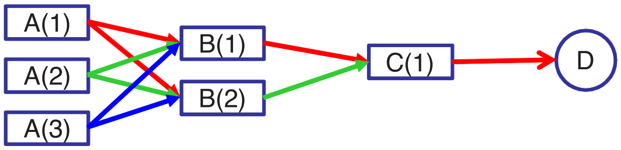


Fig. 3.9. The reverse context-expanded graph for the example in Figure 3.6.

consideration, we examine all possible paths obtained by extending $\mathcal{P}^*$ by one more hop. For each extended path $\mathcal{P} = \mathcal{P}^* + e$, we then check if there exists a path $\mathcal{Q}$ in the current set with the same local context. If not, the extended path $\mathcal{P}$ is inserted into the set; otherwise, the path $\mathcal{P}$ is compared with the existing path $\mathcal{Q}$ and the better is retained.

The practical counterpart of Bellman–Ford's algorithm is the distance vector protocol (e.g. DSDV [72]). Originally, each router maintains a table (i.e., a vector) giving the best known cost to reach each destination and the associated interface. Tables are updated by exchanging information with the neighbors. To adapt to our current scenario, first, we construct the context-expanded graph in a reverse manner. For the example in Figure 3.6, the reverse context expanded graph is shown in Figure 3.9. Here the local context is defined as the outgoing channel. Since the expanded vertices $\{v_c\}$ do not physically exist, we let each physical node play their roles. Nodes exchange the best paths to reach each physical destination in the reverse expanded graph. Once every T milliseconds each router u sends each neighbor a list of its estimated minimum costs to reach each physical node v for each applicable outgoing context c. For example, C will announce that it has a path to D; then B will announce that it has two paths to D, $B \xrightarrow{\text{CH1}} C \xrightarrow{\text{CH1}} D$ and $B \xrightarrow{\text{CH2}} C \xrightarrow{\text{CH1}} D$; next, A will announce that it has three (=number of local contexts) paths to D, $A \xrightarrow{\text{CH1}} B \xrightarrow{\text{CH2}} C \xrightarrow{\text{CH1}} D$, $A \xrightarrow{\text{CH2}} B \xrightarrow{\text{CH1}} C \xrightarrow{\text{CH1}} D$, and $A \xrightarrow{\text{CH3}} B \xrightarrow{\text{CH2}} C \xrightarrow{\text{CH1}} D$. If A needs a route to D, then A will identify the last one as the best path.

CPP can also be applied to on-demand route discovery protocols such as DSR [40] and DSR with ETX [21]. In DSR, a source node broadcasts route request packets to discover routes to a destination. When a node forwards a route request, it appends its address and the related link costs. To implement CPP in DSR, when a node receives a request it has already forwarded, it should forward it again if the path cost to a neighbor v is better than the best which it has already forwarded with the same local context.

3.4 Performance

This section documents our experience with the Context Routing Protocol using simulations and a real deployment. We begin first with describing our methodology.

3.4.1 Evaluation Methodology

3.4.2 Simulation Setup

We implemented CRP in QualNet 3.9.5, a widely used event-driven simulator for wireless networks. CRP was implemented by extending the existing DSR implementation in QualNet to support multiple interfaces, routing using a graph cache, ETX and packet pair probing for ETT calculations, periodic dissemination of link metrics, the WCETT metric, the SIM metric and the CPP route selection method. The simulations use the 802.11a MAC and a two-ray fading signal propagation model. All radios operate at a nominal physical layer rate of 54 Mbps and support autorate. We use both UDP and TCP flows in the evaluation. The simulations use 2-hop context length.

3.4.3 Testbed Setup

To evaluate CRP, we also implemented it in Windows XP. CRP was implemented by extending the MR-LQSR protocol [29] implementation which is part of the publicly available Mesh Connectivity Layer(MCL) framework. Since MCL is a loadable Windows driver that sits at layer 2.5, CRP appears like a single virtual network

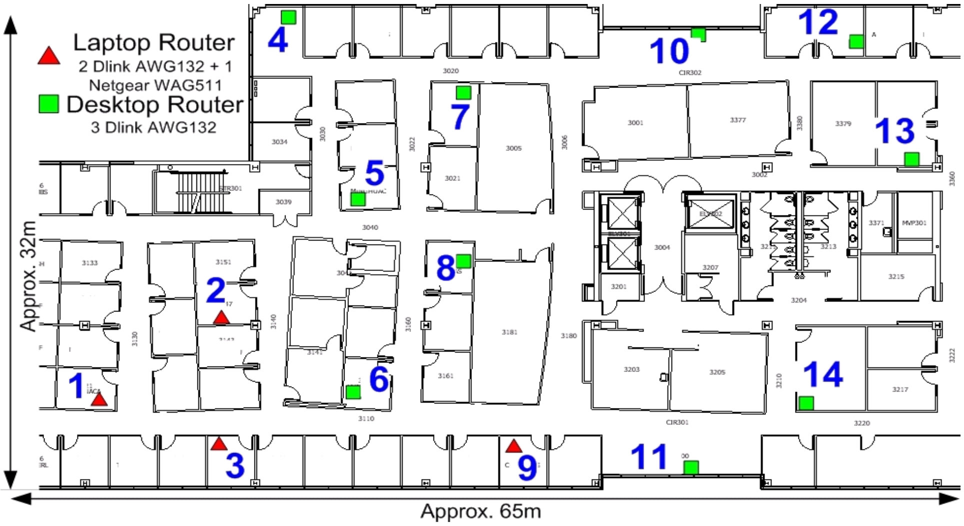


Fig. 3.10. The topology of the wireless testbed.

adapter to applications by hiding the multiple physical interfaces bound to it. This allows unmodified applications to run over CRP. CRP extends the MR-LQSR protocol, which itself is derived from DSR. MR-LQSR already implements the WCETT metric including the link probing functionality, link-state dissemination and a graph cache, all of which are leveraged by CRP. Routing operates using 48 bit virtual ethernet addresses of the MCL adapter in each node. To support multiple interfaces, each link is specified using this virtual address and the incoming and outgoing interface. The channel number being used on each link is also encoded using the lower 8 bits of the metric values. This choice for our CRP implementation allows a direct and fair comparison with the WCETT metric since it is based on the same underlying codebase. The CRP code uses 2-hop channel ids as context.

The CRP protocol was deployed on a 14 node wireless testbed running Windows XP. The testbed is located on a floor of an office building and is illustrated in Figure 3.10. 10 of the nodes are small form factor HP desktops each with 3 DLink AWG132 802.11a/b/g USB cards while 4 nodes are Toshiba tablet PCs with one Netgear WAG511 802.11a/b/g PCMCIA card and 2 DLink AWG132 802.11a/b/g cards. The driver configurations are hacked to allow multiple cards from the same vendor to co-exist in a machine. The radios on each node have their own SSID and are statically assigned the 802.11a frequencies 5180, 5240 and 5300 GHz. Thus, the 42 radios form three 14-node networks each with its own frequency and SSID glued together through the CRP virtual adapter. Each radio works in ad-hoc mode and performs autorate selection. The OS on each machine implements TCP-SACK. Finally, the nodes are static and use statically assigned private IPv4 addresses. The results are expected to not be significantly affected by external interference since no other 802.11a network was in the area.

3.4.4 Simulation Evaluation

We first demonstrate the performance of CRP in a controlled and configurable simulator setting.

CRP in a Frequency Reuse Scenario

The first scenario demonstrates how CRP can exploit frequency reuse opportunities in a correct way. We consider a chain of 10 nodes separated by 300m, each with 3 radios on 3 orthogonal channels. The transmission power used causes interference to nodes even two hops away which is typical in real networks. Figure 3.11 shows the routes selected by using WCETT (on top) and CRP (below).

Notice the route selected by WCETT: while it maximizes channel diversity, i.e. each channel is used exactly equally (3 times), it cannot optimize the ordering of the channel use (as explained in Section 3.2.4) while CRP chooses the optimal route. How does this route selection impact performance?

Figure 3.11 also shows the UDP and TCP throughput achieved using ETT, WCETT and CRP on this chain topology. The results show that CRP route selection has a significant impact on performance improving UDP throughput by 166% over WCETT and over 1000% compared to ETT. For TCP, CRP improves the throughput by around 44% and 91% with respect to WCETT and ETT respectively. Thus, CRP

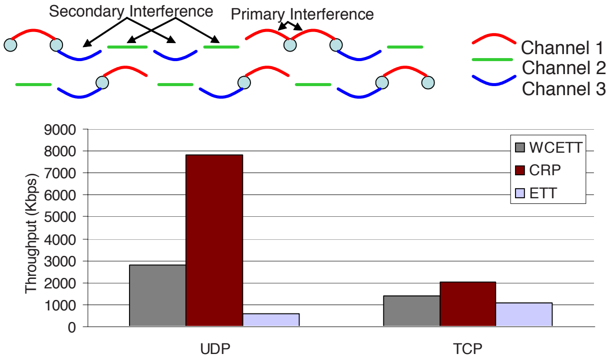
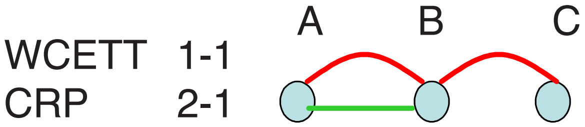


Fig. 3.11. Performance under frequency reuse.



	TCP (Kbps)	UDP (Kbps)
ETT	1735	5550
WCETT	1735	5550
CRP	3617	11230

Fig. 3.12. Performance under heterogeneous nodes.

is able to select a path with high raw capacity. However, the gains are reduced due to TCP's known inability to reach the available bandwidth effectively in multi-hop wireless networks (which is also amplified by the long chain). An interesting aspect to note here is that CRP can choose this route even by running normal Dijkstra but using the SIM metric, i.e. the benefit comes from the context in the metric. However, the next section shows when SIM by itself is not enough and demonstrates the usefulness of context-based path selection.

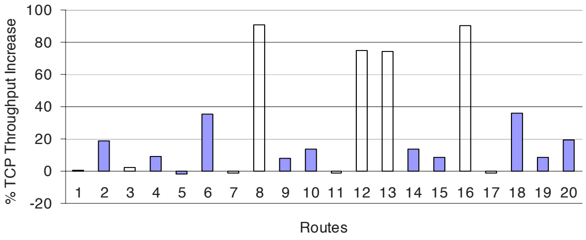


Fig. 3.13. Performance in a large network.

CRP in a Heterogeneous Node Scenario

In this scenario, we consider a simple 3 node topology as shown in Figure 3.12. Node A and B have 2 radios tuned to channel 1 and 2 while node C has one radio tuned to channel 1. Even in such a simple topology, WCETT and ETT are unable to choose the good route which can simply be decided by observation. This is because of the 'forgetfulness' of Dijkstra. Once the route to B is decided, there is no context with which to select the next hop accurately. Thus, both ETT and WCETT in MR-LQSR choose the path with both hops on channel 1 while CRP chooses the path with channels 1 and 2.

Figure 3.12 shows that the CRP route improves the TCP throughput by 108% and UDP throughput by 102% over both WCETT and ETT. Note that the performance improvement is similar for TCP and UDP since in a two-hop scenario, TCP is better able to utilize the available bandwidth.

CRP in a Large Network

We now evaluate CRP in a large network of 50 static nodes randomly placed in a 2000 m x 2000 m area. We selected 20 random non-overlapping source destination pairs in the network and initiated a TCP connection between each pair for 60 seconds (so as to observe steady state behavior). Each node had 3 radios on orthogonal channels.

The results in Figure 3.13 shows the percentage increase in TCP throughput of CRP over WCETT. We find that 7 paths have 20% or more gain in TCP throughput with the gains being as high as 90% in some cases. Typically whenever the performance of CRP and WCETT are equal, it is for two reasons: (1) The path is 3 hops or shorter in length. Since WCETT assigns channels in equal proportions, it always chooses a route similar to CRP (each link has a different channel), (2) Although rare, sometimes when the path is longer than 3 hops, by random chance the WCETT ordering of channels turns out to be optimal.

Thus, the simulation results show that gains from CRP are observed in two cases: (1) When frequency reuse is possible, and (2) When there are heterogeneous nodes in the network, i.e. not all nodes have the same number of interfaces. While these scenarios are common and important to address (typical wireless networks are bound to have some frequency reuse opportunities and networks may have all different types of nodes with different number of radios and characteristics), our testbed evaluation in the next section provides insight into other scenarios where CRP is helpful.

3.4.5 Testbed Evaluation

We now document our experience with the performance of CRP in a real wireless network deployment. We first perform a series of microbenchmarks to see its performance in known settings and then evaluate its performance as a running system.

Computational Complexity

CRP improves Dijkstra route selection at the expense of more computation and complexity. While the complexity is not a hurdle (the protocol is already implemented), it is important to find the computational complexity to ascertain the protocols applicability to real mesh networks which may have embedded devices with slow processors.

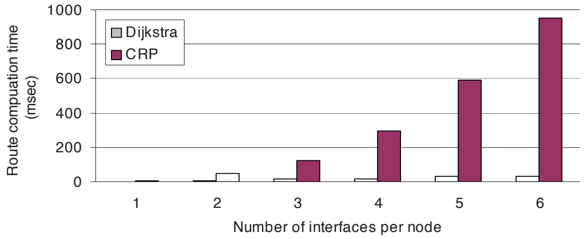


Fig. 3.14. Microbenchmark: route computation time.

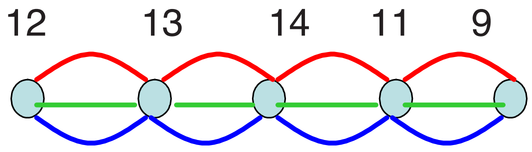
We configured our CRP implementation's graph cache with a graph topology of 100 nodes (taken from a typical neighborhood layout) and varied the number of interfaces per node. We then evaluated the time taken by CRP versus Dijkstra in computing the shortest paths to all nodes in the network. As shown in Figure 3.14, we find that our implementation, while significantly more computationally expensive than Dijkstra, can find routes in the extreme case of 100 node networks with 6 radios each in 900 ms. The current MCL implementation infact calls Dijkstra only once per second, caching routes in between. However in our current testbed with 14 nodes each with 42 radios, the computation is performed in around 10-20 ms. Thus, even if we use embedded mesh routes whose CPU speeds are 5-6 times slower than our testbed nodes, we expect to typically find routes under 100 ms. Additionally, one could proactively find such routes in the background to hide this delay as well.

Frequency Reuse Scenario

We now evaluate the real performance gains from good choice of routes in our testbed. Note that unlike the simulator which does not simulate co-channel interference[1], this gives us a better idea of the potential performance benefits.

We selected one of the 4 hops routes in our testbed between nodes 12 and 9 and performed multiple 1-minute TCP transfers taking the median performance. The

[1] QualNet physical layer code is binary only so we could not change it to simulate co-channel interference



TCP Throughput (Kbps)	ETT	WCETT	CRP
	784	1467	2342

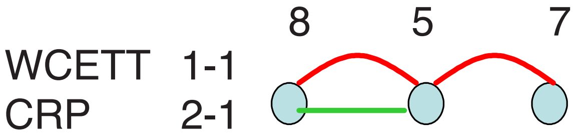
Fig. 3.15. Microbenchmark: frequency reuse.

route and channels chosen by WCETT and CRP are shown in Figure 3.15. WCETT reuses channel 2 on the last link 11→9 essentially because the channel 2 radio on node 11 had a lower ETT. Thus, WCETT took a local view not realizing this link would cause secondary interference with link 13→14. However, CRP chose the interface on channel 1 on the last link which although locally had a higher ETT, did not cause self-interference. Thus this route got a higher SIM score and was selected by CRP providing a performance gain of 60% and close to 200% with respect to WCETT and ETT respectively. This example clearly shows how using context benefit translates into real world gain.

Heterogeneous Node Scenario

We next consider the simple topology of 3 to see how much gain a CRP provides over a WCETT route in the real world. As shown in Figure 3.16, we configured node 8 and 5 with two interfaces on channel 1 and 2 while node 7 had one interface on channel 1.

WCETT chose channel 1 on both links due to the the use of Dijkstra and ETT also chose this route because the ETT of channel 2 on node 8 was around 1ms higher than channel 1. However, CRP is able to identify the correct route and chooses a route with channel 2 and channel 1. This allows CRP to have a TCP throughput



TCP Throughput (Kbps)	ETT	WCETT	CRP
	8504	8490	11135

Fig. 3.16. Microbenchmark: heterogeneous nodes.

31% higher than both WCETT and ETT. Note that the gain is lower than that observed in simulation because some co-channel interference can limit gains. From measurements, we know such interference exists despite our attempts to place the cards at some distance from each other on each machine.

Large Homogeneous Node Scenario

We now evaluate the gain from CRP in a running network topology. Each testbed node in this scenario has 3 radios. We performed multiple 1-minute TCP transfers between node 1 and all the other testbed nodes with WCETT and CRP. The % throughput gain of CRP over WCETT observed from node 1 to the other nodes (numbered on the x-axis) is shown in Figures 3.17.

The results show that CRP can improve WCETT throughput in 7 out of 13 paths by up to 100% and more than 30% on most paths. Some of the results are expected while some are non-intuitive. Among the expected results are: (1) The paths to nodes 3, 6, 4 and 11[2] have no gain since they are less than 4 hops in length and WCETT can assign costs correctly if the number of hops are less than the number of radios per node. (2) Nodes 5 and 7 are reached via 4-hop routes and the SIM metric can now choose a better path to provide gain.

[2] Node 4 can be reached with 2 hops due to it being in an open area and the waveguide effect of hallways.

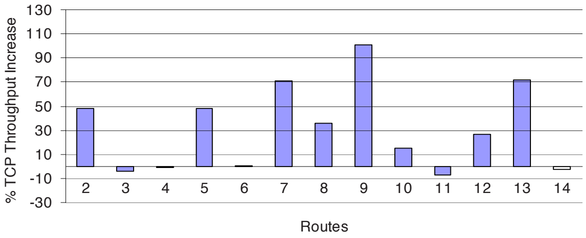


Fig. 3.17. Testbed macrobenchmark: Basic scenario. TCP throughput gain on paths from node 1 to all other nodes.

Among the unexpected results are: (1) Paths to nodes 8 and 9 have gain despite their being 3 hops away. We found this was caused by ETT variation among the interfaces of node 6 which rendered 2 interfaces almost useless due to high ETTs and effectively caused a bottleneck link which required careful route selection. Thus, random variations in link quality can create channel bottlenecks requiring the careful route selection of CRP even on short-hop paths. (2) Routes to node 10 have low gain despite reuse opportunities since the link performance is constrained by the weak link 8→10. However, going to nodes 12 and 13 through 10 gives gain since the throughput is now constrained more by ordering of channels selected than the weak link. (3) Finally, although 14 is 4 hops away, there is no gain because WCETT chooses the good route selected by CRP purely by chance.

Large Heterogeneous Node Scenario

We also evaluated the gain in a heterogeneous network scenario where all nodes do not have the same number of interfaces. We configure nodes 6, 10 and 11 to have one interface disabled. The percentage throughput gain observed from node 1 to the other nodes (numbered on the x-axis) is shown in Figure 3.18.

The results again show that CRP can provide gain in the presence of heterogeneous nodes. However, the gain from CRP in this scenario can both increase or decrease due to separate reasons. It can increase compared to the homogeneous scenario due

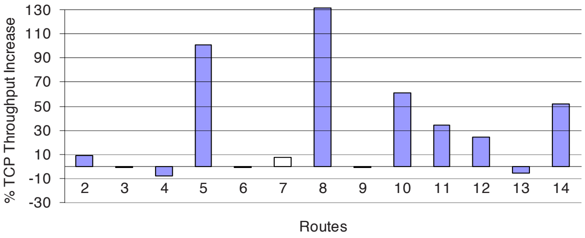


Fig. 3.18. Macrobenchmark: heterogeneous node scenario. TCP throughput gain on paths from node 1 to all other nodes.

to the increased importance of careful path selection. However, it can also reduce compared to the homogeneous scenario since we reduced the network capacity by reducing the number of radios and so some paths may not support high throughput or may not have the degree of freedom to provide an interference-free channel ordering any longer. We can see examples of both these effects. As an example of the first case, the routes to nodes 5 and 8 went through node 6 with an interface constraint and the routes to nodes 10, 11 and 14 are affected by interface constraints at nodes 10 and 11 resulting in an increased gain compared to the homogeneous scenario. However, since the route to nodes 7 and 9 did not use node 6 this time, they both reduced to a 3 hop route and there was no gain as in the homogeneous scenario. As an example of the second case, there is small or reduced gain for routes to nodes 12 and 13 since these routes contain two nodes with interface constraints making it impossible to find a non-interfering channel ordering.

Finally, some nodes close by are not affected by nodes with fewer radios and there is no difference in performance (e.g. nodes 2, 3, 4). During this experiment node 2 did not have ETT fluctuations so no gain was observed compared to the homogeneous scenario. In addition, the route to node 6, despite an interface bottleneck, was chosen correctly by WCETT purely by chance.

We performed experiments from other nodes and sources and observed a similar overall trend: around half of the paths had gain and the maximum gain observed was

152%. We chose to explain the result in more detail from a single vantage point due to lack of space.

Summary: Overall we found that CRP can improve performance over the state-of-the art approaches in a real deployment. We identified the following cases when CRP improves performance: (1) Frequency reuse is possible, i.e. the path length is such that some links in the path don't interfere. Of course this can be exploited only if enough channels are available, e.g. 3 channels in a 4 hop path. In this case the main reason for gain is the use of the SIM metric. (2) Bottlenecks in the network exist, i.e. some channels on some nodes are bottlenecked requiring *careful* route selection. These bottlenecks can occur not only in heterogeneous networks (where some nodes have interface constraints) or due to ETT variation among the interfaces of a node. In such scenarios only a context-based path selection method like CPP can effectively compute routes.

Note that we currently do not aim at solving the load balanced routing problem in handling multiple flows. We believe a better practical strategy is for link traffic to affect the link metric and then use a method, such as CRP, to find a route. Thus CRP should potentially be useful for load-balanced routing proposals.

3.5 Other Uses for Context-Based Routing

The concept of context-based routing goes beyond the application of self-interference aware routing. In this section we briefly mention some other potential applications of context-based routing.

Multi-Radio Network Coding: It is possible to combine multiple context-based metrics in systems where context is required to model multiple phenomena. As an example, we can potentially perform route selection in a wireless network with multiple radios as well as local mixing.

Multicast : While this chapter applied context-based routing to multi-radio unicast, another interesting problem we are exploring is whether context can be used to optimize multicast trees in multi-radio networks.

Optical Networks : The problem of choosing lightpaths in an WDM optical network has some synergy with a wireless network equipped with multiple radios. Each optical router has multiple incoming wavelengths and outgoing wavelength. Some costlier devices can also perform wavelength conversion. Thus, the cost of choosing an outgoing wavelength can depend on context, i.e. the incoming wavelength and the cost of conversion. It is interesting to see whether context-based path selection can potentially be applied in such networks.

3.6 Summary

In this chapter, we proposed a new routing technique that can leverage multiple radios on wireless mesh routers to improve the performance of mesh networks. We argue for a paradigm shift in the way routing metrics are designed and route selection performed in multi-radio wireless networks: we propose that using context information is the key to optimizing routes when interdependencies between links in a path exist. Our proposed context-based metric (SIM) and the associated novel context-based route selection method (CPP) jointly provide a new effective way for performing interference-aware routing in multi-radio networks. CRP can find good routes when frequency reuse is possible, heterogeneous nodes exist in the network or transient bottlenecks occur and hence is a good general purpose routing protocol for multi-radio meshes. The effectiveness of our approach is demonstrated through both simulations and a deployed implementation. While this chapter focused on a specific problem, we believe our method can be used to solve other interesting problems in wired and wireless routing.

4. PROTOCOL AND ALGORITHM FOR LEVERAGING DIRECTIONAL ANTENNAS

4.1 Introduction

There have been several significant efforts on improving the throughput of wireless mesh networks [29, 78]. These works aim to improve throughput using multiple radios that utilize multiple channels (available in the IEEE 802.11a/b/g standards) to separate the contending transmissions in the frequency domain. However, these works assume the use of omnidirectional antennas at mesh routers, where a transmission on a given channel requires all other nodes in range to remain silent or use alternative channels. Thus, although multiple channels can separate the transmissions in the frequency domain, the extent of such separation is potentially limited by the number of available channels: (1) There exists a bound on the number of channels possible due to spectrum regulation. For example, 802.11a has 12 while 802.11b has only 3 non-overlapping channels. (2) Further, the number of available channels could potentially be even lower due to stricter spectrum regulation in some countries or channels being set aside to support other communication (e.g., among mesh routers, users and mesh routers, or other networks). (3) Finally, significant co-channel interference exists even among the non-overlapping channels [29] which limits the extent of frequency separation among contending transmissions. Due to these limitations, there is a need to find other means of separating contending transmissions to improve the throughput of WMNs.

Compared to omni antennas, directional antennas offer *spatial separation* between contending transmissions and have the potential to further enhance the throughput of WMNs. In this chapter, we propose DMesh, a WMN architecture that combines spatial separation from using directional antennas with frequency separation from using orthogonal channels to improve the throughput of WMNs. An important requirement for DMesh is to accomplish this throughput improvement without inhibiting the other two key WMN requirements: cost-effectiveness and ease of deployability. The high cost of *smart* beamforming directional antennas and their form factor make it difficult to achieve these two requirements. Thus, in DMesh, we focus our effort on incorporating *practical* directional antennas that are widely and cheaply available (e.g. patch and yagi). The key challenge in DMesh is to exploit spatial separation from such practical directional antennas despite their lack of electronic steerability and interference nulling as well as the presence of significant sidelobes and backlobes. Further, enabling DMesh poses new design challenges, as it requires specialized protocols to enable routing, and channel assignment to exploit directionality. Central to our architecture is a distributed, directional channel assignment algorithm for mesh routers that exploits directional antennas and multiple channels to separate the contending transmissions in both *spatial* and *frequency* domains. Effectively, DMesh enables two degrees of separation between contending transmissions.

This chapter makes the following contributions: (1) We propose and evaluate a WMN architecture that exploits both directional antennas for spatial separation and multiple orthogonal channels for frequency separation to provide significantly increased throughput; (2) We focus on cost-effective and deployable techniques that can easily be incorporated to extend current single-interface networks such as Roofnet [4]; (3) We describe and evaluate a distributed routing protocol along with associated directional channel assignment algorithms to fully exploit the proposed architecture, using a realistic and detailed antenna model; (4) We evaluate the proposed architecture and compare it to solutions that use omnidirectional antennas in a detailed simulator; (5) We validate the performance of our architecture using experiments in our mesh network testbed with the same antennas modeled in the simulations.

Experimental results show that compared to a multi-radio multi-channel network using omni antennas, a DMesh network increases the average per source throughput from 128% for 25 traffic sources up to 231% for 50 traffic sources and provides up to 10 times smaller packet delays in an 802.11a network with 50 mesh routers. The throughput improvement from DMesh is more pronounced when the number of available channels is limited. When only 6 instead of 12 channels in an 802.11a network are used, the throughput using DMesh is improved by up to 176% for 25 traffic sources with significantly lower delay in comparison to a multi-channel omni network. DMesh also provides TCP throughput gains between 30% and 68% in 802.11a and 35% in 802.11b networks. We implemented DMesh and deployed it on a subset (outdoor portion) of the MAP testbed using commercially available practical directional antennas. Our testbed evaluation shows that DMesh increases TCP throughput in 802.11b WMNs between 31% and 57% compared to a multi-channel omni WMN.

The rest of the chapter is organized as follows. In Section 4.2, we describe DMesh, our architecture for a wireless mesh network with directional antennas and multiple channels. Our evaluation methodology and results are detailed in Sections 4.3 and 4.4. An evaluation of the real world performance of DMesh in a testbed is described in Section 4.5. Section 4.6 summarizes the related work and Section 4.7 concludes the chapter.

4.2 DMesh: A Directional Wireless Mesh Network

In this section, we first describe the architecture of DMesh followed by descriptions of the associated physical tree formation, routing protocol and distributed directional channel assignment.

We consider a typical single-channel, single-interface WMN deployment with omnidirectional antennas in which mesh routers are placed on the rooftops of subscribers [4] or other infrastructure (e.g. streetlights) and are interconnected via 802.11 links. DMesh is then used to enhance the performance of this existing network. To enable DMesh, we assume that each mesh router can have up to k additional interfaces (multiple radios) each with its own practical directional antenna. The practical directional antennas used in DMesh are non-steerable and they always point to the direction toward which they were manually placed during the network deployment. Since mesh routers are not highly form factor-constrained like other wireless devices (e.g. PDAs), the use of multiple radios and antennas is feasible. In fact, multi-radio mesh routers are already commercially available [10]. Given this DMesh architecture, the omnidirectional interface of each mesh router is always available for providing robust connectivity if the directional neighbors fail. Further, this omni interface is also used as a CONTROL interface.

The common application scenario for WMNs is likely to be Internet access in which most traffic will flow to and from the gateway nodes which have wired connections to the Internet. These gateways will typically be deployed sparsely in the network due to their higher cost. The number and placement of these gateways are likely to be based on the area of deployment and availability of resources. Alternatively, in some scenarios it may be possible to place such gateways intelligently [16].

To support the application scenario described above, we propose to build high throughput routing trees rooted at the gateways. DMesh consists of three main stages to construct such trees: (1) physical tree formation to provide gateway connectivity using the directional antennas; (2) routing state creation and maintenance to correctly deliver packets along the tree; and (3) channel assignment to separate spatially contending transmissions whenever possible to further increase the throughput.

OLSR [20] is used as the multi-hop routing protocol in our single-radio single-channel omnidirectional mesh network testbed (described in Section 4.5). DMesh extends the OLSR protocol to aid in physical formation of trees using practical directional antennas, set up and maintain corresponding routing state, and perform channel assignment. We call this extended protocol DOLSR (*Directional OLSR*).

4.2.1 Physical Tree Formation

The physical tree formation takes place as follows. When each mesh node boots up, the DOLSR daemon reads the configuration file (*olsr.conf*) to identify the interface attached to an omni antenna and starts running in the single-channel single-interface mode. The configuration file contains a list of interfaces, along with the type of antenna attached to the interface as well as a PIP (pointing IP address) for each interface. The use of PIP is explained later in this section. In this mode, the node starts listening for HNA (*Host and Network Association*) messages on its omni interface. Nodes that have Internet access (i.e. a default route to a gateway or the gateway itself) advertise their connectivity over their omni interface using periodic HNA packets. The joining node picks the best next hop node based on metrics measured to nodes from which it receives HNA messages. The metric could be of various types, for example, hop count, remaining path bandwidth or a measurement based metric (e.g. ETX [21]). After the new node joins, it continues to receive HNA messages to reevaluate its choice of a next hop.

To incorporate directional antennas into this existing network, we modified the HNA message to also advertise whether a node is willing to host directional interfaces for its neighbors. A node can host directional interfaces after it itself has Internet connectivity through a directional interface or if it is the gateway node. In addition, the node should have free interfaces to connect to new children. For example, the gateway node bootstraps DMesh by advertising its willingness to host directional interfaces in its HNA packets. The one-hop neighbors of the gateway receive these HNA packets.

When DOLSR receives HNA packets on the omni interface with willingness indicated, it also evaluates the best potential *directional* next hop node (PARENT) using a suitable metric. For example, throughput measurements using the omni interface can be used to drive the PARENT selection. This choice of PARENT is then reported, directional antennas on the PARENT node and CHILD node (joining node) are installed pointing at each other, and corresponding entries in the *olsr.conf* on the PARENT and CHILD nodes are modified by offline means to include information about the new directional interface. Specifically, the PARENT node marks the interface as *DIR* and stores the IP address of the CHILD node's interface it is pointing to and vice-versa. This IP address is referred to as the PIP and only packets received from the PIP configured for a particular interface are passed to the DOLSR protocol. The PIP is required to filter out packets received from the sidelobes from other nodes that the antenna is not pointing at. Directional interfaces are initially configured to operate on a default channel to enable connectivity. Note that in a cooperative network such as RoofNet, a CHILD could purchase antennas for its PARENT node that is willing to host an interface. On the other hand, in a commercial WMN, the ISP takes care of the installation.

After installation is complete and the updated configuration file is read in by the DOLSR daemon, the PARENT node sends out READY packets on the newly installed directional interface. The READY message acts as a trigger for the routing protocol on the CHILD node to set up the routing state (explained in the next section). Once the routing state is set up, the CHILD node also starts indicating willingness to host directional interfaces and more nodes can now join the tree. In this way, nodes install at least one directional antenna on one of their interfaces pointing to a PARENT node, and a tree will be physically formed toward the gateway. In this chapter, we evaluate our architecture over many different topologies generated by simulating the above tree formation model.

4.2.2 Routing Protocol

The *physical* placement of directional antennas in DMesh naturally forms a tree structure. On top of this physical tree, DOLSR now sets up forwarding entries to route packets along the tree. The routing state is set up as follows. Let T_a be any PARENT node. As mentioned earlier, T_a periodically broadcasts READY messages on its newly installed directional interface. Only the corresponding CHILD node will receive the READY message due to filtering based on the PIP. On receiving READY messages from T_a, the CHILD node sends a JOIN message to T_a. T_a uses the IP address received from the JOIN packet to set up a forwarding entry toward the joining node. The joining node uses the PARENT's IP address (from the READY message) as a default route and inserts this information into its routing table. Once this exchange is completed, the joining node becomes part of the tree and sends its own HNA packets on the omni interface and can potentially become a PARENT for other nodes. Note that nodes that have the gateway as a PARENT node are denoted SUPERPARENTs.

Once the JOIN packet is received, a ROUTE_SETUP message initiated by T_a is multicast along the tree simply by having each node recursively send the message to each of their active interfaces. The multicast proceeds until the gateway is reached since the gateway does not rebroadcast the message to the other SUPERPARENTs. The ROUTE_SETUP packet contains the IP address of the interface that the joining node used in the JOIN packet. This information is used to set up forwarding table entries in the routing tables of all nodes in the subtree including the gateway, to reach the newly joined node. This information also enables peer-to-peer routing (between mesh routers): a mesh router can route to all mesh routers in its own subtree rooted at its SUPERPARENT, and it can route to the mesh routers that are offsprings of other SUPERPARENTs by going through the gateway. A node only needs to consult its forwarding table to determine the next hop for any packet. A packet that is not matched with any forwarding entry is sent on the default path to the gateway, i.e. to the node's PARENT.

Switching Parents In certain situations, a node may be required to switch parents, e.g. if a better next-hop is detected through metric measurements or when a new gateway is installed. For example, when a new gateway is installed, each gateway starts its own HNA, and to the mesh routers, this is treated like any other HNA: if the metric advertised from a different gateway is lower than the one the node is currently using, the node will find that to be a better choice. When a node changes its PARENT[1], it multicasts a LEAVE message similar to the ROUTE_SETUP message to invalidate old routing entries. When a node changes its PARENT, the subtree

[1] This is done by manual realignment of the antenna and software configuration changes

rooted at that node also moves to the new parent. The new PARENT multicasts a ROUTE_SETUP message to refresh the routing entries at all nodes in the subtree of its SUPERPARENT. In this manner, the routing state is maintained despite physical topology changes. For simplicity of evaluation, in this chapter, we consider only a single gateway.

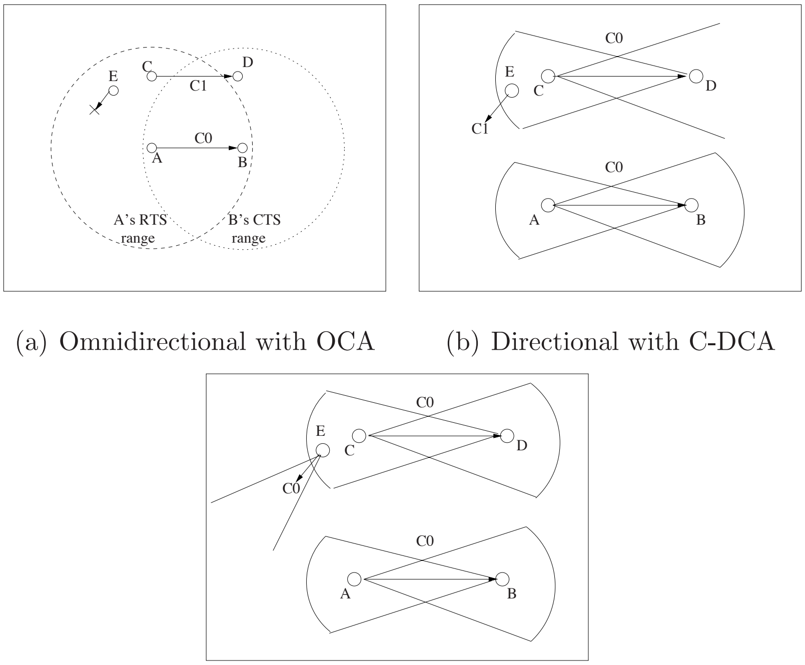
An internal node reattaching to a new PARENT can cause cycles to form. To eliminate this, we restrict the choice of the new PARENT to a node that is not part of a node's own subtree as follows: every node keeps track of nodes in the subtree rooted at itself and does not choose any of these nodes as parents, and this is accomplished by each node storing a list of nodes that sent ROUTE_SETUP messages through it. Also, in our architecture, multi-homing or multiple parents are not supported and each node has only one PARENT.

Failure Handling In DMesh, issues such as simultaneous joining and oscillations are rare since only a designated node (with matching PIP) can attach to an interface. If a node does not receive any READY messages from its directional interfaces, it defaults to using the omnidirectional interface which exists as a backup for robust connectivity.

In summary, DOLSR aids in the formation of the physical DMesh tree as well as sets up and maintains routing state in the presence of physical topology changes.

4.2.3 Distributed Directional Channel Assignment

Once the routing state is set up, one degree of separation has been achieved since more parallel transmissions can now occur in the network due to spatial separation. However, practical directional antennas cannot provide complete spatial isolation due to the presence of *sidelobes* and *backlobes*. Therefore, DMesh also uses directional channel assignment to achieve the second degree of separation by separating these spatially contending transmissions in the frequency domain to further increase the throughput of the mesh network.



(a) Omnidirectional with OCA

(b) Directional with C-DCA

(c) Directional with A-DCA

Fig. 4.1. Channel assignment schemes.

The directional channel assignment is performed by a PARENT for its CHILD node when the node joins. As mentioned earlier, both interfaces are initially configured on the same default channel to enable connectivity. The PARENT selects a new channel using a channel assignment scheme and sends an ASSIGN message to the CHILD. Both PARENT and CHILD then use *iwconfig* to set their respective directional interfaces into the selected channel. Following this, bidirectional communication can occur. The channel assignment is periodically re-evaluated every 300 seconds.

Channel Assignment Schemes

Our objective is to design a distributed directional channel assignment algorithm that provides good performance and is easy to implement and deploy. In the following, we discuss our two omnidirectional and four directional channel assignment schemes using an example. Consider the scenario in Figure 4.1(a) where there are five mesh routers (A-E) and two flows A→B and C→D. Assume that E has a flow terminating at some other node in the direction indicated.

Omni / No Channel Assignment (ONOCA) In this scheme, omnidirectional antennas are used without multi-channel techniques. If A transmits to B omnidirectionally, then the channel is reserved in the entire region due to the RTS and CTS range of A and B as shown in Figure 4.1(a). Thus, only one flow can proceed. In this network there is no spatial or frequency separation possible between contending transmissions in a single collision domain.

Omni / Channel Assignment (OCA) In this scheme, omnidirectional antennas are used with multi-channel techniques. Each node first attempts to pick a unique unused channel for communicating to its PARENT. If the node cannot obtain an unused channel, it reuses the least loaded channels among those that have been selected by nodes in its interference range. This algorithm is similar to the one described in a previous work [78] and is implemented in a distributed manner by periodic exchange of channel usage information and least loaded channel selection. A least loaded channel is defined to be the one being used by the lowest number of flows. If the number of flows using a pair of channels are the same, then the traffic transmitted over the channels over the last information exchange period is used to determine the least loaded channel. Thus, OCA exploits the frequency separation offered by multiple channels to allow contending transmissions in the same collision domain. If we apply this to Figure 4.1(a), assuming 2 channels exist in the network, both flows A→B and C→D can proceed using different channels C_0 and C_1. However, E still cannot transmit, since it is in the contention range of the other two flows. E realizes this through exchange of channel usage information and picks the least loaded channel in its vicinity.

Directional / No Channel Assignment (DNOCA) In this scheme, directional antennas are used without multi-channel techniques. Under this scheme, flows A→B and C→D can proceed since their *cone of interference* (the area defined by the sector of the transmitting nodes) is separated spatially. However, E cannot transmit since it lies in the cone of interference of flow C→D that uses the only available channel C_0.

Directional / Channel Assignment (DCA) In DCA, directional antennas are used with multi-channel techniques, and we aim to separate spatially (directionally) contending transmissions in the frequency domain. Two transmissions are defined to be *directionally* contending if either one of the nodes originating the transmission lies in the cone of interference of the other.

We propose and evaluate three directional channel assignment schemes. The first scheme is *Conservative DCA* (C-DCA). In C-DCA, given a set of channels C, a node X assigns a channel to a CHILD in two stages: (1) It first attempts to find a free channel C_i. A free channel for node X is defined as one that is not used by any node whose cone of interference contains X or its CHILD, i.e., no node in the vicinity of X and its CHILD is directionally contending on channel C_i. (2) If no such channel exists, the node X selects a channel C_i that is least loaded. If we use C-DCA in Figure 4.1(b), E will select to transmit using C_1, effectively insulating its transmission from its directionally contending transmissions. C-DCA is conservative because it always assigns a new channel to a transmission that lies in the cone of interference of another transmission. The C-DCA heuristic requires exchange of channel usage information in a local area to allow for selection of channels. For example, node E needs to receive channel usage information from C and D and perform geometric calculations to realize that it is in the cone of interference of D while executing the C-DCA algorithm. Calculating the cone of interference requires knowledge of the positions of C and D which is easily available in a static mesh network (e.g. through a localization scheme or GPS). In a mesh network with no mobility, it is even relatively simple to just encode the location of the mesh router when deploying or installing it, and no GPS hardware needs to be permanently installed in the mesh router. We use this one-time encoding method in our testbed. Other deployed testbeds such as RoofNet also maintain the GPS coordinates of their mesh routers.

Note that C-DCA effectively takes into account the interference from sidelobes on a receiving node's antenna by assuming that the antenna can potentially receive interference from any direction (not just from the main lobe). In other words, C-DCA assumes the possibility of interference if a node lies in the transmit cone of neighboring nodes regardless of the orientation of the node's own antenna. For example, E could potentially use C_0 if its flow is not directed toward C or D. However, this would make E prone to interference through power received on its sidelobes on C_0. C-DCA does not allow the selection of C_0 to avoid such interference from sidelobes.

In addition, C-DCA takes into account the interference from the transmission power from a neighboring node antenna's sidelobes. Since C-DCA assumes a *flat-topped* transmit cone to make the geometric calculations simpler and applicable to any antenna, it can only anticipate interference caused by transmission through a main lobe of a neighboring node's antenna. To take into account the power from the sidelobes of a neighboring node's antenna, we incorporate measurement-based enhancements in the channel assignment as follows: Each node overhears transmissions on all its directional interfaces and reception of data on an interface from any node whose IP address is different from the PIP for that interface indicates interference on the channel being used on that interface.

Our second assignment scheme, *Aggressive DCA* (A-DCA), assigns channels similar to C-DCA but with one important difference: X is considered to be in the cone of interference of another node Y, only if both X and Y lie in each other's cone of interference. Thus, in Figure 4.1(b), if E transmits in the direction shown, although E lies in the cone of interference of D, D does not lie in the cone of interference of E, and thus node E using A-DCA will reuse channel C_0. Thus, A-DCA is more aggressive in identifying opportunities where channels can be reused based on directionality and thus results in a reduced channel usage. This can result in more interference (e.g. if E has sidelobes it will receive interference power from D). To accommodate interference from other transmissions, A-DCA uses a "guard angle" to control the aggressiveness of channel assignment as follows: the beamwidth of all transmit and receive cones

Table 4.1
Architecture choices in a multi-radio mesh network

Antenna	Channel Assignment
Omni	ONOCA, OCA
Directional	DNOCA, C-DCA, A-DCA, M-DCA

during the geometric interference calculations are increased by guard angle degrees. Thus, a higher guard angle makes A-DCA more conservative when reusing channels while a lower guard angle makes A-DCA aggressive in reusing channels. Since the cone of interference is larger, nodes close to the transmission are likely to be separated in frequency. A typical guard angle value is 30°. Note that A-DCA does not use measurement-based enhancements, since that would limit its aggressiveness.

Finally, we evaluate a third assignment scheme, *Measurement-based DCA* (M-DCA), which uses only measurements instead of geometry to infer interference between two nodes. M-DCA works similarly to the measurement-based enhancement in C-DCA, where each node overhears transmissions in all its directional interfaces, and uses this information to discover interfering nodes.

Table 4.2.3 summarizes the various WMN architecture choices considered in this chapter. Using omnidirectional antennas, we can apply OCA and ONOCA schemes, whereas using directional antennas we can apply DNOCA, A-DCA, M-DCA and C-DCA.

Distributed Algorithm for OCA/C-DCA/A-DCA/M-DCA

In order to select a channel in a distributed manner using local knowledge, a node needs to know the channel usage in its vicinity. This channel usage information is locally exchanged since the channel can be reused at a sufficient distance. Thus each node periodically (every 60 seconds) uses the omni CONTROL interface to broadcast its own channel usage information. The broadcast uses sequence numbers to suppress duplicates and is rebroadcasted for two hops in order to cover the interference range of that node which is assumed to be twice that of the communication range. Note that in DMesh, the reachability of any node using the directional antennas is kept similar to the reachability achieved if the node were using an omnidirectional antenna by adjusting the transmission power on the directional interfaces. Also, the directional interfaces using simple non-steerable antennas cannot be used to broadcast channel usage information since they will not cover all possible nodes that need to receive this information.

In OCA/C-DCA/A-DCA, each node broadcasts a channel vector C that specifies whether it is using each channel C_i, and a rate vector R that specifies a time average of the amount of traffic it has transmitted on each channel C_i over the past 60 seconds. For A-DCA/C-DCA, nodes initially exchange their position information and cache it for future use since the mesh routers are static. In our testbed, this position information is encoded using GPS during installation. However, other localization techniques could be easily used for this purpose as well. For A-DCA/C-DCA, a destination vector D indicating the destination being communicated with on each channel C_i is included for directionality information. The final channel map maintained at every node is constructed from the individual state vectors ($< C, R, D >$) received from neighboring nodes. In addition, each CHILD also sends its PARENT a list of channels (BCL) computed from its own channel map which contains all channels being used in the CHILD's neighborhood.

Once a node collects the channel usage information from its interfering neighbors (which may be more than 1 hop away), it uses the assignment algorithms to assign channels to a new CHILD or periodically (every 300 seconds) reevaluate the channel assignments for its current children.

In summary, for all the schemes[2], each node obtains a channel usage map which stores a list of node identifiers along with their associated channel and rate vectors. For A-DCA and C-DCA, a destination vector is also obtained. For M-DCA, there is

[2] The pseudo-code for A-DCA, C-DCA and OCA are available in [85].

no need of exchanging information between nodes. Each node monitors each of its directional interfaces for arrival of data from any node other than the child assigned to it. Reception of data from a non-child node indicates interference on the channel being used on that interface. This interfering node is then added to the channel usage map.

4.3 Experimental Methodology

We use the QualNet simulator [74] to evaluate DMesh. QualNet has been widely used to study directional antennas and multi-channel networks [9, 18, 89]. We model a node with multiple interfaces with support for dynamic channel assignment on each interface.

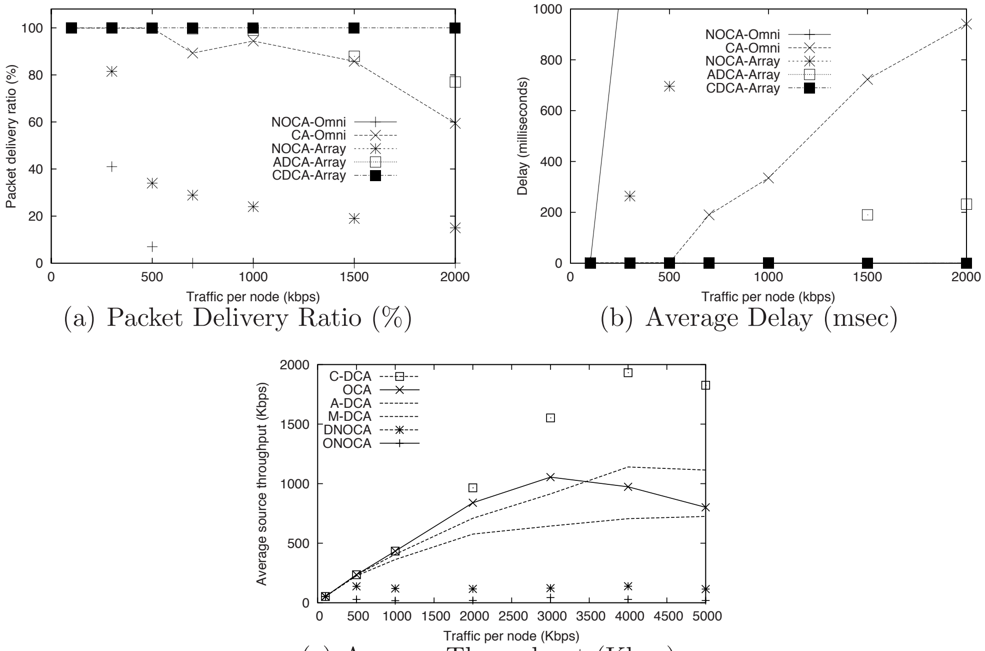
MAC, Physical Layer and Antennas We used IEEE 802.11a and 802.11b with autorate fallback. The simulator models OFDM and DSSS for 802.11a and 802.11b, respectively. Both MAC layers were verified to produce close to theoretically maximum throughput [41] and also model multi-rate operation based on inter-node distances. The two-ray path loss propagation model is used. We used real patterns for the directional antennas that faithfully model sidelobes. The patterns were taken from datasheets of commercially and cheaply available directional antennas from [36] and [59]. In fact, our testbed evaluation uses one of the antennas we model in the simulation.

Topology and Traffic We simulated a static mesh network of 50 nodes placed randomly in an area of 1000m x 1000m. Each node is assumed to have 1 omni interface, 1 directional interface to connect to a PARENT and 2 additional directional interfaces to support up to 2 children. The directional interfaces are pointed to neighbors with a random error between 0-10° to simulate manual placement of antennas. The gateway is assumed to be at the center of the area. In order to isolate the effect of increased spatial reuse obtained from directional antennas from the range enhancement that also results, the reachability of any node using the directional antenna is kept similar to the reachability achieved if the node were using an omnidirectional antenna by adjusting the transmission power. We vary the number of sources and evaluate using both UDP and TCP traffic. All the sources communicate with the gateway node to simulate an Internet access pattern. The rate for each source is picked randomly from 0 to $Rate_{max}$. $Rate_{max}$ is varied from 100 Kbps to 5 Mbps. A packet size of 1500 bytes is chosen.

Metrics We use the following three metrics to evaluate the performance: (1) *Packet delivery ratio* (PDR) — the ratio of the overall number of successfully received data packets at the gateway to the number of data packets sent; (2) *Delay* — the average time between transmission and reception of data packets. This metric accounts for all possible delays caused by queuing at the interface queue, retransmission delays at the MAC, and propagation and transfer times; (3) *Average source throughput* — the total number of bytes successfully transferred from each source during the simulation duration, divided by the simulation duration and averaged over all sources. This metric is similar to PDR, except it also gives an indication of how much raw bandwidth is achieved. For all schemes, control overhead occurs only in exchange of channel usage information, and is negligible compared to the overall data traffic. Therefore, in the results below, due to lack of space, we omit the control overhead for all schemes.

4.4 Performance Evaluation

In this section, we first evaluate the performance of various WMN architectures. We then perform controlled experiments by varying other parameters such as the number of available channels to study the tradeoffs involved in architecting a practical DMesh network. Before studying the various architectures, we evaluated several tree construction metrics (details in [85]) and chose the best tree construction metric found for use in all architectures studied. Specifically, an ETH (expected throughput) metric, in which a node probes the throughput (e.g. using the *netperf* tool) to its



(a) Packet Delivery Ratio (%)

(b) Average Delay (msec)

(c) Average Throughput (Kbps)

Fig. 4.2. Comparison of different mesh network architectures.

potential PARENTs (from whom it receives an HNA) and selects the PARENT with the highest throughput, was found to provide good performance through simple local measurements. We use this metric in our evaluation.

4.4.1 Overall Performance Comparison

In this section, we evaluate the performance gain from the different WMN architectures listed in Table 4.2.3. The beamwidth of the directional antenna is assumed to be 45°.

Figure 4.2 shows the performance of all the above mesh network architectures in terms of the packet delivery ratio, delay, and average source throughput. As expected, the PDR, delay and throughput of ONOCA are the worst among all architectures. This is because ONOCA does not exploit any degrees of separation and is thus unable to support the simultaneous traffic load in the network.

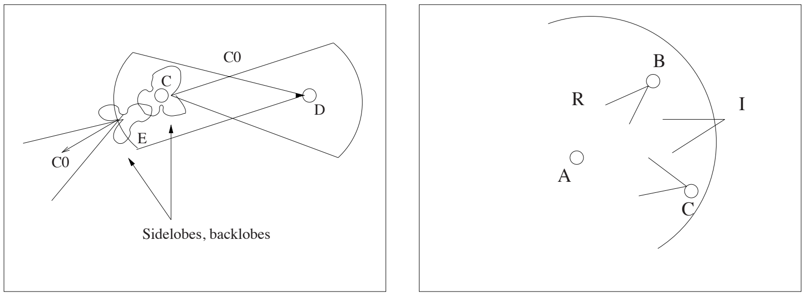
DNOCA achieves better performance than ONOCA. DNOCA exploits spatial separation to support simultaneous transmissions and thus achieves higher throughput than ONOCA. At 500 Kbps traffic, the PDR of DNOCA is 48% better than that of ONOCA. However, spatial separation *alone* is not sufficient to support high throughput. The PDR of DNOCA drops drastically as traffic increases beyond 500 Kbps. This shows that practical directional antennas cannot be effectively used as complete spatial isolators. Two factors contribute to this poor performance: (1) The beamwidth of a practical directional antenna cannot be made arbitrarily small in order to completely separate all contending transmissions. (2) The presence of sidelobes limits the usability of multiple practical directional antennas at a node since one antenna will be able to hear the other due to reception from sidelobes. Thus, any pair of interfaces on the same node will always contend with each other. We also observed the sidelobe problem in our DMesh testbed and found that the problem was not mitigated even with greater physical separation between the antennas of a pair of interfaces.

OCA also achieves better performance than ONOCA. OCA exploits frequency separation to enable higher throughput than ONOCA. At 500 Kbps traffic, OCA achieves 88% higher PDR than ONOCA. However, separation by frequency alone is not sufficient for achieving the best performance. The PDR of OCA drops to only 40% and the delay drastically worsens as the traffic increases to 5 Mbps. An important factor contributing to this degradation is that there is an upper bound on the number of distinct orthogonal channels available which limits the ability to completely isolate all contending transmissions by frequency. For example, an examination of the channel assignment in this scenario shows that OCA shares many channels among multiple flows. Moreover, as the traffic increases, the per channel load increases, thereby increasing the contention among flows and thus the throughput drops. Thus, although OCA delivers a fraction of the packets, these packets are delivered after significant amount of queuing and buffering leading to high average delays. Note that another significant factor that could restrict the performance of OCA is inter-channel interference. Although the simulation evaluation of OCA assumes no inter-channel interference[3], in our DMesh testbed, we observed that two interfaces maximally separated by orthogonal channels fail to achieve simultaneous throughput. In summary, the observed performance of both DNOCA and OCA suggests that an additional degree of separation can further improve the throughput.

Another interesting observation from the above experiments is that OCA significantly outperforms DNOCA. Although both schemes have a single degree of separation, separation by frequency alone is more effective than separation by space alone from using practical directional antennas. This effect was also observed during our testbed experiments. This is primarily because any pair of interfaces on a single node, when connected to practical directional antennas can always hear each other (due to sidelobes and backlobes) and it is difficult to separate them physically far enough apart from each other to obtain simultaneous throughput. On the other hand, any pair of interfaces on a single node separated by frequency try to filter out each others signal. Although this filtering is also not perfect, the separation achieved between contending signals from this technique is superior to the separation achieved using a pair of practical directional antennas.

C-DCA fully exploits two degrees of separation offered by directional antennas and multiple channels to maximally isolate contending transmissions. The PDR and throughput of C-DCA are the highest and the delay the lowest among all schemes across all traffic scenarios. In particular, the throughput gains achieved in comparison to OCA are up to 128% with up to 10 times lower delay. On one hand, C-DCA avoids the sidelobe problems with multiple interfaces in DNOCA through frequency separation. On the other hand, C-DCA reduces the high channel load problem in OCA through spatial separation. Finally, C-DCA can account for interference from sidelobes of nearby transmitting antennas through measurements.

[3] Qualnet does not simulate inter-channel interference and the code where this feature can be implemented is not open-source.



(a) A-DCA sidelobe interference (b) M-DCA estimation problem

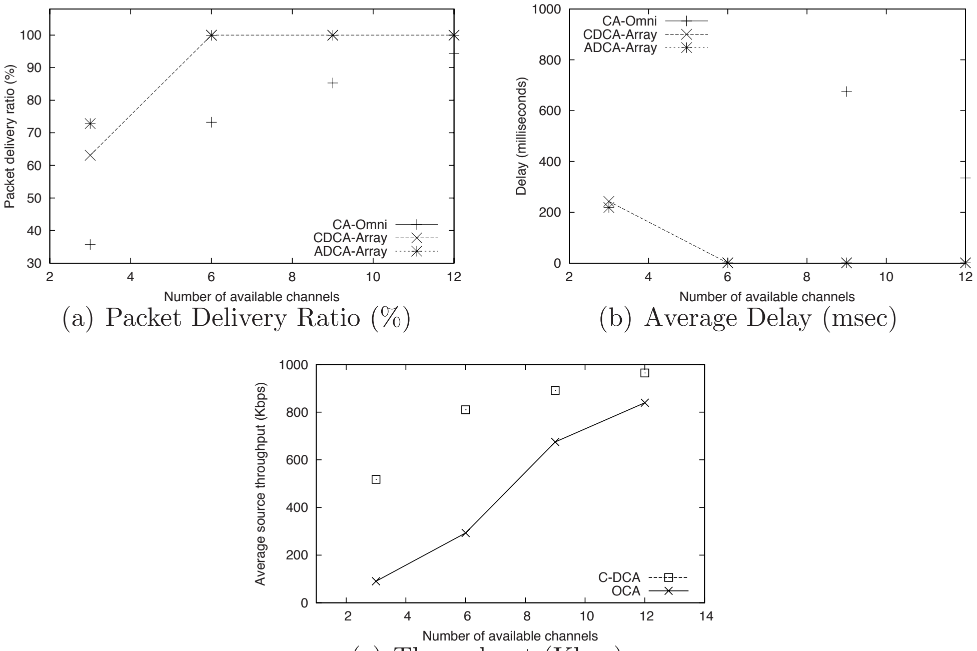
Fig. 4.3. Channel assignment issues in A-DCA and M-DCA.

As described in Section 4.2.3, C-DCA is conservative in reusing channels. In contrast, the A-DCA algorithm aggressively exploits spatial separation to reuse channels which can potentially reduce the channel usage. However, the results show that A-DCA does not offer significant benefits. The performance of A-DCA is always lower than that of C-DCA and in fact comparable to that of OCA. In fact, A-DCA outperforms OCA only when the traffic exceeds 3Mbps when the average channel load in OCA increases significantly. The performance degradation of A-DCA is because A-DCA assumes a flat-topped pattern during geometrical calculations in channel assignment. This pattern has only one main lobe, and the power is kept very low for all other angles outside the main lobe. Since our simulations use real antenna patterns (i.e. with sidelobes and backlobes causing interference), while A-DCA optimistically assumes no interference from sidelobes in order to maximize channel reuse, it does not always assign non-interfering channels correctly. An example of this is shown in Figure 4.3(a) in which node C using A-DCA reuses channel C_0 to transmit to node D since the nearby node E is directed away from C. However, sidelobes and backlobes in the antennas on C and E using the same channel C_0 can cause interference, degrading the throughput. We also performed simulations varying the guard angle in A-DCA to tune its aggressiveness. The results, reported in detail in [85], show that highly aggressive channel reuse exploiting fine-grained spatial separation is not feasible with practical directional antennas. The effectiveness of A-DCA with smart antennas that can tightly control the sidelobes is a focus of our future work.

Finally, to evaluate the accuracy of geometrical techniques in C-DCA/A-DCA, we compared them with a pure measurement-based channel assignment scheme, M-DCA. M-DCA accounts for sidelobes since it uses real measurements to infer interference between two nodes, instead of geometry. Surprisingly, Figure 4.2 shows that M-DCA achieves worse performance than A-DCA and OCA. The reason for this is that in M-DCA, each node checks if it interferes only with nodes from which it can *receive* packets, that is only from nodes in its communication range, and not in its range of interference. However, even nodes in twice the communication range may interfere because of carrier sensing. For example, in Figure 4.3(b), node A can assign channels to reduce interference from nodes B and C within its communication range R. However, node A cannot identify node I as an interferer (due to it not being able to receive packets from I) and is likely to assign the same channel node I is using, resulting in interference. Such inaccuracies in channel assignment significantly degrade the performance of M-DCA. Note that such far away nodes could be identified using using two-hop packet exchanges and interference estimated using geometric calculations. However, the objective of M-DCA is to study the effectiveness of a pure measurement-based algorithm.

In summary, exploiting two degrees of separation by integrating practical directional antennas with channel assignment significantly improves the performance of WMNs. Further, a combination of measurement-based and geometric techniques to assess interference for channel assignment, as in C-DCA, provides the best performance.

Finally, we also compared how the algorithms fare as the traffic load is increased by increasing the number of traffic sources from 25 to 50. In this case, we found that the gap between C-DCA and OCA becomes even larger. For example, when the maximum traffic is 2Mbps, the difference between the throughput of the two schemes is about 15% with 25 sources, but increases to 231% when all 50 nodes are sources.



(a) Packet Delivery Ratio (%)

(b) Average Delay (msec)

(c) Throughput (Kbps)

Fig. 4.4. Impact of a finite number of available channels. Maximum traffic per node is 3 Mbps. Beamwidth used is 45 °.

This is because when the number of sources increases, the total traffic in the network increases, and the channels become loaded faster. Hence, OCA has to reuse loaded channels, and this results in performance degradation. On the other hand, C-DCA can still exploit the second degree of separation, achieving higher throughput.

4.4.2 Impact of Available Physical Channels

In this section, we study the impact of the number of available channels on the performance of WMNs. As the number of available channels is reduced, both OCA and C-DCA have to resort to reusing channels. This increases the average contention per channel.

Figure 4.4 depicts the performance of C-DCA and OCA, as the number of available channels is varied from 3 to 12. Two observations can be made from the results. First, both OCA and C-DCA experience performance degradation as the number of available channels are reduced, since both OCA and C-DCA have to resort to reusing channels. This increases the average contention per channel and worsens performance. Second, C-DCA performance is less affected than OCA as the available channels are reduced. This is because OCA performance is directly dependent on the number of channels as it has only one degree of separation between contending transmissions. In contrast, C-DCA requires less than 12 channels (11 channels on average) to completely isolate contending flows in most of the scenarios we studied. Even when C-DCA has to reuse channels, the average load per channel is also lower. Thus, in comparison to the 12 channel case, the gains of C-DCA against OCA in PDR, throughput, and delay increase to 173%, 176% and 676%, respectively, when the number of channels is reduced to 6. In summary, C-DCA is more effective in utilizing scarce spectrum resources than OCA.

4.4.3 Additional Results

Apart from the evaluation in this section, we studied additional issues in architecting DMesh which are detailed in [85]. We studied the impact of the antenna beamwidth and found that lower beamwidth antennas (typically larger in size) provide better performance as the number of available channels reduces or the density of nodes in a collision domain increases. We also compared C-DCA and OCA in 802.11b networks and found that C-DCA outperforms OCA by about 80%. Finally, we also studied TCP performance and found that DMesh provides TCP throughput gains between 30% and 68% in 802.11a and 35% in 802.11b networks.

4.5 Testbed Evaluation

In order to verify the usefulness and performance of the DMesh architecture, we incorporated practical directional antennas in our MAP (Mesh@Purdue) [111] testbed. We use C-DCA as the channel assignment algorithm for DMesh. We also compare

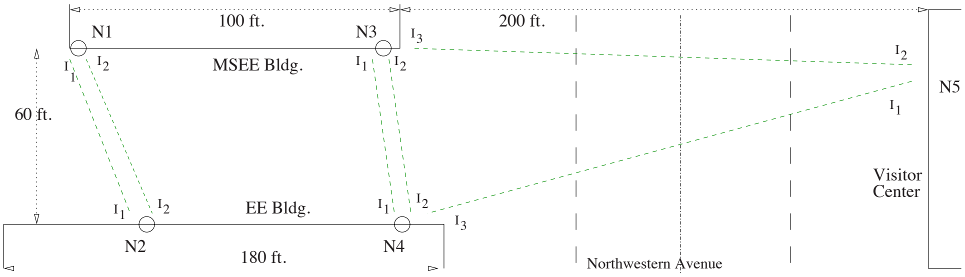


Fig. 4.5. Purdue MAP testbed schematic (top view). Only outdoor mesh routers are depicted.

DMesh with OMesh, the original MAP architecture with only omnidirectional antennas and running the OCA channel assignment algorithm.

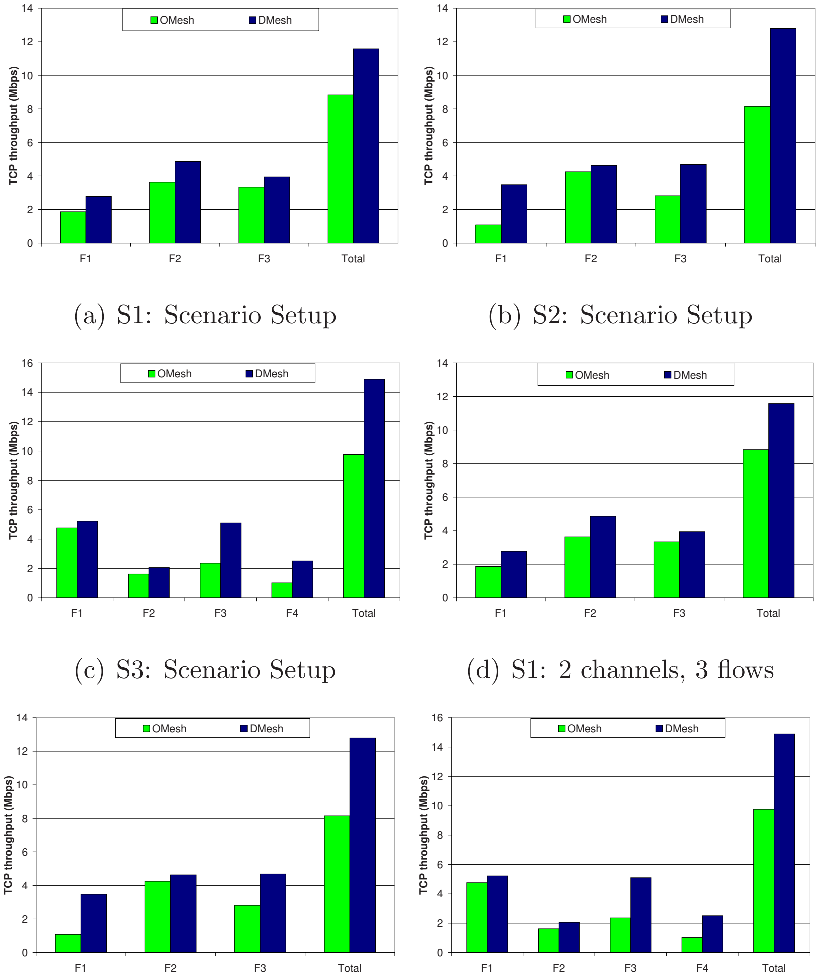
4.5.1 Setup

We deployed DMesh on a subset of our testbed across across three buildings (EE, MSEE and VC (Visitor Center)). We use Senao Engenius 2511 802.11b wireless cards on the mesh routers. Each radio is attached to a 2dBi rubber duck omnidirectional antenna with a low loss pigtail to provide flexibility in antenna placement. Each mesh router runs Linux kernel 2.4.20-8 and the open-source *hostap* drivers are used to enable the wireless cards. IP addresses are statically assigned. The wireless cards we use can support a wide range of power settings (up to 200mW). As in the simulations, the power was adjusted when the cards were used with directional antennas to keep the range similar to the omnidirectional antennas. We used directional antennas in the outdoor part of our network to interconnect the 3 buildings together. The outdoor portion of our testbed is depicted in Figure 4.5. Two of the buildings (EE and MSEE) have 2 outdoor mesh routers each equipped with 2 or 3 radios while the VC building has a single outdoor mesh router with 2 radios. These outdoor nodes provide high bandwidth interconnection across the indoor networks in each building. We used directional antennas on these outdoor nodes and compared their performance to using omnidirectional antennas. The directional antennas are the same as those modeled in the simulation. Specifically, we use a 802.11b 45° beamwidth antenna (model HG2412Y from [36]). Note that each mesh router also has an additional omni-directional control interface.

4.5.2 Evaluation

In this section, we demonstrate the advantages of the DMesh architecture using testbed experiments.

Exploiting multiple channels The first challenge in the evaluation was to obtain simultaneous throughput in the multi-channel omnidirectional scenario. As a baseline, we found that a single flow between any two nodes without any other interfering flows achieved a TCP throughput in the range of 4.5-5Mbps using *netperf*. We then ran two simultaneous flows on interfaces I_1 and I_2 on nodes N_1 to N_2 (Figure 4.5) on maximally separated channels (1 and 11). In this experiment, we found that either one radio achieved close to the full throughput in the single flow case (4.5 Mbps) with the other card being totally starved and achieving less than 0.5Mbps, or both radios achieved around 2-3Mbps each. Thus even though we used external antennas with pigtails, simultaneous throughput was not possible due to interference among the two radios. We then purchased 12 inch extension cables for each radio and further separated the antennas on both N_1 and N_2. This resulted in almost simultaneous throughput (with some loss) being achieved of around 4.3 Mbps each. Thus, antenna separation is critical to enabling multi-channel simultaneous throughput and is required on both the receiver and transmitter. In conclusion, to exploit even maximally frequency separated transmissions required antenna isolation. We also noticed that using power control to have a higher power on one interface adversely affected the other interface despite antenna separation. We thus use similar transmission powers on all interfaces of a single mesh router.



(a) S1: Scenario Setup

(b) S2: Scenario Setup

(c) S3: Scenario Setup

(d) S1: 2 channels, 3 flows

(e) S2: 2 channels, 3 flows

(f) S3: 3 channels, 4 flows

Fig. 4.6. Testbed results. OMesh is a WMN with omni antennas running OCA while DMesh is a WMN with practical directional antennas running C-DCA.

Evaluating the gain from DMesh We used 3 scenarios depicted in Figures 4.6(a), 4.6(b) and 4.6(c), each in a different topology and traffic setting, to demonstrate the throughput gains from DMesh, compared to OMesh. In scenario S1, we assume that the number of channels allowed to be used is 2. This is because the Purdue Wireless LAN (Airlink) operates on the remaining orthogonal 802.11b channel in the area and we did not want to disrupt the other users' performance. There are three flows as shown in Figure 4.6(a). Both OMesh and DMesh automatically assign separate channels on N_3 based on the OCA and C-DCA algorithms. Node N_1 chooses channel 1 to assign to its flow F_3. The performance of the flows in this scenario for OMesh and DMesh are shown in Figure 4.6(d). The results show that for OMesh, the total throughput is 8.8 Mbps. This occurs because F_1 and F_3 contend with each other causing a reduction in their individual throughputs. We also observe that unfairness exists, since F_3 achieves a higher fraction of the bandwidth than F1. Interestingly, the throughput of F_2 also reduces from its maximum individual throughput although it is on a separate channel. In contrast, using DMesh, F_2 achieves close its individual maximum throughput. The directivity reduces the inter-channel interference from other nodes. In addition F_1 and F_3 improve their throughputs due to increased spatial separation. Overall DMesh provides a 31% increase in TCP throughput. This result is close to the 35% gain observed overall in simulation results for TCP performance.

Note that although DMesh improves the throughput, the practical directional antennas do not completely isolate the transmissions spatially. For example, even in DMesh, F_1 and F_3 do not obtain their maximum individual throughput which is on average 4.5Mbps. Thus, the gain from DMesh can be attributed to a reduction in interference power levels since the interference between F_1 and F_3 is limited to the power received from their sidelobes. This reduced interference is enough to provide significant throughput gains to justify the use of such practical directional antennas in WMNs.

The pattern of the antenna used in this experiment has a very low gain in the backlobes in comparison to the sidelobes and the main lobe. To verify whether further gains are possible if the back lobe is facing the interfering flow, we used scenario 2 (Figure 4.6(b)). Similar to the previous scenario, we again assume that only 2 channels are available. In this scenario, although F_1 and F_3 are still on the same channel, the antenna used for F_1 is facing the other way with its back lobe towards the antenna for F_3. The results in Figure 4.6(e) show that OMesh performs slightly worse than in scenario 1 and achieves an aggregate throughput gain of around 8Mbps. One possible explanation is that F_1 now operates over a longer distance (and thus lower SNR) and is starved by F_3 more in comparison to scenario 1. As expected, DMesh now performs better, providing a 57% TCP throughput gain over OMesh, since the transmitting antenna for F_1 only interferes using the backlobe with F_3. The receiving antenna for F_1 whose main lobe is towards F_3 could potentially interfere with F_3. However, the power received and thus the interference are lowered since this antenna is 380 ft away and only transmits short TCP ACK and MAC layer ACK packets. Note that despite the low gain backlobes, the transmitting antenna for F_1 is not totally isolated from the antennas involved in F_3. Nodes N1, N2, and N3 can still ping each other on the directional interfaces although with a packet loss rate of close to 50%.

In the third scenario (Figure 4.6(c)), we used all 3 802.11b orthogonal channels in experiments performed late at night so as not to interfere with Airlink users. There are 4 flows depicted in the figure. Again, we find that OMesh provides an aggregate throughput of less than 10Mbps (Figure 4.6(f)). Flows F_2 and F_4 contend since OMesh runs out of channels. Although DMesh also runs out of channels, it significantly reduces interference power between F_2 and F_4 and provides an aggregate throughput of close to 15Mbps resulting in a 50% TCP throughput gain.

In summary, the testbed evaluation shows that DMesh is a viable and useful architecture to improve the performance of WMNs. Although practical directional antennas do not provide *complete* spatial isolation, the reduced interference power provides significant throughput gains. In particular, the reduced interference is useful for multiple-channel networks to obtain close to their maximum throughput potential. Costlier array-based antennas could potentially improve the spatial isolation with tightly controlled sidelobes and backlobes. However, such antennas are still not cost-effective. The cost-benefit tradeoffs in using more advanced antennas in comparison to cheaper versions is an interesting topic of our future research. As a side-note, our experiences show that using practical directional antennas was not inconvenient. Their sizes are reasonable and orienting the antennas perfectly towards each other was not critical to performance.

4.6 Related Work

Many design issues of WMNs have been studied [29,34,44,79] and many companies are offering products for deploying WMNs [10, 61]. The use of multiple radios has been proposed as a means to increase the throughput of wireless networks [8, 29, 51, 78, 79]. Multiple beamformed antennas were previously proposed in a position paper [44] for the TAPs architecture. In contrast to DMesh, TAPs proposes to use significantly costlier smart beamforming antennas but may provide higher spatial reuse. We position DMesh as an architecture to quickly and cheaply deploy high throughput mesh networks with minimal cost using commodity antennas combined with channel assignment. The work in [75, 76] uses multiple directional antennas. However, their technique does not exploit frequency separation and is designed for situations where only a single channel is available.

Many MAC layer solutions [53,63,86,90,96] as well as routing layer solutions [87] have been proposed to exploit multiple channels in wireless networks. The work in [9,51,78,79] applies multi-channel techniques to mesh networks. A recent work [78] proposes a WMN architecture with multiple interfaces and utilizes multiple channels to improve throughput. However, their architecture provides only one degree of separation (frequency) as it uses omnidirectional antennas. In contrast, DMesh combines cheap directional antennas with multiple channels to further improve the performance of WMNs. Two recent works [6, 78] study *joint* routing and channel assignment for multi-radio, multi-channel omnidirectional WMNs. In contrast to these approaches, DMesh performs directional channel assignment and decouples routing from channel assignment (similar to in [29]). The work in [29] computes routes for a network in which channel assignment has been previously done; while DMesh computes channel assignments for a routing tree that has been physically formed due to placement of antennas.

The use of beamforming antennas have been proposed for mobile ad hoc networks [18, 47, 64, 77, 77, 89]. Work has also been done on combining channel assignment and sectored separation of users in cellular networks (e.g. [70]). However, the multi-hop routing and non-existence of infrastructure devices such as base stations make WMNs very different from cellular networks. Thus, cellular network techniques cannot be directly applied to WMNs.

4.7 Summary

In this chapter, we proposed DMesh, a novel architecture for improving the performance of wireless mesh networks. Such an architecture exploits multi-radio, multi-channel nodes in the mesh network, where each interface is equipped with a practical directional antenna. We also proposed a distributed algorithm to perform routing and directional channel assignment in the DMesh architecture. By exploiting the spatial separation offered by directional antennas and the frequency separation offered by multiple non-overlapping channels, our proposed architecture allows more concurrent transmissions than an omnidirectional, multi-radio, multi-channel mesh network, and as a result achieves higher throughput. In particular, simulation results and evaluation on a mesh network testbed show that, compared to the omnidirectional, multi-radio, multi-channel mesh network, our proposed architecture improves packet delivery ratio and throughput and drastically lowers average per-packet delay. In conclusion, DMesh provides higher throughput for WMNs while remaining cost-effective and easy to deploy. DMesh can be used to naturally extend current widely deployed single radio mesh networks.

5. PROTOCOL AND ALGORITHM FOR LEVERAGING STORAGE DEVICES

5.1 Introduction

In wireless mesh networks, all Internet-access traffic flows through one or a limited few gateway nodes. This can can cause significant congestion around the gateway. In this chapter, we propose to leverage cheap storage devices on mesh routers using advanced routing techniques to alleviate this congestion.

Previous studies have shown that significant locality exists in Internet accesses from a given population of clients. Web caching has been proposed and extensively studied to exploit such locality in reducing the Internet-access traffic and the client-perceived access latency. We anticipate that similar locality will exist in the Internet-access traffic in WMNs once they become widely deployed. In this chapter, we exploit such locality among the client Internet-access traffic in a WMN and explore content caching to mitigate the congestion bottleneck at the gateway nodes of a WMN.

One way of exploiting locality in client Internet accesses in a WMN is to have the client nodes in the WMN form a peer-to-peer network and perform cooperative caching directly with each other [38, 104]. However, we argue that this approach has several disadvantages: (1) It does not leverage the available infrastructure in a WMN, i.e. the mesh routers; (2) It faces deployment challenges as it is non-transparent to clients and requires clients to contribute resources; (3) It requires the cooperative caching protocols to deal with churn and mobility of clients, e.g., clients could leave in the middle of a download; (4) It needs to deal with security and privacy issues in the presence of malicious clients which is not an issue if the infrastructure is deployed by a single service provider (e.g. Google WiFi). To avoid these disadvantages, in this chapter, we focus on transparent cooperative caching at the mesh routers.

In this chapter, we propose *MeshCache*, a transparent cooperative caching system that exploits locality in Internet-access traffic in a WMN to mitigate the gateway bottleneck effect. Unlike transparent web caching in the Internet where a web cache is attached to the gateway of an organization, MeshCache leverages the fact that a WMN typically spans a small geographic area and hence mesh routers are easily over-provisioned with CPU, memory, and disk storage, and extends the individual wireless mesh routers a WMN with built-in content caching functionality. It then performs *cooperative caching* among the wireless mesh routers. Currently available mesh routers (summarized in Table 5.1) have good processing speeds and can be expanded easily with USB-based or microdrive storage to also perform caching.

Cooperative caching among mesh routers allows clients to fetch cached data from routers within the WMN. This spreads the load in the network and hence alleviates the congestion around the gateway. In addition, cooperative caching among mesh routers in a WMN has two other performance benefits: (1) Cooperative caching allows clients to potentially obtain content from nodes closer (in network hops) than the gateway. This improves the client throughput since the throughput falls rapidly with the increased hop count in multi-hop wireless networks. Further, such local communication improves the capacity of WMNs as they scale in size [52]. (2) When there are multiple cached copies of the requested content in the network, cooperative caching enables clients to choose the *best* cached copy based on high throughput link-quality routing metrics (e.g. [21]), thereby further improving client throughput.

In this chapter, we explore two architectural design choices for MeshCache (A2 and A3 below), and compare it to a third one (A1), which is similar to the typical way that a web cache is deployed in the Internet:

- A1: A web proxy cache is connected to the WMN gateway node similar to how a web cache is attached to a gateway router in the Internet. This proxy cache transparently hijacks clients' content requests to exploit the locality within the client population of a WMN.

- A2: Each mesh router acts as a cache using expandable storage devices. When a client issues a request for a data object, its access mesh router transparently hijacks the request and searches for the object in its local cache. On a hit, the object is simply served from the access mesh router. On a miss, the access mesh router searches for a cached copy of the data object in other mesh routers including the gateway. If a copy is found, the access mesh router fetches the copy, otherwise it obtains the object from the origin server via a proxy cache at the gateway (as in A1).

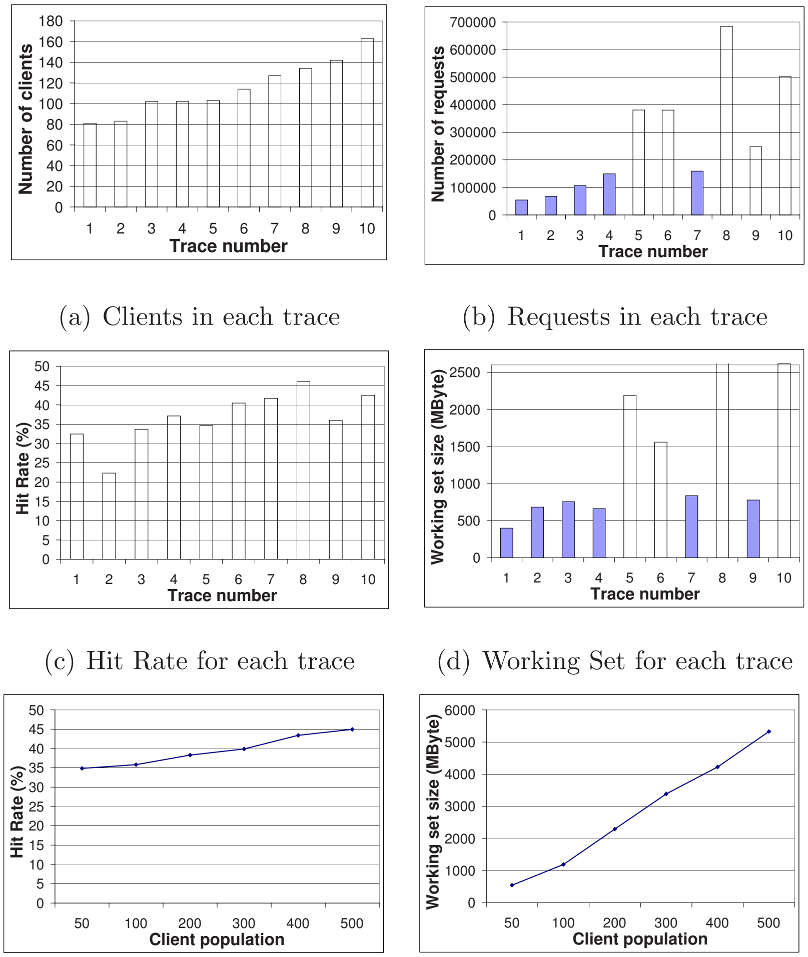
- A3: In addition to A2, when the data object is fetched either from a gateway or another mesh router along a multi-hop route, the object is cached at each mesh router along the route. This increases the availability of the data object for future requests without explicit replication. To enable such *hop-by-hop caching*, we use *per-hop transport* that breaks a single end-to-end transport connection, e.g. S to D using route S-A-B-D, into multiple single-hop transport connections along the route, e.g. S-A, A-B, B-D, and pipelines data over these sub-connections. This enables the data object to be cached at A and B in addition to S.

The architectures A2 and A3 require a cache selection protocol to locate a cached copy of content and/or choose among multiple cached copies. To this end, we design and compare three cache selection protocols for MeshCache. Our designs leverage the abundant research results from cooperative web caching in the Internet.

We vigorously evaluate the performance of MeshCache using simulations and testbed experiments. Our evaluation results shows that A3 (cooperative caching with hop-by-hop caching) outperforms the other architectures in reducing the gateway load and improving the client throughput, irrespective of the cache selection protocol used. Additionally, we found that the per-hop transport for hop-by-hop caching in A3 provides increased content availability without adversely affecting transport throughput or network overhead compared to end-to-end transport. Further, for the best performing architecture A3, we found that the cache selection based on *limited broadcast flooding* is the best strategy to alleviate the gateway bottleneck and obtain throughput improvement. Particularly, our simulations showed that: (1) A2 increased client throughput by up to 50% and reduced gateway load by up to 35% compared to A1, (2) A3 increased throughput by up to 52% and reduced gateway load by up to 66% compared to A2, and (3) the best cache selection strategy for A3 is based on limited broadcast flooding and provides a reduction in gateway load by up to 20% compared to a strategy with no search delay or overhead. Our smaller scale testbed experiments also show that MeshCache can improve the overall throughput and reduce the network load significantly. Specifically, measurement results from a deployed implementation of MeshCache on the 15-router MAP mesh network testbed [111] show that the number of transfers that achieve a throughput greater than 1 Mbps is increased from 20% in A1 to 60% in A3 while the load at the gateway is reduced by 38% in A3 compared to A1.

The contributions of this chapter are summarized as follows: (1) We propose to alleviate the bottleneck at the WMN gateway that commonly arises in WMNs by exploiting locality in client Internet accesses. (2) We present a practical cooperative caching system, *MeshCache*, and explore the design space of the MeshCache architecture and the associated cache selection protocols via extensive simulations. (3) We design and implement the MeshCache system by modifying an open source caching proxy, *Squid*, developed for the Internet, and demonstrate the benefit of MeshCache using an implementation deployed over a mesh network testbed of 15 mesh routers.

The remainder of the chapter is structured as follows. Section 5.2 presents a feasibility study of MeshCache by analyzing the potential locality in WMN traffic. Section 5.3 presents various architectural design choices for MeshCache and Section 5.4 presents various cache selection algorithms for MeshCache. Section 5.5 presents our simulation methodology and Section 5.6 presents detailed simulation results comparing different MeshCache architectures and cache selection algorithms. Section 5.7 presents the system design and implementation of MeshCache in a WMN testbed and



(a) Clients in each trace

(b) Requests in each trace

(c) Hit Rate for each trace

(d) Working Set for each trace

(e) Hit Rate v/s Population

(f) Working Set v/s Population

Fig. 5.1. Study of locality in small client populations.

Section 5.8 presents the measurement results from the testbed. Finally, Section 5.9 discusses related work and Section 5.10 concludes the chapter.

5.2 Motivation

In this section, we motivate the cooperative caching approach of MeshCache by assessing the extent of locality in the Internet-access traffic expected to be carried by a WMN.

It has been shown that Internet-access traffic has substantial locality [12], i.e., multiple users are likely to request some common data objects from the Internet. Since WMNs primarily carry Internet-access traffic, significant locality is also expected to exist in the traffic of a WMN. However, the extent of locality in the traffic is dependent on the size of the client population. A fundamental question that determines the potential performance benefits of MeshCache is whether there exists significant locality in Internet-access traffic in a WMN, given the small client population served by each gateway router.

While no measurement study has been performed specifically for Internet client traffic in WMNs, we can approximate the client population served by a gateway node of a WMN with that seen by the gateway proxy cache of a small organization. For such an approximation, we analyzed real web proxy traces collected in the week of October 19, 2005 by `www.ircache.net`. The collective trace of 1 day's traffic from 10 proxies contains 2.7 million requests originating from 1151 unique clients to 81,289 servers. The number of clients and requests in each proxy's trace are depicted in Figures 5.1(a) and 5.1(b), respectively. The number of clients in each trace ranges from 80-160 nodes which is a potential target size for a WMN with a single gateway. For example, a recent work [31] shows that 114 users can potentially be supported with a 21 mesh router WMN.

We studied the locality and working set size of the cacheable content in each trace[1]. The locality in the access patterns of such a small set of clients is encouraging as the average hit rate is around 37% with a maximum of 46% (Figure 5.1(c)). This suggests that a significant fraction of requests can be fetched from peer mesh routers if caching is enabled. The hit rate also tends to increase with the client population. We note these observed hit rates are consistent with other studies ([38, 95]) on web caching using different proxy traces for client populations of similar sizes. To study how the client population affects the hit rate, we combined all the traces into one

[1] Similar to in [38], we consider requests with SSL and dynamic content as not cacheable and always resulting a miss. We also limit the maximum size of any single cacheable object to 16MB similar to many deployed web caching systems.

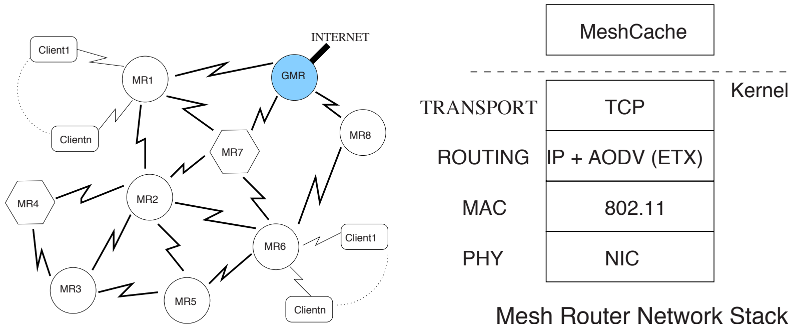
trace and simulated different client population sizes ranging from 50 to 500. For each population size k, we selected all the requests from k randomly chosen clients and ran a simulation to find the hit rate and the working set size. For each k, we tried 25 random trials to find average behavior. The results in Figure 5.1(e) show that the hit rate increases gradually from 35% to 45% as the client population grows from 50 to 500. In summary, there is significant locality to exploit in smaller client populations such as those in WMNs.

Another property of the Internet-access traffic which determines the feasibility of caching is the working set size of the traffic. Figure 5.1(d) shows the working set of a day's worth of traffic to be 1.3 GB on average with a maximum of 2.6 GB. We also measured the working set size as a function of the client population. Figure 5.1(f) shows that as the client population grows from 50 to 500, the working set size grows from 500 MB to 5.2 GB for a day's worth of traffic. These sizes can be accommodated in current mesh routers throughput expandable storage devices.

5.3 MeshCache Architecture

Before discussing the architectural design choices for MeshCache, we present our basic network model. With out loss of generality, this chapter assumes that MeshCache is deployed in a WMN similar to RoofNet [107], consisting of single-channel, single-interface 802.11 mesh routers with omnidirectional antennas and sparsely deployed gateways for Internet access. The gateways are not widely deployed due to cost and uplink constraints. A similar architecture can also be used for corporate networks to replace wired access-point-based WLAN systems [31].

Figure 5.2 depicts a typical network setup. Such a WMN provides Internet access as follows: Each client's packets are first received by the client's *access mesh routers*, i.e., a mesh router the client's interface is associated with (e.g., MR 1 and MR 6 are access mesh routers for their clients). These mesh routers then forward the packets to the *gateway mesh router* (GMR) using other mesh routers. The GMR provides



(a) Basic Network model (b) Operational layer

Fig. 5.2. MeshCache architecture.

Table 5.1
Mesh router hardware specifications popular in WMNs.

Mesh Router Device	Processor Speed	Memory	Storage
MeshCube	400Mhz	64MB	USB expandable
LocustWorld MeshBox	500Mhz	128MB	USB expandable
Soekris net4801	266Mhz	128MB	USB expandable
Netgear WGT634U	200Mhz	32MB	USB expandable
iBox Slim Mini-ITX PC	533Mhz	up to 1GB	USB expandable
Metrix Mark II	233Mhz	64MB	USB expandable

Internet connectivity through a high bandwidth wired/WiMax interface. The gateway may perform other functions such as IP address assignment or NAT. All the MRs use a routing protocol (e.g. AODV) with metrics such as ETX [21], ETT [29] to find routes to each other and to the gateway. We assume that clients use TCP at the transport layer as it is widely used for Internet access. Further, we assume that the WMN employs a 802.11 MAC layer. As shown in Figure 5.2(b), the MeshCache system is implemented over this underlying WMN by a user-level MeshCache daemon.

5.3.1 Architectural Design Choices

We now discuss the architectural design choices for MeshCache that differ in where caching is performed to exploit locality and the implications of such choices.

Architecture 1 (A1) In A1, a caching proxy is connected to the WMN gateway node similar to how a web cache is attached to a gateway router of an organization in the Internet. Client requests are transparently hijacked at the GMR and redirected to the caching proxy. As the results in Section 5.2 showed, the hit rates in the attached caching proxy are expected to be in the range of 30-40%, resulting in significant bandwidth savings between the GMR and the origin server. The requests with hits in the caching proxy could also have higher TCP throughput as they will not be affected by TCP's inefficiencies in traversing wireless-wired links [100]. However, the bottleneck at the gateway still remains since the medium access load around the gateway is not reduced in this architecture.

Architecture 2 (A2) A key observation in WMNs is that a WMN typically spans a small geographic area and hence mesh routers are easily over-provisioned with CPU, memory, and disk storage, and the MRs themselves can be potentially leveraged to cache content and provide further performance improvements. A survey of hardware specifications (Table 5.1) of some popular choices for implementing MR devices shows that these devices already have adequate processing power to implement a caching proxy. Unlike high-end web proxies and routers which service thousands of connections simultaneously, the smaller scale of a WMN reduces the computational load on the MRs and makes implementing a caching proxy on each of them computationally feasible. In addition, all of the devices can be expanded to include USB flash drives (which can store 4-8 GB currently) and some can even use USB hard drives with capacities of up to 20 GB. The small form factor and declining cost of the USB flash drives makes them a suitable candidate for addressing the storage needs of proxy cache. Moreover, Section 5.2 shows that the working set of a small client population is small enough to be accommodated by these USB devices. Cooperative caching at all the MRs further reduces the individual cache sizes required to accommodate the working set.

To exploit the above observation to enhance MeshCache's performance, in architecture A2, all MRs performs caching in addition to routing. When a client requests content, its request is transparently hijacked by the client's access MR and the content is searched for in the access MR's cache. On a local hit, the content is simply served from the access MR. However, on a local miss the access MR searches for a cached copy of the content in other MRs (including GMR). If a copy is found, the access MR fetches the copy, locally caches the content and returns it to the client. Otherwise, the access MR fetches the content from the origin server via a proxy cache at the gateway (as in A2), locally caches the content, and returns it to the client. Thus, the main difference between A2 and A1 is that *all* MRs can cache data if they are required to. They could cache content simply because they are the access MR or for example, if the content is hashed to them.

Architecture 3 (A3) Architecture A3 builds on A2 and pushes the cooperative caching technique to the extreme. Note that when content is fetched either from the origin server (via GMR) or another mesh router along a multi-hop route, the MRs along the path over which data flows can also cache the data. This increases the availability of content for future requests without explicit replication. Increased availability of data improves the cache hit rate and reduces the path length of future requests.

Ideally, this increased availability of cached content can be achieved without extra overhead if each node along the path can snoop network layer packets to assemble an entire file. However, this is difficult to achieve in practice because: (1) Routing paths may fluctuate during a transfer and the bytes of a file may travel different paths. (2) Routing paths in forward and reverse directions may be different in many cases, e.g., due to unidirectional links. We found this to commonly occur in our testbed

(Section 5.8). (3) Even if all bytes of a file pass through each MR in a path during its download, it is difficult to decipher the transport and application protocol states to assemble the packets into the application file.

To enable hop-by-hop caching, we resort to *per-hop transport*. Per-hop transport breaks a single end-to-end transport connection, for example, from S to D using route S-A-B-D, into multiple single-hop transport connections along the route, e.g., S-A, A-B, B-D, and pipelines data on these sub-connections. This allows the data object to be cached at A and B in addition to S while it flows through the route. Thus, content is transferred over an N-hop path through N pipelined transport connections, resulting in the file being cached at each intermediate node in its entirety. This per-hop transport mechanism essentially *fixes* the route for the duration of a file transfer in order to cache it along the path. We argue that this is viable since most content requests (i.e. HTTP) are small in size, and hence the underlying routing path is unlikely to deteriorate in the time the route is fixed. Note that per-hop transport precludes unidirectional links from being used for transferring content since a bidirectional TCP handshake is required at each hop. In summary, A3 is similar in operation to A2, except that when the a data object is fetched, the object is cached at every hop along the route by using per-hop transport.

In hop-by-hop transport, all the mesh routers along a path (including the gateway) initiate persistent connections, which simply replay client/server behavior. Only when the client or server explicitly closes the connection, all the hop-by-hop connections will close. Thus, if the client's browser uses a persistent HTTP connection, the persistent connection semantics is preserved from the client's point of view.

While A3 maximizes the opportunities for caching among the wireless mesh routers, the per-hop transport requires data packets to traverse up the TCP/IP stack at every hop which incurs extra overhead and delay compared to the end-to-end transport. We study these issues thoroughly in Section 5.6.1. In addition, per-hop transport requires per-connection state to be set up on all nodes along the path. While this may not be be a good idea in mobile ad hoc networks which typically consist of resource constrained devices, it is less of a concern for the mesh routers which are less resource constrained.

An important component in both architecture designs A2 and A3 is an efficient cache selection protocol to locate a cached copy of the content. We next discuss the design choices of a cache selection protocol.

5.4 Cache Selection Protocols

The design of cache selection protocols has been widely studied in the context of Internet web caching systems. Thus, we first provide a brief background of the approaches used in the Internet and draw inspiration from them in designing cooperative cache selection protocols for WMNs.

5.4.1 Cooperative Caching in the Internet

Several previous works have proposed cooperative cache selection protocols for the Internet. The works in [17, 22] introduced **hierarchy-based selection**, i.e., using a hierarchy of caches that resolve MISSES from lower levels of the hierarchy until the root is reached which fetches content from the origin server. Another technique of **reactive query-based selection** is exemplified by ICP [93] which on a MISS, queries its peer caches for the content item and chooses one of them to forward the request to. To remove the delay and processing overhead of query-based selection, **hash-based selection** (e.g. CARP [92]) has been proposed in which on a MISS, a cache fetches the content item from a cache selected by hashing the content item's URI (Uniform Resource Identifier). Finally, in **proactive dissemination-based selection** [32, 81], caches proactively distribute summarized information about their cached content items (using Bloom filters) to each other to remove the delay penalty from queries.

However, *the solutions proposed for cooperative cache selection in the Internet domain are not directly applicable to MeshCache* for the following reasons: (1) They do not optimize the network overhead in terms of the number of packets in the system which is important in WMNs. (2) They fail to incorporate advantages possible from the broadcast nature of wireless communication. (3) Since the mesh router itself is a cache in a WMN, unique cross-layer approaches can be designed to improve the cache-selection schemes (e.g. consulting the underlying routing protocol to choose from multiple cached copies based on routing metrics such as ETX [21]). Solutions developed in the Internet can only exploit the caches themselves and not the routers in between to perform cache-selection decisions. Thus, they are restricted to using application-specific metrics such as end-to-end latency.

In the next section, we explore the design space of cooperative cache selection protocols for the MeshCache system. The design for these cache selection protocols is inspired by their counterparts in the Internet, but have been adapted to operate efficiently in the wireless environment. All the protocols are implemented at the application layer to allow them to be useful for a multitude of WMN architectures. For example, they should be able to operate on top of source routing as well as hop-by-hop routing protocols. Additionally, they should operate over and be able to exploit protocols implemented with new routing metrics or multi-channel multi-radio devices [29, 78]. They should also be operable with any new transport protocol that may be invented for WMNs. At the same time, these protocols should leverage the information exposed by the underlying layers using a cross-layer approach to enhance their performance. To satisfy both these requirements, MeshCache adopts a loose coupling principle whereby MeshCache cache selection protocols interact with underlying layers via a set of predetermined APIs. This enables portability to any underlying layer that provides the APIs as well as performance benefit arising from the interaction with lower layers.

5.4.2 Cache Selection Protocols for Architecture A2

The cache selection protocol in MeshCache should select a suitable MR for each content item, and retrieve the content from the chosen MR. In the following, we present a set of design choices for the cache selection protocol.

Tree-based Hierarchy Cache Selection Protocol (THCP)

This is a basic scheme in which the access MR simply routes the content request to the GMR (gateway) in case of a local miss. The access MR selects the current best gateway[2] by querying the routing protocol (through an API *BestGateway()*) and forwards the request to the corresponding GMR. This is a two-level hierarchy-based cache selection approach with the GMRs being the parent caches for other MRs. We do not perform gateway selection at the application layer in any cache selection protocol because the routing protocol has more fine-grained information about the *best* gateway, through probing of link metrics such as loss rates [21], latency [3] or complex metrics that take into account link bandwidths and loss rates in the presence of multiple radios and multiple channels [29].

Following the selection of a GMR, an end-to-end transport connection is established to the selected GMR via a multi-hop network layer path and the content transferred.

Broadcast Cache Selection Protocol (BCP)

While THCP does not cause search overhead or search delay, it can only exploit local hits at each access MR and hits at the selected GMR, and not at the caches of other MRs in the vicinity. In contrast, the broadcast-flooding search based protocol (BCP) can locate content items in other MR nodes.

[2] There may be multiple gateways advertising Internet connectivity in a WMN.

In this protocol, on a local miss, the access MR first queries the routing protocol for the path metric, say ETT (Expected Transmission Time) [29] X, to the closest gateway (through an API *BestGatewayMetric()*). The access MR then initiates a UDP broadcast of a *content locate* message by inserting the metric X, a search path metric $Y = 0$ and a locally unique sequence number. As the *content locate* message is propagated, each node rebroadcasting the message exactly once based on the sequence number. In doing so, each node also adds the ETT to the node it got the packet from into the value Y when rebroadcasting the message. If at some node the value Y exceeds X, there is no need to search further since the originating node already has a better path, i.e., to the gateway, and the broadcast is terminated at the node. In this manner, the broadcast search is limited to paths better than the one to the gateway.

Each node that has a hit in its local cache for the content item replies with a *content found* message and does not rebroadcast the *content locate* message. The access MR then queries its routing protocol to find a node with the best path metric from among those that had hits (through an API *BestNode(*$< nodeIPlist >$*)* and selects the MR returned by the API to fetch the content. Note that if all nodes with hits have a worse routing metric than the gateway or no other nodes have a cached copy of the content, the gateway itself is chosen (same as THCP). The data is once again transferred over a multi-hop network layer path.

Another important parameter in BCP is search timeout (ST) which is the amount of time a node should wait to receive replies from a broadcast search before reverting to the default behavior (THCP). BCP uses the following technique to estimate ST: Whenever a new closest gateway is discovered by the routing protocol, a node pings that gateway once to get an approximation for the round trip latency from the node to the gateway. This ping time is then used by BCP as the search timeout. Effectively this method adapts the search timeout at each node with respect to its closest gateway.

BCP draws its inspiration from the ICP protocol for the Internet. However, there are some important differences in the operation of the two protocols. First, BCP exploits wireless multicast advantage (WMA), i.e. the ability to deliver multiple query packets with a single transmission [94]. Second, the scope of the BCP queries is limited by exploiting the proximity of the gateway. Both these differences reduce the overhead and the delay of BCP compared to ICP. Thus we take an approach of loose-coupling between the two layers: Search at the application layer and determine the best cache by querying an API that can be provided by any routing protocol with a few modifications.

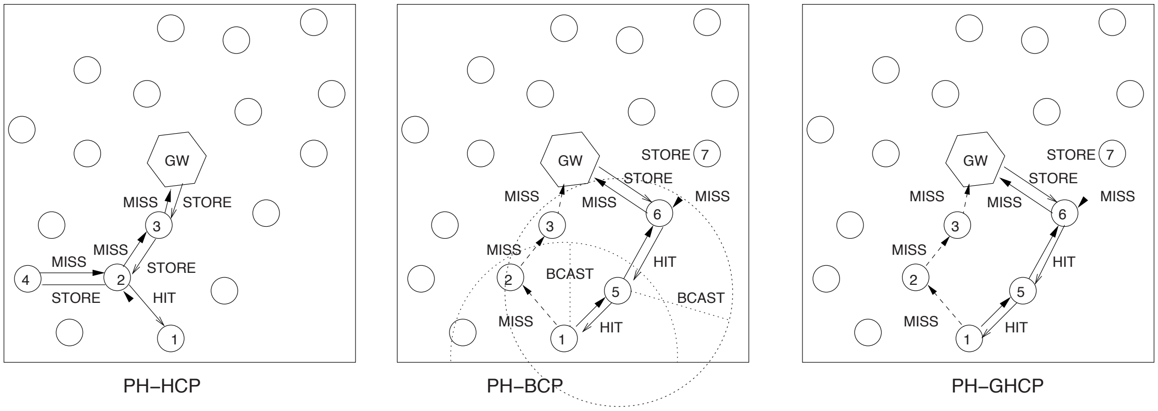
Geographic Hash Cache Selection Protocol (GHCP)

In GHCP, on a local miss, the access MR uses a well-known hash function to hash the content item's URL and maps it to a MR whose nodeID (hash of IP) is numerically closest to the hash of the URL[3]. The request is then forwarded to the hashed MR with the expectation that a hit will occur since all requests for that particular URL are redirected to the hashed MR. If the hashed MR has a miss, it uses THCP to fetch the content.

Unlike Internet hash-based cache selection protocols, GHCP controls the proximity of MRs to which the content items are hashed. The hash is not done globally by considering all the MRs in the WMN (in which case the path lengths to the hashed MR could increase dramatically). Instead, the entire area is divided geographically into virtual squares (grids), and each node hashes the URL by considering *only other MRs contained in its own virtual grid*. The grid size is a tradeoff between exploiting locality and the path length. If it is too small, there likely will be no locality and no hits; if it is too large the hash point will be far away and thus the throughput will either be bad or the gateway will always be chosen. To balance the tradeoff, we chose a grid size equal to 500mx500m in our evaluations.

In GHCP, if the hashed MR is still determined (by querying the routing protocol) to be worse in routing metrics than the best gateway, the request is simply redirected to the best gateway. Note that this scoped hashing requires knowledge of the global

[3] This is consistent hashing, same as that used in DHTs in the Internet.



PH–HCP PH–BCP PH–GHCP

Fig. 5.3. Design choices for cache selection protocols.

virtual grid boundaries and locations of the MRs to determine the MRs in each node's virtual grid. In a static WMN network, locations of mesh routers and grid boundaries can be encoded during deployment. We use this one-time encoding method in our testbed. Other deployed testbeds such as RoofNet [107] also maintain the GPS coordinates of their MRs.

5.4.3 Cache Selection Protocols for Architecture A3

The difference between the architectures A2 and A3 is that when content flows back to the access MR in A3, it is cached at nodes along the way. This caching is enabled by per-hop transport. Per-hop transport naturally forms a chained (hierarchical) organization of the caches along the way which can be leveraged to further optimize the cache selection protocols in A3. Both THCP and GHCP benefit from this optimization by searching for the request data object in this chain of caches in addition to the caches selected by their cache selection algorithms. The cache selection protocols for A3 are described below.

Per-Hop Tree-based Hierarchy Cache-Selection Protocol (PH-THCP)

PH-THCP is an optimization of THCP when architecture A3 is used. In this scheme, the access MR, on a local miss, selects a best gateway G as before and then finds the next hop node for G by querying the routing protocol (through the API *GetNextHopForNode(G)*) and establishes a transport connection to that next hop node. For example, in Figure 5.3 MR 4 has a miss in its local cache and consequently contacts MR 2. MR 2 repeats this process and on a miss forwards the request to the next hop towards G. If any intermediate node has a hit, it sends the content back to MR 4 and does not search further. Each node on the path then automatically caches the content as it is downloaded through hop-by-hop transport connections. Subsequently, when MR 1 has a miss and contacts MR 2, a hit occurs due to hop-by-hop caching and the content is fetched directly from MR 2.

Per-Hop Broadcast Cache-Selection Protocol (PH-BCP)

PH-BCP is a per-hop variant of the BCP protocol for architecture A3. In this scheme, the method to select the MR with the requested content is same as in BCP. The only difference is that the content is transferred via per-hop transport from the selected MR. Also, if no hits occur or the metrics for all the hits are worse than the gateway, the access MR reverts to using PH-THCP to forward the request to the gateway. For example, in Figure 5.3, MR 7 has a miss in its local cache and no hits from the broadcast search. It then uses PH-THCP to fetch the content from the gateway through MR 6. Subsequently, when MR 1 performs a broadcast search for the same content item, it receives a hit from MR 6 from where it receives the content, again with per-hop caching. Thus, MR 1 achieves better performance due to broadcast search and hop-by-hop caching compared to using end-to-end transport over the default path of 1-2-3-GW (shown with dotted line). PH-BCP has a delay penalty associated with it, but can be better at spreading load, as the content may be fetched from a node not on the path towards the gateway.

Per-Hop Geographic Hash Cache Selection Protocol (PH-GHCP)

PH-GHCP is a per-hop variant of the GHCP protocol for architecture A3. In this scheme, the access MR upon a local miss hashes the URL of the content item to a MR in its virtual grid (say B), similar to in GHCP. If B is closer than GMR, it then finds the next hop node towards B by querying the routing protocol (through the API *GetNextHopForNode*$(< nodeIP >)$) and the request is forwarded in a hop-by-hop fashion to B. If an intermediate node has the content, it does not forward the request further and replies to the access MR. Note that it is possible B itself does not have the content, e.g., if this is the first request for a content item. In this case, B reverts to PH-THCP to retrieve the content from the gateway and forward it to the access MR. For example, in Figure 5.3, MR 7 hashes the URL to MR 6 which on a miss fetches the content from the gateway. Subsequently, MR 1 also needs to retrieve the same content item and thus hashes the URL to node 6 from which the content item is retrieved in a hop-by-hop fashion improving performance from its default route of 1-2-3-GW. Note that although PH-GHCP has no query delay, it can cause detours in routing paths without any benefit for unpopular content items since they result in a miss at the hashed MR.

Finally, we argue that the approach of proactive dissemination of content indices used in Internet cooperative web caching is not suitable for WMNs (both A2 and A3) as proactive message exchange can cause high overhead to the wireless network. Nodes would have to proactively flood the network whenever they cache a new object, a cached object expires, or a cached object is evicted.

Note that in all protocols, we implement the routing protocol API to only provide next hop nodes that are connected bi-directionally. In the absence of this feature, hop-by-hop transport connections cannot be established.

5.5 Methodology

In the next section, we examine the feasibility of per-hop transport by comparing it to end-to-end transport, and compare the A1, A2 and A3 MeshCache architectures and the associated cache-selection protocols through detailed simulations. We describe our methodology for those experiments in this section.

We use the Glomosim simulator [105] to evaluate MeshCache. Glomosim has been widely used to study multi-hop wireless networks.

Network Model We simulate a static mesh network of 50 mesh routers placed randomly in an area of 1000m x 1000m. Each node is assumed to have 1 interface equipped with an omnidirectional antenna. All the sources communicate with the gateway node to simulate an Internet access pattern. The two-ray path loss propagation model is used. We also evaluate the performance under fading (Rayleigh) and lossy conditions.

Client Behavior Model Each mesh router aggregates queries from 5 clients. For each individual client, a successive request arrives after the current request has been served. However, the mesh router may service requests from different clients concurrently. The file request model is similar to previous caching studies [104].[4] The file request pattern of each client is based on the Zipf-like distribution that has been found to model web traces. The value of the Zipf parameter θ was chosen to be 0.8 based on measurements on web traces and similar to in previous significant studies in this area [104]. Similar to in [104], we used a request pattern in which nearby nodes have similar, although not the same request popularity distribution. The user browsing behavior is simulated using the model developed in [58]. This model also allows us to simulate the size of HTTP items retrieved, number of items per "Web

[4] We did not use the traces in Section 5.2 for the simulations as we wanted to experiment with different locality parameters and use similar workloads as existing work on caching for mobile networks [104].

page", and think time (the time elapsed between successive web page downloads from each user). We assume that 25 mesh routers have active clients.

Content Model We assume a set of 1000 files with file sizes between 1 and 100 KB from which each client makes a request based on the Zipf distribution. Similar to in [104], content items have a TTL of 5000 secs. The mesh router cache size is set to 2 MB due to the working set size and length of the experiment. Note that a production system will use much larger caches using expandable flash storage and main memory. Finally, we use the LRU cache replacement policy used in the extended Squid [110] caching proxy evaluated in our testbed evaluation in Section 5.8.

Routing, Transport and MAC We currently use TCP as the reliable transport protocol in our study since it is widely used and available as a standard part of operating systems. The routing protocol used is AODV, same as that used in our testbed evaluation in Section 5.8. We used IEEE 802.11b as the MAC/PHY layer which was verified to produce close to theoretically maximum throughput [41]. All simulation results were averaged over multiple random scenarios and each simulation modeled a one-hour period of client activity.

Metrics The metrics used in our evaluation are: (1) Network load: The average total number of packets transmitted by the network layer for a single file download. This accounts for all control as well as data packets, and is averaged across all the transfers initiated in the network. (2) Throughput: Throughput of any transfer is the ratio of the size of the transfer to the time taken to complete the transfer. (3) Average aggregate throughput (AAT): Aggregate throughput is the ratio of the total number of bytes downloaded by a single client to the total time spent by that client downloading those bytes. AAT is the average of the aggregate throughput of all the clients. (4) Hit rate: A cache hit occurs in the MeshCache system when a client has a hit in its access mesh router's cache. We refer to this as a *local* hit. Local hit could be due to another client connected to the same access MR requesting the same object. Another possibility is that when per-hop transport is enabled, an MR is an intermediate hop for a transfer to a different MR and hence has cached the content item. Now, when the intermediate MR's client requests the same object, a cache hit occurs. We log the latter type of cache hits and refer to them as *ph-local* hits. Both local and ph-local hits result in the content being fetched directly from the access MR without any communication with peer MRs or the gateway. A cache hit also occurs when an access MR obtains the content item from its peer MRs instead of the gateway. We refer to this as a *remote* hit. Once again, remote hits could be due to locality in the access pattern of two routers or due to per-hop transport. We log all remote hits.

5.6 Performance Evaluation of MeshCache Architectures

In this section, we first evaluate the feasibility of architecture A3 by studying the performance of per-hop transport mechanism with respect to the download size, hop length and wireless environment.

5.6.1 Impact of Per-Hop Transport

Per-hop transport is an elegant method to achieve hop-by-hop caching in A3. However, it is essential to understand the impact of such a per-hop mechanism on the throughput and the overhead of the transport connection. Note that per-hop transport requires establishing and growing multiple TCP connections and may require application layer data buffering due to the mismatch between upstream and downstream throughput. However, in order to conserve memory at the intermediate hops, this study uses the default 16K in-kernel socket buffers for each TCP connection and a constant application-level buffer of 8K. We first evaluate the impact of the number of hops on the per-hop transport connection.

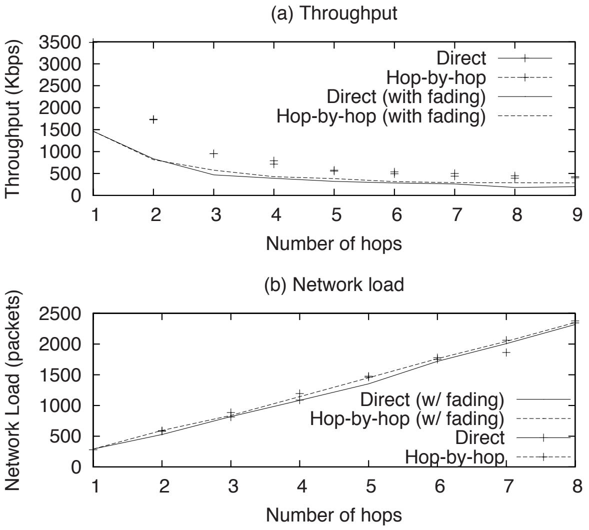


Fig. 5.4. Performance of per-hop transport with varying path length.

Performance with Varying Path Length

In this experiment, we take a chain of nodes of increasing path length and compare the throughput and the network load incurred in the presence of per-hop transport versus direct end-to-end transport. The inter-node distance in the chain is set to 200m. We vary the path length from a single hop to up to 9 hops. The choice of the maximum path length of 9 is based on observations in [11] on the path lengths in large scale WMN deployments.

Figure 5.4 demonstrates that the throughput obtained by per-hop transport is similar to that on a direct end-to-end transport connection across all values of the path length. Similarly, the network load on both kinds of transport connections are identical. Thus, using per-hop architecture does not adversely impact the throughput or the incurred network load of a transport connection and hence hop-by-hop caching is a viable design choice for the A3 MeshCache architecture. Further, in the presence of fading, which results in errors or loss of packets, the per-hop architecture has slightly better throughput than the direct transport connection.

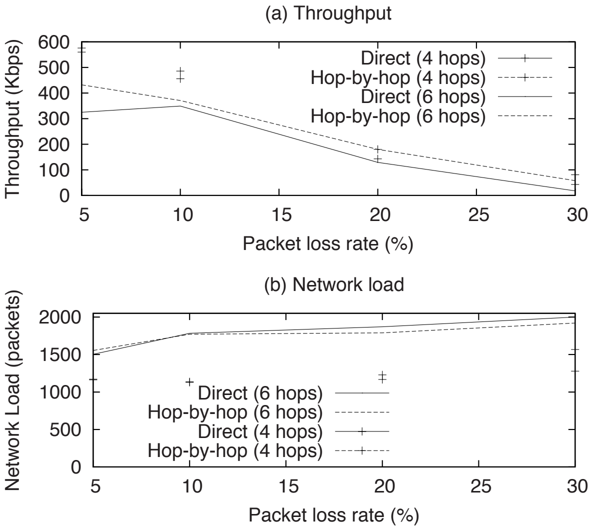


Fig. 5.5. Performance of per-hop transport as packet loss rate is varied.

The improvement from per-hop transport is explained as follows: Consider a chain of nodes A-B-C-D where A is the source and D is the destination. Suppose a packet is lost on the link C-D. In case of direct TCP connection, A will be notified when the packet is dropped on the link C-D and A will retransmit the packet. On the other hand when per-hop transport is employed, the packet loss on link C-D will be recovered by the TCP connection C-D. Thus, compared to the direct case, in per-hop transport, the detection of packet loss and the corresponding recovery is localized (between C-D) and does not affect the remaining TCP sub-connections. Also, the TCP window grows back faster after a loss on C-D than on the direct connection since the ACKs travel a single hop. Further, per-hop transport does not increase the number of packets transmitted. For example, in the direct connection, an ACK packet travels 3 hops from D to A while in per-hop transport 3 separate ACK packets are transmitted, one on each one hop sub-connection.

Performance with Varying Loss Rates

We now study the behavior of the two transport connections by varying the loss experienced by the links at each hop. Such loss of packets could be due to fading, congestion, interference, etc. We impose a loss rate at the radio layer. The loss rate is varied between 5% to 30% based on the observations in [11] that such loss rates are highly probable even on routes selected by link-quality routing protocols. Note that not all loss events at the radio layer translate to loss events at the transport layer since the MAC protocol already makes an attempt to recover from losses by retransmitting packets. Each link has a loss rate randomly chosen between 0 to the value on the X-axis in Figure 5.5 which depicts the performance of the per-hop and direct transport connections as the loss rate is varied. The per-hop transport connection outperforms the direct transport connection across all the values of the loss rate for both the 6-hop and the 4-hop chain. Also, the network load in the per-hop connection is slightly lower than that of the direct connection. This can be attributed to the quick localized detection and recovery from loss events in the per-hop transport connection.

Performance with Varying Download Size

Finally, we study the impact of the size of the file download on the transport connections by fixing the path length to be 4 hops and the loss rate at 10%. Figure 5.6 illustrates that as the download size increases, the per-hop connections outperform the direct connections by an increasing margin. This is because, as the download size increases, the duration of the connection increases, thereby increasing the number of chances presented to perform quick detection and recovery from loss. Similarly, the network load in the per-hop connection is lower than that of the direct connection.

In summary, we conclude that the per-hop transport required to support hop-by-hop caching in architecture A3 does not adversely affect performance in comparison to end-to-end transport. In fact, the throughput of per-hop transport is better than end-to-end transport under fading channels with lossy links. Since these conditions

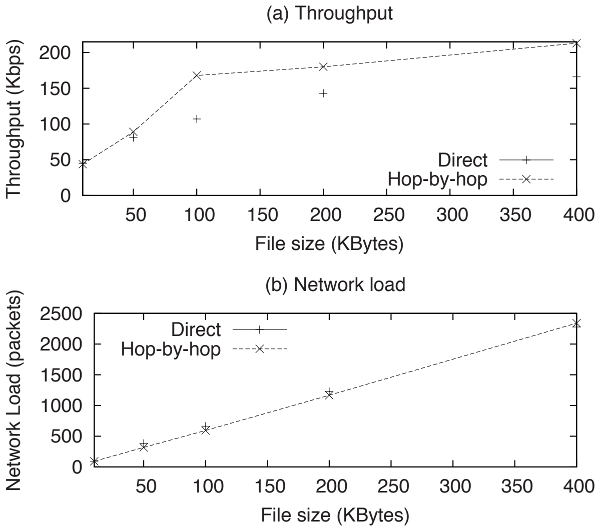


Fig. 5.6. Performance of per-hop transport with varying transfer sizes.

are typically true in wireless networks, we expect per-hop transport to generally perform better than or comparable to end-to-end transport, thereby *improving caching without a performance penalty*. Note that the above simulations capture the delay of traversing up and down the TCP/IP stack when simulating per-hop transport. Hence, the performance benefit of per-hop transport holds despite this overhead. The performance benefit of per-hop transport is also confirmed by the experimental measurements in our testbed (Section 5.8).

5.6.2 Comparison Study of MeshCache Architectures

This section presents the performance comparison of the three MeshCache architectures, A1, A2 and A3, using different cache selection protocols.

THCP Performance

We compare the performance of A2 and A3 using the THCP protocol. THCP represents architecture A2 while PH-THCP represents A3. We also include the results

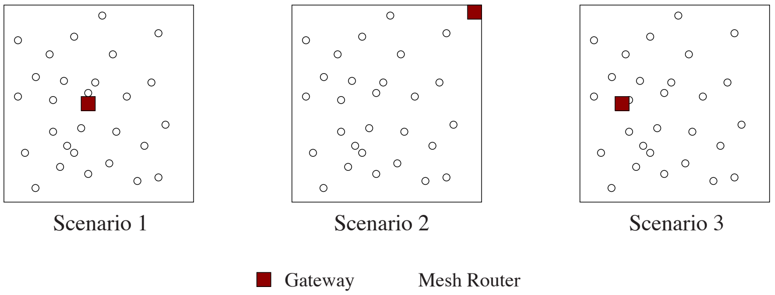


Fig. 5.7. Three different scenarios used for comparison of cache selection schemes.

Table 5.2
Performance of the architectures under THCP scheme.

Metric	AAT (Kbps)			Gateway load (KB)			Network load (Packets/file)		
Scheme	Default	THCP	PH-THCP	Default	THCP	PH-THCP	Default	THCP	PH-THCP
S1	1382.8	1479.2	1715.1	60,727	52,579	41,530	144.4	124.1	114.6
S2	467.5	561.9	853.3	60,224	52,140	17,680	279.9	242.6	171.8
S3	1312.2	1509.6	1612.7	58,128	50,215	41,011	145.8	126.4	117.4

from A1 which is referred to as *Default*. For this experiment, we consider three different placements (Figure 5.7) for the gateway node where each scenario presents a different topological balance. The comparison between Default, THCP and PH-THCP is presented in Table 5.6.2.

In Scenario 1, using THCP and PH-THCP, the gateway serves 13.4% and 31.6% lesser bytes respectively compared to Default. The reduction in gateway load is accompanied by an increase in AAT by 7% and 24% for THCP and PH-THCP, respectively when compared to Default. Also, PH-THCP has a gateway load reduction of 21% and increase in AAT by 16% with respect to THCP. THCP improves the performance when compared to Default by exploiting the local hits (hit rate of 14.4%) at each access MR. These hits are served directly by the access MR to the client and the client experiences higher throughput. In fact, the reduction in the gateway load is very close to the hit rate. PH-THCP, in addition to exploiting local hits at access MR as in THCP, has a *ph-local* hit rate of 4% and a *remote* hit rate of 13.6% due to hop-by-hop caching. Thus, all the hits (a total hit rate of 32%) in PH-THCP avoid the gateway resulting in a corresponding reduction in gateway load when compared to Default.

Similarly, the gateway load reduces by 13.4% and 70.6% in Scenario 2 and by 13.6% and 29.4% in Scenario 3 for THCP and PH-THCP respectively. Note that these reductions correspond to the total hit rate observed in these experiments. Also, since THCP exploits only local hits, its hit rate and hence reduction in gateway load remain similar across all scenarios. However, the placement of gateway affects the extent of ph-local and remote hits in PH-THCP. For instance, the reduction for PH-THCP in Scenario 2 is larger compared to that in Scenarios 1 and 3. This is due to the imbalance in the topology of Scenario 2. The imbalance causes the gateway to be accessible from only a few nodes and thus all the content obtained from the gateway is concentrated in these nodes. When other nodes request the same content, the content is more readily located in these nodes and hence results in an improved hit rate for PH-THCP.

The imbalance in Scenario 2 also causes the routes from some nodes to the gateway to be much longer than other nodes. Thus, when such nodes experience a reduction in path length due to PH-THCP as compared to THCP, there is a significant gain in AAT (82% compared to Default and 51% compared to THCP). In contrast, in Scenarios 1 and 3, the path lengths between nodes and the gateway are typically smaller. Thus most nodes already achieve good throughput that cannot be drastically improved from per-hop transport. A reduction in network load is also observed in both THCP and PH-THCP across all scenarios. A nice feature of PH-THCP is that data replication increases proportionally with the distance to the gateway, which can increase the hit rate. Thus PH-THCP is more resilient to bad gateway placement.

Table 5.3
Performance of the architectures under BCP scheme.

Metric	AAT (Kbps)			Gateway load (KB))			Network load (Packets/file)		
Scheme	Default	BCP	PH-BCP	Default	BCP	PH-BCP	Default	BCP	PH-BCP
S1	1382.8	1656.2	1705.5	60,727	38,761	32,597	144.5	110.0	107.3
S2	467.5	713.7	940.5	60,224	39,684	17,680	279.9	198.1	159.5
S3	1312.2	1546.9	1623.8	58,128	39,613	33,719	145.8	114.1	110.8

BCP Performance

In this section, we investigate the performance of the MeshCache architectures using the BCP scheme.

Here, BCP represents architecture A2 while PH-BCP represents A3. We also include the results from A1 which is referred to as *Default*. The results are presented in Table 5.3. In Scenario 1, the gateway serves 36% and 46% less bytes using BCP and PH-BCP, respectively, compared to Default. The AAT is increased by 20% for BCP and 23% for PH-BCP when compared to Default. Also, PH-BCP achieves 16% lower gateway load and 3% higher AAT than BCP.

BCP achieves better performance than Default by exploiting the local hits (hit rate of 14.4%) at each access MR as well as the remote hits from nearby MRs (hit rate of 23%). These hits are served by MRs with higher throughput than the gateway, resulting in an increase in AAT by 20%. In fact, the reduction in the gateway load in BCP is similar to its total hit rate (38%). PH-BCP further improves the availability of content and hence the hit rate (apart from the 14.4% local hit rate, *ph-local* hit rate of 4% and *remote* hit rate of 28%) due to hop-by-hop caching. Together, all the hits (total hit rate of 47%) in PH-BCP avoid the gateway resulting in a corresponding reduction in gateway load when compared to Default.

Further, the gateway load reduces by 34% and 70% in Scenario 2 and by 32% and 42% in Scenario 3 for BCP and PH-BCP respectively. Note that BCP's hit rate and hence reduction in gateway load remain similar across all scenarios since the

Table 5.4
Performance of per-hop cache selection protocols.

Metric	AAT (Kbps)			Gateway load (KB)			Packet transmissions per file		
Scheme	PH-THCP	PH-BCP	PH-GHCP	PH-THCP	PH-BCP	PH-GHCP	PH-THCP	PH-BCP	PH-GHCP
S1	1715.1	1705.5	1545.6	41,530	32,597	39,275	114.6	107.3	127.4
S2	853.3	940.5	567.4	17,680	17,680	17,812	171.8	159.5	193.6
S3	1612.7	1623.8	1478.2	41,011	33,719	37,462	117.4	110.8	131.3

availability of content is independent of the position of the gateway in A2. However, the placement of gateway affects the extent of ph-local and remote hits in PH-BCP similarly as in PH-THCP. This enhanced locality in Scenario 2 also results in 101% higher AAT in PH-BCP compared to Default. Unlike in THCP, when a cache miss occurs at the access MR, BCP looks for other MRs that have requested the same content item, resulting in remote hit rate. This also explains why the gain of PH-BCP over BCP is smaller than that of PH-TCP over THCP.

Similarly, we also found that PH-GHCP benefits from per-hop transport connections and outperforms GHCP which in turn outperforms the Default scenario. Therefore, we conclude that architecture A3 outperforms architecture A2 irrespective of the cache selection algorithm used. Further, both architectures A3 and A2 outperform A1. Next, we evaluate which cache selection algorithm should be used for the best architecture A3.

5.6.3 Comparison of Cache Selection Protocols

In this section, we compare all the cache selection protocols for architecture A3. The simulation set up is similar to the previous sections. Table 5.4 depicts the performance of the three per-hop cache selection protocols.

PH-BCP imposes lower load at the gateway than PH-THCP and PH-GHCP. For example, in Scenarios 1 and 3, it incurs 21.5% and 17.8% lower load in bytes than PH-THCP respectively, and 17% and 10% lower load in bytes than PH-GHCP respectively. This is because while PH-BCP chooses nodes anywhere in the network that improve its performance, PH-THCP restricts itself to nodes only en route to the gateway and thus suffers from load imbalance. PH-GHCP's higher gateway load can be attributed to the following: Although we use a localized hash function, a detour always occurs when using PH-GHCP. Unlike in PH-BCP where the detour is only taken when a hit is assured, PH-GHCP has many detours that result in misses. This is due to the heavy tail of the request popularity distribution. Unpopular content is typically universally unpopular [95]. Thus a significant number of requests take a detour followed by a miss, resulting in contacting the gateway and hence higher gateway load. Interestingly, in the scenario with severe imbalance (Scenario 2), all three schemes exhibit similar load at the gateway. In this case, since the gateway is placed in a corner, all the files obtained from the gateway will also be cached around the gateway due to hop-by-hop caching. Hence, any request that is already served by the gateway can be obtained en route to the gateway before reaching the gateway itself. Hence, PH-THCP and PH-GHCP have similar gateway load as in PH-BCP.

Further, the results show that for all scenarios considered, PH-BCP provides the best throughput performance. PH-BCP also has the lowest network load across all three scenarios. Even in Scenario 2, although the gateway load is similar, the AAT of PH-THCP and PH-GHCP is lower than that of PH-BCP. This is because PH-BCP spreads more load away from the gateway when compared to the other two schemes. In summary, PH-GHCP achieves lower throughput than PH-THCP and PH-BCP from taking detours. While PH-THCP incurs no search delay or search overhead, PH-BCP can alleviate congestion better by sending requests anywhere in the network and achieves better throughput and lower network and gateway load. We conclude PH-BCP is the best cache selection protocol in A3.

5.6.4 Summary

In summary, our extensive simulations have demonstrated that per-hop transport is a viable mechanism to enable hop-by-hop caching in A3. Architecture A3 using

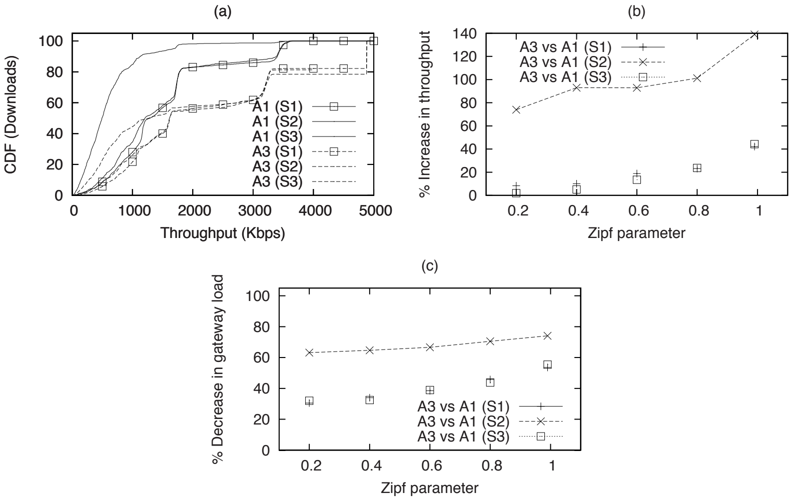


Fig. 5.8. Performance comparison of A3 and A1. PH-BCP is used for cache selection.

PH-BCP outperforms A2 by increasing the client AAT by up to 100% and reducing gateway load by up to 70%. In fact, Figure 5.8(a) demonstrates the CDF of per file throughput in A3 and A1 for the three scenarios. The CDF confirms the gain in throughput observed in A3 when compared to A1. For instance, in Scenario 2, while in A1 approximately 20% of the file downloads achieve throughput higher than 1 Mbps, in A3 50% of file downloads achieve throughput greater than 1 Mbps.

Note, however, that these results are dependent on the locality in the access pattern of the clients. As seen from the simulation parameters, the Zipf parameter used in the above study is 0.8 and is based on previous studies of caching in wireless networks [104] and measurements from web traces [12]. To study the effect of varying locality on the MeshCache performance, we vary the Zipf parameter from 0.2 to 1[5]. As seen in Figures 5.8(b)and (c), as the Zipf parameter is increased, the percentage increase in AAT in A3 vs. A1 goes up and so does the percentage reduction in gateway load. Similar to the observations above, the gain is highest in Scenario 2 due to its significant topological imbalance. Specifically, the work in [12] studied web request

[5] The larger the value of Zipf parameter, the higher the locality.

traces from a fixed group of users and found that the Zipf parameter varied between 0.64 to 0.83. For this range of Zipf parameter values, MeshCache exhibits significant improvement in throughput and reduction in gateway load.

To validate the performance and usability of MeshCache, we implemented and deployed MeshCache on our wireless mesh network testbed [111] and evaluated it extensively.

5.7 MeshCache Implementation

In this section, we describe the implementation of the MeshCache system. The MeshCache system is implemented by a user-level daemon - `MeshCacheD`. The functions of MeshCacheD are: (1) Transparently hijack web requests initiated at the clients and serve these requests from either its own cached content in case of a hit. (2) Locate the appropriate parent cache to fetch the data from in case of a cache miss. (3) When data is fetched, enable hop-by-hop transport of the data to facilitate caching at every hop. (4) Cache the content fetched for a client and maintain its freshness. We leverage Squid [110], an open source proxy cache software developed for Internet web caching, to implement the MeshCacheD. The MeshCacheD consists of three modules: (1) The *MSquid* module (MSM) is responsible for transparently hijacking client requests and serving them from its cache or the appropriate parent cache, caching and validating fetched data, and performing hop-by-hop transport. Squid software is modified to obtain the MSM. We disabled Squid's cache selection protocols and instead interfaced it with an implementation of our cache selection protocols. Further, we exploited Squid's functionality to deal with a hierarchy of caches to perform hop-by-hop caching. (2) The *Cache Selection* module (CSM) implements the cache selection protocols as described in Section 5.4 to locate a suitable parent cache given a URI. The CSM is also responsible for enabling hop-by-hop caching via the MSM. To enable both the above functions, the CSM exports **FindParentCache(givenURI)** interface to the MSM and also interacts with the underlying routing protocol via

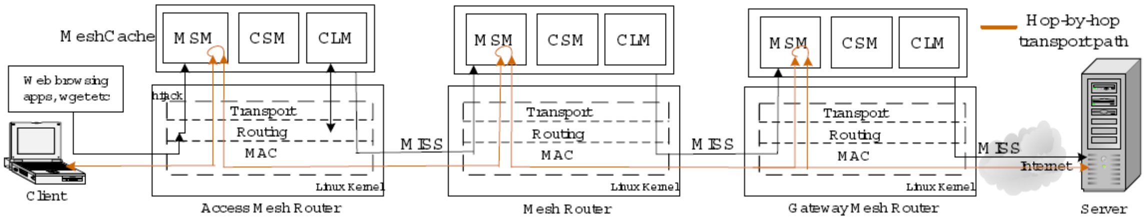


Fig. 5.9. MeshCache Implementation.

the *Cross-Layer* module (CLM). (3) The CLM is responsible for communicating with the underlying routing protocol to obtain information and export this information via known APIs to the other modules in MeshCacheD. Our current prototype implements these APIs as functions calls that operate on routing table information continuously updated to the /proc virtual file system by the routing protocol in the OS kernel. The virtual file system provides a shared memory interface between MeshCacheD CLM and the underlying routing protocol in the kernel. The information shared contains the current best next hops for known destinations, path metric to a destination node (e.g. accumulated ETX of all links) and the best known gateway with path metric. The CLM exports the following APIs on behalf of the routing protocol to other MeshCache modules: *BestGateway()*, *BestGatewayMetric()*, *BestNode*($< nodeIPlist >$) and *GetNextHopForNode*($< nodeIP >$).

In the following sections, we describe how the MeshCacheD implements architecture A2 using THCP and A3 using PH-THCP and PH-BCP. A2 is implemented to obtain a comparison point.

5.7.1 THCP Implementation

Each mesh router runs the MeshCacheD. Referring to Figure 5.9, THCP works as follows: (1) An unmodified client applications in the WMN generates a web request for a URI X.

(2) The request for X is routed through the access mesh router (AMR) for this client. (3) At AMR, the client request is transparently hijacked by the MSM using the standard Linux Netfilter [108], and queuing it up for MeshCache processing using *libipq*. (4) The MSM now captures all further communication from this client. This enables the MSM to parse the client request to locate the data item requested. (5) The MSM now checks for X in its cache. If there is a cache hit, X is served to the client from the AMR cache. (6) In case of a cache miss, the MSM issues a **FindParentCache(X)** call to the CSM. (7) Since THCP does not exploit cached objects at other mesh routers, the CSM simply returns the best gateway obtained from the CLM. (8) Upon returning from the FindParentCache() call, the MSM now contacts the gateway to fetch X. (9) The gateway either serves the file from its cache or retrieves it from the origin server. The data is now returned to the AMR that serves the data back to the requesting client. (10) AMR also caches X in its cache and maintains its freshness information for future requests.

5.7.2 PH-THCP Implementation

Referring to Figure 5.9, the PH-THCP scheme works as follows: (1) Steps 1 through 6 for the PH-THCP scheme are identical to that in the THCP scheme described above. (7) To implement PH-THCP, the CSM extracts the origin server's IP address (SIP) from the URI. It then issues a **GetNextHopForNode(SIP)** call to the CLM. (8) The CLM communicates with the routing protocol to obtain the best next hop for SIP based on the metric currently employed by the routing protocol. If the SIP is an IP address outside of the current WMN, all the messages to the SIP need to be routed via the gateway (GMR). Thus, essentially, the next hop returned to the CSM in this case will be the best next hop for GMR. (9) When the CSM receives the best next hop for SIP, say MR 1, it returns it as the best parent cache to the MSM. (10) The MSM now contacts the MSM at MR 1 requesting X. Note that the choice of MR 1 is **not** made after ensuring that MR 1 has X. (11) Thus, when the

MSM at MR 1 receives the request for X, it may have a cache hit or a miss. In the event of hit, X is returned to AMR, which in turn serves the data to the client and also caches X for future use. (12) In case of a miss, MR 1 repeats the process from steps 6-11 described above. Let us assume that the next hop for SIP from MR 1 is MR 2. If MR 2 has a cache hit, the data will now be served from MR 2 to MR 1, which caches the data and further serves it to AMR, which once again caches the data and serves it to the client requesting X. Thus, the CSM exploits Squid's behavior when contacting parent caches to perform hop-by-hop caching. (13) In case of a miss at MR 2, the process repeats itself until the request is received by the gateway itself. (14) The gateway then either serves the file from its cache or retrieves it from the origin server and returns it to the client, and each router along the way caches the file.

5.7.3 PH-BCP Implementation

Referring to Figure 5.9, the PH-BCP scheme works as follows: (1) Steps 1 through 6 for the PH-BCP scheme are identical to that in the THCP scheme described above. (7) The CSM maintains a one-to-one mapping between a URI and the corresponding IP address of the mesh router that has a cached copy of the content. This mapping is maintained with a refresh timeout. (8) When the CSM receives the **FindParentCache(X)** call, it consults its mapping table to determine if a corresponding IP address is found. (9) When the CSM finds no such mapping, it begins the PH-BCP cache selection protocol. For example, when AMR receives the request for X for the first time and experiences a cache miss, its corresponding CSM will also fail to locate a mapping for X. (10) To implement PH-BCP, the CSM extracts the data item X from the URI. It then sends a UDP *content locate* message to the broadcast IP address along with a gateway path metric calling the **BestGatewayMetric()** CLM function. (11) Any mesh router receiving the *content locate* message checks its cache to see if it contains X. In case of a hit, the mesh router unicasts a *content found* message back to AMR. All mesh routers also rebroadcast the *content locate* message to their neighbors until the metric in the message exceeds the path metric to the gateway. (12) AMR collects all the *content found* messages received in a time period of *timeout* seconds. It then picks the mesh router that maximizes the throughput out of all routers with a cache hit. Let us denote this node as FIP. Note that if AMR does not obtain any *content found* messages, it picks GMR as the node containing X. (13) It then calls **GetNextHopForNode(FIP)** and obtains the next hop for FIP, say MR 1. It returns MR 1 to the MSM. (14) The CSM also makes a note of mapping between X and FIP in its mapping table. It also send a *setup* message to MR 1 with the mapping information. (15) When MR 1 receives the setup message, it finds the best next hop for FIP and forwards the message to that mesh router. It also makes a note of this mapping in its mapping table. This process is repeated till it reaches a node such that the next hop for a message reaching FIP is FIP itself. (16) Now, the MSM of AMR contacts the MSM of MR 1 for X. MR 1 gets a cache miss and hence contacts its CSM. The CSM will now find a mapping for X created by the setup message sent by AMR. It then gets the best next hop for FIP and returns it to the MSM. (17) This process continues till FIP is contacted and a cache hit is obtained. FIP then transfers X which is pipelined till the client. Effectively, all the intermediary mesh routers cache the content thereby enabling hop-by-hop caching. (18) When X is not found in the WMN, the request reaches GMR and GMR contacts the origin server.

5.7.4 Cache Consistency

Since MeshCache uses the Squid code base, we rely on the inbuilt mechanisms of Squid to ensure object consistency. Objects downloaded in HTTP use HTTP headers (e.g., in HTTP/1.1) to specify an expiry time or other types of freshness information which is used by MeshCache on each router to determine when objects are stale. Since HTTP is likely to be the most widely used object retrieval protocol used over MeshCache, we focus on it in this chapter. If some proprietary protocol (e.g., p2p sharing) is used fetch objects then that protocol will need to specify expiry information for data items to work with MeshCache, or some default timeouts will be required.

5.8 MeshCache Performance

In this section, we evaluate the performance of MeshCache over a deployed wireless mesh network testbed.

5.8.1 Testbed Setup

Our testbed, MAP (Mesh@Purdue [111]), currently consists of 32 wireless mesh routers (small form factor desktops). During the time of the experiments in this chapter, only 15 routers had been deployed and each mesh router was equipped with a single wireless card. 9 nodes in the MSEE building were equipped with Atheros 5212 based 802.11a/b/g cards while 6 nodes in the adjoining EE building were equipped Senao Engenius 2511 802.11b wireless card. We configured the entire network to operate in 802.11b mode. Each radio is attached to a 2dBi rubber duck omnidirectional antenna with a low loss pigtail to provide flexibility in antenna placement. Each mesh router runs Linux kernel 2.4.20-8 and the open-source *hostap* and *madwifi* drivers are used to enable the wireless cards. IP addresses are statically assigned. The wireless cards we use can support a wide range of power settings (up to 200mW). We used them in their default operational mode.

Figure 5.10 shows the layout of nodes in our testbed. The testbed deployment environment is not wireless friendly, having floor-to-ceiling office walls instead of cubicles as well as some laboratories with structures that limit the propagation of wireless signals. Apart from structural impediments, interference exists in our deployment from other 802.11b networks (the Purdue Airlink network). We used channel 11 of 802.11b to operate our network since it was the band furthest away from those being already

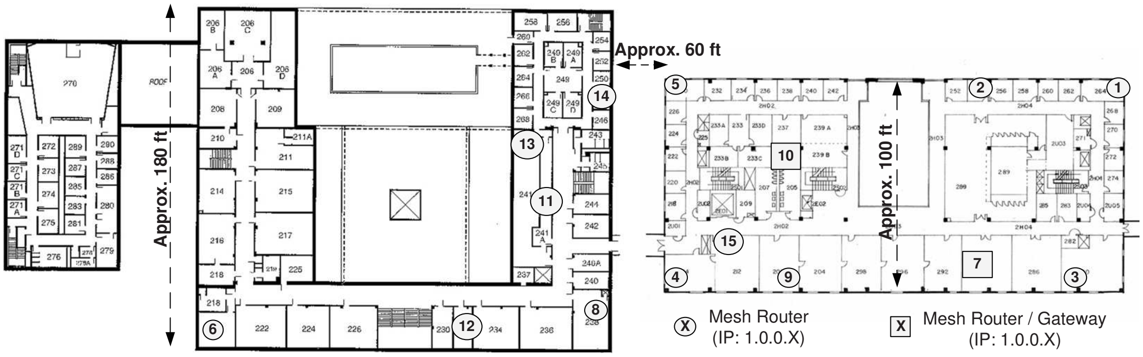


Fig. 5.10. Deployment of the MAP testbed used for MeshCache evaluation. A subset of the nodes in the current MAP testbed where used for experiments.

used in the deployment environment. In summary, the environment is highly variable and many different paths exist between nodes.

The routing protocol used for the experiments is MAP-DV, which we implemented through modifications to the kernel AODV routing protocol implementation [46] to incorporate proactive gateway discovery [25], the ETX metric [21] and ETT metric [29] and the interface to the CLM. We used *netperf* measurements to confirm that MAP-DV provides better throughput than vanilla AODV.

We compare the performance of the three MeshCache architectures using our testbed. Architecture A1 is again referred to as *Default*. Architecture A2 is represented by THCP while both PH-THCP and PH-BCP are included for architecture A3. We choose a deployment where MR 7 is the gateway node with access to the Internet while 10 out of the remaining 14 nodes are chosen as traffic sources. Each source is driven by a synthetic web trace obtained similar to in the simulation. Also, the cache size in each mesh router is set to 2 MB due to the working set size and length of the experiment. Each experiment consists of clients making requests from the trace through an unmodified *wget* client for a period of 1 hour. The experiments were repeated once every day for one week and the results averaged. Throughput per file transfer is used as the metric for this study.

5.8.2 Performance Results

We first performed measurements to quantify the interactions between per-hop transport and the underlying routing protocol. Specifically, we want to measure whether fixing of routes in per-hop transport leads to poor adaptation to network conditions by not allowing route changes for the duration of a transfer. We measured if route changes occurred in an interval of 30 secs[6] between nodes 5, 1 and 7 using a ping with the record route option every second over a period of 12 hours. We found that for node 5, routes remained the same for 92% of the intervals while 99% of the intervals resulted in same routes in the case of node 1. This shows that it is unlikely that fixing routes in per-hop transport for an interval of 30 seconds will result in poor link adaptation. The time scales of route changes are longer than the time a route is fixed in MeshCache (i.e. the time to download a typical content item).

We now describe our throughput performance evaluation experiment. In this experiment, the nodes generating traffic can be separated into nodes that are one hop away and those that are more than one hop away from the gateway for ease of explanation. Thus, MRs 2, 3, 4, 9, 10 and 15 are one hop away from MR 7 while MRs 1, 5, 8 and 12 are more than a hop away from the gateway.

The performance of the MRs that are one hop away is depicted in Figure 5.11. For each MR, the throughputs obtained per file transfer are sorted and plotted. Due to lack of space, the performance of 4 such nodes is depicted. When THCP scheme is employed, each MR benefits from the locality in its own access pattern, i.e., when the MR receives a request for a content item that it has fetched in the past and hence cached, the MR can serve the request from the cache and thus obtain a significant improvement in its throughput. For example, for node 15, compared to the default scheme, 20 transfers benefit from caching at the access MR itself. Note that the

[6] A sample download time for content.

throughput for a cache hit is very high (about 50 Mbps) and hence is not depicted in the graph. Similar benefit is observed in all the nodes when THCP is employed. For the one-hop nodes, performance using PH-THCP is largely similar to using THCP since when they do not have a local hit, they have to traverse only one hop irrespective of the cache-selection algorithm. However, the one-hop nodes can benefit from PH-THCP when any MR whose route to the gateway passes through them requests for an object that they will request in the future. In this case, when the node whose route passes through them fetches the object from the gateway, the object will be cached at this node. Node 15 shows the presence of gain from PH-THCP as it exploits the locality in the traffic pattern of itself and its neighbors. An additional 10 transfers benefit from PH-THCP as compared to THCP in node 15. Similar benefit is also observed in the other nodes. Thus, with PH-THCP, some transfers that take one hop to be served can now be served from the cache on the MR itself. The performance of PH-BCP is expected to be similar to that of PH-THCP for the one-hop nodes since PH-BCP cannot further reduce the number of hops required to serve the content to less than one. This effect is observed in nodes 3, 4 and 10. Interestingly, node 15 shows significant benefit when using PH-BCP across all transfers. This is because compared to nodes 10 or 3, the connection of node 15 to gateway 7 passes through several walls, resulting in increased attenuation of the signal and hence reduced throughput. PH-BCP enables MR 15 to choose other potential candidates for a content item (say 4, 9 or 10) which provide much higher throughput than the one hop to the gateway. Note that MR 4 does not experience any further improvement with PH-BCP as it is unable to locate a better candidate to obtain the file from.

The performance of the MRs that are more than one hop away is depicted in Figure 5.12. Similar to in the 1-hop case, the performance of the MRs using THCP is better than the default case. Once again, the nodes exploit the locality in their own access pattern. For example, THCP improves the performance of 20 transfers in MR 5. Further, PH-THCP has the potential of reducing the number of hops required to obtain a content item up to 1, thereby improving the throughput. Note

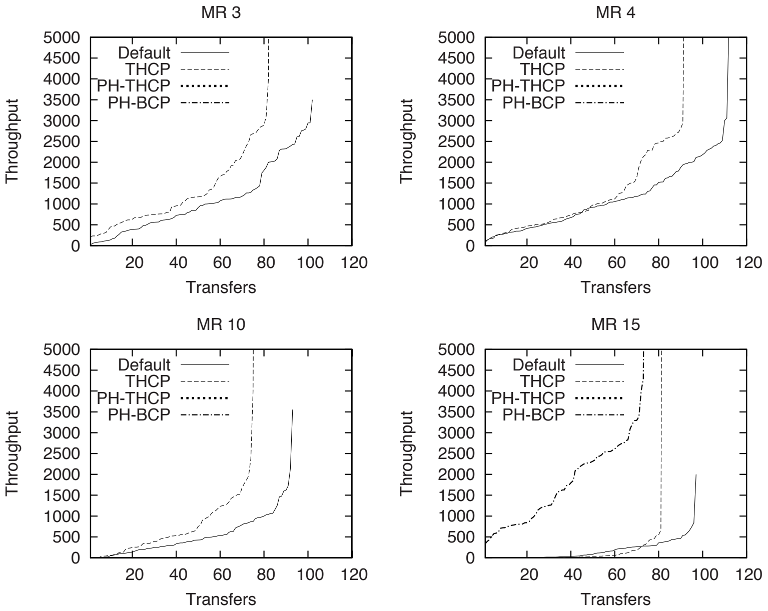


Fig. 5.11. Download performance of individual mesh routers one hop away. MR 7 is the gateway.

that similarly as in the one-hop case, these nodes will benefit when the nodes whose routes pass through them request for files that will be requested by these nodes in the future. This effect is observed in all the nodes that are two hops away. For example, MR 5 improves the performance of 10 transfers using PH-THCP. Finally, using PH-BCP further improves the performance of these nodes. For example, nodes 12 and 8 benefit from using PH-BCP by obtaining content from MRs like 4, 15 or 9 instead of the gateway 7. Also, PH-BCP enables searching all the nodes in the vicinity of a mesh router instead of only those that are en route to the gateway as in PH-THCP. Note that MRs 1 and 5 do not obtain further benefit from PH-BCP as they have limited number of neighbors (nodes 2 or 3 for MR 1, node 10 for MR 5). One among the potential neighbors is already considered by PH-THCP and hence PH-BCP does not have different potential candidates to obtain additional benefit.

The performance of the network using the MeshCache system is summarized in Figure 5.13(a) using a CDF of the total transfers in the network. In summary, in the

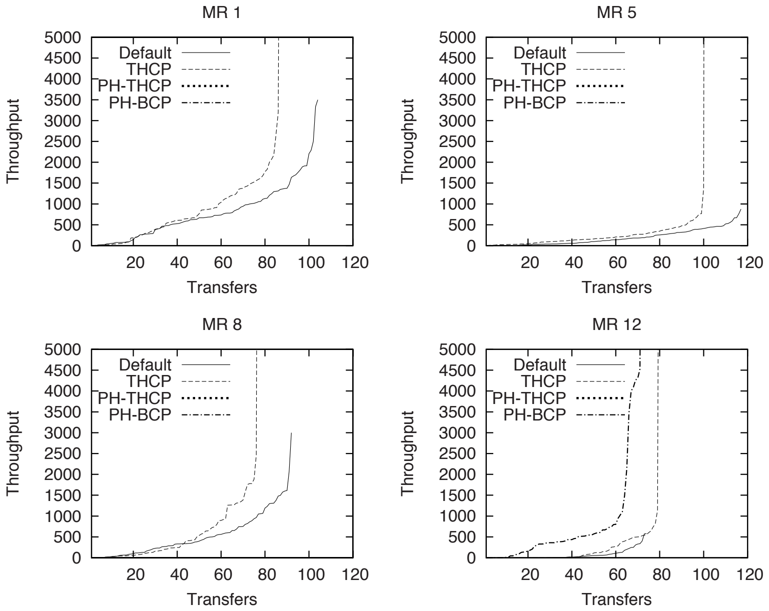


Fig. 5.12. Individual download performance of mesh routers more than one hop away. MR 7 is the gateway.

absence of caching in the network, only 20% of the transfers have a throughput greater than 1 Mbps. When a simple scheme like THCP is used, 40% of the transfers have a throughput greater than 1 Mbps. PH-THCP results in 50% of the transfers having a throughput greater than 1 Mbps. Finally, using PH-BCP results in almost 60% of the total transfers across all the nodes having a throughput larger than 1 Mbps. Figure 5.13(b) also shows that the average load per node as well as the gateway load is reduced by using MeshCache. PH-BCP reduces gateway load the most out of all the schemes.

Finally, we observed that the latency of transfers did not suffer despite the search overhead of PH-BCP. The average and median latency were in fact lower than without MeshCache (by around 5-10%). This is because without cooperative caching, packets suffer queuing delays as they get closer to the gateway which more than even out the extra delay incurred from cooperative caching, except in networks with very low load.

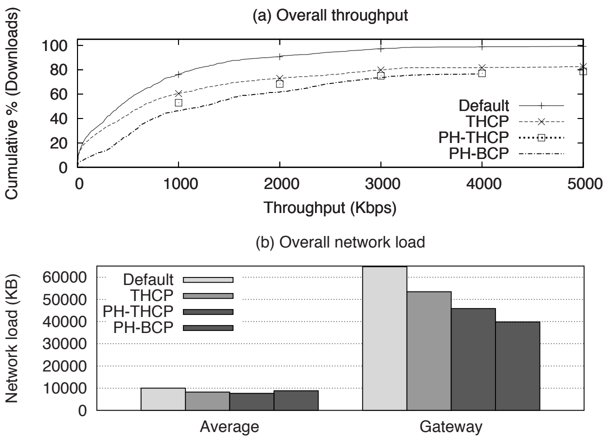


Fig. 5.13. Overall download performance of mesh routers.

5.9 Related Work

5.9.1 Wireless mesh networks

Many design issues of WMNs have been recently studied [8,11,29,44,51,78,79] and many companies are offering products for deploying WMNs [10,61]. However, research in WMNs has primarily focused on new routing protocols, improving medium access, as well as new designs for physical layer technology. To the best of our knowledge, no previous work specifically explores incorporating content caching to improve the performance of WMNs.

5.9.2 Caching in wireless networks

Content caching has been previously proposed for improving performance in *mobile* ad-hoc networks [33, 35, 82, 104]. However, MeshCache is different from these previous approaches since it is designed for static wireless mesh networks and aims to provide a transparent infrastructure-based solution to exploiting locality. In addition, MeshCache is a complete system design that incorporates cross-layer communication to enable transparent caching. Previous work has primarily focused on simulations, neglecting the practical considerations in deploying a caching infrastructure. For example, previous work assumes that files can be easily stored as they pass through the routing layer whereas this is not possible without a per-hop transport architecture or major modifications to client applications and network layers of all mesh routers.

5.9.3 Transport protocol enhancements

There have been numerous enhancements to TCP and even totally new transport protocols proposed for mobile wireless ad hoc networks. In this chapter, we evaluated MeshCache using the widely used TCP protocol. However, MeshCache is an application layer system that will work with any enhanced transport protocol as well. The per-hop transport technique used in MeshCache should benefit any new transport protocol by providing them with quick congestion detection and response as well as quick error recovery. The technique of splitting an end-to-end TCP connection into multiple ones (Split TCP) has been previously proposed to improve throughput and fairness in mobile wireless ad hoc networks [48]. However Split TCP is designed to deal with mobility and only splits the connections at a few intermediate nodes. MeshCache splits each hop of the transport connection to enable caching. Split TCP connections also naturally arise in overlay networks in the Internet [7,49,62,88]. However, the overlay split TCP is fundamentally different from the per-hop transport in MeshCache since in the Internet, the original end-to-end path may be totally different from the overlay split transport path. In this case, the overlay path can be better or worse than the end-to-end path. In MeshCache, the per-hop and end-to-end transport session both operate over the exact same underlying network path. Thus, the per-hop transport in MeshCache cannot worsen the network path used. Packet retransmissions in MeshCache also take place from the closest possible point in the path unlike in overlay networks where the per-hop connections span multiple Internet routers.

5.10 Summary

In this chapter, we proposed the MeshCache system for leveraging storage devices and exploiting the locality in client request patterns in a wireless mesh network. The MeshCache system alleviates the congestion bottleneck that commonly exists at the gateway node in WMNs while providing better client throughput by enabling content downloads from closer high-throughput mesh routers. MeshCache is loosely coupled with the underlying transport and routing protocols to maximize deployability and exploit cross-layer information awareness. Through simulation and testbed experiments with a deployed implementation, the hop-by-hop caching mechanism coupled with PH-BCP cache selection in MeshCache was shown to be an effective technique to improve WMN performance.

6. ENABLING SERVICE PROVISIONING IN WIRELESS MESHES

6.1 Introduction

In contrast to other multi-hop wireless networks that aim to provide connectivity in the face of constrained energy (sensor networks) or mobility (mobile ad hoc networks), the focus of a WMN (wireless mesh network) is quite different: A WMN aims to be a last-mile technology and thus it must compete with existing last-mile technologies. An important feature of other last-mile technologies (such as cable and DSL) is a "bitrate-for-bucks" service model, i.e., a client is typically promised a bandwidth she pays for, such as 128Kbps for $24.99/month. Thus, a fundamental requirement for WMNs is to provide clients with similar service options[1]. This requirement is also key to the adoption of WMNs. Even in places where other access technologies are unavailable or costlier, such a service model is essential to build long-term customer loyalty. The key challenge for such service provisioning is to provide guarantees to the clients of a WMN, especially in the presence of the lossy wireless environment and multi-hop routing in WMNs.

In this chapter, we assume that the bitrate-for-bucks promised in a deployed WMN are always achievable by the theoretical capacity of the network [2]. In the absence of such a condition, it is infeasible to provide any guarantee to WMN clients. Given that theoretical capacity exists to support the promised service, we explore the effectiveness of various techniques to provision the "bitrate-for-bucks" service model in WMNs. Wireless meshes are typically constructed using off-the-shelf 802.11 radios

[1] Enterprise deployments of WMNs where there is no such charging mechanism is outside the scope of this chapter.

[2] The capacity value assumes perfect scheduling by an omniscient entity which may not be attained by real protocols.

since such devices are cheaper due to economies of scale. We thus first explore the service that can be provided using a WMN built from off-the-shelf hardware and publicly available software and protocols. We find that the performance of off-the-shelf 802.11 is inadequate for our target service model. This inadequacy is tied to the fact that the 802.11 MAC layer fundamentally is oblivious of fair access to the wireless channel for multi-hop flows.

In search for a good solution, we investigate a theoretically optimal approach termed Fair Scheduling (FS) that assumes that a time-slotted architecture for medium access exists, and performs scheduling such that fair service guarantees are met while exploiting the available parallelism in the network through spatial reuse. We find that this approach with modifications to account for practical limitations can indeed provide a "bitrate-for-bucks" service model for specific network topologies and traffic patterns. However, we also find the assumptions made in the centralized scheduling idealistic and limiting the applicability of the solutions for real WMN deployments.

Finally, we propose the **APOLLO** system that leverages off-the-shelf hardware with software modifications to provision the "bitrate-for-bucks" service. APOLLO seamlessly integrates three synergistic components: (1) A theory-guided service planning and subscription technique to decide how to admit new subscribers. (2) A rate-based admission control to restrict the upload/download traffic from clients in accordance with their service plans. Importantly, APOLLO uses immediate admission control at the access mesh router of the client such that all packets that enter the mesh backbone network are "good" packets that should receive service. This ensures that the traffic in the network is below the capacity of the network. (3) A novel distributed light-weight fair scheduling scheme to deliver the admitted traffic that is robust to unfairness that can arise from interference, variability in wireless channel quality, collisions, etc. This approach rests on the fundamental observation that all such phenomena are concisely represented by an increase in the *queue length* of the affected node. APOLLO prioritizes access to the wireless channel in interference neighborhoods [3] based on the nodes' queue lengths, i.e., nodes with higher queue lengths receive transmission priority. This approach robustly deals with unfairness (and consequent queue backlog). Compared to previous time-slotted fair scheduling approaches [80, 83], APOLLO does not need to schedule particular flows on specific nodes in specific time-slots since all packets in all queues are "paid-for" packets and nodes cooperate and prioritize among each other to get these packets to their destinations.

Our evaluations show that APOLLO can effectively provide a client with the service it paid for in many different network topologies and traffic scenarios. A deployment of an APOLLO implementation over a wireless testbed also demonstrates that APOLLO works well in real-world scenarios. We believe that the use of APOLLO and a "bitrate-for-bucks" service model will aid in the widespread use and adoption of community wireless networks.

In summary, the contributions of this work are: (1) We introduce the important problem of practically provisioning a "bitrate-for-bucks" service model in WMNs which is important for their commercial adoption. (2) We demonstrate that both off-the-shelf and theoretically optimal scheduling based approaches are inadequate for implementing such a service model. (3) We design and implement the **APOLLO** system that provides such a service model and demonstrate its performance through detailed simulations and a real deployment. APOLLO uniquely proposes using admission control in WMNs as a *necessity* since (1) wireless channels have limited capacity which renders overprovisioning of the backbone network difficult; (2) shared wireless media access provides no built-in link-layer support for restricting traffic from greedy clients.

Admission control also has favorable implications towards mesh network management since it also enables robust, scalable and secure operations in a mesh by allowing flexible placement of client population, allowing planned deployment, uptime and downtime in a mesh, and preventing DDOS attacks.

[3] A set of nodes whose transmissions interfere.

The remainder of the chapter is structured as follows. Section 6.2 describes our basic network setup. Section 6.3 explores the use of off-the-shelf solutions while Section 6.4 explores the use of a optimal scheduling solution for service provisioning. Section 6.5 presents the design and implementation of our proposed system APOLLO. Section 6.6 reports on our experience with APOLLO. Finally, Section 6.7 summarizes related work and Section 6.8 concludes the chapter.

6.2 Architecture

This section describes our WMN architecture. We consider a typical wireless mesh network with omnidirectional antennas, in which mesh routers are placed on the rooftops of clients [107] or other infrastructure (e.g., streetlights), and are interconnected via wireless links. One or a few gateways are connected to the Internet and propagate traffic to and from clients. The gateways are not widely deployed due to cost and uplink constraints. Clients can be mobile or static and each one is associated with a mesh router (at a given time). We assume mesh routers communicate with their associated clients using different radios/channels from those used to talk to other mesh routers. In fact, the WMN may even use multiple radios for the backbone links. Further we assume that the WMN employs a 802.11 MAC layer.

Such a WMN provides Internet access as follows: Each client's packets are first received by the client's *access mesh router*, i.e., the mesh router the client's interface is associated with. These mesh routers then forward the packets to the *gateway mesh router* (GMR) using other mesh routers. The GMR provides Internet connectivity through a high bandwidth wired/WiMax interface. The gateway may perform other functions such as IP address assignment or NAT. All the MRs use a routing protocol (e.g. OLSR [20]) with metrics such as ETX [21] to find routes to each other and to the gateway. The WMN can also be used for peer-to-peer (P2P) traffic between any two clients. In that case packets are not sent to the gateway, but they are routed from the sender client's access mesh router to the receiver client's access mesh router using the same routing protocol used for Internet access.

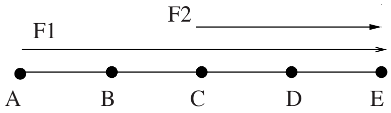


Fig. 6.1. Sample network topology.

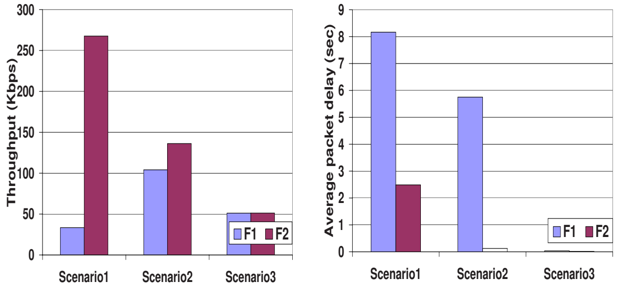
6.3 Exploring "Off-the-Shelf" Solutions

We first explore how current popular off-the-shelf solutions (e.g., 802.11 with ETX routing) work in terms of the service they provide to individual clients. We study the topology in Figure 6.1 using the Qualnet simulator [74]. In this figure we have 5 nodes, namely A, B, C, D, E, in a chain. All links are of the same distance, have similar loss rates of approximately 10% and each node is within transmission range of its one-hop neighbors. We use the 802.11b MAC layer with a rate of 2Mbps. Nodes A and C each generate one UDP [4] flow to the gateway node E. The per client service plan assigned in this topology is 140Kbps for a given price. We ascertained that the network can support this service plan (see Section 6.5.1). Further, we assume that the service plan is always chosen to be close to the capacity of the network. Overprovisioning a network is not economically viable.

Figures 6.2(a)-6.2(b) show the throughput, and average packet delay for each of the two flows in three scenarios: (1) Users demand more than the service plan (both transmit at 300Kbps) since this solution has no enforcement of service plans; (2) Users demand their service plan (both transmit at 140Kbps); and (3) Users demand less than their service plan (both transmit at 50Kbps).

From these three figures we make the following observations about the performance of off-the-shelf solutions: (1) When the flows are allowed to transmit whatever they desire, the network is overutilized, and the service provided is grossly unfair, i.e., one user ($F1$) gets less than their service plan. (2) This unfairness is also observed when the flows transmit as per their service plans and the network is operated near

[4] We use UDP to measure performance since it does not provide a conforming workload. This choice is widely used in wireless network studies.



(a) Throughput (b) Average packet delay

Fig. 6.2. PDR, throughput and delay of 2 flows in the sample network topology.

capacity in scenario 2. This unfairness becomes more prominent if we look at the average packet delays of the two flows. The average packet delay for flow $F2$ is 123 msec, but for flow $F1$ it is 5.75 sec, that is 47 times larger. (3) Finally, when the flows operate below their service plans, the network is underutilized and off-the-shelf solutions work fine because the occurrence of inefficiencies in accessing the medium are outweighed by the time the medium is idle.

Thus, off-the-shelf approaches are inadequate for our target service model and only work when the network is underutilized. When the network is utilized well or overutilized, 802.11 is unable to allocate service in an acceptable manner. The lessons learnt from this experiment are: (1) policing the traffic allowed to enter the network is a *necessity* in order to provide any guarantees as per service plans; (2) service plans can be offered by highly over-provisioning the network which however is not economically viable; (3) operating the network within (close to) network capacity, e.g., by policing the admitted traffic, alone is insufficient due to unfairness and additional mechanisms are needed to mitigate unfairness. With this in mind, in the next section we investigate a TDMA [5] approach for service provisioning that is theoretically optimal (under a set of assumptions).

[5] Assuming a new MAC layer can be deployed.

6.4 Exploring "Optimal Scheduling" Solutions

As an example of a theoretically optimal solution we consider "Fair Scheduling" (FS) for wireless mesh networks. We first provide a brief background of FS.

6.4.1 Fair Scheduling (FS)

FS assumes knowledge of routing paths (a tree from the gateway to all clients) and uses a scheduling algorithm based on spatial TDMA [65] to schedule transmissions of mesh routers such that all clients in the network receive a fair share of the bandwidth. The specific example technique for FS used in this chapter is from [83] and consists of three phases: compatibility matrix (CM) construction, clique enumeration, and clique selection.

The algorithm starts with the construction of a compatibility matrix (CM) which denotes which links in the network can transmit simultaneously. $CM[ij] = 1$ if links i and j do not interfere and can transmit simultaneously and $CM[ij] = 0$ otherwise. In [83] the CM is constructed based on the following rule: "Two links interfere if the sender of one link is within transmission range of the sender or the receiver of the other link". The algorithm then enumerates all possible cliques [6] in the CM. Links in the same clique can transmit together to exploit spatial reuse.

Let Cl^{α}_k denote the α-th clique of cardinality k and L^{α}_k denote the most loaded link in Cl^{α}_k. The load of link L_{AB} $l(L_{AB})$ is defined as the number of clients that use this link to transmit traffic to or from the gateway. Let also d^a_k denote the number of slots required to transmit the traffic of the most loaded link in clique Cl^{α}_k. Then $d^a_k = l(L^{\alpha}_k)$. Each link L_{AB} in Cl^{α}_k is activated during $l(L_{AB})$ time slots and is idle during $d^a_k - l(L_{AB})$ time slots. Hence, the clique Cl^{α}_k generates a gain $g(Cl^{\alpha}_k) = (\sum_{L_{AB} \in Cl^{\alpha}_k} l(L_{AB})) - d^a_k$.

[6] While the problem of clique enumeration is known to be NP-hard, computation is done with the Bron-Kerbosch [13] heuristic.

A scheduling is defined as a set of cliques s that fulfills the following two conditions: (i) all links that are part of the tree are included in the schedule and (ii) each link is included only once. Finding a schedule s with the minimum cycle length T_s ($T_s = \sum_{Cl^{\alpha}_{k} \in s} d^{a}_{k}$) provides a fair share while maximizing throughput. Instead of finding a minimum length schedule by exhaustive search, the authors in [83] propose a greedy heuristic according to which they maximize the total gain (g_s) of the schedule instead. The length of the calculated schedule in turn dictates the throughput (bitrate) that can be assigned to each client.

How well does FS perform in practice? To see how efficient FS is in practice, we ran it over the topology in Figure 6.1. While the length of the calculated schedule dictates a service plan of 140Kbps for both flows, surprisingly, as shown in Figure 6.3, we found that $F2$ achieves a throughput close to the service plan (118Kbps) while $F1$ starves (58 Kbps). Most importantly, FS performs worse than 802.11. Thus, theoretically optimal solutions directly applied do not work well and the next section identifies why.

6.4.2 Improving FS

We identified two practical problems related to FS that made it perform badly in practice: simplified interference estimation and ideal link assumption.

Simplified Interference Estimation The interference rule in FS is not enough in a realistic environment. A recent work [68] has shown that simplified heuristics (2-hop interference) usually fail to model interference accurately and a measurement-based approach is more accurate. In reality, a node can be affected by a transmission even if it is at a distance more than twice the transmission range from the sender. Although packets cannot be properly received at such long distances, they can still collide with other transmissions.

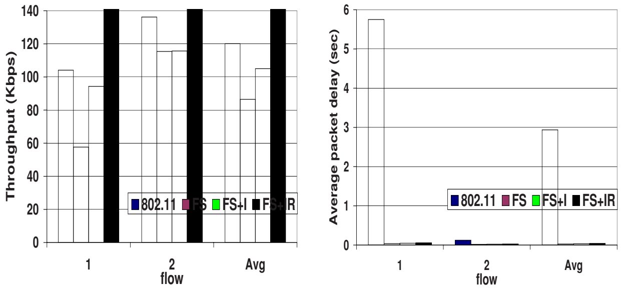
To take interference into account, we incorporated the BIR metric [68] into FS by setting $CM[ij] = 0$ if $BIR_{ij} < 0.9$ [7] (we call this $FS + I$, i.e., FS with Interference-awareness). The BIR metric can take any value between 0 and 1 (quantifying the real interference between two links) and is estimated through offline measurements.

Ideal Link Assumption FS, similar to many other scheduling algorithms in literature [54–56], assumes that the only reason for packet loss is packet collisions when neighboring nodes transmit simultaneously and scheduling avoids these losses. However, in a real network, a second important reason for packet loss is the time variability of the wireless channel. Reflections, scattering and other factors cause multipath fading, which may randomly deteriorate the quality of the links. This causes random packet loss which can severely harm performance. 802.11, although still unfair, provides higher throughput than FS because it includes a reliability mechanism (ACKs and retransmissions) that helps nodes recover from such random packet losses.

We added MAC layer retransmissions to FS by making the sender wait for a certain amount of time in each slot after each packet transmission for an ACK. If no ACK comes, the sender retransmits the packet in its next time slot. A maximum number of retransmissions is allowed for each packet, after which the packet is dropped. This number is a tradeoff between reliability and throughput. We call this the version of $FS + IR$, i.e., FS with both Interference-awareness and Reliability.

Performance of improved FS We reevaluated FS with our proposed fixes and the results are shown in Figure 6.3. While FS scheduled node A and D together, $FS + I$ schedules them separately since it figures out that these nodes interfere (their transmissions collide at node B). However, the performance of $FS + I$ is still affected by packet loss. After incorporating both fixes, $FS + IR$ provides an almost optimal fair service to both flows; both flows achieve a PDR higher than 99% and a throughput of 140Kbps, equal to their demand. Most importantly, this is achieved with only

[7] BIR_{ij} is measured as the ratio of total packet delivery rates over links i and j when both are transmitting over when transmitted individually.



(a) Throughput (b) Average packet delay

Fig. 6.3. Performance comparison of 2 flows in the sample network topology for 802.11, FS , $FS+I$ and $FS+IR$.

1 retransmission attempt per packet, while 802.11 has much worse performance even with 7 retransmissions. Hence, assuming there is an oracle that has all the necessary information to calculate the optimal schedule all the time, FS enhanced with interference-awareness and a simple reliability mechanism can provide the service model we target.

6.4.3 Practical problems with FS

In practice, however, there are a number of fundamental challenges in implementing a TDMA-based scheme like $FS + IR$ which effectively prevent such a solution from becoming deployable. In the following, we summarize these implementation challenges.

Real-time flow information, schedule calculation and propagation: Since the optimal schedule in a TDMA approach is directly related to the flows present inside the network, i.e., going through each node, and hence information about the arrival and departure of flows and changing traffic has to be propagated to the scheduler, which has to compute a new schedule and propagate it to all the nodes, all in a timely fashion. The reliable propagation of flow information and the updated schedule is not a trivial task. Also, requiring all nodes to switch to the new schedule at the same moment (in order to avoid collisions) requires global synchronization.

Dependence on routing path: The optimal TDMA schedule is directly dependent on the routing paths used by the flows. Even if a node has a single associated flow, routing path changes (which are frequent in wireless networks) necessitate frequently changing schedules.

Dependence on fine-grained time slot and synchronization: In simulation, the time slot can be easily set equal to the time required to transmit one packet. Such a fine-grained time slot is difficult to achieve in practice. Hence papers that offer implementations of time-slotted systems always use a much larger time slot that allows for many packet transmissions (e.g., [66,80]), which cannot offer maximum performance achieved by fine-grained time slotting. Further, tight time synchronization among the wireless mesh routers is a challenge in itself.

Link-quality adaption: Finally, it is difficult to incorporate effective link-layer adaption techniques such as rate adaptation into the scheduling algorithm since link quality is a highly dynamic quantity and even if known, would make the schedule computation very expensive.

6.5 APOLLO: Practical Service Provisioning

Given the performance problems with off-the-shelf approaches and the fundamental problems in adopting fair scheduling for real deployment, we propose the APOLLO system which retains the simplicity of off-the-shelf approaches (by being implemented on top of cheap 802.11 radios) and performs practical scheduling of packet transmissions. One key property makes APOLLO practical: *its scheduling is oblivious to the number, the source and destinations of multi-hop flows present in the network, and the routing paths by these flows, and it does not require fine-grained time synchronization* [8]. This section presents the design (theoretical formulation for service planning, admission control for enforcing service, lightweight scheduling to deal with unfairness) and implementation of the APOLLO system.

[8] APOLLO provides a bitrate guarantee similar to Cable/DSL and while it improves delays, strict per-packet delay guarantees would require more complex scheduling. It is not clear if it is practical to implement fine-grained scheduling in multi-hop wireless networks.

6.5.1 Service Planning

The foremost question for service provisioning is: If a new client wants to subscribe to the network, how can we decide if the client's desired rate plan can be supported based on the network link and interference characteristics, existing clients and their rate plans, etc. If not, a closely related followup question is to analyze what hardware configuration and provisioning need to be changed to accommodate the client at the lowest cost. This is an interesting research problem by itself which we are investigating. We now outline some methods that can be used for such service planning.

Planning could be done using $FS + IR$ we described in Section 6.4. Using this, we can estimate whether a new customer can be added and with what maximum rate plan. If TDMA fails to find a schedule to accommodate the new client and rate plan, there is not capacity to add the client. Note that these calculations will provide some insight but will not be exact. In fact, one benefit of APOLLO is that it can tolerate temporary fluctuations caused due to this inexact calculation, and the consequent transient congestion in operational networks. However, $FS + IR$ cannot model variable link quality, complex interference relationships (multi-way interference [24]), multiple gateways and requires routes to be predefined.

A more general and thorough albeit heavyweight service planning technique is through a linear program shown below adapted from a general model in [39]. The WMN of N nodes is modeled with a connectivity graph C whose vertices are the mesh routers (N_c) and edges the wireless links (L_c) between the routers annotated with a link capacity (Cap_{ij}) calculated via offline network measurements. f_{ijk} corresponds to the amount of flow on link l_{ij} for connection k. To model download according to a rate plan, we use a multi-commodity flow formulation with one connection for each client from its best gateway. The set of gateways is S, set of clients D, set of connections K (one per client). A virtual sink node is linked to each *existing* client with a capacity equal to the rate plan subscribed. This link is special and does not cause interference.

We also derive a conflict graph of the network through measurement [68] in which each link in L_c becomes a node and edges between nodes denote interference. This is required by the LP to determine which links can be scheduled simultaneously (spatial reuse).

Given the above inputs, the LP below can check if adding a new client at a specific location in the graph with a specific service plan is feasible as follows: A new client+plan is added to the existing graph C with sinks denoting rate plans of existing subscribers and the LP evaluated for feasibility to admit the new client. The constraint in Eq. 1 is flow conservation while the constraints in Eq. 2 say that the incoming flow to sources is 0 while outgoing flow from destinations is 0. Eq. 3 disallows negative flows. Eq. 4 and Eq. 5 force the LP to consider single-path routing (a common protocol semantic) since multiple network paths can have adverse effects on TCP; and also indicate that the amount of flow per link cannot exceed the link capacity. Eq. 6 provides a lower bound on optimal throughput using the conflict graph to determine which links can be scheduled together (details in [39]). H_x in Eq. 6 defines a schedulable set of links that can simultaneously be active and k' is the number of schedulable sets found. λ_x, $0 \leq \lambda_x \leq 1$ denotes the fraction of time allocated for set x, i.e. the links $l_{ij} \in H_x$. Finally, the LP can also be used to model improvements (more radios, mesh routers, gateways) to support the new client.

$$max \sum_{s \in S} \sum_{l_{si} \in L_C} f_{sik} \forall k \in K \quad subject\ to:$$

$$\sum_{l_{ij} \in L_C} f_{ijk} = \sum_{l_{ji} \in L_C} f_{jik},\ n_i \in N_C \setminus \{n_s, n_d\}, \forall k \in K \tag{6.1}$$

$$\sum_{s \in S} \sum_{l_{si} \in L_C} f_{isk} = 0 \quad \sum_{d \in D} \sum_{l_{di} \in L_C} f_{dik} = 0, \forall k \in K \tag{6.2}$$

$$f_{ijk} \geq 0\ \forall i, j, k \mid l_{ij} \in L_C,\ k \in K \tag{6.3}$$

$$\sum_{k \in K} f_{ijk} \leq cap_{ij} \cdot z_{ijk}\ \forall i, j \mid l_{ij} \in L_C,\ z_{ijk} \in \{0, 1\} \tag{6.4}$$

$$At\ each\ node\ n_i, \sum z_{ijk} \leq 1 \tag{6.5}$$

$$\sum_{x=1}^{k'} \lambda_x \leq 1, \quad \sum_{\kappa \in K} f_{ijk} \leq \sum_{l_{ij} \in H_x} \lambda_x \cdot cap_{ij} \tag{6.6}$$

6.5.2 Admission Control

The second component in APOLLO's design is rate-based *admission control.* This component is fundamental in all last-mile techniques such as cable and DSL. In APOLLO, this is achieved in software since shared wireless media access provides no built-in link-layer support for restricting traffic from greedy clients. Hence, similarly as in all Internet technologies, in APOLLO, each client's traffic is restricted at the client's access mesh router to the client's paid service plan. If the client tries to send or receive more traffic than that allowed by its service plan, this traffic will be dropped at its access mesh router and it will not enter the WMN. An immediate implication of this policy is that all packets that enter the WMN backbone are "paid for" and should be delivered.

6.5.3 Priority Scheduling

The third component of APOLLO's design is to address unfairness that can occur when operating the network close to capacity (Section 6.3) as well as due to channel variation, loss and replacement of routes and external interference. In these scenarios, APOLLO provides a reactive fair scheduling without requiring a fine-grained time-slotting. The basic premise is that since all packets on the backbone are "paid for", they should be handled with equal priority.

Key concept

The design of APOLLO's lightweight distributed priority scheduling is based on the key observation that while there are many different causes of unfairness in a WMN environment (interference, fading, packet collisions, 802.11 backoff algorithm), all of them typically result in a single symptom: *an increase in the queue length of the affected node*. Rather than identifying and attempting to cure the “disease” (which is significantly harder to identify and address), APOLLO directly addresses the “symptom”, i.e. the queue length. An increase in the queue length increases the average packet delays and reduces the throughput. Finally when the queue becomes full, packets start being dropped. Hence, it is critical to schedule packet transmissions in a way that will prevent increases in the queue lengths. Following this guideline, APOLLO prioritizes access to the wireless channel in each interference neighborhood to nodes based on their queue lengths, i.e., nodes that have higher queues receive transmission priorities. This is done as follows: Each node is made aware of the queue lengths of all other nodes with which it interferes (we describe the way this information becomes available in the next section). If a node has the highest queue in its neighborhood, it acquires the channel and starts transmitting packets. All other nodes defer their transmissions. As long as the information about the neighbors’ queue lengths is not outdated, all nodes in a neighborhood will make the same decision on which node should transmit; hence there will be no packet collisions. The node that obtained the right to transmit will keep sending data packets reducing the length of its built-up queue, until the moment that some other node obtains a larger queue length. In essence, APOLLO performs priority scheduling temporarily when a node’s queue builds up but does not rely on any apriori schedule, which make implementation feasible. In addition, if the maximum queue in a neighborhood is small (underutilized network), APOLLO behaves similarly to 802.11, which works fine when utilization is low. Note that in some corner cases, it is possible that APOLLO reduces spatial reuse of the channel because of dependence among nodes in overlapping interference neighborhoods, i.e., node A defers to node B which defers to node C, but node A and C could transmit together. While this is a bigger concern for slot based scheduling algorithms [80] which assign turns every cycle, APOLLO is less affected since queue backlogs are typically transient and localized. Nonetheless, a solution such as an inactivity timer [80], i.e., a deferring node that senses a channel free for some time can transmit, can be used in APOLLO as well.

State dissemination

APOLLO requires nodes to maintain correct, up-to-date information about their neighbors' queue lengths. The design of state dissemination focuses on (i) to which nodes this state is disseminated and (ii) what is the dissemination mechanism.

Dissemination neighborhood Although a reasonable answer to the first question seems to be "as many nodes as possible", such a choice is both impractical and incorrect. First, we want information to be propagated quickly, so that all nodes make a correct decision as soon as possible. Second, propagating information beyond an interference region might harm throughput. For example, if all nodes in the network know each other's queue lengths, then there will be only one node transmitting in the whole network, thus not exploiting spatial reuse. Hence, information should only be propagated to nodes within an interference region. Nodes that belong in different interference regions make different decisions. The goal is that at any given time, each interference region should have a winner node that transmits, exploiting spatial reuse at the maximum degree, while preventing collisions.

Unfortunately, identifying an interference region is not a simple task. The most common solution is to assume that all nodes within a k-hop distance interfere with each other, where k is a tradeoff between the probability of interference and channel utilization. [80] uses a 2-hop interference region. However, their simulation model is not realistic, since they only used the two-ray propagation model, without simulating any channel variability (noise, fading, etc.). As we saw in Section 6.4, with a more realistic physical model, even nodes in a 3-hop distance can interfere. We have also verified it experimentally in our 32-node testbed [24]. Hence, in APOLLO we use a 3-hop interference neighborhood. While this is a heuristic, it makes the system practical. Measuring continuously changing interference patterns is impractical and is difficult to perform online.

Dissemination mechanism APOLLO uses two CONTROL messages: **BEACON** and **LEAVE** messages. The BEACON message broadcast every 200ms[9] contains the node's current queue length, a sequence number used for recency information and a TTL value initialized to 3 to implement the 3-hop interference neighborhood. To reduce overhead, BEACONs are sent only if the difference between current queue size and the last advertised queue size is more than 5% of the maximum queue size.

While BEACONs allow nodes in an interference neighborhood to periodically evaluate all the queue lengths and decide who should transmit for the next period, the winning node may finish all its packets before the next BEACON exchange. To minimize idle time, whenever a node that is currently transmitting has no more packets to send, it initiates a LEAVE message. LEAVE messages are also broadcasted in a 3-hop neighborhood. Each node receiving a LEAVE message, updates the queue length of the node that initiated the LEAVE to 0, and then it decides to transmit if its queue has become the largest one. The problem is that the moment a node receives a LEAVE message, the information it has for all other nodes is outdated; it is what the node had learnt in the last BEACON exchange phase. For this reason, upon receiving a LEAVE message, a node in addition to rebroadcasting it, also piggybacks its current queue length in the packet. Since decisions to transmit/defer are remade on every LEAVE/BEACON message receipt, wrong decisions because of incomplete or outdated information are corrected soon.

Since recency is important for BEACON and LEAVE messages, these packets have higher priority than data packets. One approach is to transmit them immediately using 802.11 broadcast. However, since 802.11 broadcast uses no control packets such as RTS/CTS/ACK and transmits each packet only once, the CONTROL packets will likely be lost due to collisions with other CONTROL or data packets. For this reason, we use a separate channel to transmit BEACON and LEAVE messages. Using

[9] This value was obtained after experimental sensitivity analysis

a separate control channel and radio potentially wastes the capacity of the WMN. However, we argue the control radio is justified for a number of reasons: (1) When WMNs are deployed as operational commercial networks, there is a need to separate the control and data plane which enables real operation and management of WMNs and will probably be required in most cases so that network debugging, code updates, interference measurements, channel assignment, etc can be done without hurting paid customer traffic. (2) Additionally, this single control radio can be used to provision a system with multiple data radios (which are commonly available). For multiple data radios, BEACON messages now will contain a vector of the queues on each radio of a node and a LEAVE message has to specify for which radio it has been sent. (3) Finally, many newly proposed protocols ([96], [84], [73]) for WMNs require a control radio which can be leveraged by APOLLO.

As a side note, we did try other priority schemes based on queue lengths such as prioritizing nodes based on DIFS and backoff [1], before converging to our final scheme. We also tried a scheme where a node's decision on transmitting or not is not discrete (the node transmits if it has the largest queue) but continuous, i.e., based on the queue length a node decides to transmit with probability P, and P is higher for nodes with larger queues. None of these schemes worked efficiently in resolving contention, because they are probabilistic [91]. Essentially, probabilistic schemes did not respond fast enough to alleviate congestion and resulted in lower throughput due to delays and packet drops.

Putting It All Together

We summarize the operations of APOLLO with the help of the simple topology shown in Figure 6.1. Initially, since no node has any information about its neighbors, each node thinks that it has the largest queue length and transmits packets. Hence, nodes compete for the channel using the original 802.11 algorithm. Periodically, nodes exchange beacons with their queue lengths, and very soon they obtain complete knowledge of their interference neighborhood and coordinate their transmissions based on queue lengths. Very soon node C builds up a queue since it receives packets from the application layer from flow $F2$, and also packets from flow $F1$ through node B. Hence node C obtains the right to transmit and all other nodes remain silent. Depending on the application sending rate, node C might empty its queue before the next beacon message and in that case it sends a LEAVE message, giving some other node the right to transmit, or it might keep the channel until the next beacon transmission. In that case it is possible that some other node's queue has become the largest one (e.g., nodes D's which received the packets sent form C or node A's which received packets from the application layer for flow $F1$), hence node C will leave the channel, which will be assigned to the other node. Note that by using a 3-hop interference region, node A learns the queue sizes of all other nodes up to D. Hence, except for the very rare situation that two or more nodes have exactly the same queue length, only one out of the four nodes A, B, C, or D will transmit at any time.

6.5.4 Implementation

We implemented APOLLO in Linux 2.6.8. The APOLLO architecture is shown in Figure 6.4. The two main functions of APOLLO are admission control (implemented by the APOLLO Admission Control Module (AACM)) and priority scheduling (implemented by the APOLLO Signaling Module (ASM) and the APOLLO Driver Modules (ADMs) (one for each radio)). The ASM and ADMs communicate through the */proc* file system. The ASM is responsible for reading the queue length, sending BEACON and LEAVE messages, collecting information about other nodes' queues and deciding when the node should transmit. The ADM modules, one for each wireless card driver control the transmissions of the wireless radios. As mentioned in Section 6.5.3, control messages are sent through a different radio. In Figure 6.4, we denote it by CR (Control Radio) to distinguish it from the radios used for data (DRs).

APOLLO Admission Control Module While admission control according to rate plans can be implemented using the `tc` utility (http://lartc.org//howto/), we

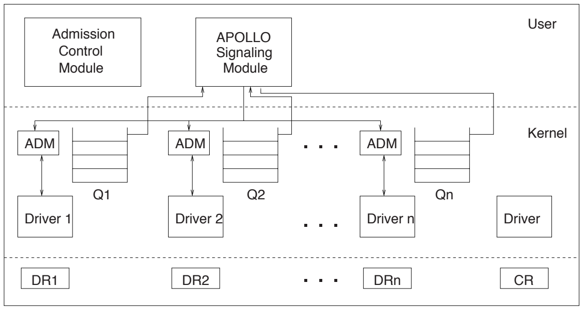


Fig. 6.4. APOLLO architecture.

perform this function using our own user-space module because: (1) Backlog in the actual device queue is used for deciding scheduling priority. So packets sent exceeding the service rate should not be allowed to fill that queue and thus should be blocked in user-space. (2) We can implement rate plans across multiple radios easily by doing it in user-space. The AACM at the access mesh router uses *libipq* and Netfilter to perform packet capturing and rate limiting for outgoing packets for associated clients according to a service plan. Similarly, incoming flows to a client from multiple sources can be limited in aggregate similarly. However, the flows must be responsive to loss (i.e. TCP) so that packets that will be ultimately dropped at the destination (due to the destination's incoming admission control) in the WMN slow down the sending rate at the sources. To handle mobility, each client is identified by its MAC address on sign-up and the client's plan is made known to all mesh routers. Thus, if a client moves and associates with a different mesh router, the admission control functionality is enabled at the new access mesh router.

APOLLO Signaling Module The ASM decides when the driver should transmit or defer, and exchanges queue length information through BEACON and LEAVE messages. ASM reads the queue length by communicating with the kernel. For each packet being transmitted from the IP layer, the *dev_queue_xmit()* procedure is called. It queues a packet in the qdisc associated with the output interface, determined by routing. Then, if the device is not stopped, all packets in the qdisc are handled by *qdisc_restart()*, which finally calls the *hard_start_xmit()* method, implemented in the driver code. ASM uses the number of backlogged packets in the qdisc to decide if the card should transmit or not, and in sending LEAVE and BEACON messages.

APOLLO Driver Module The ADM consists of changes to drivers to support APOLLO transmit/defer decisions and are driver-specific. To make APOLLO portable across drivers we made the interface of the ADM and ASM very simple: ADM for each interface communicates (through the */proc* file system) with ASM and applies the decisions made by the ASM on whether this particular interface should transmit or not. When in deferred state, ADM blocks the kernel from calling *hard_start_xmit()* using *netif_stop_queue()* and when transmitting, *netif_wake_queue()* is called to allow the kernel to call *hard_start_transmit()* and send packets to the wireless card. In this chapter we used the *madwifi* driver [57]. Similar modifications can be easily done in any other driver.

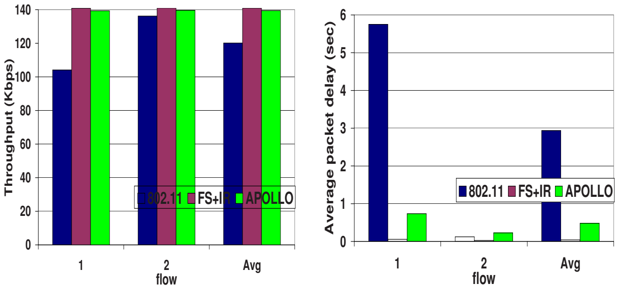
6.6 Experience with APOLLO

In this section, we describe our performance experience with the Apollo system. We first evaluate APOLLO using the Qualnet simulator in Section 6.6.1, in order to understand its behavior in a controlled environment (with controlled node positions, distances, channel conditions, etc). Next, in Section 6.6.2, we evaluate the APOLLO implementation in our testbed.

6.6.1 Simulation Evaluation

We first evaluate APOLLO in two simple scenarios using the 5-node chain of Figure 6.1 by comparing it with 802.11 and $FS + IR$ from Section 6.4. We then evaluate its performance in more general topologies, comparing it with 802.11.

For our simulations we use the Qualnet simulator [74]. In all the scenarios the *two-ray* propagation model is used. For realism, we added random noise creating 10-15% loss in each link. The nominal bit rate in both channels is 2Mbps. The transmission range is set to 250m. For all the experiments, LQSR [30] is used as



(a) Throughput (b) Average packet delay

Fig. 6.5. Performance comparison of 2 asymmetric flows in a 5 node chain chain for 802, $FS + IR$ and APOLLO.

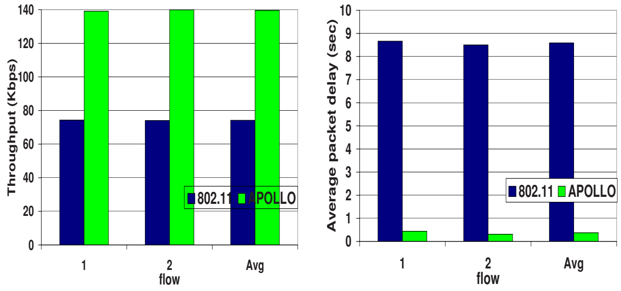
the routing protocol. The service plans used in all scenarios are made sure to be achievable according to Section 6.5.1.

Chain topologies

Asymmetric flows We first use the scenario of Figure 6.1 with the two asymmetric flows initiated at nodes A and C to gateway E. Each flow has a service plan of 140Kbps.

Figure 6.5 shows that the performance of APOLLO is almost as good as the performance of the optimal $FS + IR$ protocol. Both flows achieve PDR higher than 98% and throughputs equal to 139Kbps. Hence APOLLO provides to both clients the service they paid for, without the fine-grained synchronization required for the TDMA-based $FS+IR$. Of course, this lack of fine-grained synchronization affects the average packet delay and APOLLO cannot achieve packet delays as low as $FS + IR$. However, compared to the off-the-shelf approach, APOLLO improves the average packet delay over both flows by a factor of 6.

Symmetric flows We now evaluate APOLLO under a scenario consisting of two symmetric flows (equal distance away from the gateway) in the same chain topology with node C now as the gateway and nodes A and E with a 140Kbps service plan sending packets to node C.



(a) Throughput (b) Average packet delay

Fig. 6.6. Performance comparison of 2 symmetric flows in a 5-node chain for 802.11 and APOLLO.

The results in Figure 6.6 show that, as opposed to the asymmetric scenario, in this case 802.11 is fair, and each flow achieves a PDR of 52% and a throughput of 74Kbps. However, the achieved throughputs are still very far from the service the two clients require – 140Kbps. On the contrary, APOLLO provides again a close to optimal service to both flows, with a PDR of about 98% and a throughput of 139Kbps. Also, the average packet delay with APOLLO is 23 times lower than with 802.11. 802.11 suffers from hidden terminal [10] problem in this case while APOLLO is not affected by it since it coordinates the nodes based on queue lengths.

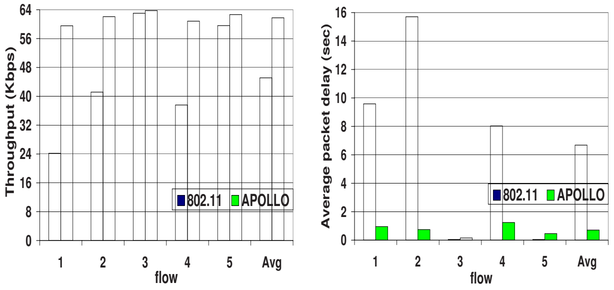
Large networks

We now evaluate the performance of APOLLO in a large network of 49 nodes forming a grid topology in a $1500 \times 1500m^2$ area. Again the distance between any two nodes is such that each node is in transmission range with its 1-hop neighbors, but not with its 2-hop neighbors.

P2P traffic This topology consists of 5 flows with randomly selected distinct sources and destinations , $F1$ – $F5$, each with a 64Kbps service plan.

The performance results in Figure 6.7 show that APOLLO achieves the client's service plan. On the other hand, 802.11 fails in providing the service plan with $F1$,

[10] We shut off RTS/CTS as do many operational networks.



(a) Throughput (b) Average packet delay

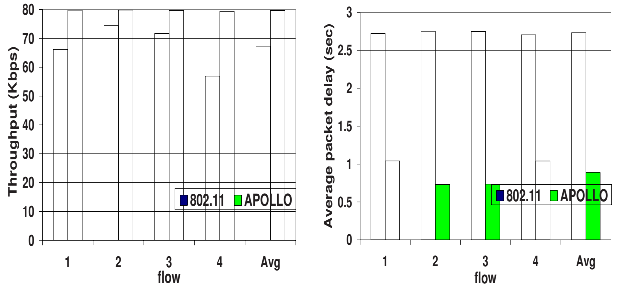
Fig. 6.7. Performance comparison of 802.11 and APOLLO in a P2P scenario.

$F2$ and $F4$ unhappy. The starved flows also have high delays due to which the average delay under 802.11 is 9 times larger than APOLLO.

Flow $F3$ is physically far from the other flows and thus it is relatively unaffected, achieving the maximum throughput, while $F5$ captures the channel more frequently than $F4$ with which it is intersected, resulting in performance degradation for $F4$. Also, flows $F1$ and $F2$ interfere with each other, and with $F4$, and this results in low throughput and high packet delays for both of them. On the other hand, APOLLO can arbitrate better whenever such capture and competition occur in each competing neighborhood.

Download from the gateway This is the most common scenario in a WMN, where clients download traffic from the Internet through the gateway. In this case we have placed the gateway at the center of the network. We have 4 download flows in a cross-shape from the gateway to 4 different clients. Two of these flows travel two hops and the other two travel three hops. Each client has a plan of 80Kbps.

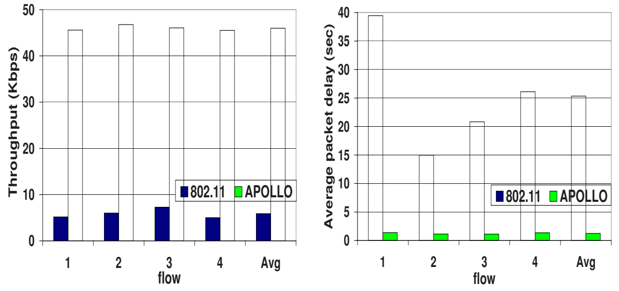
This scenario is less stressing on the MAC layer, since contention is primarily near the source, but not near the receivers of the 4 flows. Since the gateway transmits received packets from the Internet in FIFO order, it fairly schedules its transmissions for all four flows. The results in Figure 6.8 show that with APOLLO, all four clients get exactly what they pay for. With 802.11 there is less unfairness compared to the P2P scenarios and no flow starves, but the two 2-hop flows ($F2$ and $F3$) achieve



(a) Throughput

(b) Average packet delay

Fig. 6.8. Performance comparison of 802.11 and APOLLO in a download scenario.



(a) Throughput

(b) Average packet delay

Fig. 6.9. Performance comparison of 802.11 and APOLLO in an upload scenario.

higher throughput than the two 3-hop flows ($F1$ and $F4$). Although its performance is improved, 802.11 still fails to provide the requested service. Also, the average packet delays are 2-4 times larger with 802.11 than with APOLLO.

Upload to the gateway This scenario is the inverse of that in the previous paragraph, where clients send traffic to the gateway. Note that this is the most difficult scenario to handle, since contention (i.e. number of senders) increases as packets from all four flows travel towards their common destination. Since, in general, upload traffic is expected to be lower than download, in this scenario each of the 4 clients has a 50Kbps service plan.

The results in Figure 6.9 show that 802.11 completely fails in this scenario. All four flows starve getting throughputs lower than 8Kbps, only 16% of their service plan. Also, in Figure 6.9(b) we observe that the average packet delays are extremely high for all four flows, varying between 15 sec and 40sec. On the other hand, APOLLO can fairly deal even with this challenging scenario. All four flows receive similar throughputs, 45-46Kbps, very close to the service plan, and delays are 13 to 28 times lower compared to 802.11. Since the queues of all 4 nodes around the gateway are built equally fast (all flows have the same rate), APOLLO assigns transmission rights to them in a round-robin fashion, eliminating collisions. On the other hand, with 802.11 all nodes compete for the channel, but without any coordination and this finally results in large packet delays, due to exponentially increased backoff times and many retransmissions, and finally to packet drops, since queues are gradually being filled.

6.6.2 Testbed Evaluation

We now evaluate APOLLO on our wireless network testbed MAP [111], shown in Figure 6.10. MAP consists of 32 mesh routers (small form factor desktops) spread out across four academic buildings on the Purdue campus (EE, MSEE, PHYSICS and ME). Each router has two radios. Each radio is attached to a 2dBi rubber duck omnidirectional antenna with a low loss pigtail to provide flexibility in antenna placement. Each mesh router runs Mandrake Linux 10.1 and the open-source *madwifi* drivers are used to enable the wireless cards. IP addresses are statically assigned. The testbed deployment environment is not wireless friendly, having floor-to-ceiling office walls instead of cubicles as well as some laboratories with structures that limit the propagation of wireless signals. We used channels 1 and 11 of 802.11b to operate our network since they provide the largest frequency separation. Autorate is turned on. The routes used in all scenarios have been found by *OLSR* [20] using the ETX metric.

We evaluate APOLLO in various scenarios and report our experience.

Dominant flow scenario In this scenario, we assume node 1 is the gateway and nodes 22 and 16 generate traffic to node 1. Node 22 can reach node 1 through a 2-hop path (22 → 13 → 1) while node 16 can reach node 1 directly. We assume that clients that use node 22 as their access mesh router can be supported at an aggregate

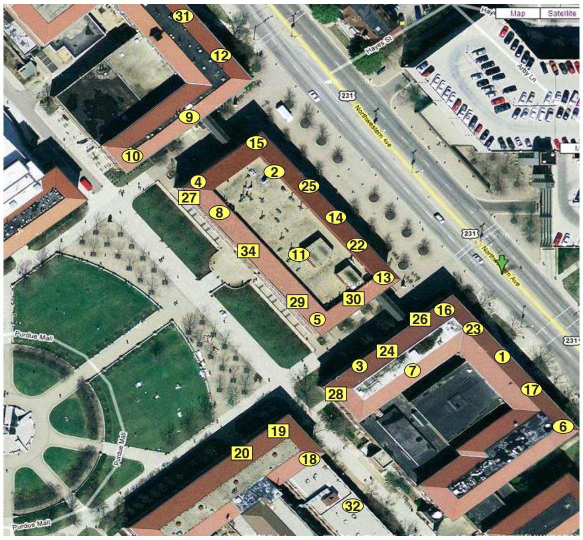
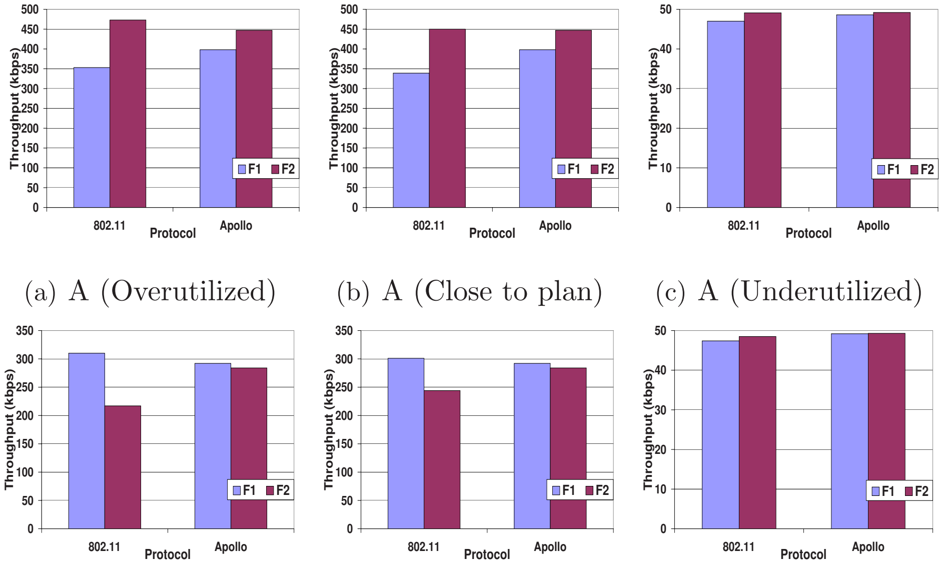


Fig. 6.10. Top view of the MAP testbed topology used in the experiments.



(a) A (Overutilized) (b) A (Close to plan) (c) A (Underutilized)

(d) B (Overutilized) (e) B (Close to plan) (f) B (Underutilized)

Fig. 6.11. Throughput comparison of 802.11 and APOLLO in A: dominant flow, and B: balanced flow scenario.

rate of 400Kbps and clients that use node 16 can be supported at an aggregate rate of 450Kbps. We emulate this scenario by generating two UDP flows $F1$ (400Kbps service plan) and $F2$ (450Kbps service plan), from nodes 22 and 16 to 1, respectively, using `iperf`.

The results in Figure 6.11(a)(b)(c) show that when clients transmit data close to their service plan in 802.11, $F2$ can achieve throughput equal to the service plan of 450Kbps, but $F1$ receives only 339Kbps. The reason is that $F2$ is a 1-hop flow, while $F1$ is a 2-hop flow. Moreover, from Figure 6.10 we observe that node 16 is closer to the destination (node 1) than node 13 (the intermediate hop of flow $F1$). Hence, node 16 can capture the channel more frequently than node 13, which also suffers higher packet loss. On the other hand with APOLLO, we observe that both flows achieve throughputs almost equal to their service plans (398Kbps and 447Kbps, respectively). APOLLO detects that node 13 has a larger queue than node 16 and allows it to transmit when it experiences a queue backlog. APOLLO is similar to 802.11 when the plans are underutilized (with sending rate 50Kbps) and better than 802.11 if clients are able to send more than their service plan (with sending rate 800Kbps).

Balanced-flows scenario In this scenario we have two 2-hop UDP flows, $F1$ ($11 \rightarrow 5 \rightarrow 28$) and $F2$ ($20 \rightarrow 18 \rightarrow 28$), each of which has a service plan of 300Kbps. However, now both flows have equal number of hops.

The results in Figure 6.11(d)(e)(f) also show that with 802.11, when the clients transmit close to their service plan, flow $F1$ achieves a throughput equal to the service plan, while flow $F2$ achieves a throughput of 244Kbps, 81% of the service plan. Thus, unfairness can also manifest between flows of similar hop-lengths due to link quality differences. With APOLLO, the unfairness becomes much smaller and the two flows achieve throughputs of 292Kbps and 284Kbps, respectively, or 97.3% and 94.6% of their service plans. Again, APOLLO is similar to 802.11 when the plans are underutilized (with sending rate 50Kbps) and better than 802.11 if clients are able to send more than their service plan (with sending rate 600Kbps).

TCP upload While there have been many attempts to build new transport protocols for multi-hop wireless networks, TCP is still predominantly deployed in operating systems used on clients and mesh routers. We thus measured TCP performance as part of a service plan. We used three flows $F1$ (Node $1 \rightarrow 28$ with route

Table 6.1
Throughput achieved by three flows F1, F2 and F3 from a gateway by themselves, combined, and with APOLLO. A-xk refers to an APOLLO x Kbps plan.

	Auto upload (Kbps)			Auto download (Kbps)		
	802.11 self	802.11 comb.	A-256k plan	802.11 self	802.11 comb.	A-300k plan
F1	274	38	255.2	295	89	293.4
F2	1061	356	256.1	1001	455	297.1
F3	1887	1100	256.1	1765	1200	298.3

1-16-24-28), $F2$ (Node 16 → 28 with route 16-24-28) and $F3$ (Node 24 → 28 with route 24-28).

Table 6.1 shows the performance of these flows when uploading to gateway node 28 individually under 802.11, combined under 802.11, and combined under APOLLO with a service plan. The results show that when the TCP flows from the 3 clients run together in 802.11, F1 is almost shut off. However, APOLLO is able to enforce a rate plan of 256Kbps at each node. Thus, in this scenario, APOLLO allows multi-hop TCP flows from clients to achieve their service plans.

TCP download We repeated the experiment for the case of downloading from the gateway node. In this case, the three flows are reversed. Again the results in Table 6.1 shows that APOLLO can support a service plan of 300Kbps at each mesh router thus supporting up to 9 clients with plans of 100Kbps. Unlike in 802.11, APOLLO can allocate service to clients more flexibly, e.g. support more clients at mesh router 1 if needed by reducing clients elsewhere.

Link failure APOLLO has the ability to deal with temporary fluctuations in performance due to route recomputation, external interference, and fading. We emulate such a scenario by injecting a transient link failure artificially in the driver. For this experiment, we sent two UDP flows, from nodes 16 and 22 to node 13, both one hop away from 13. We assume both mesh routers have clients generating traffic of

700Kbps. When running simultaneously, the two flows could both get this throughput. We artificially disturbed the link 22 → 13 multiple times during a 70 second period. In both cases, the throughput of node 16 remained unaffected, but throughput of node 22 was different under 802.11 and APOLLO. With 802.11, node 22 achieved only 433Kbps or 62% of its service plan, while with APOLLO the throughput was 657Kbps, or 93% of its service plan. The reason for this difference is as follows. In the intervals following the failures, APOLLO notices that node 22 has a large queue length and it gives it transmission rights for most of the time, allowing it to send the queued packets. The fact that throughput of node 16 is not reduced shows that the capacity is enough for both nodes to achieve 700Kbps. In spite of that, node 22 cannot recover with 802.11. In this case, 802.11 does not take into account the queue built in node 22, and lets the two nodes contend for the channel in the normal link operation intervals. This results in time wasted due to collisions and backoffs, and finally does not give node 22 sufficient time to recover from the failures.

Caveat APOLLO does not guarantee that a TCP flow reaches the bandwidth prescribed by the service plan. APOLLO relies on increased queue length as a symptom. If there are many losses due to interference or bad links, some TCP flows may be unable to increase their window. Similar problems with TCP flows cause low gain from network coding [45]. This is not a problem per se with the MAC layer capacity allocated by APOLLO, but that TCP does not achieve the underlying bandwidth made available. Reducing transport layer unfairness in lossy scenarios remains a place where further research is needed. APOLLO provides an environment where a better designed transport protocol (which can for example distinguish random packet losses from congestion) can achieve good performance.

6.7 Related Work

There is a lot of work on priority scheduling in wireless ad hoc networks [1,42,43, 101]. In [1], three schemes varying different parameters of 802.11 MAC protocol are compared: backoff interval, DIFS, and maximum frame length and the DIFS-based scheme is found to perform well in noisy environments. Another backoff-based priority scheme is described in [42, 43] which piggybacks the priority tag of a node's head-of-line packet onto RTS and data packets. Nodes use information about priorities of all flows in a 2-hop neighborhood to set their backoff. Finally [101] proposes to use uses two narrow-band busy tone signals (BT1 and BT2) to ensure medium access for high priority stations. DIFS periods of high/low priority stations are set accordingly to ensure that low-priority stations in a 2-hop neighborhood will hear BT1 or BT2 before they try to transmit a packet and will defer. In all these works, priorities of different flows are statically assigned based on QoS requirements, while in our case priorities change dynamically, based on queue lengths, in order to achieve a fair service. Hence our scheduling has a more challenging objective, namely to avoid starvation of any flow, as opposed to differentiation mechanisms where the goal is to ensure that high-priority flows will not starve because of a low-priority flow. Moreover, all these schemes are probabilistic and they cannot effectively provide service plans as explained in Section 6.5.3.

Several works exist on fair scheduling in multihop wireless networks, e.g., [34, 54–56, 80, 83]. In [54–56], fair queuing to maximize spatial reuse while ensuring a minimum fair bandwidth allocation for each flow is proposed. The algorithms are centralized and suffer from many of the problems related to TDMA. A core node is required to calculate/propagate the schedule. Moreover, the schedule needs to be recalculated every time a flow comes or leaves the network. In [56], a distributed approach to the fair queuing problem is proposed which is backoff-based (probabilistic), and hence it is not guaranteed that it will perform well in realistic, noisy environments [91]. Also, the fairness model is different and there is no implementation evaluation.

In [34], per-mesh-router (TAP) fairness and a per-TAP fairness scheduling is proposed in which adjacent TAPs exchange periodically information about offered load and link capacities which is used to compute the time shares of different flows on each link and enforce the estimated time shares using rate limiting. The algorithm is only evaluated using simulations and assuming perfect information on offered loads and link capacities. In practice it is not easy to propagate this information, especially when network conditions change rapidly. In contrast, APOLLO requires much less information to be propagated (queue lengths) and we demonstrate its efficiency through a testbed implementation. Additionally, per-TAP fairness does not allocate per-client service plans. The only work that considers a per-mesh-client fairness model is Fair Scheduling (FS) [83], which we described in Section 6.4. This algorithm also has the same practical problems as TDMA (Section 6.4).

All the previous algorithms were evaluated only using simulations in ideal environments. *Overlay MAC Layer* (OML) in [80] provides an implementation. OML uses a TDMA-like solution, in which time is divided in slots (with a coarser time granularity) and slots are allocated to nodes according to a WFQ policy, using pseudo random hash functions. OML still suffers from the same problems as TDMA, which we described in Section 6.4.3. The weights depend on the number of flows, the routing paths used and the type of traffic used. Hence it is very hard or even infeasible (in case of P2P traffic or auto rate adaptation) to assign weights to the nodes in order to support given service plans. In [14], the authors identify severe unfairness between flows in mesh networks. They found that rate limiting improves the fairness of TCP flows. In our work we implement admission control as one of the basic components of APOLLO, and we show that it alone cannot always guarantee a fair service.

6.8 Summary

In this chapter, we argue that a “bitrate-for-bucks” service model is key to the successful adoption of WMNs as a last-mile technology. We demonstrate that neither an off-the-shelf approach nor an optimal scheduling approach are viable in providing such a service. We then propose, design, implement, and evaluate APOLLO, a system that implements service provisioning. We believe this work addresses a critical challenge in the deployment and management of WMNs.

7. CONCLUSIONS

Wireless mesh networks find useful applications in metro-area public Internet access, community wireless networks, and transient networks (e.g., disaster relief). This thesis showed however that several performance challenges are faced by current wireless mesh networks in terms of performance and usability through real measurements of a large deployed wireless network.

We proposed to enhance mesh routers with new hardware and software innovations to address the performance challenges of wireless mesh networks. This proposed X-router is enhanced with a network coding layer in the software stack, multiple radios that can be configured with multiple channels, practical directional antennas and cheap storage devices. The key contribution of this thesis are the protocols and algorithms that take advantage of these lower layer technologies systematically and efficiently and lead to significant performance improvement in mesh networks. For example, network coding may not mix enough packets if flows don't get routed in certain ways. This thesis proposed MMSR to exploit network coding and demonstrated real performance gains. Additionally, taking advantage of multiple radios involves careful examination of channel performance on each radio which may be different in different regions of the network. A bad choice of route could mean no interference reduction. This thesis proposed CRP using the SIM metric to avoid self-interference and demonstrated performance improvements over state-of-the-art approaches. Similarly, adding directional antennas which have imperfect directivity patterns and sidelobes and backlobes requires intelligence to make best use of spatial reuse. This thesis proposed DMesh that provided performance improvements despite imperfect spatial and frequency reuse in real wireless mesh networks. Finally, simply adding storage to mesh routers may not improve performance since upper layer protocols need to intelligently exploit these devices. This thesis proposed MeshCache that was able to extract significant throughput gain via leveraging cheap storage and exploiting the locality in the access patterns of users.

Orthogonal to the performance of the wireless mesh network, an important issue to address is how to use a wireless mesh network to provision services to provide to consumers. In this thesis, we proposed the **APOLLO** system that leverages off-the-shelf hardware with software modifications to provision the “bitrate-for-bucks” service. APOLLO seamlessly integrates three synergistic components: (1) A theory-guided service planning and subscription technique to decide how to admit new subscribers. (2) A rate-based admission control to restrict the upload/download traffic from clients in accordance with their service plans. Importantly, APOLLO uses immediate admission control at the access mesh router of the client such that all packets that enter the mesh backbone network are “good” packets that should receive service. This ensures that the traffic in the network is below the capacity of the network. (3) A novel distributed light-weight fair scheduling scheme to deliver the admitted traffic that is robust to unfairness that can arise from interference, variability in wireless channel quality, collisions, etc.

An additional contribution of this thesis is that all the proposed algorithms and protocols were evaluated via an implementation deployed on a large wireless network. Thus, the contributions of this thesis are shown to have direct practical application and significance.

7.1 Looking Forward

A useful direction looking forward is to build the $X - router$ as an embedded device and look at how to integrate all the protocols and algorithms proposed in this thesis into the software of the router. There are several interesting challenges in this direction of research regarding device architecture, operating system and network stack design, and upper layer protocols. Reducing internal interference among the radios is heavily influenced by the placement of radio interfaces on the device and the tuning efficiency of the radios themselves. There are several interesting issues regarding how to integrate the network coding layer and the multiple physical interfaces in the network stack of the X-router. Finally, upper layer protocols proposed in this thesis may require modifications to work well in an integrated manner on a X-router.

Other useful directions in architecting high-performance mesh networks looking forward as well as extending the protocols and algorithms proposed in this thesis are:

- Exploiting the MeshCache content cache for current and future applications in WMNs such as swarming or streaming is an interesting area of further research. The storage device could potentially be used as a buffer for storing blocks of a media stream or chunks for a BitTorrent-like swarming application.

- Exploiting software steerable antennas for improving performance as antenna technology makes them cheaper is an interesting issue for further research. Such antennas can potentially have lower beamwidths that allow more spatial reuse as well as control their sidelobes and backlobes. In addition, the network topology can be dynamically changed to adapt to the current demands by steering the antennas appropriately. Again, leveraging such devices using intelligent upper layer protocols will be key to effectively utilizing their potential.

- Exploiting cognitive radios for improving mesh network performance is also a related issue for further research. While software steerable antennas allow us to dynamically change the physical space a signal occupies; using cognitive radios allows us to dynamically change the frequency space a signal occupies by looking for regions that are signal free. Both can lead to interference reduction. In this scenario, upper layer protocols would need to decide how to select frequencies and bandwidths for each radio and route packets. An even more sophisticated approach would be required if such cognitive radios would be coupled with steerable directional antennas to allow dynamic tuning in both physical and frequency domains.

- Integrating random linear network coding [15] into the X-router software stack is another interesting direction of research. The way this style of network coding works is to generate mixtures of packets at each hop and rebroadcast them. This allows the use of opportunistic reception. It is interesting to see whether such intra-flow network coding is compatible with inter-flow network coding currently proposed as part of the X-router.

- Investigating and improving multicast performance in wireless mesh networks is an important future direction of research. This dissertation has focused on more traditional unicast applications used in Internet access. However, multicast is also an important service that can be enabled in wireless mesh networks. It provides an efficient means of supporting collaborative applications such as video conferencing, online games, webcast and distance learning, among a group of users. A interesting research problems is to determine how the enhancements in the X-router can be leveraged to improve multicast performance.

- There may be other promising applications of routing using context information that can be utilized in other application scenarios where link interdependencies exist such as multicast in wireless networks and multi-wavelength optical networks.

In summary, this dissertation has laid the groundwork for building high-performance mesh networks that can potentially be leveraged to provide Internet access to the billions of remaining people without such access on the planet. The proposals in this dissertation successfully leverage lower layer enhancements via intelligent upper layer protocol design to significantly improve the *performance* of mesh networks. The service provisioning architecture proposed in this dissertation then improves the *usability* of such a high-performance mesh network. By improving the performance and usability of mesh networks, we believe that a significant barrier to their widespread deployment has been alleviated.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Imad Aad and Claude Castellucia. Differentiation mechanisms for ieee 802.11. In *Proc. of IEEE Infocom*, 2001.

[2] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *Proc. of BroadNets*, 2004.

[3] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *Proc. of BroadNets*, 2004.

[4] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b mesh network. In *Proc. of ACM SIGCOMM*, August 2004.

[5] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Comput. Netw.*, 47(4):445–487, 2005.

[6] Mansoor Alicherry, Randeep Bhatia, and Li Eran Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *Proc. of ACM Mobicom*, August 2005.

[7] Y. Amir and C. Danilov. Reliable communication in overlay networks. In *Proc. of DSN*, 2003.

[8] Paramvir Bahl, Atul Adya, Jitendra Padhye, and Alec Wolman. Reconsidering wireless systems with multiple radios. *ACM MC2R*, October 2004.

[9] Paramvir Bahl, Ranveer Chandra, and John Dunagan. SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proc. of ACM Mobicom*, September 2004.

[10] Bel Air Networks. `http://www.belairnetworks.com`.

[11] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proc. of ACM MobiCom*, 2005.

[12] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proc. of IEEE INFOCOM*, 1999.

[13] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16, 1973.

[14] Joseph Camp, Joshua Robinson, Christopher Steger, and Edward Knightly. Measurement driven deployment of a two-tier urban mesh access network. In *Proc. of ACM Mobisys*, 2006.

[15] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi. Trading structure for randomness in wireless opportunistic routing. In *ACM SIGCOMM*, 2007.

[16] Ranveer Chandra, Lili Qiu, Kamal Jain, and Mohammad Mahdian. Optimizing the placement of internet taps in wireless neighborhood networks. In *Proc. of ICNP*, October 2004.

[17] Anawat Chankhunthod, Peter Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A hierarchical Internet object cache. In *Proc. of USENIX ATC*, 1996.

[18] R.R. Choudhury, X. Yang, R. Ramanathan, and N.Vaidya. Using directional antennas for medium access control in ad hoc networks. In *Proc. of ACM MobiCom*, September 2002.

[19] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). Internet Engineering Task Force (IETF), RFC3626, October 2003.

[20] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot. Optimized link state routing protocol (OLSR). RFC 3626, Oct 2003.

[21] Douglas S. J. De Couto, Daniel Aguayo, John C. Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, 2003.

[22] Peter B. Danzig, Richard S. Hall, and Michael F. Schwartz. A case for caching file objects inside internetworks. In *Proc. of ACM SIGCOMM*, 1993.

[23] Saumitra M. Das, Dimitrios Koutsonikolas, and Y. Charlie Hu. Practical service provisioning for wireless meshes. In *Proc. of CoNEXT*, 2007.

[24] Saumitra M. Das, Dimitrios Koutsonikolas, Y. Charlie Hu, and Dimitrios Peroulis. Characterizing multi-way interference in wireless mesh networks. In *Proc. of ACM WiNTECH 2006*, 2006.

[25] Saumitra M. Das, Himabindu Pucha, and Y. Charlie Hu. Symmetrical fairness in infrastructure access in multi-hop wireless networks. In *Proc. of ICDCS*, 2005.

[26] Saumitra M. Das, Himabindu Pucha, and Y. Charlie Hu. Mitigating the gateway bottleneck via transparent cooperative caching in wireless mesh networks. *Ad Hoc Netw.*, 5(6):680–703, 2007.

[27] Saumitra M. Das, Himabindu Pucha, Dimitrios Koutsonikolas, Y. Charlie Hu, and Dimitris Peroulis. Dmesh: Incorporating practical directional antennas in multichannel wireless mesh networks. *Selected Areas in Communications, IEEE Journal on*, 24(11):2028–2039, Nov. 2006.

[28] Saumitra M. Das, Himabindu Pucha, Konstantina Papagiannaki, and Y. Charlie Hu. Studying wireless routing link dynamics. In *Proc. of IMC*, 2007.

[29] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proc. of ACM MobiCom*, September 2004.

[30] Richard Draves, Jitendra Padhye, and Brian Zill. Comparison of routing metrics for static multi-hop wireless networks. In *Proc. of ACM SIGCOMM*, 2004.

[31] Jakob Eriksson, Sharad Agarwal, Victor Bahl, and Jitu Padhye. A feasibility study of mesh networks for an all-wireless office. In *Proc. of ACM MobiSys. Also MSR Tech. Rep. MSR-TR-2005-170,*, June 2006.

[32] Li Fan, Pei Cao, Jussara Almeida, and Andrei Broder. Summary cache: A scalable wide-area Web cache sharing protocol. In *Proc. of ACM SIGCOMM*, 1998.

[33] Valerie Issarny Francoise Sailhan and. Energy-aware web caching for mobile terminals. In *Proc. ICDCS Workshops*, 2002.

[34] Violeta Gambiroza, Bahareh Sadeghi, and Edward W. Knightly. End-to-end performance and fairness in multihop wireless backhaul networks. In *Proc. of ACM MobiCom*, 2004.

[35] Takahiro Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *Proc. INFOCOM 2001*, 2001.

[36] HyperLink-Technologies. Home page `http://www.hyperlinktech.com`.

[37] International World Statistics. `http://www.internetworldstats.com/stats.htm`.

[38] Sitaram Iyer, Anthony Rowstron, and Peter Druschel. Squirrel: A Decentralized Peer-to-Peer Web Cache. In *Proceedings of ACM PODC*, July 2002.

[39] Kamal Jain, Jitendra Padhye, Venkat Padmanabhan, and Lili Qiu. Impact of interference in multihop wireless network performance. In *Proceedings of ACM Mobicom*, 2003.

[40] David B. Johnson and David A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic, 1996.

[41] Jangeun Jun, Pushkin Peddabachagari, and Mihail Sichitiu. Theoretical maximum throughput of ieee 802.11 and its applications. In *Proc. of NCA*, 2003.

[42] Vikram Kanodia, C. Li, A. Sabharwal, Bahareh Sadeghi, and Edward W. Knightly. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Proc. of ACM Mobicom*, 2001.

[43] Vikram Kanodia, C. Li, A. Sabharwal, Bahareh Sadeghi, and Edward W. Knightly. Distributed priority scheduling and medium access in ad hoc networks. *ACM Wir. Ntws.*, 8, 2002.

[44] R. Karrer, A. Sabharwal, and E. Knightly. Enabling large-scale wireless broadband: the case for TAPs. *ACM CCR*, 34(1), 2004.

[45] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Medard, and Jon Crowcroft. Xors in the air: Practical wireless network coding. In *Proc. of ACM SIGCOMM*, August 2006.

[46] Kernel AODV. http://w3.antd.nist.gov/wctg/aodv_kernel/index.html.

[47] Y.B. Ko and N.H. Vaidya. Medium access control protocols using directional antennas in ad hoc networks. In *Proc. of IEEE INFOCOM*, March 2000.

[48] S. Kopparty, S. Krishnamurthy, M. Faloutsos, and S. Tripathi. Split-tcp for mobile ad hoc networks. In *Proc. of GLOBECOM*, 2002.

[49] Gu-In Kwon and John W. Byers. ROMA: Reliable Overlay Multicast with Loosely Coupled TCP Connections. In *Proc. of IEEE INFOCOM*, 2004.

[50] P. Kyasanur and N.H. Vaidya. Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks. In *MC2R 10(1):31–43, Jan*, 2006.

[51] Pradeep Kyasanur and Nitin H. Vaidya. Routing and interface assignment in multi-channel multi-interface wireless networks. In *Tech. Rep. UIUC*, October 2004.

[52] Jinyang Li, Charles Blake, Douglas S.J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proc. of ACM MobiCom*, March 2001.

[53] Y. Liu and E. Knightly. Opportunistic fair scheduling over multiple wireless channels. In *Proc. of IEEE INFOCOM*, 2003.

[54] Haiyun Luo and Songwu Lu. A topology-independent fair queueing model in ad hoc wireless networks. In *Proceedings of IEEE ICNP*, 2000.

[55] Haiyun Luo, Songwu Lu, and Vaduvur Bharghavan. A new model for packet scheduling in multihop wireless networks. In *Proc. of ACM MobiCom*, 2000.

[56] Haiyun Luo, Paul Medvedev, Jerry Cheng, and Songwu Lu. A self coordinating approach to distributed fair queueing in ad hoc wireless networks. In *Proceedings of IEEE Infocom*, 2001.

[57] madwifi. `http://madwifi.org`.

[58] Bruce A. Mah. An empirical model of http network traffic. In *Proc. of IEEE INFOCOM*, 1997.

[59] MAXRAD. Home page `http://www.maxrad.com`.

[60] Meru Networks. `http://www.merunetworks.com`.

[61] Mesh Networks. `http://www.meshnetworks.com`.

[62] Akihiro Nakao, Limin Wang, and Larry Peterson. MSB: Media Streaming Booster. Technical report, Princeton University CS TR-666-02, December 2002.

[63] A. Nasipuri and S. Das. A multichannel csma mac protocol for mobile multihop networks. In *Proc. of IEEE WCNC*, 1999.

[64] A. Nasipuri, J. Mandava, H. Manchala, and R.E. Hiromoto. On-demand routing using directional antennas in mobile ad hoc networks. In *Proc. of IEEE ICCCN*, October 2000.

[65] Randolph Nelson and Leonard Kleinrock. Spatial TDMA: A colllision-free multihop channel access protocol. *IEEE Transactions on communications*, 33(9), 1985.

[66] Michael Neufeld, Christian Doerr, Jeff Fifield, Troy Weingart, Douglas C. Sicker, and Dirk Grunwald. Multimac: An adaptive mac framework for dynamic radio networking. In *Proc. of DySPAN*, 2005.

[67] OECD Broadband Statistics. `http://www.oecd.org/sti/ict/broadband`.

[68] Jitendra Padhye, Sharad Agarwal, Venkat Padmanabhan, Lili Qiu, Ananth Rao, and Brian Zill. Estimation of Link Interference in Static Multi-hop Wireless Networks. In *Proceedings of IMC*, 2005.

[69] Jitendra Padhye, Sharad Agarwal, Venkat Padmanabhan, Lili Qiu, Ananth Rao, and Brian Zill. Estimation of Link Interference in Static Multi-hop Wireless Networks. In *Proceedings of IMC*, 2005.

[70] Jung-Lin Pan, Stephen S. Rappaport, and Petar M. Djuric. Multibeam cellular communication systems with dynamic channel assignment across multiple sectors. *Wirel. Netw.*, 5(4):267–278, 1999.

[71] C. Perkins, E. Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (AODV) routing. IETF, RFC3561, July 2003.

[72] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proc. of ACM SIGCOMM*, August 1994.

[73] Yi Qu and Anand Srinivasan. Multi-channel olsr with dedicated control interface. In *Proc. of SPECTS*, 2006.

[74] QualNet. http://www.scalable-networks.com.

[75] Bhaskaran Raman and Kameswari Chebrolu. Revisiting mac design for an 802.11-based mesh network. In *Proc. of ACM HotNets*, November 2004.

[76] Bhaskaran Raman and Kameswari Chebrolu. Design and evaluation of a new mac protocol for long-distance 802.11 mesh networks. In *Proc. of ACM Mobicom*, August 2005.

[77] R. Ramanathan. On the performance of ad hoc networks with beamforming antennas. In *Proc. of ACM MobiHoc*, October 2001.

[78] Ashish Raniwala and Tzicker Chiueh. Architectures and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *Proc. of IEEE INFOCOM*, March 2005.

[79] Ashish Raniwala, Kartik Gopalan, and Tzicker Chiueh. Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. *ACM MC2R*, 8(2), April 2004.

[80] Ananth Rao and Ion Stoica. An overlay mac layer for 802.11 networks. In *Proceedings of Mobisys 2005*, April 2005.

[81] Alex Rousskov and Duane Wessels. Cache digests. In *Proc. of WCW*, 1998.

[82] Fran&#231;oise Sailhan and Val&#233;rie Issarny. Cooperative caching in ad hoc networks. In *Proc. of MDM*, 2003.

[83] Naouel Ben Salem and Jean-Pierre Hubaux. A fair scheduling for wireless mesh networks. In *Proc. of WiMesh*, 2005.

[84] Jingpu Shi, Theodoros Salonidis, and Edward W. Knightly. Starvation mitigation through multi-channel coordination in csma multi-hop wireless networks. In *Proc. of ACM MobiHoc*, 2006.

[85] S.M.Das, H.Pucha, D. Koutsonikolas, Y.C.Hu, and D. Peroulis. DMesh: Incorporating Practical Directional Antennas in Wireless Mesh Networks. Technical report, TR-ECE-05-16, Purdue University, August 2005.

[86] Jungmin So and Nitin Vaidya. Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In *Proc. of ACM MobiHoc*, 2004.

[87] Jungmin So and Nitin H. Vaidya. A routing protocol for utilizing multiple channels in multi-hop wireless networks with a single transceiver. In *Tech. Rep. UIUC*, October 2004.

[88] Martin Swany. Improving Throughput for Grid Applications with Network Logistics. In *Proc. of SC*, 2004.

[89] M. Takai, J. Martin, A. Ren, and R. Bagrodia. Directional virtual carrier sensing for directional antennas in mobile ad hoc networks. In *Proc. of ACM MobiHoc*, June 2002.

[90] A. Tzamaloukas and J. J. Garcia-Luna-Aceves. A receiver-initiated collision-avoidance protocol for multi-channel networks. In *Proc. of IEEE INFOCOM*, 2001.

[91] Nitin Vaidya and Paramvir Bahl. Fair scheudling in broadcast environments. Technical report, TR-99-61, Microsoft Research, Dec 1999.

[92] Vinod Valloppillil and Keith W. Ross. Cache array routing protocol v1.0. Internet draft, 1998.

[93] Duane Wessels and K Claffy. ICP and the Squid Web cache. *IEEE JSAC*, 16(3), 1998.

[94] Jeffrey E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides. Energy-efficient broadcast and multicast trees in wireless networks. *Mob. Netw. Appl.*, 7(6), 2002.

[95] Alec Wolman, Geoff Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry Levy. On the scale and performance of cooperative Web proxy caching. In *Proc. of ACM SOSP*, 1999.

[96] Shih-Lin Wu, Chih-Yu Lin, Yu-Chee Tseng, and Jang-Ping Sheu. A new multi-channel mac protocol with on-demand channel assignment for multi-hop mobile ad hoc networks. In *Proc. of I-SPAN*, 2000.

[97] Y. Wu, P.A. Chou, and S.-Y. Kung. Information exchange in wireless networks with network coding and physical-layer broadcast. In *Proc. of CISS*, 2005.

[98] Yunnan Wu, Saumitra M. Das, and Ranveer Chandra. Routing with a markovian metric to promote local mixing. In *Proc. of INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, May 2007.

[99] Yunnan Wu, Jitendra Padhye, Ranveer Chandra, Venkat Padmanabhan, and Philip A. Chou. The local mixing problem. In *Proc. Information Theory and Applications Workshop*, San Diego, CA, February 2006. Univ. of California, San Diego.

[100] Luqing Yang, Winston K.G. Seah, and Qinghe Yin. Improving fairness among tcp flows crossing wireless ad hoc and wired networks. In *Proc. of ACM MobiHoc*, 2003.

[101] Xue Yang and Nitin H. Vaidya. Priority scheduling in wireless ad hoc networks. In *Proceedings of ACM Mobihoc*, 2002.

[102] Y. Yang, J. Wang, and R. Kravets. Designing routing metrics for mesh networks. In *Proc. of WiMesh*, 2005.

[103] Y. Yang, J. Wang, and R. Kravets. Interference-aware load balancing for multihop wireless networks. In *Technical Report UIUCDCS-R-2005-2526, Dept. of Comp. Science, UIUC*, 2005.

[104] Liangzhong Yin and Guohong Cao. Supporting cooperative caching in ad hoc networks. In *Proc. INFOCOM 2004*, 2004.

[105] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *Proc. of PADS Workshop*, May 1998.

[106] Champaign-Urbana community wireless network. `http://www.cuwireless.net`.

[107] MIT Roofnet. `http://www.pdos.lcs.mit.edu/roofnet`.

[108] Netfilter. http://www.netfilter.org.

[109] Southampton wireless network. `http://www.sown.org.uk`.

[110] Squid Web Cache. http://www.squid-cache.org.

[111] `http://www.engineering.purdue.edu/MESH`.

[112] Wireless leiden. `http://www.wirelessleiden.nl`.

[113] Seattle wireless. `http://www.seattlewireless.net`.

[114] Bay area wireless users group. `http://www.bawug.org`.

VITA

VITA

Saumitra M. Das was born in Agra, India on September 8th, 1978. He graduated from Kendriya Vidyalaya, Khanapara, Assam in 1994 and Jai Hind College, Mumbai in 1996. He then received a Bachelor of Engineering degree in Electronics Engineering from the University of Mumbai, V.E.S. Institute of Technology, India in 2000. Saumitra initiated his graduate studies by earning a Master of Science degree from Carnegie Mellon University, Pittsburgh, USA in 2002 working with Dr. Rohit Negi in Electrical and Computer Engineering. Saumitra received his Ph.D. degree from the School of Electrical and Computer Engineering at Purdue University working with Dr. Y. Charlie Hu in 2007.

Saumitra's research interests are in wireless communication and its application to mesh networks, sensor networks and mobile robotics. His doctoral work has resulted in more than 35 research publications including 2 book chapters, over 10 journal papers and over 20 conference papers in premier forums for wireless networking, distributed systems and robotics. Saumitra's work can be found in the following selected papers:

Journal Papers and Book Chapters

[1] H. Pucha, S.M.Das and Y.C. Hu, Imposed Route Reuse in Ad Hoc Network Routing Protocols using Structured Peer-to-Peer Overlay Routing, In *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, Vol 17(12), pp. 1452-1467, December 2006.

[2] S.M.Das, H. Pucha, D. Koutsonikolas, D. Peroulis and Y. C. Hu, DMesh: Incorporating Practical Directional Antennas in Multi-Channel Multi-Radio Wireless Mesh Networks, In *IEEE Journal on Selected Areas in Communication (JSAC),*

Special Issue on Multi- Hop Wireless Mesh Networks, Vol 24(11), pp. 2028-2039, November 2006.

[3] S.M.Das, H. Pucha and Y. C. Hu, Microrouting: A Scalable and Robust Communication Paradigm for Sparse Ad Hoc Networks, In *International Journal of Distributed Sensor Networks (IJDSN)*, Vol 2(1), pp. 79-100, January-March 2006. Previous version in *IEEE IPDPS WMAN workshop 2005.*

[4] S.M.Das, Y.C. Hu, C.S.G. Lee, and Y-H. Lu, Efficient Communication Protocols for Networking Mobile Robot Teams, In *IEEE Journal of Communication and Networks (JCN)*, Vol. 9(3), pp. 296-311, September 2007. Previous versions in *IEEE ICRA 2004 and IEEE ICRA 2005.*

[5] S.M.Das, H. Pucha and Y.C. Hu, Mitigating the Gateway Bottleneck via Transparent Cooperative Caching in Wireless Mesh Networks, In *Elsevier Ad Hoc Networks Journal (AHN), Special Issue on Wireless Mesh Networks*, Vol 5(6), pp. 680-703, August 2007.

[6] H. Pucha, S.M.Das and Y.C. Hu, The Performance Impact of Traffic Patterns on Routing Protocols in Mobile Ad Hoc Networks, In *Computer Networks Journal*, Vol 51(12), pp. 3595- 3616, August 2007. Previous version in *ACM MSWiM 2004.*

[7] S.M.Das, H. Pucha and Y.C. Hu, Distributed Hashing for Scalable Multicast in Wireless Ad Hoc Networks, to appear in *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2007.

[8] S.M.Das, H. Pucha and Y.C. Hu, On the Scalability of Location Services for Geographic Ad Hoc Routing, In *Computer Networks Journal (COMNET)*, Vol 51(13), pp. 3693-3714, September 2007. Previous version in *IEEE INFOCOM 2005.*

[9] S. Roy, D. Koutsonikolas, S.M.Das and Y. C. Hu, High-Throughput Multicast Routing Metrics in Wireless Mesh Networks, to appear in *Elsevier Ad Hoc Networks Journal*, 2007.

[10] D. Koutsonikolas, S.M.Das and Y.C. Hu, An Interference-Aware Fair Scheduling for Multicast in Wireless Mesh Networks, to appear in *Elsevier Journal of Parallel and Distributed Computing, Special Issue on Wireless Mesh Networks*, 2007.

[11] Y.C.Hu, S.M.Das and H. Pucha, Peer-to-Peer Overlay Abstractions in MANETs, Book Chapter, in Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, Jie Wu, editor, CRC Press, 2005.

[12] S.M.Das, D. Koutsonikolas and Y.C.Hu, Measurement-based Characterization of a Wireless Mesh Network, Book Chapter, in Handbook of Wireless Mesh and Sensor Networking, McGraw-Hill International, NY, 2007.

Conference and Workshop Papers

[13] S.M.Das, D. Koutsonikolas and Y.C. Hu, Practical Service Provisioning for Wireless Meshes, In *3rd ACM SIGCOMM International Conference on emerging Networking EXperiments and Technologies (CoNEXT 07)*, 2007.

[14] S.M.Das, H.Pucha, K. Papagiannaki and Y.C .Hu, Studying Wireless Routing Link Metric Dynamics, In *7th ACM SIGCOMM/USENIX Internet Measurement Conference (IMC 07)*, 2007.

[15] Y. Wu, S.M.Das and R. Chandra, Routing with a Markovian Metric to Promote Local Mixing, In *IEEE INFOCOM*, 2007.

[16] S.M.Das, D. Koutsonikolas, Y. C. Hu and D. Peroulis, Characterizing Multi-Way Interference In Wireless Mesh Networks, In *ACM MobiCom International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WiNTECH 2006)*, 2006.

[17] S. Roy, D. Koutsonikolas, S.M.Das and Y. C. Hu, High-Throughput Multicast Routing Metrics in Wireless Mesh Networks, In *26th IEEE International Conference on Distributed Computing Systems (ICDCS 06)*, 2006.

[18] S.M.Das, H. Pucha and Y. C. Hu, Symmetrical Fairness in Infrastructure Access in Multi- Hop Wireless Networks, In *25th IEEE International Conference on Distributed Computing Systems (ICDCS 05)*, 2005.

[19] S.M.Das, H. Pucha and Y. C. Hu, Performance Comparison of Scalable Location Services for Geographic Ad Hoc Routing, In *IEEE INFOCOM 05*, 2005.

[20] H. Pucha, S.M.Das and Y.C. Hu, Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks, In *6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 04)*, 2004.

[21] H. Pucha, S.M.Das and Y.C. Hu, The Performance Impact of Traffic Patterns on Routing Protocols in Mobile Ad Hoc Networks, In *7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 04)*, 2004.

[22] Y.C.Hu, S.M.Das and H. Pucha, Exploiting the Synergy between Peer-to-Peer and Mobile Ad Hoc Networks, In *9th Workshop on Hot Topics in Operating Systems (HOTOS 03)*, 2003.