The Dissertation Committee for Yi Li
certifies that this is the approved version of the following dissertation:

Model-driven Optimization of Multihop Wireless Networks

Committee:

Yin Zhang, Supervisor

Simon S. Lam, Supervisor

Lili Qiu

Mohamed G. Gouda

Gustavo de Veciana

Model-driven Optimization of Multihop Wireless Networks

by

Yi Li, B.E., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2008

To my parents, sister, and teachers

Acknowledgments

First and foremost, my deep appreciation goes to Prof. Yin Zhang, Prof. Simon S. Lam and Prof. Lili Qiu. My PhD work is under their supervision and support. From them, I have learned how excellent networking research is done - rigorous, focused, motivated, open to new ideas, and involved into research.

I would like to express my appreciation to Prof. Mohamed G. Gouda, and Prof. Gustavo de Veciana for serving on my committee.

I owe a special gratitude to my family. My parents and sister always give me selfless care, support, and love.

I would also like to thank my colleagues and members of my research group, especially Min Sik Kim, Xincheng Zhang, Huaiyu Liu, DongYoung Lee, Feng Wang, Eric Rozner, Zifei Zhong, and Gaurav Deshpande.

This research was supported in part by National Science Foundation Research grants CNS-0434515, CNS-0627020, CNS-0546755, ANI-0319168, CNS-0546720, and Microelectron Computer Development Fellowship from the Department of Computer Sciences of the University of Texas at Austin.

Model-driven Optimization of Multihop Wireless Networks

Publication No. _____

Yi Li, Ph.D.
The University of Texas at Austin, 2008

Supervisors: Yin Zhang
Simon S. Lam

Abstract – Interference is fundamental to wireless networks. It is hard to achieve good performance when design routing metrics or algorithms without taking it into account. We study interference in wireless networks through empirical experiments and simulations. We find out that current routing protocols face difficulties in effectively managing it, which can lead to severe problems. For instance, a simple network of two links with one flow is vulnerable to severe performance degradation if interference is not properly accounted for. Motivated by these observations, we develop a simple and effective model to capture effects of interference in a wireless network. Different from the existing interference models, our model captures IEEE 802.11 DCF under both homogeneous and heterogeneous traffic and link characteristics, and is simple enough to be directly used as a basic building block for wireless performance optimization. Based on this model, we develop optimization algorithms for several objectives, such as network throughput and fairness. Given traffic demands as input, these algorithms compute rates at which individual

flows must send to achieve these objectives. We implement these algorithms in Qualnet simulations and 19-node testbed. Our experiment and simulation results show that our methods can systematically account for and control interference to achieve good performance. More specifically, when optimizing fairness, our methods can achieve almost perfect fairness; when optimizing network throughput, they can lead to 100-200% improvement for UDP traffic and 10-50% for TCP traffic.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Multi-hop wireless networks are becoming increasingly ubiquitous in the form of city-wide mesh networks [37, 36]. These networks have witnessed significant research and deployment activities recently. Many researchers have focused on improving their throughput through better routing [4, 5, 40, 32].

However, the performance of these networks today leaves much to be desired [33, 35, 34]. Users of almost all existing deployments have been complaining about poor performance. In many cases, complaints occurred even when the users are close to the BSs, which suggests that the routing backhaul formed by the BSs might be a major contributor [33].

A fundamental property that distinguishes wireless networks and wired networks is the presence of interference. Most routing protocols for wireless mesh networks pay little attention to directly managing interference. Early protocols such as AODV [26], DSDV [25] and DSR [14] ignore interference and simply implement shortest hop-count routing. The next generation of protocols such as ETX [4] , ETT [5] and WCETT [5] route based on measured link quality. The quality of a link can capture some interference effects, such as packets collision or even hidden terminal, but it cannot systematically capture all interference effects.

We start our work from studying ill-effects of interference on behaviors of wireless networks through simulations and testbed experiments. We uncover two problems which lead us to understand interference behaviors of wireless nodes. The first problem is that not controlling how much nodes send can severely degrade network throughput if they send more than what a path can support. This occurs because, due to interference, any additional traffic on a link can reduce the capacity of other links. We show how the degradation can be sharp even in a simple setting of a single flow traversing two links. We also show that end-to-end congestion control (e.g., using TCP) is not sufficient by itself to prevent this behavior.

The second problem is that current protocols are unable to accurately estimate link and path quality for purposes of path selection. The underlying issue is that quality is measured by sending probes, without considering interference. The probes measure quality under current routing patterns but, due to interference, the quality can change arbitrarily with any change in the routing pattern. As such, these measurements have limited predictive values because they cannot tell whether rerouting existing flows would result in better network throughput or which path is best for a new flow.

Due to these problems, we find out that extracting predictable performance from or managing these networks today is notoriously hard. A single new flow can lead to a disproportionate decline in network performance. And attempts to increase network performance, for instance by adding relay nodes to shorten link distances, can end up reducing performance. This is in sharp contrast to wireline network management, where network operators have many effective techniques to

predict and improve performance.

We seek to develop analogous techniques for multi-hop wireless networks. As motivating examples, we focus on three basic capabilities that are not available today.

- Network operators should be able to tell if the network can support the current or a planned traffic demand.
- They should be able to perform "what if" analyses to evaluate the impact of configuration changes such as addition of new flows or routing changes.
- They should be able to determine safe sending rates of various flows based on policy and path capacity.

In order to achieve these capabilities, we need an accurate while simple model. However, despite much work on interference and MAC modeling, none of the existing models for multi-hop networks fulfill our need. Many existing models make simplifying assumptions about signal propagation [12], traffic [10, 31, 9], topology [2, 17, 9, 10], or the MAC [13]. These assumptions often do not hold for real networks [16]. Other models are too complex to be used directly for optimization because they require enumeration of all possible network configurations [27].

We develop a new model that captures the complex interference- and MAC-induced dependencies in a network. These dependencies are underlying causes of unpredictable behaviors.

The model that we develop strikes a balance between simplicity and realism. It targets the widely used CSMA/CA-based 802.11 MAC. Based on easily collected

measurements from a network itself, it characterizes the set of feasible network configurations and traffic assignments using a small set of constraints. Despite its simplicity, our model can handle real-world complexities such as hidden terminals, non-uniform traffic, and non-binary interference.

We then develop optimization algorithms to compute rate-limits for flows according to a specified performance objective. These algorithms take flow demands as input and use our model as a basic building block. Two performance objectives that we consider in our work are maximizing fairness and maximizing total network throughput. To our knowledge, such goal-driven and precise optimization for multi-hop wireless networks was not possible before.

We evaluate our model and optimization algorithms using a multi-hop wireless testbed and simulation experiments. The results show that our methods are highly effective. Across a range of topology and traffic configurations, they are able to accurately approximate the throughput that a network yields. They rarely under-predicts, and for 80% of the cases, their estimations are within 20% of actual throughputs. When maximizing fairness using our methods, we achieve close to perfect fairness amongst flows for both UDP and TCP traffic. When maximizing throughput, we find that our methods can improve network throughput by more than 100% for UDP-based traffic and 25% for TCP-based traffic. Interestingly, we find that in our experiments the exact of choice of routing protocol is not important to achieve good performance for objectives we study, such as total throughput, and fairness. What matters instead is that flows be rate-limited per the desired performance goal.

Chapter 2

Related Work

2.1 Wireless Mesh Networks

Wireless mesh networks seek to build a resilient and high-performance infrastructure to provide users pervasive Internet access. In a mesh network, each client accesses a local high-speed access point (HAP), and multiple stationary HAPs communicate with one another over a wireless channel and form a multihop, wireless backbone for data delivery. This backbone eventually forwards users' traffic to a few gateway APs (GAPs) that additionally connect to wired Internet.

Compared to ad-hoc wireless networks, mesh networks are composed of static wireless nodes that have ample energy supply. Each wireless node can be equipped with multiple radios and each radio can be configured to a different channel to enhance network capacity.

2.2 Routing Metrics

2.2.1 HOP

Many traditional routing protocols, such as DSDV, AODV and DSR, use hop count as the routing metric. Hop count reflects the effects of path length on the performance of flows, however, it does not consider the difference of transmission

rates and packet loss ratios among different wireless links, and also interference in a wireless network. This can result in some poor performance paths which have high loss ratio.

2.2.2 Expected Transmission Count ETX

ETX is defined as the expected number of MAC layer transmissions needed for successfully delivering a packet through a wireless link. The weight of a path is defined as the summation of ETXs of all links along the path. The ETX of a link is calculated using the forward and reverse delivery ratios of the link . Let p_f denote the measured probability that a data packet successfully arrives at the recipient; Let p_r denote the probability that the ACK packet is successfully received. Then the expected transmission count

$$ETX = \frac{1}{p_f \times p_r} \tag{2.1}$$

ETX can capture link quality, but it does not include interference or the fact that different links may have different transmission rates.

2.2.3 Expected Transmission Time ETT

To improve ETX, Draves et al. propose ETT [5] metric, which considers difference in link transmission rates. The ETT of a link is defined as the expected MAC layer duration of a successful transmission of a packet on this link. The relationship between the ETT of a link l and its ETX can be expressed as

$$ETT_l = \frac{ETX_l \times s}{b_l} \tag{2.2}$$

where b_l is the transmission rate of link l and s is the packet size. However, ETT does not consider flow interference either.

2.2.4 Weighted Cumulative Expected Transmission Time WCETT

To reduce intra-flow interference, Draves et al [5] propose WCETT, which reduces the number of nodes on the path of a flow that transmit on the same channel. The WCETT metric of a path p is defined as follows:

$$WCETT_p = (1-\alpha) \times \sum_{i \in p} ETT_i + \alpha \times \max_{1 \leq j \leq K} X_j \tag{2.3}$$

Where X_j is the summation of ETT of the links in path p operating on channel j; k is the number of orthogonal channels available and $0 \leq \alpha \leq 1$ is a tunable parameter. The first component in the WCETT metric helps in finding path with links having less ETT. The second component improves the channel diversity and helps in finding paths with less intra-flow interference. WCETT does not explicitly consider the effects of inter-flow interference, although it does heuristic way to reduce intra-flow interference. Therefore, WCETT may route flows to dense areas where congestion is more likely.

2.2.5 Metric of Interference and Channel Switching MIC

MIC [40] considers inter-flow interference. MIC for a path p is defined as follows:

$$MIC(p) = \frac{1}{N \times min(ETT)} \sum_{link\ l\ \in p} IRU_l + \sum_{node\ i\ \in p} CSC_i \tag{2.4}$$

where N is the total number of nodes in the network and min(ETT) is the smallest ETT in the network. The two components of MIC, IRU (Interference-aware Resource Usage) and CSC (Channel Switching Cost), are defined as follows:

$$IRU_l = ETT_l \times N_l \tag{2.5}$$

$$CSC_i = \begin{cases} w_1, & \text{if CH(prev(i))} \neq CH(i) \\ w_2, & \text{if CH(prev(i)) = CH(i)} \end{cases} \tag{2.6}$$

$$0 \leq w_1 \ll w_2 \tag{2.7}$$

Where N_l is the set of neighbors that interfere with the transmissions of link l. CH(i) represents the channel assigned for node i's transmission and prev(i) represents the previous hop of node i along the path p. MIC incorporates inter-flow interference by scaling up the ETT of a link by the number of neighbors interfering with the transmission on that link. However, the degree of interference caused by each interfering node on a link is not the same. And also interference is also depends on how active the interfering node, that is to say, it depends on traffic generated by the node. MIC fails to capture these characteristics of interference.

2.2.6 Per-hop Round Trip Time RTT

RTT [1] is based on one-hop round trip time between a pair of hosts to determine the quality of links between those hosts. A link in a congested region or a lossy link usually has a large RTT, so RTT can help to avoid highly loaded or lossy links. But as it is a load-dependent metric, it is inevitable that RTT can lead to route instability.

2.2.7 iAWARE

iAWARE [32] is an interference aware routing metric. It uses SNR and SINR of a node to estimate the interference observed by a node. More specifically, it defines interference ratio $IR_i(u)$, $(0 < IR_i(u) \leq 1)$, for a node u in a link $i = (u, v)$ as follows:

$$IR_i(u) = \frac{SINR_i(u)}{SNR_i(u)} \tag{2.8}$$

And then iAware define the metric of a link j as follows:

$$iAWARE_j = \frac{ETT_j}{IR_j} \tag{2.9}$$

where $IR_j = min(IR_j(u), IR_j(v))$

Intuitively, the higher IR_j, the less the link is interfered. iAWARE captures the receiver-side interference, but it does not fully capture the sender-side interference.

2.3 Wireless Routing Protocols

A lot of protocols has been proposed to solve multihop routing problems in wireless networks.

2.3.1 Destination-sequenced Distance Vector DSDV

DSDV uses the distance vector shortest path algorithm to select a single path to a destination. Every node maintains a routing table that lists all available destinations, the number of hops to reach a destination and a sequence number

assigned by the destination node. The sequence number is used to distinguish stale routes from new ones. Each node periodically transmits their routing tables to their immediate neighbors. DSDV is suitable for creating ad hoc networks with small number of nodes.

2.3.2 Ad hoc On-demand Distance Vector Routing AODV

AODV is an improvement on the DSDV. Each node finds routes to destinations on-demand as opposed to DSDV that maintains the list of all routes. A source nodes broadcasts a route request message to initialize a path discovery process if it does not have a valid route to a destination. Its neighbors forward the request to their neighbors until either the destination or an intermediate node with a fresh enough route to the destination. During the process of forwarding, intermediate nodes record the address of the neighbor from which the first copy of the broadcast is received in their route tables. By this way, intermediate nodes can establish a reverse path for the route reply message to reach the source node.

2.3.3 Dynamic Source Routing DSR

DSR is an on-demand routing protocol that is based on the concept of source routing. Each node maintains route caches that contain source routes of which this node is aware. The protocol consists of two major phases: route discovery and route maintenance. When a node has a packet to send to some destination, it first consults its route cache to determine whether it already has a route to the destination. If it has an unexpired route to the destination, it uses it. otherwise, it

initiates route discover by broadcasting a route request packet. Each intermediate node checks whether it knows of a route to the destination. If it does not, it forwards the packet to its neighbors. A route reply is generated when either the destination or an intermediate node with current information about the destination receives the route request packet.

2.3.4 Link Quality Source Routing LQSR

LQSR [30] is a link-state routing protocol which uses a complete view of the network topology to compute shortest paths. Each node periodically broadcasts its link-state advertisements. In addition, it also uses a route discovery procedure as in DSR to reduce broadcasting overhead. During route discovery, LQSR obtains up-to-date link state information of the traversed links. LQSR uses WCETT as the routing metric to define the best path from a source to a destination.

2.4 Interference Modeling

There is a rich body of work on modeling wireless interference.

In [2], Bianchi presents a simple analytical model to compute the saturation throughput performance of the 802.11 Distributed Coordination Function. The model assumes a finite number of nodes and ideal channel conditions. It also assumes all nodes can carrier sense each other and each node has saturate demand. The work provides a fundamental model on a wireless network. But it only considers single cell WLANs and has specific traffic demands.

In [12],Gupta and Kumar study the capacity of wireless networks under

models of interference: a protocol model that assumes interference to be an all-or-nothing phenomenon and a physical model that considers the impact of interfering transmissions on the signal-to-noise ratio. They show that in a network comprising of n identical nodes, each of which is communicating with another node, the throughput per node is $\theta(\frac{1}{sqrt(n*(logn))})$ assuming random node placement and $\theta(\frac{1}{sqrt(n)})$ assuming optimal node placement and communication pattern.

The work of [13] defines a conflict graph, F, whose vertices correspond to the links in the connectivity graph. There is an edge between the vertices l_{ij} and l_{pq} in F if the links l_{ij} and l_{pq} can not be active simultaneously. l_{ij} and l_{pq} cannot be active at the same time if any of the following is true. l_{ij} and l_{pq} have a node in common; node i can carrier sense p; node p can carrier sense i; node i is in the interference range of node q or node l is in the interference range of node j. The work assumes optimal scheduling, which is hard to achieve in real wireless network.

The work of [20] presents models for the physical layer behaviors of packet reception and carrier sense with interference in a static wireless network. They use measurement of a real network as input, and aim to model interference in a general network topology where not all nodes are within communication range. Their work models two competing broadcast senders.

The work of [27] provides a general wireless model to estimate throughput and goodput in the presence of interference. Their model is based on interference measurement in an N-node network, which is more accurate than abstract models of RF propagation such as those based on distance. They model the more common

case of unicast and heterogeneous nodes with different traffic demands and different radio characteristics. Their work targets on one-hop demand.

2.5 Rate Control

The importance of rate control and scheduling has been well recognized. The work of [19] presents a framework for joint optimization of rate control and scheduling in multihop wireless networks. They propose a dual approach through which the rate control problem and the scheduling problem can be decomposed. But their work is not based on 802.11 MAC scheduling and hard to achieve in a real wireless network.

IFRC [29] enables fair rate control for sensor networks in which all nodes send traffic towards one or more sinks. Nodes detect congestion by measuring their average queue length, and adapt their rates according to an AIMD control law. Their work is specific to tree topologies and sensor network workloads.

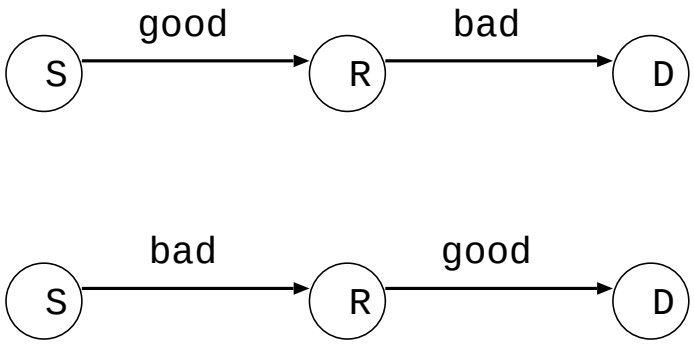


Figure 3.1: Two topologies that differ in where the lossy link occurs

Chapter 3

Pathologies Caused by Interference

In this section, we use simulations and testbed experiments to show problems with current routing protocols for wireless mesh networks.

3.1 Sensitivity of Wireless Network Throughput to Bottleneck Link Location

Current routing protocols provide no feedback as to how much traffic a node can send. In this section, we show that lack of rate feedback can lead to severe performance degradation and even congestion collapse. By congestion collapse, we refer to a situation in which the goodput of the network decreases sharply when the load is increased beyond a certain point.

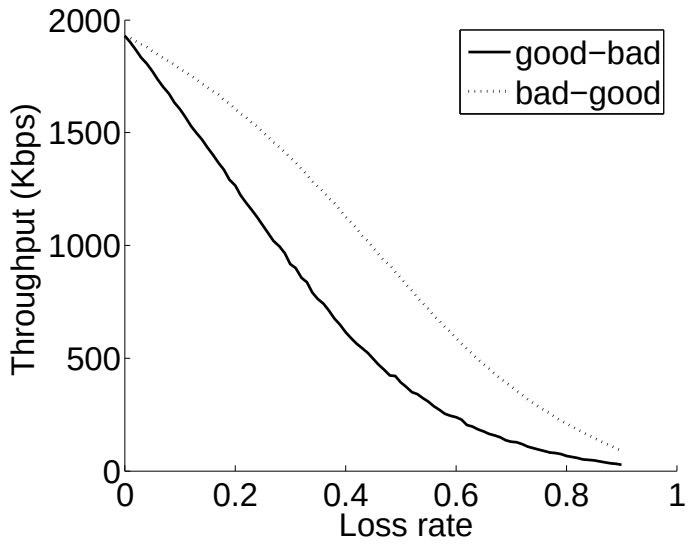


Figure 3.2: Throughput as a function of loss rate when *S* sends as fast as possible

We illustrate our point using the two simple topologies in Figure 3.1. Both have one reliable ("good") link and one lossy ("bad") link but the order of the two links is different. Using QualNet [28], we simulated the case of *S* sending 512-byte UDP packets to *D* as fast as possible. Unless otherwise specified, our evaluation uses 802.11a and 6Mbps MAC data rate.

Figure 3.2 shows that the throughput of the two topologies as a function of loss rate on the bad link are very different. At a loss rate of 0.5, the throughput of the good-bad topology is less than half of the bad-good topology.

The reason for this disparity is the following. For a successful reception in the good-bad topology, *S* needs to transmit a packet to *R* only once, but *R* has to transmit to *D* more than once. Since *R* sees more packet loss due to the lossy link

between *R* and *D*, *R* has a larger expected contention window than *S*, which makes *R* has smaller probability to access the medium to transmit a packet. As a result, the 802.11 MAC allocates more air time to *S* than *R* under saturated demands. So the incoming traffic at *R* is more than the outgoing traffic and many packets sent by *S* are eventually dropped at *R* due to queue overflow. These wasted transmissions of *S* compete with *R* for air time and reduce the throughput of the good-bad topology. Such wastage does not exist in the bad-good topology because *R* can send all incoming traffic. To our knowledge, this sensitivity of wireless network throughput to bottleneck link location has not been reported previously.

This problem cannot be solved by RTS/CTS because both transmitters can hear each other and there is no hidden terminal. Moreover, simply changing the MAC allocation policy will not fix the problem in the general case because the bottleneck can be multiple hops away from the source.

The wastage in a good-bad topology can lead to a very sudden decline in throughput as the sending rate is increased. Figure 3.3 plots the throughput of the two topologies as *S* increases its sending rate. The loss rate of the bad link is configured to 0.5. In the good-bad topology, increasing the sending rate beyond a threshold sharply degrades throughput. This threshold represents the sending rate of *S* at which *R* receives enough air time to relay all received packets. Beyond it, *R* cannot keep up as it receives less air time and the medium is increasingly occupied by the transmissions from *S* that are eventually dropped. The throughput stabilizes when the air time utilization of *R* decreases to half.

The graph also shows that the two topologies have the same maximum ca-

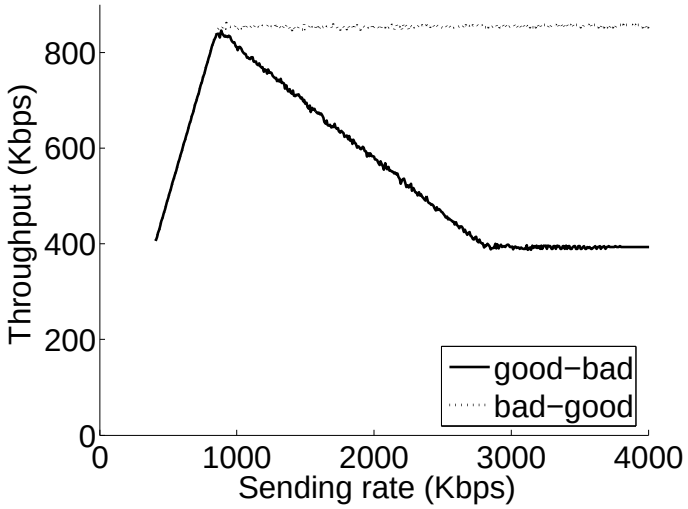
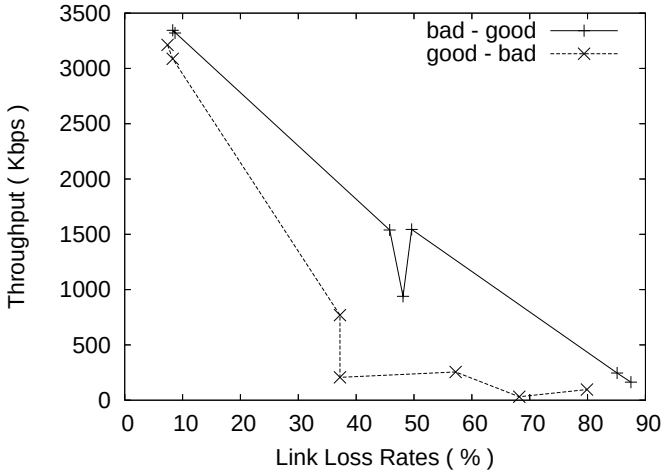


Figure 3.3: Throughput as a function of the sending rate when the loss rate of the bad link is 0.50

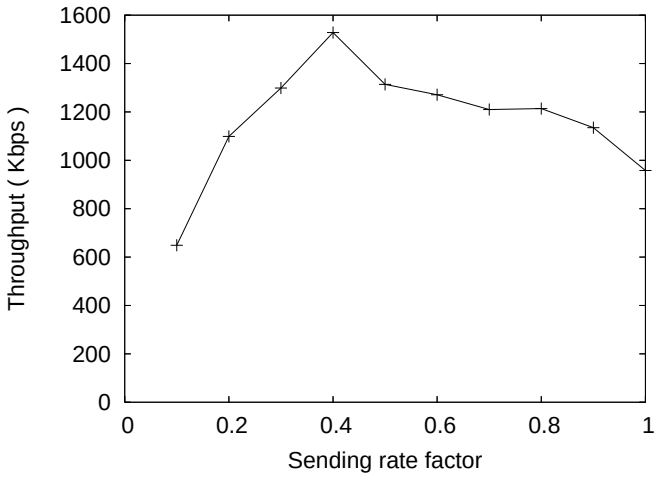
pacity, but in the good-bad case, it can be achieved only if we limit S to the threshold sending rate. However, none of the current routing protocols give rate feedback. Moreover they cannot even distinguish between these two paths. The path quality as measured by current protocols will be the same for both topologies.

This sharp decline in throughput is reminiscent of congestion collapse in the Internet. But it is unique in that it can be caused by a single flow over a very simple topology. Known examples of congestion collapse in wired networks [7] involve more flows and complex topologies. A key difference is that the capacity of the bottleneck link in a wired network is not impacted by other links, but in wireless networks interference reduces bottleneck capacity when other links are active.

Figure 3.4 confirms that the effect above is present in the more realistic testbed setting as well. We emulate different loss rates in the testbed by changing



(a) Throughput vs. loss rates in the two topologies



(b) Throughput vs. sending rate in the good-bad topology

Figure 3.4: Testbed experiments confirm the importance of rate feedback.

the distance between the machines and varying layers of foils around the wireless cards. Figure 3.4(a) shows that the two topologies perform differently when *S* sends as fast as possible. Figure 3.4(b) shows the sudden throughput decline in the good-bad topology when the bad link has roughly 50% loss. The *x*-axis in this graph

denotes the fraction of the fastest possible sending rate (*e.g.*, sending rate factor $= 1$ indicates that the source sends packets back-to-back). The curve is not as smooth because the loss rate in the testbed cannot be precisely controlled. Overall, these results confirm the ill-effects of not providing rate feedback.

We further study the effect of bottleneck location in different topologies or traffic patterns. Our results show that congestion collapse can arise in diverse set of scenarios.

3.1.1 Scenario 1: Linear Topology

Bottleneck link	throughput (Mbps)
1-2	1.013
2-3	0.403
3-4	0.245
4-5	0.798
5-6	0.789

Table 3.1: Throughput of a CBR flow under a varying bottleneck location in a linear chain topology 1-2-3-4-5-6, where node 1 sends to node 6, and the bottleneck link has 50% loss rate and the other links are reliable and only subject to collision losses.

We study the effect of bottleneck location in 5-hop network topology. Table 3.1 shows that the bottleneck location significantly affects network performance in a more general linear-chain topology. The performance does not monotonically decrease as the bottleneck moves closer to the destination. This is because on one hard, an earlier bottleneck limits the wasteful transmissions, on the other hand, an earlier bottleneck has more packets going through it and hence the total number of packets dropped is also higher. Nevertheless applying rate limit can result in higher throughput when the bottleneck location is anywhere but the first hop. When the

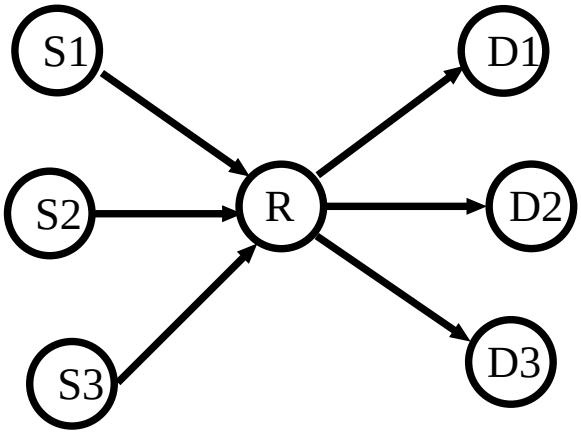


Figure 3.5: A star topology

bottleneck is at the first hop, effectively it does rate limit.

3.1.2 Scenario 2: Star Topology

We first show how the structure of the topology itself can lead to congestion collapse. Consider the star topology shown on the left in Figure 3.5. All sources (S_i) and the relay can carrier sense each other, and each source is sending traffic to its corresponding destination. Every link is reliable and has unit capacity when active by itself. Since R can either send or receive at any given instant, the maximum capacity of this network is 1/2.

As sources slowly increase their sending rate to a point where R is receiving half of the time, the network throughput reaches its capacity. Beyond that it declines sharply. With sources sending as fast as possible, the network throughput is 1/4, which is R's share of the medium. Thus, without rate-limiting, the topology experiences an efficiency loss of a factor of two. When the number of flows sharing

a relay increases, the loss in efficiency is higher. If N flows share the same relay and the sources send as fast as possible, the relay node gets $\frac{1}{N+1}$ of the airtime which is also the total throughput. But if each flow is rate limited to $\frac{1}{2N}$, the relay node can get $\frac{1}{2}$ of the airtime which is also the total throughput. The throughput degradation without rate-limiting is, thus, $\frac{N+1}{2}$. Because this factor increases with N, in theory, the benefit of rate-limiting can be arbitrarily large. By rate limiting each source to 1/6, we can achieve the maximum capacity. Simulations confirm this effect. Without rate limit, the network throughput with UDP flows is 1.25 Mbps. With rate limit, it is 2.16 Mbps, an improvement of 73%. The improvement is slightly less than a factor of 2 because in simulation R manages to get 29% of the airtime which is slightly higher than its fair share of 25%.

Such bottlenecks can be either present in the topology itself or created by the routing protocol, if it tries to route many flows through a single relay. The ETX and ETT metric of each path that uses R is 2. Suppose there were an alternate path in this topology that did not go through R. These protocols will continue using R unless the alternate path had a lower metric. But taking a broader view of topology, using a path with a higher metric may sometimes be preferable because of the effects shown above.

TCP traffic Similar problems occur with TCP as well because TCP's built-in rate control and congestion response are not well-suited for the wireless environment. Consider a star topology shown in Figure 3.7, where all links are reliable. There are two competing TCP flows $1 \rightarrow 5$ and $2 \rightarrow 4$. We find performance degradation due to overload when the central node cannot relay all the traffic sent by its neighbors.

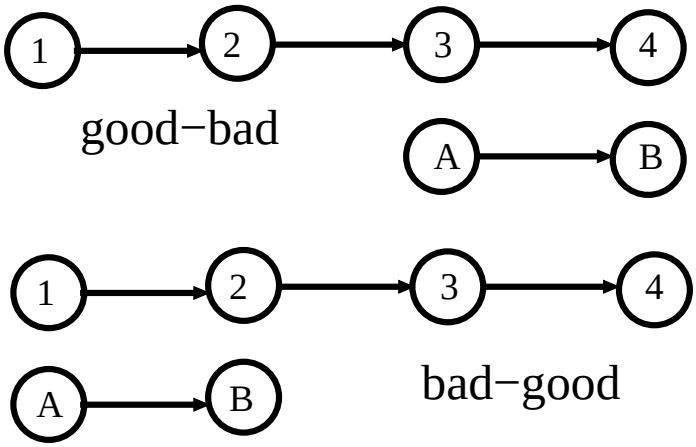


Figure 3.6: The topologies for the TCP case. The TCP flow goes from 1 to 4. The UDP flow goes from *A* to *B*.

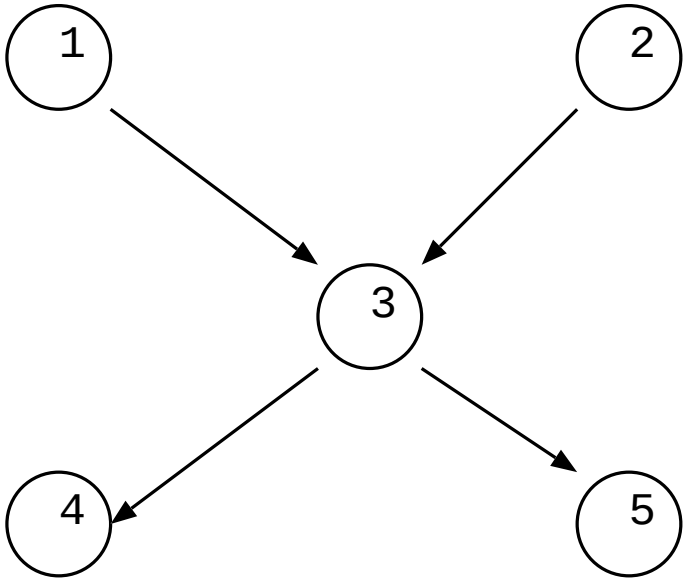


Figure 3.7: The topology for the TCP example.

With 1024-byte packets, in the absence of additional rate limiting, the two flows get 0.805 Mbps and 0.740 Mbps, respectively. In comparison, if we limit their application-layer sending rates using our optimization framework and constrain the

burstiness of TCP by limiting the TCP sender buffer to 2 packets, the two flows get 1.066 Mbps and 1.064 Mbps, respectively, which translates to 37.9% increase in total throughput. With 512-byte packets, rate limiting results in 20.8% increase in total throughput. This example demonstrates that TCP is unable to appropriately set its rate to where it can maximize throughput. This is likely because TCP's aggressive bandwidth probing makes the flows stabilize at a loss rate higher than the loss rate under maximum throughput [8].

Mixed traffic We now show that similar problems occur when both TCP and UDP traffics exist. Consider the topologies in Figure 3.5. They are similar to those in Figure 3.1 except that instead of loss we create the bottleneck by sending traffic on the link X-Y. This "background" traffic is a 1.37 Mbps CBR source. In the bad-good case, S and R can carrier sense X and Y, and in the good-bad case, R and D can carrier sense X and Y. Simulations over this topology reveal effects similar to those with UDP. The throughput of the TCP flow from S to D is 1.25 Mbps for the bad-good case but only 0.27 Mbps for the good-bad case. The background flow obtains similar throughput in both cases. We also find a similar congestion collapse vulnerability in the good-bad case as the maximum allowed rate of the TCP flow (controlled using receiver window) increases.

Thus, TCP is unable to appropriately set its rate to where it can maximize throughput, even though we use a single, long-running TCP flow in this experiment. Since BSs will typically aggregate traffic from multiple users, they will likely relay multiple TCP flows of variable lengths. Such traffic will be even less responsive.

TCP's response to congestion does not prevent the anomalous behavior be-

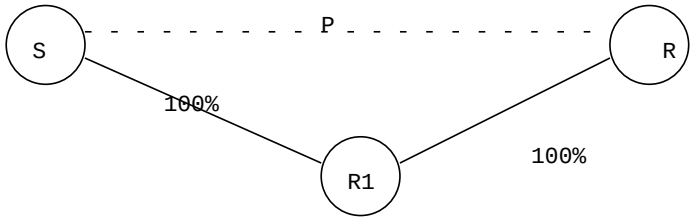


Figure 3.8: A three-node topology in which the two-hop is reliable and the one-hop path is lossy.

cause it reacts to losses that it is able to observe. But the retransmission mechanism of 802.11 is able to hide many layer 2 losses – up to 6 per packet in our configuration which is similar to the default in many wireless NICs. Thus, TCP does not react even though medium resources are being wastefully utilized. It is to highlight this effect that we use cross-traffic to create bottlenecks. (Cross traffic creates capacity differential between the two links that stems from reduced air time rather than a heavy loss rate, which reduces the number of losses exposed to TCP.)

3.2 Inability to Differentiate between Routing Options

We now demonstrate how existing routing protocols fail to find good paths. Because they do not properly account for interference, their quality metric does not always reflect which paths are better.

3.2.1 Scenario 1: Triangle Topology

As our first scenario, consider the simple, triangular topology shown in Figure 3.8. There are two paths between the source and destination. Both links on the two-hop path are reliable but the one-hop path is lossy. The graph 3.9 plots the

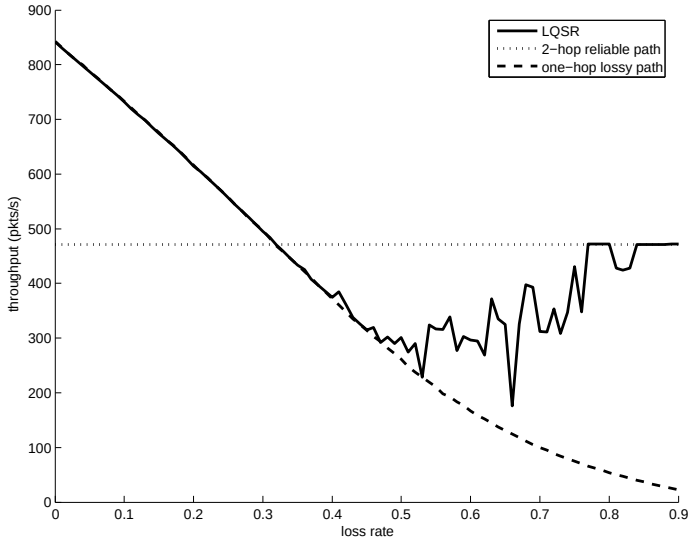


Figure 3.9: Throughputs of the one-hop path, the two-hop path and ETX routing as a function of loss rate.

throughput of this topology as we vary the loss rate on the one-hop path. It also plots the throughput of the two paths. The source is sending as fast as possible. We see that there is a significant range of loss rate in which the throughput of ETX is lower than that of the best available path. In the region where the ETX curve hugs the one-hop curve, ETX is consistently picking the one-hop path even when it is worse. In the region where the ETX curve is between the two other curves, the route is flapping between the two paths.

The reason for flapping is ETX's inability to decide which path is better. Once ETX starts using a path, it begins to appear worse. ETX then switches to the other path, and the cycle repeats itself.

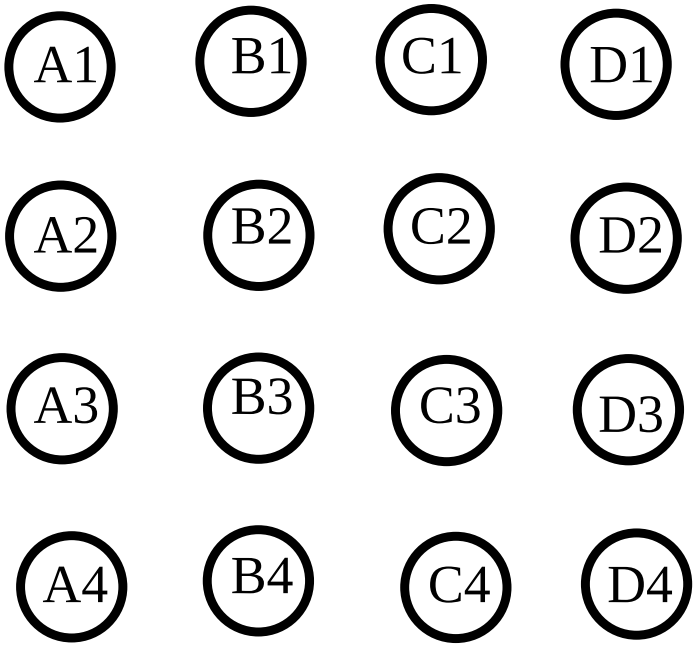


Figure 3.10: Example topologies where current protocols make poor routing path choices. Left: A grid. Right: A chain with interfering senders.

3.2.2 Scenario 2: Grid Topology

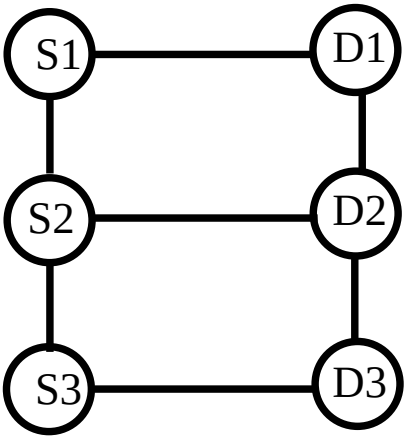
As our second example, consider the topology on the left in Figure 3.10. In this topology, adjacent nodes interfere with each other. There are two flows, one from *A*2 to *D*2 and another from *A*3 to *D*3. ETX picks the shortest paths *A*2-*B*2-*C*2-*D*2 and *A*3-*B*3-*C*3-*D*3, for a total throughput of 1.3 Mbps equally divided among the two flows. But we find that a better routing pattern is using a slightly longer path for the top flow, *A*2-*A*1-*B*1-*C*1-*D*1-*D*2, while keeping the same shortest path for the bottom flow. The total throughput then is 1.9 Mbps which represents an improvement of 46%. The flows improve individually as well, by 30% and 62%, respectively.

The reason ETX is not able to find the better routing pattern is because its link quality measurements only consider loss. In this case the interferers are close enough to coordinate their transmissions, and interference that stops a node from sending is not reflected in the metric. Similar limitation also exists in MIC and iAWARE routing metrics. To find high-throughput paths, the routing protocol must capture not only receiver-side interference that causes loss but also sender-side interference that stops nodes from transmitting.

3.3 Unfairness and Starvation

Another category of shortcoming of current routing protocols that we discuss is their unfairness. That routing over multi-hop wireless networks can be unfair towards flows that traverse longer distances is already well-known [11, 38], and we do not repeat those observations here. We show another form of unfairness which is not due to the effects of multi-hop.

Consider the topology in Figure 3.11. Each source S_i is sending traffic to its respective destination D_i. The default simulated throughputs of the three flows when they are sending as fast as possible is shown in the top table. The middle flow is starved because $S2$ is exposed to interference from both other sources. In such cases, rate-limiting sources in advantageous positions may be desirable. The bottom table shows the throughputs when source is rate limited to half of maximum sending rate. As this example shows, boosting fairness may sometimes come at the cost of total throughput. While the middle flow significantly improves, the total throughput decreases because the middle flow subtracts throughput from both



*S*1-*D*1	3.55 Mbps
*S*2-*D*2	0.04 Mbps
*S*3-*D*3	3.55 Mbps
Total	7.14 Mbps

*S*1-*D*1	1.96 Mbps
*S*2-*D*2	2.22 Mbps
*S*3-*D*3	2.07 Mbps
Total	6.18 Mbps

Figure 3.11: An example topology that exhibits starvation. Top table: Default throughputs. Bottom table: Throughputs when the top and bottom flows are rate limited.

sources. Nevertheless, we believe that it is important for routing protocols to consider both throughput and fairness concerns and at the very least address complete starvation.

3.4 Exponential Backoff

A node's contention window increases exponentially when it observes some packets loss. The effect of backoff can be significant when link loss rate is high, especially when the size of a packet is small and the data rate is high.

3.4.1 Short Lossy Paths vs. Long Reliable Paths

ETX uses the total number of transmissions to unify the goals of both favoring short paths and favoring reliable paths. An interesting question is whether two paths with the same ETX yield similar performance, even though one is long

(in terms of hop count) and reliable and the other is short (in terms of hop count) and lossy. That is, are all transmission counts equal?

Figure 3.12: Link's throughput vs. loss rate

We compare the two network paths shown in Figure 3.12, where one path has one hop with delivery rate of p, and the other path consists of $1/p$ hops with 100% delivery rate on each hop and all links on the path interfering with each other. Since hop count is an integer, we only use the value of p when $1/p$ is an integer. The ETX metrics of the two paths are the same. Figure 3.13 compares the throughput of short and long paths. As we can see, even though both paths have the same ETX, their throughput is quite different. The reliable and long paths significantly out-perform those shorter and lossy paths.

There are two main reasons for long and reliable paths to out-perform short and lossy paths.

First, when all links on both short and long paths interfere, long reliable paths yield higher throughput than short lossy paths due to the effect of backoff. More specifically, Figure 3.14 shows throughput over a one-hop path with loss rate varying from 0 to 0.9. As we can see, the throughput decreases much faster than increase in loss rate. For example, when link loss rate is 50%, its throughput is only 261 packets/second, which is only 0.31 of the throughput over a reliable link (i.e., 842 packets/second over a reliable one-hop path). Such a large decrease in throughput is because when loss rate is 50%, the time to successfully transmit the

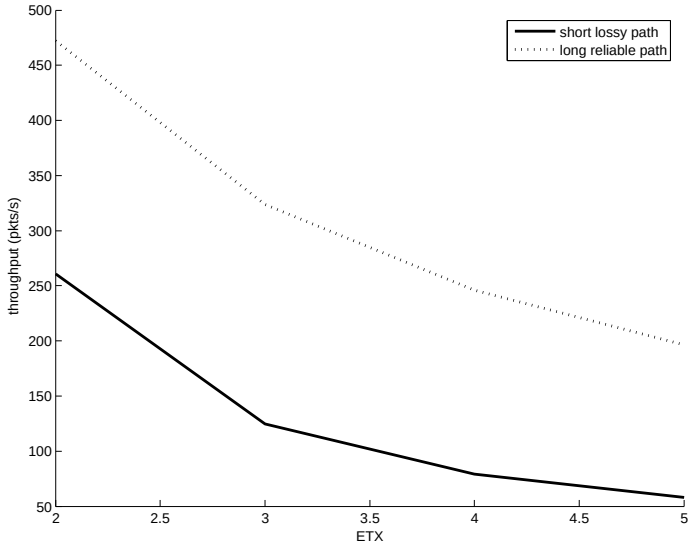


Figure 3.13: Throughput of short and long paths.

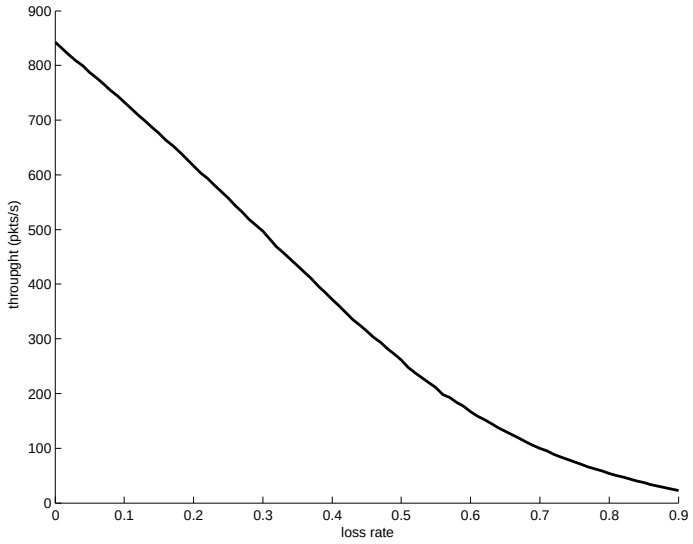


Figure 3.14: Throughput over a one-hop link with varying loss rate.

packet more than doubles as a result of exponential backoff. In comparison, transmitting over two reliable hops yields 471 packets/second, which is 80% more than the one-hop path with the same ETX. Moreover, as shown in Figure 3.13, the dif-

ference further increases as ETX values increase. For example, the throughput of a 4-hops path is 245 pkts/sec, which is 210% more than the one-hop path with the same ETX.

3.5 Inaccurate Estimation of Path Quality

An important requirement for a routing protocol is to facilitate the selection of good routing paths. In this section, we show how current wireless routing protocols are ineffective at this function because their measures of link quality may not reflect actual quality. We first show problems with basic link quality measurements and then with path quality measurements. We use ETX [4] as the representative of current protocols. Other protocols [5, 40, 32] also suffer from the pathologies described below, because they are based on ETX.

Measured ETX values may not reflect actual link quality that data traffic will experience when using that link. We demonstrate the problem with a concrete example. Figure 3.15 shows that the ETX values of all links in a 4x4 grid in absence of any traffic. Every link has good quality, with an ETX value close to 1. Figure 3.16 shows the snapshot of ETX values after a UDP flow is introduced from node 2 to 3. ETX values of links close to Link 2-3 increase because the probes on these links collide with data traffic on Link 2-3.

Do these ETX values reflect link quality that data traffic would experience?

To answer this question, we focus on Links 3-4, 4-8, 6-7, and 7-8. We retain the flow on Link 2-3 and inject traffic with a varying rate on one of the other links. Figure 3.18 shows the throughput of the data traffic. Comparing with

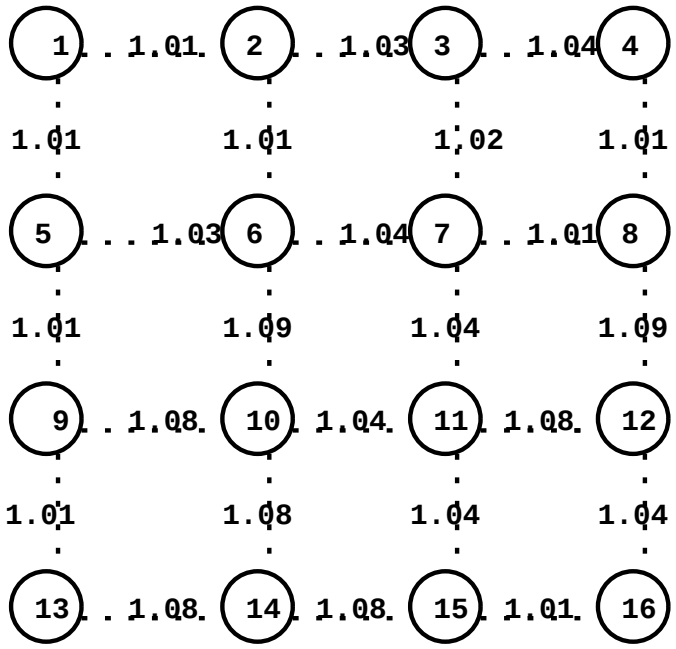


Figure 3.15: Measured ETX values under no traffic

Figure 3.16, we observe that the measured ETX values are poor indicator of the actual performance experienced by data traffic. For example, the ETX value of Link 3-4 is 16.67, which is much higher than that of Link 6-7; however the throughput of the two links are similar across all sending rates. Similarly, even though Links 7-8 and 4-8 have higher ETX values than Link 6-7, they have similar or higher throughput than Link 6-7.

The discrepancy between measured ETX and actual traffic performance arises from two factors. First, the ETX metric is determined by packet loss rates at receivers, so it only captures receiver-side interference but fails to capture sender-side interference that stops nodes from transmitting. To find high-throughput paths,

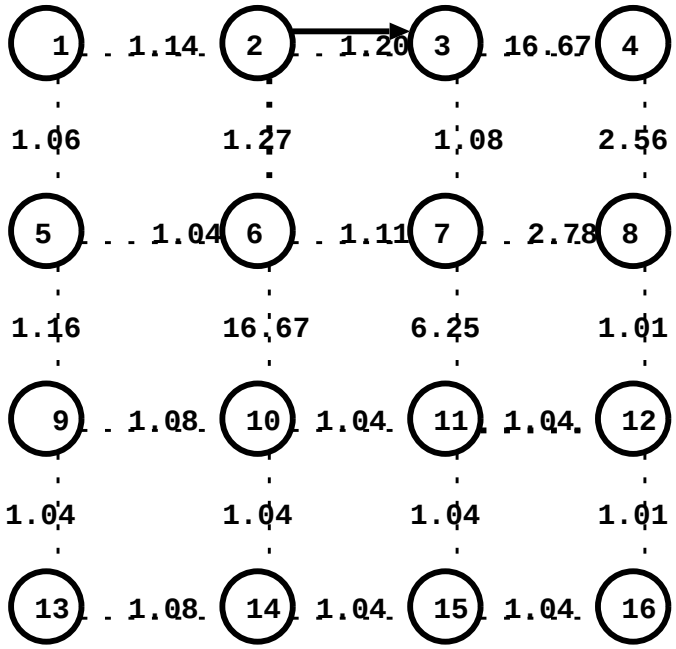


Figure 3.16: Measured ETX values under one UDP flow from node 2 to 3.

the routing protocol must capture both receiver-side and sender-side interference.

Second, the characteristics of probing traffic and data traffic can be quite different in terms of, for instance, volume, packet sizes and generation pattern, which makes the two observe different loss rates. For example, Figure 3.17 shows that as Link 7-8 carries more traffic, its loss rate decreases due to decreasing competing background traffic. The loss rate of data traffic can be higher or lower than that of probe traffic depending on volume. Different packet sizes and generation pattern of DATA/ACK and probe packets also contribute to the discrepancy. For example, Link 3-4 has ETX value of 16.67 in Figure 3.16 because the probe traffic on Link 4-3 collides heavily with data traffic on Link 2-3 due to the hidden terminal prob-

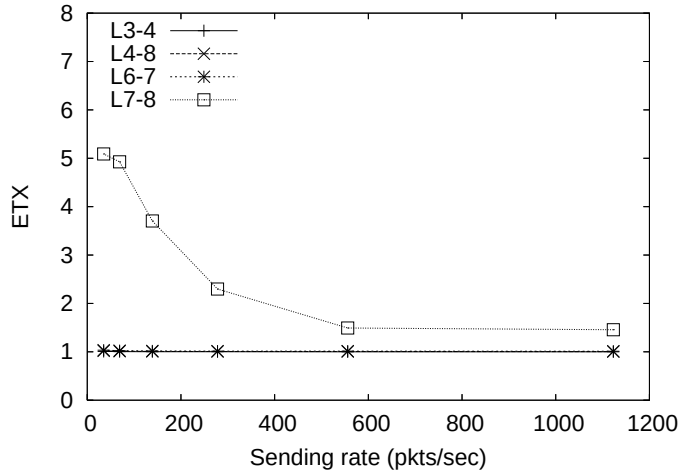


Figure 3.17: Measured ETX of data traffic under two UDP flows (one on Link 2-3 and one on the link in the legends.

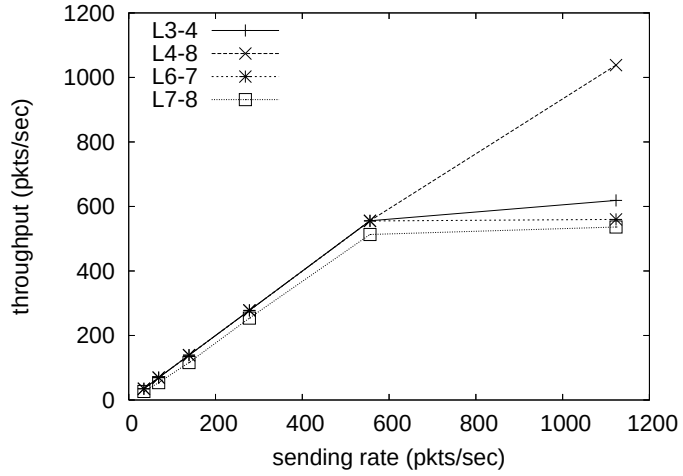


Figure 3.18: Throughput of a link when sending data traffic over it while keeping the same UDP flow from node 2 to 3 as competing traffic.

lem. However, as shown in Figure 3.17, the data traffic on Link 3-4 has low ETX, because ACKs on Link 4-3 seldom collide with data traffic on Link 2-3. This col-

lision rate is low because the ACKs have a smaller packet size and are generated immediately after the DATA packets on Link 3-4, during which time Node 2 often defers to them based on the NAV reservation in the DATA packets.

3.6 Discussion

We showed two problems that stem from not providing rate feedback to traffic sources in the face of wireless interference. The first was vulnerability of the network to congestion collapse in which increasing the load beyond a threshold may lead to a sharp decline in network throughput. The second was starvation of certain flows. These problems arise in range of scenarios, and rate-limiting traffic sources helps solve both. We also showed a third problem that stems from current protocols not accounting for sender-side interference. All problems we discuss in this chapter show it is necessary to take into account interference when manage wireless network. However, since the medium is open, an additional traffic can effect the capacity on other links. It is difficult to handle the interference accurately without a systematical approach.

Chapter 4

Modeling Interference

In this chapter, we develop an interference model to capture behaviors of 802.11 DCF in a real wireless network. We first revisit the background on 802.11. Then we list modeling requirements which need to be taken into account in our model. Finally, we describe the interference model.

4.1 Background on 802.11

The IEEE 802.11 standard [23] specifies two types of coordination functions for station to access the wireless medium: distributed coordination function (DCF) and point coordination function (PCF). Our work focus on DCF. DCF is based on carrier sense multiple access with collision avoidance (CSMA/CA), which is a random access scheme with carrier sense and collision avoidance through random backoff.The basic CSMA/CA mechanism is shown in Figure 4.1 A node determines the medium to be idle when the total energy received at a node is less than the CCA(clear-channel assessment) threshold. It starts transmission using the following rules.

If the medium is sensed idle for at least the duration of DIFS, a node starts transmission immediately. If the medium is busy, nodes have to wait for the duration

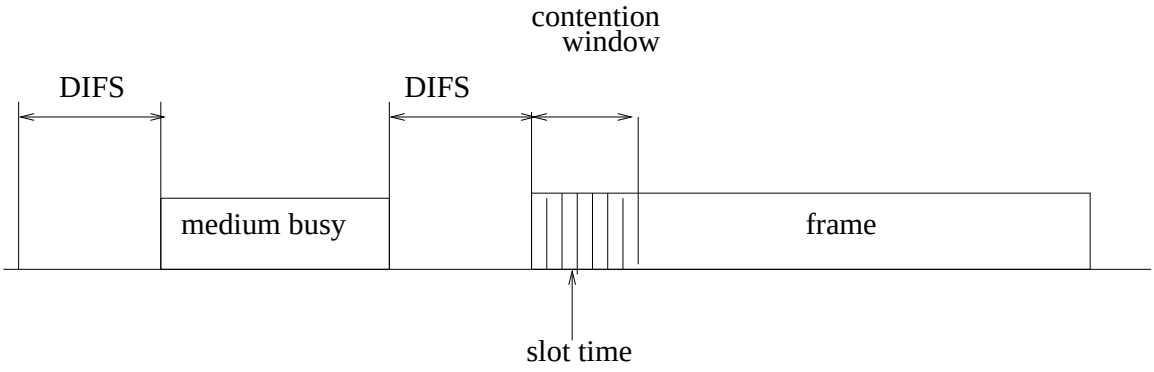


Figure 4.1: IEEE 802.11 DCF

of DIFS, entering a contention phase afterwards. Each node now chooses a random backoff time within a contention window and additionally delays medium access for this random amount of time. As soon as a node senses the channel is busy, it freezes its counter. The node has to wait until the medium is ideal again for at least DIFS. When the randomized additional waiting time for a node is over and the medium is still idle, the node can access the medium immediately. The additional waiting time is measured in multiples of slots. The number of slots is a random number uniformly chosen between $[0, CW]$, where CW is the contention window. In the case of broadcast, CW is always the minimum contention window, CW_{min}. In the case of unicast, if the receiver successfully receives the packet, it waits for a short interframe spacing time(SIFS) and then transmits an ACK frame. If the sender does not receive an ACK, it doubles its contention window to reduce its access rate. When the contention window reaches its maximum value, it stays at that value until a transmission succeeds, in which case the contention window is reset to CW_{min}.

4.2 Modeling Requirements

Many existing models make simplifying assumptions about signal propagation [12], traffic [10, 31, 9, 27], topology [2, 17, 9, 10], or the MAC layer [13]. These assumptions often do not hold for real networks [15]. Our model satisfies the following requirements:

- Instead of assuming saturated traffic demands, our model handles heterogeneous traffic demands. It can handle both TCP and UDP traffic.
- Instead of assuming binary loss and symmetric communication, our model considers the reality that loss rates on the two directions of a link can be quite different.
- Instead of assuming sender side symmetric interference, our model handles asymmetric interference. That is to say, even two nodes can carrier sense each other, they may have different defer probability when both of them are active.
- Our model can handle the impact of the hidden terminal problem.
- Our mode can handle both broadcast and unicast traffic.

Section 4.2.1 shows our measurement study on a wireless network. Our study shows that asymmetric loss and interference are quite common in a wireless network. It is necessary to capture such network characteristics if we want to have an accurate model. Section 4.2.2 describes an example of the hidden terminal problem.

4.2.1 Measurement Study on a Wireless Testbed

In order to capture characteristics of a real wireless network, such as link loss and interference, we conduct measurement study on a wireless testbed. The testbed we use is a controlled testbed which includes 19 wireless nodes located inside an office building and a controller. Each wireless node runs Linux and is equipped with a NetGear WAG511 NIC. We run 802.11a with a fixed bit rate of 6 Mbps. We are not aware of other 802.11a users in our building. Each node also equipped with an Ethernet connection. The controller uses these Ethernet connections to send demands and collect measurement results from the testbed. Figure 4.2 shows the structure of our testbed. The controller communicates with other mesh nodes through wireline communication. It uses ssh communication to send commands and receives measurement results from these mesh nodes.

We conduct the following measurement to study the interference and link inherent loss. We use the approach outlined by Padhye *et al.* [24] due to its simplicity. In our measurement, we use probing packets with payload size 1024 bytes. We let one node, say A, floods using broadcast packets for 1 minute. All other nodes listen to the transmission and record packets they can receive. Let $S1_A$ denote A broadcast rate when it broadcasts alone. Let $R1_{AC}$ denote C receiving rate when A broadcasts alone. Then we let two nodes, say A and B, flood simultaneously for 1 minute. All other nodes listen to the transmission and record packets they can receive. $S2_A^{AB}$ denotes A's broadcast rate when A and B are simultaneously sending. Let $R2_C^{AB}$ denotes receiving rate of C from A as both A and B are active simultaneously. After we get $S1$ $S2$ $R1$ $R2$, we can computer link inherent loss rate and

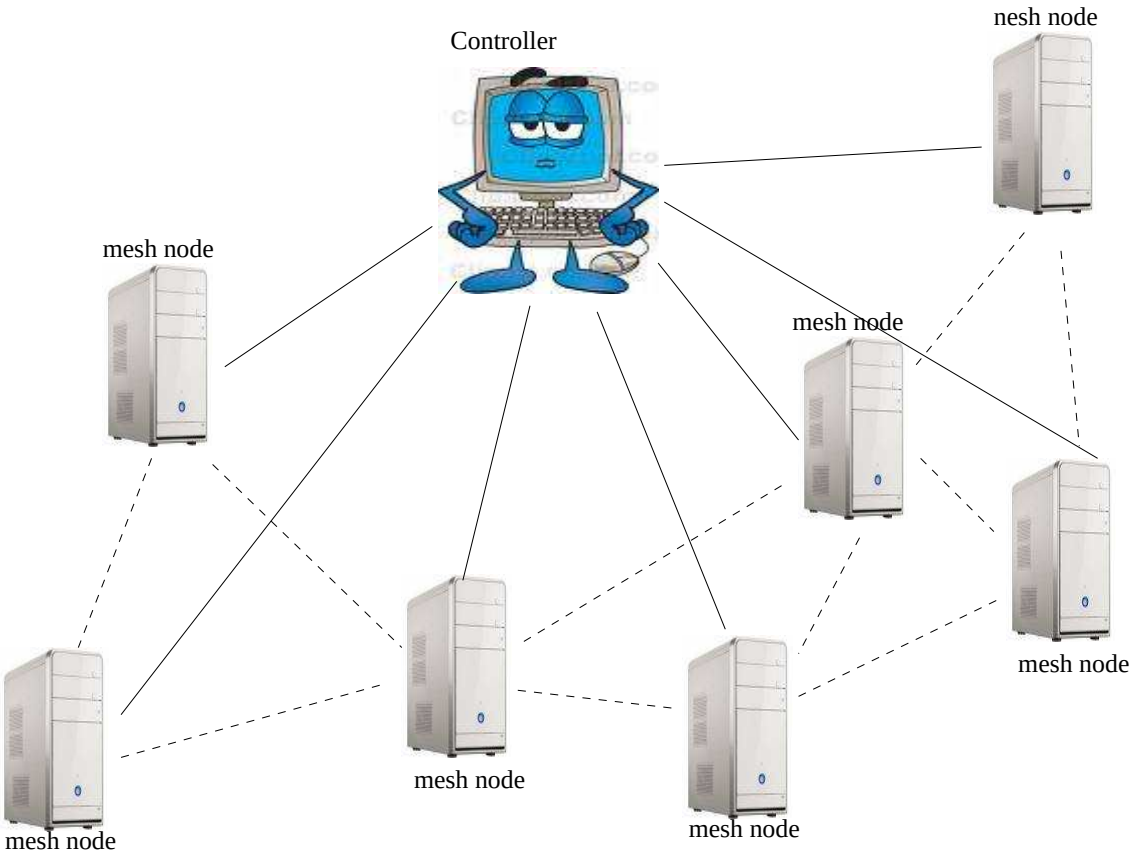


Figure 4.2: A controlled testbed

estimate interference.

Figure 4.3 shows CDF of link inherent loss rate in our testbed. We prune links exceeding 90% loss rates and finally there are 80 links. Around 65% links have almost zero loss rate. We compare two direction loss rates of each link. Figure 4.4 shows loss rate difference of two directions. In our testbed, less than 50% of links have symmetric loss rates. More than 40% of links have larger than 10% loss rate difference on two directions.

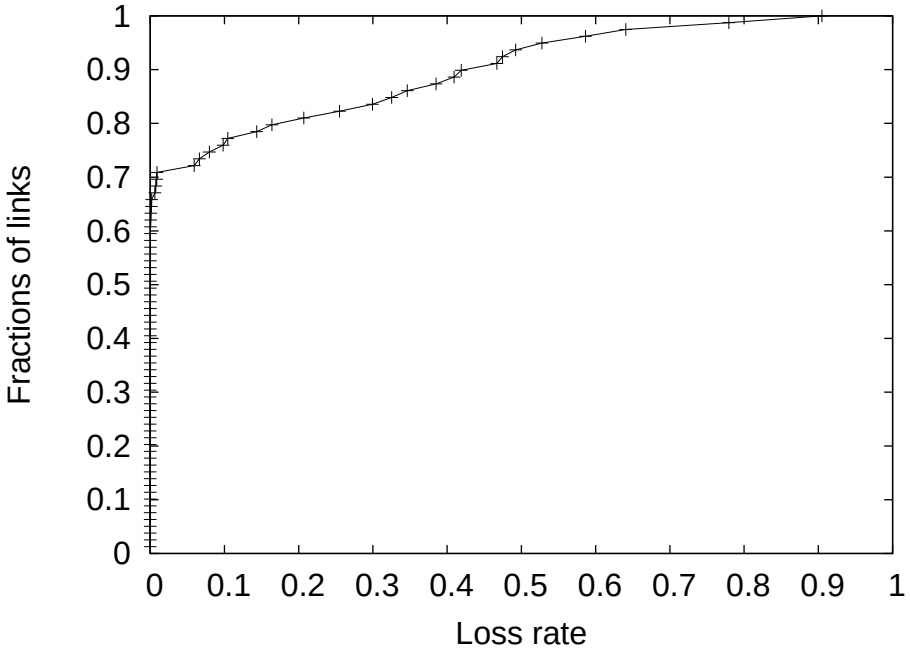


Figure 4.3: CDF of link loss rate in our testbed.

In our measurement, we find out that asymmetric interference is also common in our testbed. Table 4.1 shows the active probability of one node when another node is active simultaneously. Each entry $A(i, j)$ in the table is computed as following

$$A(i, j) = \frac{S2(i, j)}{S1(i)} \tag{4.1}$$

where $S2(i, j)$ is the sending rate of node i when both i and j are active, and $S1(i)$ is the sending rate of i when only i is active. $A(i, j)$ shows the probability of i to take the channel to send packets when interfered by node j

From this table, we find out that 55% of node pairs have symmetric interference, while more than 45% node pairs have asymmetric interference.

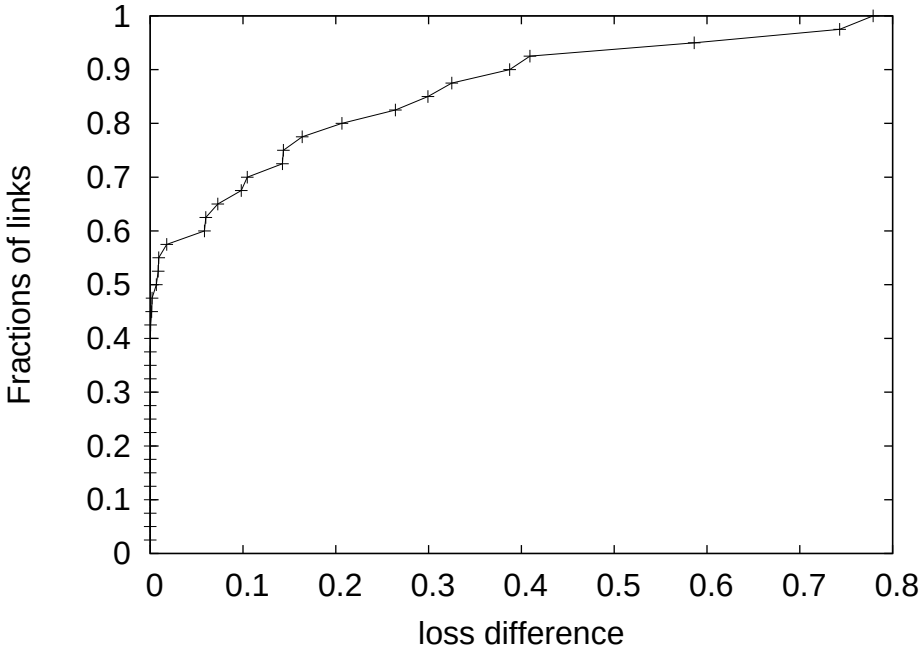


Figure 4.4: Two direction loss rate difference

4.2.2 The Hidden Terminal Problem

In wireless networks, interference is location based. Thus the hidden terminal problem may happen frequently [6]. In order to capture behaviors of hidden terminal, our model needs to capture asynchronous loss. Asynchronous loss happens when two links' transmissions are overlap; at least one link's transmission interfere the transmission of the other link; and two senders cannot carrier sense each other.

Figure 4.5 shows a scenario with four nodes. There are three radio ranges : transmission range (R_{tx}), carrier sensing range (R_{cs}) and interference range (R_i). Transmission Range(R_{tx}) represents the range within which a packet is successfully

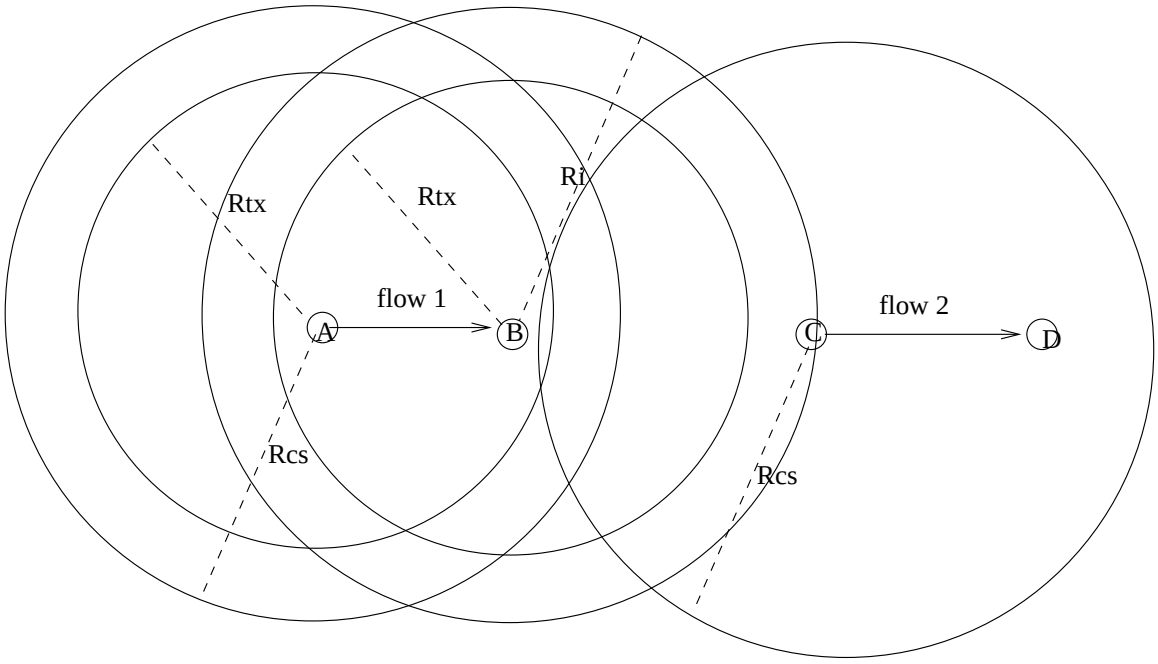


Figure 4.5: A hidden terminal case

received if there is no interference from other radios. Transmission range is mainly determined by transmission power and radio propagation properties, such as attenuation. Carrier Sensing Range (R_{cs}) is the range within which a transmitter triggers carrier sense detection. This is usually determined by the antenna sensitivity. In IEEE 8021.11 MAC, a transmitter only starts a transmission when it senses the medium is free, which means the energy it senses is below carrier sense threshold. Interference Range (R_i) is the range within which stations in receiving mode will be "interfered with" by an unrelated transmitter and thus suffer a loss.

The hidden terminal problem in this network happens when there are two flows AB and CD. As node A cannot carrier sense node C, node C is a hidden

node. C's transmissions can cause packets corrupted at receiver B. Therefore, the hidden terminal problem happens. Although in this scenario, RTS/CTS can avoid data received at node B to be corrupted by node C, the RTS sent from node A almost fail due to the collision caused by node C. Thus if flow CD saturates the link *CD*, the flow AB gets starved as no RTS sent from A can be successfully received by B. We have to limit traffic from C to D if we want the traffic to go from A to B. This scenario shows an ideal case. However, in reality, all these ranges may not be circular. They are not even contiguous sometime. In our model, we infer interference and condition loss rates caused by hidden nodes through interference measurement.

4.3 Basic Interference Model

Based on our measurement study, it shows that in order to model a real wireless networks, we not only need to model standard 80211 behavior, but also need to take into account asymmetric interference, asymmetric link loss rate and heterogeneous demand. The following part of the chapter shows how we model a wireless network.

We first develop a basic model of 802.11 DCF for the base case in which all flows are one-hop UDP flows and RTS/CTS is disabled. We then extend the model to support RTS/CTS, multi-hop flows, and different transport protocols in Section 4.4.

4.3.1 Assumptions

Our model makes two key assumptions:

*A*1. It assumes pairwise interference, *i.e.,* the interference relationship between two links is independent of activities on other links. Previous works show that pairwise interference is good approximation in real networks [24, 21].

*A*2. It assumes that different types of loss (*e.g.,* collision loss and inherent wireless medium loss) are independent.

While these assumptions do not always hold in practice, they are a reasonable approximation to the reality. Under these assumptions, we do not need to model intricate interactions among different links, *e.g.,* links *A* and *B* interfere only when links *C* and *D* are active. As a result, our model becomes significantly simplified — it has $O(n^2)$ complexity and only $O(n)$ constraints, where n is the number of active links. In our analysis, we use the normalized system throughput, defined as the fraction of time the channel is used to successfully transmit payload bits. We are able to express the normalized throughput as the ratio

$$normalized\ throughput = \frac{E[payload\ information\ transmitted\ in\ a\ slot\ time]}{E[length\ of\ a\ slot\ time]} \tag{4.2}$$

In Section 6.2, we use simulations and testbed experiments to show that our model is quite accurate despite these simplifications.

4.3.2 Constraints

we follow Bianchi's approach [2] to divide time into *variable-length slots (VLS)* for each link.

- When the link senses a clear channel and either has no data to send or its backoff counter has not yet reached 0, the current VLS lasts for a regular slot time T_{slot}.
- When the link senses a clear channel, has data to send, and its backoff counter is 0, it sends a packet and the current VLS lasts for the entire packet transmission.
- When the link senses a busy channel, the current VLS lasts until the channel is clear for a DIFS duration.

Our model consists of four types of constraints that capture the inter-dependency between throughput, transmission probability, packet loss rate, and VLS duration of different links. We describe these constraints below. Table 4.2 summarizes the notations, where constants are in upper case and variables are in lower case. To ensure consistency, we use slot time T_{slot} as the common unit for the calculation of time in our model.

Throughput constraint The throughput constraint relates throughput to transmission probability, packet loss rate, and VLS duration. Let τ_i be the probability for Link i to start a packet transmission during a VLS. Let p_i be the loss probability for such a packet transmission. Let μ_i be the expected duration of a VLS at Link i.

Let EP_i be the expected payload transmission time at Link i. Then, the throughput for Link i, denoted by g_i, is simply the fraction of time that it spends on successful payload transmissions:

$$g_i = \frac{EP_i \times \tau_i \times (1 - p_i)}{\mu_i} \tag{4.3}$$

VLS duration constraint The VLS duration constraint relates the expected VLS duration μ_i to transmission probability τ_j:

$$\mu_i = T_{\text{slot}} + \sum_j \left[(W_{ij} - T_{\text{slot}}) \times \tau_j \right] \tag{4.4}$$

where W_{ij} ($j \neq i$) is the expected amount of time for Link i to wait due to carrier-sense for Link j to complete a transmission, and W_{ii} is the expected amount of time for Link i to complete a transmission.

We estimate W_{ij} and W_{ii} as follows. Let L_j^{dat} be the inherent DATA loss rate on Link j. Let D_{ij}^{src} and D_{ij}^{dst} be the probabilities for Link i to carrier sense Link j's source and destination, respectively. Let T_j^{dat} be the expected duration of DATA transmission on Link j, which consists of a DIFS duration, a MAC preamble duration, the transmission time for the payload and packet headers. Let T_j^{ack} be the expected duration of ACK transmission on Link j, which consists of a SIFS duration, a MAC preamble duration, and the transmission time for an ACK. We then estimate W_{ij} and W_{ii} as:

$$\begin{aligned} W_{ij} &= D_{ij}^{\text{src}} \times T_j^{\text{dat}} + D_{ij}^{\text{dst}} \times T_j^{\text{ack}} \times (1 - L_j^{\text{dat}}) \\ W_{ii} &= T_i^{\text{dat}} + T_i^{\text{ack}} \times (1 - L_i^{\text{dat}}) \end{aligned}$$

We have made two simplifications above. We ignore the effect of collision loss on VLS duration and use only the inherent DATA loss rate L_j^{dat} to estimate the probability for a DATA transmission to succeed. This simplification turns W_{ij} and W_{ii} into constants instead of variables at the expense of slightly overestimating the expected VLS duration. We also ignore the effect of NAV on W_{ij}, *i.e.*, we assume that Link i waits for Link j's ACK only if it is transmitted. In reality, if Link i successfully receives Link j's DATA, it would wait even if no ACK is transmitted because of the NAV value embedded in Link j's DATA. The latter simplification may result in slight underestimation of the expected VLS duration, but the effect is small because ACK is typically much shorter than DATA.

Loss rate constraint The loss rate constraint relates packet loss rate to transmission probability. To compute packet loss rate p_i, we model both inherent medium loss and collision loss. Following [27], we further distinguish between two types of collision loss: (i) synchronous loss that occurs when the two senders can carrier sense each other; and (ii) asynchronous loss that occurs when at least one sender cannot carrier sense the other.

Assuming independence among different types of loss caused by different links, the packet success probability of Link i is

$$1 - p_i = (1 - L_i^{\text{dat}}) \times (1 - L_i^{\text{ack}}) \times \prod_{j \neq i} \left[(1 - \ell_{ij}^{\text{sync}}) \times (1 - \ell_{ij}^{\text{asyn}}) \right]$$

where L_i^{dat} and L_i^{ack} are the inherent loss rate of DATA and ACK on Link i; ℓ_{ij}^{sync} and ℓ_{ij}^{asyn} are synchronous and asynchronous collision loss on Link i caused by Link j, which can be modeled as follows.

- The synchronous collision loss rate is given by $\ell_{ij}^{\text{sync}} = S_{ij}\tau_j$, where τ_j captures the probability for Link j to start transmitting at the same time as Link i, and S_{ij} is the probability for a packet on Link i to get lost due to collision with a packet on Link j conditioned on the fact that the two packet transmissions start at the same time. Note that a packet is lost when either its DATA or ACK is lost. So S_{ij} combines the conditional loss rates of DATA and ACK.

- The asynchronous collision loss rate is given by $\ell_{ij}^{\text{asyn}} = 1-(1-\theta_j)^{A_{ij}}$, where $\theta_j \stackrel{\triangle}{=} \frac{\tau_j}{\mu_j}$ is the probability for Link j to start transmitting at a random time instant. It is obtained by normalizing τ_j by the expected VLS duration μ_j. A_{ij} is the asynchronous collision loss exponent defined as

$$A_{ij} \stackrel{\triangle}{=} \int_{-T_\mu}^{T_\mu} C_{ij}(x)dx,$$

where T_μ is the maximum duration of a packet transmission, $C_{ij}(x)$ is the conditional probability for a packet on Link i to get lost due to collision with a packet on Link j when the start times of the two packet transmissions differ by offset x. Thus, $C_{ij}(0) = S_{ij}$. To understand the intuition behind the definition of A_{ij}, imagine that we divide time into bins of fixed width Δx. For a given time bin at offset x, the probability for Link j to start a transmission in it is $\theta_j \Delta x$. Similar to the analysis of synchronous collision loss, the probability for Link j's packet to cause collision loss in Link i's packet at offset x is given by $C_{ij}(x)\theta_j\Delta x$. The probability for Link i's packet to succeed despite collision with Link j's packet can therefore be approximated as $1-C_{ij}(x)\theta_j\Delta x \approx (1-\theta_j)^{C_{ij}(x)\Delta x}$. Assuming independent collision loss for

different offsets, the total asynchronous collision loss probability for Link i can therefore be approximated by

$$1 - \prod_{x \in [-T_\mu, T_\mu]} (1-\theta_j)^{C_{ij}(x)\Delta x} = 1 - (1-\theta_j)^{\sum_{x \in [-T_\mu, T_\mu]} C_{ij}(x)\Delta x}$$

whose limit becomes $1-(1-\theta_j)^{\int_{-T_\mu}^{T_\mu} C_{ij}(x)dx} = 1-(1-\theta_j)^{A_{ij}}$ as Δx tends to 0.

Putting it all together, we can model packet loss rate p_i as a function of transmission probability τ_j and $\theta_j = \frac{\tau_j}{\mu_j}$:

$$\begin{aligned} p_i \;=\; 1 \;-\; & (1-L_i^{\text{dat}}) \times (1-L_i^{\text{ack}}) \times \\ & \prod_{j \neq i} \left[(1 - S_{ij}\tau_j) \times (1-\theta_j)^{A_{ij}} \right] \end{aligned} \tag{4.5}$$

Feasibility Constraint With 802.11 DCF, the transmission probability τ_i is feasible if and only if it is bounded by a function of the packet loss rate p_i. Specifically, we have [2, 27]

$$\tau_i \leq \frac{2}{2 + CW(p_i)}, \tag{4.6}$$

where $CW(p_i) = CW_{\min} + p_i \times (1 + CW_{\min}) \times \sum_{k=0}^{M-1} (2p_i)^k$ is the expected contention window size under packet loss rate p_i, $CW_{\min}$ is the minimum contention window size in slots. For 802.11a, $CW_{\min}$=15, $M = \log_2\left(\frac{CW_{\max}+1}{CW_{\min}+1}\right)$, and $CW_{\max}$=1023.

4.4 Extensions to the Basic Model

We now extend the basic model above to support RTS/CTS, multi-hop flows, and TCP traffic.

RTS/CTS To support RTS/CTS, we make two modifications. First, in the VLS constraint (Eq. 4.4), constants W_{ij} and W_{ii} are updated to account for the additional delay introduced by RTS and CTS. Second, the loss rate constraint (Eq. 4.5) is extended to incorporate the inherent RTS and CTS loss rates, L_i^{rts} and L_i^{cts}, and the additional collision losses involving RTS and CTS.

Multi-hop Flows Given routing information, we can convert multi-hop UDP flows into one-hop UDP flows. Specifically, let $\mathbf{x} = \langle x_d \rangle_{m\times 1}$ be the vector of end-to-end flow rates. Let $R = [R_{id}]_{n\times m}$ be the $n \times m$ routing matrix, where R_{id} is the fraction of Flow d that traverses Link i. Let $\mathbf{g} = \langle g_i \rangle_{n\times 1}$ be the vector of link loads. Then, we have

$$\mathbf{g} = R \cdot \mathbf{x} \tag{4.7}$$

Note that the conversion above applies only when the end-to-end flow rates are *feasible*. If the end-to-end flow rates are infeasible, a multi-hop flow may result in more traffic on hops near the origin, which cannot be carried forward by the subsequent hops. Restricting to only feasible flow rates is not a problem for model-driven optimization because we only need to consider feasible flow rate assignments.

TCP Traffic Finally, when TCP is used as the transport layer protocol, we also need to take into account the TCP acknowledgment traffic. To convert multi-hop TCP demands into one-hop link demands, we replace the routing matrix R in Eq. 4.7 with a new routing matrix R_{TCP} that combines the forward and reverse direction of TCP connections. Specifically, let R_{fwd} and R_{rev} be the routing matrix

for the forward and reverse direction of TCP connections, respectively. We define

$$R_{\text{TCP}} \triangleq R_{\text{fwd}} + \alpha \times R_{\text{rev}}, \tag{4.8}$$

where the coefficient α reflects the size and frequency of TCP acknowledgments. Assuming that TCP acknowledgments contain no payload, without TCP delayed acknowledgments, we simply set $\alpha = \frac{H}{H+EP}$, where H is the total size of IP and TCP headers, and EP is the expected payload size. With TCP delayed acknowledgments enabled, we set $\alpha = 0.5 \times \frac{H}{H+EP}$.

4.5 Model Initialization

To apply our model, we need to initialize the following constants through interference measurement $S1$, $S2$, $R1$, and $R2$.

- inherent loss rate of a link
- defer probability between a pair of senders
- conditional loss probability

Infer Raw Loss Rate of a Link We estimate the raw loss rate of a link e from node A to node B as following. Assume the sending rate of node A is S(A), the receiving rate of link e is $R(e)$ and L_e is the raw loss rate of this link. We have

$$L_e = 1 - \frac{R(e)}{S(A)} \tag{4.9}$$

Infer Defer Probability for a Pair of Senders Given a pair senders i and j, we need to estimate the following conditional probability D_{ij}.

$$D_{ij} = Pr\{i \textit{ is not active} | \ j \textit{ is active}\} \tag{4.10}$$

We call D_{ij} the probability that i defers to j. Let $S2(i,j)$ be the broadcast sending rate of node i when both i and j are broadcasting; τ be the probability of a node to broadcast a packet; T_{sbcast} be expected duration of a broadcasting transmission; EP be expected payload transmission time. The probability of a slot is idle observed by node i is when i is not active and j is not carrier sensed as active. So this probability is

$$(1-\tau)*(1-D_{ij}*\tau) \tag{4.11}$$

Then we have

$$\begin{aligned} S2(i,j) &= \frac{E[payload\ information\ transmitted\ in\ a\ slot\ time]}{E[length\ of\ a\ slot\ time]} \\ &= \frac{\tau * EP}{slot + (T_{sbcast} - slot)*(1-(1-\tau)*(1-D_{ij}*\tau))} \\ &\approx \frac{\tau * EP}{slot + (T_{sbcast} - slot)*(\tau + D_{ij}*\tau)} \end{aligned}$$

Then we can derive

$$D_{ij} = \frac{\frac{EP}{S2(i,j)} - \frac{slot}{\tau}}{T_{sbcast} - slot} - 1 \tag{4.12}$$

Infer Conditional Broadcast Loss Probability Assume $R2(e,j)$ as the broadcast receiving rate of link e when j is active. Define condLoss(e, j) as the following conditional loss probability

$$condLoss(e,j) = Pr\{\ a\ packet\ on\ e\ is\ lost|\ j\ is\ active\} \tag{4.13}$$

We have

$$R2(e,j) = S2(i,j) \times (1 - rawLoss(e)) * (1 - olap_{ij} * condLoss(e,j)) \tag{4.14}$$

Where $olap_{ij}$ is defined as

$$Prob\{a\ packet\ \ from\ i\ overlaps\ with\ a\ packet\ \ from\ j\}. \tag{4.15}$$

In order to derive condLoss(e,j) from this equation, we need to estimate $olap_{ij}$ first. To estimate $olap_{ij}$, we treat the traffic from j as the background traffic. We assume the background traffic as ON/OFF process with exponentially distributed ON and OFF periods. Let T_j denotes average duration that j transmits packets; T_{off} denote average duration that j is idle. We define a random variable T as the exponential variable with average T_{off}. T_j and T_{off} can be computed as following:

$$\begin{aligned} T_j &= S2(j,i) \times \frac{T_{bcast}}{EP} \\ T_{off} &= 1 - T_j \end{aligned}$$

The CDF of random variable T is defined as follows.

$$F(t) = P\{T \leq t\} = 1 - e^{\frac{-t}{T_{off}}}; \tag{4.16}$$

We consider the following four cases based on different combination of D_{ij} and D_{ji}.

- $D_{ij} = 1$ and $D_{ji} = 1$ (i can carrier sense j and j can carrier sense i)

$$olap_{ij} = TAU \tag{4.17}$$

- Case 2 $D_{ij} = 1$ and $D_{ji} = 0$ (i can carrier sense j while j cannot carrier sense i)

  The probability that the OFF period is longer then t is

  $$P\{T > t\} = 1 - F(t) = e^{\frac{-t}{T_{off}}}; \tag{4.18}$$

  As i can carrier sense j, i overlaps j's transmission happens when j's idle time is less than i's transmission time, which is T_{sbcast}. Then we have

  $$olap_{ij} = 1 - P\{T > T_{sbcast}\} = 1 - e^{\frac{-T_{sbcast}}{T_{off}}}; \tag{4.19}$$

- Case 3 $D_{ij} = 0$ and $D_{ji} = 1$ (i cannot carrier sense j while j can carrier sense i)

  As j can carrier sense i, the OFF period is always longer than i's transmission period. So the overlap probability is the probability that j is active. Therefore

  $$olap_{ij} = \frac{t_j}{t_j + e^{\frac{-T_{sbcast}}{T_{off}}} * (1 - t_j)}; \tag{4.20}$$

- Case 4 $D_{ij} = 0$ and $D_{ji} = 0$, (i and j cannot carrier sense each other)

  i's transmission does not overlap j's transmission only if j is not active and the OFF period is longer than T_{sbcast}.

  So the overlap probability is

  $$olap_{ij} = 1 - (1 - t_j) \times e^{\frac{-T_{bcast}}{T_{off}}} \tag{4.21}$$

After we get $olap_{ij}$, we can compute condLoss(e, j) as follows.

$$condLoss(e, j) = \frac{1 - \frac{R2(e,j)}{S2(i,j) \times (1 - rawLoss(e))}}{olap_{ij}} \tag{4.22}$$

Table 4.1: Asymmetric interference

1.00	0.55	0.77	0.55	1.00	1.00	1.00	1.00	0.55	0.49	1.00	1.00	1.00	1.00	0.61	1.00	1.00	1.00	1.00
0.56	1.00	0.54	0.56	1.00	0.78	0.48	1.00	0.58	0.56	1.00	1.00	1.00	0.45	0.77	1.00	0.44	1.00	1.00
0.83	0.57	1.00	0.95	1.00	0.56	0.56	1.00	1.00	0.43	1.00	1.00	1.00	1.00	0.56	1.00	0.45	1.00	1.00
0.56	0.56	0.87	1.00	1.00	1.00	0.35	1.00	0.51	0.56	1.00	1.00	1.00	1.00	0.68	1.00	0.56	1.00	1.00
1.00	1.00	1.00	1.00	1.00	0.44	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	0.92	0.56	1.00	0.85	1.00	0.46	0.56	1.00	0.91	0.56	1.00	0.56	1.00	0.56	1.00	0.52	1.00	1.00
0.76	0.63	0.56	0.75	1.00	0.64	1.00	1.00	1.00	0.56	1.00	1.00	1.00	1.00	0.56	1.00	0.56	1.00	1.00
0.71	0.44	0.45	0.44	0.43	0.56	0.49	1.00	1.00	1.00	0.56	0.55	0.56	1.00	0.56	1.00	0.48	0.55	1.00
0.56	0.53	1.00	0.61	1.00	0.99	1.00	1.00	1.00	0.65	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.56
0.62	0.56	0.67	0.56	1.00	0.46	0.56	1.00	0.53	1.00	1.00	1.00	1.00	0.82	0.70	1.00	0.56	1.00	1.00
1.00	0.56	0.40	0.93	0.41	0.56	0.53	0.54	1.00	1.00	1.00	0.60	0.98	1.00	0.56	1.00	1.00	1.00	1.00
1.00	1.00	0.52	1.00	1.00	0.41	0.75	0.56	1.00	1.00	0.51	1.00	0.58	0.55	0.66	0.55	1.00	0.56	1.00
0.95	0.20	0.48	0.78	0.05	0.57	0.58	0.57	1.00	1.00	0.41	0.52	1.00	1.00	0.56	1.00	0.55	0.55	1.00
0.36	0.65	1.00	0.04	1.00	1.00	0.11	1.00	1.00	0.97	1.00	0.57	1.00	1.00	0.72	0.56	1.00	0.95	1.00
1.00	0.32	0.55	1.00	1.00	0.56	0.55	0.56	1.00	0.40	0.55	0.44	0.56	1.00	1.00	1.00	1.00	0.10	1.00
1.00	0.61	0.06	0.22	1.00	1.00	0.20	1.00	1.00	1.00	1.00	0.55	1.00	0.56	0.67	1.00	1.00	1.00	1.00
0.02	0.67	0.66	0.56	1.00	0.59	0.56	0.64	1.00	0.56	1.00	1.00	0.58	1.00	0.02	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	0.69	1.00	0.56	1.00	1.00	1.00	0.56	0.56	1.00	0.92	0.99	1.00	1.00	1.00
0.54	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.56	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

EP_i	expected payload transmission time for link i (in slots)
T_i^s	expected duration of transmission step s for link i (in slots)
E_i^s	inherent loss rate for link i's transmission step s
S_{ij}^{st}	synchronized collision loss probability for link i's transmission step s caused by link j's transmission step t
A_{ij}^{st}	expected size of the collision region for j's transmission step t to cause asynchronous collision loss in i's transmission step s
D_{ij}^s	probability for i to sense j's transmission step s
W_{ij}	expected waiting time for link i when j is transmitting (because i can carrier sense j or hear j's CTS). We have $W_{ij} \approx \sum_s (D_{ij}^s * T_j^s * \prod_r^{s-1} (1 - E_j^r))$
τ_i	probability for link i to transmit in a random variable-length slot (VLS)
ℓ_{ij}^{st}	collision loss rate for i's transmission step s due to j's transmission step t
ℓ_i^s	collision loss rate of link i for transmission step s: $\ell_i^s = \sum_j \sum_t \ell_{ij}^{st}$
p_i	packet loss rate of link i. We have $p_i = 1 - (1 - \sum_s \ell_i^s) * \prod_s (1 - E_i^s)$
μ_i	expected VLS duration of link i. We have $\mu_i = 1 + \sum_j (W_{ij} * \tau_j)$
g_i	throughput of link i. We have $g_i = \frac{EP_i * \tau_i * (1 - p_i)}{\mu_i}$
θ_i	probability for i to start sending at a random time instant: $\theta_i = \frac{\tau_i}{\mu_i} = \frac{g_i}{EP_i * (1 - p_i)}$

Table 4.2: Model constants (upper case) and variables.

Chapter 5

Model-Driven Optimization

In this chapter, we apply our model to optimize wireless performance. Our overall optimization strategy is to compute sending rates for all flows based on their demands, the network topology, and the optimization objective. We first describe an algorithm to test whether a given flow rate assignment is achievable in Section 5.1. We then consider maximizing fairness in Section 5.2 and maximizing total throughput in Section 5.3. Table 4.2 shows constants that are used in our model.

5.1 Flow Throughput Feasibility Testing

Our goal is to test whether a given set of link throughput g_i's is achievable. The main challenge is that there is strong inter-dependency between the transmission probability and the loss rate of different links. The transmission probability of a Link i, τ_i, depends on the transmission probability of the other links, which in turn depends on τ_i. To address the inter-dependency, we use an iterative procedure to jointly estimate the transmission probabilities and loss rates. We initialize the collision loss and transmission probabilities at all links to be 0. We then iteratively update link transmission probabilities and loss rates based on the other links' transmission probabilities and loss rates derived in the previous iteration. Figure 5.1

▷ Input: a vector of link throughput $\langle g_i \rangle$; ▷ Output: whether $\langle g_i \rangle$ is feasible

1. **initialization**: $feasible = 0$, $\tau_i = 0$, $p_i = 0$ ($i = 1, 2, \ldots, n$)
   // iterative model evaluation (MaxIter = 20 by default)
2. **for** *iter* = 1 **to** *MaxIter*
3. $\quad \theta_i = \frac{g_i}{EP_i \times (1-p_i)} \quad i = 1, 2, \ldots, n$
4. $\quad \langle \tau_i \rangle = \mathsf{estimate\_tau\_from\_theta}(\langle \theta_i \rangle)$
5. $\quad \langle p_i \rangle = \mathsf{compute\_packet\_loss\_rates}(\langle \tau_i \rangle, \langle \theta_i \rangle)$ *// according to Eq. 4.5*
6. $\quad$ **if any** *i* whose ($\tau_i > \frac{2}{2+CW(p_i)}$)
7. $\quad\quad feasible = 0$; **break** *// early stop: infeasible*
8. $\quad$ **end if**
9. $\quad g'_i = \frac{\tau_i \times (1-p_i) \times EP_i}{T_{\text{slot}} + \sum_j [(W_{ij} - T_{\text{slot}}) \times \tau_j]}$
10. $\quad$ **if** ($\max_i \{|g_i - g'_i|\} < TOL$) *// convergence test (TOL = 0.01 by default)*
11. $\quad\quad feasible = 1$; **break** *// early stop: feasible*
12. $\quad$ **end if**
13. **end for**
14. **return** *feasible*

Figure 5.1: Link throughput feasibility testing.

outlines the algorithm.

To estimate $\langle \tau_i \rangle$ given $\langle \theta_i \rangle$ (Line 4 in Figure 5.1), we note that $\theta_i = \frac{\tau_i}{\mu_i} = \frac{\tau_i}{T_{\text{slot}} + \sum_j [(W_{ij} - T_{\text{slot}}) \times \tau_j]}$. Therefore, we can estimate $\langle \tau_i \rangle$ by solving the following system of linear equations

$$\left\{ T_{\text{slot}} + \sum_j \left[(W_{ij} - T_{\text{slot}}) \times \tau_j \right] \right\} \times \theta_i = \tau_i, \quad i = 1, 2, \ldots, n \tag{5.1}$$

The iterative procedure continues until the number of iterations reaches a threshold, or the throughput values no longer change significantly, or a feasibility constraint (Eq. 4.6) is violated. We bound the number of iterations to twenty, which works well in our experiments.

5.2 Fair Rate Allocation

Given the feasibility test for link throughput, we use it as a basic block for achieving weighted max-min fair rate allocation. This allocation takes routing and

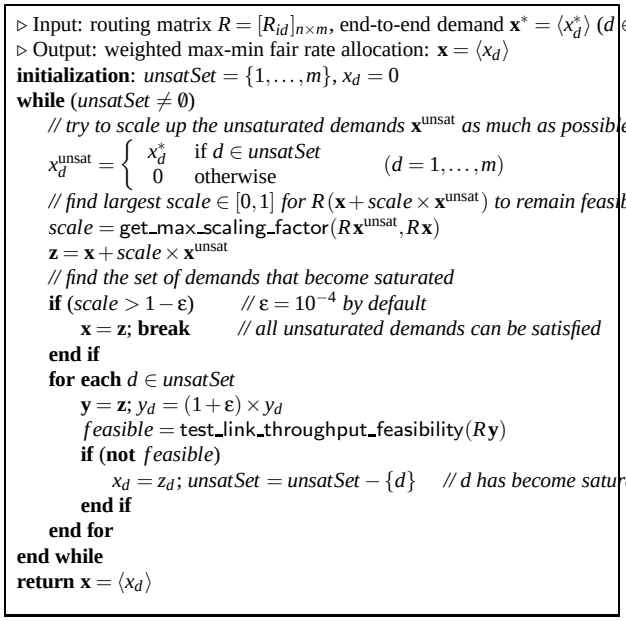


Figure 5.2: Algorithm for fair rate allocation

traffic demand matrices as input.

Figure 5.2 outlines the algorithm, which is effectively based on iterative water-filling. Let $\mathbf{x}^* = \langle x_d^* \rangle$ be the end-to-end demand. Let $R = [R_{id}]_{n\times m}$ be the routing matrix, where R_{id} is the fraction of Flow d that traverses Link i. The vector of link loads is given by $R \cdot \mathbf{x}$. Initially, the algorithm marks all demands as unsaturated. In each iteration, the algorithm tries to scale up all the unsaturated demands as much as possible until at least one unsaturated flow is saturated, *i.e.*, it cannot be scaled up further without violating a feasibility constraint. The maximum scaling factor $scale \in [0,1]$ is found efficiently through bisection search in the subroutine $\mathsf{get\_max\_scaling\_factor}(\mathbf{g}^{\text{unsat}}, \mathbf{g}^{\text{sat}})$ (Line 4 in Figure 5.2). The iteration continues to scale up the remaining unsaturated demands until all demands are saturated.

```
1.  initialization: x_d^(0) = 0, τ_i^(0) = 0, for ∀d∀i
2.  for k = 1 to KMAX
3.      let x^opt and τ^opt be the optimal solution to the linear program (LP_k)
4.      x^(k) = x^opt
5.      repeat    // ensure solution feasibility
6.          x^(k) = x^(k−1) + α × (x^(k) − x^(k−1))
7.          feasible = test_link_throughput_feasibility(R x^(k))
8.      until (feasible = true)
9.      x^(k) = 0.99 × x^(k)
10. end for
11. return x^(k)
```

Figure 5.3: Algorithm for maximizing total throughput.

5.3 Total Throughput Maximization

We optimize the network for maximum total throughput by formulating a non-linear optimization problem. This problem is solved by linearizing the non-linear constraints and solving a series of linear programs.

As before, let $\mathbf{x}^* = \langle x_d^* \rangle$ be the end-to-end demand and $R = [R_{id}]_{n \times m}$ be the routing matrix. Let R_i be the i-th row vector of R. The problem of maximizing total end-to-end throughput can be cast into the following non-linear optimization problem (NLP).

$$\begin{aligned} \text{maximize} \quad & \sum_d x_d \\ \text{subject to} \quad & \begin{cases} R_i \mathbf{x} \le F_i(\tau) & \forall i \\ G_i(\tau) \le 0 & \forall i \\ 0 \le x_d \le x_d^* & \forall d \\ 0 \le \tau_i \le 1 & \forall i \end{cases} \end{aligned} \tag{NLP}$$

where $F_i(\tau) = \frac{EP_i \times \tau_i \times (1-p_i)}{T_{\text{slot}} + \sum_j (W_{ij} - T_{\text{slot}}) \times \tau_j}$ and $G_i(\tau) = \tau_i - \frac{2}{2 + CW(p_i)}$. Therefore, constraints $R_i \mathbf{x} \le F_i(\tau)$ encode the linear relationship between end-to-end throughput $\mathbf{x}$ and link throughput; constraints $G_i(\tau) \le 0$ encode the feasibility constraint (Eq. 4.6).

We solve the NLP above through iterative linear programming, as shown in

Figure 5.3. In each iteration, we linearize the non-linear constraints in the NLP using their first-order approximation. Specifically, let $\mathbf{x}^{(k-1)}$ and $\tau^{(k-1)}$ be the estimate of $\mathbf{x}$ and τ in iteration $(k-1)$. Let $F_i^*(\tau)$ and $G_i^*(\tau)$ be the first-order approximations of $F_i(\tau)$ and $G_i(\tau)$, respectively. $F_i^*(\tau)$ and $G_i^*(\tau)$ are then:

$$F_i^*(\tau) = F_i(\tau^{(k-1)}) + \sum_j (\tau_j - \tau_j^{(k-1)}) \times \frac{\partial}{\partial \tau_j} F_i(\tau^{(k-1)}) \tag{5.2}$$

$$G_i^*(\tau) = G_i(\tau^{(k-1)}) + \sum_j (\tau_j - \tau_j^{(k-1)}) \times \frac{\partial}{\partial \tau_j} G_i(\tau^{(k-1)}) \tag{5.3}$$

How to compute all the partial derivatives are shown in Appendix 1.

Substituting $F(\tau)$ and $G(\tau)$ with $F^*(\tau)$ and $G^*(\tau)$ in (NLP), we obtain the following linear program:

$$\begin{array}{ll} \text{maximize} & \sum_d x_d \\ \text{subject to} & \begin{cases} R_i \mathbf{x} \le F_i^*(\tau) & \forall i \\ G_i^*(\tau) \le 0 & \forall i \\ 0 \le x_d \le x_d^* & \forall d \\ 0 \le \tau_i \le 1 & \forall i \end{cases} \end{array} \tag{LP$_k$}$$

We then derive $\mathbf{x}^{(k)}$ and $\tau^{(k)}$ by solving the linear program (LP$_k$). The optimal solution to (LP$_k$), however, cannot be directly used because the LP is only an approximation to the original NLP. The resulting solution may not satisfy the constraints in the original NLP. To ensure $\mathbf{x}^{(k)}$ satisfies NLP, we apply a simple line search to find a point on the line between $\mathbf{x}^{(k-1)}$ and $\mathbf{x}^{(k)}$ that is feasible. During the line search, the distance between $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k-1)}$ shrinks exponentially fast. Since we guarantee the feasibility of $\mathbf{x}^{(k-1)}$, we can quickly find a feasible solution. In our evaluation, we set the shrinkage ratio to $\alpha = 0.5$. Finally, to better deal with

numerical imprecision in our feasibility test, we scale down $\mathbf{x}^{(k)}$ by 1% at the end of each iteration (Line 9 in Figure 5.3).

Since our problem is NLP, we cannot guarantee a global optimal solution. To improve the quality of the final solution, we use multiple starting points. We always include an all-zero starting points (*i.e.,* all flows are inactive). To favor flows that are more likely to achieve higher throughput, we also add $(N_{\text{init}} - 1)$ starting points, each with only a single active flow. Specifically, for each $d = 1, \ldots, n$, we find the largest $x_d^{\text{init}} \leq x_d^*$ such that it is feasible for flow d to send at rate x_d^{init} while all other flows are inactive. This can be done efficiently using the subroutine get_max_scaling_factor (see Section 5.2). We then select the $(N_{\text{init}} - 1)$ flows with the largest x_d^{init}, reduce their rates by a constant factor (2 by default) so that they are not too close to the boundary of the feasible solution space, and include the resulted traffic assignments as our starting points. In our experiments, we use $N_{\text{init}} = 4$ starting points. However, our experience suggests that even a single all-zero starting point often yields good performance.

5.4 Discussion

We now discuss certain practical aspects of our optimization strategy. Our algorithms can be implemented at a central location, such as in Tesseract [39], or in a fully distributed manner. The distribution is similar to that in link-state protocols such as OSPF, in which all nodes implement the same algorithm, over the same data, and thus arrive at consistent solutions. Apart from topology information, distributing our algorithms also needs demand estimates for various flows.

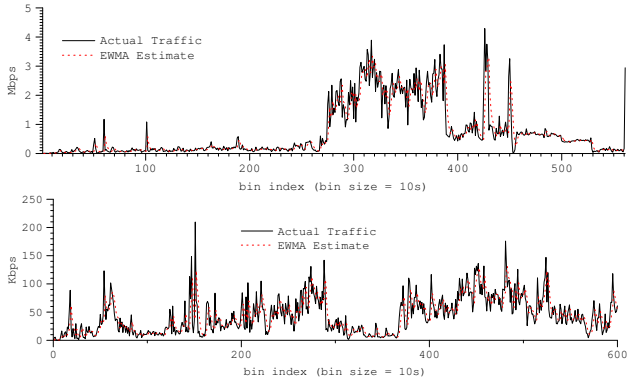


Figure 5.4: The amount of traffic sent to an AP in 10-second intervals. Top: At a WiFi hotspot. Bottom: At SIGCOMM 2004.

Another aspect that is related distributed implementation is the computational requirements of our approach. An exact quantification is a subject of ongoing work, but in our experiments we have not found it to be a problem. In our unoptimized implementation, rate computations are practical for online optimization. For instance, in our experiments, it takes roughly three seconds to optimize ten flows in 25-node topologies.

Finally, our methods use flow demands as inputs for optimization. We propose that nodes base their estimates on recent history. Such a strategy is effective only if there is temporal stability in flow demands. While wireless meshes are not significantly deployed yet to settle this question with certainty, we gain insight into this issue by studying wireless usage in two different environments – at a WiFi hotspot in Seattle and at the SIGCOMM 2004 conference [20]. Figure 5.4 shows for 10-second windows, the actual traffic sent to an AP and the traffic predicted by EWMA (α=0.5) over history. We see that traffic exhibits a high degree of temporal stability and EWMA predicts future traffic fairly accurately. What visually appears

as sharp peaks and valleys in traffic are in fact composed of multiple time intervals, compressed so that we can show a two-hour period. The average traffic volume is 723.5 Kbps for the hotspot trace and 43.77 Kbps for the SIGCOMM trace. The mean absolute error (MAE), defined as $mean(|Estimated - Actual|)$, is 200 Kbps for the hotspot trace and 15 Kbps for the SIGCOMM trace. Our rate-limiting would actually even out those spikes if there is not enough capacity in the network. Suppose the APs that we measure were nodes in a city-wide wireless mesh, aggregating traffic from similar clients and sending it to a nearby gateway on the multi-hop mesh backhaul. Then, by extrapolating from these environments, we judge that the nodes would be able to obtain reasonable estimates of their demands.

Chapter 6

Evaluation

We evaluate the accuracy of our approach using extensive testbed and simulation experiments. The former provides a setting with real-world complexities. The latter lets us conduct a broader range of experiments more easily and also lets us vary parameters such as topology that we cannot vary for the testbed.

6.1 Evaluation Methodology

6.1.1 Strawman: Conflict Graph Model

We compare our approach to one based on the conflict graph (CG) model of interference[13]. We note that the use of CG model has not been proposed in practical settings, but it provides an interesting comparison point in our evaluation. CG models interference but abstracts away the details of the MAC. The comparison lends insight into the importance of modeling the MAC.

The CG-based model assumes that packet transmissions at individual nodes can be finely controlled. It represents wireless links as conflict vertices and draws a conflict edge between two conflict vertices if and only if the corresponding wireless links interfere. Based on the definition, it is clear that links corresponding to conflict vertices in a clique in the conflict graph cannot be active simultaneously. Therefore,

an upper bound of optimal wireless throughput can be computed by solving a linear program (LP) which specifies the goal of maximizing the total traffic delivered to the destination while satisfying flow conservation and clique constraints.

We apply this formulation to derive the rate limits that maximize the total throughput. When applied to different route selection schemes, we enforce traffic to follow the selected routes by adding the following linear constraints. For each demand d and each link e, $T_{d,e} \leq Cap_e z_e$, where $T_{d,e}$ is the amount of traffic routed for demand d on link e, Cap_e is the capacity of link e, and $z_{d,e} = 1$ if e is used to route demand d and 0 otherwise.

To maximize fairness, we use a similar formulation. The main difference is that we change the objective to maximize the sum of the total throughput of all the flows and the portions of their demands that are achieved. This can be expressed as $\sum_d \sum_{r(e)=dest(d)} T_{d,e} + \lambda\alpha x_d$, where $r(e)$ is the receiver of link e, $dest(d)$ is destination of demand d, x_d is traffic demand, α is the minimum proportion of its demand that can be achieved, and λ controls the relative importance of these two objectives. Our evaluation uses $\lambda = 100$ to significantly favor the solution with high fairness when maximizing fairness.

6.1.2 Qualnet Simulation

Our simulations are based on Qualnet v3.9.5. We use 802.11a with a fixed bit rate of 6 Mbps and free-space model of signal propagation, which provides a communication range of 230 meters. The interference range of 253 meters. We add following additional functions in existing QualNet.

1. Artificial Loss

   In order to simulate different wireless environments, we add a function which can set MAC layer loss to a link. rfIWe assume bits loss are independent. So the inherent loss rate of a link is given as following:

$$Loss_{ij} = (1 - size_{data}^{BER_{ij}}) \times *(1 - size_{ack}^{BER_{ji}}) \tag{6.1}$$

2. Rate limiting

   In order to compare different routing schemes with and without rate limiting, we add application rate limiting functions into Qualnet.

3. Routing Package

   This package is implemented in python. The input of this package are wireless link information, such as link source, destination and inherent loss rate. It constructs a graph based on the information. The package set link weight according to different routing schemes and then apply shortest path algorithms to find routes. The output of it are static routes for all flows, which are installed into Qualnet simulation.

We generate traffic using both TCP and UDP and consider two types of application demands: (i) for *saturated demands*, sources always have traffic to send; and (ii) for *random demands*, the demand of a source is picked randomly from a uniform distributed between 0 and the maximum link load. We vary the number of flows from 1 through 20 where each flow is between a unique sender-receiver pair.

We consider two kinds of topologies in this paper: 5x5 grid topologies and 25-node random topologies. Both occupy a 900x900 m^2 area. We also study other network densities and find that the results are qualitatively similar. So we omit them from this paper in the interest of brevity.

For each scenario, we conduct 10 random trials. In each trial, flow sources and destinations are picked randomly. For random traffic demands and random topologies, each trial also randomly generates the demands and the topology.

We evaluate the performance with and without RTS/CTS. When RTS/CTS is enabled, we set RTS threshold to 100 bytes so that (small) TCP ACKs do not incur RTS/CTS overhead. In order for TCP to be robust to high link loss rates, we use TCP NewReno and set the MAC-level short and long retry counts to 16. This is the largest maximum retry count allowed in madwifi-old driver, which we use in our testbed.

Since several routing metrics (*e.g.*, ETX [4] and MIC [40]) are designed for wireless networks with lossy links, we extend Qualnet simulator to generate directional inherent packet losses. In our evaluation, we randomly assign bit-error-rate (BER) of links such that the data packet loss rates are uniformly distributed between 0 and 80%. As wireless link loss rates depend on frame sizes, our evaluation considers both small and large frames. They have respective application payload sizes of 106 bytes and 1024 bytes. The broadcast probes used to measure link quality for routing are also 106 bytes, as in [4].

6.1.3 Testbed Experiment

In our testbed experiments, we use the lowest transmission power for our nodes to increase network diameter. In this setting, we measured the diameter to be 7 hops, though routing paths may be longer.

We implement routing protocols using click [3]. Click is a software architecture for building flexible and configurable routers. A Click router is assembled from packet processing modules called elements. Individual elements implement simple router functions link packet classification, queueing, scheduling, and interfacing with network devices. Click defines a declarative language to make configurations.

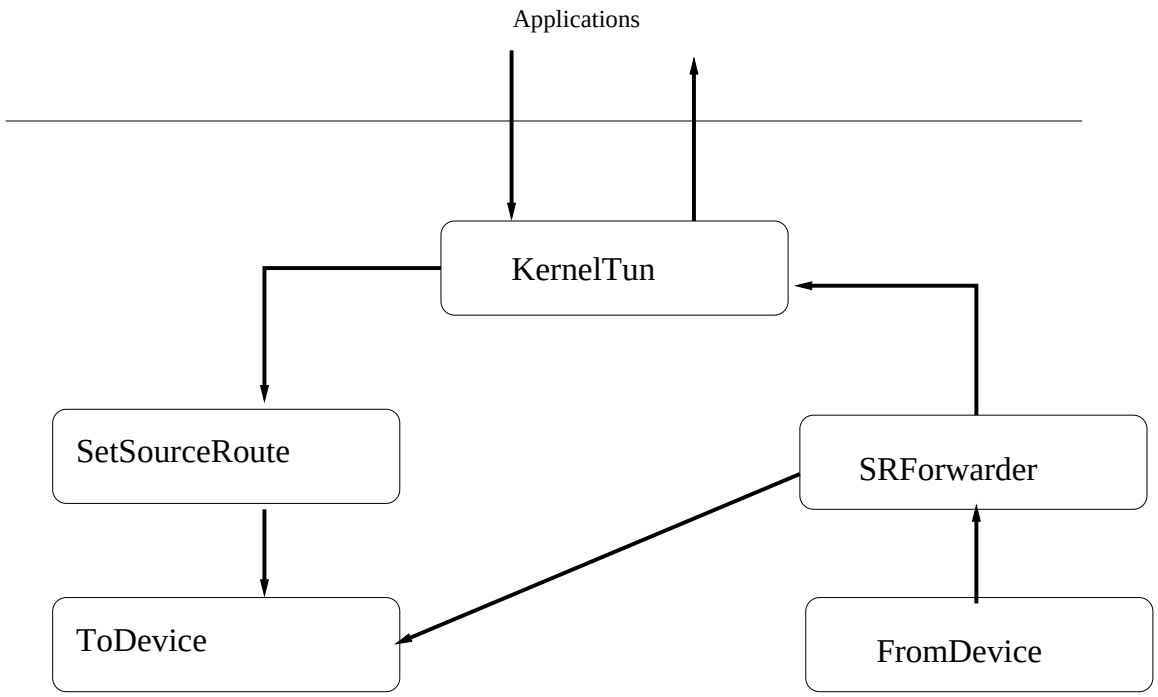


Figure 6.1: Click configuration

Figure 6.1 shows the configuration of the Click router in our experiments. When an application sends out a packet, the packet goes through KernelTap, and then the Click router handles the packet. The Click router inserts source routes into the packet's header based on the destination of the packet before it sends it out through ToDevice. When an incoming packet is received by the network card, it first goes through the Click router. The Click router checks the destination of this packet, if the destination is itself, it hands the packet to the ordinary kernel IP process code, and then the corresponding application can handle it; otherwise, it forwards the packet by sending it out of the network card.

We use *nuttcp* [22] to generate and measure UDP and TCP throughput. To rate limit flows, we let *nuttcp* to generate application traffic at the rates derived from the models.

Figure 6.2 shows process of our simulations and experiments. In Qualnet Simulation, the first step is to generates a topology, link loss rates and traffic demands; the second step is to generate conflict graph based on the topology, and then infer $S1$, $S2$, $R1$, $R2$; the third step is to feed $S1$, $S2$, $R1$ and $R2$ into our model; the fourth step is to compute routes , and estimate flow rates; the final step is to install computed routes and flow rates in Qualnet to measurement performance.

In Testbed Experiments, the first step is to conduct interference measurement to collect $S1$, $S2$, $R1$, $R2$. The rest steps are similar to simulations except that the final step is to install computed routes and flow rates in the testbed to measurement performance.

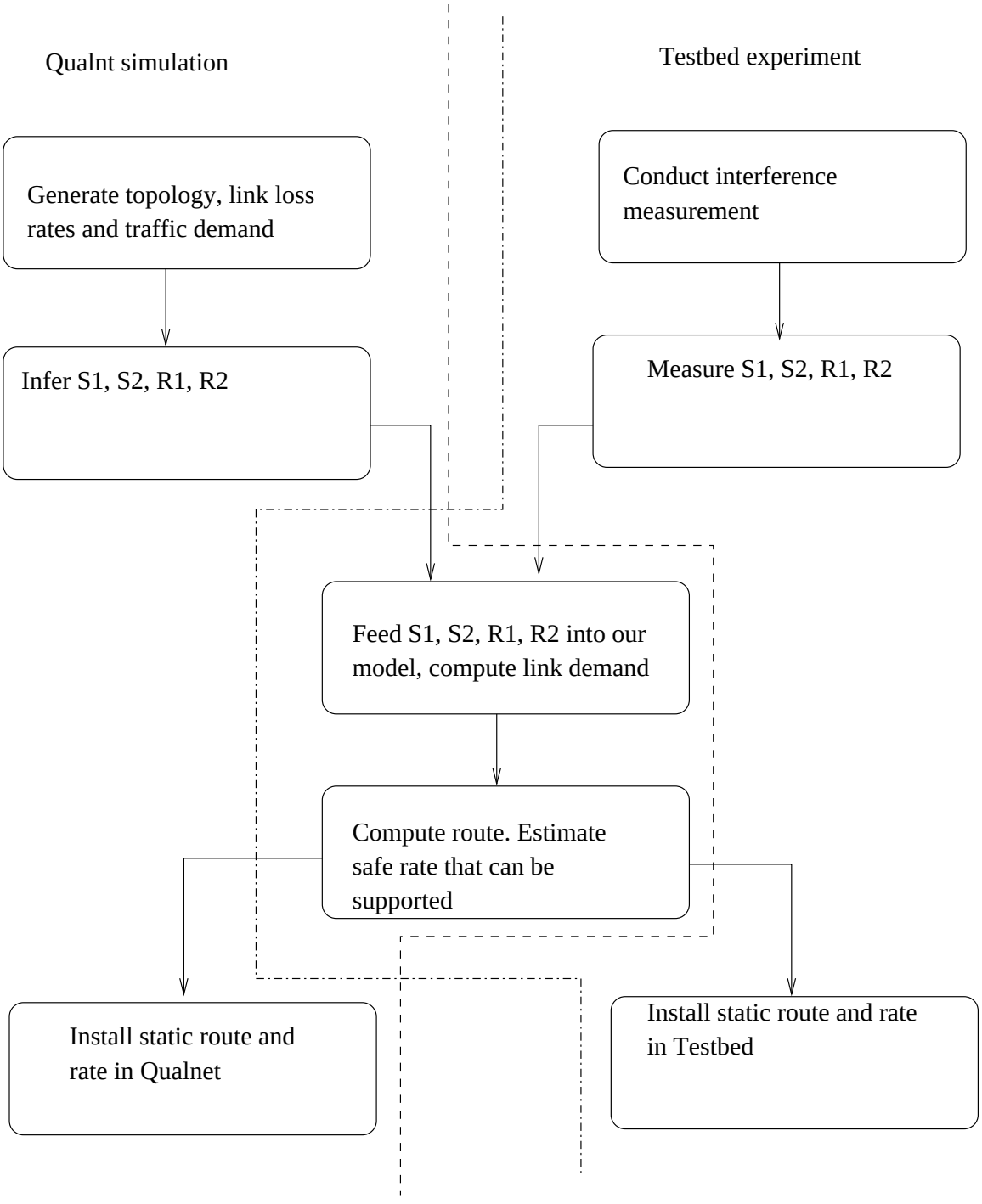


Figure 6.2: Process of Simulation and Experiment

6.2 Model Validation

Below we show that our model is accurate in a range of settings.

A good interference model should closely approximate achievable throughput given traffic demands as input, which implies that: (i) the throughput estimate should be achievable in the network, i.e., the model should not over-predict throughput; and (ii) the network should not be capable of delivering more throughput, i.e., the model should not under-predict. It is straightforward to evaluate for over-prediction – instantiate the estimated throughput to the network and check if the actual throughput comes close.

Evaluating under-prediction is more tricky. We would like to increase the load on the network beyond what the model estimates and check how often that leads to higher network throughput. However, given multiple flows, there are numerous ways to increase network load. Our experiments use a simple uniform scaling approach that increases each flow throughput by the same factor. We use scaling factors of 1.1, 1.2, and 1.5, which correspond to increasing load by 10%, 20%, and 50%.

Figure 6.3 shows the format in which we present results in this section. To evaluate under-prediction, the left graph shows a scatter plot of actual and estimated throughput. The two lines on the scatter plot correspond to $y=x$ and $y=0.8x$. They help judge the accuracy of the model visually. There will be no points above $y=x$ as the network can never achieve more throughput than what is instantiated. The points below $y=0.8x$ correspond to instances where the actual throughput is less than

80% of what is predicted by our model. The right graph is a CDF of the ratio of actual and estimated throughput, before and after scaling. The *y*-value of the point where a scaled curve reaches x=1 represents the fraction of cases where our model under-predicted by at least the scaling factor. The figures aggregate results across all flow counts that we generate. These counts vary between 1-20 in simulations and 1-16 in our testbed experiments.

In the experiments below, we use a data packet payload of 1024 bytes and use ETX to select routes. We find qualitatively similar results for smaller payloads (not shown) and other routing schemes (Section 6.4).

6.2.1 Simulation Experiments

Figure 6.3 shows the accuracy of predicting the throughput in a grid topology with saturated UDP demands and without RTS/CTS. We can see from the scatter plot that the vast majority of the points lie between the lines, which implies that we over-predict network throughput by more than 20% in very few cases. From the scale=1 CDF on the right, we can see that there are fewer than 15% such cases. Meanwhile, the worst-case overestimate is under 50%. A major cause for these over-predictions is that our model assumes pairwise interference. The model over-predicts when neither two senders interfere with a link alone but their total noise collectively interferes with the link.

The scaled CDFs show that our model does not under-predict either in this configuration. In almost all cases, the network is unable to achieve demands that have been scaled by even 10%.

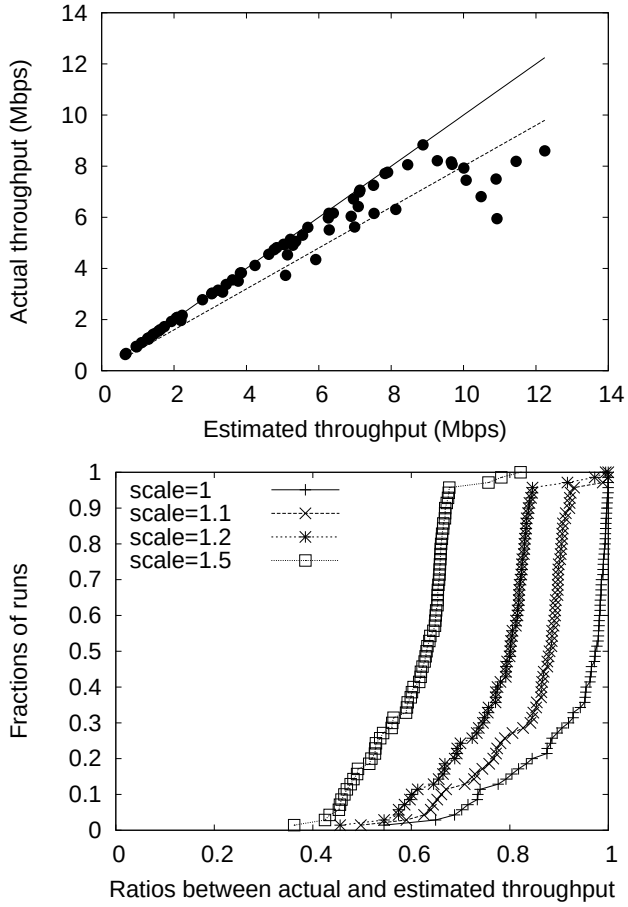


Figure 6.3: Throughput prediction accuracy in simulation of our model for grid topologies, saturated UDP traffic, and RTS/CTS = OFF.

For the same configuration, Figure 6.4 shows the accuracy of the CG-based model. Clearly, this model vastly over-predicts what the network can achieve, because of the assumptions it makes about the ability of the nodes to finely coordinate their transmissions. From the CDFs, we can see the network achieves less than half of the predicted throughput in half of the cases. Thus, modeling 802.11 DCF, as our model does, is key to accurate predictions of network throughput. Interestingly,

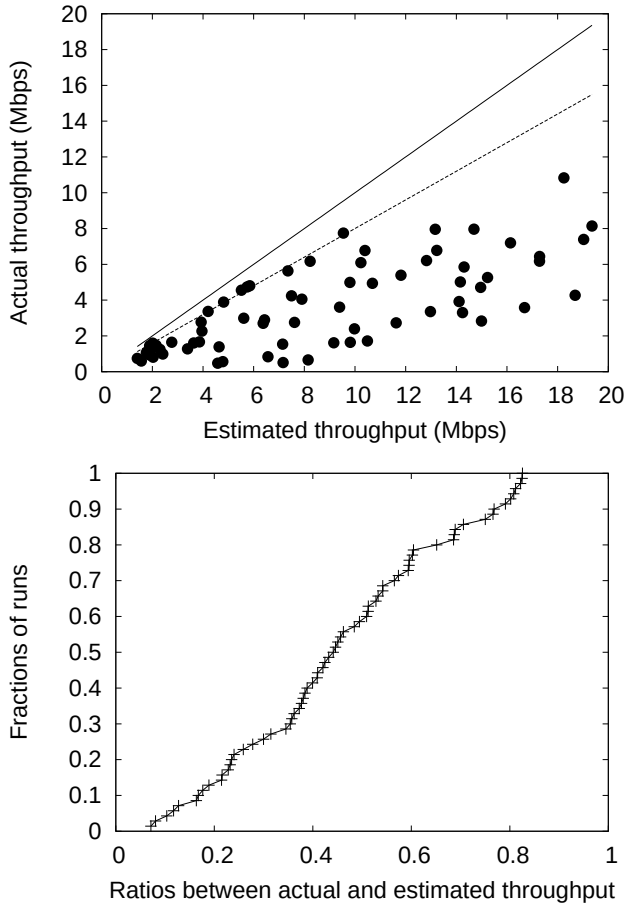
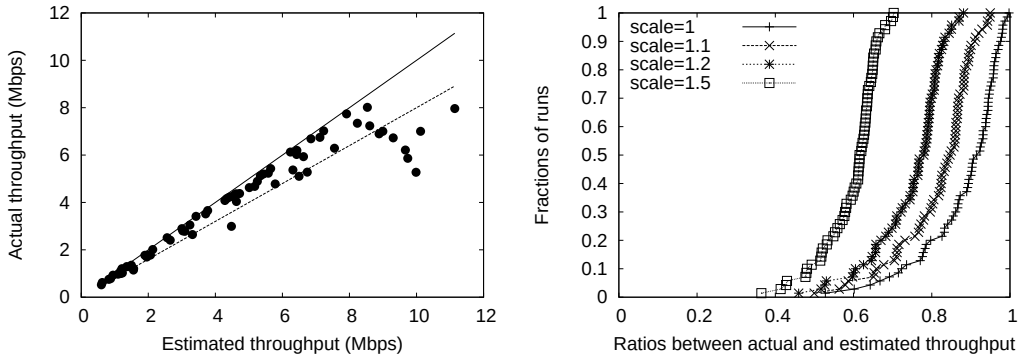


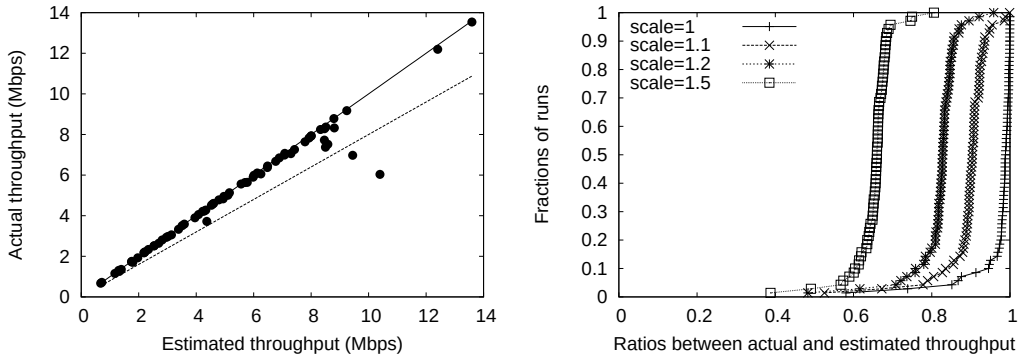
Figure 6.4: Throughput prediction accuracy in simulation of the CG-based model for grid topologies, saturated UDP demands, and without RTS/CTS.

the inaccuracy of the CG-based model also hints at the performance cost of the CSMA-based 802.11 MAC under heavy load.

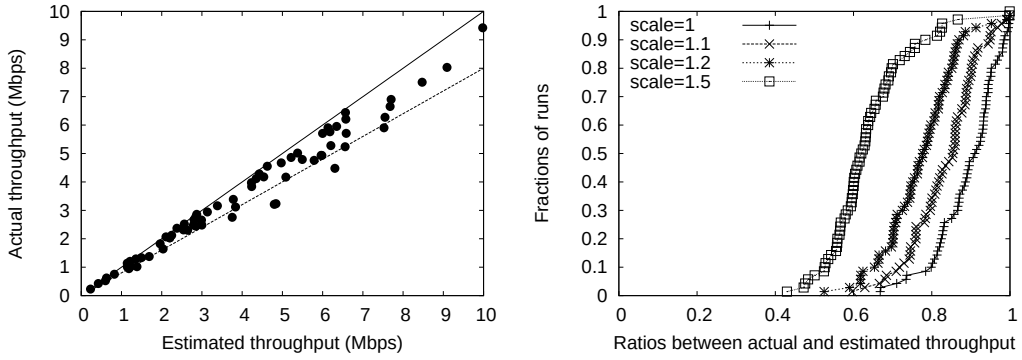
Figure 6.5 shows that our model is robust across a wide range of other simulated configurations. For TCP traffic, it overestimates throughput by more than 20% in fewer than 20% of the cases. This accuracy is less than that for UDP because TCP creates bursty traffic and losses, which we do not currently model. However,



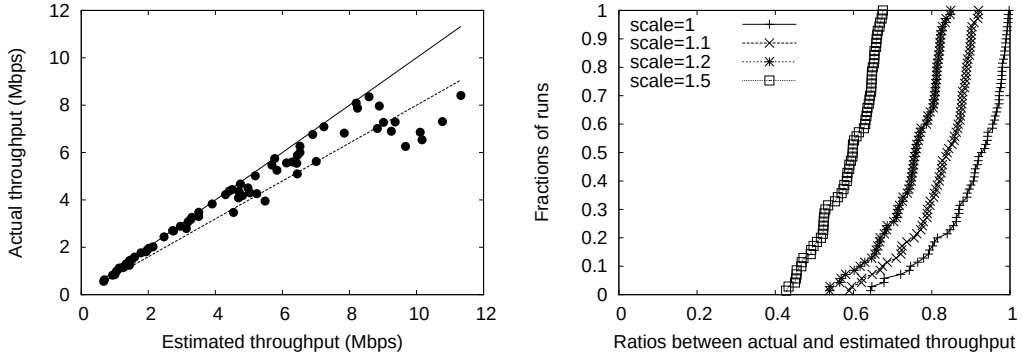
(a) *Grid topology,* **saturated TCP demands**, *without RTS/CTS*



(b) **Random topology**, *saturated UDP demands, without RTS/CTS*

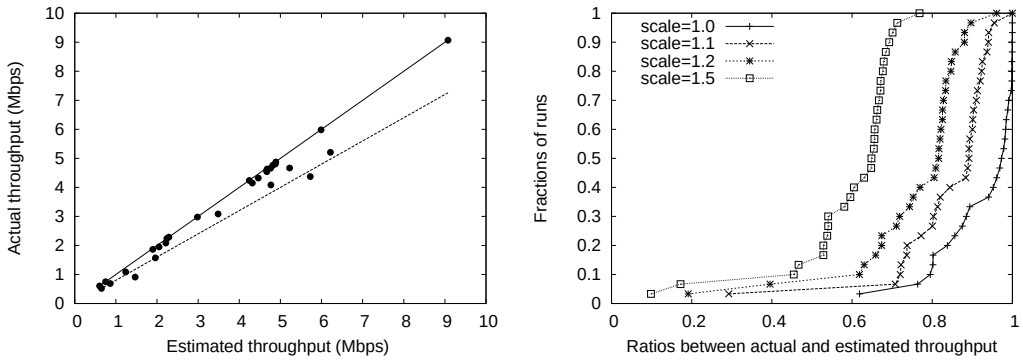


(c) *Grid topology,* **random UDP demands**, *without RTS/CTS*

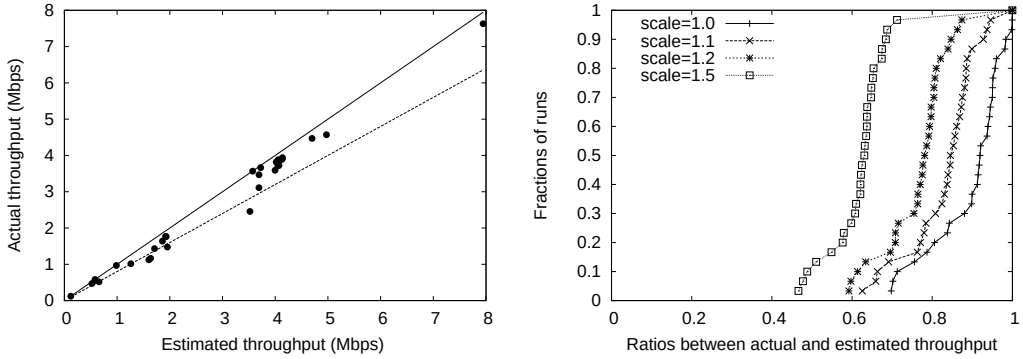


(d) *Grid topology, saturated UDP demands,* **with RTS/CTS**

Figure 6.5: Throughput prediction accuracy in simulation of our model for various configurations. The difference from the base configuration in Figure 6.3 is in bold.



(a) *Saturated UDP demands*

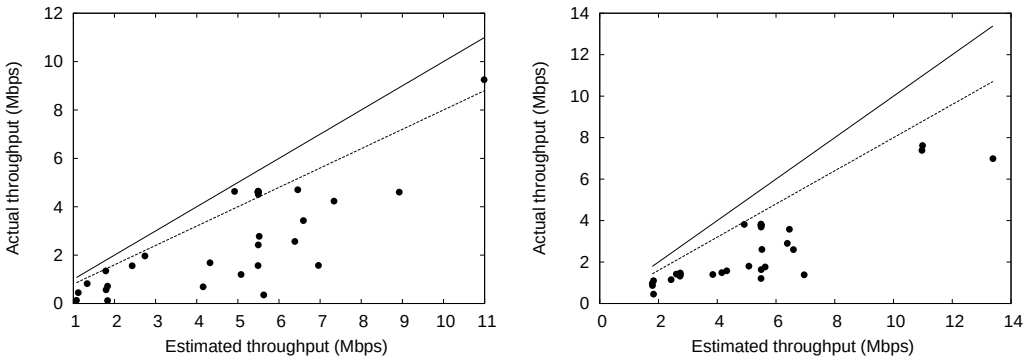


(b) *Saturated TCP demands*

Figure 6.6: Throughput prediction accuracy of our model in our testbed. RTS/CTS=OFF.

as for UDP, we never under-predict the network's TCP throughput even by 10%.

The remaining graphs in the figure show that the accuracy of our model is high even when we switch from grid to random topologies, or from saturated demands to randomly assigned demands, or from not using RTS/CTS to using it.



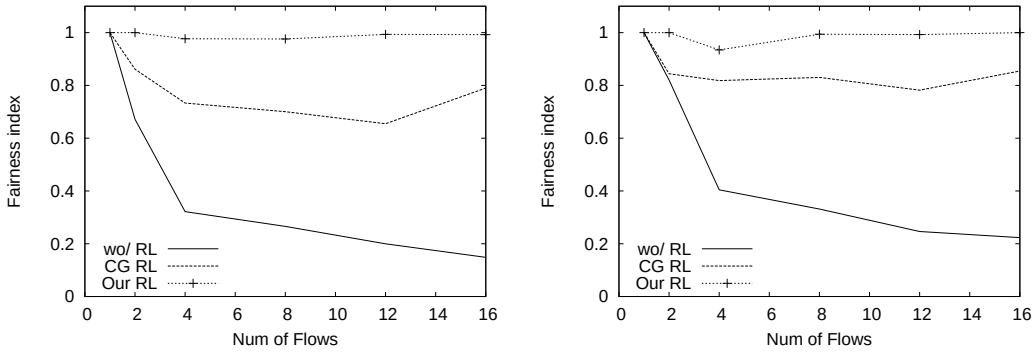
(a) Saturated UDP traffic (b) Saturated TCP traffic

Figure 6.7: Throughput prediction accuracy in our testbed using CG-model for saturated demands and RTS/CTS = OFF.

6.2.2 Testbed Experiments

Figure 6.6 shows that our model is fairly accurate in the more realistic testbed setting as well. For UDP, only in 10% of the cases we over-predict throughput by more than 20%. For TCP, this over-prediction occurs for 20% of the cases, which is similar to that in simulation. The worst-case over-prediction is less than 40% for both TCP and UDP. Meanwhile, as in simulation, our model does not under-predict either. For both TCP and UDP, the network is unable to achieve demands that have been scaled by even 10%.

Figure 6.7 shows the throughput prediction accuracy using CG-model. We see that, as in simulation, the CG-model consistently over-estimates the achievable rates. Almost all the points are outside the cone formed by $y = x$ and $y = 0.8x$, which indicates that in most cases its estimated demands are not achievable within 80%.



(a) *Saturated UDP traffic* (b) *Saturated TCP traffic*

Figure 6.8: Fairness comparison in our testbed. RTS/CTS=OFF.

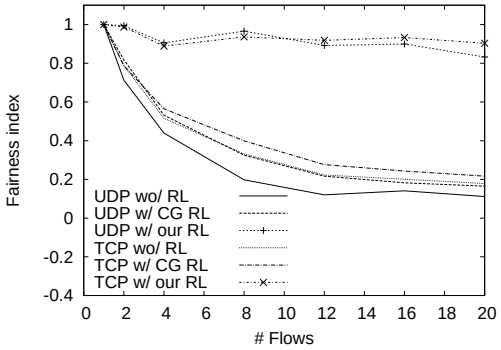
6.3 Performance Optimization

Can the accuracy of our model in predicting the throughout supported by the network be harnessed to improve performance, using the methods we outlined earlier? We answer this question in this section by first considering fairness maximization and then throughput maximization. We compare results with no rate-limiting, as it happens today, and with rate-limiting using the conflict graph (CG) model.

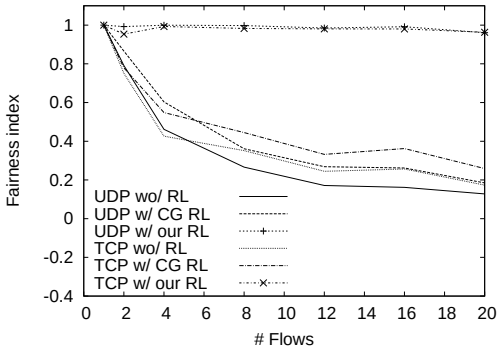
6.3.1 Maximizing Fairness

Figure 6.8 shows the fairness index for TCP and UDP traffic in our testbed. We see that the fairness index with our algorithm is remarkably close to 1 for both kinds of traffic and across all offered loads. Without rate-limiting as well as with the CG-based rate limiting, fairness degrades quickly as load increases.

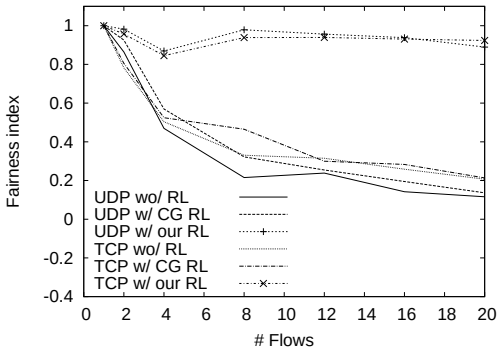
Figure 6.9 shows the fairness provided by our model-driven approach holds in a range of simulated configurations, for both TCP and UDP traffic, including grid and random topologies, with and without RTS/CTS, and with saturated or random



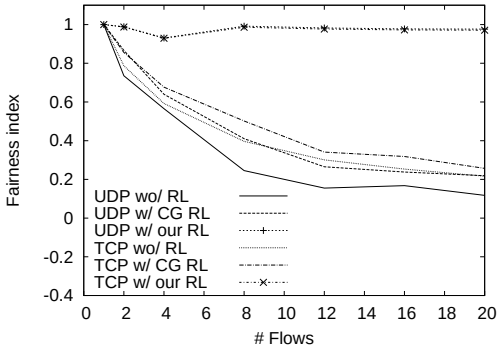
(a) *Grid topology, saturated demands, RTS/CTS=OFF*



(b) **Random topology**, *saturated demands, RTS/CTS=OFF*

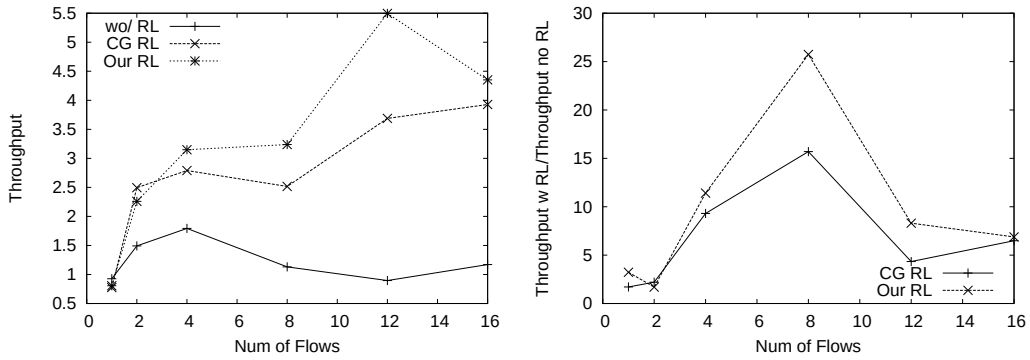


(c) *Grid topology,* **random demand**, *RTS/CTS=OFF*

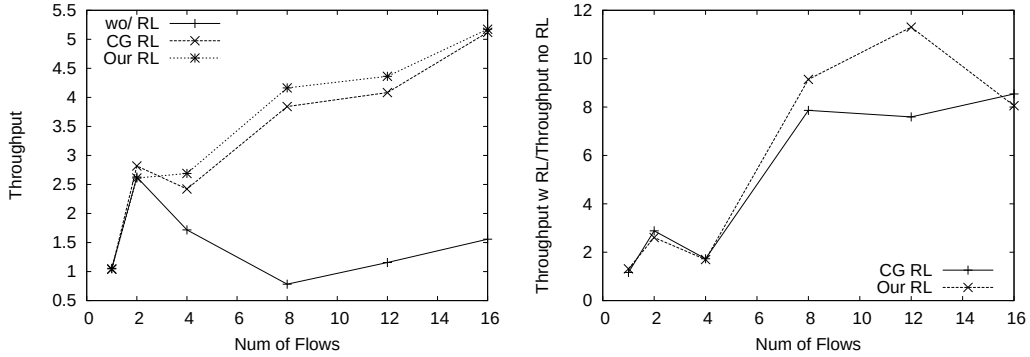


(d) *Grid topology, saturated demand,* **RTS/CTS=ON**

Figure 6.9: Fairness improvement in simulation for difference configurations. The aspect of a configuration that differs from the first one is in bold.



(a) *Saturated UDP demand*



(b) *Random UDP demand*

Figure 6.10: UDP throughput improvement in our testbed with rate-limiting.

demand models.

6.3.2 Maximizing Total Throughput

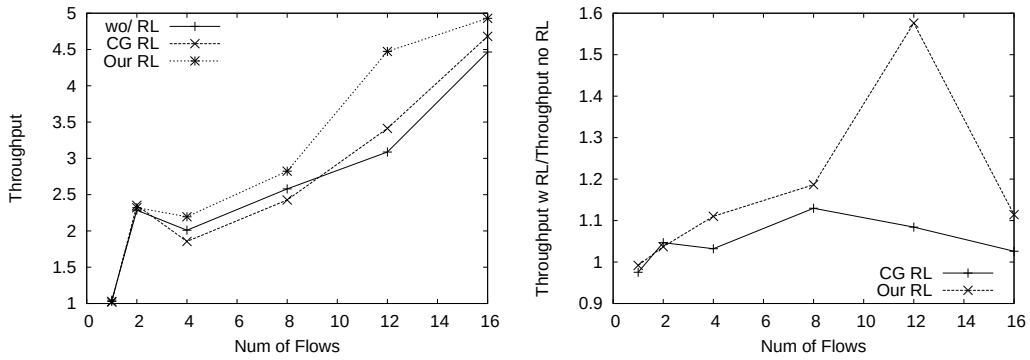
We consider the performance objective of maximizing total throughput.

Figure 6.10(a) shows that the benefits of rate limiting for saturated UDP traffic in our testbed are significant. The graph on the left plots the average total throughput, and the graph on the right plots the average normalized throughput (*i.e.*, the throughput under rate limit normalized by the throughput under no rate

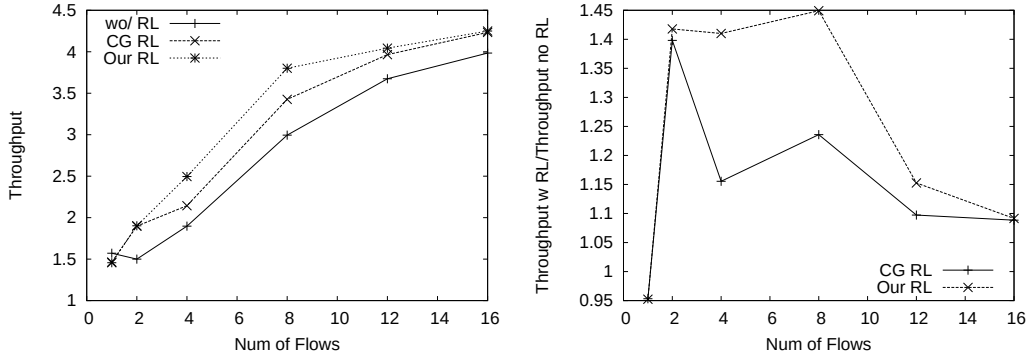
limit). In terms of absolute throughput, UDP traffic experiences over 200% improvement; in terms of normalized throughput, the average improvement ranges from 200% to 1700%. The larger improvement in the latter suggests that rate limiting is especially beneficial to the flows that experience low throughput under no rate limiting. Unlike the task of increasing fairness, the CG-based model helps boost network throughput. Like our model, it is able to identify interference-related bottlenecks and impose rate limits. However, because the CG-based model significantly over-predicts throughput (Section 6.2), the loss rate in the network is much higher and the throughput is consistently lower. Figure 6.10(b) shows the benefit of rate limiting extends to random UDP demands.

Figure 6.11 shows that the gain from rate-limiting saturated and random TCP flows is a more modest 10-40%. This lower improvement for TCP is expected given that we experiment with infinitely long flows that react well to congestion, thus minimizing interference-related losses. However, we believe that rate-limiting will provide substantial benefits when TCP traffic is composed of many short transfers, as is common for Web transactions, because an aggregate of short TCP flows is significantly less responsive to losses than long TCP flows.

Figure 6.12 shows the network throughput improvement for various simulated configurations with UDP traffic. The error bars denote standard deviation. We see results consistent with the testbed across all configurations, except that the CG-based model does as well as our model only in random topologies. This is likely because there are a small number of bottleneck links in the random topologies and CG also tries to avoid them.



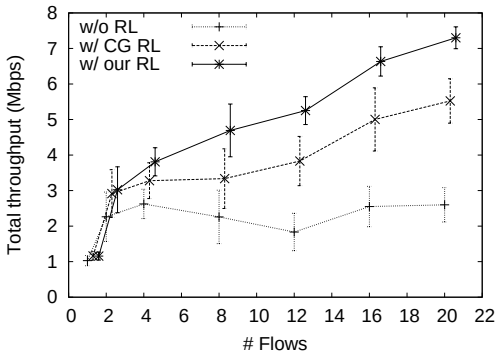
(a) *Saturated TCP demand*



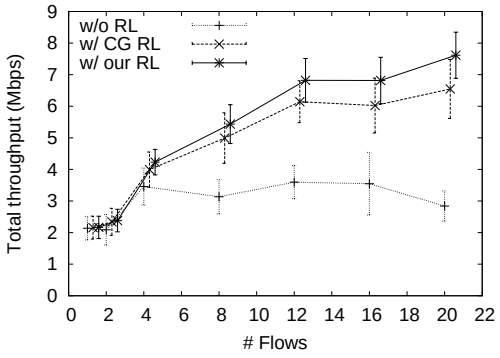
(b) *Random TCP demand*

Figure 6.11: TCP Throughput improvement in our testbed with rate-limiting.

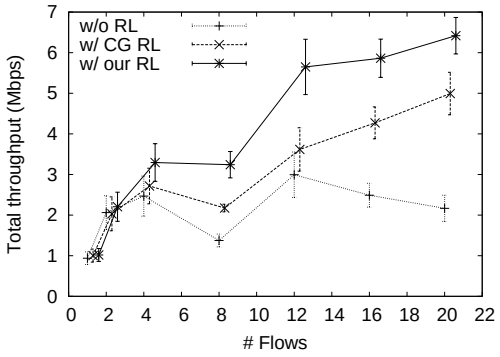
Figure 6.13 shows the effectiveness of rate limiting for TCP traffic in simulated configurations with and without RTS/CTS. We see, as with the testbed, the benefit of rate-limiting tends to increase with more flows. Its benefit increases to 20-40% when the number of flows reaches 20. In general, rate-limiting helps TCP traffic less than UDP traffic.



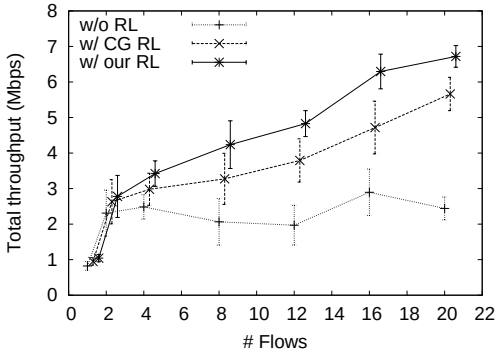
(a) *Grid topology, saturated UDP demand, RTS/CTS=OFF*



(b) **Random topology**, *saturated UDP demand, RTS/CTS=OFF*

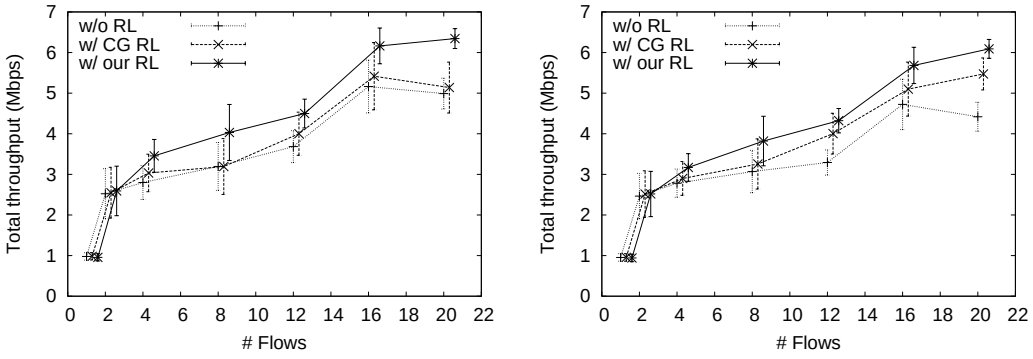


(c) *Grid topology,* **random UDP demand**, *RTS/CTS=OFF*



(d) *Grid topology, saturated UDP demand,* **RTS/CTS=ON**

Figure 6.12: Throughput improvement in simulation with rate limiting for UDP traffic for various configurations. The aspect that differs from the first configuration is in bold.



(a) *RTS/CTS=OFF* (b) *RTS/CTS=ON*

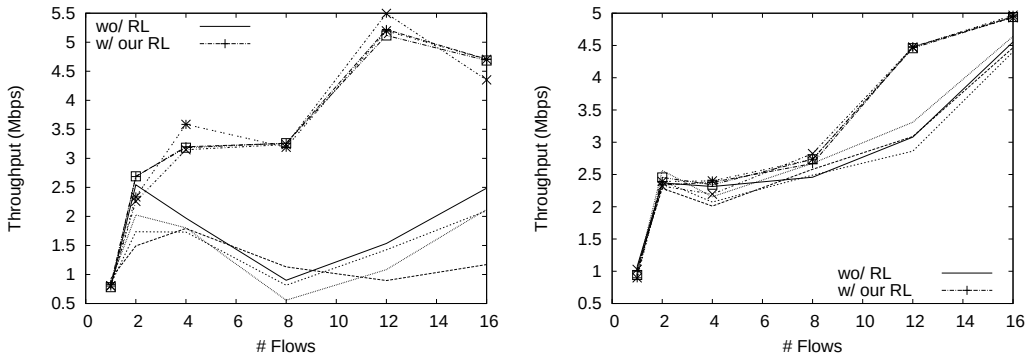
Figure 6.13: Throughput improvement in simulation with rate limiting for saturated TCP demand and grid topology.

6.4 The Role of Routing

All the results above are based on routing paths chosen by the ETX protocol. In this section, we show that, surprisingly, the choice of the exact routing protocol makes little difference in our experiments. We study three other protocols and find that all four behave similarly. What seems to matter most is whether flows are being rate-limited.

The three other protocols that we study are the following.

- *HOP* selects a path with minimum hop-count.
- *MIC* [40] scales ETX values of a link by multiplying it by the sum of the neighbors of the two end points. It then selects a path with the minimum scaled ETX value.
- *CG* selects the routes by casting the routing problem to a maximum flow problem augmented with interference constraints derived by a conflict graph [13].



(a) *Saturated UDP demand* (b) *Saturated TCP demand*

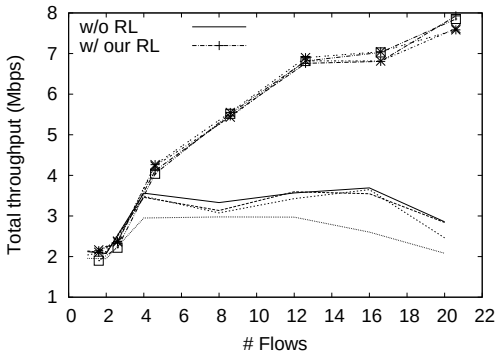
Figure 6.14: Throughput in our testbed of the four routing methods with and without rate-limiting. The top four lines in each graph are for the case of rate-limiting and the bottom four are for non-rate-limiting.

These routes are close to optimal if nodes can finely coordinate transmissions.

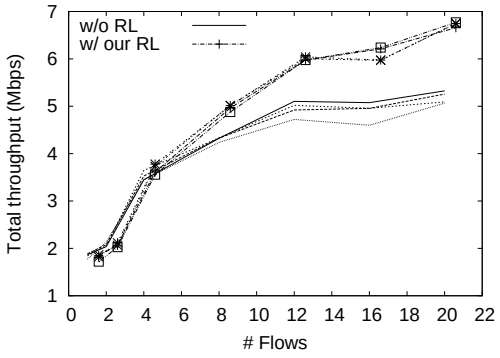
We consider only the goal of maximizing throughput in this paper, but we obtain similar results for maximizing fairness.

Figure 6.14 shows UDP and TCP performance under different routing schemes. The bottom four curves are the performance of different routing schemes under no rate limiting, and the top four curves show the results using rate limiting based on our model, with the objective of maximizing total throughput. We see that the routing schemes are almost indistinguishable. Rate-limiting does matter, however. For each scheme, rate-limiting using our model provides 50-400% gain for UDP and 10-45% for TCP.

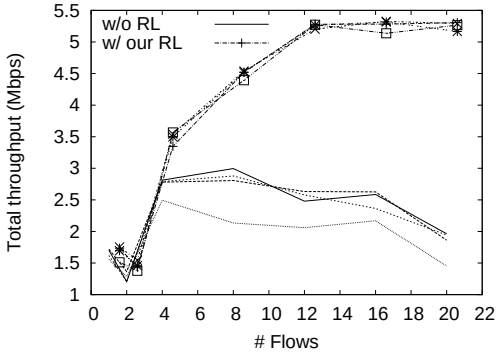
In Figure 6.15, we can see the same effect in other simulated configurations. Routing does not seem to matter whether we have TCP or UDP traffic, saturated or random demands, big or small payloads. To rule out differences in probe packet



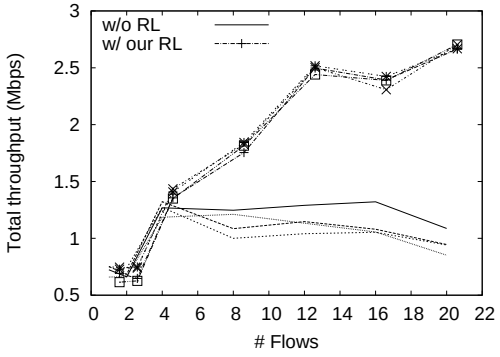
(a) *Random topology, saturated UDP demand, 1024-byte payload*



(b) *Random topology,* **saturated TCP demand**, *1024-byte payload*



(c) *Random topology,* **random UDP demand**, *1024-byte payload*



(d) *Random topology, saturated UDP demand,* **106-byte payload**

Figure 6.15: Throughput in simulations of the four routing methods – HOP, ETX, MIC, and CG – with and without rate-limiting. The top four lines in each graph are for the case of rate-limiting and the bottom four are for non-rate-limiting. The aspect that differs from the first configuration is in bold.

size and payload size, which may cause ETX to select the wrong path, we also considered probe-sized payload packets. As Figure 6.15(d) shows, that does not make a significant difference either.

These routing protocols differ in how they account for interference, but they all have their shortcomings on that front (see our previous work [18] for more details). For example, the ETX metric is determined by packet loss rates at receivers, so it only captures receiver-side interference but fails to capture sender-side interference that stops nodes from transmitting. Moreover, the characteristics of probing traffic and data traffic can be quite different in terms of, for instance, volume, packet sizes and generation pattern, which makes the two observe different loss rates. Therefore, the ETX metric does not accurately predict the actual performance experienced by data traffic. The MIC metric is based on ETX, so it has similar issues. The CG-based routing assumes perfect scheduling and tends to select longer detours, which perform well under perfect scheduling but not under 802.11.

What we show is that once we have properly managed interference through rate-limiting, the small variations in routing paths produced by these protocols have relatively low impact on total network throughput. We also repeat that our methods for rate-limiting are agnostic to the choice of the underlying routing protocol. They can thus work with whichever routing method that provides better performance in the given setting.

Chapter 7

Conclusions and Future Work

In this section, we summarize our contributions, and point out directions for future research.

7.1 Conclusions

We study interference in wireless networks through empirical experiments and simulations. We find out that current routing protocols face difficulties in effectively managing it. For instance, Wireless network throughput is sensitive to bottleneck link location; existing routing metrics, such as ETX, does not always reflect which paths are better; the effect of backoff can be significant when loss rate is high; exiting routing metrics measures of link quality may not reflect actual quality.

As interference depends on many factors, such as location, environment, and traffics. It is inevitable to consider interference in a systematic way. Our work demonstrates the feasibility of predictable performance optimization for wireless networks, thus making the task of managing and optimizing them as predictable as that for wired networks. The foundation of our approach is a new model that captures interference, traffic, and MAC-induced dependencies in the network using only a small set of constraints. Our model is realistic enough to handle real-world

complexities such as hidden terminals, non-uniform demands, and non-binary interference, and yet it is lightweight enough to drive network optimization.

Evaluations of our methodology using a testbed and simulations showed that it is very effective. Across a range of topology and traffic configurations, it was able to accurately approximate the throughput that the network yielded. It rarely under-predicted, and for 80% of the cases, it estimated within 20% of the actual throughput. When maximizing fairness using our methods, we achieved close to perfect fairness amongst flows for both UDP and TCP traffic. When maximizing throughput, we found that our methods can improve network throughput by 100-200% for UDP-based traffic and 10-50% for TCP-based traffic.

7.2 Future Work

We plan to address several practical issues in order to apply the approach to operational wireless networks.

- we plan to develop novel measurement techniques to passively and accurately estimate interference and seed our model.
- we plan to evaluate our approach under realistic traffic demands that change with time.
- we plan to improve the efficiency of disseminating the inputs to our algorithms by adapting the update frequency based on the rate of change and applying delta encoding.

Appendix

Appendix 1

Derivatives of F and G

In this appendix, we compute the derivatives of F and G.

$$F_i(\tau) = \frac{EP_i \times \tau_i \times (1-p_i)}{T_{\text{slot}} + \sum_j \left[(W_{ij} - T_{\text{slot}}) \times \tau_j\right]} \tag{1.1}$$

$$G_i(\tau) = \tau_i - \frac{2}{2 + CW(p_i)} \tag{1.2}$$

Let $\theta_i = \frac{\tau_i}{T_{\text{slot}} + \sum_j \left[(W_{ij} - T_{\text{slot}}) \times \tau_j\right]}$ We can rewrite Eq. 1.1 as

$$F_i(\tau) = EP_i \times \theta_i \times (1-p_i) \tag{1.3}$$

Do derivatives, then we have

$$\frac{\partial}{\partial \tau_i} F_i = EP_i \times \frac{\partial}{\partial \tau_i}\theta_i \times (1-p_i) - EP_i \times \frac{\partial}{\partial \tau_i} p_i \times \theta_i \tag{1.4}$$

$$\frac{\partial}{\partial \tau_i} G_i = 1 + \frac{2}{(2+CW(p_i))^2} \times \frac{dCW(p_i)}{dp_i} \times \frac{\partial}{\partial \tau_i} p_i \tag{1.5}$$

In order to compute Eq. 1.4 and 1.5, we need to compute $\frac{\partial}{\partial \tau_i}\theta_i$, $\frac{\partial}{\partial \tau_i} p_i$, and $\frac{dCW(p_i)}{dp_i}$

$$\frac{\partial}{\partial \tau_k}\theta_i = \begin{cases} -\frac{\tau_i [W_{ik} - T_s lot]}{\left\{T_{\text{slot}} + \sum_j \left[(W_{ij} - T_{\text{slot}}) \times \tau_j\right]\right\}^2} & k \neq i, \\ \frac{1}{T_{\text{slot}} + \sum_j \left[(W_{ij} - T_{\text{slot}}) \times \tau_j\right]} - \frac{\tau_i \times (W_{ii} - T_{\text{slot}})}{\left\{T_{\text{slot}} + \sum_j \left[(W_{ij} - T_{\text{slot}}) \times \tau_j\right]\right\}^2} & k = i. \end{cases} \tag{1.6}$$

We have

$$p_i = 1-(1-L_i^{\text{dat}})\times(1-L_i^{\text{ack}})\times\prod_{j\neq i}\left[(1-S_{ij}\tau_j)\times(1-\theta_j)^{A_{ij}}\right] \tag{1.7}$$

$$Let\ S_i = \prod_{j\neq i}\left[(1-S_{ij}\tau_j)\times(1-\theta_j)^{A_{ij}}\right] \tag{1.8}$$

Therefore

$$lnS_i = \sum_{j\neq i} ln(1-S_{ij}\tau_j) + \sum_{j\neq i} A_{ij} ln(1-\theta_j) \tag{1.9}$$

$$\frac{\partial}{\partial\tau_k} lnS_i = \frac{-S_{ik}}{1-S_{ik}\tau_k} - \sum_{j\neq i} A_{ij}\frac{\frac{\partial}{\partial\tau_k}\theta_j}{1-\theta_j} \tag{1.10}$$

$$\frac{\partial}{\partial\tau_k} p_i = C_i\times S_i\times\left[\frac{S_{ik}}{1-S_{ik}\tau_k} + \sum_{j\neq i} A_{ij}\frac{\frac{\partial}{\partial\tau_k}\theta_j}{1-\theta_j}\right] \tag{1.11}$$

Where $C_i = (1-L_i^{\text{dat}})(1-L_i^{\text{ack}})$

$$\begin{aligned} CW(p_i) &= CW_{min} + p_i\times(1+CW_{min})\times\sum_{k=0}^{M-1}(2p_i)^k \\ &= CW_{min} + p_i\times(1+CW_{min})\times\frac{1-(2p_i)^M}{1-2p_i} \\ &= CW_{min} + \frac{1+CW_{min}}{2}\times\frac{2p_i-(2p_i)^{M+1}}{1-2p_i} \end{aligned}$$

$$\frac{dCW(p_i)}{dp_i} = (1+CW_{min})\left[\frac{1-(M+1)(2p_i)^M}{1-2p_i} + \frac{2p_i-(2p_i)^{M+1}}{(1-2p_i)^2}\right]$$

$$
\begin{aligned}
&= (1+CW_{min})\frac{1-2p_i-(M+1)(1-2p_i)(2p_i)^M+2p_i-(2p_i)^{M+1}}{(1-2p_i)^2} \\
&= (1+CW_{min})\frac{1+(2p_i)^M\left[M\dot{2}p_i-(M+1)\right]}{(1-2p_i)^2} \qquad (1.12)
\end{aligned}
$$

We can plug in Eq. 1.6, 1.11 into Eq. 1.4 to compute the derivative of $F_i(\tau)$ and Eq. 1.11 , 1.12 into Eq. 1.5 to compute the derivate of $G_i(\tau)$.

Bibliography

[1] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *Microsoft Technical Report, MSR-TR-2003-41*, June 2003.

[2] G. Bianchi. Performance analysis of the IEEE 802.11 distributed corrdination function. In *IEEE Journal on Selected Areas in Communications*, Mar. 2000.

[3] Click. http://pdos.csail.mit.edu/click/.

[4] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MOBICOM*, Sept. 2003.

[5] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proc. of ACM MOBICOM*, Sept. - Oct. 2004.

[6] F.A.Tobagi and L. Kleinrock. Packet switching in radio channels: Part ii-the hidden terminal problem in carrier sensing multiple access and busy tone solution. *IEEE Trans.on Commun.*, pages 1417–1433, 1975.

[7] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4), 1999.

[8] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla. The impact of of multihop wireless channel on TCP performance. *IEEE Trans. on Mobile Computing*, Mar. 2005.

[9] Y. Gao, J. Lui, and D. M. Chiu. Determining the end-to-end throughput capacity in multi-hop networks: Methodolgy and applications. In *Proc. of ACM SIGMETRICS*, Jun. 2006.

[10] M. Garetto, J. Shi, and E. Knightly. Modeling media access in embedded two-flow topologies of multi-hop wireless networks. In *Proc. of ACM MOBICOM*, Aug. - Sept. 2005.

[11] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang. TCP over wireless multi-hop protocols: Simulation and experiments. In *Proc. of 1999 IEEE International Conference on Communications (ICC)*, Jun. 1999.

[12] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2), Mar. 2000.

[13] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proc. ACM MOBICOM*, Sept. 2003.

[14] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multihop wireless ad hoc networks. In *Ad Hoc Networking*, 2001.

[15] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proc. of the ACM/IEEE*

International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Oct. 2004.

[16] David Kotz, Calvin Newport, and Chip Elliott. The mistaken axioms of wireless-network research. Technical Report TR2003-467, Dartmouth College, Computer Science, Hanover, NH, July 2003.

[17] A. Kumar, E. Altman, D. Miorandi, and M. Goyal. New insights from a fixed point analysis of single cell IEEE 802.11 wireless LANs. In *Proc. of IEEE INFOCOM*, Mar. 2005.

[18] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, Z. Zhong, G. Deshpande, and E. Rozner. Effects of interference on throughput of wireless mesh networks: Pathologies and a preliminary solution. In *Proc. of HotNets-VI*, Nov. 2007.

[19] X. Lin and N. B. Shroff. Joint rate control and scheduling in multihop wireless networks. In *Proc. of IEEE Conference on Decision and Control*, 2004.

[20] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the mac-level behavior of wireless networks. In *Proc. of SIGCOMM*, 2006.

[21] Dragos Niculescu. Interference map for 802.11 networks. In *Proc. of Internet Measurement Conference (IMC)*, November 2007.

[22] nuttcp. ftp://ftp.lcp.nrl.navy.mil/pub/nuttcp/.

[23] L. M. S. C. of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.

[24] J. Padhye, S. Agarwal, V. N. Padmanabhan, L. Qiu, A. Rao, and B. Zill. Estimation of link interference in static multi-hop wireless networks. In *Proc. of Internet Measurement Conference (IMC)*, Oct. 2005.

[25] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proc. of ACM SIGCOMM*, 1994.

[26] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, Feb. 1999.

[27] L. Qiu, Y. Zhang, F. Wang, M. K. Han, and R. Mahajan. A general model of wireless interference. In *Proc. of ACM MOBICOM*, Sept. 2007.

[28] The Qualnet simulator from Scalable Networks Inc. http://www.scalable-networks.com/.

[29] Sumit Rangwala, Ramakrishna Gummadi, Ramesh Govindan, and Konstantinos Psounis. Interference-aware fair rate control in wireless sensor networks. In *ACM SIGCOMM*, September 2006.

[30] J.Padhye R.Draves and B.Zill. The architecture of the link quality source routing protocol. In *Microsoft Technical Report, MSR-TR-2004-57*, 2004.

[31] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-based models of delivery and interference. In *Proc. of ACM SIGCOMM*, 2006.

[32] Anand Prabhu Subramanian, Milind M. Buddhikot, and Scott Miller. Interference aware routing in multi-radio wireless mesh networks. In *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, September 2006.

[33] Questions for Tropos: Does Google's mountain view network fold under pressure? http://www.muniwireless.com/article/articleview/5395.

[34] Wi-Fi city sees startup woes. http://www.wired.com/techbiz/media/news/2006/04/70720.

[35] Another muni WiFi network gets an early thumbs down. http://www.techdirt.com/blog/wireless/articles/20061226/145441.shtml.

[36] City-wide Wi-Fi rolls out in UK. http://news.bbc.co.uk/2/hi/technology/4578114.stm.

[37] Cities unleash free Wi-Fi. http://www.wired.com/gadgets/wireless/news/2005/10/68999.

[38] Shugong Xu and T. Saadawi. Revealing TCP unfairness behavior in 802.11 based wireless multi-hop networks. In *Personal, Indoor and Mobile Radio Communication*, September 2001.

[39] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, and H. Zhang. Tesseract: A 4d network control plane. In *Proc. of NSDI*, Apr. 2007.

[40] Yaling Yang, Jun Wang, and Robin Kravets. Designing routing metrics for mesh networks. In *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, September 2005.

Vita

Yi Li was born in XianYou, FuJian, China. He received two B.E degrees from Tsinghua University in 1998. One is from the Department of Materials Science and Engineering, and the other one is from the Department of Computer Sciences. He received M.S degree in Computer Sciences from Peking University in 2001. Yi joined the Ph.D. program in the Department of Computer Sciences at the University of Texas at Austin in Fall 2001. He is an recipient of MCD fellowship during 2001-2005.

Permanent address: 79 YangQiao Middle Street, MinFuYuan, Bldg 1,
Room 201
Fuzhou, FuJian Province, P.R. China 350000

This dissertation was typeset with LaTeX[†] by Yi Li.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.