Predictable Performance Optimization for Wireless Networks

Yi Li[⋆] Lili Qiu[⋆] Yin Zhang[⋆] Ratul Mahajan[†] Eric Rozner[⋆]
University of Texas at Austin[⋆] Microsoft Research[†]
{ylee,lili,yzhang,erozner}@cs.utexas.edu ratul@microsoft.com

ABSTRACT

We present a novel approach to optimize the performance of IEEE 802.11-based multi-hop wireless networks. A unique feature of our approach is that it enables an accurate prediction of the resulting throughput of individual flows. At its heart lies a simple yet realistic model of the network that captures interference, traffic, and MAC-induced dependencies. Unless properly accounted for, these dependencies lead to unpredictable behaviors. For instance, we show that even a simple network of two links with one flow is vulnerable to severe performance degradation. We design algorithms that build on this model to optimize the network for fairness and throughput. Given traffic demands as input, these algorithms compute rates at which individual flows must send to meet the objective. Evaluation using a multi-hop wireless testbed as well as simulations show that our approach is very effective. When optimizing for fairness, our methods result in close to perfect fairness. When optimizing for throughput, they lead to 100-200% improvement for UDP traffic and 10-50% for TCP traffic.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.4 [**Performance of Systems**]: [Modeling techniques]

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Multi-hop wireless networks, Modeling, Optimization, Interference

1. INTRODUCTION

Wireless networks are becoming increasingly ubiquitous in the form of WLANs, city-wide meshes, and sensor networks. But extracting predictable performance from these networks today is notoriously hard. A single new flow can lead to a disproportionate decline in network performance. 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 optimization techniques for wireless networks that enable a prediction of the resulting performance at the level of individual flows. As motivating examples, consider three basic capabilities that are not available today, and to our knowledge, cannot even be approximated easily for real networks. First, network operators should be able to predict if the network can support the current or a planned traffic demand. Second, they should be able to perform "what if" analysis to predict the impact of configuration changes such as addition of new flows or routing changes. Finally, they should be able to predict safe sending rates of various flows based on policy and path capacity.

We show in the body of the paper that the final capability above is especially important for good performance. Without appropriate rate-limiting, network throughput can decline sharply when flows send more than what the path can support. This degradation can be severe even in a simple setting of a single flow traversing two links. End-to-end congestion control, *e.g.*, using TCP, while helpful, is not sufficient to prevent this pathological behavior.

In this paper, we propose a novel model-driven approach for optimizing wireless networks. We focus on static, 802.11-based, multi-hop networks, though we believe that the general methodology is applicable to other scenarios. The cornerstone of our approach is a new model that captures the complex interference, traffic, and MAC-induced dependencies in the network. These dependencies are the underlying cause of unpredictable behavior. Capturing them accurately and in a way that subsequently allows for optimization is the fundamental challenge in fulfilling our goal.

Despite much work on interference and MAC modeling, none of the existing models for multi-hop networks fulfills our need. Many existing models make simplifying assumptions about signal propagation [13], traffic [10, 30, 8, 27], topology [1, 18, 8, 10], or the MAC layer [14]. These assumptions often do not hold for real networks [17]. Many models also have high complexity and may require an exponential number of constraints [14] or states [27].

Our model strikes a balance between simplicity and realism. Based on easily collected measurements from the network itself, it characterizes the set of feasible network configurations and traffic assignments using very few constraints. Given n links that are actively sending traffic, our model has $O(n^2)$ complexity and only $O(n)$ constraints. Despite its simplicity, our model can handle real-world complexities such as hidden terminals, non-uniform traffic, multi-hop flows, and non-binary interference.

We then develop optimization algorithms that compute rate-limits for flows according to the specified performance objective. These algorithms take flow demands as input and use our model as a basic building block. The two performance objectives that we consider in this paper 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.

Evaluation using a multi-hop wireless testbed and simulation experiments shows that our approach is highly effective. Across a range of topology and traffic configurations, it is able to accurately approximate the throughput that the network yields. It rarely under-predicts, and for 80% of the cases, its estimate is within 20% of the actual throughput. 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 100-200% for UDP-based traffic and 10-50% for TCP-based traffic. Interestingly, we also find in our experiments that the exact choice of routing protocol is not important for good performance; what matters instead is that flows be rate-limited per the desired performance goal.

In summary, our work to predictably optimize wireless networks makes the following contributions.

- It shows that rate-limiting flows to levels that the network can safely support is critical for good performance; otherwise, network throughput can sharply degrade even in very simple settings (Section 2).
- We develop a novel approach to optimize multi-hop wireless networks (Section 3). Our approach includes a simple yet realistic model of network throughput under interference- and MAC-induced dependencies (Section 4). We design algorithms that use this model to optimize for fairness amongst flows and for throughput (Section 5).
- We evaluate our approach using extensive testbed and simulation based experiments (Sections 6–9). The evaluation shows that it can accurately predict network throughput, achieve close to perfect fairness, and substantially improve network throughput.

2. MOTIVATION

The goal of our work is to enable systematic optimization of multi-hop wireless networks, whose resulting performance can be predicted at the level of individual flows. We motivate this goal using examples of abilities that an operator may want but does not have today. These abilities are all pretty basic when it comes to managing networks and are available in wired networks today.

1. Determine whether the current or a planned traffic matrix can be supported by the network. This is an essential capability for network planning as it tells the operators when to add more resources, *e.g.*, additional radios on orthogonal channels or directional antennae to reduce interference.
2. Perform "what if" analysis on various configuration changes, to judge the impact of a change on the network. Such an analysis should be able to answer questions such as the following. What if a new flow is added between two nodes? What if a particular link or node is removed from the network? What if a particular routing change is implemented?
3. Compute and cap the sending rate of individual flows based on network polices and path capacity. This again is especially important in wireless networks. A flow that sends more impacts not only those flows that it shares links with (as is the case in wired networks), but also many other flows in the vicinity due to interference. More importantly, as we show below, a flow that sends more than what the path supports can cause a sharp decline in throughput. This decline is reminiscent of congestion collapse in wired networks.

As we discuss later, our optimization strategy relies on computing flow rates such that the specified performance objective is met. Here, we show that limiting flow rates is essential to obtaining good performance from the network. Without it, severe performance degradation can occur.

UDP traffic We illustrate this point using the two simple topologies in Figure 1(a). 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 experiments use 802.11a and 6 Mbps MAC bit rate throughout the paper.

Figure 1(b) 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 the 802.11 MAC allocates the medium fairly among *S* and *R* under saturated demands, 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 those from *R* 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.

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 the good-bad topology leads to a sharp decline in throughput as the sending rate is increased. Figure 1(c) plots the throughput of the two topologies as *S* increases its sending rate. The loss rate 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* can relay all received packets. Beyond it, *R* cannot keep up as the medium is increasingly occupied by the transmissions from *S* that are eventually dropped. The throughput stabilizes when the medium usage of *R* decreases to half.

The graph also shows that the two topologies have the same maximum capacity, 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 is caused by a single flow over a very simple topology. Known examples of congestion collapse in wired networks [6] involve more complex configurations. 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 2 confirms that this pathology can be replicated in the more realistic testbed setting as well. We emulate different loss rates in the testbed by changing the distance between the machines and varying layers of foil around the wireless cards. Figure 2(a) shows that the two topologies perform differently when *S* sends as fast as possible. Figure 2(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

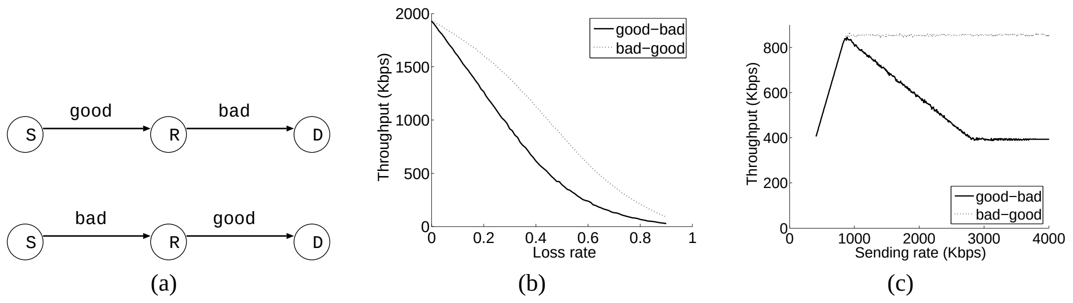


Figure 1: The importance of rate feedback. (a) **Two topologies that differ in where the lossy link occurs.** (b) **Throughput as a function of loss rate when** S **sends as fast as possible.** (c) **Throughput as a function of the sending rate when the loss rate of the bad link is 0.5.**

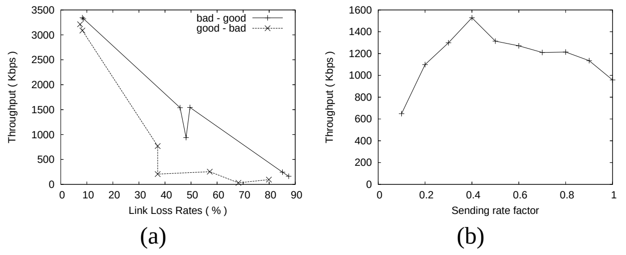


Figure 2: Testbed experiments confirm the importance of rate control. (a) Throughput vs. loss rate in the two topologies. (b) Throughput vs. sending rate in the good-bad topology when the loss rate on the bad link is 0.5.

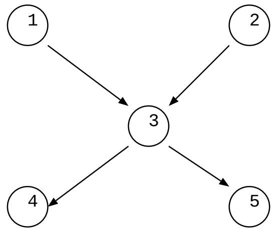


Figure 3: The topology for the TCP example.

sending rate. When the sending rate factor is 1, 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 controlling sending rates.

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, 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. 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 (see Section 5) 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 [7].

3. APPROACH

The previous section showed that dependencies introduced by interference and MAC can lead to significant performance problems in wireless networks. Thus, any optimization strategy must be able to model and predict these effects. We develop a model-driven optimization approach. We focus on IEEE 802.11-based networks, though our framework would be useful for other MAC protocols as well, which we plan to investigate in the future.

We need a model with the following two properties. First, it provides an accurate and compact characterization of the feasible solution space, which can then be incorporated in an optimization procedure to find a high-performance configurations. Second, it strikes a balance between fidelity, generality, and tractability, which is crucial for optimization to be effective and efficient.

We develop a novel model that satisfies both requirements. It has low complexity. Given n links that are actively sending traffic, our model has $O(n^2)$ complexity. It consists of $O(n)$ constraints that capture the inter-dependency between throughput, transmission probabilities, and packet loss rates of different links. These constraints collectively characterize the set of network configurations and traffic assignments that can be achieved in 802.11 networks. In contrast, many existing models are complex and may require an exponential number of state variables [27] or constraints [14] in the worst case. Despite its simplicity, our model is general and realistic. It is based on easily collected measurements from the underlying network and is thus more accurate than abstract models of RF propagation such as those based on distance. It also deals with real-world complexities such as hidden terminals, non-uniform traffic, multi-hop flows, and non-binary interference. In contrast, existing interference models often impose restrictive assumptions about signal propagation [13], traffic [10, 30, 8, 27], topology [1, 18, 8, 10], or the MAC layer [14]. These assumptions rarely hold for real networks [17].

Our model can be used in several ways for wireless network optimization. First, it can be used to answer "what-if" questions, such as testing whether desired end-to-end throughput can be achieved (see Section 5.1). It can also predict the performance under different network configurations, such as the impact of enabling or disabling RTS/CTS. Second, in conjunction with an efficient search mechanism, the model can be directly applied to optimize certain performance objectives when there are only a small number of control parameters. This is the case for fair rate allocation, which has a single optimization parameter (see Section 5.2). Third, the constraints of our model can be linearized and incorporated into an optimization procedure. This is the case for network throughput maximization, which has many parameters (see Section 5.3).

EP_i	Expected payload transmission time for Link i
T_{slot}	Regular slot time
T_i^{dat}	DATA duration on Link i from source to destination)
T_i^{ack}	ACK duration on Link i from destination to source)
L_i^{dat}	Inherent DATA loss rate on Link i from source to destination)
L_i^{ack}	Inherent ACK loss rate on Link i from destination to source)
D_{ij}^{src}	Probability for Link i's source to carrier sense Link j's source
D_{ij}^{dst}	Probability for Link i's source to carrier sense Link j's destination
S_{ij}	Synchronous collision loss probability for Link i due to Link j's transmission
A_{ij}	Asynchronous collision loss exponent for Link i due to Link j's transmission
W_{ij}	Expected waiting time for Link i when Link $j \neq i$ is transmitting. W_{ii} is the expected time for Link i to complete a transmission.
τ_i	Probability for Link i to transmit in a random variable-length slot (VLS)
p_i	Total packet loss rate on Link i.
μ_i	Expected VLS duration of Link i
g_i	Throughput of Link i
θ_i	Probability for Link i to start sending at a random slot time

Table 1: Model constants (upper case) and variables.

4. OUR MODEL

In this section, we develop a model for IEEE 802.11 that can be used for model-driven optimization. We describe our model in terms of network "links." Links are unidirectional, and each link has a unique source-destination pair of nodes.

4.1 Basic Model of 802.11 DCF

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 4.2.

4.1.1 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 [25, 23].

*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 Section 7, we use simulations and testbed experiments to show that our model is quite accurate despite these simplifications.

4.1.2 Constraints

Following Bianchi's approach [1], we 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 1 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:

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

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

$$\mu_i = T_{\text{slot}} + \sum_j \left[(W_{ij} - T_{\text{slot}}) \times \tau_j\right] \tag{2}$$

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 $\theta_j \triangleq \frac{\tau_j}{\mu_j}$ 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

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

  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 $\theta_j = \frac{\tau_j}{\mu_j}$:

$$p_i \;=\; 1 \;-\; (1-L_i^{\text{dat}})\times(1-L_i^{\text{ack}})\times \prod_{j\neq i}\Big[(1-S_{ij}\tau_j)\times(1-\theta_j)^{A_{ij}}\Big] \qquad (3)$$

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 [1, 27]

$$\tau_i \leq \frac{2}{2+CW(p_i)}, \qquad (4)$$

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.

4.2 Extensions to the Basic Model

We now extend the basic model above to support RTS/CTS, multi-hop flows, and TCP traffic. In the interest of space, we only present the key ideas.

RTS/CTS To support RTS/CTS, we make two modifications. First, in the VLS constraint (Eq. 2), 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. 3) 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 $\mathbf{x} = \langle x_d\rangle_{m\times1}$ 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 $\mathbf{g} = \langle g_i\rangle_{n\times1}$ be the vector of link loads. Then, we have

$$\mathbf{g} = R\cdot\mathbf{x} \qquad (5)$$

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. 5 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

$$R_{\text{TCP}} \triangleq R_{\text{fwd}} + \alpha\times R_{\text{rev}}, \qquad (6)$$

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 $\alpha = 0.5\times\frac{H}{H+EP}$.

4.3 Model Initialization

To apply our model, we need to initialize the constants in Table 1. The key constant are: (i) inherent loss rates L_i^{dat}, L_i^{ack}, L_i^{rts} and L_i^{cts}; (ii) carrier sense probabilities D_{ij}^{src} and D_{ij}^{dst}; and (iii) collision loss parameters S_{ij} and A_{ij}. For simplicity, we estimate these parameters by conducting pairwise broadcast measurements [25], but our model can just as easily use the inputs inferred by more scalable approaches [27].

1. We first let one Node a send alone. All the other nodes record the receiving rates from a. Dividing the receiving rates by a's sending rate yields the inherent loss rates for all links from a. ACK, RTS, and CTS are smaller than the smallest UDP packets. We approximate their inherent loss rate by broadcasting UDP packets with 1-byte payload.

2. We next have two nodes a and b send simultaneously. By comparing a's sending rates when both a and b are sending, we can estimate the probability for a to carrier sense b. Specifically, we can show that a's broadcast packet sending

rate when b is transmitting is given by

$$\frac{\tau_a}{T_{\text{slot}} + (T_a - T_{\text{slot}}) \times \tau_a + (T_b - T_{\text{slot}}) \times \tau_b \times D_{ab}},$$

where $\tau_a = \tau_b = \frac{2}{2+CW_{\min}}$ for saturated broadcast traffic, D_{ab} is the probability for a to carrier sense b, T_a and T_b are the packet transmission times including preamble and header. So we can easily compute the single unknown D_{ab} based on the measured sending rate of a.

3. To estimate S_{ij} and A_{ij}, we evaluate the conditional loss probability $C_{ij}(x)$ for different offsets x between the start times of the two packet transmissions on Links i and j. Based on carrier sense probabilities, we estimate the probability for a broadcast transmission on Link i to overlap with a broadcast transmission on Link j. We denote this overlapping probability by O^{bcast}_{ij} and compute it by applying the two-sender broadcast model of [27]. We denote the conditional loss probability for Link i when transmissions on the two links overlap by C^{bcast}_{ij}. It is computed based on

$$R^{\text{bcast}}_{ij} = (1 - L^{\text{bcast}}_i) \times (1 - O^{\text{bcast}}_{ij} \times C^{\text{bcast}}_{ij})$$

Above, L^{bcast}_i is the inherent loss rate on Link i, and R^{bcast}_{ij} is the broadcast receive rate for Link i when both links are sending. Finally, we compute $C_{ij}(x)$ by combining the collision loss rate for all steps of a packet transmission, *i.e.*, DATA, ACK, and if applicable, RTS, CTS.

5. MODEL-DRIVEN OPTIMIZATION

In this section, 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.

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 4 outlines the algorithm.

To estimate $\langle \tau_i \rangle$ given $\langle \theta_i \rangle$ (Line 4 in Figure 4), 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

$$\left\{ T_{\text{slot}} + \sum_j \left[(W_{ij} - T_{\text{slot}}) \times \tau_j \right] \right\} \times \theta_i = \tau_i, \quad i = 1, 2, \ldots, n \quad (7)$$

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) is violated. We bound the number of iterations to twenty, which works well in our experiments.

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

Figure 4: Link throughput feasibility testing.

```
▷ Input: routing matrix $R = [R_{id}]_{n \times m}$, end-to-end demand $\mathbf{x}^* = \langle x^*_d \rangle$ $(d \in [1, m])$
▷ Output: weighted max-min fair rate allocation: $\mathbf{x} = \langle x_d \rangle$
1.  initialization: $unsatSet = \{1, \ldots, m\}$, $x_d = 0$
2.  while ($unsatSet \neq \emptyset$)
        // try to scale up the unsaturated demands $\mathbf{x}^{\text{unsat}}$ as much as possible
3.      $x^{\text{unsat}}_d = \begin{cases} x^*_d & \text{if } d \in unsatSet \\ 0 & \text{otherwise} \end{cases}$    $(d = 1, \ldots, m)$
        // find largest $scale \in [0, 1]$ for $R(\mathbf{x} + scale \times \mathbf{x}^{\text{unsat}})$ to remain feasible
4.      $scale = \text{get\_max\_scaling\_factor}(R\mathbf{x}^{\text{unsat}}, R\mathbf{x})$
5.      $\mathbf{z} = \mathbf{x} + scale \times \mathbf{x}^{\text{unsat}}$
        // find the set of demands that become saturated
6.      if ($scale > 1 - \varepsilon$)      // $\varepsilon = 10^{-4}$ by default
7.          $\mathbf{x} = \mathbf{z}$; break      // all unsaturated demands can be satisfied
8.      end if
9.      for each $d \in unsatSet$
10.         $\mathbf{y} = \mathbf{z}$; $y_d = (1 + \varepsilon) \times y_d$
11.         $feasible = \text{test\_link\_throughput\_feasibility}(R\mathbf{y})$
12.         if (not $feasible$)
13.             $x_d = z_d$; $unsatSet = unsatSet - \{d\}$    // $d$ has become saturated
14.         end if
15.     end for
16. end while
17. return $\mathbf{x} = \langle x_d \rangle$
```

Figure 5: Algorithm for fair rate allocation

5.2 Fair Rate Allocation

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

Figure 5 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 get_max_scaling_factor($\mathbf{g}^{\text{unsat}}, \mathbf{g}^{\text{sat}}$) (Line 4 in Figure 5). The iteration continues to scale up the remaining unsaturated demands until all demands are saturated.

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

```
1.  initialization: $x_d^{(0)} = 0$, $\tau_i^{(0)} = 0$, for $\forall d \forall i$
2.  for $k = 1$ to $KMAX$
3.      let $\mathbf{x}^{\text{opt}}$ and $\tau^{\text{opt}}$ be the optimal solution to the linear program (LP$_k$)
4.      $\mathbf{x}^{(k)} = \mathbf{x}^{\text{opt}}$
5.      repeat    // ensure solution feasibility
6.          $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \alpha \times (\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$
7.          $feasible$ = test_link_throughput_feasibility($R\mathbf{x}^{(k)}$)
8.      until ($feasible$ = true)
9.      $\mathbf{x}^{(k)} = 0.99 \times \mathbf{x}^{(k)}$
10. end for
11. return $\mathbf{x}^{(k)}$
```

Figure 6: Algorithm for maximizing total throughput.

cast into the following non-linear optimization problem (NLP).

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

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).

We solve the NLP above through iterative linear programming, as shown in Figure 6. 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:

$$F_i^*(\tau) = F_i(\tau^{(k-1)}) + \sum_j (\tau_j - \tau_j^{(k-1)}) \times \frac{\partial}{\partial \tau_j} F_i(\tau^{(k-1)}) \qquad (8)$$

$$G_i^*(\tau) = G_i(\tau^{(k-1)}) + \sum_j (\tau_j - \tau_j^{(k-1)}) \times \frac{\partial}{\partial \tau_j} G_i(\tau^{(k-1)}) \qquad (9)$$

In the interest of brevity, we omit the details on how to compute all the partial derivatives above.

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

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

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 6).

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

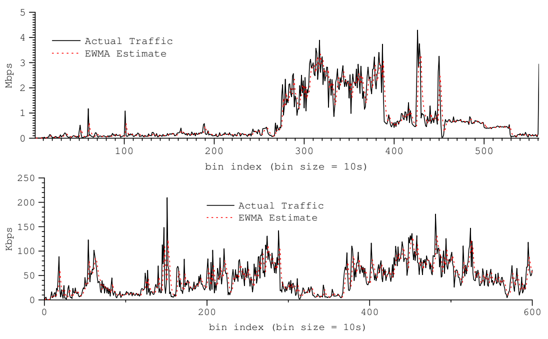


Figure 7: The amount of traffic sent to an AP in 10-second intervals. Top: At a WiFi hotspot. Bottom: At SIGCOMM 2004.

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 $x_d^{\text{init}} \le x_d^*$ 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 [34], 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.

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 [22]. Figure 7 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.

6. EVALUATION METHODOLOGY

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 and also lets us vary parameters such as topology that we cannot control for the testbed.

We divide our empirical results across three sections.

• In Section 7, we evaluate the accuracy of our model.

• In Section 8, we evaluate the degree to which our model can improve performance for both goals, maximizing fairness and maximizing total throughput. We quantify fairness using the classic Jain's fairness index, which is defined as $(\sum x_i)^2/(n * \sum x_i{}^2)$ for demands $x_1 ... x_n$.

• In Section 9, we show that it is rate-limiting that is critical to network performance. Exactly how the routes are chosen is less important.

In the first two sections, we use ETX as the routing protocol. ETX selects the path that minimizes the total number of expected transmissions from a source to its destination [3].

6.1 Strawman: Conflict Graph Model

We compare our approach to one based on the conflict graph (CG) model of interference[14]. 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.

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, $T_{d,e} \leq Cap_e \times z_{d,e}$, where $T_{d,e}$ is the amount of traffic successfully 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 maximizing the total throughput across all the flows and the portions of their demands that are achieved. This can be expressed as $\sum_d \sum_{r(e)=dest(d)} T_{d,e} + \lambda \alpha \sum_d x_d$, 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. In addition, we add the constraints to ensure that each flow receives throughput no less than αx_d (*i.e.*, for each Demand d, $\sum_{r(e)=dest(d)} T_{d,e} \geq \alpha x_d$). Our evaluation uses $\lambda = 100$ to significantly favor the solution with high fairness when maximizing fairness.

6.2 Simulation Experiments

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 generate traffic using both TCP and UDP and consider two types of application demands: (i) *saturated demands*, in which sources always have traffic to send; and (ii) *random demands*, in which the demand of a source is picked randomly from a uniform distribution 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 750x750 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 [3] and MIC [35]) are designed for wireless networks with lossy links, we extend the 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 [3].

6.3 Testbed Experiments

Our testbed consists of 19 nodes located inside an office building. Each 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. We use the lowest transmission power for our nodes to increase the network diameter. In this setting, we measured the diameter to be 7 hops, though routing paths may be longer. Other settings are consistent with the simulations.

The routing protocols are implemented using click [2]. We use *nuttcp* [24] to generate and measure UDP and TCP throughput. To rate limit flows, we let *nuttcp* generate application traffic at the specified limit. Without rate limit, each source generates application traffic as per its demand. We experiment with 1-16 flows.

As is common [3, 4, 35], we measure link quality using broadcast probes. Figure 8 shows CDF of link loss rate in our testbed. We prune links exceeding 90% loss rates in our route selection.

7. MODEL VALIDATION

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

Methodology 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

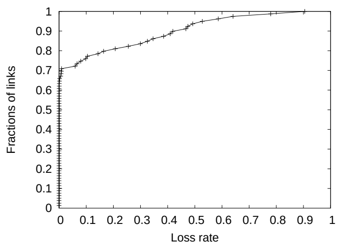


Figure 8: CDF of link loss rate in our testbed.

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 9 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 9).

7.1 Simulation Experiments

Figure 9 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%.

For the same configuration, Figure 10 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

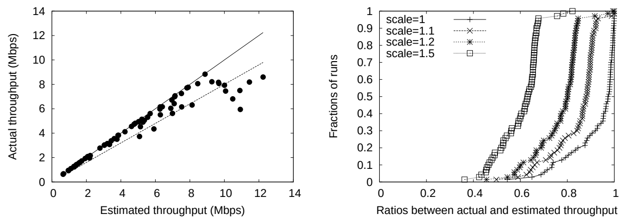


Figure 9: Throughput prediction accuracy in simulation of our model for grid topologies, saturated UDP traffic, and RTS/CTS = OFF.

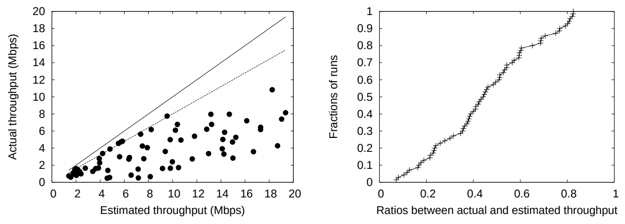


Figure 10: Throughput prediction accuracy in simulation of the CG-based model for grid topologies, saturated UDP demands, and without RTS/CTS.

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

Figure 11 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, 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.

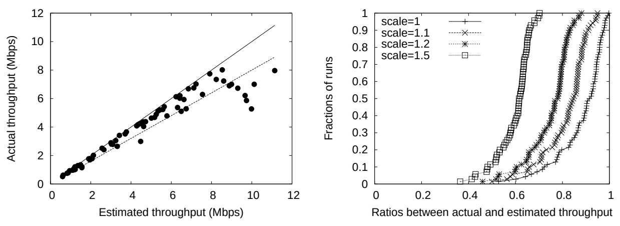
7.2 Testbed Experiments

Figure 12 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%.

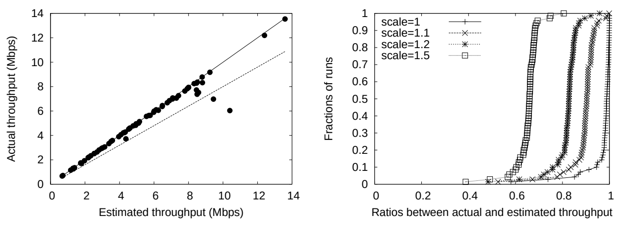
Figure 13 shows the throughput prediction accuracy using CG-model. We see that, as in simulation, the CG-model consistently over-estimates the achievable rates. Almost all the points are outside the cone formed by $y = x$ and $y = 0.8x$, which indicates that in most cases its estimated demands are not achievable within 80%.

8. PERFORMANCE OPTIMIZATION

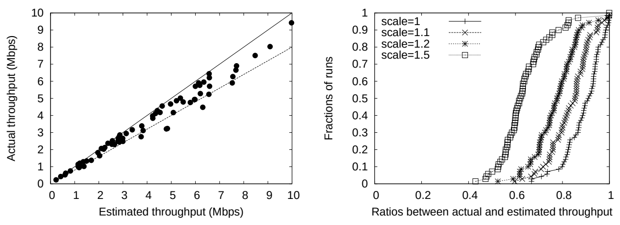
Can the accuracy of our model in predicting the throughout supported by the network be harnessed to improve performance, using the methods we outlined earlier? We answer this question in this section by first considering fairness maximization and then throughput maximization. We compare results with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



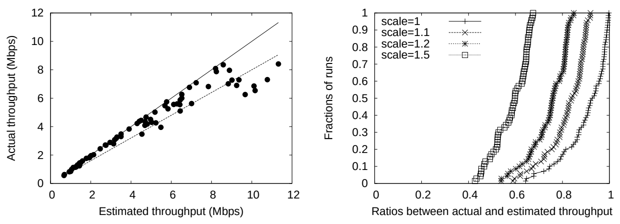
(a) *Grid topology,* **saturated TCP demands**, *without RTS/CTS*



(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 11: Throughput prediction accuracy in simulation of our model for various configurations. The difference from the base configuration in Figure 9 is in bold.

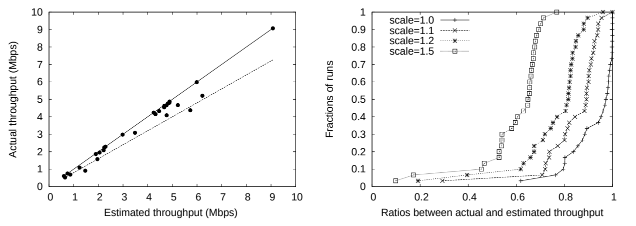
8.1 Maximizing Fairness

Figure 14 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, fairness degrades quickly as load increases. Even with the CG-based rate limiting, fairness is substantially lower than with our rate limiting.

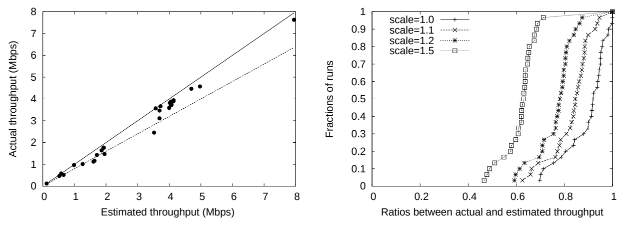
Figure 15 shows the fairness provided by our model-driven approach holds in a range of simulated configurations, for both TCP and UDP traffic, including grid and random topologies, with saturated or random demand models, and with and without RTS/CTS.

8.2 Maximizing Total Throughput

We next consider the performance objective of maximizing total throughput. Figure 16(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 100%

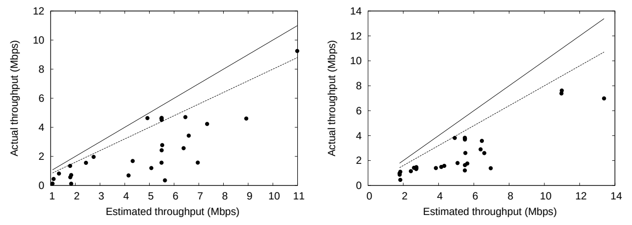


(a) *Saturated UDP demands*



(b) *Saturated TCP demands*

Figure 12: Throughput prediction accuracy of our model in our testbed. RTS/CTS=OFF.



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

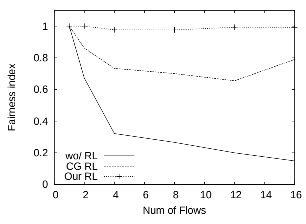
Figure 13: Throughput prediction accuracy in our testbed using CG-model for saturated demands and RTS/CTS = OFF.

improvement; in terms of normalized throughput, the average improvement ranges from 100% to 2400%. The larger improvement in the latter suggests that rate limiting is especially beneficial to the flows that experience low throughput under no rate limiting. Like our model, the CG-based model is able to identify interference-related bottlenecks and impose rate limits. Therefore it helps boost network throughput. However, because the CG-based model significantly over-predicts throughput (Section 7), the loss rate in the network is much higher and the throughput is consistently lower. Figure 16(b) shows the benefit of rate limiting extends to random UDP demands.

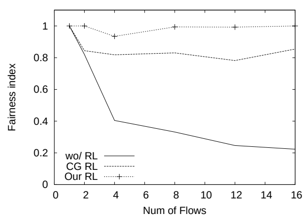
Figure 17 shows that the gain from rate limiting saturated and random TCP flows is a more modest 10-50%. This lower improvement for TCP is expected given that we experiment with infinitely long flows that react well to congestion, thus minimizing interference-related losses. However, we believe that 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 18 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.

Figure 19 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

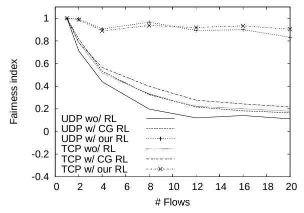


(a) *Saturated UDP traffic*

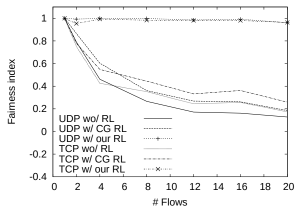


(b) *Saturated TCP traffic*

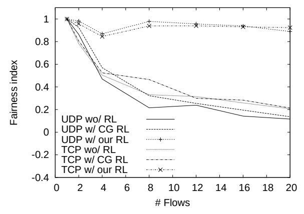
Figure 14: Fairness comparison in testbed. RTS/CTS=OFF.



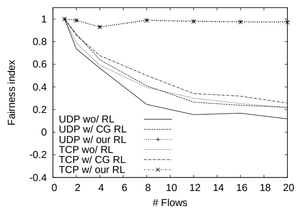
(a) *Grid topology, saturated demands, RTS/CTS=OFF*



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

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.

9. THE ROLE OF ROUTING

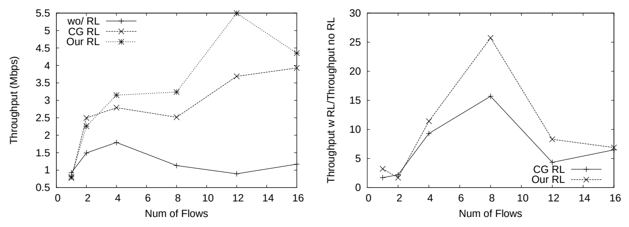
All the results above are based on routing paths chosen by the ETX protocol. In this section, we show that, surprisingly, the choice of the exact routing protocol makes little difference in our experiments. We study three other protocols and find that all four behave similarly. What seems to matter most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

The three other protocols that we study are the following.

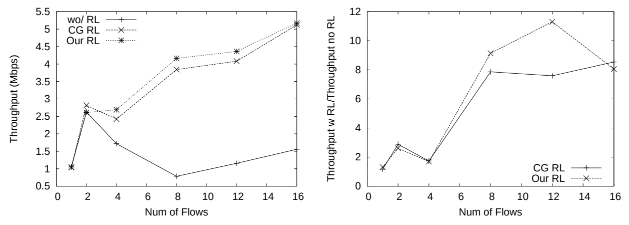
- *HOP* selects a path with minimum hop-count.
- *MIC* [35] scales ETX values of a link by multiplying it by the sum of the neighbors of the two end points. It then selects a path with the minimum scaled ETX value.
- *CG* selects the routes by casting the routing problem to a maximum flow problem augmented with interference constraints derived by a conflict graph [14]. 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 20 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

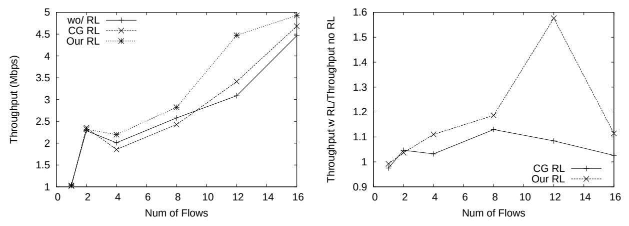


(a) *Saturated UDP demand*

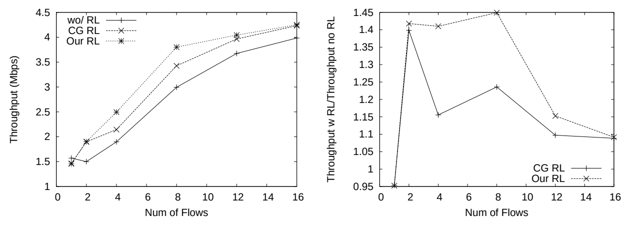


(b) *Random UDP demand*

Figure 16: UDP throughput improvement in our testbed with rate limiting.



(a) *Saturated TCP demand*



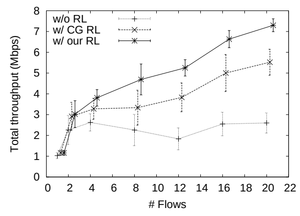
(b) *Random TCP demand*

Figure 17: TCP throughput improvement in our testbed with rate limiting.

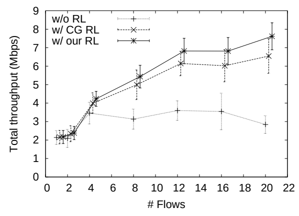
show the results using rate limiting based on our model, with the objective of maximizing total throughput. We see that the routing schemes are almost indistinguishable. Rate-limiting does matter, however. For each scheme, rate-limiting using our model provides 50-400% gain for UDP and 10-45% for TCP.

In Figure 21, 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 size and payload size, which may cause ETX to select the wrong path, we also considered probe-sized payload packets. As Figure 21(d) shows, that does not make a significant difference either.

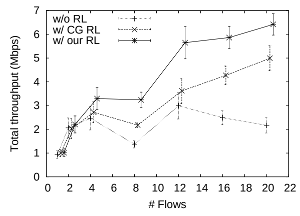
These routing protocols differ in how they account for interference, but they all have their shortcomings on that front (see our previous work [20] for more details). For example, the ETX metric is determined by packet loss rates at receivers, so it only captures receiver-side interference but fails 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,



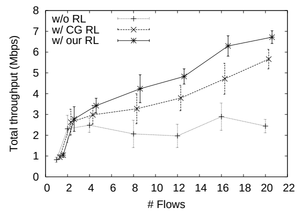
(a) *Grid topology, saturated UDP demand, RTS/CTS=OFF*



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

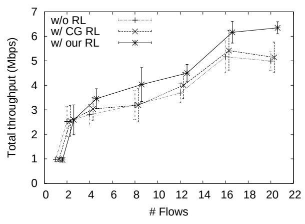


(c) *Grid topology,* **random UDP demand**, *RTS/CTS=OFF*

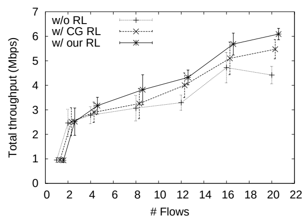


(d) *Grid topology, saturated UDP demand,* **RTS/CTS=ON**

Figure 18: Throughput improvement in simulation with rate limiting for UDP traffic for various configurations. The aspect that differs from the first configuration is in bold.

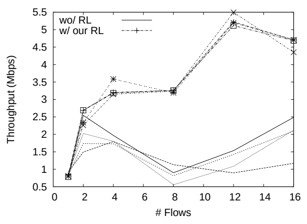


(a) *RTS/CTS=OFF*

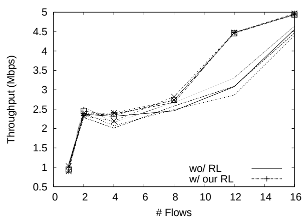


(b) *RTS/CTS=ON*

Figure 19: Throughput improvement in simulation with rate limiting for saturated TCP demand and grid topology.



(a) *Saturated UDP demand*

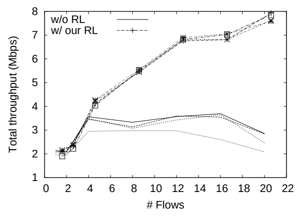


(b) *Saturated TCP demand*

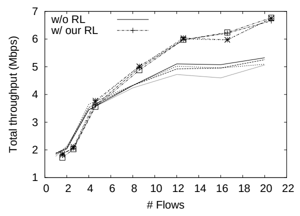
Figure 20: Throughput in our testbed of the four routing methods with and without rate-limiting. The top four lines in each graph are for the case of rate-limiting and the bottom four are for non-rate-limiting.

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.

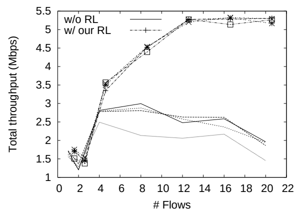
What we show is that once we have properly managed interference through rate-limiting, the small variations in routing paths produced by these protocols have relatively low impact on total network throughput. We also repeat that our methods for rate-limiting are agnostic to the choice of the underlying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



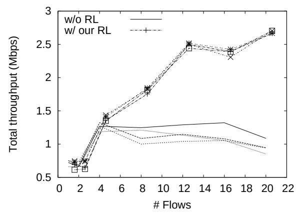
(a) *Random topology, saturated UDP demand, 1024-byte payload*



(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 21: 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.

can thus work with whichever routing method that provides better performance in the given setting.

10. RELATED WORK

Our work builds on a large body of prior work that we broadly classify into three categories: (i) interference modeling; (ii) rate control and scheduling; and (iii) routing.

Interference modeling There is a rich body of work on modeling wireless interference. One class of works focuses on asymptotic performance bounds. The seminal work by Gupta and Kumar analyzed the capacity of a wireless network under certain traffic patterns and topologies [13]. Other researchers have since extended this work to other traffic patterns [19], mobility [12], and network coding [11]. While these works lend useful insight into the performance of wireless networks in the limit, their models are abstract by necessity and cannot be used to model any specific real network.

The second class of works studies wireless performance for a given network topology and traffic demands but does not model the MAC and instead assumes that packet transmissions can be finely scheduled across links [14]. As we show in Section 7, these models significantly over-estimate the performance of 802.11 networks.

The third class of works model the performance of 802.11 DCF MAC. Most models in this class target scenarios where every node is within communication range of the others [1, 18, 8, 10] or where traffic demands are restricted (e.g., a single flow [10, 8] or two flows [30] or to a single neighbor [9]). In addition, all the above models, except [30], assume binary interference. [31] is one of the few that supports non-binary interference. But it requires detailed measurement of the current network condition, such as channel busy probability and packet loss rates (including collision rate). So it has limited prediction power – it can only estimate the effect of introducing one new flow to the network. Two recent models [27, 16] are more general but even they target only one-hop demands and are also too expensive to be used for optimization.

In contrast, we tackle the most general case of multi-hop topologies and end-to-end flows, but even so our model is lightweight enough to be directly used for optimization.

Rate control and scheduling The importance of rate control and scheduling has been well recognized. Some existing works [33, 21, 5] propose joint optimization of rate control and scheduling. Different from these works, our approach works with existing 802.11 MAC scheduling. IFRC enables fair rate control for sensor networks in which all nodes send traffic towards one or more sinks [29]. It is specific to the tree topologies and sensor network workload. Our prior work shows preliminary evidence on the importance of rate limiting and the possibility of using conflict graphs for that purpose [20]. In this paper, we develop a more accurate model so that we can perform predictable performance optimization.

Routing Most routing protocols for wireless networks follow a least-cost-path model but differ in the methods for estimating link cost. Some use hop count [26, 15], some use expected number of transmissions (ETX) [3], while others use ETX scaled by factors such as modulation or the number of neighbors [4, 35, 32]. These sequence of metrics are motivated by improving the performance of routing. We show, however, that rate-limiting flows is key to predictable and high performance – severe performance degradation can occur in its absence – and the differences in routing metrics appear to matter less.

11. CONCLUSION

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.

In the future, we plan to address several practical issues in order to apply the approach to operational wireless networks. First, we plan to develop novel measurement techniques to passively and accurately estimate interference and seed our model. Second, we plan to evaluate our approach under realistic traffic demands that change with time. Third, 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.

Acknowledgments This work is supported in part by NSF Grants CNS-0546755, CNS-0434515, CNS-0546720, and CNS-0627020.

12. REFERENCES

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

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

[3] 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.

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

[5] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length based scheduling and congestion control. In *Proc. of IEEE INFOCOM*, Mar. 2005.

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

[7] 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.

[8] 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.

[9] M. Garetto, T. Salonidis, , and E. Knightly. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. In *Proc. of IEEE Infocom*, March 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. Gastpar and M. Vetterli. On the capacity of wireless networks: the relay case. In *Proc. of IEEE INFOCOM*, Jun. 2002.

[12] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. In *Proc. of IEEE INFOCOM*, Apr. 2001.

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

[14] 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.

[15] 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.

[16] A. Kashyap, S. Das, and S. Ganguly. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *Proc. of ACM MOBICOM*, Sept. 2007.

[17] 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.

[18] 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.

[19] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Proc. of MOBICOM*, Jul. 2001.

[20] 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.

[21] 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.

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

[23] D. Niculescu. Interference map for 802.11 networks. In *Proc. of Internet Measurement Conference (IMC)*, Nov. 2007.

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

[25] 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.

[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] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-aware fair rate control in wireless sensor networks. In *ACM SIGCOMM*, Sept. 2006.

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

[31] T. Salonidis, M. Garetto, A. Saha, and E. Knightly. Identifying high throughput paths in 802.11 mesh networks: a model-based approach. In *ICNP*, Oct. 2007.

[32] A. P. Subramanian, M. M. Buddhikot, and S. Miller. Interference aware routing in multi-radio wireless mesh networks. In *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, Sept. 2006.

[33] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In *IEEE Transactions on Automatic Control*, Dec. 1992.

[34] 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.

[35] Y. Yang, J. Wang, and R. Kravets. Designing routing metrics for mesh networks. In *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, Sept. 2005.