Impact of Interference on Multi-hop Wireless Network Performance

Kamal Jain   Jitendra Padhye   Venkat Padmanabhan   Lili Qiu

Microsoft Research
One Microsoft Way, Redmond, WA 98052.

{kamalj, padhye, padmanab, liliq}@microsoft.com

ABSTRACT

In this paper, we address the following question: given a specific placement of wireless nodes in physical space and a specific traffic workload, what is the maximum throughput that can be supported by the resultant network? Unlike previous work that has focused on computing asymptotic performance bounds under assumptions of homogeneity or randomness in the the network topology and/or workload, we work with any given network and workload specified as inputs.

A key issue impacting performance is wireless interference between neighboring nodes. We model such interference using a *conflict graph*, and present methods for computing upper and lower bounds on the optimal throughput for the given network and workload. To compute these bounds, we assume that packet transmissions at the individual nodes can be finely controlled and carefully scheduled by an omniscient and omnipotent central entity, which is clearly unrealistic. Nevertheless, using ns-2 simulations, we show that the routes derived from our analysis often yield noticeably better throughput than the default shortest path routes even in the presence of uncoordinated packet transmissions and MAC contention. This suggests that there is opportunity for achieving throughput gains by employing an interference-aware routing protocol.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Wireless networks—*multi-hop, interference*

General Terms

Algorithm, Performance

Keywords

wireless, multi-hop, interference, performance

1. INTRODUCTION

Multi-hop wireless networks have been studied since the 70's [6]. Several new applications of such networks have recently emerged. Community wireless networks [1, 3] are multi-hop wireless networks that provide "last-mile" access to peoples' homes. This approach is an alternative to to cable modem and DSL technologies. In large networks of sensors [10] the scale and the environment are such that a multi-hop wireless network is the only feasible means of communication.

A fundamental issue in multi-hop wireless networks is that performance degrades sharply as the number of hops traversed increases. For example, in a network of nodes with identical and omnidirectional radio ranges, going from a single hop to 2 hops halves the throughput of a flow because wireless interference dictates that only one of the 2 hops can be active at a time.

The performance challenges of multi-hop networks have long been recognized and have led to a lot of research on the medium access control (MAC), routing, and transport layers of the networking stack. In recent years, there has also been a focus on the fundamental question of what the optimal capacity of a multi-hop wireless network is. The seminal paper by Gupta and Kumar [16] showed that in a network comprising of n identical nodes, each of which is communicating with another node, the throughput capacity per node is $\Theta(\frac{1}{\sqrt{n \log n}})$ assuming random node placement and communication pattern and $\Theta(\frac{1}{\sqrt{n}})$ assuming optimal node placement and communication pattern. Subsequent work has considered alternative models and settings, such as the presence of relay nodes and mobile nodes, and locality in inter-node communication, and their results are less pessimistic [13, 20, 12].

This paper also deals with the problem of computing the optimal throughput of a wireless network. However, a key distinction of our work from previous work such as [16] is that we work with any given wireless network configuration and workload specified as inputs. In other words, the node locations, ranges, etc. as well as the traffic matrix indicating which source nodes are communicating with which sink nodes are specified as the input. We make no assumptions about the homogeneity of nodes with regard to radio range or other characteristics, or regularity in communication pattern. This is in contrast to previous work that has focused on asymptotic bounds under assumptions such as node homogeneity and random communication patterns.

We use a *conflict graph* to model the effects of wireless interference. The conflict graph indicates which groups of links mutually interfere and hence cannot be active simultaneously. We formulate a multi-commodity flow problem [8], augmented with constraints derived from the conflict graph, to compute the optimal throughput that the wireless network can support between the sources and the sinks. We show that the problem of finding optimal throughput is NP-hard, and we present methods for computing upper and lower bounds on the optimal throughput.

We show how our methodology can accommodate a diversity of wireless network characteristics such as the availability of multiple non-overlapping channels, multiple radios per node, and directional antennae. We also show how multiple MAC protocol models as well as single-path and multi-path routing constraints can be accommodated.

We view the generality of our methodology and the conflict graph framework as a key contribution of our work.

To compute bounds on the optimal throughput, we assume that packet transmissions at the individual nodes can be finely controlled and carefully scheduled by an omniscient and omnipotent central entity. While this is clearly an unrealistic assumption, it gives us a best case bound against which to compare practical algorithms for routing, medium access control, and packet scheduling. Moreover, ns-2 simulations show that the routes derived from our analysis often yield noticeably better throughput than the default shortest path routes, even in the presence of real-world effects such as uncoordinated packet transmissions and MAC contention. In some cases, the throughput gain is over a factor of 2. The reason for this improvement is that in optimizing throughput, we tend to find routes that are less prone to wireless interference. For instance, a longer route along the periphery of the network may be picked instead of a shorter but more interference prone route through the middle of the network.

We use our technique to evaluate how the per-node throughput in a multi-hope wireless network varies as the number of nodes grows. Previous work (e.g., [16]) suggests that the per-node throughput falls as the number of nodes grows. But this result is under the assumption that nodes always have data to send and are ready to transmit as fast as their wireless connection will allow. In a realistic setting, however, sources tend to be bursty, so nodes will on average transmit at a slower rate than the speed of their wireless link. In such a setting, we find that the addition of new nodes can actually improve the per-node throughput because the richer connectivity provides increased opportunities for routing around interference "hotspots" in the network. This more than offsets the increase in traffic load caused by the new node.

The rest of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we present details of our conflict graph model and methods for computing bounds on the optimal network throughput. In Section 4, we present results obtained from applying our model to different network and workload configurations. Section 5 concludes the paper.

2. RELATED WORK

A number of papers have been published on the problem of estimating the capacity of a multi-hop wireless network. Here, we consider the work that is most closely related to ours.

In their seminal paper [16], Gupta and Kumar studied the capacity of wireless networks under two 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 capacity per node is $\Theta(\frac{1}{\sqrt{n \log n}})$ assuming random node placement and $\Theta(\frac{1}{\sqrt{n}})$ assuming optimal node placement and communication pattern. These results are shown under the protocol model, but the latter result also holds in the case of the physical model under reasonable assumptions. According to the intuitive explanation in [20], while the overall one-hop capacity of the network grows as $O(n)$, the average path length grows as $O(\sqrt{n})$, so the end-to-end throughput per node is $O(\frac{1}{\sqrt{n}})$.

Li *et al.* [20] have extended the work of Gupta and Kumar [16] by considering the impact of different traffic patterns on the scalability of per node capacity. They point out that a random traffic pattern represents the worst case from the viewpoint of per-node capacity. They also show that for traffic patterns with power law distance distributions, the per-node capacity stays roughly constant as the network size grows, provided the distance distribution decays more rapidly than the square of the distance. Li *et al.* also consider the interactions of packet forwarding with the 802.11 MAC and show that the use of 802.11 instead of a global scheduling scheme does not affect the asymptotic bound on per-node capacity derived in [16].

In [13], Grossglauser and Tse introduce mobility into the model presented in [16], and show that the average long-term throughput per source-destination pair can be kept constant even as the number of nodes per unit area increases, provided we allow for delays of the order of the time-scale of mobility. This is achieved by exploiting mobility to keep data transfers local, and transmitting only when the transmitter and receiver are close to each other, at a distance of $O(\frac{1}{\sqrt{n}})$, thereby reducing total resource usage and interference. While this is encouraging, in many practical situations (such as community wireless networks), mobility may be too infrequent (or even non-existent) to be exploitable.

Gastpar and Vetterli [12] extend the work of Gupta and Kumar [16] in a different direction. Instead of the simple point-to-point coding assumption made in [16] (which basically treats each transmitter-receiver pair as being independent of other pairs), they consider a *network coding* model where nodes could cooperate in arbitrary ways (for instance, to boost the transmit power). Further, they assume that there is a single source and single destination picked at random, and that the rest of the nodes act as relays. They show that the throughout capacity of the network under these conditions is $O(\log n)$ (compared to $O(1)$ for the point-to-point coding model of [16]). While the use of network coding in this context is a promising line of research, we note that the point-to-point coding model corresponds to current radio technology such as 802.11.

The recent work of De Couto *et al.* [9], based on two experiments in a 802.11b-based multi-hop wireless testbeds shows that minimizing the hop count of an end-to-end path is not sufficient for achieving good performance. The reason they point out is that link quality can vary widely and that the long hops that may be included in "short" paths may incur a high packet error rate. In our work, we also reach the same conclusion regarding the limitations of the hop count metric, but for a somewhat different reason — because of wireless interference limits capacity, a circuitous but less interference-prone route, say along the periphery of a network, may perform better than the shortest hop count route.

Finally, Yang and Vaidya [25] introduce the notion of a "conflict graph" in the context of their work on priority scheduling in wireless ad hoc networks. However, the conflict graph is defined on flows rather than links, and is used only to interpret experimental results showing that the 802.11 MAC causes flows with a high degree of conflict to suffer disproportionately compared to flows with a low degree of conflict. There is no attempt to analyze the conflict graph derive throughput bounds.

In summary, there is a large body of work on the wireless capacity problem, much of it focused on asymptotic bounds under assumptions such as node homogeneity and random communication patterns. In contrast, our work focuses on computing throughput bounds for a given wireless network and traffic workload, using a conflict graph to model the constraints imposed by wireless interference. We do not consider how factors such as mobility [13] or coding [12]. And like [16], we do *not* compute the information theoretic capacity of the network.

3. COMPUTING BOUNDS ON OPTIMAL THROUGHPUT

We now present our framework for incorporating the constraints imposed by interference in a multi-hop wireless network and then present methods for computing bounds on the optimal throughput that a give network can support for a given traffic workload. We begin with some background and terminology.

3.1 Background and Terminology

Consider a wireless network with N nodes arbitrarily located in a plane. Let $n_i, 1 \leq i \leq N$ denote the nodes, and d_{ij} denote the distance between nodes n_i and n_j. Each node, n_i, is equipped with a radio with communication range R_i and a potentially larger interference range R'_i. For ease of exposition, we assume that there is a single wireless channel. We consider two models, the *Protocol Model* and the *Physical Model* , to define the conditions for a successful wireless transmission. These models are similar to those introduced in [16].

Protocol Model: In the protocol model, if there is a single wireless channel, a transmission is successful if both of the following conditions are satisfied:

1. $d_{ij} \leq R_i$
2. Any node n_k, such that $d_{kj} \leq R'_k$, is not transmitting

Note that the second requirement implies that a node may not send and receive at the same time nor transmit to more than one other node at the same time. Note also that this model differs from the popular 802.11 MAC in an important way — it requires only the receiver to be free of interference, instead of requiring that both the sender and the receiver be free of interference. We discuss how to adapt the model for an 802.11-style MAC in Section 3.5.

Physical Model: Suppose node n_i wants to transmit to node n_j. We can calculate the signal strength, SS_{ij}, of n_i's transmission as received at n_j. The transmission is successful if $SNR_{ij} \geq SNR_{thresh}$, where SNR_{ij} denotes the signal-to-noise ratio at the node n_j for transmissions received from node n_i. The total noise, N_j, at n_j consists of the ambient noise, N_a, plus the interference due to other ongoing transmissions in the network. Note again that there is no requirement that the noise level at the sender also be low.

Our goal is to model wireless interference using a general framework that would then enable us to compute the optimum throughput that the wireless network can support for a given traffic workload. We assume that packet transmissions at the individual nodes can be finely controlled and carefully scheduled by an omniscient and omnipotent central entity. So here we do not concern ourselves with issues such as MAC contention that may be unavoidable in a distributed network; such real-world issues are considered in the ns-2 simulation results presented in Section 4. In our problem formulation here, we focus on maximizing the total throughput between source-destination pairs assuming that the sources always have data to send and the destination nodes are always ready to accept data. Our framework carries over easily to a different objective function such that one tries to maximize the minimum throughput across all source-destination pairs and thereby achieves a degree of fairness. Moreover, we can also accommodate the case where the rate at which nodes generate data or are willing to accept data is bounded simply by creating a *virtual* source or sink node and connecting it to the real source or sink via a *virtual link* of speed equal to the source or sink rate. The virtual link is special in that it is assumed not to interfere with any other link in the network. The virtual link is just convenient construct to help us model the bound on the source or sink rate.

The communication between the sources and destinations can be either direct or be routed via intermediate nodes. We say that a network throughput D is feasible if there exists a schedule of transmissions such that no two interfering links are active simultaneously, and the total throughput for the given source-destination pairs is D.

In the rest of this section, we consider the following three scenarios in detail: (i) multipath routing under the protocol interference model, (ii) multipath routing under the physical interference model, and (iii) single-path routing under both models. We end the section by discussing several other generalizations, and provide a summary description of our framework.

3.2 Multipath Routing under the Protocol Interference Model

Given a wireless network with N nodes, we first derive a *connectivity graph* C as follows. The vertices of C correspond to the wireless nodes (N_C) and the edges correspond to the wireless links (L_C) between the nodes. There is a directed link l_{ij} from node n_i to n_j if $d_{ij} \leq R_i$ and $i \neq j$. We use the terms "node" and "link" in reference to the connectivity graph while reserving the terms "vertex" and "edge" for the *conflict graph* presented in Section 3.2.1.

Let us first consider communication between a single source, n_s, and a single destination, n_d. In the absence of wireless interference (e.g., on a wired network), finding the maximum achievable flow between the source and the destination, given the flexibility of using multiple paths, can be formulated as a linear program corresponding to a max-flow problem, as shown in Figure 1. Here, f_{ij} denotes the amount of flow on link l_{ij}, Cap_{ij} denote the capacity of link l_{ij}, and L_C is a set of all links in the connectivity graph.

$$
\begin{array}{llllr}
\max & \sum_{l_{si} \in L_C} f_{si} & & & \\
\text{subject to} & & & & \\
\sum_{l_{ij} \in L_C} f_{ij} & = & \sum_{l_{ji} \in L_C} f_{ji} & \forall n_i \in N_C \setminus \{n_s, n_d\} & <1> \\
\sum_{l_{is} \in L_C} f_{is} & = & 0 & & <2> \\
\sum_{l_{di} \in L_C} f_{di} & = & 0 & & <3> \\
f_{ij} & \leq & Cap_{ij} & \forall l_{ij} \in L_C & <4> \\
f_{ij} & \geq & 0 & \forall l_{ij} \in L_C & <5>
\end{array}
$$

Figure 1: LP formulation to optimize the throughput for a single source-destination pair.

The maximization states that we wish to maximize the sum of flow out of the source. The first constraint represents flow-conservation, i.e., at every node, except the source and the destination, the amount of incoming flow is equal to the amount of outgoing flow. The second constraint states that the incoming flow to the source node is 0. The third constraint states that the outgoing flow from the destination node is 0. The fourth constraint indicates the amount of flow on a link cannot exceed the capacity of the link. The final constraint restricts the amount of flow on each link to be non-negative. The above formulation does not take into account wireless interference, which may impose additional constraints on when the individual wireless links may be active. We turn to this issue next.

3.2.1 Conflict Graph

To incorporate wireless interference into our problem formulation, we define a *conflict graph*, F, whose vertices correspond to the links in the connectivity graph, C. There is an edge between the vertices l_{ij} and l_{pq} in F if the links l_{ij} and l_{pq} may not be active simultaneously. Based on the protocol interference model described in Section 3.1, we draw such an edge if any of the following is true: $d_{iq} \leq R'_i$ or $d_{pj} \leq R'_p$. This encompasses the case where a conflict arises because links l_{ij} and l_{pq} have a node in common (i.e., $i == p$ or $i == q$ or $j == p$ or $j == q$). Note, however, that we do not draw an edge from a vertex to itself in the conflict graph.

3.2.2 Hardness Result

Next, we present a hardness result for computing the optimal throughput under the protocol interference model. We begin with a few definitions, Given a graph H (with vertex set V_H), an *independent set* is a set of vertices, such that there is no edge between any two of the vertices. The *independence number* of graph H is the size of the largest independent set in H. Then, we have the following hardness result.

THEOREM 1. *Given a network and a set of source and destination nodes, it is NP-hard to find the optimal throughput under the protocol interference model. Moreover, it is NP-hard to approximate the optimal throughput.*

It can be shown that the problem of finding the independence number of a graph (a known hard problem even to approximate) can be reduced to the optimal throughput problem. Moreover, this reduction is approximation preserving. Hence the above hardness result. We discuss the reduction briefly in the Appendix.

Since it is NP-hard to approximate the optimal throughput, we now look at heuristics for obtaining lower and upper bounds on the throughput. For this, we need to define some more terms. An independent set of a graph H can be characterized using an *independence vector*, which is a vector of size $|V_H|$. This vector is denoted by $\mathbf{x}_I$, where I is an independent set. The j^{th} element of this vector is set to 1 if and only if the vertex v_j is a member of the independent set I, otherwise it is zero. We can think of $\mathbf{x}_I$ as a point in an $|V_H|$-dimensional space. The polytope defined by convex combination of independence vectors is called the *independent set polytope* (also known as the *stable set polytope*).

3.2.3 Lower Bound

The problem of deriving a lower bound is equivalent to the problem of finding a network throughput D that has a feasible schedule to achieve it. We make the following observation. Links belonging to a given independent set in conflict graph F can be scheduled simultaneously. Suppose there are a total of K maximal independent sets in graph F. (A maximal independent set is one that cannot be grown further.) Let $I_1, I_2, \ldots I_K$ denote these maximal independent sets, and $\lambda_i, 0 \leq \lambda_i \leq 1$ denote the fraction of time allocated to the independent set I_i (i.e., the time during which the links in I_i can be active). If we add the schedule restrictions imposed by the independent sets to the original linear program (Figure 1), the resulting throughput always has a feasible schedule, and therefore constitutes a lower bound on the maximum achievable throughput.

We formalize our above observation as follows. Given a conflict graph F, we define a *usage vector*, U, of size $|V_F|$, where U_i denotes the fraction of time that the link i can be active. A usage vector is *schedulable* if the corresponding links can be scheduled, conflict free, for the fraction of the time indicated in the usage vector. If we think of the usage vector as a point in an $|V_F|$-dimensional space, we have the following theorem, the proof for which is not included due to space constraints.

THEOREM 2. *A usage vector is schedulable if and only if it lies within the independent set polytope of the conflict graph.*

Theorem 2 implies that the optimal network throughput problem is a linear program, no matter how many sender-receiver pairs we have. In fact, the problem is one of maximizing a linear objective function over a feasible polytope. This feasible polytope can be described as the intersection of two polytopes — the flow polytope and the independent set polytope of the conflict graph. The *flow polytope* is the collection of feasible points described by the flow constraints (Figure 1), ignoring wireless conflicts. The flow polytope is a simple structure on which a linear objective function can easily be optimized. Independent set polytope, on the other hand, is a difficult structure and no characterization of it is known (because there may be exponentially many independent sets).

Theorem 2 implies that any convex combination of independence vectors is schedulable. In general, however, an arbitrary point inside the independent set polytope will be a convex combination of an exponentially many independence vectors. To get around this computational problem, we only want to pick "easy" points in the independent set polytope. An obvious notion of "easy" is that the point picked should be a convex combination of a small number of (i.e., polynomially many) independence vectors. We will be using this notion explicitly in the algorithm as follows. We derive a lower bound on the optimal throughput by finding K' independence vectors in the conflict graph F, and adding the following constraints to the LP formulation shown in Figure 1.

- $\sum_{i=1}^{K'} \lambda_i \leq 1$ (because only one maximal independent set can be active at a time)
- $f_{ij} \leq \sum_{l_{ij} \in I_i} \lambda_i Cap_{ij}$ (because the fraction of time for which a link may be active is constrained by the sum of the activity periods of the independent sets it is a member of).

Note the solution produced by solving this linear program is always feasible (i.e., schedulable). This is due to the fact that all links belonging to independent set I_i can be simultaneously active for λ_i fraction of time, and we have required that the $\sum_{i=1}^{K'} \lambda_i \leq 1$. Moreover, theorem 2 assures us that when we include all independent sets, the solution will be exact, i.e., this will be the maximum value of D that is feasible. To help tighten the lower bound more quickly, we should consider using maximal independence sets. While finding *all* maximal independent sets is also NP-complete [11], the lower bound obtained by considering a subset of the (maximal) independent sets has the nice property that as we add more constraints, the bound becomes tighter, eventually converging to the optimal (i.e., the maximum feasible bound) when we add all the constraints.

3.2.4 Upper Bound

In this section, we derive an upper bound on the network throughput. Consider the conflict graph. A *clique* in the conflict graph is a set of vertices that mutually conflict with each other. Theorem 2 implies that the total usage of the links in a clique is at most 1. This gives us a constraint on the usage vector. We can find many cliques and write corresponding constraints to define a polytope. We can then maximize the throughput over the intersection of this polytope with flow polytope. This will give us an upper bound on the throughput.

Unfortunately, it is computationally expensive to find all the cliques, and even if we could find them all, there is still no guarantee that our upper bound will be tight. This can be illustrated by the following example. Suppose the conflict graph is the pentagon depicted in Figure 2. As we can see, the only cliques in the graph are formed by the adjacent pairs of nodes. Adding the clique constraints alone to the LP would suggest that a sum of link utilization equal to 2.5 is achievable. But actually at most 2 links can be active at a time. This suggests that we need to add constraints

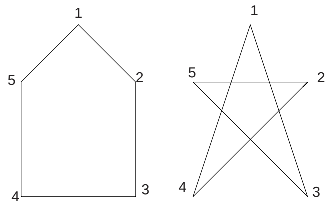

Figure 2: A pentagon and its complement graph. The former is an odd hole, and the latter is an odd anti-hole.

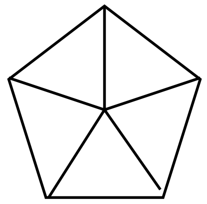
Figure 3: An example that shows it is not sufficient even if we add all clique, hole, anti-hole constraints.

corresponding to *odd holes* and *odd anti-holes.* An odd hole is a circle (i.e., loop) formed by an odd number of edges, without a chord in between. (The pentagon in Figure 2 is an odd hole.) The sum of the link utilization in an odd hole containing k vertices can be no more than $\lfloor \frac{k}{2} \rfloor$. An odd anti-hole is the complementary graph of an odd hole. Figure 2 shows an example of an anti-hole with 5 nodes. The sum of link utilizations in an odd anti-hole can be no more than 2.

Unfortunately, even if we consider the constraints imposed by the odd holes and odd anti-holes (in addition to those imposed by the cliques), we are not guaranteed to have a feasible solution. For example, consider the conflict graph, as shown in Figure 3. We can assign a utilization of 0.4 to all the vertices on the pentagon and 0.2 to the center of the pentagon, while satisfying all clique, hole, and anti-hole constraints. But there is no feasible schedule to achieve this, because this solution does not lie in the stable-set polytope. In fact, the upper bound based only on clique constraints is tight only for a special class of conflict graphs called perfect graphs. *Perfect graphs* are the graphs without any odd holes or odd anti-holes. Thus, in our present formulation, the upper bounds may not always be tight. We will discuss this further in Section 5.

3.3 Multipath Routing under the Physical Interference Model

As before, we begin by creating a connectivity graph C, whose vertices correspond to the nodes in the network. Based on the physical interference model, there exists a link, l_{ij}, from n_i to n_j if and only if $SS_{ij}/N_a \geq SNR_{thresh}$ (i.e., the SNR exceeds the threshold at least in the presence of just the ambient noise).

Using the connectivity graph, we can write an LP formulation to optimize network throughput for a wired network. As discussed before, the solution to the linear program, as shown in Figure 1, provides an upper bound on network throughput. However, this bound is not very useful since it does not take interference effects into account.

To take interference effects into account, we construct a conflict graph F. Unlike in the protocol model, conflicts in the physical model are not binary. Rather, the interference gradually increases as more neighboring nodes transmit, and becomes intolerable when the noise level reaches a threshold. This gradual increase in interference suggests that we should have a weighted conflict graph, where the weight of a directed edge from vertices l_{pq} to vertices l_{ij} (denoted by w_{ij}^{pq}) indicates what fraction of the maximum permissible noise at node n_j (for link l_{ij} to still be operational) is contributed by activity on link l_{pq} (i.e., node n_p's transmission to node n_q). Specifically, we have

$$w_{ij}^{pq} = \frac{SS_{pj}}{\frac{SS_{ij}}{SNR_{thresh}} - N_a}$$

where SS_{pj} and SS_{ij} denote the signal strength at node n_j of transmissions from nodes p and i, respectively, and $\frac{SS_{ij}}{SNR_{thresh}} - N_a$ is the maximum permissible interference noise at node n_j that would still allow successful reception of node n_i's transmissions. The edges of the conflict graph are directed, and in general w_{ij}^{pq} may not be equal to w_{pq}^{ij}.

3.3.1 Lower Bound

In the protocol model, we derive a lower bound on the network throughput by finding independent sets in the conflict graph F, and adding the constraints associated with the independent sets to the LP for the wired network. Analogous to independent sets, we introduce the notion of *schedulable sets* in the physical model. A schedulable set H_x is defined as a set of vertices such that for every vertex $l_{ij} \in H_x$, $\sum_{l_{pq} \in H_x} w_{ij}^{pq} \leq 1$. It follows that all links in a schedulable set can be active simultaneously. Suppose we schedule the links belonging to H_x for time $\lambda_x, 0 \leq \lambda_x \leq 1$. We now take the original LP for the wired network (in Figure 1), and include the following constraints:

- $\sum_{x=1}^{K'} \lambda_x \leq 1$, where K' is the number of schedulable sets found
- $f_{ij} \leq \sum_{l_{ij} \in H_x} \lambda_x Cap_{ij}$

To tighten the bound, we should consider using maximal *schedulable sets* in graph F (i.e., a schedulable set such that adding additional vertices to the set will violate the schedulable property). We have the following theorem, which is similar to the Theorem 2 in the protocol model.

THEOREM 3. *A usage vector is schedulable if and only if it lies in the schedulable set polytope of the conflict graph.*

3.3.2 Upper Bound

To derive an upper bound, we consider maximal sets of vertices in F such that for any pair of vertices l_{pq} and l_{ij}, $w_{ij}^{pq} \geq 1$. These correspond to the cliques in the protocol interference model. Therefore for each such set, we add a constraint that the sum of their utilization has to be no more than 1.

These constraints may result in a loose bound since there may not be very many cliques. To tighten the upper bound, we further augment the linear program with the following additional constraints. After we find a maximal schedulable set, say vertices v_1, v_2, ..., v_t, adding any additional vertex, denoted as v_a, to the set will make the set unschedulable. Therefore we have the following constraint: $U_1 + U_2 + ...U_t + U_a \leq t$, where as before U_i denotes the fraction of time for which physical link l_i (corresponding to vertex v_i in the conflict graph) is active. By adding as many such constraints as possible, we can tighten the upper bound. Still, the bound is not guaranteed to converge to the optimal even if we include all such sets.

3.4 Single-path Routing

So far we have considered multipath routing. As many existing routing algorithms [18, 24, 23, 22] are confined to single-path routing, it is useful to derive a throughput bound for single-path so that we can compare how much the current protocols deviate from the theoretical achievable throughput under the same routing restriction. The way we enforce the single-path restriction for the flow from a source to a destination is by adding the following additional constraints to the LP problem for the wired network (shown in Figure 1):

- For each link l_{ij}, $f_{ij} \leq Cap_{ij} \cdot z_{ij}$, where $z_{i,j} \in \{0, 1\}$
- At each node n_i, $\sum z_{ij} \leq 1$

Here z_{ij} is a 0–1 variable that indicates whether or not link l_{ij} is used for transmissions, and f_{ij} is the amount of flow on the link. The basic intuition for these constraints is that in a single-path routing, at any node in the network, there is at most one out-going edge that has a non-zero flow. The first and the third condition ensure that at node n_i at most one z_{ij} will have a value of 1.

In theory, solving integer linear program is a NP-hard [11], but in practice, programs such as lp_solve [7] and CPLEX [5] can solve mixed-integer problems.

3.5 Other Generalization

The basic conflict graph model is quite flexible, and can be generalized in many ways.

Multiple Source-Destination Pairs: We can extend our formulations in the previous sections from a single source-destination pair to multiple source-destination pairs using a multi-commodity flow formulation [8] augmented with constraints derived from the conflict graph. We assign a connection identifier to each source-destination pair. Instead of the flow variables f_{ij}, we introduce the variable f_{ijk} to denote the amount of flow for connection k on link l_{ij}. Referring to Figure 1, the flow conservation constraints at each node apply on a per-connection basis (constraint <1>); the total incoming flow into a source node is zero only for the connection(s) originating at that node (constraint <2>); likewise, the total outgoing flow from a sink node is zero only for the connection(s) terminating at that node (constraint <3>); and the capacity constraints apply to the sum of the flow levels of all connections traversing a link (constraint <4>).

This generalization is applicable to all variants of the problem.

Multiple wireless channels: It may be the case that instead of just one channel, each node can tune to one of M channels, $M \geq 1$. This can be easily modeled by introducing M links between nodes i and j, instead of just 1. In general, links corresponding to different channels do not conflict with each other, reflecting the fact that the channels do not mutually interfere. However, the links emanating from the same node do conflict, reflecting the constraint that the single radio at each node can transmit only on one channel at a time.

Multiple radios per node: Each wireless node may equipped with more than one radio. If each node has radios, this can be modeled by introducing M links betwe each pairs of nodes. If we assume that each of these radi is tuned to a separate channel, and that a node can comm nicate on multiple radios simultaneously, then the confli graph will show no conflict among the M links between pair of nodes.
Directional antennas: We can combine the use of dire tional antennas with the basic protocol model of communic tion. Instead of specifying a range for each node, we simp specify a list of nodes (or points in space) where transm sions or interference from this node can be perceived. T connectivity graph and the conflict graph are modified take this into account.
Other models of interference: In the simple examp we considered an optimistic model of interference that d not require the sender to be free of interference. But a mc realistic model, which more closely reflects the situation in 802.11, would require both the sender and the receiver to be free of interference. This reflect the fact that 802.11 may perform virtual carrier sensing using an RTS–CTS exchange, and that for successful communication, the sender must be able to hear the link layer acknowledgment transmitted by the receiver. Therefore, we draw an edge in the conflict graph between vertices l_{ij} and l_{pq} if $d_{ab} \leq R'_a$ for $ab = iq, qi, ip, pi, jp, pj, jq,$ or qj.

3.6 Summary

In this section, we presented the concept of a conflict graph, and discussed how it could be used to derive upper and lower bounds on the optimal throughput that a wireless network can support, for a given set of sources and destinations. We also generalize the conflict graph to handle a wide range of scenarios. We have shown that the lower bound derived from our framework is always schedulable, and will be optimal once all the independent sets are found. If the upper and lower bounds are equal, then these correspond to the optimal solution.

4. RESULTS

This section presents several results based on our model. The section is organized as follows. In Section 4.1, we present illustrative results that demonstrate the flexibility of our model. In Section 4.2, we use our model to provide insights into the tradeoff between the richer connectivity provided by increasing the size of a wireless mesh network, and the increase in cumulative traffic load due to the new mesh participants. In Section 4.3, we illustrate how optimal routing can bring benefits even in absence of optimal scheduling (i.e., in the presence of MAC contention and other inefficiencies). In Section 4.4, we discuss the issue of convergence of the upper and lower bounds on the optimal throughput. Finally, in Section 4.5, we present a discussion of the computational costs of our model.

4.1 Illustrative results

In this section, we present several illustrative results to demonstrate the capabilities of our model. We begin by defining a metric for computational effort. In Section 3, we have described the procedure for finding upper and lower bounds on throughput. Let us consider the protocol model of interference, and let us focus on the lower bound. We have shown that as we include more distinct independent sets, the lower bound becomes progressively tighter. In other words, the more *effort* we spend looking for independent sets in our conflict graph, the better the bound will be. Since we can not always hope to find optimal solutions, any upper or lower bounds discovered by our model need to be presented along with the amount of effort required to find those bounds. Thus we require a metric to measure this *effort*. We use the following simple algorithm to find distinct independent sets:

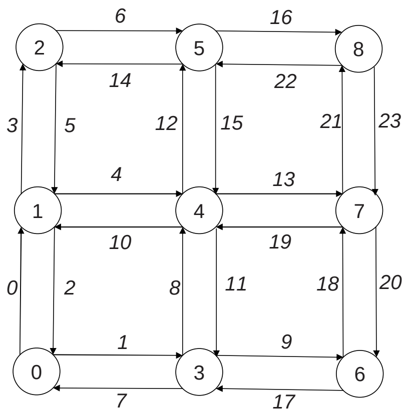

Figure 4: 3x3 Grid

1. Start with an empty independent set *IS*.

2. Consider a random ordering of vertices in the conflict graph.

3. Consider the vertices of the graph in that order. Always add the first vertex to *IS*.

4. Add a new vertex if and only if it does not have an edge to any of the vertices added to *IS* so far (i.e., if it does not conflict with any of the existing vertices). Once we consider all the vertices, *IS* will be of size at least one.

5. We check to see if we have previously discovered this independent set, and if not, we add constraints based on this independent set to our linear program. Otherwise we discard the set.

We consider this entire sequence as one unit of *effort*. Note that one unit of effort does not always result in addition of a constraint or variable to the linear program. Moreover, there is a complex relationship between the number of variables and constraints in a linear program, and the amount of time required to solve it. Thus, the metric is only a rough guide for amount of actual time (or CPU cycles) spent while finding the bound. In Section 4.5, we will provide further discussion about the relationship between the effort metric, and actual time spent in computation. The effort metric is defined in a similar manner by considering cliques in case of searching for the upper bound, and by considering schedulable sets in case of the physical model.

We consider the topology shown in Figure 4. The network consists of 9 nodes, placed in a 3x3 grid. We make no claims that this topology is representative of typical wireless networks. We have deliberately chosen a small, simple topology, to facilitate detailed discussion of the results.

link	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0
1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1	0	0	0
2	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0
3	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	1	0	0	1	0
4	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	0	0	1	0	0	1	0
6	1	0	1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	0	0	1	0	1	1	1
7	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1	0	1	1	1	1	0	0	0
8	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	0	1	0	0	1	1	0	1	1	1	1	0	1	0	1	1	1	1	1	0	1
10	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
14	1	0	1	1	1	1	1	0	1	0	1	1	1	1	0	1	1	0	0	1	0	1	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
16	0	0	0	1	1	1	1	0	1	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1
17	1	1	1	0	1	0	0	1	1	1	1	1	1	1	0	1	0	0	1	1	1	1	0	1
18	0	1	0	0	1	0	0	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
20	0	1	0	0	1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1
21	0	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
22	0	0	0	1	1	1	1	0	1	0	1	1	1	1	1	1	1	0	1	1	1	1	0	1
23	0	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 1: Conflict Graph in matrix form

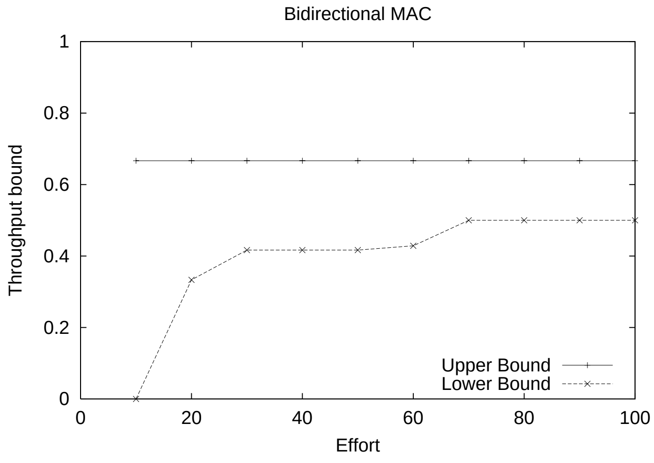

Figure 5: Throughput with a bidirectional MAC

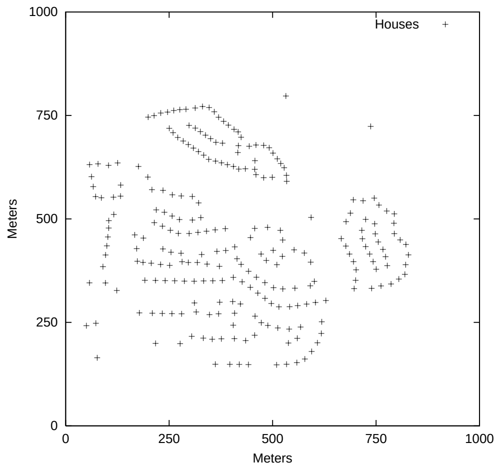

Figure 6: Neighborhood Map

We start with several simplifying assumptions. We will relax these assumptions as we proceed through the section. We assume that the range of each node is one unit, i.e., just enough to reach its lateral neighbors, but not the diagonal ones. We also assume that the interference range is equal to the communication range. We assume an 802.11-like protocol model of interference described in Section 3.5. This model requires both the sender and the receiver to be free of interference for successful communication. We term this a *bidirectional* MAC. The resulting conflict graph for this scenario is shown in the matrix form in Table 1. A 0 indicates that the links are not in conflict with each other, while 1 indicates otherwise. For example, when node 0 is transmitting to node 3, node 1 can hear these transmissions, and hence can not transmit to node 2. Thus, links 1 ($0 \rightarrow 1$) and 3 ($1 \rightarrow 2$) are in conflict.

We allow multipath routing. We assume that all wireless links have an identical capacity (i.e., speed) of 1 unit and that all nodes have infinite buffers. We designate node 0 to be the sender, and node 8 to be the receiver. The sender always has data to send, and the receiver is always willing to consume the data.

In this scenario, it is easy to see that the optimal throughput is 0.5. A convenient way to visualize the optimal transmission schedule is to imagine that time is divided into slots of equal size, and in each slot we can transmit one packet between neighboring nodes, subject to constraints imposed by the conflict graph. Then, the following transmission schedule will achieve optimal throughput: (i) $0 \rightarrow 1$ (ii) $1 \rightarrow 2$ (iii) $0 \rightarrow 3$ *and* $2 \rightarrow 5$ (iv) $3 \rightarrow 6$ *and* $5 \rightarrow 8$ (v) $0 \rightarrow 1$ *and* $6 \rightarrow 7$ (vi) ... We can continue in this manner indefinitely. It is easy to see that in alternate timeslots, node 0 gets to transmit to either node 1 or 3. Hence the optimal throughput is 0.5.

In Figure 5, we show the upper and lower bound on throughput calculated by our model, as we devote increasing amount of effort. As shown, the upper bound quickly converges to the stable value of 0.667, which is somewhat higher than the optimal value. This is a clear indication of the fact that clique constraints alone are not sufficient to guarantee optimality, even in such small graphs (as noted in Section 3.2.4). The lower bound, on the other hand, steadily converges to the optimal value of 0.5. We have verified that our program has discovered all independent sets and cliques with 100 units of efforts.

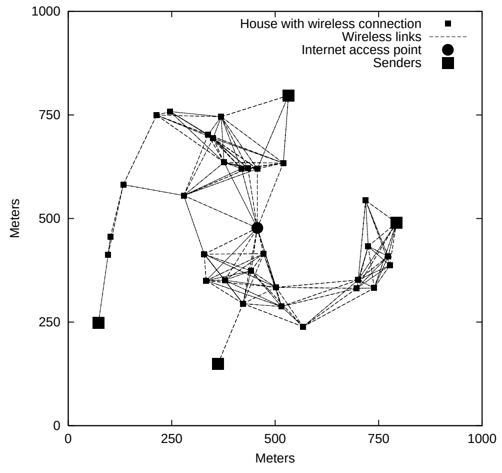

Figure 7: Mesh formation in the neighborhood

Our model can also incorporate single path routing, multiple source-destination pairs, multiple channels as well as multiple radios. We demonstrate this flexibility with a community mesh networking scenario, in which multiple users share an Internet connection, using a multi-hop wireless network. We consider a map of a real suburban neighborhood shown in Figure 4.1. There are 252 houses in an area of 1 square kilometer. We select 35 of these houses at random, and assume that these houses are equipped with hardware that enables them to participate in a wireless mesh network. We assume that communication range of the wireless technology is 200 meters, while the interference range is 400 meters. In Figure 4.1, we show the resulting network (i.e., the connectivity graph). We select a node that is roughly at the center of the area and designate it as the Internet access point. We assume that there are four senders, located as shown in the Figure. All the senders communicate with the Internet access point, and the metric of interest is the cumulative throughput of these senders. We assume that all wireless links are of unit capacity.

We begin with a baseline case, for which we assume a bidirectional MAC and single path routing. Our linear program is set to optimize the sum of the throughputs of the four flows, with no consideration for fairness. In this case, with about 5000 units of effort, upper and lower bounds converge, and our model indicates that the maximum possible cumulative throughput is 0.5.

We may now ask what we can do to improve the cumulative throughput. We consider four possibilities: (I) Employ multi-path routing (II) Double the range of each radio. We also double the interference range. (III) Leave the radio range unchanged, but use two non-overlapping channels instead of one. A node may communicate on only one of the two channels at any given time, but may switch between channels as often as necessary. (IV) Use two radios instead of one at each node. The radios are assumed to be tuned to two fixed, non-overlapping channels, so a node may communicate on the two channels simultaneously. The throughput bounds in each of the four scenarios are shown in Table 2. In each case, the upper and the lower bounds converge to the same value, which indicates that the solution is optimal.

Scenario	Optimal Throughput
I	0.5
II	0.5
III	1
IV	1

Table 2: Throughputs for neighborhood mesh in various scenarios

The results indicate that neither multipath routing nor doubling the range of the radio increases cumulative throughput in the scenario we considered. On the other hand, by using two channels instead of one, the network may achieve the maximum possible throughput of 1. The maximum possible throughput is 1 because the Internet access point has only one radio. On the other hand, even if we use two radios, the throughput remains at one. It is not hard to see why. The situation is equivalent to having two separate copies of the baseline network, and then adding up their throughputs. These scenarios illustrate that the model we have developed could be a useful, albeit rudimentary, tool for analysis and capacity planning in wireless multi-hop networks.

4.2 Tradeoff between connectivity and throughput

In Section 3, we discussed how our model can accommodate nodes which do not send data in a greedy fashion, i.e. they have a lower send rate. In [17, 20], the authors have shown that the per node capacity in the network decreases as the number of nodes in the network goes up. These results, however, were derived under the assumption that each node sends data as fast as it can. In other words, the desired sending rate of the node is assumed to be 1. However, if each node has a lower desired sending rate, the richer connectivity provided by additional nodes might help increase per node throughput, by allowing better routes to be discovered. We now explore this hypothesis using our model.

We consider a 7x7 grid, whose nodes are 200 meters apart horizontally, and vertically. We assume that the communication range is 250 meters, and the interference range is 500 meters. We set the link capacity to 1. We assume a bidirectional MAC, similar to the one used to plot Figure 5. We use single-path routing.

We pick N nodes from the 49 available nodes, at random, and without replacement. Half of these nodes are designated as senders, and the other half are designated as receivers. The senders and the receivers form $N/2$ flows in the network. Each sender is paired with only one receiver. We first calculate the fraction of flows for which the source and the destination lie in the same connected component of the topology. We call this fraction the *connectivity ratio*. The connectivity ratio for various values of N is shown in Figure 8. The results show that after 24 nodes (i.e. 12 flows) are selected, the connectivity ratio becomes 1.

We then assign a sending rate of D to each sender. Then, using our model, we calculate the optimal throughput using single-path routing. We divide the cumulative throughput by the number of flows (i.e. $N/2$) to obtain average per-flow throughput, and normalize it further by dividing it by D. The resulting normalized per-flow throughput for various values of N and D is plotted in Figure 9.

Note that when the sending rate is 0.01, the normalized

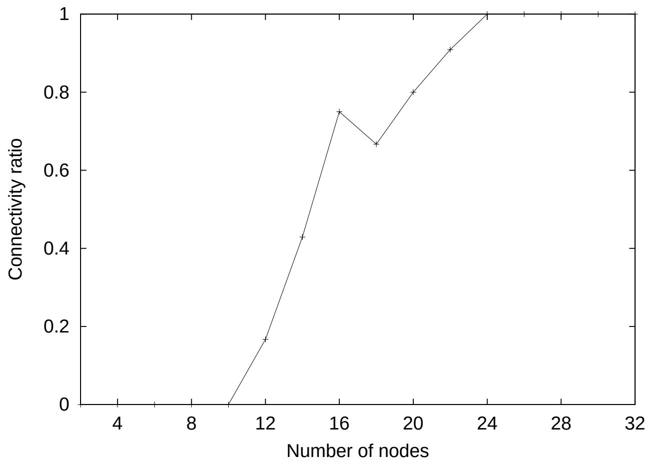

Figure 8: Connectivity Ratio for 7x7 grid

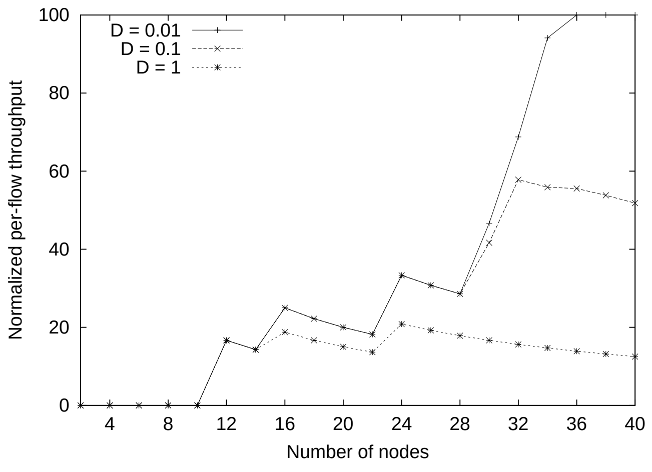

Figure 9: Normalized per-flow throughput

Scheduling / Routing	Optimal	802.11
Optimal	Optimal throughput (solve LP)	Alternative routing scheme (ns simulation)
Shortest-path	State of art under optimal scheduling (solve LP)	State of art (ns simulation)

Figure 10: Four scenarios.

per-flow throughput continues to rise even after the connectivity has reached 1. This means that the richer connectivity provided by additional nodes allows for newer routes, and allows extra traffic to be sent through the network. However, if each node sends at rate 1, the node might have little capacity left to forward traffic from other nodes. Thus, the average per-flow throughput peaks early (i.e the network is saturated), and then declines slowly, as new nodes join the network, but fail to transmit most of their desired traffic. For sending rate of 0.1, the results are between these two cases.

We stress that these results have been derived by assuming optimal routing, as well as optimal scheduling of packets. In the next section, we further discuss the impact of these two assumptions. We also note that we have used a simple grid topology in this example, and more experiments with complex topologies are needed to verify the generality of this result.

4.3 Benefits of optimal routing in absence of optimal scheduling

As shown in the previous sections, the optimal throughput is achieved by selecting optimal routes and scheduling the links on the routes appropriately. A natural question to ask is how much performance improvement is due to the optimal route selection, and how much is due to the optimal scheduling. Motivated by this question, we empirically examine four scenarios shown in Figure 10. They correspond to (i) optimal routing with optimal scheduling, (ii) shortest-path routing with optimal scheduling, (iii) optimal routing under 802.11 MAC, and (iv) shortest-path routing under 802.11 MAC. We first briefly describe the approach we use to derive throughput for each case, and then present the results.

Given a network topology, we apply the algorithm described in Section 3 to compute the optimal throughput under single-path routing. This corresponds to *scenario (i)*.

To derive the performance of optimal routing under 802.11, we run *ns-2* [4] simulations. To ensure that the packets follow the optimal routes, we specify the optimal routes obtained in Scenario (i) as the static routes in ns-2. The throughput numbers from these simulations correspond to *scenario (iii)*.

We then repeat our simulation using AODV [24], a standard shortest path routing protocol. The resulting throughput corresponds to the performance of the *scenario (iv)*.

To minimize the impact of AODV routing overhead, all nodes are static and simulations are run for 50 seconds, long enough to make the initial route setup overhead negligible.

Based on the AODV simulation results, we obtain a set of links that are used in the shortest paths between sources and destinations. We then modify the LP formulation (from Section 3) used to compute bounds on the optimal throughput by excluding all but those links that lie on one or more of the shortest paths. (We do so by setting the capacity of such links to zero.) We solve the resulting LP, and obtain the throughput for *scenario (ii)*.

Our aim is to compare throughput in scenario (i) to throughput in scenario (ii). Similarly, we compare scenarios (iii) and (iv) against each other. Note that we *do not* compare the throughput obtained by solving the LP model with the throughput obtained from ns-2 simulations.

We consider these four scenarios in a 7x7 grid (49 nodes). The horizontal and vertical separation between adjacent nodes is 200 meters. We assume the communication range to be 250 meters, and the interference range to be 500 meters. All other parameters are at their default settings in *ns-2*. For each simulation run, we randomly pick a few pairs of nodes as sources and destinations; the source sends packets to the corresponding destination at a constant bit rate equal to the wireless link capacity.

Table 3 and Table 4 show the throughput ratios between optimal routing and shortest path routing, under optimal scheduling (based on our LP model) and under the 802.11 MAC (based on ns-2 simulations), respectively. In all cases, optimal routing yields comparable or better throughput than the shortest path routing when optimal scheduling is used. The benefit of optimal routing varies with the number of flows, as well as with the locations of communicating nodes. For instance, when the two flows are far apart and do not interfere with each other, the optimal path achieves the same throughput as the shortest path (e.g., numFlow=2 and run=1, 5); when the two flows interfere with each other, the optimal path takes a detour, which results in reduced interference and hence higher throughput (e.g., the case of numFlow=2 and run= 2, 3, 4). For similar reasons, the optimal path outperforms the shortest path even under the 802.11 MAC (as evident from the numbers for the corresponding runs in Table 4).

On the other hand, the optimal path routing does not always outperform the shortest path routing under 802.11 MAC; this is increasingly common when the number of flows is higher. This occurs because as network load increases, it is harder to find paths that do not interfere with other flows in the absence of optimal scheduling.

The above results are encouraging, and suggest that there is a potential to improve throughput by making route selection interference-aware. In ongoing work, we are continuing to investigate the benefits of interference-aware routing under a wider range of scenarios.

4.4 Convergence of upper and lower bounds

In most of the previous results in this section, the upper and the lower bounds converged, assuring us of the optimality of the solution. When they did not converge, e.g., Figure 5, we were able to assure ourselves of optimality of the lower bound by manual verification. In general, however, the bounds may not converge, as there is no guarantee that even after adding all the clique constraints the upper

numFlow	run 1	run 2	run 3	run4	run 5
2	1.00	1.25	1.6	1.3812	1.00
4	1.4092	1.0001	1.4369	1.4333	1.1429
8	2.1023	1.0000	1.0451	1.1131	1.1194

Table 3: Throughput ratios between optimal routing and shortest path routing, both under optimal scheduling in a 7x7 grid.

numFlow	run 1	run 2	run 3	run4	run 5
2	1.0768	2.4323	1.5278	1.7974	1.1863
4	1.0662	1.5374	0.7854	1.0181	1.5475
8	3.5474	1.2239	0.5011	1.1389	0.3905

Table 4: Throughput ratios between "optimal" path routing and shortest path routing, both under 802.11 MAC in a 7x7 grid.

bound will be *schedulable*. This leads to the question: how do we decide when to stop looking for even tighter bounds? Given that the conflict graph may have an arbitrarily complex structure, we cannot wait until the upper and lower bounds are within a small percentage of each other for this may never happen. Even after all the cliques are found, the upper bound may stay well above the optimal feasible solution. Thus, there is no easy way to decide when to stop the calculations. The data we present next does indicate, however, that convergence is quite good in many scenarios.

4.5 Computational Costs

In Section 4.1, we mentioned that the *effort* metric provides only a rough indication of the computational costs of finding the bounds. We now provide more data in this regard. Note that much of the data provided is for the MATLAB [2] solver to which we had ready access; as noted below, *the CPLEX [5] solver offered a a speedup of about 7X*, albeit on a somewhat faster CPU. Unfortunately, we only had limited access to the CPLEX resource. So there is the potential for significant improvements over the computational costs reported here.

In Table 5, we consider the relationship between the size of the network and the amount of time required to compute upper and lower bounds. The table shows the bounds computed after 150,000 units of efforts for several grid sizes, and the time required to compute them. In each case, there is a single flow in the network, with its source and destination nodes at diagonally opposite corners of the grid. The rest of the parameters are similar to those used to plot Figure 5. Note that the upper and lower bounds are not equal in all cases, which indicates that we might not have found the

Grid Size	Upper Bound	Lower Bound	Time (minutes)
3x3	0.25	0.25	2
5x5	0.5	0.5	2
7x7	0.495	0.5	25
9x9	0.474	0.5	35
11x11	0.479	0.5	40

Table 5: Upper and lower bounds after 150,000 units of effort

Effort	Upper Bound	Lower Bound	Time (minutes)
10000	0.443	0.5	2
50000	0.48	0.5	5
100000	0.49	0.5	13
150000	0.495	0.5	25
200000	0.5	0.5	41

Table 6: Upper and lower bounds after varying effort for a 7x7 grid

Flows	Upper Bound	Lower Bound	Time (minutes)
2	0.578	0.583	34
3	0.707	0.75	31
4	0.758	0.833	29
5	0.799	0.875	31
6	0.849	0.925	34
7	0.861	1.00	36

Table 7: 7x7 grid, multiple flows, 150,000 units of effort

optimal solution in all cases. The computations were done using MATLAB 6.1 [2], on a machine with 1.7Ghz Pentimum processor, and 1.7GB of RAM.

In Table 6, we consider the relationship between the amount of *effort*, and the closeness of upper and lower bounds, as well as the time required to compute those bounds. The results are based on the 7x7 grid, with rest of the parameters similar to those used for Table 5. As we discussed in Section 4.1, with more effort, we are likely to add more variables as well as more restrictive constraints in the linear program. So the bounds become tighter.

In Table 7, we consider the relationship between the number of flows in the network, and the amount of time required to compute bounds for a given amount of effort. The results are based on a 7x7 grid, with multiple flows. For each flow, the source is in the bottom row of the grid, and it communicates with a destination located in the same column, but in the top row. All other parameters are same as Table 5.

The software used to solve the linear program is also a significant factor in the amount of time required to find the optimal solution. In Table 8, we show the amount of time taken by CPLEX [5] to solve the 7x7 grid case, with 6 and 7 flows on a 2.7GHz Pentium machine, with 3.7GB of RAM. While we can not compare these entries directly with the corresponding entries in Table 8, as the machines used to run MATLAB and CPLEX are different, the speedup is still quite significant: a reduction by a factor of 7, from 34-36 minutes down to 5 minutes. Moreover, MATLAB cannot solve the Mixed Integer Programs that result when single-path routing is used. We could only solve these using CPLEX. Unfortunately, we only had limited access to the CPLEX software, so we are unable to report the full set of numbers for CPLEX[1].

Flows	Upper Bound	Lower Bound	Time (minutes)
6	0.849	0.925	5
7	0.861	1.00	5

Table 8: 7x7 grid, multiple flows, 150,000 units of effort, with CPLEX

Since these numbers are based on a single run, and are based only on grid graphs, which have a regular connectivity pattern, we cannot draw general conclusions from them. However, some trends are useful to note. We observe that for grid networks, the amount of time required to solve the problem increases with the number of nodes. We also see that for a given effort level, the time required to compute the bounds does not depend significantly on the number of flows in the network. However, the difference between the upper and lower bounds for a given amount of effort tends to increase with increase in number of flows.

In case of irregular graphs, such as the neighborhood graph shown in Figure 4.1, we have observed that the amount of time required to solve the depends significantly on connectivity and interference patterns.

Finally, we note that we have not included any results involving physical model of communication in this section. We have also not included results that demonstrate the use of links of different capacities. While we have solved such networks (physical models of interference, links of different capacities etc.), we could not do a detailed study due to resource constraints. Therefore, we have chosen to focus on the protocol model of interference in this paper.

5. CONCLUSION AND FUTURE WORK

In this paper we have presented a model and methodology for computing bounds on the optimal throughput that can be supported by a multi-hop wireless network. A key distinction compared to previous work is that we work with any given wireless network configuration and workload specified as inputs. No assumptions are made on the homogeneity of nodes with regard to radio range or other characteristics, or regularity in communication pattern. We use a *conflict graph* to model wireless interference under various conditions (multiple radios, multiple channels, etc.). We view the generality of our methodology and the conflict graph framework as a key contribution of our work.

Although the bounds that we compute on the optimal throughput assume the ability to finely control and carefully schedule packet transmissions, the optimal routes yielded by our analysis often outperform shortest path routes even under "real-world" conditions such as uncoordinated scheduling and MAC contention. In ns-2 simulations, we have observed a throughput improvement of over a factor of 2 in some cases. The reason for this significant improvement is that the optimal routes often tend to be less interference-prone than the default shortest path routes.

We have also considered the impact of new nodes on the per-node throughput in multi-hop wireless networks. Contrary to previous results, we have found that the addition of new nodes can be beneficial for all nodes, under the (realistic) assumption that each node is active for only a small fraction of the time. The richer connectivity (and hence increased opportunities for routing around interference "hotspots" in the network) contributed by new nodes more than offset the increase in traffic load they cause.

In ongoing work, we are continuing to investigate the benefits of interference-aware routing under a wide range of scenarios. Our next step after that would be to design a prac-

[1]Although we expect to be able to do so for the final version of this paper should it be accepted for publication.

tical interference-aware routing protocol, which presents interesting challenges such as constructing the conflict graph and computing optimal routes in a distributed manner.

6. REFERENCES

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

[2] Matlab version 6.1. http://www.matlab.com/.

[3] Seattle wireless. http://www.seattlewireless.net/.

[4] Ns (network simulator), 1995. URL http://www-mash.cs.berkeley.edu/ns/.

[5] Ilog cplex suite, 2003. URL http://www.ilog.com/products/cplex/.

[6] Abramson, N. The aloha system — another alternative for computer communications. *Proc. AFIPS Fall Joint Computer Conference* (1970), 281–285.

[7] Berkelaar, M. lp_solve: linear programming code. ftp://ftp.ics.ele.tue.nl/pub/lp_solve/.

[8] ChavtaL, V. *Linear Programming.* W. H. Freeman and Compnay, 1983.

[9] Couto, D. S. J. D., Aguayo, D., Chambers, B. A., and Morris, R. Performance of multihop wireless networks: Shortest path is not enough. In *1st Workshop on Hot Topics in Networks* (Oct. 2002).

[10] Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. Next century challenges: Scalable coordination in sensor networks. In *ACM MOBICOM* (Aug. 1999).

[11] Garey, M. R., and Johnson, D. S. *Computers and Intracctability: A guide to the theory of NP completness.* W. H. Freeman and Company, 1979.

[12] Gastpar, M., and Vetterli, M. On the capacity of wireless networks: the relay case. In *IEEE INFOCOM* (Jun. 2002).

[13] Grossglauser, M., and Tse, D. Mobility increases the capacity of ad-hoc wireless networks. In *IEEE INFOCOM* (Apr. 2001).

[14] Grotschel, M., Lovasz, L., and Schrijver, A. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1* (1981), 169–197.

[15] Grotschel, M., Lovasz, L., and Schrijver, A. *Geometric Methods in Combinatorial Optimization.* Academic Press, 1984.

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

[17] Gupta, P., and Kumar, P. R. The capacity of wireless networks. *IEEE Transactions on Information Theory 46*, 2 (Mar. 2000), 388–404.

[18] Johnson, D. B., and Maltz, D. A. Dynamic source routing in ad-hoc wireless networks. In *Mobile Computing* (1996), T. Imielinski and H. Korth, Eds., Kluwer Academic Publishers.

[19] Khachiyan, L. G. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady 20* (1979), 191–194.

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

[21] M.Chudnovsky, Robertson, N., P.D.Seymour, and R.Thomas. The Strong Perfect Graph Theorem. Submitted for publication. URL http://www.math.gatech.edu/~thomas/spgc.html, February 2003.

[22] Park, V. D., and Corson, M. S. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. of IEEE INFOCOM'97* (Apr. 1997).

[23] Perkins, C. E., and Bhagwat, P. Highly dynamic destination-sequenced distance vector routing (dsdv) for mobile computeres. In *Proc. of ACM SIGCOMM'94* (Sep. 1994).

[24] Perkins, C. E., and Royer, E. M. Ad-hoc on-demand distance vector routing. In *Proc. of IEEE WMCSA'99* (Feb. 1999).

[25] Yang, X., and Vaidya, N. H. Priority scheduling in wireless ad hoc networks. In *ACM MobiHoc* (June 2002).

APPENDIX

.1 Proof of Theorem 1

Suppose we are given a graph G and we want to compute the cardinality of its maximum independent set. We now construct a wireless network such that the optimal throughput it can support under the protocol interference model is same as the cardinality of the maximum independent set of G. Create two wireless nodes, a source s and a receiver r. For every vertex in G add a wireless link of unit capacity between s and r. For every edge between two nodes in G, assume a conflict between the corresponding wireless links in the network. (Such a network may arise, for instance, if nodes s and r are each equipped with multiple radios set either to the same (i.e., interfering) channel or to separate (i.e., non-interfering) channels.) It is not hard to see that that the optimal throughput is achieved if and only if a maximum independent set in G is scheduled. Thus finding the optimal capacity of the wireless network is equivalent to to finding the cardinality of the maximum independent set of graph G, which is known to be a hard problem.

The above proof may come across as contrived since the wireless network we constructed is unlikely to arise in practice. This raises an interesting question of whether realistic wireless networks could give rise to complex conflict graphs? Our answer is both yes and no. Our answer is "yes" because the maximum independent set problem is hard due to the existence of odd holes and odd anti-holes in the given graph[2]. As shown in Figure 11, very realistic and simple grid graphs could have conflict graphs with many odd holes and odd anti-holes. On the other hand, our answer is "no" because realistic conflict graphs may have some special property or structure that could make the problem of finding the maximum independent set easy. We have been unable to identify any such property, but our failure does not mean that no such property exists (though the complex conflict graphs arising from the simple grid graphs, as in Figure 11, diminish our optimism). In view of this, we believe that the heuristic approach presented in Section 3 may be reasonable.

[2]If a graph does not have any odd holes or anti-holes then the graph is termed *perfect* [21], and for perfect graphs there are polynomial time algorithms to solve the maximum independent set problem [14].

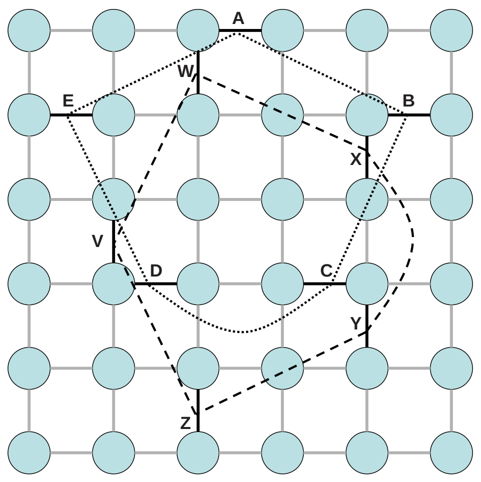

Figure 11: A 6x6 grid connectivity graph. ABCDE and VWXYZ are examples of odd holes in the corresponding conflict graph, assuming an 802.11-style MAC, communication range equal to the lateral spacing between neighbors, and interference range equal to twice the communication range. These odd holes also happen to be odd anti-holes.

.2 Polynomial Time Algorithm in Special Case

Even in special cases where polynomial time algorithms may exist, they may be too expensive to be of practical interest. One such special case arises in the context of grid graphs when the conflict radius is zero. By zero conflict radius we mean that two links conflict if and only if they share an endpoint. In this simple and somewhat unrealistic setting, the conflict graph is nothing but the *line graph* of the underlying grid network. (The line graph, $L(G)$, of a graph, G, is a graph on the edges of G, i.e., the vertices of $L(G)$ correspond to the edges of G. There is an edge between two vertices of $L(G)$ if the corresponding edges in G have a vertex in common.) Our network in this case is a grid. A grid is a bipartite graph, and bipartite graphs are perfect. The line graph of a perfect graph is perfect too. Hence the conflict graph of a grid graph with a zero conflict radius is a perfect graph. A perfect graph has the property that its set of clique constraints define its independent set polytope. So if we write a linear program with all the clique constraints together with the flow constraints then we can find the optimal network throughput. The problem, however, is that the number of cliques could still be exponentially many. (Although this does not happen with grid graphs, it could very well happen with other perfect graphs.) A solution is to use the ellipsoid algorithm [19] to optimize linear functions over a polytope. This algorithm does not require all the constraints in an explicit form to optimize a linear function over a polytope, hence we do not have to enumerate the exponentially many clique constraints. The ellipsoid algorithm only needs a subroutine that given a potential solution indicates whether the constraints are satisfied or not, and if not identifies at least one constraint which is not satisfied. Such a subroutine is called *separation oracle.* The separation oracle for our problem would be one that finds a violated clique constraint given a usage vector. This can be accomplished using the Grotschel semidefinite programming algorithm for finding the heaviest clique [15]. However, both the ellipsoid algorithm and the semidefinite algorithm have a running time of $O(n^3)$, so in combination their running time is $O(n^6)$. Thus this polynomial time algorithm is not very practical, which makes our heuristic approach attractive even in this special case.

.3 Finding Violated Odd Hole Constraints

Now we present a separation oracle that given a conflict graph G and a candidate solution λ finds a violated odd hole constraint, if any. Such an oracle could be used to improve the convergence rate of the algorithm presented in Section 3. Note that this separation oracle is applicable to general graphs; for the perfect conflict graph considered in Section .2 above, there are no odd holes anyway.

Consider an odd hole, H, of the given conflict graph G. Any vector λ inside the independent set polytope of G must satisfy the following: $\sum_{i\in H}\lambda_i \leq (|H|-1)/2$. A violated odd hole is one for which this constraint is not satisfied. Before attempting to find a violated odd hole, we may assume that the given λ satisfies all the edge constraints, i.e., $\lambda_i+\lambda_j \leq 1$ for every edge in G, because if it does not then we can include the violated edge constraint to shrink the upperbounding polytope. After making this assumption we define a weight function on the edges. For every edge ij of the graph G, we define its weight to be $1-\lambda_i-\lambda_j$, which is guaranteed to be non-negative. With this weight function we find the lightest (i.e., least-weight) odd cycle in the graph. The lightest odd cycle can be found using a bipartite graph construct as explained in the next paragraph. Let C be the lightest odd cycle. $\sum_{ij\in C}(1-\lambda_i-\lambda_j) < 1$ is equivalent to $\sum_{i\in C}\lambda_i > \frac{|C|-1}{2}$. So, if the weight of the lightest odd cycle is less than 1 then the cycle is a violated odd hole. If the weight of the lightest odd cycle is 1 or more then there is no violated odd hole.

Now we come to the question of efficiently finding the lightest odd cycle. Let G be the graph in which we need to find the lightest odd cycle. We construct a bipartite graph, B, as follows. For every vertex v in G we put two vertices v_l and v_r in B (the subscripts l and r can conceptually be thought of as representing the left and right "halves" of the the bipartite graph B). For every edge uv in G we put two edges u_lv_r and u_rv_l in B. Now an odd cycle in G becomes an odd length path in B e.g., $uvwu$ becomes $u_lv_rw_lu_r$. So for every vertex u in G we find the shortest path from u_l to u_r in B. The shortest such path in B yields the lightest odd cycle in G.