Real-Time Video Streaming in Multi-hop Wireless Static Ad Hoc Networks

Guy Even, Yaniv Fais, Moti Medina, Shimon (Moni) Shahar, and Alexander Zadorojniy

School of Electrical Engineering, Tel-Aviv Univ., Tel-Aviv 69978, Israel.
{guy,yanivfai,medinamo,moni,sasha}@eng.tau.ac.il

Abstract. We deal with the problem of streaming multiple video streams between pairs of nodes in a multi-hop wireless ad hoc network. The nodes are positioned on a plane, know their locations, and are synchronized (via GPS). We consider a 802.11g WiFi network in which NICs can hop between non-overlapping frequency channels in a synchronous fashion. We design a centralized algorithm that computes a frequency assignment and a schedule whose goal is to maximize the minimum throughput over all the video streams. In addition, we developed a localized flow-control mechanism to stabilize the queue lengths.

We simulated traffic scheduled by the algorithm using OMNET++/MixiM (i.e., physical SINR interference model with 802.11g) to test whether the computed throughput is achieved.

1 Introduction

We address the problem of routing real-time video streams in static ad hoc wireless networks. Our goal is to develop and implement an efficient algorithm and test it in the realistic physical model. Many works have been published on the topic of multi-hop routing in wireless networks including real-time video streaming (see [SYZ$^+$05,KPS$^+$06,Sha05,vDS05]). In these works it is acknowledged that cross layer algorithms are required to utilize the capacity of the network. These papers leave the issue of developing integrated realistic solutions open. In particular, a solution must address a combination of specifications including: maximize throughput, fairness, minimize delay, stability of throughput, stability of queue lengths in intermediate nodes, bound number of lost packets, sensitivity to distance in hops, and predictability.

One of the main issues in wireless networks is how interferences are modeled. In the communication community, one uses the signal-to-interference-plus-noise ratio (SINR) to determine if a received signal is decoded without an error [Gal68]. On the other hand, the algorithms community has used the graph model (or protocol model) to model feasible communication patterns [JPPQ05,ABL05]. For the graph model, multi-hop routing algorithms with a constant approximation ratio have been developed [KMPS05,ABL05,BSTZ07,Wan09]. In fact, Wan [Wan09] even presents a (theoretical) PTAS for the problem. On the other hand, to date approximation algorithms for throughput maximization in the SINR model do not have a constant approximation ratio. For example, in [CKM$^+$08], the approximation ratio is logarithmic in the

ratio between the longest link and the shortest link (for uniform transmission powers). In [Kes11], a logarithmic approximation ratio is obtained for a related problem of minimizing the makespan, however, this algorithm is based on unlimited transmission powers.

The study of wireless algorithms in the SINR model has been motivated by its realistic appeal. In fact, it has been argued that performance of graph based algorithms is inferior to algorithms the SINR model [GOW07]. A closer look at studies that compare the interference models and algorithms for these models shows only a constant gap. In [GK00] the same asymptotic throughput is obtained in both models with respect to random instances. In [MWW06], an example is analyzed in which the ratio between obtainable throughput in the SINR model and the graph model is about 2, even with very long links and nonuniform transmission powers. In [BR03], the theorems do not utilize the ability to increase the interference radius or to apply collision avoidance methods used in the 802.11 MAC.

The questions we study in this paper are as follows. (i) How much of the traffic computed by a graph based routing algorithm can be routed in reality? (ii) How could one integrate a graph based routing algorithm in a system that supports real-time video streaming? Such a system must combine goals such as: fairness, predictable throughput, few lost packets, bounded intermediate queues, reasonable and steady end-to-end delay.

Previous Work. The necessity of cross layer designs has been recognized for satisfying the special characteristics of real-time video streaming over wireless networks [Sha05,SYZ+05,KPS+06]. We continue this line of work.

The multi-hop routing problem for ad hoc networks was investigated thoroughly. One of the commonly used heuristics for routing is based on finding paths with maximum bottleneck, namely, paths for which the edge with the lowest capacity is maximum [DPZ04]. We used this algorithm in our benchmarks (we call it SHORTP). A different approach for the routing problem is based on solving a linear program. In [KMPS05,JPPQ05,ABL05], routing algorithms in the graph model are designed, analyzed, and simulated. One drawback in [KMPS05,JPPQ05,ABL05] is that the simulations were run also in the graph model and not in the physical model. Wan [Wan09] pointed out various errors in previous algorithms and presented a new linear program that corrects the problem. He proved that: (i) there is a 23-approximation algorithm based on the linear program, and (ii) there is a polynomial time scheme (PTAS) for the problem, however, this PTAS is not practical. The linear program presented in [Wan09] is more complicated than the LP in [KMPS05,ABL05,BSTZ07] and is harder to solve. It is not clear if Wan's method is efficient enough to work even with moderate sized networks.

Chafekar et al. [CKM+08,Cha09] considered routing algorithms in the SINR model. The approximation ratio of their algorithm with uniform power assignments is logarithmic in the ratio between the longest link and the shortest link. From a high level view, their algorithm is similar to the algorithms in [KMPS05,ABL05,BSTZ07]. Namely, an LP is formulated for a multi-commodity problem that includes also interference constraints. The solution of the LP is transformed into a schedule by applying greedy coloring. The analysis in [CKM+08] is rather involved since it needs to translate interferences caused by a transmitters to interferences accumulated by each receiver. Their

communication model does not include ACK packets. Hence, interference is caused only by the sender and not by the receiver.

In [Cha09], simulations are described in the physical model, but these simulations do not use the 802.11 MAC (i.e., no RTS, CTS, ACK packets are used).

Three scenarios are simulated in [Cha09]: random network, grid, and a realistic road-traffic network. The path loss exponent in the simulations is $\alpha = 6$, which is considered rather high and fit for indoor environments (but not open air environments). In the grid scenario, the nodes are located 10 meters apart (both horizontally and vertically), and the communication range also equals 10 meters. Thus, communication is possible only between closest neighbors. Interference caused by a node that is located a diagonal away (i.e., $10\sqrt{2}$ meters away) is 8 times smaller, and interference caused by a node 20 meters away is 64 times smaller. Thus, interferences in this setting fade very quickly, justifying a small gap between the SINR model and the graph based protocol model. Another aspect in the simulation of [Cha09] is that routing is limited to single paths (i.e., no splitting). It is not clear if this is an implementation issue or a result of the simulated instances.

Our Contributions.

1. We do not modify the 802.11g MAC. This approach has two advantages. First, we do not bypass the wireless NIC and its collision avoidance features. Hence, even if the algorithm suggests a schedule with interferences, these interferences are resolved by the MAC. Second, the network can support limited additional traffic that is not routed or scheduled by the algorithm (i.e. messages for controlling the network). We choose the 802.11g because of it popularity in laptops and mobile devices.

2. Simulation in the physical model. The simulation is in a standard 802.11g setting using OMNET++/MixiM (Sec. E.1). In this setting, all WiFi frames are transmitted (i.e., RTS,CTS, packet, ACK), and interferences between frames are analyzed using the SINR and the Modulation Coding Schemes (MCS).

3. We introduce new interference constraints that constitute an intermediate model between the physical SINR-model and the graph based protocol model (Sec. 3.1). The interference set of a link is a function of the signal-to-noise ratio of the link and the MCS of the link. As the link is closer to the SINR-threshold, the interference set grows, so that SINR is not in the "waterfall" region of the PER function. One advantage of this new interference model is that it is easy to formulate interference constraints in the linear program formulation (see Sec. C in the Appendix).

4. We formulate the problem of minimizing end-to-end delay due to a schedule that supports a given multi-flow. We developed and implemented a scheduling algorithm that addresses this problem of reducing end-to-end delays while supporting a similar throughput. (Sec 4.3). In [KMPS05,ABL05,BSTZ07] the effect of the schedule on the delay is not mentioned.

5. We developed and implemented a flow control algorithm that stabilizes the queue lengths and controls the data-rate along the links. This flow control algorithm is executed locally by the nodes.

6. We combine the various aspects of video streaming to analyze performance that is crucial for real-time video streaming. Namely, we measured the throughput, end-to-end delay, fraction of dropped packets, queue lengths, and the stability of these parameters.

Techniques. Following [KMPS05,ABL05,BSTZ07,CKM$^+$08], we formulate an LP, and apply greedy coloring to obtain a schedule. Interestingly, the greedy coloring incurs high end-to-end-delays, so we developed a path-peeling scheduler that trades delay for throughput. Stability is maintained by a flow control algorithm that monitors flow through incoming and outgoing links, and continuously balances the two. This method works thanks to the ability of video encoders to adjust the compressed bit-rate.

2 Problem Definition

Setting. We consider a WiFi 802.11g static ad hoc network with 3 non-interfering radio channels with the assumptions: (i) Single radio: each node has a single wireless network interface controller (WNIC). (ii) Each node is equipped with a GPS so that it knows its location and the nodes are synchronized. (iii) The WNICs support quick synchronized hops between frequency channels. (iv) Isotropic antennas. (v) We also assume that the nodes have already joined the network and that there is at least one node (i.e., center node) that holds full information about the network (i.e., nodes and locations). Accumulating this information can be done in a distributed low-bandwidth fashion after building a spanning tree [Awe87].

Problem Definition. The input to the algorithm consists of:

1. A set V of n nodes in the plane. A transceiver is located in each node.
2. A set of k video stream requests $\{r_i\}_{i=1}^k$. Each stream request is a triple $r_i \triangleq (a_i, b_i, d_i^*)$, where a_i is the source (e.g., camera) of the stream, b_i is the destination, and d_i^* is the required data-rate.

Ideally, we would like to satisfy all the requests, namely, for each video stream r_i, route packets using multi-hops from a_i to b_i. We assume that there is a path in the network between each source-destination pair (otherwise, the request is rejected).

Let d_i denote the data-rate achieved for the ith stream. The service ratio ρ_i of the ith demand is defined by $\rho_i \triangleq d_i/d_i^*$. Our goal is to maximize the minimum service ratio, namely, $\max \min_i \rho_i$.

Additional performance measures are: (i) End-to-end delay - this is the time it takes a packet to reach its destination. We are interested in reducing the maximum delay (among the packets that are delivered) since the video is real-time. In addition, the maximum delay determines the size of the jitter buffer in the receiving side. (ii) Number of dropped packets. Queue management may drop packets. A dropped packet never reaches its destination. (iii) Queue lengths in intermediate nodes tell us how much memory should be allocated and also give an indication of the delay per hop.

3 Preliminaries

3.1 Interference Models

Bidirectional interference. The delivery of a message in the WiFi MAC requires transmission of frames by both sides (e.g., RTS and packet are transmitted by the sender, CTS and ACK are transmitted by the receiver). Hence, interferences can be caused also by frames transmitted by a the receiving side.

The SINR model. The SINR model, also called the physical interference model, defines successful communication as follows. Let $d_{u,v}$ denote the distance between nodes u and v. Suppose a subset $S_t \subseteq V$ of the nodes are transmitting simultaneously in the same frequency channel as u. The signal-to-interference-plus-noise ratio (SINR) for the reception by $v \in V \setminus S_t$ of the signal transmitted by $u \in S_t$ in the presence of the transmitters S_t is defined by

$$\text{SINR}(u, v, S_t) \triangleq \frac{P/d_{u,v}^{\alpha}}{N + \sum_{x \in S_t \setminus u} P/d_{x,v}^{\alpha}}.$$

Each transmitter can use one of several modulation coding schemes (MCS). The message transmitted by u in an MCS m is successfully received by v if $\text{SINR}(u, v, S_t) \geq \beta_m$, where β_m is the minimum SINR-threshold for the MCS m.

Protocol model. The protocol model, also called the graph model, is specified by two radii: (i) A communication distance r. (ii) An interference distance R. The rule for successful communication between two nodes u and v is that v receives the message from u if $d_{u,v} < r$ and every other node x that transmits at the same time satisfies $d_{xv} > R$. In this model, a communication graph is defined over the nodes. Two nodes are linked by an edge if their distance is less than the communication distance r.

Since the WiFi MAC requires transmission by both sides, an *interference* is defined between two links (u, v) and (u', v') if $\min\{d_{u,u'}, d_{u,v'}, d_{v,u'}, d_{v,v'}\} < R$. We say that a subset L of links is *non-interfering* if no two links in L interfere. In the protocol model, a *schedule* is a sequence $\{L_i\}_i$ of subsets of non-interfering links.

Our new model. The new model is an intermediate model between the SINR model and the protocol model. The idea is that, as the SNR of a link grows, the link can tolerate more interference. Hence, the interference distance is not fixed.

Consider a pair (u, v) of nodes and an MCS m. The triple (u, v, m) is a *link* in the new model if $\text{SINR}(u, v, \emptyset) \geq \beta_m$.

Since both sides of a link transmit and receive, the interference set of a link must take into account interferences caused by other transmissions both in the receiver and the sender. However, the frames sent by the receiving side are in MCS 0, therefore, reception of these frames depends on the SINR-threshold β_0.

The interference set $V_{u,v,m}$ of the link $e = (u, v, m)$ is defined by

$$\begin{aligned} V_{u,v,m} \triangleq \{x \in V \setminus \{u\} \,|\, \text{SINR}(u, v, \{x\}) \geq \mu \cdot \beta_m \text{ or} \\ \text{SINR}(v, u, \{x\}) \geq \mu \cdot \beta_0\}. \end{aligned}$$

The motivation for this definition is that transmissions of nodes in $V_{u,v,m}$ interfere with the reception of v by u, or vice versa. The choice of $\mu = 1.585$ gives us a margin of 2dB above the SINR-threshold. This margin keeps the SINR above the threshold due to interferences caused by transmitters not in $S_{u,v,m}$.

We also define the interfering set of edges with respect to the link $e = (u, v, m)$.

$$I_{u,v,m} \triangleq \{e' = (u', v', m') \mid \{u', v'\} \cap (V_{u,v,m} \cup V_{v,u,m}) \neq \emptyset\}$$
$$\setminus \{(u, v, m)\}.$$

The interference set $I_{u,v,m}$ contains a link e' if either endpoint of e' interferes with reception at the endpoints u or v.

Notation. Let u and v denote nodes and m denote an MCS. A link is a triple (u, v, m) such that $\text{SINR}(u, v, \emptyset) \geq \beta_m$. This definition implies that there can be multiple parallel links between u and v, each with a different MCS. We denote the set of links by E. The set $E_{out}(v)$ (resp. $E_{in}(v)$) denotes the set of links that emanate from (resp. enter) v. Let $E(v)$ denote the set of links $E_{in}(v) \cup E_{out}(v)$. For a link $e = (u, v, m)$, let $\text{MCS}(e) = m$, i.e., the MCS m of the link e.

4 Algorithm MF-I-S

4.1 Networks Governed by Time-Slotted Frequency Tables

Two tables govern the communication in the network. The first table A is a time-slotted frequency table. The dimensions of A are $F \times T$, where F denotes the number of frequency channels and T denotes the number of time slots. There is one row for each frequency channel and one column for each time slot. (In our implementation we used $F = 3$ and $T = 200$). The table A determines a periodic schedule. The second table is a multi-flow table *mf*. The dimensions of *mf* are $|E| \times [1..k]$ (recall that k equals the number of video streams). The entry $mf(e, s)$ specifies the number of packets-per-period that should be delivered along link e for stream s.

Each table entry $A[j, t]$ is a subset of links, i.e., $A[j, t] \subseteq E$. The table governs communication in the sense that, in slot t', the links in $A[j, t' \pmod{T}]$ try to deliver packets using frequency channel j.

We use $A[\cdot, t]$ to denote the set of links $\cup_{j \in F} A[j, t]$. Since we assume that each node is equipped with a single radio, it follows that two links that share an endpoint cannot be active in the same time slot. Hence, for every node v, $E(v) \cap A[\cdot, t]$ may contain at most one link.

A time-slotted frequency tables schedules active links as listed in Algorithm TX-RX in Appendix B. Each node v executes Algorithm TX-RX(v) locally. Since $E(v) \cap A[\cdot, t]$ may contain at most one link, a node v is either a receiver, a sender, or inactive in each time slot.

4.2 Algorithm Specification

The input to the routing algorithm is specified in Sec. 2. The output consists of two parts: (i) a time-slotted frequency table A, and (ii) a multi-flow $mf(e, s)$, for every link

e and stream $1 \leq s \leq k$. We note that the units of flow are packets-per-period. The period equals $T \cdot \sigma$, where σ is the duration of a time slot, and T equals the number of time-slots in a period.

The multi-flow $mf(e, s)$ determines the routing and the throughout of each stream. The role of the frequency/time-slot table A and the multi-flow tables is to specify a periodic schedule that determines which links are active in which time slots (see Sec. 4.1).

Although we use fixed length packets (e.g., 2KB), the MCS of a link determines the amount of time required for completing the delivery of a packet. This means, that within one time slot, multiple packets may be delivered along a single link. Let $pps(e)$ denote the number of packets-per-slot that can be delivered along e. Namely, node u can transmit at most $pps(e)$ packets to node v along link $e = (u, v, m)$ in one time-slot. Note that the value of $pps(e)$ is a function of the MCS of the link e.

We say that table A *supports* the flow *mf* if the following properties hold:

1. Every entry $A[j, t]$ in the table is a set of non-interfering links. Thus, the links in $A[j, t]$ may be active simultaneously.
2. The data-rates $mf(e, s)$ are supported by the table. Namely,

$$\sum_{s=1}^{k} mf(e, s) \leq |\{A[j, t] : e \in A[j, t]\}| \cdot pps(e), \tag{1}$$

4.3 Algorithm Description

Algorithm MF-I-S consists of two parts: (i) computation of a multi-commodity flow with conflict constraints, and (ii) scheduling of the multi-commodity flow in a time-slotted frequency table. We elaborate on each of these parts.

Multi-commodity flow with conflict constraints. We formulate the problem of routing and scheduling the video streams by a linear program (LP). A similar LP is used in [KMPS05,ABL05,BSTZ07] with respect to the graph model. We use our new interference model for the interference constraints.

The variables $f_i^j(e)$ of the LP signify the amount of flow along link e in frequency channel j for stream i. The full LP appears in Appendix C. Let $f^j(e) \triangleq \sum_{i=1}^{k} f_i^j(e)$, namely, $f^j(e)$ is the flow in frequency j along link e. Let $c(e) \triangleq T \cdot pps(e)$ denote the number of packets-per-period that can be delivered along the link e.

We elaborate on two main features of the LP:

1. The conflict constraints. The ratio $f^j(e)/c(e)$ equals the fraction of the time that the link e is active in transmission in frequency j. Since each node is equipped with a single WNIC, transmissions emanating or entering the same node may not occur simultaneously (in any frequency). In addition, the links in I_e may not transmit in frequency j whenever e is transmitting in frequency j. Thus, the conflict constraint is formulated as follows. For every link $e = (u, v, m) \in E$, and for each frequency $j \in [1..3]$:

$$\frac{f^j(e)}{c(e)} + \sum_{j' < j} \sum_{e' \in E(u) \cup E(v)} \frac{f^{j'}(e')}{c(e')} + \sum_{e' \in I_e} \frac{f^j(e')}{c(e')} \leq 1$$

2. Max-Min throughput. For each requested stream r_i, we define the supply ratio ρ_i to be the ratio between the flow allocated to the i'th stream and the demand d_i^* of the stream. The objective of the LP is to maximize $\min_i \rho_i$. A secondary objective is to maximize the total throughput.

Scheduling of the multi-commodity flow in a time-slotted frequency table. In the scheduling step we are given the multi-commodity flows $f_i^j(e)$. The task is to allocate entries in a time-slotted frequency table A that supports these flows.

We first determine how many time-slots should be allocated for $f^j(e)$, for each link e and each frequency channel j. Similarly to Eq. 1,

$$|\{t \in [1..T] : e \in A[j,t]\}| \cdot pps(e) \geq f^j(e)$$

Hence,

$$|\{t \in [1..T] : e \in A[j,t]\}| \geq \left\lceil \frac{f^j(e)}{pps(e)} \right\rceil . \tag{2}$$

The greedy scheduler. The simplest way to assign flows to the table A is by applying a greedy algorithm (similar to greedy coloring). The greedy algorithm scans the links and frequency channels, one by one, and assigns $\ell(e,j)$ slots to each link e and frequency channel j. Based on [ABL05,KMPS04,BSTZ07], the interference constraints in Eq. 12 imply that the greedy algorithm succeeds in this assignment provided that

$$\ell(e,j) = \left\lfloor \frac{f^j(e)}{pps(e)} \right\rfloor . \tag{3}$$

The issue of dealing with this rounding problem (i.e., the difference between the round-down and the round-up in Eqs. 2 and 3) is discussed in [Wan09], where it is pointed out that routing all the flow requires a super exponential period T. Such a period is obviously not practical; the computation of the table takes too long, the table is too long to be broadcast to all nodes, and the schedule will incur huge delays.

We show that the rounding problem is not an important issue both theoretically and in practice. Since each flow f_i can be decomposed into at most $|E|$ flow paths, it follows that the values of $\{f_i^j(e)\}_{e \in E, j \in F}$ can be "rounded" so that at most $|E| \cdot \max_e \{pps(e)\}$ packets are lost per period. Note that this lost flow can be made negligible by increasing the period T. As T increases, the amount of flow per period tends to infinity, and hence, the lost flow is negligible. In our experiments E.1, we used a period of $T = 200$ time slots, with a duration of $5ms$ per slot. The greedy scheduler was able to schedule almost all the flow in all the instances we considered. The multi-flow table is set so that $mf(e,s)$ equals the amount of flow from $f_s(e)$ that the scheduler successfully assigned.

The greedy scheduler incurred a delay roughly of one period per hop. The reason is that it schedules all the receptions to a node before the transmissions from the node. To avoid this delay, we designed a new scheduler, described below.

The path-peeling scheduler. The path peeling scheduler tries to reduce the time that an incoming packet waits till it is forwarded to the next node. This is achieved as follows.

1. Decomposes each flow f_i into flow paths such that the flow along each path equals the bottleneck, i.e., the minimum $pps(e)$ along the path. Let $\{f_i(p)\}_{p \in \mathcal{P}(i)}$ denote this decomposition.
2. While not all the flow is scheduled,
   (a) For $i = 1$ to k do:
   (b) If $\mathcal{P}(i) \neq \emptyset$, then schedule a path $p \in \mathcal{P}(i)$ and remove p from $\mathcal{P}(i)$.

The scheduling of a flow path $p \in \mathcal{P}(i)$ tries to schedule the links in p one after the other (cyclically) to reduce the time a packet needs to wait in each node along p. The scheduling simply scans the links in p in the order along p, and finds the first feasible time slot (in cyclic order) for each link $e \in p$.

We point out that in Line 2a, we schedule one path from each stream to maintain fairness in allocation and delays. On the average, each stream suffers from the same "fragmentation" problems in the table A.

In our experiments, the path-peeling scheduler succeeded in scheduling 70% of the flow. The advantage, compare to the greedy scheduler, is that delays are significantly reduced.

5 Flow Control

The multi-flow table computed by the algorithm determines the number of packets $mf(e, s)$ that should be sent along each link e for stream s during each period. Each node v monitors the following information for each link $e \in E_{out}(v)$.

1. $P(e, s, t)$ - the number of packets belonging to stream s sent along the link e during the period t.
2. $P^+(e, s, t)$ - the maximum number of packets belonging to stream s that can be sent along the link e during the period t. Note that $P^+(e, s, t) \geq P(e, s, t)$; inequality may happen if the queue $Q(e, s)$ is empty when a packet is scheduled to be transmitted along the link e. Note that if e is not planned to deliver packets of stream s, then $P^+(e, s, t) = 0$.

We remark that a node v can also monitor $P(e, s, t)$ for a link $e \in E_{in}(v)$. However, the value $P^+(e, s, t)$ for a link $e \in E_{in}(v)$ must be sent to v (e.g., by appending it to one of the delivered packets).

The Flow-Control algorithm is executed locally by all the nodes in the network. Let $e = (u, v, m)$ denote a link from u to v, and let s denote a stream. Each node executes a separate instance per stream. In the end of each period t, each node u "forwards" the value of $P^+(e, s, t)$ to node v. In addition, in the end of each period t, node v sends "backwards" the value $R(e, s)$ to u. The value $R(e, s)$ specifies the number of packets from stream s that v is willing to receive along the link e in the next period $t + 1$.

The Flow-Control algorithm is listed as Algorithm 1. It equalizes the incoming and outgoing packet-rates in intermediate nodes as follows. The requested packet-rate $R(e, s)$ is initialized to be the value $mf(e, s)$ derived from the table. The Flow-Control algorithm is activated in the end of each period. It uses the values $P(e, s, t)$ and $P^+(e, s, t)$ for every link e incident to v. Some of these values are computed locally and some sent

---

Algorithm 1 Flow-Control(v, s) - a local algorithm for managing the local queue and requested incoming rate at node v for stream s.

---

1. Initialize: for all $e \in E_{in}(v)$, $R(e, s) \leftarrow mf(e, s)$.
2. For $t = 1$ to ∞ do
   (a) Measure $P(e, s, t)$ for every $e \in E(v)$, and $P^+(e, s, t)$ for every $e \in E_{out}(v)$.
   (b) Receive $P^+(e, s, t)$ for every $e \in E_{in}(v)$, and $R(e, s)$ for every $e \in E_{out}(v)$.
   (c) $R_{in} \leftarrow \min\{\sum_{e \in E_{out}(v)} R(e, s),$
       $\sum_{e \in E_{out}(v)} P^+(e, s, t), \sum_{e \in E_{in}(v)} P^+(e, s, t), \}.$
   (d) For every $e \in E_{in}(v)$: $R(e, s) \leftarrow R_{in} \cdot \frac{P^+(e, s, t)}{\sum_{e' \in E_{in}(v)} P^+(e', s, t)}$.
   (e) Drop oldest packets from $Q(v, s)$, if needed, so that $|Q(v, s)| \leq R_{in}$.

---

by the neighbors. The incoming packet-rate R_{in} is computed in line 2c, and is divided among the incoming links in line 2d. Excess packets in the queue $Q(v, s)$ are dropped so that the number of packets in $Q(v, s)$ is at most R_{in}. The rational is that, in the next period, at most R_{in} packets will be delivered, and hence, excess packets might as well be dropped.

We now elaborate on the boundary cases of the flow-control for a source a_s and a destination b_s of stream s. The destination b_s simply sends an fixed request for each incoming link $e \in E_{in}(b_s)$, i.e., $R(e, s) \leftarrow mf(e, s)$. The source a_s, does not execute line 2d; instead, it sets the packet-rate of the video encoder to R_{in}.

6 Experimental Results

In this section we summarize the main experimental results. An elaborated discussion of the experimental results appear in Appendix E.

Scenarios. We ran the experiments on a circle with 24 nodes (evenly spread requests) and a grid with 49 nodes (random requests). We used $k \in \{8, 12, 16\}$ requests each with a demand of 10 Mbps.

Benchmarks. We ran the experiments using six algorithms.

1. MF-I-S. In the MF-I-S benchmark all three parts of our algorithm are used: computation of a multicommodity flow with interference constraints, the path-peeling scheduler, and the Flow-Control algorithm.
2. SHORTP-S. A shortest path maximum bottleneck routing algorithm with the path-peeling scheduler. Let $pps(e)$ denote the number of packets-per-slot in the MCS used by the link e. Let $hops(p)$ denote the number of hops along a path p.
   We define a (lexicographic) order over paths from a_s to b_s as follows: $p \leq q$ if (1) $\min_{e \in p} pps(e) \geq \min_{e \in q} pps(e)$ or (2) $\min_{e \in p} pps(e) = \min_{e \in q} pps(e)$ and $hops(p) \leq hops(q)$. Formally, in SHORTP-S, the stream s is routed along a path p that is minimal in the lexicographic order.

In SHORTP-S, the paths are computed in an oblivious manner, namely, congestion does not play a role. This means that we must execute a flow control algorithm to adjust the data-rate.

Each stream in the SHORTP-S benchmark is assigned a random frequency channel.

3. The remaining algorithms are MF (only multi-commodity flow without interference constraints without a scheduler), MF-I (only multicommodity flow with interference constraints without a scheduler), MF-S (multi-commodity flow without interference constraints with a scheduler), SHORTP (shortest paths but without a scheduler). A detailed description appears in Appendix E.3. We point out that whenever the scheduler is not invoked, each node must have 3 WNICS. The reason is that a node does not know the frequencies of incoming packets.

We made the following change in the WiFi WNICs when there is a scheduler. The noise threshold for allowing a transmission of an RTS frame is reduced to match the interference distance. The reduced threshold relaxes the conservative collision avoidance to allow for simultaneous transmissions by links approved by the scheduler.

Results. We focus on two properties: min-throughput (i.e., the lowest throughput over all the streams) and the end-to-end delay.

k	MF-I-S's min Throughput	SHORTP-S's min Throughput	Ratio
	Mbps	Mbps	
8	0.576	0.45	1.28
12	0.448	0.325	1.3785
16	0.368	0.22	1.6727

Table 1: Comparison of steady state min-throughput between MF-I-S and SHORTP-S in the grid scenario. The number of requests is denoted by k.

Table 1 lists the effect of the number of requests k on the minimum throughputs of MF-I-S and SHORTP-S in the grid scenario. MF-I-S outperforms SHORTP-S by 28-67%.

In Figure 1 we compare MF-I-S with the greedy scheduler and the path-peeling scheduler. The path peeling scheduler significantly reduces the end-to-end delay while slightly reducing the throughput. Note that the min-throughput is bigger with the path peeling scheduler (i.e., stream #8), hence, fairness is improved.

In Tables 2 and 3, a summary of the simulation results is listed. It is evident that the dropped packets rate is exceedingly high when the scheduler is not used. The reason is that the flow control algorithm fails to stabilize the queue length, as one would expect. Note that MF-I-S uses longer paths to avoid congestion and interference.

The circle scenario is highly symmetric (i.e., 24 nodes and 12 streams) so we can suggest an optimal solution. This solution is a table with two slots. In slot 1, the "odd" links are scheduled. In slot 2, the "even" links are scheduled. We need only two frequencies: so that links separated by a link use different frequencies. In this solution, the flow along each link is half its capacity.
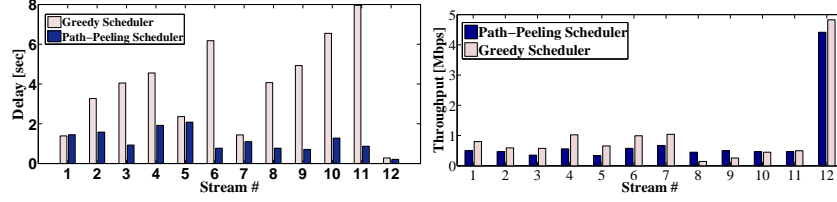
Fig. 1: Comparison of end-to-end delays and throughputs between the greedy scheduler and the path-peeling scheduler. MF-I-S was run in the grid scenario with $k = 12$ requests and uniform demands $d_i^* = 10$Mbps.

Since the capacity of a link in this scenario is 8.2 Mbps , it follows that the flow along a link is 4.1 Mbps. Since streams are routed along disjoint paths, the throughput per stream is also 4.1 Mbps.

	#radios	throughput			Delay	delay based on hops				hops	drops	PER
	per node	min	sum	max	max	max		min		avr	max	avr
		Mbps	Mbps	Mbps	sec	#hops	sec	#hops	sec		%	%
MF-I-S	1	0.512	12.08	4.544	2.25	14	2.25	1	0.2	7.17	0.7	1.85
SHORTP	3	0.064	8.9	1.568	**24**	9	23	1	1.35	4.5	24	1.09
SHORTP-S	1	0.402	5.02	0.4342	1.5	9	1.32	1	0.95	4.5	0	0.15
MF-I	3	0.064	4.32	1.2	3.9	14	2.6	1	1.2	7.17	**95**	5.54
MF	3	0	3.14	0.704	4.9	12	2.4	1	1.3	6.67	**78**	6.63
MF-S	1	0.1548	4.26	0.8	**19**	12	15	1	3.7	6.67	0.5	0.87

Table 2: Comparison of the benchmarks for the grid scenario with $k = 12$ requests and $d_i^* = 10$Mbps for each stream. The experiment's duration is 25 seconds.

	#radios	throughput			Delay	delay based on hops				hops	drops	PER
	per node	min	sum	max	max	max		min		avr	max	avr
		Mbps	Mbps	Mbps	sec	#hops	sec	#hops	sec		%	%
MF-I-S	1	2.37	32.1	2.97	1.23	2	1.23	2	0.68	2	0	0
SHORTP	3	4.82	56.2	4.88	2.5	2	2.5	2	1	2	34	0
SHORTP-S	1	1.98	30.4	2.78	**5.35**	2	4.1	2	0.35	2	0	0
MF-I	3	1.35	18.4	1.75	1.5	2	1.5	2	1	2	**36**	3
MF	3	1.3	17.7	1.7	1.7	2	1.7	2	1	2	**14**	2.48
MF-S	1	0.85	15.42	1.75	2.64	2	2.64	2	4	2	0.65	0

Table 3: Comparison of the benchmarks for the circle scenario with $k = 12$ requests and $d_i^* = 10$Mbps for each stream. The experiment's duration is 25 seconds.

7 Conclusions

The algorithm consists of two parts: a multi-commodity flow computation and a scheduler. Our simulations demonstrate the robustness of the scheduler. Namely, the flows *mf* that are supported by the time-slotted frequency table A are successfully routed in the SINR-model. This is true even though the scheduler uses an interference model that is similar to the graph model (not the SINR-model).

The role of the multi-commodity flow computation with interference constraints is to maximize the minimum throughput. Indeed, in the grid scenario, routing along shortest paths resulted with smaller throughputs.

The flow control algorithm succeeds in stabilizing the queue lengths for all benchmarks that used the scheduler. Without the scheduler, stability was not obtained, and many packets were dropped.

Our results show that one can compute a routing and scheduling that succeeds in the SINR-model while using a simpler interference model. In addition, we successfully combined the various goals required to support video streaming.

8 Discussion

We propose a centralized algorithm for computing a routing, scheduling, and frequency assignment for real-time video streams in static ad hoc wireless networks. The algorithm consists of two parts: a linear program and a scheduler. In addition, each node locally runs a flow-control algorithm to control the queues and stabilize data-rate along the links. Although the algorithm is centralized, it can be executed by multiple nodes in the network provided that they hold full information of the network (i.e., locations, requests). The output of the algorithm consists two tables that can be easily broadcast to all the nodes.

We implemented the algorithm and experimented using a setting that uses the physical model (with a 802.11g MAC) to verify the validity of the algorithm. Our experiments show that the traffic routed and scheduled by the algorithm is successfully delivered in two congested scenarios in the SINR-model.

We propose a scheduling algorithm, called the path peeling scheduler, that is designed to reduce the end-to-end delay incurred by the greedy scheduler. The path peeling scheduler succeeded in reducing the delay in streams with many hops. Even in a congested scenario, the path peeling scheduler successfully scheduled at least 70% of the flow.

Our simulation provides evidence that a schedule computed in a model quite close to the graph model is feasible in the SINR-model. This can be attributed to the sharp threshold that characterizes successful decoding as a function of the SINR in AWGN channels.

References

[ABL05] M. Alicherry, R. Bhatia, and L.E. Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 58–72. ACM, 2005.

[Awe87] B. Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 230–240. ACM, 1987.

[BR03] A. Behzad and I. Rubin. On the performance of graph-based scheduling algorithms for packet radio networks. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, volume 6, pages 3432–3436. IEEE, 2003.

[BSTZ07] C. Buragohain, S. Suri, C. Tóth, and Y. Zhou. Improved throughput bounds for interference-aware routing in wireless networks. *Computing and Combinatorics*, pages 210–221, 2007.

[Cha09] D.R. Chafekar. *Capacity Characterization of Multi-Hop Wireless Networks-A Cross Layer Approach*. PhD thesis, Virginia Polytechnic Institute and State University, 2009.

[CKM$^+$08] D. Chafekar, VSA Kumart, M.V. Marathe, S. Parthasarathy, and A. Srinivasan. Approximation algorithms for computing capacity of wireless networks with SINR constraints. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1166–1174. IEEE, 2008.

[DPZ04] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 114–128. ACM, 2004.

[Gal68] R.G. Gallager. *Information theory and reliable communication*. John Wiley & Sons, Inc. New York, NY, USA, 1968.

[GK00] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Transactions on information theory*, 46(2):388–404, 2000.

[GOW07] O. Goussevskaia, Y.A. Oswald, and R. Wattenhofer. Complexity in geometric SINR . In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 100–109. ACM, 2007.

[JPPQ05] K. Jain, J. Padhye, V.N. Padmanabhan, and L. Qiu. Impact of interference on multihop wireless network performance. *Wireless networks*, 11(4):471–487, 2005.

[Kes11] T. Kesselheim. A constant-factor approximation for wireless capacity maximization with power control in the SINR model. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.

[KMPS04] VS Kumar, M.V. Marathe, S. Parthasarathy, and A. Srinivasan. End-to-end packet-scheduling in wireless ad-hoc networks. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1021–1030. Society for Industrial and Applied Mathematics, 2004.

[KMPS05] VS Kumar, M.V. Marathe, S. Parthasarathy, and A. Srinivasan. Algorithmic aspects of capacity in wireless networks. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 133–144. ACM, 2005.

[KPS$^+$06] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer. Application-driven cross-layer optimization for video streaming over wireless networks. *Communications Magazine, IEEE*, 44(1):122–130, 2006.

[MDS10] M.K. Marina, S.R. Das, and A.P. Subramanian. A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks. *Computer Networks*, 54(2):241–256, 2010.

[MWW06] T. Moscibroda, R. Wattenhofer, and Y. Weber. Protocol design beyond graph-based models. *Hotnets*, 2006.

[Sha05] Y. Shan. Cross-layer techniques for adaptive video streaming over wireless networks. *EURASIP journal on applied signal processing*, 2005:220–228, 2005.

[SYZ$^+$05] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod. Cross-layer design of ad hoc networks for real-time video streaming. *Wireless Communications, IEEE*, 12(4):59–65, 2005.

[vDS05] M. van Der Schaar. Cross-layer wireless multimedia transmission: challenges, principles, and new paradigms. *Wireless Communications, IEEE*, 12(4):50–58, 2005.

[Wan09] P.J. Wan. Multiflows in multihop wireless networks. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 85–94. ACM, 2009.

A   Special characteristics of real-time video streaming

Streaming of real-time video in a multi-hop ad hoc network is a challenging task with unique characteristics. (1) End-to-end delay in streaming of real-time video should be as small as possible. We assume that a delay of 1-2 seconds is tolerable if the video has to travel across 10 hops. One implication of this constraint is that end-to-end acknowledgments and TCP are not an option. (2) Unless erasure codes are employed, loosing even a small fraction of the packets incurs an intolerable degradation in the video quality. We assume that video has acceptable quality if less that 0.5% of the packets are dropped. In wireless networks, each link can have a PER of 1%-5%. Thus, after 10 hops, one is left with intolerable erasures. On the other hand, erasure codes incur an extra end-to-end delay since they need to accumulate data for a block before encoding can take place. This means that relying on WiFi acknowledgments and retransmit capabilities can be useful to avoid packet drops if the PER is small. (3) A useful feature in video coding is the ability to adjust the compressed bit-rate. This means that the video encoder can be continuously controlled to generate a video stream of a requested bit rate. We rely on this feature in our flow control algorithm. This feature separates video streaming from other applications such as FTP.

B   TX-RX Algorithm

A listing of the TX-RX algorithm appears as Algorithm 2. We elaborate below how a queue $Q(v, s)$ with the highest priority for transmission along link e in line 2 in the Transmit procedure is defined.

Upon invocation of Transmit(e, j), where $e = (v, u, m)$, the node v needs to decide which packet to transmit. The node v uses the multi-flow table *mf* to determine the set S_e of streams that are routed along e. Since delay is a major issue, it is reasonable to use an EDD-like policy (Earliest Due Date), i.e., pick the oldest packet in the queues $Q(v, s)$, for $s \in S_e$. However, such a policy ignores the remaining number of hops a packet needs to traverse. We prefer the approach that emphasizes fairness. That is, assign a priority that equals the ratio of the number of packets of stream s transmitted along e in the last period divided by the required number. The lower this ratio, the higher priority of the stream. This approach also combines well with the flow control algorithm described in Sec. 5.

Algorithm 2 TX-RX(v) - a local transmit-receive algorithm for node v as specified by a time-slotted frequency table A.

For time slot $t' = 0$ to ∞ do

1. $t = t' \pmod{T}$.
2. if $E_{in}(v) \cap A[\cdot, t] \neq \emptyset$ then {reception mode}
   (a) Let $e \in E_{in}(v) \cap A[\cdot, t]$, where $e(u, v, m) \in A[j, t]$.
   (b) While slot t is not over call Receive(e, j).
3. if $E_{out}(v) \cap A[\cdot, t] \neq \emptyset$ then {transmission mode}
   (a) Let $e \in E_{out}(v) \cap A[\cdot, t]$, where $e(v, u, m) \in A[j, t]$.
   (b) While slot t is not over call Transmit(e, j).

Receive(e, j) - where link $e = (u, v, m)$ and j is a frequency channel.

1. Set tuner to reception in frequency channel j.
2. Upon reception of a packet p from stream s, insert p to $Q(v, s)$.

Transmit(e, j)- where link $e = (v, u, m)$ and j is a frequency channel.

1. Set tuner to transmission in frequency channel j.
2. Pick a queue $Q(v, s)$ with a highest priority for transmission along e.
3. $p \leftarrow DEQUEUE(Q(v, s))$.
4. Transmit p along e.

C   The Linear Programming Formulation

The main variables of the LP are the flow variables $f_i^j(e)$ which signify the amount of flow along link e in frequency channel j for stream i. In Eq. 5 we require that the flows are nonnegative. In Eq. 6 we define $f_i(e)$ to be the combined flow along e for stream i over all frequency channels. In Eq. 7 we define $f^j(e)$ to be the combined flow along e in frequency channel j over all k streams. Eq. 8 is simply a flow conservation constraint for stream i in every intermediate node. Eq. 9 is simply a capacity constraint for every link. In Eq 10, the supply ratio ρ_i is defined to be the fraction of the demand for stream i that is supplied. In Eq. 11, ρ is defined to equal the minimum supply ratio, i.e., $\rho = \min_i \rho_i$. Finally, in Eq. 12 the interference constraints are defined; we elaborate on them below.

The objective is to maximize the minimum supply ratio ρ. As a secondary objective, we maximize the sum of flows. Therefore, the constant λ in the objective function is small (e.g., $\lambda = 1/20$).

We point out that the capacity constraints in Eq. 9 are redundant since they are implied by the interference constraints in Eq. 12.

In our experiments, we noticed that the LP-solver found a solution with flow cycles. We removed these cycles before applying the scheduling step. Interestingly, the issue of flow cycles was not mentioned in previous works [ABL05,BSTZ07]

$$\max \quad \rho + \lambda \cdot \sum_{i=1}^{k} d_i^* \cdot \rho_i \quad \text{subject to} \tag{4}$$

$$f_i^j(e) \geq 0 \qquad \forall i \in [1..k], \forall j \in [1..3], \forall e \in E \tag{5}$$

$$\sum_{j=1}^{3} f_i^j(e) = f_i(e) \qquad \forall e \in E \tag{6}$$

$$\sum_{i=1}^{k} f_i^j(e) = f^j(e) \qquad \forall e \in E \tag{7}$$

$$\sum_{e \in E_{out}(v)} f_i(e) - \sum_{e \in E_{in}(v)} f_i(e) = 0 \qquad \forall i \in [1..k], \forall v \in V \setminus \{a_i, b_i\} \tag{8}$$
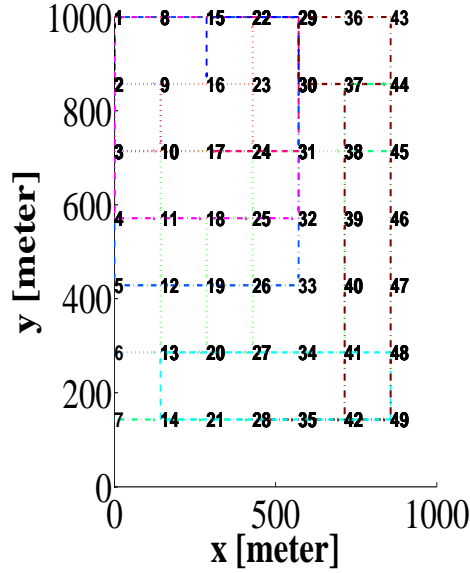
$$\sum_{i=1}^{k} f_i(e) \leq c(e) \qquad \forall e \in E \tag{9}$$

$$\sum_{e \in E_{out}(v)} f_i(a_i) - \sum_{e \in E_{in}(v)} f_i(a_i) = d_i^* \cdot \rho_i \qquad \forall i \in [1..k] \tag{10}$$

$$\rho \leq \rho_i \qquad \forall i \in [1..k] \tag{11}$$

$$\frac{f^j(e)}{c(e)} + \sum_{j' < j} \sum_{e' \in E(u) \cup E(v)} \frac{f^{j'}(e')}{c(e')} + \sum_{e' \in I_e} \frac{f^j(e')}{c(e')} \leq 1 \qquad \forall e = (u, v, m) \in E, \forall j \in [1..3] \tag{12}$$

D   The Grid Scenario

(a) Grid scenario layout

Fig. 2: The grid scenario with 49 nodes, and $k = 12$ requests. Flow paths, computed by MF-I-S, are depicted. An example of the splitting of flow can be seen for the request from node 49 to node 13. This request is split in to two paths along the perimeter of the rectangle $(13, 48, 14, 49)$.

E   Experimental Results

E.1   General Setting

WiFi parameters. In the benchmarks that use the scheduler, each node has a single 802.11g WNIC. In the benchmarks that do not use the scheduler, each node has three 802.11g WNICs. The reason is that, in absence of the scheduler, a node does not know to which frequency channel to tune in each moment.

Each WNICs transmits in one of three non-overlapping frequencies. All WNICs transmit at a fixed power ($100mw$). The path loss exponent is $\alpha = 4.1$. The noise figure is $N = -100dBm$. The maximum communication is roughly $150m$ (in MCS 0). An interference $250m$ away can cause a decrease in the SINR of roughly $1dB$. We used fixed size packets with a payload of $2KB$. Thus, a video stream with a 1Mbps generates 64 packets per second.

Software tools. We used Coin-OR CLP to solve the linear programs. We implemented the scheduler in C++. The simulation was implemented using OMNET++/MixiM. There-

fore, the simulation is done in the physical model taking into account path loss, multiple interferences, partial interference between frames, and all the details of the 802.11g protocol.

Algorithm parameters. We used $T = 200$ time slots in the time-slotted frequency table. Each time slot has a duration of 5ms.

Implementation details. The following simplifications we made in the implementation. (1) Virtual flow control messages are used. They are sent without delay in the end of each period. We justify this simplification since flow control messages are very sparse. (2) Packets of only one stream are sent along each link in each time slot. This simplification only reduces the throughput of the implementation.

E.2   Scenarios

We ran the experiments on two main types of arrangements of the nodes in the plane: a circle and a grid.

1. In the grid arrangement, we positioned 49 nodes in a $1km \times 1km$ square. The nodes are positioned in a 7×7 lattice, so that the horizontal and vertical distance between adjacent nodes is $1000/7 = 142$ meters (see Fig. 2). The source and destination of the streams in the grid arrangement are chosen randomly.
2. In the circle arrangement, we positioned 24 nodes on a circle of radius 500 meters. The nodes were positioned every $360/24$ degrees. The source and destination of the streams in the circle arrangement are chosen deterministically, by $a_1 = \lceil 24/k \rceil$, $b_i = (a_i + \lfloor 24/k \rfloor) \bmod 24$, $a_{i+1} = b_i$, where k denotes the number of streams.

We point out that random locations of 50 nodes in a square kilometer induces a communication graph with a high degree and a diameter of 2 or 3 [MDS10]. In addition, the interference set of each link contains almost all the other links. Hence, this setting has a low capacity and is not an interesting setting for the problem we study.

The requests demand d_i^* is set to 10Mbps. Such a demand with $k \geq 6$ streams is above the capacity of the network. This enables us to study the performance in a congested setting.

E.3   Benchmarks

We ran the experiments using six algorithms:

1. SHORTP- a shortest path maximum bottleneck routing algorithm. Let $pps(e)$ denote the number of packets-per-slot in the MCS used by the link e. Let $hops(p)$ denote the number of hops along a path p. In SHORTP, the stream s is routed along a path p from a_s to b_s such that, for every path p' from a_s to b_s, the following holds:

$$\min_{e \in p} pps(e) \geq \min_{e \in p'} pps(e) \text{ , and}$$
$$\min_{e \in p} pps(e) = \min_{e \in p'} pps(e)$$
$$\Longrightarrow hops(p) \leq hops(p').$$

The paths assigned to the k streams are divided evenly among the three frequency channels.

Each node in this benchmark contains three radios. This means that each node contains three standard 802.11g WNICs, each working in different frequency channel. Since the frequency channels are non-overlapping, one WNIC may receive while another WNIC is transmitting. Each WNIC receives and transmits packets according to the WiFi MAC. Fairness between the streams is obtained as follows. Each WNIC is given FIFO-queue for each stream, the packets of which it needs to transmit. Each WNIC uses a simple round-robin policy for determining the queue from which the next packet is transmitted.

The paths are computed in an oblivious manner, namely, congestion does not play a role. This means that we must execute a flow control algorithm to adjust the data-rate. To execute the Flow-Control algorithm without any changes, we trivially cast this routing to our setting as follows. We define the multi-flow $mf(e, s)$ to equal d_s^* if e is in the path assigned to stream s, and 0 otherwise. The time-slotted frequency table A has a single time slot (i.e., $T = 1$) whose duration is one second. Namely, the table A has three entries, one for each frequency channel. The table entry for frequency channel j lists the links that use frequency channel j.

2. MF-I-S. In the MF-I-S benchmark all three parts of our algorithm are used: computation of a multicommodity flow with interference constraints, the path-peeling scheduler, and the Flow-Control algorithm,

   We emphasize that in this benchmark, each node contains a single radio; namely, each node has a single standard 802.11g WNIC capable of hopping between the three frequency channels in the beginning of each time slot.

3. SHORTP-S- same as SHORTP except that every node contains a single radio. In addition, the path-peeling scheduler is applied, hence $T = 200$, and a random frequency channel is assigned to every request.

4. MF-S - same as MF-I-S except that the LP does not include interference constraints. The scheduler resolves interferences, so it is interesting to see how much throughput is scheduled by the scheduler, and whether this throughput is routed in the simulation.

   We point out that the interference constraints constitute a large part of the LP constraints. By omitting them, the LP becomes shorter, easier to solves, and naturally, the LP solution has a higher throughput.

   Since the LP lacks interference constraints, the scheduler may fail to schedule the flow. We modified the scheduler in this case so that it augments the table A by adding time-slots. This augmentation has an adverse effect of reducing throughput and increasing delay.

5. MF-I - same as MF-I-S but without the scheduler. Instead, the multi-flows are assigned in a single-slot schedule, as in Algorithm SHORTP. We point out that in this benchmark, each node is equipped with three WNICs as in Algorithm SHORTP. The motivation for this benchmark is that the multi-flow takes into account congestion and interference. Since the WiFi MAC deals with avoiding collisions, so it is interesting to see how it succeeds in scheduling the multi-flows in a distributed manner.

   As in SHORTP, each node has three WNICS in this benchmark.

6. MF - similar SHORTP except that the streams are routed according to a multi-commodity flow. The multi-commodity flow is computed by an LP without interference constraints. This benchmark helps understand whether non-oblivious congestion aware routing improves performance.

As in SHORTP, each node has three WNICS in this benchmark.

E.4 Results

Routing results. The routing result for $k = 12$ streams is depicted for the grid scenario in Fig. 2 in Appendix D. Note that a request may be split among multiple paths.

Benchmark comparison. In Tables 2 and 3 we summarize the measured performance of the benchmarks for the grid and circle scenarios with $k = 12$ requests and a demand $d_i^* = 10$Mbps for each stream. The experiment's duration is 25 seconds. Notice that the MF-I-S MF-S and SHORTP-S benchmarks use only one WNIC per node, while the other benchmarks use three WNICs per node.

We begin by discussing the grid scenario: (1) The minimum throughput obtained by MF-I-S is the highest. (2) The sum of the throughputs obtained by MF-I-S is 35% higher than that of SHORTP, and 140% higher than that of SHORTP-S. (3) MF-I-S uses longer paths to avoid congestion and interference. (4) The scheduled benchmarks (i.e., MF-I-S SHORTP-S and MF-S) have a very small drop rate. (5) All benchmarks have reasonable PER.

In the circle scenario we obtained the following results: (1) The minimum throughput obtained by MF-I-S is 0.49 that of SHORTP, and 0.5 that of SHORTP-S. (2) The sum of the throughputs obtained by SHORTP is only 75% more than that of MF-I-S. MF-I-S. (3) Drop rate are small also for SHORTP.

In light of the fact that MF-I-Suses a single WNIC per node, it is clear that it outperforms all other algorithms. Most importantly, the end-to-end delay in MF-I-Sis much shorter.

Comparison between SHORTP *and* SHORTP-S. Table 4 depicts the effect of applying the path-peeling scheduler on the greedy algorithm SHORTP, that is we compare algorithm SHORTP and SHORTP-S in the grid scenario. Recall that SHORTP-S has a single radio in every node. The number of requests k is 12. The demand for every stream request is 10 Mbps.

It can be seen that SHORTP-S has no drops, hence our algorithm MF-I-S is compared to SHORTP-S.

Effect of number of streams. Table 1 depicts the effect of the number of requests k on the minimum throughputs of MF-I-S and SHORTP-S in the grid scenario. The number of requests k is $8, 12, 16$. The demand for every stream request is 10 Mbps.

Clearly, the min-throughput decreases as the number of requests increases, as the same resources need to serve more requests. The advantage of MF-I-S is maintained for this range of requests.

	SHORTP's drops percentage	SHORTP-S's drops percentage
max	46	0
min	0	0
average	11.66	0

Table 4: Comparison of the drop percentage between SHORTP and SHORTP-S in the grid scenario. The drop percentage is the ratio between the number of dropped packets and the number of the transmitted packets. The simulations used $k = 12$ streams and a uniform demand of $d_i^* = 10$ Mbps.

Effect of single radio on SHORTP. Since SHORTP uses 3 WNICs per node, we experimented also with a single WNIC per node. Figure 3a depicts the effect of single radio on total throughput of SHORTP. The ratio is almost constant and equals 3, as expected.

The experiment was made in the grid arrangement, where The number of requests is $k = 12$. The demand for every stream request is 10 Mbps.
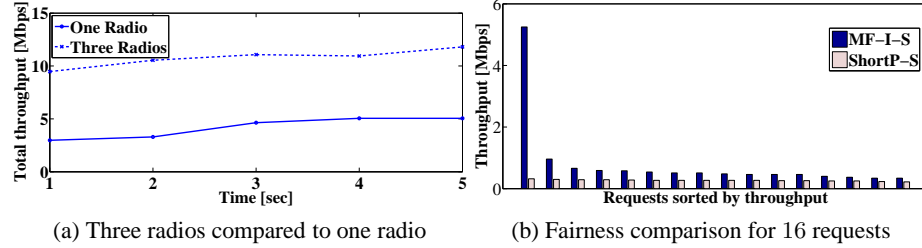

(a) Three radios compared to one radio   (b) Fairness comparison for 16 requests

Fig. 3: (a) Comparison of total throughput between one and three radios, for SHORTP in the grid scenario with $k = 12$ and $d_i^* = 10$Mbps. (b) Throughput comparison between SHORTP-S and MF-I-S. Streams in each benchmark are sorted by their throughput for the grid arrangement with $k = 16$. The experiment's duration is 5 seconds.

Fairness. Figure 3b compares the sorted throughputs achieved for the streams in MF-I-S and SHORTP-S. The experiment was made in the grid arrangement, with $k = 16$ streams and $d_i^* = 10$ Mbps.

It can be seen that every request achieves less throughput in SHORTP-S than in MF-I-S. Note that SHORTP-S balances the throughput among its requests.

The Flow-Control algorithm. Figure 4 depicts the effects of the Flow-Control algorithm in the grid arrangement with $k = 12$ and $d_i^* = 10$Mbps. In Fig. 4a, the requested rates $R(e, s)$ are depicted. It can be seen that only slight perturbations occur over time. This justifies our simplified implementation that uses virtual flow-control messages.

In Fig. 4b, the queue lengths of the stream in three different nodes are depicted. The oscillation is due to the periodic schedule. The queue length is controlled and stabilizes.

In Fig. 4c, the drop ratio is depicted for the worst stream. The drop rate ranges from 0 to 1%.

In Fig. 4d, the differences between the maximum and minimum throughput for MF-I-S and SHORTP-S are depicted for all streams two seconds after the beginning of the experiment. It is evident that MF-I-S and SHORTP-S are stable since the differences are smaller.
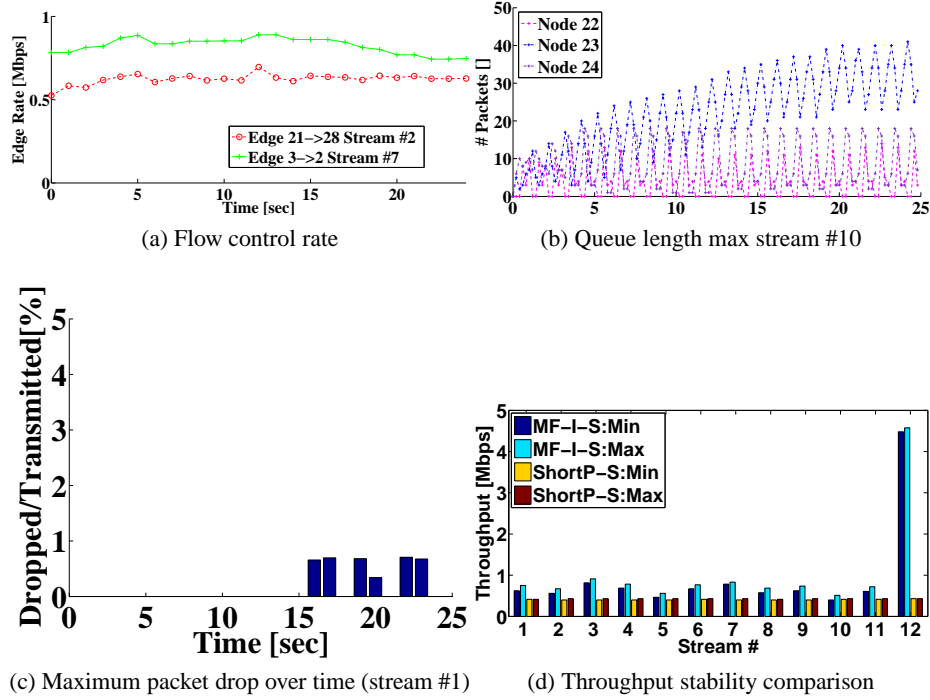
(a) Flow control rate

(b) Queue length max stream #10

(c) Maximum packet drop over time (stream #1)

(d) Throughput stability comparison

Fig. 4: Influence of the Flow-Control algorithm in the grid arrangement with $k = 12$ and $d_i^* = 10$Mbps. The experiment's duration is 25 seconds: (a) Change in the requested packet rate by the flow-control over time in MF-I-S, (b) Queue lengths of a stream in three different nodes in MF-I-S, (c) Ratio of dropped packets to transmitted packets over time in worst stream in MF-I-S, (d) Comparison of stability of throughput between MF-I-S and SHORTP-S.

Comparison with greedy scheduler. In Figure 5 we compare MF-I-S with the greedy scheduler and the path-peeling scheduler. The throughput obtained with the path-peeling scheduler is smaller, the end-to-end delay is significantly improved.
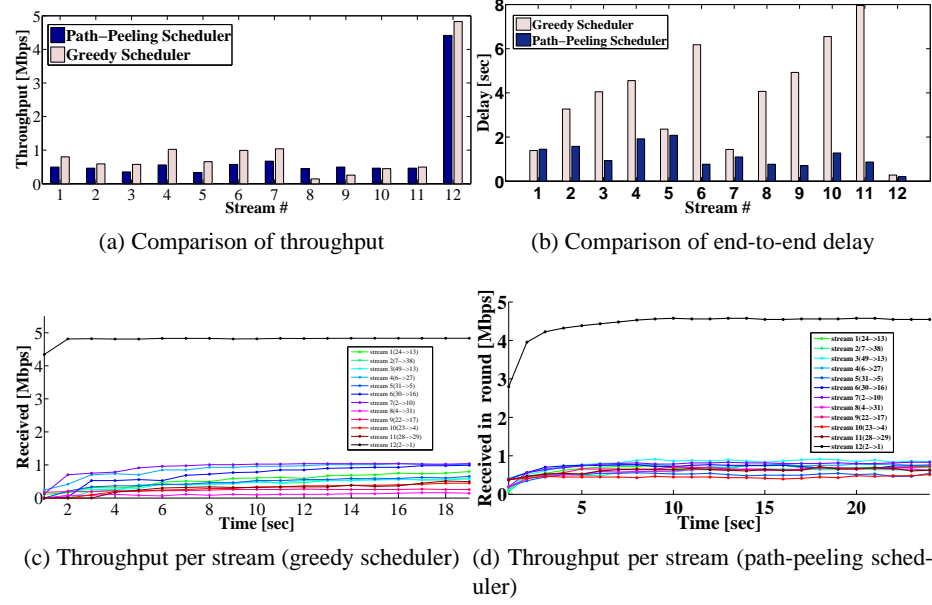
(a) Comparison of throughput

(b) Comparison of end-to-end delay

(c) Throughput per stream (greedy scheduler)

(d) Throughput per stream (path-peeling scheduler)

Fig. 5: Comparison of MF-I-S with the greedy scheduler and the path-peeling scheduler in the grid arrangement with $k = 12$ and $d_i^* = 10$Mbps. The experiment's duration is 25 seconds.