A New Model for Packet Scheduling in Multihop Wireless Networks

Haiyun Luo, Songwu Lu
UCLA Computer Science Department
Los Angeles, CA 90095-1596
email: {hluo, slu}@cs.ucla.edu

Vaduvur Bharghavan
Coordinated Science Laboratory
University of Illinois
email: bharghav@crhc.uiuc.edu

Abstract

The goal of packet scheduling disciplines is to achieve *fair* and *maximum* allocation of channel bandwidth. However, these two criteria can potentially be in conflict in a generic-topology multihop wireless network where a single logical channel is shared among multiple contending flows and spatial reuse of the channel bandwidth is possible. In this paper, we propose a new model for packet scheduling that addresses this conflict. The main results of this paper are the following: (a) a two-tier service model that provides a minimum "fair" allocation of the channel bandwidth for each packet flow and additionally maximizes spatial reuse of bandwidth, (b) an ideal centralized packet scheduling algorithm that realizes the above service model, and (c) a practical distributed backoff-based channel contention mechanism that approximates the ideal service within the framework of the CSMA/CA protocol.

1 Introduction

In recent years, researchers have developed numerous resource management algorithms and protocols for wireless mobile networking environments [1, 2, 3, 4, 5, 6], e.g., QoS oriented MAC layer design, packet scheduling, mobility management, admission control and resource reservation to name a few. The end goal of all these proposals is to devise effective management schemes for capacity-constrained and highly dynamic wireless networks in order to support communication intensive applications with service assurances that are comparable to their wireline counterparts. In many of these proposed designs, fair distribution of bandwidth and maximization of resource utilization have been identified as two important design goals, notably for scheduling disciplines [2, 3, 7]. Fairness is critical to ensure that well-behaved users are not penalized because of the excessive resource demands of aggressive users. Maximizing resource utilization is critical to effectively support communication-intensive applications, e.g., web browsing, video conferencing and remote transfer of large files, which can easily stress the bandwidth-constrained wireless channel.

Achieving both fairness and maximization of channel utilization in packet scheduling is particularly challenging in a shared-medium multihop wireless network. Since wireless transmissions are locally broadcast in the shared physical channel, location-dependent contention exists among flows in a neighborhood [8]. How to ensure fair channel allocation among spatially contending packet flows through packet scheduling has not been addressed in related literature. Besides, the multihop nature of a shared-channel wireless network makes spatial channel reuse possible [2, 4]. How to maximize channel reuse, and hence the aggregate network capacity, poses another challenge. Unfortunately, the two goals of ensuring fairness and maximizing resource utilization have inherent conflicts in shared-medium multihop wireless networks, as we will illustrate in this paper. Two extreme approaches for resolving this conflict are to either maximize the aggregate channel utilization without any fairness considerations (potentially starving some packet flows), or to enforce strict notions of fairness across all flows in the network at the cost of possibly significant reductions in the aggregate channel utilization.

In this paper, we investigate a model for packet scheduling that arbitrates these two design criteria in order to resolve the inherent conflict between them. The main results of this paper are the following: (a) a two-tier service model that provides a minimum "fair" allocation of the channel bandwidth for each packet flow and additionally maximizes spatial reuse of bandwidth, (b) an ideal centralized packet scheduling algorithm that realizes the above service model, and (c) a practical distributed backoff-based channel contention mechanism that approximates the ideal service within the framework of the CSMA/CA protocol. We evaluate our approach through simulations and simple analysis.

The organization for the rest of the paper is as follows. Section 2 explores the design issues and the solution space. Section 3 proposes a channel sharing model and a centralized packetized algorithm that achieves the proposed model within analytically provable performance bounds. Section 4 presents a distributed backoff-based channel contention mechanism that has the same long-term expected behavior for channel sharing as the proposed model. Section 5 evaluates the proposed mechanism through simulations. Section 6 discusses related work, and Section 7 concludes the paper.

2 Design Issues and Solution Space

2.1 Network Model

We consider a packet-switched multihop wireless network in which the wireless medium is shared among multiple contending users, i.e., a single physical channel with capacity C is available for wireless transmissions. Transmissions are locally broadcast and only receivers within the transmission range of a sender can receive its packets. Each link-layer packet flow is a stream of packets being transmitted from the source to the destination, where the source and destination are neighbors. We define two flows as *contending flows* if either the sender or the receiver of one flow is within the transmission range of the sender or the receiver of the other flow[1] [8, 14].

We make three assumptions [2, 7, 8, 10]: (a) neighborhood is a commutative property and hence flow contention is also commutative, (b) a node cannot transmit and receive packets simultaneously, and (c) a collision occurs when a receiver is in the reception range of two simultaneously transmitting nodes, thus unable to cleanly receive signal from either of them; we ignore capture effect in this paper. We do not explicitly consider mobility and non-collision-related channel errors in this paper.

2.2 Design Issues

A. Location-dependent contention and spatial reuse

The locality of wireless transmissions implies that collisions, and hence contention for the shared medium, are location dependent. The location-specific nature of contention, coupled with the multi-hop nature of the network, allows for spatial channel reuse. Specifically, any two flows that are not interfering with each other can potentially transmit data packets over the physical channel simultaneously. The selection of simultaneous transmitters thus determines the aggregate channel utilization, hence the packet scheduling discipline needs to perform a judicious selection of such simultaneous transmissions while taking into account fairness considerations across flows.

In a wireline or packet cellular network, packets are scheduled independently at each link, and the scheduler at a link only needs to consider flows that are contending for that link. Fluid fairness defined for such networks is, in essence, a *local* property for transmitting flows over each link and packet scheduling algorithms for achieving the fluid fairness model, e.g., Weighted Fair Queueing, ensure *local fairness* in the time domain among contending flows that share a single link. In a shared-medium multihop wireless network, fairness cannot be defined with respect to "local" flows alone, because of the possibility of spatial channel reuse, and the location-dependent constraints in the selection of flows for simultaneous transmission. As a result, fairness has to be defined with respect to contending flows in both the time domain and the spatial domain.

B. Conflict between fairness and maximizing channel utilization

In a wireline link or a cell in a packet cellular network, at most one flow can transmit at any time, and the scheduling of packets across different links/cells is independent. In the target environment, multiple flows may transmit simultaneously, but the transmission of a flow in a region has an impact on which other flows can transmit in the rest of the network. The "global" nature of packet scheduling in multihop shared channel wireless networks leads to a conflict between achieving fairness and maximizing aggregate channel utilization. For example, consider the five backlogged flows in Figure 1. In order to maximize aggregate channel utilization, a simple solution is to starve flows F_1, F_2, and F_4 and let F_3 and F_5 transmit all the time. This way, the aggregate channel utilization is $2C$ (where C denotes the physical channel capacity). It is easy to verify that the aggregate channel utilization will be less than $2C$ if flows F_1, F_2, and F_4 receive non-zero channel allocations.

[1]Following the CSMA/CA medium access paradigm, we assume that data transmission will be preceded by a control handshake. Thus the nodes in the neighborhood of both the sender and the receiver must defer transmission to ensure a successful handshake.

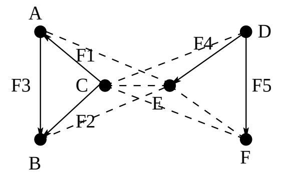

Figure 1: Fundamental conflicts between fairness and maximal utilization

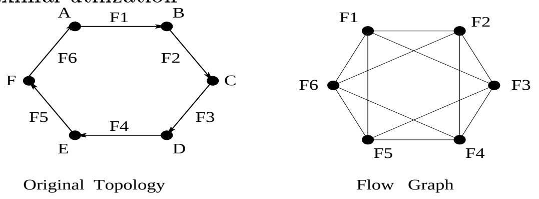

Figure 2: Generating a flow graph

The above example illustrates the fundamental conflict between achieving flow fairness and maximizing overall system throughput. Specifically, some flows may need to be starved to maximize channel utilization and conversely, enforcing any notion of "fairness" across flows may result in sub-optimal channel utilization. The basic issue is thus the trade-off between these two conflicting criteria.

2.3 Solution space

The goal of this work is to address the trade-offs between achieving fairness and maximizing channel utilization.

At one end is the approach that achieves some predefined notion of fairness without taking channel utilization into account [22]. At the other end is the approach that always tries to schedule the largest number of non-conflicting backlogged flows at any time, thereby maximizing aggregate channel utilization while potentially starving some flows. In this work, we explore the middle ground - we enforce a basic notion of fairness that ensures that each flow receives a minimum channel allocation; subject to this constraint, we seek to maximize aggregate channel utilization. Of course, the interesting question is how the channel utilization improves as the fairness model becomes coarser. Depending on the requirements of the system, the network administrator can potentially choose a particular point in the solution space.

Specifically, we investigate two points in the solution space:

1. A flow i with weight r_i receives a lower bound on channel allocation of $\frac{r_i}{\sum_{j \in B(t1)} r_j} C(t1, t2)$ over an infinitesimal time period $(t1, t2)$, where $B(t)$ is the set of backlogged flows in the entire network at time t. Subject to this lower bound on channel allocation, the scheduling discipline tries to maximize aggregate channel allocation. This fairness model is *global* and *topology independent* in the sense that it assumes the worst case of all flows contending with each other.

2. A flow i with weight r_i receives a lower bound on channel allocation of $\frac{r_i}{k \cdot \sum_{j \in B(i,t1)} r_j} C(t1, t2)$ over an infinitesimal time period $(t1, t2)$, where $B(i, t)$ is the set of backlogged flows within a two-hop distance (in the node graph) of flow i at time t and k is a constant. Subject to this lower bound on channel allocation, the

scheduling discipline tries to maximize aggregate channel allocation. This fairness model is *local* and *topology dependent* because it provides a lower bound on channel allocation with respect to the current contention in the locality of the flow.

The first approach provides for coarser fairness than the second model, requires global backlogged flow information to achieve the schedule, but provides a priori worst case bounds on channel allocation that does not change with the network topology and results in possibly higher aggregate channel utilization. In the rest of the paper, the model and algorithms that we propose can achieve both these approaches, and we evaluate the fairness and utilization trade-offs for these two approaches.

3 The Packet Scheduling Model

In this section, we propose an idealized packet scheduling framework that addresses the design issues identified in the previous section. We first describe a fluid channel sharing model in which each packet flow is treated as a fluid flow. We then describe a packetized algorithm that emulates the fluid model in a packet switched network and analyze its properties. Our framework is idealized because we assume complete knowledge of the network topology and flow information at the scheduler.

3.1 The fluid model and the flow contention graph

In the fluid model, the granularity of channel sharing is a bit, and each flow f is assigned a weight r_f [11]. The goal is to assign a minimum channel allocation to each flow proportional to its weight, and subject to this constraint, maximize the aggregate channel utilization.

The first step in our model is to convert flows in a generic network topology into a *flow contention graph,* which characterizes the space-time contention relationship among transmitting flows. In a flow contention graph, each vertex represents a backlogged flow, and an edge between two vertices denotes that these two flows are contending (as defined in Section 2.1). Vertices that are not connected denote flows that can transmit simultaneously. Thus, an independent set in the flow contention graph denotes a set of non-conflicting transmissions. Figure 2 illustrates the generation of the flow contention graph from the network topology. $\{F_1, F_4\}$ and $\{F_2, F_5\}$ are independent sets and can thus transmit simultaneously.

Looking at the flow contention graph provides an insight into why fair scheduling in the target domain is a uniquely difficult problem. Disconnected subgraphs in the flow contention graph can be scheduled independently. In a wireline network, (link-layer) flows that share the same output link form a clique and the network is represented by a collection of disjoint cliques; thus each clique can be independently scheduled and there is at most one transmitter in a clique at a time. In a shared channel multihop wireless network, the task is to identify a sequence of independent sets (i.e. simultaneous transmitters) subject to the topology constraints of the graph, such that each flow receives a minimum representation in the sequence of independent sets and at the same time, the aggregate cardinality of these sets is maximized.

Our approach is to first achieve the fairness model by selecting a set of flows for transmission in a *fair queueing phase*, and then maximize channel utilization by selecting additional flows for transmission in a *maximum independent set phase* subject to the selection of the flows in the fair queueing phase. The precise details of the algorithm in the two phases decide whether the fairness model is *global* or *local* (as defined in Section 2.3).

3.2 Achieving a minimum fair share through fair queueing

Fluid fair queueing mandates that when a set of flows F share a channel, a flow i with weight r_i receives a channel allocation of $C\frac{r_i}{\sum_{j\in B(t)} r_j}\delta t$ over any small time window δt, where C is the channel capacity and $B(t)$ is the set of backlogged flows at time t. Several packetized scheduling algorithms exist to approximate the fluid fair queueing model. We now present a hybrid variant of Start time Fair Queueing (STFQ) [18] and Worst-case Fair Weighted Fair Queueing (WF2Q) [17], which we use as the starting point of our scheduling discipline in the idealized scheduling framework.

Each flow has a queue for its packets. Packets in a flow are served in FIFO order. Each packet has two tags, a *start tag* and a *finish tag.* The start tag of the n^{th} packet of flow i is specified as

$$s_{i,n} = max\{v(t_{i,n}), f_{i,n-1}\}$$

and the finish tag of the n^{th} packet of flow i is specified as

$$f_{i,n} = s_{i,n} + L/r_i$$

where $s_{i,n}$ and $f_{i,n}$ denote the start and finish tags, $v(t)$ is the *virtual time* at time t, $t_{i,n}$ denotes the arrival time of the packet, and L is the fixed packet size.

The virtual time $v(t)$ at time t is set to the start tag of the packet currently being transmitted on the channel.

After the transmission of a packet, the next packet to transmit is selected according to the following algorithm.

- Among all packets whose start tag is not greater than $v(t) + L$, the packet with the minimum finish tag is selected.

- If there is no such packet, then the packet with the minimum start tag is selected.

Ties are broken arbitrarily.

We now present the idealized packet scheduling algorithms for achieving the global and local fairness models respectively.

Recall that in the global fairness model, a backlogged flow i receives a channel allocation of at least $C\frac{r_i}{\sum_{j\in B(t)} r_j}\delta t$ in time $(t, t + \delta t)$, where $B(t)$ is the set of all backlogged flows in the network. This fairness property is identical to the one approximated by the packetized fair queueing algorithm above. Thus, we use this algorithm to provide a "basic" allocation, and subject to this allocation, we seek to maximize the aggregate channel reuse according to the following algorithm.

1. Select the head of line packet of flow i^* according to the packetized fair queueing algorithm described above.

2. Select the maximum independent set S_{i^*} in $G - N[i^*]$, where $N[i]$ denotes the closed neighborhood of node i in the flow contention graph.

3. Schedule packets for transmission in $\{i^*\} \cup S_{i^*}$. Increment the start and finish tags for flow i^*, but not for any of the flows in S_{i^*}.

The fact that the tags are not incremented for the flows in S_{i^*} enables the scheduler to achieve the maximum possible additional channel reuse given the allocation for i^* "for free", i.e. the flows that receive additional channel allocation are not charged for it by increasing their tags. We present a simple analysis of the properties of this algorithm in Section 3.3.

Recall that in the local fairness model, a backlogged flow i receives a channel allocation of at least $C \frac{r_i}{\sum_{j \in B(t)} r_j} \delta t$ in time $(t, t+\delta t)$, where $B(t)$ is the set of all backlogged flows in its closed neighborhood. In contrast to the global fairness model, achieving local fairness using the packetized fair queueing algorithm is a little more subtle, and requires the following modification: let D be a "basic" set of flows as defined below; the virtual time $v(t)$ is set to the maximum of the start tags of the head of line packets of the flows in D. With this modified packetized fair queueing algorithm, we now define the algorithm for achieving the local fairness model as follows. After the transmission of a packet,

1. Set D to NULL. For each flow, if the start tag of the head of line packet of a flow is not greater than $v(t)+L$, then set the state of the flow to `contend`, else set the state of the flow to *no-contend*.

2. If there is no flow in `contend` state, then add the flow with the minimum start tag to D and skip to the next step. Otherwise, while there are flows in the `contend` state, select the flow f with the minimum finish tag of the head of line packet and add f to the set D. Set all flows in the closed neighborhood of f, $N[f]$, to `no-contend`.

3. Update the virtual time $v(t)$ to the maximum start tag of the head of line packets among flows in D. Update the start and finish tags of the flows in D.

4. Select the maximum independent set S in the graph $G - N[D]$.

5. Schedule the flows in $S \cup D$ for transmission. Do not increment the start and finish tags of the flows in S.

The set D contains the flows that receive channel allocation as a result of the local fairness property, while the set S contains the flows that receive additional channel allocation in order to maximize aggregate channel utilization.

3.3 Approximating the maximum independent set

In the previous section, our idealized scheduling algorithms uses a maximum independent set generation algorithm in order to maximize channel utilization subject to minimum fairness constraints. A maximum independent set of a graph is a subset of vertices with largest cardinality such that no two vertices in the subset are neighbors in the graph. While this is a well known NP-complete problem [13], we use a minimum-degree greedy algorithm to approximate the maximum independent set [15]. It has been shown in [15] that this algorithm achieves a performance ratio of $(\Delta + 2)/3$ for approximating independent sets in graphs with degree bounded by Δ. Figure 3 shows the pseudocode for the algorithm.

```
S: the set of nodes in the graph
v : a node in the set G
N(v) : adjacent node set of v
d(v) : degree of node v
B: output set

    B ← φ
    while S ≠ φ
        choose v such that d(v) = min d(w), w ∈ S
        B ← B ∪ v
        S ← S − {{v} ∪ N(v)}
    return B
```

Figure 3. Minimum-degree Greedy Algorithm

3.4 Slot queues and packet queues

In our idealized scheduling algorithms, we update the start and finish tags when a flow receives channel allocation as a part of its "fair share", but not as a part of additional channel allocation for maximizing utilization. In order to accommodate this selective updating of tags, we decouple "slots", the unit of channel allocation, from "packets", the unit of transmission. A flow maintains two queues: a slot queue and a packet queue. Start and finish tags are associated with slots and not packets.

When a packet arrives for a flow, it gets added to the packet queue, and a new slot is added to the slot queue. Corresponding start and finish tags are assigned to the new slot. If a flow receives service through the fair queueing phase, then it transmits the head of line packet from the packet queue and deletes the head-of-line slot from the slot queue. If it receives service through the maximum independent set phase, it transmits a packet from the packet queue, but leaves the slot queue unchanged.

When all packets are fixed size, the slot queue and packet queue decoupling is easily accomplished, as described above. For variable length packets, the same decoupling principle works, but is more involved and not discussed further in this paper.

3.5 Analytical Properties of the Packetized Algorithm

We now briefly characterize the properties of the idealized scheduling algorithm analytically. Due to space constraints, we only present the properties for the algorithm that achieves the global fairness model.

Fairness and throughput in the basic channel

First note that each backlogged flow will always receive a basic fair service by assuming that no spatial reuse were available. That is, each flow receives at least a fair share from the basic physical channel capacity C. Then both the long-term throughput and packet delay bounds, developed for a standard WFQ scheduler [12] hold for the basic physical channel.

Theorem 3.1 *(Short-term fairness over the basic channel) Let $W_f(t_1, t_2)$ denote the service (in bits) that flow f receives in the basic channel during $[t_1, t_2]$. Then the difference in the service received by two backlogged flows f and m is given as:*

$$\left| \frac{W_f(t_1,t_2)}{r_f} - \frac{W_m(t_1,t_2)}{r_m} \right| \leq \frac{L}{r_f} + \frac{L}{r_m} . \qquad (1)$$

Theorem 3.2 *(Short-term throughput over the basic channel) Consider a backlogged flow f over $[t_1, t_2]$. Let $W_f(t_1, t_2)$ denote the service (in bits) that flow f receives in the basic* *channel during $[t_1, t_2]$. Then the following throughput bound for flow f holds:*

$$W_f(t_1, t_2) \geq \frac{r_f}{\sum_{i \in B(t_1,t_2)} r_i} C(t_2 - t_1) - L, \qquad (2)$$

where $B(t_1, t_2)$ denotes the backlogged flow set over $[t_1, t_2]$, and C is the basic channel capacity.

Spatial reuse

We now characterize the optimality of spatial reuse and the spatial reuse gain.

Theorem 3.3 *(Optimality of spatial reuse) Consider all the feasible scheduling policies that allocate each backlogged flow at least a weighted fair share of channel C. Then the optimal solution to the maximum independent set problem of Section 3.2 maximizes spatial reuse of bandwidth in this feasible scheduling policy space.*

Proof This theorem can be proved via contradiction. Given a network topology, denote all the maximum independent sets (MIS), which are sorted by the descending order of their cardinality, as $C_1, C_2, \ldots, C_m$. Consider an arbitrary flow f. In order to provide a basic fair share for f, our scheduling algorithm described above will generate a flow f dependent MIS C_k. Let us assume that there is another MIS C_l with its cardinality $l > k$, which also includes flow f and can provide the same basic fair share for flow f as C_k. It is easy to see that the choice of C_l will result in larger spatial reuse than MIS C_k, while providing the same basic fair share for flow f. However, this is impossible, because it means that both C_l and C_k will be the flow f dependent MIS but their cardinalities are different, i.e., $l > k$. This contradicts the definition of a maximum independent set for a given flow. □

Remark 3.1 *If we enlarge the space to include all possible scheduling policies in Theorem 3.3, then the optimality of spatial reuse may not hold true any more. In fact, a policy in which only certain flows are selected and others remain starved, may achieve even higher spatial reuse.*

Remark 3.2 *Theorems 3.1 and 3.3 state that our algorithm arbitrates fairness and maximal resource utilization in the following sense: fairness is ensured among backlogged flows in the basic channel, and spatial reuse is maximized subject to the fairness constraint in the basic channel.*

Define the spatial reuse gain γ to be the ratio of total spatial reuse of bandwidth $\mathcal{R}$ and the basic channel capacity C, i.e., $\gamma = \frac{\mathcal{R}}{C}$. Then the following theorem characterizes the spatial reuse gain:

Theorem 3.4 *(Spatial reuse gain) Consider n backlogged flows. Let flow f's independent set have n_f flows, obtained via an approximation algorithm to the maximum independent set problem. Then the spatial reuse gain is given by:*

$$\gamma = \sum_{f=1}^{n} r_f n_f, \qquad (3)$$

where r_f denotes the normalized weight of flow f, i.e., $\sum_{f=1}^{n} r_f = 1$. □

4 A Distributed Implementation

In Section 3, we have presented an ideal centralized algorithm, where the scheduler is assumed to have the perfect knowledge of the per-flow information at each node in the entire network flow graph. However, packet scheduling in a multihop wireless network is an inherent distributed computation problem. How to design an effective distributed implementation of packet scheduling in such networks that approximates the ideal centralized algorithm of Section 3 is the task of this section.

4.1 Two Design Issues

Distributed nature of packet scheduling in multihop wireless networks In a multihop wireless network, spatially contending flows may originate from different sending nodes. Unlike wireline or packet cellular networks, no single logical entity for scheduling of these flows is available. Besides, per flow information, e.g., the backlogged status and packet arrivals for each flow, is "distributed" among these sending nodes, and each sender does not have direct access to other flows' information at other senders. Consider Figure 2 again, each of the six senders A to F does not know the packet-level information of flows at the other five nodes. This illustrates that packet scheduling in a multihop wireless network is a distributed computation problem by its nature.

Information propagation in a broadcast medium If we adopt a global topology-independent fairness model, when a new flow joins the network (possibly after an admission control process) or an existing flow exits the network, we may have to propagate this information, *in minimum time*, to the entire network graph. However, if we adopt a local topology-dependent fairness model (see Section 2.3), we do not need a global networkwide infrastructure for flow information propagation. Flow information only needs to be propagated to its one-hop neighborhood in the flow contention graph.

In the following, we will focus on the problem of information propagation if a global topology-independent fairness model is adopted. In a network that has point-to-point links, the optimal solution to propagate information from a given node to *all* the rest nodes of the network in minimum time is to build up a shared, minimum-height spanning tree, and the solution can be obtained using breadth-first search algorithm or a more generic Dijkstra's algorithm. However, in a shared-channel multihop wireless network, the wireless medium is a local broadcast channel, and there are potential collisions for packet transmissions in a spatial locality. As a result, propagating information along a minimum-height spanning tree may not be optimal any more! This can be illustrated through the example shown in Figure 4. Figure 4.(a) shows the standard spanning tree, and in a network with point-to-point links only, the transmission times to propagate information from root A to all the rest nodes will be 3 units (i.e, the height of the tree). However, since both B and C are within range of E (the dotted line between two nodes in the Figure denotes that they are within communication range of each other), in order to propagate to all the nodes, sibling nodes B and C cannot transmit concurrently to their children (otherwise, E perceives collisions). Hence, A has to transmit to B and C sequentially (but not concurrently), and it takes 4 units to reach to all nodes, as shown in Figure 4.(b). However, if we construct

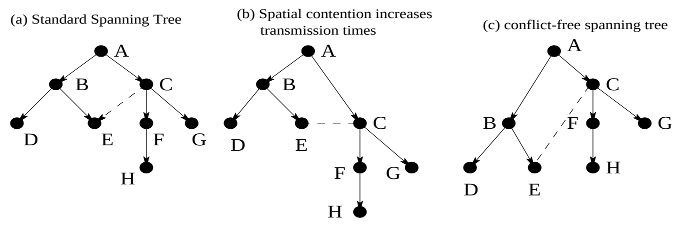
(a) Standard Spanning Tree
(b) Spatial contention increases transmission times
(c) conflict-free spanning tree
A
B
C
D
E
F
G
H

Figure 4: Information propagation along a spanning tree with spatial contention

the tree as in Figure 4.(c), we only need 3 units to propagate information from A to all nodes.

In essence, unlike in a point-to-point link medium, sibling nodes (located at the same level) in the tree may not be able to concurrently transmit in a broadcast medium due to spatial contention. This effectively increases the total propagation time needed to propagate information to all the nodes along the tree.

4.2 A backoff-based distributed algorithm

4.2.1 Algorithm description

In this section, we describe a backoff-based distributed implementation that effectively approximates our proposed algorithm of Section 3. A brief overview of our implementation is as follows (a pseudo code for the backoff-based implementation is shown in Figure 5):

We assume a CSMA/CA based MAC protocol. For each data packet transmission, there is a RTS-CTS-DATA-ACK data handshake, and each transmitting station will sense the carrier before sending out its RTS message. Each backlogged flow will set an appropriate waiting time (i.e., backoff value in terms of mini-slots) before it transmits a RTS message. Proper setting of the backoff value depends on the choice of the specific scheduling and fairness model. In our implementation, we set the flow with the minimum scheduling precedence in the basic channel to have a zero waiting time, and it will transmit immediately, in order to provide worst-case fair allocations. In the meantime, other flows will set their backoff waiting time to be equal to their flow degrees. Therefore, the node with smallest degree in the flow contention graph will transmit a RTS request first; upon hearing the RTS message, all its neighboring nodes will backoff until the completion of this packet transmission. In the meantime, flows beyond its two-hop neighborhood may potentially transmit concurrently; again flows with smaller flow degrees will get higher priority for transmission. This way, our implementation can realize the minimum-degree greedy approximation to the maximum independent set problem described in Section 3.

```
$b_f$: flow $f$'s backoff value in minislots
$z_f$: allocated transmission slots for flow $f$
$r_f$: flow $f$'s weight
$F$: the flow set in the flow contention graph
$S_f(0), S_f(2), \ldots, S_f(r_f - 1)$: the scheduling order of flow $f$ using WRR with spreading /* $S_f(i) \in \{0, 1, \ldots, \sum_{k \in F} r_k\}$*/
$k_f$: the number of packet transmissions that flow $f$ has received in the current cycle
$d(f)$: flow $f$'s degree in the flow contention graph
$\mathcal{N}(f)$: $f$'s one-hop neighborhood in the flow contention graph
$c_f$: flow $f$'s slot location in the current cycle
$c_f$: $0, 1, .., \sum_{f \in F} r_f; 0, 1, .., \sum_{f \in F} r_f; \ldots$ /*global fairness*/
$c_f$: $0, 1, .., \sum_{i \in \mathcal{N}(f)} r_i; 0, 1, .., \sum_{i \in \mathcal{N}(f)} r_i; \ldots$ /*local fairness*/

initialization for flow $f$:
    if(GLOBAL_FAIRNESS_MODEL)
        $z_f \leftarrow S_f(0)$;
    if(LOCAL_FAIRNESS_MODEL)
        $tmp \leftarrow (\sum_{i \in \mathcal{N}(f)} r_i)/r_f$;
        $z_f \leftarrow$ random $(tmp)$;    /* $z_f \in \{0, 1, \ldots, tmp - 1\}$ */

at time $t$, for flow $f$
if $c_f = z_f$   {        /* transmitting in the basic channel */
    $b_f \leftarrow 0$;          /* backoff is reset to zero */
    $k_f \leftarrow (k_f + 1) mod(r_f)$;  /*update transmissions for $f$*/
    if(GLOBAL_FAIRNESS_MODEL)
        $z_f \leftarrow S_f(k_f)$;  /* next scheduling order */
    if(LOCAL_FAIRNESS_MODEL)
        $tmp \leftarrow (\sum_{i \in \mathcal{N}(f)} r_i)/r_f$;
        $z_f \leftarrow k_f \times tmp +$ random $(tmp)$;    }
else      /* for spatial reuse transmission */
    $b_f = d_f$ /* backoff value set to be the flow degree */
```

Figure 5. Pseudocode for backoff-based implementation

In our implementation, we may adopt either a global topology-independent fairness model or a local fairness model (see Section 2). Our implementation framework works for both cases. In the following, we will focus on the distributed implementation for the global fairness model, which is more involved than the local fairness model.

If we adopt a global fairness model (see Section 2.3), the flow information, e.g., the number of flows in the network, each flow's weight, has to be propagated in the entire network topology. Therefore, an information propagation infrastructure needs to be available for this purpose. To this end, we construct a core-based shared tree for information propagation. The shared tree supports a collision-free downstream (i.e., from core node to other nodes in the network) message multicast in the network topology.

Approximating the fair queueing algorithm in the basic channel For the global topology-independent fairness model, we need to approximate the fair queueing algorithm described in Section 3.1.2. To this end, the core node (of the shared tree) maintains per-flow information, and calculates a scheduling order for each flow using a weighted round robin with spreading [3]. Then the core node will propagate the scheduling order for each flow along the shared multicast tree. Consider a flow f in the flow set F of the network topology, we normalize the flow weights for flows in F such that the smallest flow weight in F is normalized to be one, then we set the weight of flow f to be equal to its normalized weight r_f. If we define a "cycle" as $\sum_{f \in F} r_f$ slots, then each flow should transmit exactly r_f slots in each cycle. The WRR with spreading is essentially an approximation of WFQ algorithm by assuming that *each flow were always* *backlogged* and the packet size is the same for each flow. Its worst-case performance bound, in terms of throughput, packet delay and fairness, is the same as the WFQ algorithm. However, if certain flows become idle, then the above algorithm will deviate from the WFQ algorithm. Specifically, extra bandwidth (due to idle flows) will not be allocated to backlogged flows that are waiting to be served in the basic channel; instead, we will give spatial reuse higher priority. That is, the slot allocated to an idle flow in the basic channel will not be allocated to another backlogged flow in the basic channel; it will be shared among multiple concurrent transmitting nodes (that belong to a flow-dependent maximum independent set).

Realizing the minimum-degree greedy algorithm In our implementation, we take a backoff-based approach to the minimum-degree greedy approximation of the maximum independence set problem. The backoff based mechanism works as follows: for each packet transmission, each flow sets a backoff timer and waits for a number of mini-slots, before transmitting a RTS request to its neighboring flows. Upon hearing a RTS request, every flow (in its neighbors) will disable its backoff timer and restrains from transmission until the transmitting flow finishes its current packet transmission. In our implementation, we set the backoff value to be equal to its flow degree. Therefore, flows with smaller flow degree will always transmit before the flows with larger degree if there are no transmissions going on in its neighborhood (i.e., no RTS-CTS handshake is heard in its neighborhood). This effectively approximates the minimum-degree greedy algorithm.

At the start of the algorithm, flow degree discovery can be achieved by piggybacking the information in the initial packet several transmissions. If we adopt a global fairness model, discovering the flow degree will take at most $\sum_{f \in F} r_f$ packet transmissions considering the fact that it takes $\sum_{f \in F} r_f$ transmissions for flows in the basic channel to transmit at least each packet per flow.

The underlying MAC-layer support In our MAC-layer design, a sequence of RTS-CTS-DATA-ACK handshake is initiated for each data packet transmission, and this message exchange is preceded by a backoff of certain number of minislot times. When a node has a packet to transmit, it will also wait for an appropriate number of mini-slots (for flows with minimum scheduling order in the basic channel, its backoff value is zero; for flows in concurrent transmissions due to spatial reuse, its backoff is set to be the flow degree).

In general, the backoff period (before each flow's transmissions) will generate overheads for channel utilization. However, the period of each minislot can be set to be small (but larger than twice of the one-hop propagation delay). This may decrease the bandwidth overhead; besides, reducing the minislot size and increasing the backoff value (in terms of minislots) also help to reduce the probability of potential collisions among neighboring conflicting flows.

4.2.2 Information propagation via the conflict-free shared tree

When a new flow comes in or an existing flow terminates its transmission, if we adopt a global topology-independent fairness model, this flow information has to be propagated to all senders in the graph. To this end, the initiating flow will propagate this information to a pre-specified core node in the specific graph, and the core node will multicast this information to each sender in the network topology. In the multicast message, the core node will also include a TTL field (set to be equal to or more than the height of the tree). Upon receiving this message, each node records the TTL field and waits until its TTL expires and then updates its information accordingly. This way, nodes in the network graph can synchronize their information updates.

Our design goal is to propagate this information, *in minimum time*, from the core node to the rest of nodes in the network graph. This is equivalent to constructing a conflict-free minimum height spanning tree. We seek to build up a core-based shared tree that provides minimum time transmissions from the core node to all other nodes in the tree and ensures conflict-free concurrent delivery for sibling nodes at the same height of the tree.

Constructing conflict-free shared tree In this section, we give an overview of our conflict-free shared tree algorithm (a pseudo code is shown in Figure 6). In our algorithm, we start with a standard core-based shared spanning tree; this can be achieved by constructing the spanning tree for each node using the breadth-first search algorithm, and selecting the minimum-height spanning tree from these trees.

Given the spanning tree, we resolve collisions among sibling transmitting nodes through delaying packet transmissions along some branches of the tree (see Figure 4(c) for an example). For this purpose, each transmitting node maintains a delay counter C_d, which records the delay time for the packet transmissions in its branch.

We use a backoff-based mechanism to construct a conflict-free shared tree. We take an up-down approach (i.e., starting from the root node) and start from the nodes closest to the core node. Every transmitting node senses the channel and waits for a backoff number of minislots before initiating its RTS-DATA multicast message (note that no CTS or ACK is used due to the multicast nature of this problem, and DATA here means multicast message). We set the backoff value of a transmitting node to to be the difference between the height of the tree and the height of the current branch that the node belongs to. Therefore, the higher the branch, the smaller the waiting time. This way, we give priorities to the branches with larger height, and may delay the transmissions of other shorter branches in the presence of potential collisions. When a transmitting node hears either RTS or collisions, it increments its delay counter C_d by 1, thus delaying transmissions along its branch.

At the receiver side, a single receiver may be within the transmission range of multiple transmitters. In our algorithm, whenever it hears a collision, it broadcasts a NACK message to the senders. Upon receiving the NACK message, the transmitting nodes will randomly decide whether to increase their delay counter C_d by one or not.

```
$D_c$: delay counter of a node
$b_n$: backoff value of node $n$ in minislots

Sender side:
 $D_c = 0$;
 $b_n$ =tree_height-branch_height;
 while $b_n > 0$
     wait for one minislot;
     if(CLEAR_CHANNEL)
         $b_n \leftarrow b_n - 1$;
     else
         $D_c \leftarrow D_c + 1$;
         $b_n$ =tree_height-branch_height;
         wait until the completion of current transmission;
 transmit RTS and DATA;
 if (received NACK)
     update $D_c \leftarrow D_c + 1$ with probability $p$
Receiver side:
 if(hears a collision)
     broadcast a NACK message;
```

Figure 6. Pseudocode for conflict-free multicast tree

4.2.3 Further comments

In the implementation we present above, we set the backoff value for spatial reuse flows the same as their flow degree in the flow graph. If the flow degree is large, the backoff value will be large and this may potentially increase the waiting time overhead. Consider a simple case that the flow degrees in a flow contention graph will be $\{1, 5, 7, \ldots\}$ in the ascending order. Then according to the algorithm we described above, the waiting time (in terms of minislots) will be set to be $\{1, 5, 7, \ldots, \}$. However, a more efficient way to do this is to set the backoff value as $\{1, 2, 3, \ldots\}$, which we call normalized flow degree. To do this, if the global fairness model is adopted and the shared-tree is available, we may also propagate the flow degree information back to the core node, and the core node sorts the flow degree and propagates the normalized flow degree back to each flow. Then the flow can set the waiting time to be the normalized flow degree.

Another potential drawback of the implementation presented above is that the core code maintains per-flow information (i.e., flow's weight, etc.). Eliminating per-flow information management at the core node is possible through the following approach: the core node only maintains the aggregate flow information $\sum_{i \in F} r_i$. Upon receiving the aggregate flow information $\beta \sum_{i \in F} r_i$ from the core node where $\beta \geq 1$ is a positive number, each flow will generate r_i random numbers in the range of $[0, \beta \sum_{i \in F} r_i]$, as its local scheduling order for the basic channel. Definitely, the scheduling order generated this way is not guaranteed to be globally unique, thus multiple flows may seek to transmit simultaneously over the basic channel. However, as long as they are locally unique in a spatial neighborhood, it will not generally lead to collisions. Besides, the choice of β reflects a tradeoff between backoff efficiency and potential collision probabilities.

5 Simulations

In this section, we evaluate our algorithms by simulations. Some key features of our proposed algorithms are as follows: (local or global) fair share of the basic channel, maximum spatial reuse, minimum-time information propagation along the conflict-free multicast tree, and fully distributed implementation. In the following, we present three examples to illustrate the effect of these features.

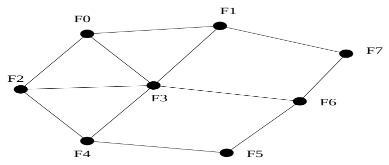

Figure 7: Example 1: Flow contention graph

We use the following performance measures to evaluate the algorithms. W_f^G : number of transmitted packets of flow f during the simulation lifetime by using the global topology-independent fairness model; W_f^L : number of transmitted packets of flow f during the simulation lifetime by using the local topology-dependent fairness model; Each of our simulations has a typical run of 100,000 time units. In all cases, we assume that the physical channel capacity C is one slot per time unit.

We present three simulation examples. Example 1 illustrates the features of ensuring a (local or global) fair share of the basic channel and additionally increasing spatial reuse. We also show that our algorithm results in larger total effective throughput than an algorithm that enforces strict fairness on the aggregate bandwidth (basic service plus spatial reuse) received by flows. Example 2 shows that our conflict-free multicast tree algorithm results in smaller delivery time to all nodes than a collision-unaware minimum-height spanning tree. We also compare the performances of the distributed implementation of local fairness model and global fairness model. Example 3 evaluates a complicated scenario of 36 nodes and 21 flows, and compares the throughput properties of the proposed distributed implementations.

Example 1: Features of the centralized algorithm In this example, we test the centralized algorithm of Section 3 in providing a basic fair share of the channel and increasing spatial reuse of bandwidth. The flow contention graph is shown in Figure 7. The results for infinite sources using the global fairness model and the local fairness model are shown in Table 1. We observe that each flow receives a fair share of the physical channel in proportional to its weight, and in addition receives spatial reuse. For this example, our algorithm achieves 300% of aggregate throughput (assuming that the throughput with the physical channel C is 100%), thus 200% spatial reuse gain, for both the global fairness model and the local fairness model. Note that even though the basic channel is fairly allocated among flows, spatial reuse is flow dependent and is not fair. Thus the aggregate service received by each flow is not generally in proportional to its flow weight. For the table, we observe that the local topology-dependent fairness model generally achieves a better fairness in terms of the aggregate service that each flow receives than the global topology-independent fairness model. In Table 1, we also give the result for an algorithm that always ensures absolute fairness (in proportional to flow's weights); as a consequence of strict fairness enforcement, we can achieve only 240% effective throughput, with spatial reuse gain to be 140%.

Example 2: Distributed implementation In this example, we evaluate the distributed implementation described in Section 4. The node graph and flow contention graph for this example are shown in Figures 8 and 9. We first evaluate the algorithm that seeks to construct the conflict-free minimum spanning tree for the node graph. The tree built

Flow	r_f	W_f^L	W_f^G	W_l (absolute fairness)
0	3	44907	60526	60000
1	1	25076	14593	20001
2	1	27737	21156	20000
3	1	20011	15440	20003
4	2	29685	19339	40012
5	1	63510	78733	20006
6	1	29165	14089	20006
7	2	59909	76124	40012

Table 1: Ex.1: Comparisons of three fairness models

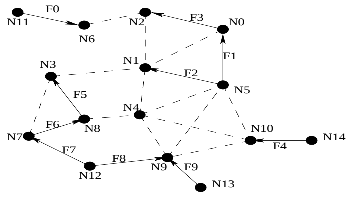

Figure 8: Ex 2: Node graph

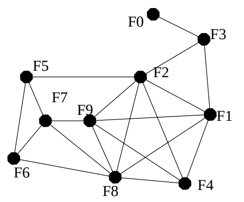

Figure 9: Flow contention graph

using our algorithm is shown in Figure 10. We also plot the standard minimum-height spanning tree in Figure 11. As a result, the total transmission time saves 2 units, reducing from 6 units to 4 units.

Next we compare the performance of the local fairness model and the global fairness model in the distributed implementation. We choose all ten flows to be infinite sources. The results are shown in Table 2. In this scenario, our distributed implementation achieves an aggregate throughput of 300% in both cases. However, the local fairness model achieves better fairness (measured according to the flow's weight) in terms of the aggregate service (i.e., the service received in the basic channel plus the service due to spatial reuse) than the global topology-independent fairness model.

Example 3: A more complex scenario In this scenario, we evaluate a scenario with 36 nodes and 21 flows, as shown in Figures 12 and 13. Figures 15 and 14 show the standard spanning tree and the conflict-free spanning tree. Tables 3 show the service received by each flow (of infinite arrivals) in both the global fairness model and the local fairness model. The total effective throughput is 683.1% for the local fairness model and is 686.4% for the global fairness model. In this case, we can see that the algorithm using topology-independent fairness model usually results in higher aggregate throughput, thus higher spatial reuse, but the fairness property for the aggregate service is less favorable than in the topology-dependent local fairness model.

Flow	r_f	W_f^L	W_f^G
0	3	64196	81731
1	2	11529	9353
2	1	15189	10571
3	1	35804	18269
4	2	49108	59961
5	1	55805	72767
6	1	33555	19402
7	2	10640	7831
8	3	8911	9620
9	1	15263	10495

Table 2: Example 2: Distributed Implementation

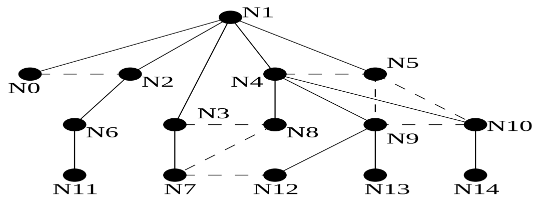

Figure 10: Example 2: Conflict-free tree

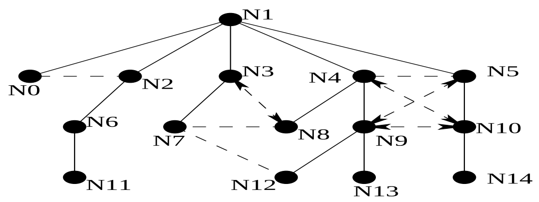

Figure 11: Example 2: Standard spanning tree

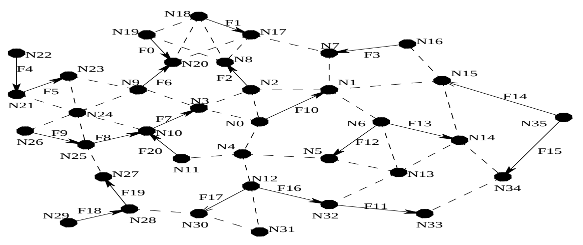

Figure 12: Node Graph of Example 3

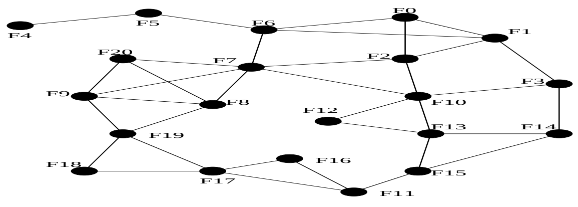

Figure 13: Flow contention graph of Example 3

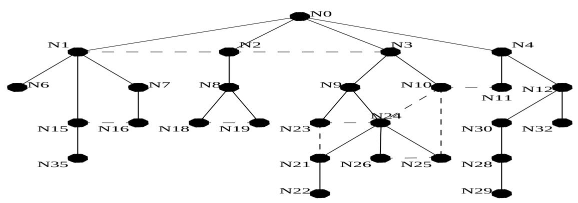

Figure 14: Example 3: Conflict-free tree

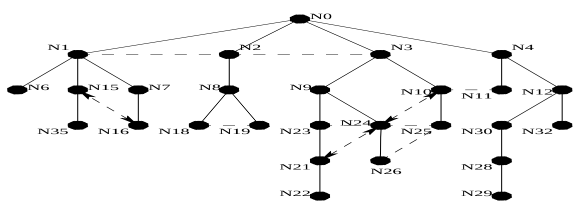

Figure 15: Example 3: Standard spanning tree

Flow ID	r_f	W_f^L	W_f^G
0	1	19940	4625
1	6	56880	87414
2	4	21706	7724
3	1	20412	3300
4	1	31163	5872
5	7	68837	94128
6	3	13996	3661
7	6	13463	6723
8	6	15697	6357
9	7	12301	6619
10	6	11488	6150
11	6	61101	88922
12	1	67602	87396
13	6	20910	6454
14	5	51376	82960
15	1	16060	3164
16	1	21484	4615
17	6	17415	6463
18	1	64692	86646
19	6	17893	6891
20	1	58539	80301

Table 3: Performance of Example 3: distributed implementation

6 Discussions and Related Work

6.1 Further Issues

In previous sections, we present the basic design of the proposed packet scheduling model and the packetized algorithm as well as its implementation, we now return to discuss a few aspects in more details.

Variable packet size In the design of our algorithm in Sections 3 and 4, we assumed that each packet has a fixed packet size, which is a realistic assumption in typical wireless scenarios. However, if packets do have variable size in some atypical scenarios, we still have a partial solution in the algorithm of Section 3. In essence, variable packet size complicates concurrent packet transmissions; and computation of maximum independent set. When multiple packets are transmitted simultaneously through spatial reuse of the physical channel, if the HOL packets of multiple non-contending flows have different lengths, these packets will take different amount of time to finish transmissions. In the extreme case, a large-packet-size flow may capture more capacity than flows with smaller packet sizes. One solution is to maintain a "credit/debit" (in bits) for each flow to account for the actual service (in bits) that each flow receives, and then modify the scheduling and adaptive coloring algorithms accordingly. The algorithm for a standard maximum independent set approximation can also be adapted to handle variable packet size, or be formulated as a nonlinear programming problem. Due to lack of space, we do not discuss the details here.

Multihop flows Packet flows in a multihop wireless network may traverse multiple hops to reach their destinations. In our proposal, we break each multihop flow into multiple single-hop flows, and each one-hop flow is handled by its local sending/forwarding node. This is identical to what has been done for multihop flows in wireline packet scheduling; anyway, packet scheduling is a per-hop behavior.

Handling mobility In a multihop wireless network, communicating nodes can be mobile, thus changes of the network topology may be frequent. Note that both our proposed local topology-dependent fairness model and global topology-independent fairness model apply well in the mobile environment. However, frequent node mobility may change the core-based conflict-free shared multicast tree significantly. Fortunately, if the events of flow joins and leaves are not frequent, this will not become a serious issue.

Another related issue is scalability of the proposed algorithm. In general, we do not believe that scalability is a main concern for typical wireless networking scenarios where the total number of nodes is still relatively small, as well as the number of flows in a bandwidth-constrained wireless scenario. However, we do intend to carefully investigate this issue in the future. Finally, we plan to carefully study the issue of interaction between our proposed scheduling model and the underlying MAC layer protocol support.

6.2 Related Work

Packet scheduling has been the subject of intensive study in the networking literature and numerous algorithms have been proposed, among which are WFQ [11], WF^{2}Q [17] and STQ [18], etc.. In recent years, there are several research efforts on adapting fair packet scheduling to wireless cellular networks, notably IWFQ [3], CIF-Q [19], SBFA [20] and WFS [9]. The goal of these wireless fair scheduling algorithms has been to hide short bursts of location-dependent channel errors from well-behaved flows by dynamically swapping channel allocations between backlogged flows that perceive channel errors and backlogged flows that do not, with the intention of reclaiming the channel access for the former when it perceives a clean channel. Therefore, lagging flows (that lag behind their error-free reference service due to channel errors) receive compensation from leading flows. The proposed algorithms differ in terms of how the swapping occurs, between which flows the swapping takes place, and how the compensation model works.

In multihop wireless networks, providing minimum throughput bounds and bounded delay access has been studied at the MAC layer [10, 2, 4]. A popular approach has been to establish transmission schedules and allocate stations to different time slots of a TDMA cycle in a way that no collisions occur. The design goal is to design conflict-free link scheduling schemes that seek to maximize the spatial reuse of the bandwidth and remain immune to topological changes in a mobile ad hoc networking environment. Another study [7] also investigates the fair link activation problem in such a network. However, all these previous studies seek to provide throughput bounds or weighted fairness for wireless *links*, not for packet *flows*; hence, they do not address the problem of packet scheduling of packet flows. Besides, these algorithms tend to work with a fixed TDMA cycle, and do not have the dynamic scheduling feature. Furthermore, the focus of these MAC-layer studies has been on the mechanisms of channel access by assuming that the packet scheduling algorithm has been worked out, rather than the other way around. Finally, these works do not consider the problem of arbitrating fairness and maximal channel utilization.

There are two recent works that also address fairness issues in multihop wireless networks [21, 22]. In [21], the authors have studied the problem of distributed fair queueing in multihop wireless networks. However, the focus of [21] is to ensure fairness by adapting the fair queueing algorithm to these networks, and it does not make explicit efforts to maximize spatial reuse subject to fairness constraints. In [22], the authors seek to design novel MAC-layer supporting mechanisms for any pre-specified fairness model, and the design focus there is how to achieve a given fairness model through appropriate MAC layer designs.

7 Conclusion

In this paper, we have proposed new packet scheduling models for an multihop wireless network, and our model ensures fair allocation of basic channel service while seeking to maximize spatial reuse. We then describe a packetized algorithm that realizes the scheduling model with analytically provable performance bounds. We further design a backoff-based distributed implementation which closely emulates the ideal centralized algorithm. We demonstrate the effectiveness of our proposed algorithm through both simulations and analysis. Ongoing work seeks to improve the design of the distributed implementation, to perform more extensive simulations, and to refine the analytical bounds of the proposed algorithm.

References

[1] C. Chang, J. Chang, K. Chen and M. You, "Guaranteed quality-of-service wireless access to ATM," *IEEE JSAC*, 1997.

[2] J. Ju and V.O.K. Li, "An optimal topology-transparent scheduling method in multihop packet radio networks," *IEEE Trans. Networking,* 6(3), June 1998.

[3] S. Lu, V. Bharghavan and R. Srikant, "Fair scheduling in wireless packet networks," *IEEE Trans. Networking*, August 1999.

[4] I. Chlamtac and A. Farago, "Making transmission schedules immune to topology changes in multi-hop packet radio networks," *IEEE/ACM Trans. Networking,*, 2(1), February 1994.

[5] S. Choi and K. G. Shin, "Predictive and adaptive bandwidth reservatoin for hand-offs in QoS-sensitive cellular networks," *ACM SIGCOMM'98*, 1998.

[6] Anup Kumar Talukdar, B. R. Badrinath, "Rate adaptation schemes in networks with mobile hosts," *ACM MOBICOM'98*, 1998.

[7] I. Chlamtac and A. Lerner, "Fair algorithms for maximal link activation in multihop radio networks," *IEEE Trans. Communications,* 35(7), July 1987.

[8] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A medium access protocol for wireless LANs," *ACM SIGCOMM'94*, 1994.

[9] S. Lu, T. Nandagopal, and V. Bharghavan, "Fair scheduling in wireless packet networks," *ACM MOBICOM'98*, October 1998.

[10] Z. Tang and J.J. Garacia-Luna-Aceves, "A protocol for topology-dependent transmission scheduling in wireless networks," *WCNC'99*, September 1999.

[11] A. Demers, S. Keshav and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM'89*, August 1989.

[12] A. Parekh, "A generalized processor sharing approach to flow control in integrated services networks," *PhD Thesis*, MIT Laboratory for Information and Decision Systems, Technical Report LIDS-TR-2089, 1992.

[13] P. Crescenzi and V. Kann, "A compendium of NP optimization problems," *http://www.nada.kth.se/ viggo/index-en.html.*

[14] V. Bharghavan, "A new protocol for medium access in wireless packet networks," *online document,* 1999.

[15] M. M. Halldorsson and J. Radhakrishnan, "Greed is good: Approximating independent sets in sparse and bounded-degree graphs," *ACM STOC'94*, 1994.

[16] J. Kuri and S. Kasera, "Reliable multicast in multi-access wireless LANs," *IEEE INFOCOM'99*, 1999.

[17] J.C.R. Bennett and H. Zhang, "WF^{2}Q: Worst-case fair weighted fair queueing," *IEEE INFOCOM'96*, 1996.

[18] P. Goyal, H.M. Vin and H. Chen, "Start-time fair queueing: A scheduling algorithm for integrated service access," *ACM SIGCOMM'96.* August 1996.

[19] T.S. Ng, I. Stoica and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," *IEEE INFOCOM'98*, March 1998.

[20] P. Ramanathan and P. Agrawal, "Adapting packet fair queueing algorithms to wireless networks," *ACM MOBICOM'98*, October 1998.

[21] N. H. Vaidya and P. Bahl, "Fair scheduling in broadcast environments," *Microsoft Research Tech. Rep. MSR-TR-99-61.*

[22] T. Nandagopal, T. Kim, X. Gao and V. Bharghavan, " Achieving MAC Layer Fairness in Wireless Packet Networks." *to appear in Mobicom 2000*, Boston, MA, August 2000.