Self-coordinating Localized Fair Queueing in Wireless Ad Hoc Networks

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

Abstract

Distributed fair queueing in a multihop, wireless ad-hoc network is challenging for several reasons. First, the wireless channel is shared among multiple contending nodes in a spatial locality. Location-dependent channel contention complicates the fairness notion. Second, the sender of a flow does not have explicit information regarding the contending flows originated from other nodes. Fair queueing over ad-hoc networks is a distributed scheduling problem by nature. Finally, the wireless channel capacity is a scarce resource. Spatial channel reuse, i.e., simultaneous transmissions of flows that do not interfere with each other, should be encouraged whenever possible. In this paper, we re-examine the fairness notion in an ad-hoc network using a graph-theoretic formulation, and extract the fairness requirements that an ad-hoc fair queueing algorithm should possess. To meet these requirements, we propose Maximize-Local-Minimum Fair Queueing (MLM-FQ), a novel distributed packet scheduling algorithm where local schedulers self-coordinate their scheduling decisions and collectively achieve fair bandwidth sharing. We then propose Enhanced MLM-FQ (EMLM-FQ) to further improve the spatial channel reuse and limit the impact of inaccurate scheduling information resulted from collisions. EMLM-FQ achieves statistical short-term throughput and delay bounds over the shared wireless channel. Analysis and extensive simulations confirm the effectiveness and efficiency of our self-coordinating localized design in providing global fair channel access in wireless ad-hoc networks.

keywords

Localized algorithm, Packet scheduling, MANET fair queueing

1 Introduction

Wireless ad-hoc networks offer convenient, infrastructure-free data communication services to wireless users. Emerging applications for the ad-hoc networking technology include distributed sensor networks, zero-configuration teleconferencing during disaster relief and emergency operations, and ubiquitous data communications in the battlefield. In such scenarios, users exchange information via communication-intensive applications, such as Web browsing, video conferencing and file transferring. Therefore, the issue of providing fair and delay-bounded wireless channel access, notably through effective packet scheduling, has come to the fore.

Fair queueing has been a popular scheduling paradigm in both wireline networks [1, 6, 8, 19, 24] and packet cellular networks [20, 14]. However, these algorithms do not directly apply in a wireless ad-hoc network due to *location-dependent channel contention*, *spatial channel reuse*, and *incomplete scheduling information*. Since wireless transmissions are local broadcast, contention for the shared channel is location dependent. This implies that packet scheduling is no longer a *local* decision at the sender, as opposed to that in the wireline or packet cellular networks. Instead, a node has to consider the scheduling decisions made by other neighboring nodes that share the same wireless channel. Moreover, a local scheduler implemented at a single node does not have explicit flow information, e.g., the packet arrival time and the flow status, regarding the contending flows originated from its neighbors. Therefore fair queueing in a wireless ad-hoc network is a distributed scheduling problem by nature. Finally, the wireless channel capacity is limited. Improving channel utilization through spatial channel reuse, i.e., simultaneously scheduling flows that do not interfere with each other, is highly desirable.

It turns out that, even the notion of *fairness* has to be carefully defined in a wireless ad-hoc network. In the popular fluid fairness model used in wireline fair queueing, a scheduler is implemented at each node and packet flows over its output link are modeled as fluid flows through a channel of capacity C. Each flow is assigned a weight r_f. Over any infinitesimally small time window $[t, t + \Delta t]$, a backlogged flow f is allocated a capacity of $C\Delta t(r_f / \sum_{i \in B(t)} r_i)$, where $B(t)$ is the set of backlogged contending flows at the given node. However, this fairness model does not directly apply in a

wireless ad-hoc network. Since contention is location dependent, each node's contending flow set is tailored to its neighboring nodes. In the worst-case, *all flows in a wireless ad-hoc network may be "connected" with direct or indirect contending relationship, and fairness has to be defined with respect to all the flows in the entire network*. Consequently, fair queueing in wireless ad-hoc networks is no longer a local activity at a specific output link and has to comply with global fairness requirements.

In this paper, we first re-examine the fairness notion in an ad-hoc network, and propose a new fairness model that captures the features of location-dependent contention and spatial reuse. The fair share of each packet flow is defined with respect to the corresponding flow contending graph. Each one-hop flow always receives a fair share in the bottleneck area of the network, represented by the maximal clique in the flow contending graph. We then propose two novel algorithms called Maximize-Local-Minimum Fair Queueing (*MLM-FQ*) and Enhanced MLM-FQ (*EMLM-FQ*). MLM-FQ identifies the flows that currently receive minimal services in their locality, and ensures their access to the wireless channel. The scheduling coordination in MLM-FQ is localized within the flow's one-hop neighborhood in the flow contending graph. EMLM-FQ further enhances MLM-FQ along three dimensions. First, it offers larger fair share to each flow, characterized by the fair share in the maximum clique of the flow contending graph. Second, it is more resilient against incomplete and/or erroneous scheduling information caused by collisions. Finally, EMLM-FQ realizes delay and throughput decoupling, leading to more efficient utilization of the wireless channel among applications with different delay/bandwidth requirements. Extensive simulation results demonstrate the effectiveness and efficiency of our localized and fully-distributed design.

The rest of the paper is organized as follows. Section 2 identifies the key design challenges and proposes our fairness model. Section 3 presents and analyzes MLM-FQ and EMLM-FQ. Section 4 presents a distributed implementation of EMLM-FQ within the CSMA/CA paradigm. Section 5 evaluates the performance of our implementation through simulations. We discuss several design issues in Section 6 and compare with the related work in Section 7. Finally Section 8 concludes this paper.

2 Models and Issues

In this section, we first describe the network model and present a graph-theoretic formulation of the fair queueing problem. We then identify the design challenges and propose a fairness model for wireless ad-hoc networks.

2.1 Network Model

We consider a packet-switched multihop wireless network in which a single physical channel with capacity C is shared among contending nodes. Transmissions are local broadcast and only nodes within the transmission range of the sender can receive the packets. A flow is defined as a stream of packets from the source to the destination.

We assume that the underlying MAC layer follows the popular CSMA/CA paradigm [3, 10]. Each transmission is preceded by a control handshake. A sender that intends to send a packet transmits a short control message RTS (Request-to-Send) to the receiver. If it is not constrained by any nearby on-going transmissions, the receiver responds with a CTS (Clear-to-Send) message. The sender then transmits DS (Data-Sending) on receipt of CTS, announcing among the sender's neighborhood the success of the RTS-CTS handshake. The DATA packet is transmitted after the DS message. Finally the receiver confirms the receipt of the DATA packet by replying with an ACK message. Sender's one-hop neighboring nodes that overhear the RTS message are constrained from transmission until the end of the CTS message. Both sender's and the receiver's one-hop neighboring nodes, overhearing the DS and CTS messages respectively, are constrained from transmitting until the end of the ACK message.

Based on the above channel access mechanism, the nodes in the neighborhood of both the sender and the receiver must defer their transmissions. Two flows are contending with each other if either the sender or the receiver of one flow is within the transmission range of the sender or the receiver of the other [3]. We further make the following three assumptions [3, 10, 21, 17]: (a) A collision occurs when a receiver is in the reception range of two transmitting nodes, thus unable to decode the signal from either of them. We ignore capture effect in this work. (b) A node cannot transmit and receive packets simultaneously. (c) Neighborhood is a commutative property. Hence, flow contention is also commutative.

2.2 Flow Contending Graph

We use the node graph to represent the network topology. Each node is represented by a vertex in the node graph, and any two nodes within the transmission range of each other are connected with an edge. Each flow is marked with an arrow from its sender to its receiver. In the example of Figure 1, six nodes (N_i, $i = 0, \cdots, 5$) are placed in a linear topology, and we have five flows (F_j, $j = 0, \cdots, 4$), where the sender and the receiver of F_j are N_j and N_{j+1}, respectively.

We derive the *flow contending graph* from the node graph. In a flow contending graph, each vertex represents a backlogged flow, and an edge between two vertexes denotes that these two flows are *contending* with each other. If two vertexes are not connected directly with an edge, these two flows

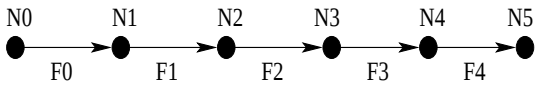


Figure 1: Node Graph

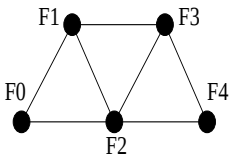


Figure 2: Flow Contending Graph

can transmit simultaneously for spatial reuse of the wireless channel. For example, Figure 2 shows the flow contending graph derived from the node graph in Figure 1.

2.3 Location-dependent Contention

In a wireless ad-hoc network, the contending flow set is flow specific. For two contending flows, their contending flow sets may be different due to their spatial locations. In the example of Figure 2, the contending flow set of F_0 is $\{F_1, F_2\}$, while the contending flow set of F_1 is $\{F_0, F_2, F_3\}$. Location-dependent contention has the following two major impacts on fair queueing:

First, for two flows that contend for the shared wireless channel, conventional fluid fairness model defined for wireline networks or packet cellular networks does not directly apply. There does not exist well-defined resource (e.g., the outgoing wireline or wireless link with certain bandwidth), or a consistent set of competing flows, over which the fairness can be defined. In fact, even the number of competing flows in the contending flow set may vary from one flow to another.

Second, for two flows that do not directly contend with each other, they may be "connected" by other flows that contend with both of them. In the above example, F_0 and F_3 can potentially be scheduled simultaneously without interfering each other. However, they both contend with F_1 (and F_2). If a fair share is allocated for each flow, then F_0's share has to be defined with respect to F_1's, while F_1's share has to be "fair" compared with F_3's. The consequence of this coupling is that F_0's fair share of the wireless channel capacity has to be defined with respect to F_3's, even though they do not directly contend with each other! It is easy to see that the generalization of this indirect contending relationship may connect all flows in the network, and the fairness has to be defined globally over the entire flow set.

2.4 Spatial Channel Reuse

Spatial channel reuse may greatly improve the channel utilization and the aggregate throughput of the network. However, it further complicates the fairness notion. We again use the example of Figure 2 for an illustration. Assume all flows have equal weight, and $T_0 < T_1 < T_2 < T_3 = T_4$ where T_i is the service tag for flow F_i, i.e., the aggregate amount of traffic that F_i is served during the current backlog period. Accordingly, F_0 should be scheduled for transmission as it receives the minimum service, while neither F_1 nor F_2 should be scheduled to avoid collisions with F_0. For flows F_3 and F_4, although they do not interfere with F_0, neither of them can be scheduled without violating the strict fairness requirement. This example shows the potential conflict between fairness and spatial channel reuse. Enforcing perfect fairness may lead to lower channel utilization, while spatial channel reuse may result in larger unfairness bound.

2.5 Distributed Flow State

Because a single wireless channel is shared among senders within spatial locality, a local scheduler does not have complete information of the contending flows originated from neighboring nodes. For example in Figure 2, F_0 contends with both F_1 and F_2. However, the sender of F_0 (node N_0) does not have either F_1 or F_2's current flow states, e.g., which flow has outstanding packets, when packets for a particular flow arrive and how many packets are waiting in the queue. Similarly, the sender of F_2 does not have the flow state of F_0 either. Moreover, if fairness has to be defined over potentially all flows in the entire network, global flow state information may be required. This information is obviously very costly to obtain in a wireless ad-hoc network even of medium size (e.g., several hops wide), if not impossible [17]. There is no single infrastructure point, e.g., the base station, where such complete information is readily available.

2.6 Fairness Model

In summary, we extract the following desired requirements for fair queueing in a wireless ad-hoc network:

- Location-dependent contention results in indirect contention relationship among flows in a large area. Any scheduling decision made at a node may generate global effects. The notion of fairness must be properly defined to ensure performance isolation among contending flows.
- Wireless channel is a scarce resource; spatial channel reuse should be encouraged whenever possible. However, this should not be achieved at the cost of arbitrary violation of flow performance isolation and fairness.
- Wireless ad-hoc networks lack infrastructure support.

The fair queueing design must be fully distributed and localized, without any non-local communications or computation involved.

To satisfy the above requirements, we define the fairness model as follows.

1. For all backlogged flows in a connected flow contending graph, the flow that receives minimum service, normalized to its weight, should be guaranteed to receive service. That is, each flow should be granted a minimum fair share of the wireless channel bandwidth.
2. Subject to the minimum fair share and channel bandwidth constraints, simultaneous transmissions should be scheduled to increase the wireless channel utilization. A flow located in a lightly contended area, i.e., with a small number of contending flows, may benefit more from spatial channel reuse than a flow located in a heavily contended area, i.e., with a large number contending flows.

Two observations can be made on the above fairness model. First, the fairness is defined over all flows in a *connected* flow contending graph. In a large network where flows are "clustered" into different localities – resulting multiple disconnected flow contending subgraphs – the fairness is defined over each connected subgraph independently. Second, the fairness model does not prohibit spatial channel reuse as long as the spatial reuse is scheduled in an order that does not conflict with the scheduling of the flow that receives minimum services. We show in the next section how the above fairness model enables localized implementation to achieve both efficiency and scalability.

3 A Self-coordinating Approach

In this section, we propose and analyze two novel distributed and localized fair queueing algorithms to realize the fairness model defined in Section 2.6. Both algorithms are self-coordinating in the sense that each local scheduler coordinates with its local neighbors to meet the desired global fairness requirements. The coordination effort is also fully localized in the sense that only local computation and local information propagation are involved.

3.1 Overview

In our algorithms, each node is responsible for assigning tags and scheduling flows that originate from itself. At the same time, it also maintains a table that records service tags for other flows in its one-hop neighborhood. We use Start-time Fair-queueing (SFQ) [8] to assign start tags and finish tags for the flows because of its ease in virtual time maintenance. In each table entry, we record the following information: $[flow_{id}, flow_{tag}]$, where the $flow_{tag}$ is the most recent service tag that the node is updated for flow $flow_{id}$.

Our proposed algorithms are based on the following two mechanisms:

- *Maximizing local minimum* by transmitting flows with local minimum service tags: a node immediately transmits a flow only if this flow has the minimum service tag $flow_{tag}$ among all backlogged flows in its table.
- *Using backoff mechanism to increase spatial reuse*: if a flow does not have the local minimum service tag in its one-hop flow neighborhood, we set a backoff timer. If the backoff timer expires and the channel is idle, the flow will transmit. Furthermore, we tailor the flow's backoff value to both the flow's local fairness (i.e., its received service compared to its neighbors) and its local contention degree in the flow contending graph (i.e., the number of contending flows in its neighborhood).

3.2 Maximize-local-minimum Fair Queueing

As defined in the first requirement of fair queueing in Section 2.6, the flow with the global minimum service tag must be selected for transmission prior to other flows. Note that this is non-trivial in a wireless ad-hoc network, since identifying the flow with the global minimum service tag in general requires sorting the service tags of all flows. Therefore, how to achieve this goal using only the local information and performing only local computation becomes a severe design challenge.

In this paper, we take a novel approach to identifying the flow with global minimum service tag. Since identifying a global minimum involves global search, we identify all flows with *local* minimum service tags, and schedule all such flows for transmission. This comes the name of our first algorithm: Maximizing-Local-Minimum Fair Queueing (MLM-FQ). As far as the service tag is concerned, since the global minimum must be a local minimum (but not *vice versa*), we know that the flow with the global minimum tag must be among these flows that have local minimum tags and it is guaranteed to be transmitted first. Hence, "maximizing local minimum" policy is a superset of the "maximizing global minimum" policy.

We can also draw our model to an analogy of the 3-dimensional model shown in Figure 3, which illustrates 2-D in the spatial domain, and 1-D in the temporal domain. Our proposed model is similar to the phenomenon that a rain storm washes a three-dimensional terrain: the water always fills in those lowest (local) spots first, marked with a Δ sign in the figure.

The detailed operations consist of four parts:

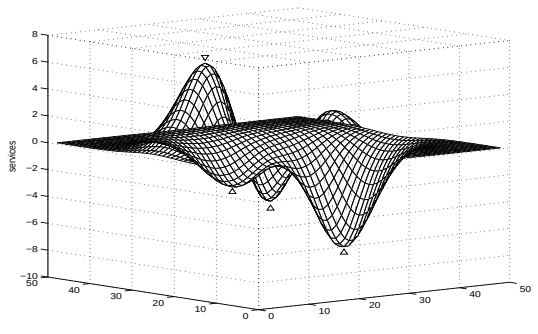
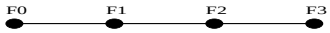


Figure 3: A 3-D Distributed Fair Queueing Analogy



	S0 0	S1 1	S2 2	S3 3
F0 Transmit Pkt Size: 103	103	1	2	3
F1 Transmit Pkt Size: 101	103	102	2	3
F2 Transmit Pkt Size: 99	103	102	101	3
F3 Transmit Pkt Size: 97	103	102	101	100
F3 Transmit Pkt Size: 103	103	102	101	203
F2 Transmit Pkt Size: 101	103	102	202	203

Figure 4: Worst-case MLM-FQ minimum fair share

1. *Local state maintenance*: Each node n maintains a local table E_n, which records the current service tags for all flows in its one-hop neighborhood of the flow contending graph. Each table entry has the form of $[f,\ T_f]$, where T_f is the current service tag of flow f, i.e., the start tag of flow f.

2. *Tagging operations*: For each flow f in the local table, we simulate the SFQ algorithm to assign two tags for each arriving packet: a start tag and a finish tag. Specifically, for the head-of-line packet k of flow f, with arrival time $A(t_k^f)$ and packet size L_p, its start tag S_k^f and finish tag F_k^f are assigned as follows:

   (a) If f is continually backlogged, then $S_k^f = F_{k-1}^f;\ \ F_k^f = S_k^f + L_p/r_f$.

   (b) If f is newly backlogged, then $S_k^f = \max_{g\in E_n}\{V_g(A(t_k^f))\};\ \ F_k^f = S_k^f + L_p/r_f$, where $V_g(t)$ is flow g's virtual time at t.

3. *Scheduling loop*: At node n, whenever it senses the channel clear,

   (a) If one of its own flows, say f, has the *smallest* service tag in E_n, transmit the head-of-line packet of flow f immediately;

   (b) Otherwise, node n backs-off with a timer to break potential deadlock.

4. *Table updates*: whenever node n hears a new service tag $T_g^{'}$ for any flow g on its table E_n, it updates the table entry for flow g to $[g, T_g^{'}]$. Whenever node n transmits a head-of-line packet for flow f, it updates flow f's service tag in E_n, and piggybacks the service tag in the handshake messages.

3.2.1 Analytical Properties of MLM-FQ

The analytical properties of MLM-FQ are listed as follows. Due to space constraints we only outline the proof.

Proposition 3.1 *(Bounded Unfairness of MLM-FQ) Assuming consistent table states. In a connected flow contending graph, if each network node adopts the MLM-FQ, then for any two backlogged flows f and g their received services $W_f(t_1,t_2)$ and $W_g(t_1,t_2)$ during time interval $[t_1,t_2]$ satisfy:*

$$|\frac{W_f(t_1,t_2)}{r_f} - \frac{W_g(t_1,t_2)}{r_g}| < \Delta$$

where r_f is flow f's weight, and Δ is a topology-dependent constant.

Proof: The proof is based on mathematical induction for any two nodes in the connected node graph, and the fact that a flow receiving global minimum (normalized) service must be a local minimum. □

Corollary 3.1 *(Long-term Fairness of MLM-FQ) Assuming consistent table states, for any continually backlogged flow f, MLM-FQ achieves long-term fairness for flow f:*

$$\lim_{t\to\infty} \frac{W_f(0,t)}{t} = r_f/k_f$$

where k_f is a topology-dependent constant.

Proposition 3.2 *(Minimum Fair Share) Assuming consistent table states, MLM-FQ guarantees that each continually backlogged flow f receives a minimum fair share of the channel C. That is,*

$$W_f(t_1,t_2) \geq C\frac{r_f}{k\sum_{g\in\mathcal{N}} r_g}(t_2-t_1) - \alpha$$

where k and α are two topology-dependent constants, and $\mathcal{N}$ denotes all flows in the connected flow contending graph.

Proof The proof is based on induction in the connected graph and calculations derived from SFQ [8]. □

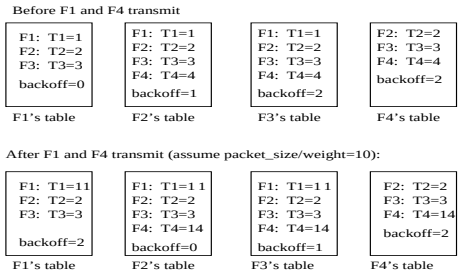


Figure 5: Potential Collision of Service Tag Propagation

Note that the minimum fair share analyzed above is for the worst-case scenario, where no spatial channel reuse is allowed if the channel reuse results in violation of the MLM-FQ fairness requirement. We illustrate such scenario through the example of the flow contending graph in Figure 4. Suppose all flows F_i have equal weights of 1.0, and their service tags start with $S_i = i$ ($i = 0, \cdots, 3$). Then F_0 has the local minimum service tag and will be scheduled at T_0. Even though neither F_2 nor F_3 interferes with F_0, they can be scheduled simultaneously with F_0 at T_0 because they do not possess local minimum service tags. The service tag of F_0 is updated as 103 after its packet (with a size of 103) is transmitted. Now F_1 has the smallest service tag, transmits its packet (with a size of 101) at T_1, and updates its service tag as 102. F_2 and F_3 transmit their packets at T_2 and T_3, respectively. The example shows that only a single flow has the local minimum service tag at a given time instant from T_0 to T_5. MLM-FQ dictates that only the single flow with the minimum service tag can be scheduled, therefore no spatial channel reuse can be achieved in this scenario. Consequently, each flow receives only the minimum fair share as defined in proposition 3.2. In practice, this worst-case scenario may not last very long in a random network topology with random flow contending graph and random packet sizes, but it may happen.

3.2.2 Impact of Inconsistent Table States

So far we assume the states maintained in local tables are consistent. That is, a node maintains updated service tags for all flows in its one-hop neighborhood. This assumption, however, may not always hold due to collisions. We use the example in Figure 5 to illustrate this problem.

Assume that the current flow tags T_i's for flow F_i's be $T_1 = T_4 = 0.5$, $T_2 = T_3 = 1$. Further assume identical weights and unit-size packets for all four flows. MLM-FQ will schedule F_1 and F_4 concurrently, since they have local minimum service tags. When they transmit their head-of-line packets over the broadcast wireless channel, they piggyback their new service tags, i.e., $T_1' = T_4' = 1 + 0.5 = 1.5$, to update the table entries maintained at the senders of F_2 and F_3. As we can see in our later implementation in Section 4, the new service tag is piggybacked in DS and ACK control messages. Although these two control messages are small, they still may collide at the sender of F_2 or F_3, leading to inconsistent table states. After the transmissions of F_1 and F_4, the table maintained at the sender of F_1 will be $\{T_1 = 1.5, T_2 = T_3 = 1\}$, and the table maintained at the sender of F_4 will be $\{T_4 = 1.5, T_2 = T_3 = 1\}$. Therefore, they will both backoff to yield the shared wireless channel to F_2 and F_3. However, due to the collision of the service tag propagation, the tables maintained at the senders of F_2 and F_3 are still $\{T_1 = T_4 = 0.5, T_2 = T_3 = 1\}$. Because of the stale table entries $T_1 = T_2 = 0.5$, the senders of F_2 and F_3 will continue to backoff, trying to yield the wireless channel to F_1 and F_4. A deadlock occurs and lasts until the deadlock detection timers expire.

Channel access deadlock due to inconsistent table states results in the waste of the wireless channel resource. We enhance the basic MLM-FQ algorithm to solve this problem as described next.

3.3 Enhanced MLM-FQ

From the rain storm analogy shown in Figure 3, we see that if water only fills in the lowest local spots, it takes a long time to make the entire terrain drenched by the water and become "flat" eventually. Similar phenomenon can be observed from distributed fair queueing: MLM-FQ policy alone may result in very low aggregate network utilization in a large network topology (Proposition 3.2). Inspired by the observation that some water will also rinse higher-altitude terrains while flowing into the lowest spots, we increase spatial channel reuse in fair queueing for wireless ad-hoc networks the same way. To this end, we simultaneously schedule other non-interfering flows, based on the second requirement in our fairness model (Section 2.6). We use a backoff-based mechanism to achieve the ordered spatial channel reuse.

Specifically, for any flow f, *we set its backoff value to be the total number of flows with smaller service tags (i.e., start tags)*. We call this mechanism Enhanced Maximize-Local-Minimum Fair Queueing (EMLM-FQ). Note that for flows with local smallest service tag, they are scheduled with zero backoff, thus the property of "maximize-local-minimum" still holds. As we analyze later, EMLM-FQ actually provides a higher statistical minimum service for each flow than MLM-FQ. In EMLM-FQ, higher priorities are given to flows that locate in a less contended area, and the backoff is tailored to the current contention level in a flow's current location, i.e., the number of contending flows.

The detailed operations of EMLM-FQ differ from MLM-FQ only at the scheduling loop:

3 *Scheduling loop*: At node n, whenever it senses the channel clear,

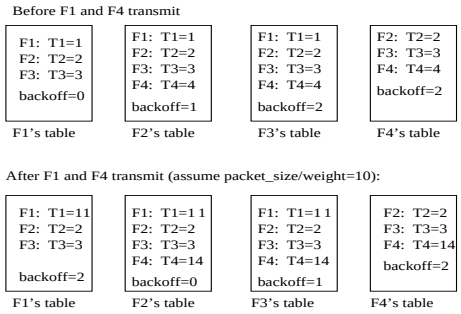


Figure 6: How EMLM-FQ Works

(a) If one of its own flows, say f, has the *smallest* service tag in E_n, transmit the head-of-line packet of flow f immediately;

(b) Otherwise, for each flow f with n as its sender, if $T_f > V_{\min}$, where $V_{\min}$ is the virtual time at node n, defined as the minimum service tag in E_n, set the backoff period B_f of flow f as $B_f = \sum_{g \in E_n} I(T_g(t) < T_f(t))$, plus cw minislots. $I(x)$ denotes the indicator function, i.e., $I(x) = 1$ if $x > 0$, and $I(x) = 0$ otherwise. cw denotes a random backoff for tie-breaking.

(c) If flow f's backoff timer expires and the channel is consistently idle, transmit the head-of-line packet of flow f.

Figures 5 and 6 show how the algorithm works as an example. Assume that the initial virtual time $V = 0$, and the initial service tags for the four flows be $T_i = i$ ($i = 1, \cdots, 4$). The table maintained at each sender of the four flows and the backoff calculated for each flow are shown in Figure 6. Note that MLM-FQ in this scenario will schedule F_1 only, as it is the only flow that possesses local minimum service tag. However, EMLM-FQ will schedule both F_1 and F_4, thus achieving higher channel utilization.

3.3.1 Analytical Properties of EMLM-FQ

Because EMLM-FQ allows spatial channel reuse, the unfairness bound for MLM-FQ in Proposition 3.1 does not hold true for EMLM-FQ. However, we prove that EMLM-FQ achieves higher channel utilization *not* at the cost of fairness. As a matter of fact, it provides higher minimum service in the network.

Proposition 3.3 *Assuming consistent table states, EMLM-FQ ensures that each backlogged flow f that locates in the maximal clique of the flow contending graph receives a minimum fair share of the channel C as*

$$W_f(t_1, t_2) \geq C \frac{r_f}{k \sum_{g \in \mathcal{S}} r_g}(t_2 - t_1) - \alpha$$

where k and α are two topology-dependent constants, and $\mathcal{S}$ denotes the set of the flows in the maximal clique of the flow contending graph.

Proposition 3.4 *(Minimum Fair Share) Assuming consistent table states, EMLM-FQ ensures that each backlogged flow f receives a minimum fair share of the channel C as*

$$W_f(t_1, t_2) \geq C \frac{r_f}{k \sum_{g \in \mathcal{S}} r_g}(t_2 - t_1) - \alpha$$

where k and α are two topology-dependent constants, and $\mathcal{S}$ denotes the set of the flows in the maximal clique of the flow contending graph.

Proof The proof is based on the observation that EMLM-FQ is work-conserving in the following sense. In the flow contending graph, either a flow (or multiple flows) in f's neighborhood is transmitting, or f itself is transmitting. Considering the worst case where flow f has a neighboring flow m that contends with flow f only. That is, flow m's maximum contending backoff period is one. At any time constant, as long as flow f has a backoff period longer than one, it cannot acquire the channel. That is, flow f has a strong neighboring contending flow (i.e., m with a small degree in the flow contending graph), and its minimum fair share is constrained by the neighboring flow that receives the smallest fair share. Since the flow contending graph is connected, generalization of above contention process will finally reach a flow in the maximal clique of the flow contending graph, with which f's fair share is defined. By Proposition 3.3 the above minimum fair share follows. □

Note that, compared with the minimum fair share of MLM-FQ, EMLM-FQ provides a higher worst-case minimum fair share that scales with the total number of flows. This property is desirable in a wireless ad-hoc network where the flow contending graph is sparse but the total number of flows is large.

3.3.2 Impact of Inconsistent Table States

EMLM-FQ is more resilient to inconsistent table states compared with MLM-FQ. Because a flow contends for channel once its backoff timer expires, EMLM-FQ eliminates channel contention deadlock observed for MLM-FQ. However, stale table states may cause an EMLM-FQ scheduler to temporarily violate the maximize-local-minimum fairness. In this section, we use the same example in Section 3.2.2 to show that this violation only impacts short-term fairness, and the unfairness is statistically bounded.

Note that stale table entries can only appear for contending flows that originate from a different node, with service tags smaller than the updated value. In the example of service tag propagation collision shown in Figure 5 and Section 3.2.2,

stale table entries of flow F_1 and F_4, maintained at the sender of F_2, can result in a backoff period of 2 time slots for flow F_2. In the meantime, F_1 and F_4 will backoff 2 time slots as well, observing from their tables that F_2 and F_3 have smaller service tags. In this case, the extra random backoff breaks the tie. As long as its backoff timer does not persistently expire later than either F_1 or F_4 and the new service tags of F_1 or F_4 do not repeatedly collide, the sender of F_2 will eventually update its table entries for F_1 and F_4. Once the table entries are updated, F_2 schedules itself continuously to reclaim all its service lag.

As a generalization of above analysis, we have the following proposition:

Proposition 3.5 *(Probabilistic Minimum Fair Share) Let $W'_f(t_1, t_2)$ denote the service that a flow f receives during time interval $[t_1, t_2]$ with consistent table states as defined in Proposition 3.4, and d denotes f's degree in the flow contending graph. In the presence of inconsistent table states, there exists a probability $p < 1$, depending on the local topology and d, such that f's service lag is statistically bounded as:*

$$P\left[W_f(t_1, t_2) \leq W'_f(t_1, t_2) - nL\right] \leq p^n$$

for $n \leq \frac{C(t_2-t_1)}{L_{max}}$, and $P = 0$ otherwise. L denotes the packet size of flow f, C denotes the channel capacity, and L_{max} denotes the maximum packet size of f's contending flows.

3.4 Delay and Throughput Decoupling

We further improve the performance of EMLM-FQ with delay and throughput decoupling. This allows us to accept applications with different delay and throughput requirements and to enlarge the schedulable region. With this feature in place, we do not need to support both high-throughput, high-delay applications and low-throughput, low-delay applications at the cost of excess bandwidth allocations.

The idea behind delay and throughput decoupling is as follows. Each flow is assigned both a rate weight r_f and a delay weight r_d. The start tags of the incoming packets are assigned according to the flow's rate weight r_f, but their finish tags (i.e., the service tag) are based on the delay weight r_d. Essentially, the introduction of r_d enables us to switch the scheduling order among arrived packets in a way that decouples the long-term rate of arriving packets (according to r_f) from the delay requirement for packets (specified by r_d). For a delay-sensitive flow, instead of scheduling it according to its rate weight, we could do some short-term swapping of transmission order according to its deadlines. These delay sensitive flows can be served earlier than the flows that are not delay-sensitive. The detailed tagging operations modify those of Section 3.2 as follows, and packets are scheduled according to their finish tags:

(a) If f is continually backlogged, then $S_k^f = S_{k-1}^f + L_p^{k-1}/r_f$. Otherwise, $S_k^f = \max_{g \in W}\{V_g(A(t_k^f))\}$

(b) $F_k^f = S_k^f + L_p^k/r_d$ where r_d is the delay weight.

4 EMLM-FQ Implementation

This section presents an implementation of EMLM-FQ within the CSMA/CA MAC paradigm. Our implementation seeks to address the following practical issues:

- *Exchange of the table information between a flow's sender and its receiver*: In the models described in Section 2, each node maintains information for flows within one-hop neighborhood in the flow contention graph. However, one-hop neighborhood in a flow contending graph will translate to the two-hop neighborhood in the node graph. Therefore, given a flow f, our proposed algorithms require us to maintain flow information for flows that are within the transmission range of both f's sender and receiver. The sender has to retrieve the flow information that is maintained at the receiver to make scheduling decision.
- *Propagation of each transmitting flow's updated service tag*: In our model, the table of each node needs to record the most recent service tag for each neighboring flow. Whenever a flow transmits, either the senders or the receivers of its neighboring flows should update the new service tag for this flow. We piggyback this information in small control messages to decrease the probability of collisions.

4.1 Basic Message Exchange

In our protocol, each packet transmission follows a basic sequence of RTS-CTS-DS-DATA-ACK handshake, preceded by a backoff of certain time slots. Specifically, when a node has a packet to transmit, it checks its local table and sets a backoff timer to be the number of flows with smaller service tags, plus a random backoff for tie-breaking. This way, the local minimum-tag flow backs off for zero time slot and contends for the channel immediately. If the backoff timer of flow f expires without overhearing any ongoing transmission, it starts RTS (carrying B_f^R to be explained in the next section) to initiate the handshake. If the node overhears some ongoing transmission during its backoff period, it cancels its backoff timer and defers until the ongoing transmission completes. At the same time, it updates its local table for the tag of the on-going transmitting flow. When other nodes hear a RTS, they defer for one CTS transmission time to permit the sender to receive a CTS reply. When a receiver receives an

RTS, it checks its local table. If B_f^R is greater than or equal to the backoff value for flow f in the receiver's local table, it responds with CTS. Otherwise, the receiver simply drops RTS. Once a sender receives the CTS, it cancels all remaining backoff timers (for other flows) and transmits DS (Data-Sending that announces the success of RTS-CTS handshake [3]). Other nodes that hear either a CTS or a DS message defer until the DATA-ACK transmission completes.

4.2 Table Information Exchange

Since the one-hop neighboring flow information of any flow may be distributed at both the sender and the receiver, the backoff value has to be set accordingly. In EMLM-FQ, for flows that have smallest service tags in their local tables, the backoff is set to zero; for each flow f in concurrent transmissions for channel reuse, its backoff is set to be the number of flows whose service tags are less than flow f in its local table. Therefore, we should set the backoff value by taking into account the tables maintained by both the sender and the receiver. That is, $B_f = B_f^S + B_f^R$, where B_f is flow f's backoff, B_f^S is the backoff according to its sender's table, and B_f^R is the backoff according to the table at the receiver's side. However, the sender's table does not have the information of the receiver's table. For simplicity of discussion, let us assume that the sender's table and the receiver's table do not have identical entries for the same flow. If a flow indeed appears in both tables, the receiver deletes this flow to avoid double counting; this can be easily achieved at the flow join-in phase.

Suppose node N is the sender of flow f. Therefore, N knows precisely the backoff value B_f^S for its own flow f, but does not know B_f^R. We will let the sender estimate B_f^R. To this end, whenever a flow f is transmitting through the RTS-CTS-DS-DATA-ACK sequence, the ACK packet carries two parameters: M_f and b_f in order for the sender to estimate B_f^R later on. M_f tells us how many bytes other flows in the receiver's table have to be served before flow f can transmit its packet. $M_f = \sum_{j \in B}(T_f - T_j)w_f$, where w_f is flow f's weight, T_j is flow j's current tag in the receiver table. The flow set B denotes all flows that have smaller tag T_j than T_f. b_f denotes the backoff value for flow f at its receiver's table. When the sender N receives this information, it records M_f as well as the current time t_f. At later time instant t, sender N estimates $B_f^R \approx b_f \cdot (M_f - C \cdot (t - t_f))/M_f$ where C is the channel capacity, and sets backoff for flow g as $B_f = B_f^S + B_f^R$. This way, we avoid the overhead of per-packet B_f^R update between the sender and receiver.

4.3 Service Tag Update

In order to propagate a flow's service tag to all the one-hop neighbors in of both the sender and the receiver, and reduce the chance of information loss due to collisions during this service tag information propagation, we attach the tag T_f for flow f in all four control messages RTS, CTS, DS and ACK. However, we do not use the updated tag for flow f in RTS and CTS packets, since RTS and CTS do not ensure a successful channel acquisition. Instead, we propagate the old tag to help correct potential stale information in the one-hop neighborhood of the sender and the receiver. After a successful RTS-CTS handshake, we attach the updated flow tag in DS and ACK, to inform neighboring nodes of the new *updated* service tag of the current transmitting flow f. Note that a sender always has the complete and correct information regarding its own flows, i.e., current service tag. Therefore, only *accurate* information is propagated for the scheduling purpose.

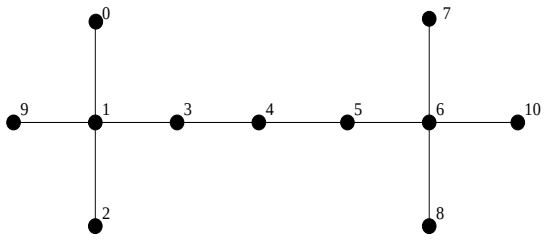


Figure 7: Scenario 5.1, Dumbbell Topology

5 Performance Evaluation

In this section, we use simulations to evaluate the performance of EMLM-FQ with different overlaying applications. EMLM-FQ algorithm is implemented within the ns-2 simulator. The radio model is based on the existing commercial hardware with a wireless transmission range of 250 meters and channel capacity of 2Mbps. Each simulation runs for 300 seconds and the results are compared with those using FIFO and IEEE 802.11 MAC.

The applications of interest include: FTP-driven TCP traffic, CBR-driven (constant bit rate) UDP traffic, audio-driven UDP traffic and video-driven UDP traffic. For the FTP, CBR, and audio traffic, the modules in the ns-2 distribution are used. For the video traffic, we use the actual traces to drive the ns-2 simulations. All packets are set to be of 512 bytes, except that video traffic has varying packet sizes based on the actual traces. Dynamic Source Routing (DSR) is used for routing.

5.1 Performance Isolation

In this example, we demonstrate the effectiveness of the EMLM-FQ protocol in preventing aggressive traffic from capturing the wireless channel. We use a dumbbell topology in Figure 7 with three end-to-end traffic flows: two FTP-

Flow	802.11	EMLM-FQ
0 to 7 (TCP)	7.17	38.18
2 to 8 (TCP)	10.99	37.42
9 to 10 (UDP)	187.64	65.80

Table 1: Scenario 5.1, Throughput (kbps)

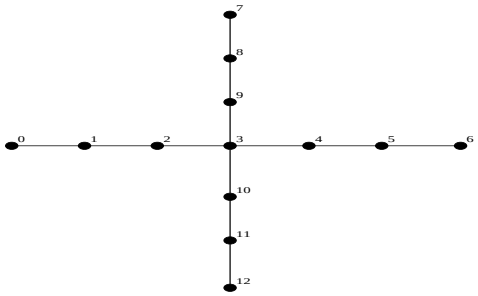


Figure 10: Scenario 5.2, Cross Topology

driven TCP flows and one CBR-driven UDP flow. The two TCPs originate from node 0 and node 2, and end at node 7 and node 8, respectively. The CBR-driven UDP packets are sent at an aggressive rate from node 9 to node 10 so that there will always be packet backlogged at node 9.

The results of the aggregate throughputs are shown in Table 1, and the instantaneous throughputs are shown in Figures 8 and 9 to demonstrate short-term fairness. As we can see, without the fairness regulation, IEEE 802.11 almost starves these two TCP flows, while UDP flow occupies the majority of the channel bandwidth. In contrast, EMLM-FQ provides better performance isolation among these three flows. Note that TCP throughputs are still lower than the UDP flow. The reason is that EMLM-FQ is work-conserving, and the TCP flow is not always backlogged due to its congestion control.

5.2 Fairness among TCP Flows

In this example, we study EMLM-FQ's performance in providing fair channel access among contending TCP flows under similar conditions. We design a cross topology shown in Figure 10, with six hops on each branch. Two FTP-driven TCP flows are deployed end-to-end on both the two branches, one from node 0 to node 6, and the other from node 7 to node 12.

The results are shown in Table 2. Due to the binary exponential backoff algorithm used by the IEEE 802.11 standard, at node 3 where these two flows contend, one flow captures the channel and wins over the other. In this case, TCP running from node 0 to node 6 ends up with occupying the majority of the channel capacity. EMLM-FQ, on the other hand, provides a much better channel sharing for the two contending TCP flows.

Flow	802.11	EMLM-FQ
0 to 6 (TCP)	119.90	52.07
7 to 12 (TCP)	22.04	52.50

Table 2: Scenario 5.2, Throughput (kbps)

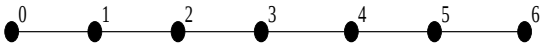


Figure 11: Scenario 5.3, Linear Topology

5.3 Fairness among Long and Short TCP Flows

In this example, we study the performance of the EMLM-FQ protocol in providing fair channel access among contending TCP flows of different hop lengths. It is well-known that due to TCP end-to-end congestion control, short TCP flows tend to have smaller RTT and therefore higher throughput than long TCP flows. This problem remains in the wireless domain and is exacerbated by channel contention. Here, we use a linear topology of six hops shown in Figure 11 with two FTP-driven TCP flows, one originating from node 0 to node 6 (long), and the other from node 1 to node 5 (short).

The results of the simulation are shown in Table 3. IEEE 802.11 clearly favors the shorter TCP. The reason, besides the inherent TCP AIMD congestion control, is that as the result of having one flow at higher rate, it tends to grab the channel more easily. Though we cannot change the TCP AIMD congestion control mechanism, EMLM-FQ handles the channel sharing well. We can see from the results that the shared wireless channel bandwidth is much better distributed among the long and short TCP flows, compared with the results of IEEE 802.11.

5.4 Real-time Traffic Delay Jitter

In this example, we study the performance of EMLM-FQ with multimedia traffic. We use the dumbbell topology with three video-driven UDP traffics based on the actual MPEG traces of three movies. The simulation setting is similar to that of Scenario 5.1 with three end-to-end flows from node 0 to node 7, node 2 to node 8, and node 9 to node 10, respectively. For typical streaming video, we are largely concerned with the delay jitter, since a large delay jitter will result in poor playback quality. The results of the delay jitters of the three video traffic are shown in Table 4. As we can see, having EMLM-FQ schedule the channel access does

Flow	802.11	EMLM-FQ
0 to 6 (long TCP)	10.36	58.53
1 to 5 (short TCP)	176.24	81.69

Table 3: Scenario 5.3, Throughput (kbps)

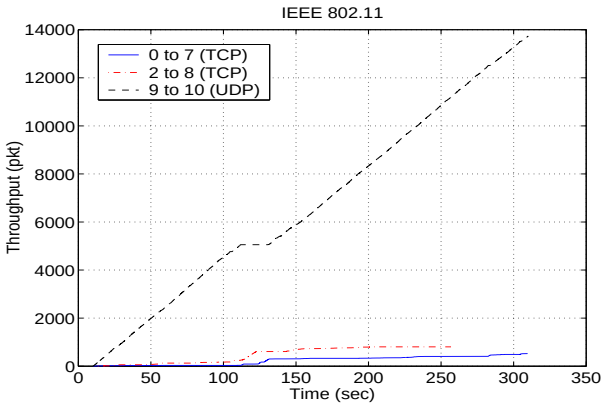


Figure 8: Scenario 5.1, IEEE 802.11 Throughput

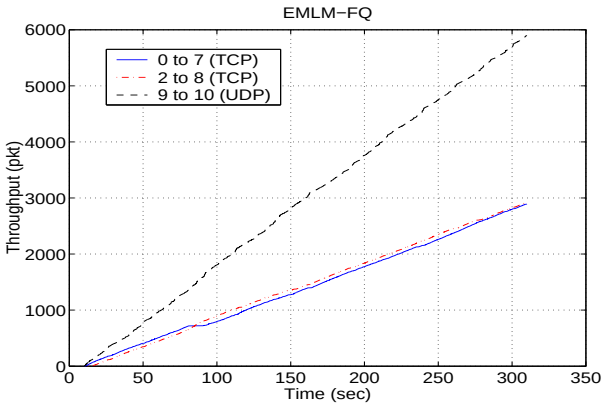


Figure 9: Scenario 5.1, EMLM-FQ Throughput

Flow	802.11	EMLM-FQ
0 to 7	0.1849	0.0489
2 to 8	0.1244	0.1030
9 to 10	0.6066	0.1274

Table 4: Scenario 5.4, Delay Jitter (sec)

Flow	EMLM-FQ w/o DD	EMLM-FQ w/ DD
0 to 7 (audio)	8.71	8.71
2 to 8 (CBR)	9.93	9.92
9 to 10 (CBR)	9.96	9.96

Table 5: Scenario 5.5: Throughput (kbps)

provide smaller delay jitter. IEEE 802.11 standard employs a binary exponential backoff method in resolving contention, and whenever a node successfully transmits a packet, the contention window is reset all the way down to the default minimum window size. This sudden change of the backoff window size results in large transmission burstiness and significant delay jitters.

5.5 Delay and Throughput Decoupling

In this example, we study the effectiveness of our delay and throughput decoupling mechanism in reducing the average delay. We use a simulation setting similar to the dumbbell topology of Figure 7, with two CBR-driven UDP flows and one audio-driven UDP traffic. Two CBR-driven flows originate from node 2 and node 9, and end at node 8 and node 10, respectively. The audio traffic goes from node 0 to node 7. The audio source operates at 8.8 kbps, while the two CBR sources are at 100 kbps.

Because the IEEE 802.11 leads to the starvation of the audio flows, we only compare the result of EMLM-FQ with and without delay decoupling. The simulation results are presented in Tables 5 and 6. We can see in Table 5 that the

Flow	EMLM-FQ w/o DD	EMLM-FQ w/ DD
0 to 7 (audio)	0.4133	0.3016
2 to 8 (CBR)	0.3028	0.3216
9 to 10 (CBR)	0.3563	0.4071

Table 6: Scenario 5.5: Average Delay (sec)

long-term throughput remains almost the same with slight variations due to randomness, not affected by our decoupling mechanism. From Table 6, we can see that the audio traffic benefits from the delay decoupling by reducing its average delay from about 0.41 second to about 0.3 second. Correspondingly, the average delay for the other two CBR-driven UDP flows, which are not delay sensitive, increases. This shows that the decoupling mechanism better serves different applications with different delay/bandwidth requirements.

5.6 Fairness and Aggregate Throughput with Large Number of Flows

We finally simulate a large network topology with 21 CBR-driven UDP flows. The network topology is shown in Figure 12, and the corresponding flow contending graph is shown in Figure 13. The throughput for each flow is given in Figure 14. We use the standard max-min fairness metric [2] and the equality fairness metric [4] to evaluate the throughput distribution. The results are shown in Table 7. As we can see, EMLM-FQ performs extremely well in this large wireless ad-hoc network and achieves both better fairness and higher aggregate throughput than IEEE 802.11 with FIFO. Note that although EMLM-FQ significantly improves the fairness in terms of both max-min and equality, it is not designed to achieve perfect equality fairness, which may prohibit spatial channel reuse and result in very low channel utilization (Section 3.2.1). Instead, EMLM-FQ seeks to maximize the minimum service, and enhances channel utilization through opportunistic spatial reuse. Its surprisingly higher aggregate

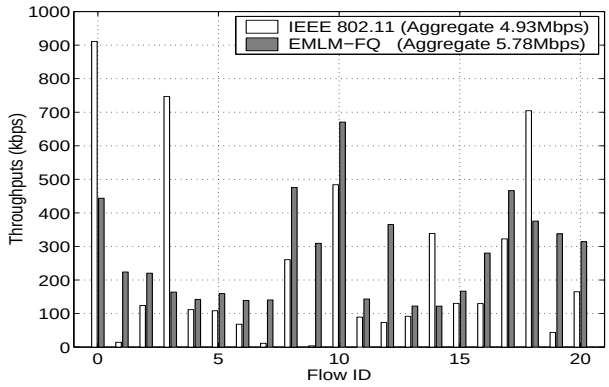


Figure 14: Scenario 5.6, Throughput

Pkt Scheduling	Max-min Fairness	Equality Fairness	Aggregate Throughput
802.11+FIFO	0.004	0.457	4.93 Mbps
EMLM-FQ	0.182	0.780	5.78 Mbps

Table 7: Scenario 5.6: Fairness and Aggregate Throughput

throughput in this large example, compared with that of IEEE 802.11, comes from the fact that EMLM-FQ better coordinates the channel access and therefore reduces the channel contention and collision significantly.

6 Discussions

We further discuss three design issues in this section.

Node mobility A wireless ad-hoc network topology may change dynamically due to node mobility, resulting in degraded performance of designs that require global topology information or perform global computation. Note that both our proposed localized algorithms and their implementations require only one-hop neighboring flow information (i.e., flows' current service tags) and simple local computation. This feature makes our design resilient to the impact of node mobility. However, if a node is mobile, it does take several packet transmission times to discover its new neighborhood and establish the corresponding table entries in the new location.

Scalability Another feature of our proposed design is that it scales well in an ad-hoc network with large number of flows. This is because each node only communicates with its one-hop neighbors in the node graph, and only local contending flows' information is maintained. The computation workload performed at each node, i.e., tag assignment, sorting flows based on current virtual times, etc., is also light.

Power consumption Our algorithms require that each node with backlogged flows tune its wireless interface into promiscuous mode, so that scheduling information can be obtained from overhearing the wireless channel. However, for certain wireless interfaces [7], promiscuous mode (or idle mode) power consumption is almost the same as that in receiving mode, and comparable to the energy consumption in transmitting mode. In order to save energy with such hardware, a sender can schedule multiple packets once it grabs the channel, so that other nodes can turn their wireless interfaces into sleep mode during this period. This way, we can tradeoff the fairness granularity (i.e., fine-grained versus coarse-grained) for power savings.

7 Related Work

Packet scheduling has been intensively studied in the networking literature. Numerous algorithms have been proposed, among which are WFQ [6, 19], WF^{2}Q [1], SFQ [8], etc.. A lot of research efforts have been put to adapt fair packet scheduling to cellular wireless networks, notably IWFQ [14], CIF-Q [18, 20] and WFS [15]. 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 recent years, distributed fair packet scheduling in a wireless LAN has attracted a lot of interests in the research community [22]. The major challenge there is the lack of a centralized scheduling point where the scheduling decision can be enforced. The solution is to piggyback the virtual time and broadcast it in the wireless LAN so that each node maintains such a virtual time by overhearing the channel. Kanodia et al. [12] study coordinated scheduling operations along the multi-hop data path by dividing the path into areas of wireless LANs. At each area, flow priorities are piggybacked and broadcasted, and each node backoffs accordingly to achieve the prioritized transmission and end-to-end performance assurance. However, two contending flows in a wireless ad-hoc network may contend with different sets of flows, in contrast to the scenario in a wireless LAN where each flow contends with all other flows. The flow contending graph in a wireless LAN is completely connected, which does not hold in a wireless ad-hoc network. Adding the spatial channel reuse as a second unique characteristic of wireless ad-hoc networks, the distributed fair queueing algorithms proposed for wireless LAN do not directly apply.

In multihop wireless networks, providing minimum through-

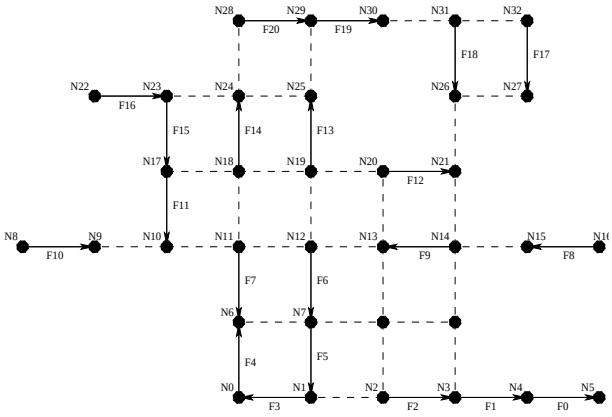


Figure 12: Scenario 5.6: Node Graph

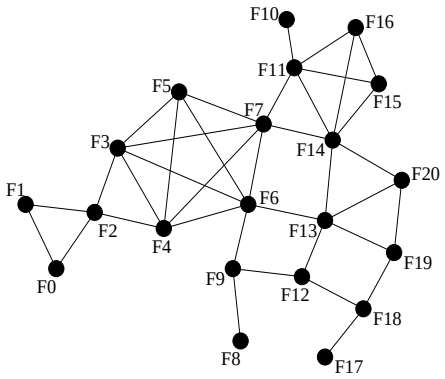


Figure 13: Scenario 5.6: Flow Contending Graph

put and bounded delay access has been studied at the MAC layer and some representative works on this topic include [5, 11, 21]. These works seek to design conflict-free link scheduling schemes that attempt to maximize channel spatial reuse and remain immune to topological changes in the presence of node mobility, and they are typically developed within the TDMA-like MAC protocol. The focus of these MAC-layer designs has been the mechanisms of channel access by assuming that the packet scheduling algorithm has been worked out, rather than the other way around.

Some recent work has been reported for fair packet scheduling in wireless ad-hoc networks. In [17], the authors seek to formulate the problem of ad-hoc fair queueing, with the goal to maximize channel spatial reuse while ensuring fairness. The focus has been to define an ideal centralized model, though it also provides a tree-based distributed implementation. A centralized model and a distributed implementation protocol for packet scheduling in ad-hoc networks are presented in [16], but the focus there is how to resolve the conflicts between fairness and maximal throughput and arbitrate the tradeoff between these two. Huang et al. [9] propose centralized and distributed algorithms to assign max-min fair share for flows, but left the critical implementation details. Kanodia et al. [13] take a similar mechanism to achieve distributed ordered channel access. However, it does not explicitly address the impacts of inconsistent contending flow sets, and the priority propagation collisions for the scenarios where nodes locate in the middle of two simultaneous on-going transmissions. Busy tone priority scheduling [23] limits the impacts of hidden terminals and collisions of information propagation, but at the cost of additional two narrowband busy tone signals.

8 Conclusion

In this paper, we have proposed two new localized and fully distributed fair queueing algorithms for wireless ad-hoc networks. Our design seeks to devise scalable and efficient solutions to provide fair channel sharing and increase spatial channel reuse. In our solutions, multiple localized schedulers coordinate their scheduling decisions and collectively achieve the desired global fairness. Only local communication and computation are involved to scale with the network sizes and facilitate implementation. Moreover, the design better serves applications with different delay and bandwidth requirements through bandwidth and delay decoupling. The proposed scheduling algorithms have provable performance bounds, while practical implementation issues are addressed in the popular CSMA/CA MAC paradigm. We demonstrate the effectiveness of our proposed design through both simulations and analysis.

References

[1] J. C. Bennett and H. Zhang. WF^2Q: Worst-case Fair Weighted Fair Queueing. In *IEEE INFOCOM*, 1996.

[2] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1987.

[3] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A Medium Access Protocol for Wireless LANs. In *ACM SIGCOMM*, 1994.

[4] D. Chiu and R. Jain. Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[5] I. Chlamtac and A. Farago. Making Transmission Schedules Immune to Topology Changes in Multi-hop Packet Radio Networks. *IEEE/ACM Transactions on Networking*, 2(1):23–29, Feb 1994.

[6] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *ACM SIGCOMM*, 1989.

[7] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM*, 2001.

[8] P. Goyal, H. M. Vin, and H. Cheng. Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks. In *ACM SIGCOMM*, 1996.

[9] X. L. Huang and B. Bensaou. On Max-min Fairness and Scheduling in Wireless Ad-Hoc Networks: Analytical Framework and Implementation. In *ACM MOBIHOC*, 2001.

[10] IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. IEEE standard 802.11, 1997.

[11] J.-H. Ju and V. Q. Li. An Optimal Topology-transparent Scheduling Method in Multihop Packet Radio Networks. *IEEE/ACM Transactions on Networking*, 6(3):298–306, June 1998.

[12] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. Distributed Multi-Hop Scheduling and Medium Access with Delay and Throughput Constraints. In *ACM MOBICOM*, 2001.

[13] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. Ordered Packet Scheduling in Wireless Ad Hoc Networks: Mechanisms and Performance Analysis. In *ACM MOBIHOC*, 2002.

[14] S. Lu, V. Bharghavan, and R. Srikant. Fair Scheduling in Wireless Packet Networks. *IEEE/ACM Transactions on Networking*, 7(4), August 1999.

[15] S. Lu, T. Nandagopal, and V. Bharghavan. Fair Scheduling in Wireless Packet Networks. In *ACM MOBICOM*, 1998.

[16] H. Luo and S. Lu. A Topology-Independent Fair Queueing Model in Ad Hoc Wireless Networks. In *IEEE ICNP*, 2000.

[17] H. Luo, S. Lu, and V. Bharghavan. A New Model for Packet Scheduling in Multihop Wireless Networks. In *ACM MOBICOM*, 2000.

[18] T. Ng, I. Stoica, and H. Zhang. Packet Fair Queueing Algorithms for Wireless Networks with Location-dependent Errors. In *IEEE INFOCOM*, 1998.

[19] A. K. Parekh. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks. Ph.D. Dissertation LIDS-TR-2089, MIT Laboratory for Information and Decision Systems, 1992.

[20] P. Ramanathan and P. Agrawal. Adapting Packet Fair Queueing Algorithms to Wireless Networks. In *ACM MOBICOM*, 1998.

[21] Z. Tang and J. J. Garcia-Luna-Aceves. A Protocol for Topology-dependent Transmission Scheduling in Wireless Networks. In *IEEE WCNC*, 1999.

[22] N. H. Vaidya, P. Bahl, and S. Gupta. Distributed Fair Scheduling in a Wireless LAN. In *ACM MOBICOM*, 2000.

[23] X. Yang and N. Vaidya. Priority Scheduling in Wireless Ad Hoc Networks. In *ACM MOBIHOC*, 2002.

[24] L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks. In *ACM SIGCOMM*, 1990.