A Self-Coordinating Approach to Distributed Fair Queueing in Ad Hoc Wireless Networks

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

Abstract— **Distributed fair queueing in shared-medium ad hoc wireless networks is non-trivial because of the unique design challenges in such networks, such as location-dependent contention, distributed nature of ad hoc fair queueing, channel spatial reuse, and scalability in the presence of node mobility. In this paper, we seek to devise new distributed, localized, scalable and efficient solutions to this problem. We first analyze an ideal centralized fair queueing algorithm developed for ad hoc networks, and extract the desired global properties that the localized algorithms should possess. We then propose three localized fair queueing models, in which local schedulers self-coordinate their local interactions and collectively achieve the desired global properties. We further describe a novel implementation of the proposed models within the framework of the popular CSMA/CA paradigm and address several practical issues. Our simulations and analysis demonstrate the effectiveness of our proposed design.**

I. Introduction

The explosive growth of the Internet and the convergence of wireless communication and networking have jump-started several wireless networking technologies such as MANET, bluetooth and sensor networks. These emerging wireless technologies are envisioned to support a rich set of data applications, e.g., both error-sensitive and delay-sensitive applications, over the bandwidth-constrained wireless medium. With this vision in mind, the issue of providing fair and bounded delay channel access among multiple contending hosts over a scarce and shared wireless channel has come to the fore. Fair queueing has been a popular paradigm to achieve this goal in both wireline and packet cellular networking environments [1]–[9]. However, the problem of designing fully distributed, scalable, and efficient fair queueing algorithms in the shared-channel ad hoc wireless network remains largely unaddressed because of the unique issues in such networks. These issues include location-dependent contention, distributed nature of ad hoc fair queueing, channel spatial reuse, and how to manage a potential large number of flows in a dense and mobile network graph.

Two recent related works [10], [12] have formulated the problem of fair packet scheduling in ad hoc networks and addressed some of the issues raised above. In each work, the authors have proposed an *ideal centralized* fair queueing model by assuming a centralized scheduler and perfect knowledge of each flow in the entire network topology, and then designed a distributed implementation to *approximate* the idealized model. Therefore, the focus of these two works has been the problem formulation and an appropriate ideal centralized model for fair queueing in shared-channel multihop wireless networks. As a consequence, the distributed implementation proposed in these papers can at best conceptually approximate the proposed centralized model.

In essence, the unique characteristics of ad hoc wireless networks such as location-specific contention create spatial coupling effects among flows in the network graph, and the fundamental notion of fairness may require non-local computation among contending flows. Adding these features together, fair queueing in shared-channel multihop wireless environments is no longer a local property at each output link and has to exhibit global behaviors; this has to be achieved through distributed and localized decisions at each node.

In this paper, we re-examine the problem of distributed fair packet scheduling in a shared-medium multihop wireless network. If fair packet scheduling is indeed a distributed computation problem by its nature, an ideal centralized model cannot lead us too far except provide us a possibly unachievable performance bound. Therefore, we take a different approach in this work: we analyze an ideal centralized fair queueing algorithm in ad hoc networks, and extract the *desired global properties* from it. Our goal is to devise distributed and localized solutions such that local schedulers must *self-coordinate their local interactions* to achieve the desired global behavior. To this end, we propose a suite of fully distributed and localized 3-D (2D of the graph topology and 1D in the time domain) fair queueing models that use local flow information and perform local computations only, and describe our implementation of these models within the popular CSMA/CA paradigm. Through both simulations and analysis, we show that these local fair queueing models perform locally coordinated scheduling decisions and *collectively* achieve desired global properties such as fairness, scaling and network efficiency.

Two key contributions of this paper are the following: (a) a suite of *fully distributed* and *localized* fair queueing models, which *collectively* exhibit desirable global fairness properties; and (b) a novel table-driven distributed implementation within the framework of CSMA/CA paradigm. Our simulations and analysis demonstrate that our proposed approach provides a localized, scalable, and efficient solution to distributed fair queueing in ad hoc networks.

The rest of the paper is organized as follows. Section II describes the network model and identifies the key design challenges for ad hoc fair queueing. Section III analyzes a centralized algorithm, extracts desired global properties for distributed fair queueing algorithms, and proposes a suite of distributed and local fair queueing models in an ad hoc network. Section IV describes a distributed implementation of the model. Section V presents a simulation-based performance evaluation of the proposed algorithm. Section VI provides some discussions and describes related work, and Section VII concludes the paper.

II. Models and Issues

A. Network Model

In this paper, 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 trans-

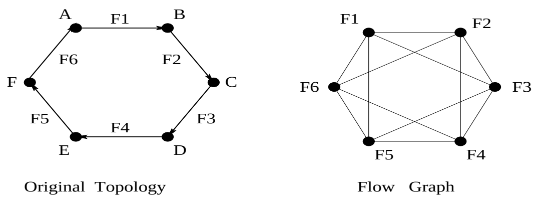

Fig. 1. Location-dependent contention

mission 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 neighboring nodes that are within transmission range of each other[1]. 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 flow[2] [13].

We make three assumptions [13]–[17]: (a) 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 work, (b) a node cannot transmit and receive packets simultaneously, and (c) neighborhood is a commutative property; hence, flow contention is also commutative.

In the following, we do not consider non-collision-related channel errors. For simplicity of presentation, we only consider fixed packet size in this paper, which is a realistic assumption in typical wireless networks. Our proposed models work equally well for the variable-packet-size case, and we will come back to this point in Section VI.

B. Issues of Fair Queueing in Ad Hoc Networks

Fair packet scheduling in ad hoc wireless networks is non-trivial because of two unique challenges of such networks:

(a) Location-dependent contention Since wireless transmissions are locally broadcast, collisions and contention for the shared medium, are location dependent. Consider the example shown in Figure 1, which shows a six-node network graph and each arrow-line denotes a packet flow from the sender to the receiver. Flow F_1 contends with flows F_2, F_3, F_5, F_6, since these four flows are within the transmission range of either the sender or the receiver of flow F_1. Therefore, these four flows should restrain from transmissions when F_1 transmits. Similarly, Flow F_2 contends with flows F_1, F_3, F_4, F_6. Hence, each flow has a different contending flow set depending on its location.

Location-dependent contention, together with the multi-hop nature of an ad hoc network, also allows for channel spatial reuse. Specifically, any two flows that are not interfering with each other can potentially transmit data packets over the physical channel simultaneously. Consider the example of Figure 1 again. Flows F_1 and F_4 are not contending with each other, and they can transmit concurrently, thus enabling channel spatial reuse.

Location-dependent contention as well as spatial reuse brings new issues to packet scheduling in ad hoc networks. In a wireline or packet cellular network, packets are scheduled *independently* at each output link. The scheduler at a link only needs to consider flows that are contending for that link, and no coordination efforts are needed for scheduling decisions made at neighboring nodes. Therefore, fair queueing model defined for such networks is, in essence, a *local* property for transmitting flows over each link, and fair queueing algorithms ensure local fair sharing of bandwidth (as defined by the fairness model) in the time domain among contending flows that share a single link.

In a shared-medium multihop wireless network, location-dependent contention generates coupling effects among flows in a network graph. Hence, flow scheduling is no longer a local decision, and such decisions cannot be made with respect to "local" flows alone and be done independent of other neighbors. Fair queueing becomes a 3-dimensional problem; it should be performed in both the time domain and the spatial domain. It needs coordinated design among neighbors.

(b) Distributed nature of ad hoc fair queueing In wireline networks, a switch is making scheduling decisions for its packet flows and it has direct access to the exact flow information, e.g., which flow has outstanding packets, when packets for a particular flow arrive and how many packets are waiting in the queue at any instant. In a packet cellular network, the base station is the natural logical choice for the scheduling entity in the cell, and the base station can access uplink flow information with the help of other supporting mechanisms, e.g., through MAC-layer techniques [5].

In an ad hoc network, contending flows may originate from different sending nodes, and each node needs to implement a local scheduler for its transmitting flows. Hence, no *single* logical entity for scheduling of these flows in the network graph is available. Besides, the flow information is "spreaded" out in these sending nodes, and each sender does not have direct access to other flows' information at other senders. Consider Figure 1 again, each of the six senders $A-F$ does not know the packet-level flow information at the other nodes. This illustrates that ad hoc fair queueing is a distributed computation problem by its nature.

C. Design Requirements for Fair Queueing in Ad Hoc Networks

In essence, wireless fair queueing in an ad hoc network is inherent global computation. This is due to the fact that location-specific contention, together with the notion of fairness, creates global coupling effects in the entire connected graph. Any scheduling decision made at a node may generate global domino effects in a connected graph. Hence, packet scheduling design cannot be isolated at each node. The notion of fairness must be defined with respect to at least a set of contending flows, if not all flows in the connected network graph. In addition, fair scheduling is a distributed computation problem and the proposed solution has to be fully distributed and localized. Thus, any distributed and localized solution to ad hoc fair queueing must coordinate their local interactions in order to achieve the desired global properties. This should be achieved within the constraints imposed by the networking environment.

Therefore, any fair packet scheduling algorithms proposed for multihop wireless networks must meet the following five stringent design requirements:

- *The solution must be fully distributed, and it involves only local computations by using local information only.*
- *The solution must exhibit desired global behavior, e.g., fairness property.*

[1]As in wireline fair queueing, multi-hop flows are treated as multiple single-hop flows.

[2]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.

• *The solution must be scalable.* The number of nodes in the ad hoc network can be large and the target can be a dense network, the solution should scale well in the presence of a large number of nodes and a dense graph. Besides, the solution should equally scale well in the presence of frequent node mobility and failures.
• *The solution must be efficient.* Because of channel spatial reuse, 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 in order to increase spatial reuse, while taking into account fairness considerations across flows.
• *The design must be coordinated among interacting nodes*. The nature of location-specific contention implies that, any scheduling decision made at a node may have global impact and incur domino effects in the entire connected network graph. As a result, packet scheduling in such a network has to be coordinated among neighbors that have contending flows, and this coordination should be conducted in both the time domain and the spatial domain.

III. A SELF-COORDINATING APPROACH TO AD HOC FAIR QUEUEING

In this section, we will develop a suite of novel distributed fair queueing models for ad hoc wireless networks.

A. *Desired Global Properties*

In order to develop localized models for distributed fair queueing in ad hoc networks, we first analyze an ideal centralized algorithm. Our purpose is to *extract* several *desired global properties* that the localized model should possess.

A.1 Flow graph versus node graph

To facilitate presentation, we first convert packet flows in a generic network topology into a *flow graph*. The flow graph precisely characterizes the spatial-domain, as well as the time-domain, contention relationship among transmitting flows. In a flow graph, each vertex represents a backlogged flow, and an edge between two vertex denotes that these two flows are contending with each other. If two vertices are not connected, these two flows can transmit simultaneously, thus spatial reuse is possible. Therefore, the flow graph explicitly describes which flows are contending and which flows can be concurrently transmitting. As an example, Figure 1 shows the flow contention graph for the six flows in the node graph.

A.2 What a centralized fair queueing model can do

Let us first re-visit an ideal centralized fair queueing algorithm designed for ad hoc networks. We start with the popular Start-time Fair Queueing (SFQ) [4] algorithm, which serves as the basis and has been further adapted to the ad hoc wireless networks in related works [10] [12]. In SFQ, each arriving packet is assigned two tags: a start tag and a finish tag. Specifically, a packet with sequence number k of flow f arriving at time $A(t_k^f)$ is assigned two tags: a start tag S_k^f and a finish tag F_k^f, defined as follows:

$$S_k^f = \max\{V(A(t_k^f)), F_{k-1}^f\};\;\; F_k^f = S_k^f + L_p/r_f \qquad (1)$$

where L_p denotes the packet size in bits, and $V(\cdot)$ is the system virtual time, taken to be the start tag of the packet currently being served in the scheduler during any busy period. Then, SFQ selects the flow with the minimum service tag (i.e., the start tag) at the moment and transmits its head-of-line packet.

Now let us adapt SFQ to the ad hoc networking environment. Here is a brief summary of the proposed modifications to SFQ in order to address ad hoc wireless networking issues [10] [12]:
• *On tagging*: for all the flows in the connected graph, we can still use the same tagging mechanism as SFQ. However, the propagation of system virtual time $V(t)$ to every flow in the network is a non-trivial issue [3].
• *Scheduling*: The scheduling decision serves two purposes: (a) the scheduler needs to select the flow f_{min} with the *smallest* start tag for its next packet transmission, in order to ensure a basic fair share for every flow; (b) in addition to flow f_{min}, the scheduler selects multiple non-interfering flows for concurrent transmissions in order to increase spatial reuse. In part (b), different centralized algorithms may have different policies for channel spatial reuse. [10] selects flows that are not interfering with f_{min}, starting from those with smallest service tags at t. [12] seeks to maximize spatial reuse and selects flows that solve a corresponding minimum-coloring problem.

A.3 Existing approach to approximating the centralized algorithm

Existing distributed algorithms seek to approximate the above centralized algorithm within the framework of CSMA/CA MAC protocol [10] [12]. These approximations are based upon the following observations on the above centralized algorithm:
• *On tagging*: If each flow f can "hear" or learn an approximation of $V(t)$ in the system, i.e., the start tag of the flow that is currently being served by the local scheduler, then f can assign both the start tags and the finish tags for its packets using (1). This way, flow f does not need to know exact information on other flows. However, the cost to pay is that $V(t)$ is not the global system virtual time, it is only valid locally, since each flow can at most learn its two-hop (i.e. both the sender and the receiver) neighborhood in the node graph.
• *Scheduling*: The scheduler needs to select the flow with the *smallest* service tag for its next packet transmission. In order to achieve this, at each node n, it sets a backoff interval B_f for each backlogged flow f before it transmits a packet of flow f. The backoff value $B_f(t)$ is set to be $B_f(t) = \eta * (S_k^f - V(t))$, where S_k^f is the service tag of the head-of-line packet of flow f, and η is a constant. This way, the flow with smallest service tag at t will transmit first (since it has smallest backoff period $B_f(t)$), and other contending flows will restrain from transmissions once they hear that flow f is transmitting through carrier sensing. In addition, flows that are not interfering with f will transmit concurrently, starting from the one with smaller backoff value.

At a first glance, it seems that the above approach works well as an elegant distributed implementation of fair queueing in such environments. However, a more careful examination reveals four limitations of this approach in a large ad hoc wireless network:
• (a) Local virtual time $\bar{V}(t)$ versus global system virtual time $V(t)$: in an ad hoc network, each node can only learn at best the start tag of the transmitting packet in its *local neighborhood*, thus the $\bar{V}(t)$ that it hears is in general different from the global system $V(t)$ as required by (1) in a large ad hoc network.

[3]In another recent work [18], the authors also proposed a local fairness model in a different context. In the local fairness model, each flow's virtual time $V_f(t)$ is defined only with respect to the one-hop neighboring flows in the flow graph.

This brings inaccuracy for the approximation as well as creating ambiguity in the setting of the backoff interval for each flow (i.e., each backoff is set with respect to different reference virtual time). This may lead to unbounded unfairness (see Section III-B.2 for an example).

• (b) Problems with backoff setting: at any time t, if there exists either significantly large or only minor difference of $B_f(t)$ among flows (e.g., caused by large difference in flow weights), then the backoff-based approach results in either large system overhead (by waiting for an extended period of backoff interval) or potential collisions (due to inability to detect other transmissions). Besides, it may also need fine-grain timer support.

• (c) Global synchronization: in order for the above backoff approach to work well, global synchronization is needed such that all flows will tick their backoff timer simultaneously. Without global synchronization, it cannot accurately approximate the centralized algorithm.

• (d) Potential unbounded flow unfairness in the system: due to channel spatial reuse, certain flows will always transmit (concurrently with the flow of smallest service tag) more often than others, and the above approach cannot *always* bound the unfairness bound between any two flows in the system. We further elaborate this point later.

A.4 Extracting desired global properties

However, the centralized algorithm described in Section III-A.2 is still very useful to help us in extracting several global properties that we would like our localized model to possess:

• *Minimum fair share*: Our fairness for each flow f is still measured with respect to its flow weight r_f. That is, each flow is served in proportional to its flow weight in its local scheduler. To ensure minimum fair share for each flow in the connected flow graph, the flow that receives the minimum normalized service (normalized according to its flow weight r_f) must transmit first. Equivalently, *the flow with the global minimum service tag at t should always transmit before other flows*. This is what we called "*maximize global minimum*" property.

• *Increasing channel spatial reuse*: since wireless channel is bandwidth constrained, the proposed model should encourage concurrent transmissions as much as possible. This way, the aggregate network channel utilization is improved.

• *Bounding unfairness if needed*: While channel spatial reuse increases network efficiency, it may cause certain flows' services unbounded in some topological scenarios, to be shown in later Section III-B.2. Some have argued that channel spatial reuse should be encouraged whenever it is possible (even at the risk of unbounded unfairness) as long as each flow receives a basic fair share [18]. However, for the purpose of simple accounting and in certain application scenarios, we may still want to limit flow unfairness. Therefore, in our design, we provide the *option* of bounding a flow's unfairness. Specifically, for any two backlogged flows f and g in a local neighborhood, the aggregate services that they receive during $[t_1, t_2]$ satisfy: $|\frac{W_f(t_1,t_2)}{r_f} - \frac{W_g(t_1,t_2)}{r_g}| \leq \delta$ where δ is a constant that characterizes the unfairness bound. Effectively, this requirement leads to long-term fairness for backlogged flows.

In addition, the proposed model should also possess the following scaling property, which is important in a large-scale or dense network:

• *Scaling property*: the proposed model should scale well in the presence of a large number of nodes/flows and in a dense graph. In order to achieve this, each node should maintain only local information and perform local scheduling computation.

B. *A Self-Coordinating Fair Queueing Model*

We now describe a self-coordinating approach to distributed fair queueing in ad hoc networks. The proposed model is fully decentralized and localized, and possesses all the desired global properties that are identified in Section II-A.4. Our model is self coordinating in the sense that each local scheduler will coordinate its local interactions with its neighbors in order to exhibit the desired global behavior. This is achieved without global computation or global information propagation.

B.1 Overview of the proposed model

In the model, each network node maintains a local table, and the table records information of all flows in its one-hop neighborhood in the flow graph. Each node is responsible for assigning tags and scheduling flows that originate from it, and for recording flow tags for other flows in its locality. It still uses SFQ to assign start tags and finish tags for the flows. Besides, 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 hears for flow $flow\_id$.

Our proposed model uses three mechanisms to achieve the global properties through coordinations of local interactions at each node:

• *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 the backoff mechanism to increase spatial reuse: if a flow does not have the local minimum service tag in its one-hop flow neighborhood, we still set a backoff timer for this flow in order to increase channel spatial reuse. Once the backoff timer expires and the channel is idle, the flow will transmit. Furthermore, *we tailor the flow's backoff value setting to both the flow's local fairness* (i.e., its received service compared to its neighbors) and *its local contention degree* (i.e., the number of contending flows in its neighborhood) *in the flow graph*.

• Using a *sliding window* to limit the unfairness bound as an option: each node may maintain an upper bound δ for flow unfairness; whenever any flow's service tag reaches beyond what is allowed by δ, the flow is restrained from transmissions temporarily.

We now describe these three mechanisms in more details.

B.2 Three mechanisms in the model

1. Maximizing local minimum

As observed in Section III-A.4, in order to ensure minimum fair share for each flow, the flow with the global minimum service tag in the entire network must be selected for transmission before any other flows. Note that this is a non-trivial issue in a distributed fair queueing model, since identifying the flow with the global minimum service tag in general requires sorting all flows in the entire network graph and it is a global computation. Therefore, how to use local information and local computation only to achieve this goal 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 that cannot avoid global computation, we identify all flows with local minimum service tags, and schedule all such flows for transmission. This is what we called *maximizing local minimum* policy. As far as service tag is concerned, since the global minima must be a local minima (but not vice versa), we know that the flow with the global minimum

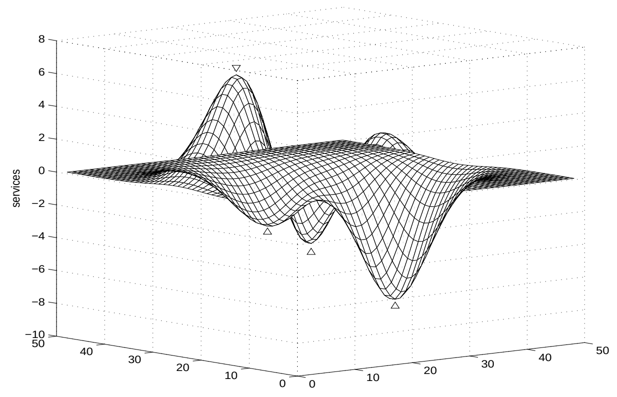

Fig. 2. A 3-D Distributed Fair Queueing Model

tag must be among these transmitted 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, but not verse versa.

We can also draw our model to an analogy of the 3-dimensional model shown in Figure 2, which illustrates 2-D in the spatial domain, and 1-D characterizes the aggregate service received up to t (time domain). Our proposed model is similar to the phenomenon that a rain storm washes a three-dimensional terrain: the water always fills in the three lowest (local) spots first in the Figure (marked with a Δ sign in the figure). Once the lowest spots have been filled, it starts to fill other new lowest points in the terrain.

2. New backoff mechanism to increase spatial reuse

From the rain storm analogy shown above, we see that if water only fills in the lowest local spots each time, it takes a long time to make the entire terrain drenched by the water and become "flat" eventually. The same thing is true for distributed fair queueing. The "maximizing local minimum" policy alone may result in very low aggregate network utilization in a large network topology. Note that while water flows into the lowest spots faster, some water will also rinse higher-attitude terrains. We will increase spatial reuse in ad hoc fair queueing the same way. To do this, we need to simultaneously schedule other non-interfering flows. Note that channel spatial reuse is location specific and is dependent on how many flows are contending at a particular position in the graph. We will use a backoff-based mechanism to achieve this. The novelty in our model is that we tailor the backoff setting to both the local contention degree (i.e., how many flows contend with a flow; this is equivalent to its degree in the flow graph) and the fairness model.

Specifically, for any flow f, *we set its backoff value to be the total number of flows that have smaller service tags (i.e., start tags) than flow* f. Three properties of this policy are: (a) the backoff value is decoupled from the numerical difference in virtual times, this removes limitation (b) described in Section III-A.3; (b) it gives higher priority to flows that have received less fair services than flow f (i.e., they have smaller service tags); and (c) the backoff is also tailored to current contention level in flow f's current location (i.e., how many flows are contending with f). Also note the minimum-service-tag flow has a backoff value 0, thus having the highest priority for local channel access.

3. Sliding window to Limit flow unfairness if needed

However, if we encourage spatial reuse whenever we can, together with the "maximizing local minimum" policy, the aggregate service received by each flow may not be in proportional to its flow weight. Consider a simple example in Figure 3, where there are five flows with identical weight 1. It is easy to see that over a long term, flow F_5 will receive three times of service compared to other flows. This is because F_5 can always transmit simultaneously with one of flows F_1, F_2, F_3 through spatial reuse. Therefore, flow unfairness for F_5 may become unbounded due to channel spatial reuse. In order to limit the unfairness bound if needed, we provide a sliding window mechanism at each node as an option.

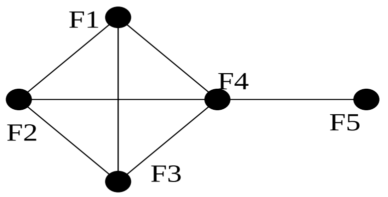

Fig. 3. Spatial reuse may cause unbounded flow unfairness

At any time instant, each node schedules packets of a flow within a sliding window ρ (defined in virtual time), to increase channel spatial reuse. However, when flow f's service tag exceeds the window, it stops in order to bound unfairness.

C. Model Description

In this section, we describe the three models in algorithmic details, that progressively incorporate the three mechanisms described above and we name as "Maximize-Local-Min Fair Queueing" (MLM-FQ), "Enhanced-Maximize-Local-Min Fair Queueing" (EMLM-FQ), and "Bounded-Fair Maximize-Local-Min Fair Queueing" (BFMLM-FQ), accordingly. The MLM-FQ model implements the "maximize-local-minimum" policy only, while EMLM-FQ in addition implements the backoff-based spatial reuse mechanism, and BFMLM-FQ implements all three.

We only present BFMLM-FQ algorithm here as a complete description of all three proposed mechanisms. The detailed operations consist of four parts:

1. *Local state maintenance*: Each node n maintains a local table E_n, which records each flow's current service tag for all flows in its one-hop neighborhood of the flow 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 most recent 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, which arrival time is $A(t_k^f)$ and packet size is 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
$S_k^f = \max_{g \in \mathcal{S}}\{V_g(A(t_k^f))\}; \;\; F_k^f = S_k^f + L_p/r_f$, where $\mathcal{S}$ consists of all flows stored in the table of node n, and $V_g(t)$ is flow g's virtual time at t.
3. *Scheduling loop*: At node n, whenever it hears that the channel is clear,
(a) if one of flows, say f, whose sender is n, has the *smallest* service tag in the table E_n of node 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 <

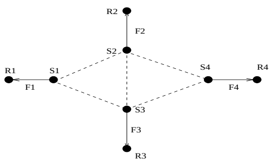

Fig. 4. An illustrative example

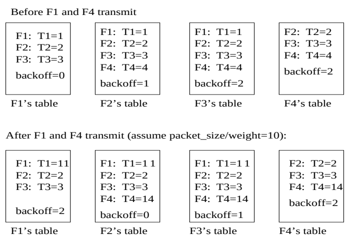

Fig. 5. How our algorithm works

$V_{\min} + \delta$ (where δ is the sliding window size and $V_{\min}$ is the virtual time at node n, defined as the minimum service tag in table E_n), set the backoff period B_f of flow f as $B_f = \sum_{g \in \mathcal{S}} I(T_g(t) < T_f(t))$ minislots, where $I(x)$ denotes the indicator function, i.e., $I(x) = 1$, if $x > 0$; $I(x) = 0$, otherwise.

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

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 the table entry.

We provide an illustrative example to show how our algorithm works in Figure 4. In the example, we have four flows and the dotted line denotes the two nodes are within the communication range. Let us assume that the initial virtual time $V = 0$, and the initial service tags for the four flows are $T_1 = 1, T_2 = 2, T_3 = 3, T_4 = 4$. The table maintained at each sender of the four flows and the backoff calculated for each flow are shown in Figure 5.

D. Analytical Properties of the Model

In the following, we briefly characterize the properties of the proposed model. Due to space constraints, we only outline the proofs.

Proposition III.1: (bounded flow unfairness of MLM-FQ) In a connected network graph, if each network node adopts the "Maximize-local-minimum" fair queueing (MLM-FQ) model, then for any two backlogged flows f and g in the network, their received services $W_f(t_1, t_2)$ and $W_g(t_1, t_2)$ during time interval $[t_1, t_2]$ satisfy:

$$|\frac{W_f(t_1,t_2)}{r_f} - \frac{W_g(t_1,t_2)}{r_g}| < \Delta \tag{2}$$

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 graph, and the fact that a flow receiving global minimum (normalized) service must be a local minimum too. □

Corollary III.1: (Long-term fairness of MLM-FQ) For any continually backlogged flow f, the MLM-FQ model achieves long-term fairness for flow f:

$$\lim_{t \to \infty} \frac{W_f(0,t)}{t} = r_f / k_f \tag{3}$$

where k_f is a topology-dependent constant.

Proposition III.2: (Minimum fair share for each flow in the network) Each of the three models MLM-FQ, EMLM-FQ and BFMLM-FQ guarantees that each continually backlogged flow f receives a minimum fair share of the channel C. That is,

$$W_f(t_1,t_2) \geq C \frac{r_f}{k \sum_{g \in \mathcal{S}} r_g}(t_2 - t_1) - \alpha \tag{4}$$

where k and α are two topology-dependent constants, and $\mathcal{S}$ denotes all flows that are in the one-hop neighborhood of flow f in the flow graph.

Proof The proof is based on induction in the connected graph and calculations derived from SFQ [4]. □

Remark III.1: The above proposition only gives the lower bound on the service that each flow receives. However, the upper-bound can be unlimited for the EMLM-FQ model.

Proposition III.3: (bounded flow unfairness of BFMLM-FQ) In a connected network graph, for any two backlogged flows f and g in the network, their received services $W_f(t_1, t_2)$ and $W_g(t_1, t_2)$ during time interval $[t_1, t_2]$ under the BFMLM-FQ model satisfy:

$$|\frac{W_f(t_1,t_2)}{r_f} - \frac{W_g(t_1,t_2)}{r_g}| < \beta \tag{5}$$

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

Proof Since the sliding window mechanism bounds the service difference in each of the one-hop neighborhood in the flow graph. Using arguments similar to Proposition III.1, the results follow readily. □

IV. A DISTRIBUTED IMPLEMENTATION WITHIN THE CSMA/CA FRAMEWORK

In this section, we will present a practical implementation of the proposed models within the framework of the CSMA/CA MAC paradigm. Our implementation seeks to address the following practical issues:

• *Exchange of the table information at a flow's sender and its receiver*: In the models described in Section III, each node in an ad hoc network maintains information for flows within one-hop neighborhood in the flow contention graph. However, one-hop neighborhood in a flow graph will translate to the two-hop neighborhood in the real node graph in practice (recall in Figure 1, flow F_1 one-hop neighborhood includes F_2, F_3, F_5, F_6). Therefore, given a flow f, our proposed algorithms require us to maintain flow information for flows that are within the transmission range of either f's sender or its receiver. However, for any given node, our goal is to maintain flow information (i.e., service tags) for flows only within its one-hop neighborhood in the node graph. That is, no node needs to be aware of flow information at nodes that are more than one hop away in the node graph. We will address this issue in our implementation.

• *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 in the flow graph. Whenever a flow transmits, either the senders or the receivers of its neighboring flows should update the new service tag for this flow. This operation is critical for the implementation of our model and all related approaches [10]. One common approach is to include the new service tag either in the control messages of RTS and CTS, or packets DATA or ACK. However, spatial reuse may prevent certain flows always hear collisions and never got their relevant table entries updated (e.g., in Figure 1, when F_1 and F_4 are transmitting simultaneously, F and C nodes will always hear collisions and will not be able to hear the new service tags of F_1 and F_4).

• *Hidden terminal problem in the shared-channel multihop wireless network.* This is a well-known problem in such networks [13].

A. Protocol Overview

A.1 Basic Message Exchange Sequence

In our protocol, each data transmission follows a basic sequence of RTS-CTS-DS-DATA-ACK handshake, and this message exchange is preceded by a backoff of certain number of minislot times. When a node has a packet to transmit, it waits for an appropriate number of minislots before it initiates the RTS-CTS handshake. Specifically, the node checks its local table and sets a backoff timer for flow f to be the number of flows with tags smaller than the tag of flow f. This way, the local minimum-tag flow backs off for zero minislot and contends for the channel immediately. If the backoff timer of 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, it cancels its backoff timer and defers until the ongoing transmission completes; In the meantime, it updates its local table for the tag of the on-going neighboring 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 a 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 (other motivations for DS have been explained in [13]). When hosts hear either a CTS or a DS message, they will defer until the DATA-ACK transmission completes.

A.2 Maintaining table information at both the sender and the receiver

Since the one-hop neighboring flow information of any flow is distributed at either its sender or its receiver, this will specifically affect the backoff value setting for each flow. In the model described in Section III, for flows that have smallest service tags in their local tables, the backoff is zero; for each flow f in concurrent transmissions due to channel reuse, its backoff is set to be the number of flows in the table whose service tags are less than flow f. According to this policy, we should set the backoff value for a flow, by taking into account both tables at the sender and the receiver. That is, flow f's backoff $B_f = B_f^S + B_f^R$, where 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 at the receiver 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 can simply delete this flow to avoid double counting; this can be easily achieved at the flow join-in phase. Then, in this scenario, a straightforward solution would be to broadcast the receiver's table in its one-hop neighborhood (of the node graph). However, if the table is big and being updated frequently, this will induce significant overhead, also this information may have to be retransmitted when collision happens. In essence, we have to reduce the communications between the sender and the receiver, which are prone to collisions.

In our design, we provide a better solution: if node N is the sender of flow g, sender N knows precisely the backoff value B_g^S for a flow at the sender's table, but does not know B_g^R. We will let the sender estimate B_g^R. To this end, whenever a flow g is transmitting through the RTS-CTS-DS-DATA-ACK sequence, the ACK packet carries two parameters: M_g and b_g in order for the sender to estimate B_g^R later on. M_g tells us how much services (in bytes) toward other flows have to be served before flow g transmits its packet in the receiver's table. $M_g = \sum_{j \in B}(T_j - T_g)w_g$, where w_g is flow g'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_g of flow g. b_g denotes the backoff value for flow g at its receiver's table. When the sender N receives this information, it records M_g for flow g, as well as the current time t_g when the sender receives this information. Then, at any given later time instant t, sender N estimates $B_g^R \approx b_g \cdot (M_g - C \cdot (t - t_g))/M_g$ where C is the channel capacity. Then the sender sets backoff for flow g as $B_f = B_f^S + B_f^R$. When the backoff timer B_f expires, we initiate RTS-CTS handshake and convey B_f^R back to the receiver to verify its estimation.

A.3 Propagating a flow's updated service tag

In order to propagate a flow's service tag to all its one-hop neighbors in the node graph 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 packets 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 transmission. We still propagate this old flow tag to help correct some stale information in the one-hop neighborhood of the sender or the receiver. On the other hand, once the handshake of RTS and CTS is successful, 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. Whenever collision happens, we invoke the standard random backoff algorithm.

Note that a flow f's sender always has *correct* information (i.e., current service tag) on f and it is responsible to propagate this accurate information to its neighbors. Hence, only *accurate* information will be propagated in each local neighborhood of the network graph.

B. Simple Overhead Analysis

We now perform a simple analysis on the message overhead for data transmissions in our protocol. It is easy to show that the efficiency of our protocol, defined as the ratio of data portion (in bits) in each data transmission (including the control message overhead) is given as:

$$\eta = \frac{T_{data}}{T_{data} + T_{RTS} + T_{CTS} + T_{DS} + T_{ACK}}$$

If we follow the 802.11 MAC specification, we choose a very conservative parameter set: 24 bytes for RTS, 18 bytes for CTS and DS, 20 bytes for ACK, 6 backoff slots (each slot is set to be 5 bytes), and the data packet is 512 bytes. Then the efficiency will be always larger than 83% and the overhead is less than 17%, and this is acceptable in practice in order to achieve performance bounds at the packet granularity; also this is comparable to the overhead of IEEE 802.11 MAC. More importantly, since we make explicit efforts in scheduling, the chance of collisions is also reduced. In typical simulations for fair flow sharing, we observe that the performance gain because of reduced collisions by judicious scheduling may compensate this overhead, thus effectively maintaining or increasing the overall throughput.

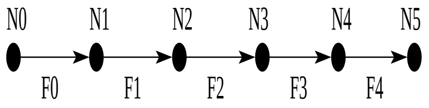

Fig. 6. Ex. 1: Node graph

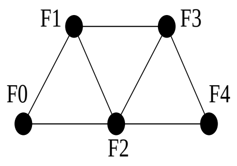

Fig. 7. Ex. 1: Flow graph

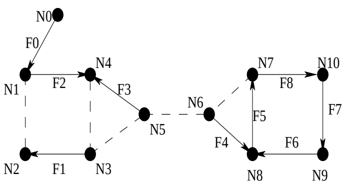

Fig. 9. Ex. 2: Node graph

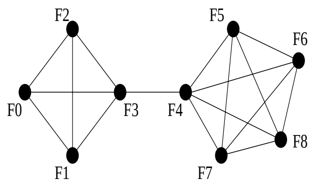

Fig. 10. Ex. 2: Flow graph

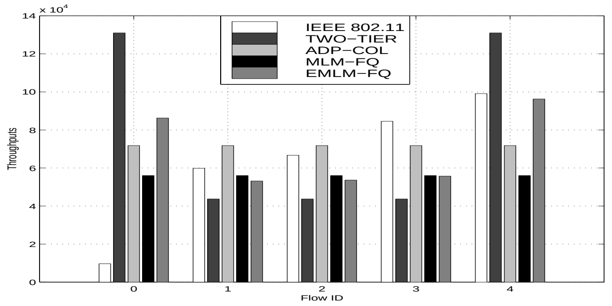

Fig. 8. Ex. 1: Throughput comparison

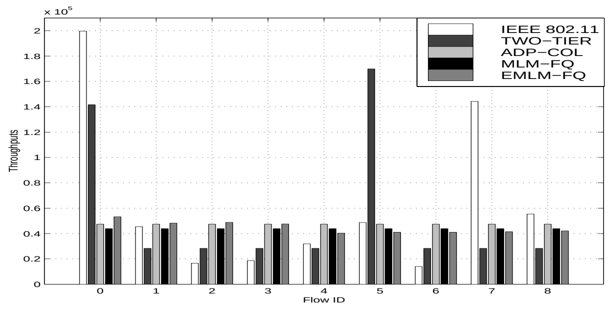

Fig. 11. Ex. 2: Throughput comparison

V. Simulation Evaluation

In this section, we use simulations to evaluate our proposed algorithms. We compare our Maximize-local Minimum Fair Queueing (MLM-FQ) algorithm, Enhanced Maximize-local Minimum Fair Queueing (EMLM-FQ) algorithm, with three other protocols proposed in the literature: the two-tier fair scheduling (TWO-TIER) model [18], the adaptive-coloring-based fair queueing (ADP-COL) algorithm [12], and FIFO with the IEEE 802.11 MAC protocol. The simulator was implemented within the *ns-2* simulator. The radio model is based on existing commercial wireless network with a radio transmission range of 250 meters and channel capacity of 2Mbit/sec. We simulate flows with Constant Bit Rate application model in *ns-2*, where the data packet size is set to be 512 bytes. Each simulation is run for 1000 seconds, and the throughput is counted as number of packets. We use identical flow weights in order to compare with the IEEE802.11 standard.

A. Simulation Scenario 1

In this example, we simulate five flows in a linear topology shown in Figure 6 (the flow graph is shown in Figure 7). The throughputs using the five algorithms are given in Table I. We observe that MLM-FQ indeed achieves inter-flow fairness, and EMLM-FQ achieves much better spatial reuse in addition to ensuring minimum fair throughput (8% higher than FIFO with IEEE 802.11 MAC). In this case, we also observe that the TWO-TIER achieves highest system throughput due to its design efforts to maximize spatial reuse [18]. Besides, ADP-COL achieves maximal inter-flow fairness due to its adaptive coloring design [12], but this is at the cost of more design complexities [12].

Flow	802.11 MAC	TWO-TIER	ADP-COL	MLM-FQ	EMLM-FQ
0	9649	131010	71805	56036	86291
1	59882	43714	71794	56034	53153
2	66735	43715	71791	56035	53656
3	84605	43716	71790	56036	55739
4	99124	131009	71810	56034	96268
Total	319995(100%)	393164(122%)	358990(112%)	280175 (80%)	345107(108%)

TABLE I
EX. 1: THROUGHPUT COMPARISONS

B. Scenario 2

In this scenario, we simulate nine flows in two cliques as shown in Figure 9 (the flow graph is shown in Figure 10). The simulation results are given in Figure 11 and Table II. In this case, since the flows form two cliques that are both heavily congested thus making spatial reuse infrequent, we do not achieve much spatial reuse in EMLM-FQ (only 2% increase compared with MLM-FQ), but both algorithms ensure minimum fairness. The Table also shows that IEEE 802.11 achieves best overall throughput, but this is achieved at the cost of throughput sufferings of flows F_2, F_3 and F_6.

C. Scenario 3

In this example, we simulate ten flows in the node graph of Figure 12 (the flow graph is Figure 13). The results are shown in Figure 14 and Table III. In this case, we can see that MLM-FQ still achieves inter-flow fairness, but at the cost of overall throughput (49% of 802.11, which starves flows F_4, F_7, F_8 and F_9). EMLM-FQ improves the overall throughput to 85%. In this case, TWO-TIER achieves best throughput, and ADP-COL achieves maximal fair throughput, but these are achieved at the cost of higher design and implementation complexity.

D. Scenario 4

We next simulate a large topology with 21 flows (shown in Figures 15 and 16). The simulation results are given in Figure 17 and Table IV. From the results, we observe that MLM-FQ still ensures absolute inter-flow fairness at the cost of overall channel utilization. However, in this case, EMLM-FQ performs

Flow	802.11 MAC	TWO-TIER	ADP-COL	MLM-FQ	EMLM-FQ
0	199653	141575	47468	43857	53228
1	45424	28311	47448	43856	48263
2	16644	28312	47449	43857	48782
3	18673	28313	47467	43856	47552
4	31823	28314	47448	43855	40359
5	48686	169868	47449	43856	41053
6	14005	28312	47450	43855	41012
7	144219	28313	47448	43856	41409
8	55403	28314	47449	43855	42075
Total	574530(100%)	509632(88%)	427076(74%)	394703(69%)	403733(71%)

TABLE II
EX. 2: THROUGHPUT COMPARISONS

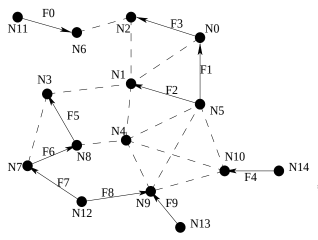

Fig. 12. Ex. 3: Node graph

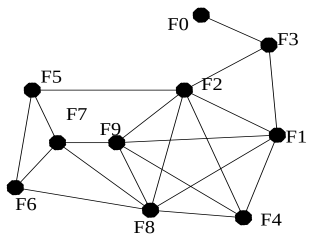

Fig. 13. Ex. 3: Flow graph

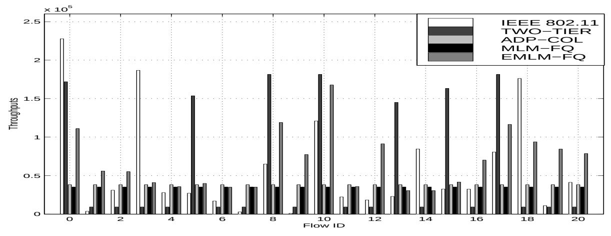

Fig. 17. Ex. 4: Throughput comparison

Flow	802.11 MAC	TWO-TIER	ADP-COL	MLM-FQ	EMLM-FQ
0	227785	172003	38163	35223	110946
1	3573	9286	38144	35224	55976
2	31006	9286	38163	35225	55045
3	186793	9287	38144	35227	40967
4	27874	9286	38143	35226	35543
5	27073	153710	38143	35228	39834
6	17023	9286	38144	35229	34790
7	2810	9286	38143	35231	35123
8	65098	181432	38163	35230	118945
9	938	9286	38143	35230	77309
10	120975	181431	38163	35233	167683
11	22330	9286	38163	35231	35850
12	18345	9287	38143	35232	91333
13	22942	145149	38163	35230	30559
14	84657	9286	38143	35231	30483
15	32583	163292	38163	35231	41662
16	32420	9286	38143	35230	70045
17	80647	181439	38163	35232	116547
18	176150	9287	38143	35232	93859
19	10933	9286	38144	35232	84380
20	41223	9286	38143	35231	78564
Total	1233178(100%)	1308463(106%)	801167(64%)	739818(60%)	1445443(117%)

TABLE IV

EX. 4: COMPARISONS OF FIVE PROTOCOLS

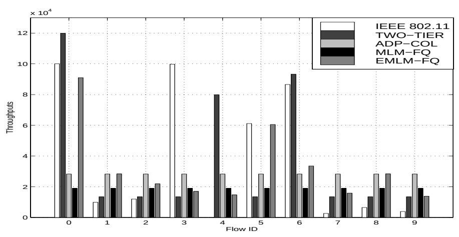

Fig. 14. Ex. 3: Throughput comparison

extremely well and achieves the highest system throughput with minimum fair throughput assurances. In this case, we can see that 802.11 starves multiple flows, TWO-TIER still gives very high throughput, and ADP-COL achieves maximal fair throughput for each flow.

VI. DISCUSSIONS AND RELATED WORK

In previous sections, we have described our proposed localized models and their practical implementations within the CSMA/CA paradigm. We now comment on several design issues and related works in the literature.

A. Further Comments

A.1 Node mobility

In an ad hoc network, each node can be mobile, thus changing the network topology dynamically. In a highly mobile wireless network, any model that requires global topology information or global computation does not work. Note that both our proposed localized models and their implementations require only one-hop neighborhood flow information (i.e., each flow's ID, and its current service tag) and simple local computation, this feature makes our design work well in the presence of node mobility. However, if a node is mobile, it does take several packet transmission times to discover its new neighborhood and its table in the new location. Ongoing work seeks to evaluate this aspect via simulations.

Flow	802.11 MAC	TWO-TIER	ADP-COL	MLM-FQ	EMLM-FQ
0	100020	119921	28306	19032	90994
1	9867	13501	28287	19039	28371
2	11984	13503	28289	19033	21884
3	99804	13502	28304	19037	16973
4	1	79906	28282	19037	14695
5	61129	13503	28281	19039	60540
6	86587	93319	28306	19035	33528
7	2625	13511	28287	19031	15864
8	6486	13513	28288	19033	28350
9	3881	13514	28286	19038	13871
Total	382384(100%)	387693(101%)	282916(74%)	190354(49%)	325070(85%)

TABLE III

EX. 3: THROUGHPUT COMPARISONS

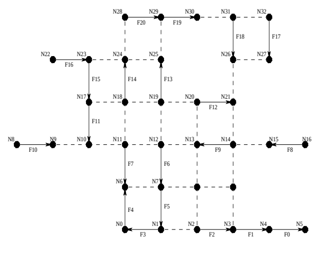

Fig. 15. Ex. 4: Node graph

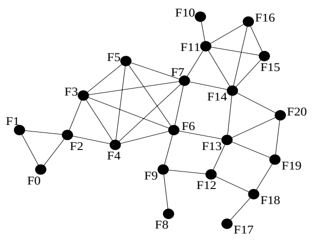

Fig. 16. Ex. 4: Flow graph

A.2 Scalability

Another feature of our proposed design is that it scales well in a large-scale or dense ad hoc network. This is because in our design each node only communicates with its one-hop neighbors in the node graph; the information maintained is also minimal, only parameters such as current service tag and flow weight per flow are needed. The computation workload (i.e., tag assignment, sorting flows based on current virtual times, etc.) performed at each node is also very light.

A.3 Other issues

Our design described in this paper mainly targets fixed packet size, since this is realistic in typical wireless networks. But our design also works for variable packet size scenarios since the tagging update mechanism has counted the packet size and our scheduling is based on the service tags of local contending flows. In our maximize-local-min algorithm, non-local-minimum flows defer transmissions for flows with local minimum tags. In our enhanced maximize-local-min algorithm, flows contend for spatial reuse; even though large packet size may hurt the local minimum flows temporarily, the local minimum flows always have highest priority for channel access (i.e., they have backoff zero) once the channel is idle.

A.4 Comparisons between fully distributed algorithms and our localized coordination approach

A final comment is on the difference between our localized coordination approach and a standard fully distributed algorithm. We should clarify that our self-coordinating approach is a class of fully distributed algorithms, but it emphasizes more on structured design on fully distributed algorithm, while a common distributed algorithm design for fair queueing is typically done in an ad hoc manner and seeks to approximate the centralized algorithm. In contrast, we will use the centralized algorithm to extract the key global *properties* that we would like to achieve through the local algorithms, then develop local models that capture these global behaviors, and finally design implementation protocols to realize the proposed local models.

B. Related Work

Packet scheduling for both wired and packet cellular networks has been an intensive research area in the networking field [1]–[9]. In multihop wireless networks, providing minimum throughput and bounded delay access has been studied at the MAC layer and some representative works on this topic include [14]–[17]. 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.

Three more recent works address packet scheduling issues in ad hoc wireless networks [10] [12] [18]. [10] studies the problem of fair queueing in ad hoc networks, and the focus has been to define an appropriate centralized model (i.e., the GSR model in the paper). It also provides a backoff-based distributed implementation, but leaves out many critical details. Another independent work [12] also seeks to formulate the problem of ad hoc fair queueing, and the authors seek to maximize channel spatial reuse while ensuring fairness. Again the focus has been to define an ideal centralized model, though it also provides a tree-based distributed implementation. Finally, [18] presents a centralized model and a distributed implementation protocol for packet scheduling in ad hoc networks, but the focus there is how to resolve the conflicts between fairness and maximal throughput and arbitrate the tradeoff between these two.

In another recent work [11], the authors address the issue of distributed fair queueing in a wireless LAN environment. The proposed design only works for a completely connected graph. Interested readers may wonder whether we can apply the same algorithm in multihop wireless networks. It turns out this is not feasible for an ad hoc network for multiple reasons. First, in a fully connected graph, it may be true that we do not need to maintain a virtual clock, but this is definitely not true for ad hoc networks. Since the backoff approach of [11] has to subtract a system virtual time from each flow's current tag to compute the backoff time for each packet transmission, the accurate acquisition of the system virtual time becomes critical. This problem is trivial in a complete graph studied in [11]. It becomes unclear how to get the global system virtual time in a large-scale multi-hop ad hoc network where multiple flows may be transmitting simultaneously. Again, if we redefine the system virtual time only to a local neighborhood, what fairness model that it achieves becomes unclear. We illustrate this issue through the example of Figure 4. In the same problem setting, according to [11], the initial backoff values are set as $B_1 = T_1 - V1 = 1, B_2 = 2, B_3 = 3, B_4 = 4$. When F_1 is transmitted, the updated backoffs for F_2, F_3 are $B_2 = T_2 - 1 = 1, B_3 = 2$. When the backoff timer of F_4 expires, the backoffs for F_2 and F_3 will become $B_2 = 2 - 4 = -2, B_3 = 3 - 4 = -1$. This example shows the difficulty in directly applying the design of [11] in a multihop ad hoc network. Besides, large difference in flow weights (e.g., 0.1 for flow F_1, 0.001 for flow F_2) will bring large variance in the backoff values, and affect the performance. But our design does not have this problem. Furthermore, our MLM-FQ algorithm does not need synchronization, which is a clear advantage over the backoff-based approach that needs accurate synchronization. We do use the backoff mechanism in our design, but it is a secondary mechanism to improve spatial reuse. Besides, our backoff mechanism does not need global synchronization, and the local synchronization needed is provided by the data transmission handshake.

VII. Conclusion

In this paper, we have proposed a suite of new localized and fully distributed fair queueing model for ad hoc wireless networks. Our proposed models seek to devise scalable and efficient solutions to ad hoc fair queueing problem, and they provide fairness and increase spatial reuse. Our models rely on local information and local computations only, and multiple localized schedulers coordinate their interactions and collectively achieve desired global properties such as fairness, scaling and efficiency. We demonstrate the effectiveness of our proposed design 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 algorithms.

References

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

[2] 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.

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

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

[5] S. Lu, V. Bharghavan and R. Srikant, “Fair scheduling in wireless packet networks,” *IEEE/ACM Trans. Networking*, August 1999.

[6] M. Srivastava, C. Fragouli, and V. Sivaraman, “Controlled Multimedia Wireless Link Sharing via Enhanced Class-Based Queueing with Channel-State-Dependent Packet Scheduling,” *IEEE INFOCOM'98*, March 1998.

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

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

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

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

[11] N. H. Vaidya, P. Bahl and S. Gupta, “Distributed fair scheduling in a wireless LAN,” *ACM MOBICOM'00*, August 2000.

[12] H. Luo and S. Lu, “A topology-independent fair queueing model in ad hoc wireless networks,” *IEEE ICNP'00*, Nov. 2000.

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

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

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

[16] 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.

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

[18] H. Luo, S. Lu and V. Bharghavan, “A new model for packet scheduling in multihop wireless networks,” *ACM MOBICOM'00*, August 2000.