Achieving Delay and Throughput Decoupling in Distributed Fair Queueing Over Ad Hoc Networks

Jerry Cheng and Songwu Lu
UCLA Computer Science Department, Los Angeles, CA 90095
E-mails: {chengje, slu}@cs.ucla.edu

***Abstract*— This paper describes an algorithm that achieves delay and throughput decoupling in distributed fair scheduling in multihop ad-hoc wireless networks. The solution allows to support both low-bandwidth, low-delay and high-bandwidth, high delay applications in a single framework, without wasting much bandwidth to realize the low-delay requirement. We demonstrate the effectiveness of our algorithm in servicing various types of applications through *ns-2* simulations.**

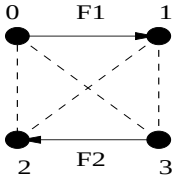

Fig. 1. Example 1

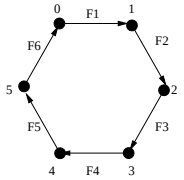

Fig. 2. Example 2

I. Introduction

In an ad hoc network, groups of networking devices communicate with one another using wireless radios. By virtue of their ability to operate with a peer-to-peer network model, ad hoc networks offer convenient infrastructure-free communication over the shared wireless channel. In such a wireless network, users are expected to run a rich set of data applications, e.g., both error-sensitive and delay-sensitive applications, over the bandwidth-constrained wireless medium. Therefore, 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 design paradigm to achieve packet-level qualities of service (QoS), in terms of bandwidth, delay, and fairness, in wired networks as well as wireless cellular network [1]–[5]. Several recent research efforts [7] have formulated the problem of fair packet scheduling in ad hoc networks to address the unique characteristics of such networks. These issues include location-dependent contention, the distributed nature of ad hoc fair queueing, channel spatial reuse, and how to manage a potentially large number of flows in a dense and mobile network graph. Furthermore, the algorithms have to be fully distributed and scalable.

In this paper, we propose a novel distributed fair queuing algorithm that ensures fair sharing of the wireless channel and provides delay and throughput decoupling. Delay and throughput decoupling is important over the bandwidth-constrained ad hoc wireless networks. Some network applications are more concerned with the fair share of bandwidth, e.g., file transfer, without stringent requirement on packet delay, while others, e.g., audio traffic, are more interested in low average delay but low throughput. How to efficiently support both high-throughput, high-delay applications and low-delay, low-throughput applications, within a single framework poses a unique challenge to a distributed fair queueing design. We also describe the implementation of the algorithm within the popular CSMA/CA paradigm. Through both simulations and analysis, we show that the distributed fair queueing design performs locally coordinated scheduling decision, decouples the delay and throughput requirement of applications, and *collectively* achieves desired global properties such as fairness, scaling and network efficiency.

The rest of this paper is organized as follows. Section II describes background information. Section III characterizes the desirable properties of the solution, and proposes a distributed fair queueing algorithm that achieves delay and throughput decoupling. Section IV describes the distributed implementation of our design and Section V presents our simulation results. Section VI discusses the design and Section VII discusses some related work. Section VIII concludes this paper.

II. Background

We focus our study on the packet-switched wireless network, where data transmissions are performed over the shared, multihop wireless channel. Since wireless data is locally broadcast, a transmission is successful only if the receiver is within the range of the sender. To achieve delivery from the source to the destination, the packet may have to traverse multiple wireless hops. We make the following standard assumptions: (1) a collision occurs when the receiver is within the range of two simultaneously transmitting nodes, (2) a node cannot transmit and receive a packet simultaneously, (3) the nodes have the ability to perform local carrier sensing, and (4) we do not consider random channel errors; link-layer retransmissions should be able to help from recovering from such interference-induced losses. The unique issues of such networks include:

1) *Channel Contention*: In a wireline network, the transmission of each link can be done independently and simultaneously. In a wireless environment, however, this feature does not hold. Each node must compete with neighboring nodes to access this shared medium. This may lead to severe transmission collisions. Consider Figure 1 as an example, where a 4-node topology is shown with 2 directed

lines and 4 dashed lines. The 2 direct lines represent two single-hop transmissions and the four dashed lines represent the nodes which are within the radio transmission range of one another. All the nodes are within a common neighborhood. In a wireline environment, Flows F_1 and F_2 can transmit packets simultaneously. However, if nodes N_0 and N_3 transmit simultaneously in a wireless environment, collisions will occur for both flows. In order to make this network functional, the transmission precedence has to be organized properly.

2) *Location-Dependent Spatial Reuse*: If two flows are out of each other's transmission range, they may transmit simultaneously without interfering with each other. This leads to better aggregate throughput. In the ring topology of Figure 2, there are three sets of flows that can transmit simultaneously by spatial reuse, Flows F_1 and F_4, Flows F_2 and F_5, and Flows F_3 and F_6. Without proper scheduling, some flow sets may be scheduled more often, resulting in unfair share of the channel.
3) *Delay*: In an ad-hoc network, packet delay may vary a lot. For throughput-sensitive data traffic such as FTP, delay is not a main concern. However, for delay-sensitive traffic such as audio, delay has a profound impact as excessive delay or delay jitter can render a real-time conversation inaudible.

III. Designing a Distributed Fair Queuing Algorithm

A. Basic Requirements

The following are the desirable properties for a distributed fair queueing algorithm:

- Fully Distributed: Since the packet arrival information is only available at each node, a distributed solution is needed.
- The solution must be efficient: If fairness is the only goal, we can schedule only one flow to transmit at a time in an orderly fashion. However, a lot of wastage can occur due to the squandered spatial reuse. There lies the tradeoff between the fairness concern and the aggregated throughput. The solution must handle this tradeoff properly in order to maximize the overall throughput. Specifically, the set of non-interfering flows that can transmit simultaneously needs to be selected judiciously so that the system throughput can be maximized under the fairness constraint.
- The design needs to be application aware: Different applications will have different requirements; some care more about throughput and others are more interested in average delay. Servicing delay-sensitive traffic once such packets arrive may provide a good delay bound, but might violate the fairness constraint. Strictly enforcing fairness at a fine granularity, on the other hand, would cause delay-sensitive traffic to miss its deadline. The solution has to meet requirements of both applications.

B. A Novel Distributed Fair Queuing Algorithm

We now present our distributed fair queuing algorithm that achieves delay and throughput decoupling. The base algorithm [9], is adapted from the popular Start-time Fair Queuing (SFQ). In SFQ, there are two main actions: tagging and scheduling. Tagging helps the system to maintain a sense of lead and lag in the amount of service each flow receives, while scheduling advances the system service by transmitting the lowest-tag flow in order to preserve fairness across all flows. The operations of the algorithm are as follows:

- Tagging: In SFQ, the current system virtual time is the tag of the packet being served. However, in distributed fair queueing, this information is not readily available at each node. Allowing a system-wide flood of the virtual time is too costly. Instead, we use a localized virtual time in the local neighborhood. During each transmission, each node can piggyback the current service tag with the packet, while the neighboring nodes overhearing the packet keep a copy of the service tag in order to determine the local virtual time. The local virtual time obviously may differ from the global virtual time. The tradeoff here is the inaccuracy in approximation. The tagging operation is as follows: 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:
  1) If f is continually backlogged, then $S_k^f = F_{k-1}^f;\ \ F_k^f = S_k^f + L_p/r_f$.
  2) If f is newly backlogged, then $S_k^f = \max_{g \in W}\{V_g(A(t_k^f))\};\ \ F_k^f = S_k^f + L_p/r_f$, where W consists of all flows stored in the table of node n, and $V_g(t)$ is flow g's virtual time at t.
- Scheduling: Identifying the smallest tag among all backlogged nodes is a global computation. We take a table-driven, backoff-based approach in scheduling transmission. The approach uses local information only and involves local computation. With the tagging and a method of exchanging tags in place, each node has the knowledge of its local neighborhood. These tags are stored in a table and ordered so that each node can learn whether that node itself has the minimum tag. Since we are also interested in maximizing spatial reuse, we do not confine the transmission to the minimum-tag holders only. Instead, at each node, we set its backoff value to be the total number of flows that have a smaller service tag. This way, the flow with the smallest service tag will transmit first (since it has the smallest backoff period), and other contending flows will restrain from transmissions once they hear the transmission through carrier sensing. In addition, flows that are not interfering with the minimum-tag flow can transmit concurrently, starting from the one with smaller backoff value. This will improve the spatial reuse and overall channel utilization. For each flow f, it sets its backoff period B_f as $B_f = \sum_{g \in W} I(T_g < T_f)$ minislots, where T_f and T_g denote the service tags of flow f and flow g, respectively, S is the set of all the neighboring flows in the table, and $I(x)$ denotes the indicator function, i.e., $I(x) = 1$, if $x > 0$; $I(x) = 0$, otherwise.

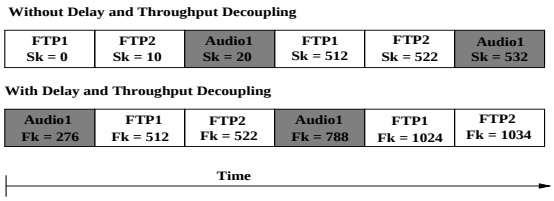

Fig. 3. Delay and Throughput Decoupling

The combination of the above two mechanisms allows us to select a set of non-interfering flows for transmission, including the flows with local minimum service tags. We call our method the Extended-Maximizing-Local-Minimum Fair Queuing (EMLM-FAQ).

C. Delay and Throughput Decoupling

Wireless application may experience large variation of delay due to the nature of shared wireless medium. In this section, we improve our distributed fair queueing scheduling of the previous section and provide a better quality of service to wireless traffic through our method of delay and throughput decoupling.

The idea behind delay decoupling is novel and simple. Among all the packets that can be scheduled in a round of transmission, we identify a set of delay-sensitive applications. Instead of applying the regular SFQ scheduling, we do local swapping of transmission order. The packets of delay-sensitive applications will be scheduled earlier than the packets of non-delay-sensitive applications. This way, we provide a better delay bound for them without violating the fairness property. The detailed operations are:

1) If f is continually backlogged, then $S_k^f = S_{k-1}^f + L_p^{k-1}/r_f$.
2) If f is newly backlogged, then $S_k^f = \max_{g \in W}\{V_g(A(t_k^f))\}$.
3) For both of above case $F_k^f = S_k^f + L_p^k/r_d$. where r_d is the delay weight.

Instead of scheduling packets according to its start tag, we would schedule it according to the finish tag. The finish tag is not used in calculating the next start tag, thus preserving the long-term fairness feature.

Figure 3 illustrates a scheduling example of two FTP applications and one audio stream. In this example, each data packet is 512 bytes. Without decoupling, scheduling these streams would cause the audio stream to be transmitted at the end of each round. With decoupling, we assign the audio stream with a delay weight of 2 and the two FTP sessions a delay weight of 1. Furthermore, we schedule the packets based on the finish tag calculated above. Through local swapping in transmission orders, the audio stream benefits from experiencing smaller delays.

IV. Implementation

This section describes a practical implementation of the distributed algorithm. The implementation is done within the CSMA/CA paradigm.

A. Queuing Level

At the queuing level, the goal is to maintain per application fairness within each outgoing flow. The basic operation is queuing and dequeueing of packets.

1) Queuing Process: When the queue first receives an outgoing packet, it will attempt to identify the packet type through the packet header. If the packet appears to be a control packet for the network, it is queued at the end of the high-priority control packet queue. However, if the packet appears to be a data packet, the queue manager will extract the end-to-end sender and receiver addresses, as well as the next-hop destination. Together with the node's ID, each queue in the network can be identified by these 4-tuple. The queue manager then buffers each application with respect to each outgoing wireless flow.

2) Dequeueing Process: When dequeuing, queuing manager will first check the control packet queue. If the control packet queue is empty, it will attempt to dequeue packets from the data packet queue. For the data packet queue, queuing manager will first check the flow tag table that contains all the flow tags. Out of all the links that originate from this node and are backlogged, it will select the link with the lowest flow tag to service first. Within each link, there could be many different backlogged streams. The queue manager's task is to maintain a separate tagging system within each link. We employ the delay decoupling method and choose the appropriate packet and deliver it to the MAC layer.

B. MAC Layer

1) Basic Message Exchange Sequence: In our protocol, each data transmission follows a basic sequence of RTS-CTS-DS-DATA-ACK handshake. Our message exchange sequence is preceded by a backoff of a 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. The number of minislots is set equal to the number of neighboring flows that has smaller flow tag than the node that wishes to transmit. If the backoff timer of a node expires without overhearing any ongoing transmission, the node is free to start RTS to initiate the handshake. If the node overhears some ongoing transmission, it cancels its backoff timer and defers until the completion of ongoing transmissions. For a receiver, when a RTS is received, it first checks its local table. If the flow's tag is also the smallest tag in the receiver flow tag table, then it responds with CTS. Otherwise, the receiver simply drops RTS. Once the CTS is received, the sender will continue with the DS-DATA-ACK sequence.

This method will prioritize the transmission of the node that has the smallest flow tag. Effectively, it ensures the fairness among the local neighbors. In addition, it allows for spatial reuse by permitting nodes that do not interfere with other ongoing transmissions to transmit, thus increasing the system throughput.

2) Maintaining table information at both the sender and the receiver: The information of the neighboring flows is distributively stored at the sender and receiver. This information needs to be combined and known by the sender to make the scheduling decision. A straightforward solution would be to broadcast

the receiver's table to the sender periodically. However, significant overhead will be induced if the table is large and updated frequently. 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 transmitted 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 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 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.

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 the 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 tags 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 correct some stale information in the one-hop neighborhood of the sender or the receiver. When the handshake of RTS and CTS is completed, 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.

V. Simulation Evaluation

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

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 sessions, the model in the ns-2 package is used. For the video traffic, actual trace is used through the traffic trace functionality of the ns-2 simulator. For TCP, TCP New Reno is used. All packets are set to 512 bytes, except video traffic has a varying packet size. We used Dynamic Source Routing (DSR) as the routing protocol.

We present four different scenarios. Scenarios 1 and 2 are designed to illustrate the fairness of our fair queueing algorithm servicing different types of streams. Scenarios 3 and 4 are designed to evaluate the effectiveness of our method in decoupling delay and throughput.

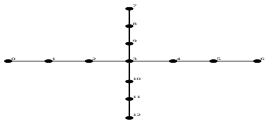

Fig. 4. Cross Topology

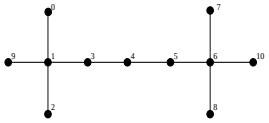

Fig. 5. Dumbell Topology

Flow	MAC 802.11	EMLM-FQ
0 to 7 (TCP)	525	2796
2 to 8 (TCP)	805	2741
9 to 10 (UDP)	13743	4819

TABLE I

SCENARIOS 1: THROUGHPUT

Flow	MAC 802.11	EMLM-FQ
0 to 6	8782	3814
7 to 12	1614	3845

TABLE II

SCENARIO 2: THROUGHPUT

A. Scenario 1

In this example, we demonstrate the effectiveness of the EMLM-FQ protocol in preventing malicious traffic from capturing the wireless channel via aggressively transmitting its own traffic. We use a dumbbell topology of Figure 5 with three streams of two FTP-driven TCP streams and one CBR-driven UDP stream. The two TCP streams originate from node 0 and node 2 and terminate at node 7 and node 8, respectively. The CBR-driven UDP is sent at an aggressive rate from node 9 to node 10 so that there will always be packets backlogged at node 9.

The result of the simulation is shown in Table I. Without fair scheduling, IEEE 802.11 fails to provide adequate service to the two TCP streams, while UDP captures the majority of the channel bandwidth. EMLM-FQ provides better sharing and prevents the two TCP streams from starvation.

B. Scenario 2

In this example, we want to demonstrate that our fair queueing algorithm can help preserve the fair sharing of two TCP flows having the same setup. We use a cross topology shown in Figure 4, with six hops on each path. Two FTP-driven TCP streams are used, one from node 0 to node 6, while the other from node 7 to node 12.

The result is shown in Table II. FIFO scheduling with 802.11 MAC will lead to unfair share of the channel. EMLM-FQ, on

Flow	MAC 802.11	EMLM-FQ
0 to 7	0.184881	0.048897
2 to 8	0.124382	0.103009
9 to 10	0.606635	0.127416

TABLE III

SCENARIO 3: DELAY JITTER IN SECONDS

Flow	EMLM wo DD	EMLM w DD
0 to 7 (audio)	653	653
2 to 8 (CBR)	7444	7443
9 to 10 (CBR)	7467	7469

TABLE IV

SCENARIO 4: THROUGHPUT

Flow	EMLM w/o DD	EMLM w/ DD
0 to 7 (audio)	0.413267	0.301599
2 to 8 (CBR)	0.302766	0.321640
9 to 10 (CBR)	0.356335	0.407058

TABLE V

SCENARIO 4: AVG. DELAY (SEC)

the other hand, is operating at its best by providing fair service among the two contending TCP flows.

C. Scenario 3

In this scenario, we focus on multimedia traffic. We simulate a dumbbell topology with three video-driven UDP streams using the actual MPEG trace of three movies. The set-up is similar to Scenario 1 with three streams from node 0 to node 7, node 2 to node 8, and node 9 to node 10, respectively. For typical streaming video, we are interested in the delay jitters of the network. Having a large delay jitter will result in poor picture quality that appears to be choppy.

The delay jitter of the three video streams is shown in Table III. As the table shows, having EMLM-FQ schedule the channel access does provide better performance in terms of delay jitters. The original MAC 802.11 standard employs a binary exponential backoff method in resolving contention. Whenever a node successfully transmits its packet, the contention window is reset to the minimum window size. This results in bursty transmissions and will lead to larger delay jitters.

D. Scenario 4

In this scenario, we study the effectiveness of our delay and throughput decoupling algorithm. This scenario takes a similar setup as the dumbbell topology of Figure 5, with two CBR-driven UDP flows and one audio-driven UDP flow. The two CBR-driven flows originate from node 2 and node 9, and end at node 8 and node 10, respectively. The audio stream goes from node 0 to node 7, and is operating at 8.8 Kbits/sec. The two CBR sources transmit data at 100 Kbits/sec.

With the 802.11 MAC standard, the audio stream will starve under the more aggressive CBR-driven UDP streams. Hence, we exclude 802.11 MAC from study, and focus on comparing results of EMLM-FQ with and without delay decoupling. The simulation result is presented in Tables IV and V. In Table IV, we see that the overall throughput remains almost the same with only slight variations due to randomness. After verifying the throughput, we move on to study the average delay of the three traffic streams. From Table V, we can see that the audio traffic benefits from the delay decoupling by reducing its average delay from about 0.41 sec to about 0.3 second. This is achieved at the cost of slight increase of average delay of the two CBR-driven UDP flows.

E. Simulation Summary

After examining the set of scenarios we have simulated, we conclude that EMLM-FQ holds a a clear advantage over the original FIFO scheduling with 802.11 MAC. The utilization of multiple queue prevents the aggressive UDP from overflowing the buffer of the bottleneck node and starving other well-behaving applications. In addition, the multiple queue allows delay and throughput decoupling.

The distributed fair queuing algorithm also fares quite well in comparison to FIFO scheduling with 802.11 MAC. The fair scheduling component helps reduce the bandwidth capture effects. Unlike FIFO with 802.11 MAC, EMLM-FQ shows that its flows, under similar conditions, can obtain an equal share of service from the network. It makes packet transmissions more regular as compared to the random burstiness nature of FIFO. As a result, multimedia traffic such as video traffic would experience a smaller delay jitter, which translates to a better streaming video quality.

The EMLM-FQ does have the drawback of lower throughput as compared to FIFO with 802.11 MAC. There are three factors contributing to this feature. First, to facilitate the exchange of flow tag between neighboring nodes, we have added a DS packet that introduces some extra overhead. Second, the added backoff period before the packet transmission for our distributed scheduling purpose is another source of overhead. These two factors are the tradeoffs for the ability to provide fair queuing to the wireless network. Third, in a wireless environment, fairness may be in conflict with maximizing the aggregated system throughput. Note that achieving maximum system throughput is equivalent to identifying a maximal set of flows that can transmit simultaneously and having this set of flows continuously transmit without relinquishing the channel. However, fairness may be violated in this extreme scenario.

VI. Discussion

We now discuss several related issues.

A. Synchronization

The distributed fair queuing for channel access uses a novel table-driven backoff approach to prioritize different links' transmissions. For this method to be successful, the contending nodes must be roughly synchronized with respect to each other. This is not a problem within the local neighborhood of each node. The ability to carrier sense the channel allows the nodes to achieve local synchronization. The problem is more severe when the node's *nearby* contenders are not within the transmission range. In this situation, the node will have no way of knowing for sure when those farther away nodes start to backoff their number of minislots. In terms of long-term fairness, this is not a problem as the notion of lead and lag of each link is kept by their respective service tags. However, it does have an effect on the short-term fairness in that the exact order of transmission might be changed.

B. Table Inconsistency

Although efforts are made for each node to broadcast its service tag correctly, there is no guarantee for the service tag information to be delivered to each neighboring node. Collision and/or channel error can sometimes induce table inconsistency between neighboring nodes. This is another possible cause for coarse fairness as stale table entries can make the neighboring nodes end up having a different transmission order. However, this inconsistency issue is temporal and correctable. Future service tags updating from overhearing any of the more recent control packets will help correct this temporary table inconsistency.

VII. Related Work

Packet scheduling has been one of the main networking research areas. Many efforts have been made in studying fair queuing on wireline and cellular wireless networks [1]–[5]. These solutions provide the basis for our study but do not address the issues of the multihop wireless network.

In multihop wireless network, [7] and [8] study the problem of ad-hoc fair scheduling and the focus of each work centers around defining fair queuing in the ad hoc wireless network. In particular, each work provides a trade-off between fairness and overall throughput. The approaches these papers took are to assume an ideal centralized scheduler and offer a glance of how it can be distributively implemented. In two more recent proposals [6] and [9], the methods are refined in the distributed implementation of fair queuing. The solution of [6] is designed for a wireless LAN where all the nodes were within the communication range. The work in [9] is our previous effort in designing distributed fair queuing for multihop wireless networks. The contributions include a table-base backoff algorithm that provides a mechanism for maintaining fairness and conserving the spatial reuse. These research efforts, however, do not take on an application perspective, and cannot support multiple classes of application, with different delay and throughput requirements, in a single framework. This is the focus of this work.

VIII. Conclusion

In this paper, we proposed a distributed fair queueing solution for providing scheduling service in the ad hoc wireless network. At the link level, the distributed fair queueing handles the scheduling of wireless channel access base on only local information and computation. At the queueing level, it achieves delay and throughput decoupling. The simulation results demonstrate the effectiveness of our proposed solution in servicing both bandwidth-sensitive and delay-sensitive applications.

References

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

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

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

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

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

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

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

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

[9] H. Luo, P. Medvedev, J. Cheng, and S. Lu, "A Self-Coordinating Approach to Distributed Fair Queueing in Ad Hoc Wireless Networks," *IEEE INFOCOM'01*, April 2001.