Evaluation of Packet Scheduling Algorithms in Mobile Ad Hoc Networks

Byung-Gon Chun **Mary Baker**
bgchun@cs.stanford.edu *mgbaker@cs.stanford.edu*
Computer Science Department, Stanford University, Stanford, California

We examine the queuing dynamics at nodes in an ad hoc mobile network and evaluate network performance under different packet scheduling algorithms using Dynamic Source Routing (DSR) and Greedy Perimeter Stateless Routing (GPSR) as the underlying routing protocols. Typically, packet schedulers in ad hoc networks give priority to control packets over data packets and serve data packets in FIFO order. We find that setting priorities among data packets can decrease end-to-end packet delay significantly. In particular, we find that among the algorithms we studied, those that give priority to data packets with short distance metrics show the smallest delay and the highest throughput, without increasing routing overhead. In addition, we show that with both DSR and GPSR, giving priority to control packets over data packets affects the performance significantly when mobility is high. With DSR, giving priority to control packets reduces the average delay. In contrast, with GPSR, this scheduler increases the average delay.

I. Introduction

In mobile ad hoc networks, the mobility of nodes and the error-prone nature of the wireless medium pose many challenges, including frequent route changes and packet losses. Such problems increase packet delays and decrease throughput. As traffic load in the network increases, the performance degradation gets worse. Research in this area has focused primarily on routing protocols – how to route packets hop by hop as efficiently as possible [1, 2, 3, 4, 5, 6, 7] and medium access control (MAC) – how to share the medium efficiently [8, 9, 10, 11]. However, there is little understanding of the queuing dynamics in the nodes of these networks and there is no thorough investigation of the effects of different packet scheduling algorithms in the queues of the nodes. In this paper, we analyze different packet scheduling algorithms to find those that most improve performance in congested networks.

Ad hoc networks have several features, including possible frequent transmissions of control packets due to mobility, the multi-hop forwarding of packets, and the multiple roles of nodes as routers, sources, and sinks of data, that may produce unique queuing dynamics. We believe that the choice of scheduling algorithm to determine which queued packet to process next may have a significant effect on overall end-to-end performance when traffic load is high. This belief motivated us to evaluate several applicable scheduling algorithms.

The questions we address are

- How do the queuing dynamics change under different degrees of mobility, traffic loads, and routing protocols?
- What are the effects on performance of giving high priority to control traffic? Are the effects dependent on the routing protocols used?
- What are the effects on performance of setting priorities in data traffic? What scheduling algorithms improve performance?

To answer these questions, we first analyze queueing in the nodes of an ad hoc network. Next, we evaluate the effects of different scheduling algorithms on delay and throughput.

We perform this study with two very different routing protocols: the Dynamic Source Routing (DSR) protocol [1] and the Greedy Perimeter Stateless Routing (GPSR) protocol [7]. DSR is an on-demand, non-geographic routing protocol and GPSR is a proactive, geographic routing protocol. We use ns-2 [12] with wireless extensions [3] as the simulation tool.

We first observe that the benefit of giving priority to control packets over data packets depends on whether the routing protocol used is DSR or GPSR. With DSR, a priority scheduler (which gives priority to control packets over data packets and serves data packets in FIFO order) reduces the average delay compared to a no-priority (straight FIFO) scheduler. In contrast, with GPSR, the priority scheduler increases the average delay compared to the no-priority scheduler. With

both DSR and GPSR, there is little difference in average throughput from giving priority to control packets.

Our most important result is that the scheduling algorithms that give higher weight to data packets with smaller numbers of hops or shorter geographic distances to their destinations reduce the average delay significantly and improve the average throughput. With DSR, servicing more data packets with fewer remaining hops reduces the average delay by up to 32% compared to the priority scheduling. With GPSR, servicing more data packets with shorter geographic distance reduces the average delay by up to 32%. The use of our scheduling algorithms affects the routing overhead very little.

The rest of this paper is organized as follows. Section 2 describes the salient features of the DSR and GPSR protocols. Section 3 details the scheduling algorithms studied. Section 4 describes the methodology and the performance metrics used. In Section 5, we explain the queuing dynamics observed. We describe the simulation results of giving priority to control packets in Section 6. We present the simulation results of different scheduling algorithms in Section 7. Section 8 describes related work in this field. In Section 9, we present interesting future research questions. Finally, Section 10 details our conclusions.

II. Routing Protocol Description

In this section, we provide the essential details of the DSR and GPSR protocols. We choose these two protocols to experiment with diverse aspects of routing protocols: on-demand, proactive, non-geographic, and geographic routing features.

II.A. DSR

DSR is an on-demand, source routing protocol. Transmitting nodes discover the route to their destination nodes on demand. This route is included in the data packets as the route header.

The DSR protocol consists of two phases - *Route Discovery* and *Route Maintenance*. When a source node A wants to send a packet to a destination node B, it first checks if it already has a route to B stored in its cache. If the route is not stored, a *Route Request* (RREQ) packet is broadcast with the address of node A in the route record. An intermediate receiving node checks if its route cache has a route to the destination node. In that case, it appends the route in the route record and sends back a *Route Reply* (RREP) packet by using the reverse route (assuming symmetrical links). If the intermediate receiving node does not know the route, it appends its own address to the route record and broadcasts another RREQ packet. Using the route cache helps conserve network resources by limiting the number of route discovery packets. It is possible that a node receives route request packets for the same source-destination pair but with different route headers that represent different routes. In this case, the node chooses the route with the lowest cost, usually the shortest path.

When a destination node receives the RREQ packet, it then appends its address to the route record and sends back an RREP packet. If the links are symmetric, it constructs the route to the source by reversing the route in the received route record. If asymmetric, the destination initiates a route discovery to the source and receives the RREP packet upon its successful completion. When the source begins transmitting data packets that are addressed to the destination node through the discovered route, any intermediate receiving node caches the route to the destination node, if the node has not cached already. The details of other optimizations can be found in [13].

In the *Route Maintenance* phase, if a transmitting node encounters a fatal error or does not receive acknowledgments of the transmitted packets from the downstream node, it generates a *Route Error* (RERR) packet. It also removes the route(s) that use this failed link from its cache. Furthermore, nodes between this node and the source remove the route(s) with the reported failed link from their caches upon receipt of the RERR packet. When the RERR packet makes its way to the source, a new route discovery process may be initiated.

II.B. GPSR

GPSR is a proactive, geographic routing protocol. In a geographic routing protocol like GPSR, each node can determine its location using a mechanism such as the Global Positioning System (GPS). In GPSR, every packet carries its destination position. A forwarding node determines the next hop of a packet based on the neighbors' positions. It chooses the neighbor node geographically closest to the destination of the packet as the next hop. This forwarding procedure continues until the packet reaches its destination.

Each node periodically broadcasts its position. By monitoring the beacons, a node maintains a forwarding table of the neighbors' positions. To avoid synchronization of beacons, a node jitters each beacon's transmission. In addition, to minimize the cost of sending beacons, a node piggybacks its position to data packets it forwards.

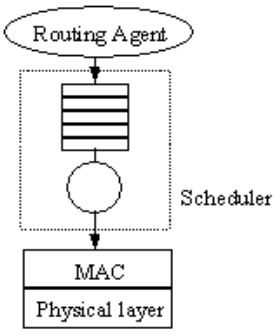


Figure 1: A mobile node. The scheduler is positioned between the routing agent and the MAC layer.

A packet can arrive at a node that does not have a closer neighbor than itself to the destination and is not a neighbor to the destination. When it encounters such a hole, GPSR switches from the *geographic forwarding* mode to the *perimeter forwarding* mode. Perimeter forwarding uses a planar graph to route around the hole. When the position of a forwarding node is closer than that of the node that meets the hole, GPSR switches back to the geographic forwarding mode.

III. Scheduling Algorithms Studied

Scheduling algorithms determine which packet is served next among the packets in the queue(s). The scheduler is positioned between the routing agent and above the MAC layer (Figure 1). All nodes use the same scheduling algorithm. We consider the conventional scheduling (priority scheduling) typically used in mobile ad hoc networks [3, 4] and also propose other applicable scheduling policies to study. All scheduling algorithms studied are non-preemptive.

As a buffer management algorithm, the drop tail policy is used with no-priority scheduling. The drop tail policy drops incoming packets when the buffer is full. For the scheduling algorithms that give high priority to control packets, we use different drop policies for data packets and control packets when the buffer is full. When the incoming packet is a data packet, the data packet is dropped. When the incoming packet is a control packet, we drop the last enqueued data packet, if any exists in the buffer, to make room for the control packet. If all queued packets are control packets, we drop the incoming control packet.

We explain the scheduling algorithms we analyze below.

III.A. Scheduling Algorithms for Analysis of Giving High Priority to Control Traffic

In on-demand routing protocols such as DSR, under frequent topology changes, delivering control packets (routing packets) quickly can be more important than in proactive routing protocols for propagating route discoveries or route changes quickly. To study the effect of timely delivery of control packets in on-demand and proactive routing protocols, we compare a scheduling algorithm that does not distinguish control packets from data packets (Section III.A.1) with a scheduling algorithm that gives high priority to control packets (Section III.A.2).

III.A.1. No-priority Scheduling

No-priority scheduling services both control and data packets in FIFO order. We include this scheduling algorithm to contrast with the effect of giving high priority to control packets.

III.A.2. Priority Scheduling

Priority scheduling gives high priority to control packets. It maintains control packets and data packets in separate queues in FIFO order. Currently, this scheme is used in most comparison studies about mobile ad hoc networks [3, 4].

III.B. Scheduling Algorithms for Analysis of Setting Priorities in Data Traffic

After examining the effects of giving priority to control traffic, we look at the effects of setting priorities in data traffic. We devise different scheduling algorithms by using distance metrics, considering fairness, and applying the multiple roles of nodes as both routers and data sources. All the scheduling algorithms we explain below give higher priority to control packets than to data packets. Their differences are in assigning weight or priority among data queues (Figure 2).

A class of scheduling algorithms (Sections III.B.1 and III.B.2) uses distance metrics to setting priorities in data traffic. It is well understood that in a single node if the task sizes are known, shortest-remaining-processing-time (SRPT) scheduling is the policy that minimizes mean response time [14]. We apply this concept to an ad hoc network with multiple nodes. Although remaining processing time is not known in many cases, in a network we can assume that the remaining processing time of a packet is likely to be proportional to the “distance” (remaining hops or phys-

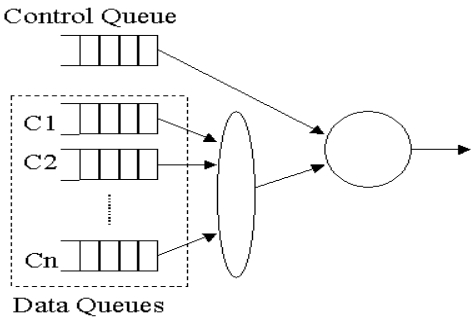


Figure 2: Packet scheduler. The control queue has higher priority than data queues. Among data queues, we experiment with various scheduling algorithms.

ical distance) from a forwarding node to a destination. We study weighted-hop scheduling with DSR and weighted-distance scheduling with GPSR.

Besides the scheduling algorithms using distance metrics, we study round robin scheduling (Section III.B.1) and greedy scheduling (Section III.B.2) to see how fairness or greediness of a node's packet forwarding affect the performance.

III.B.1. Weighted-hop Scheduling

Weighted-hop scheduling gives higher weight to data packets that have fewer remaining hops to traverse. The fewer hops a packet needs to traverse, the more potential it has to reach its destination quickly and the less queuing it incurs in the network. In Figure 2, the data packet scheduler serves packets in weighted round robin fashion. We use a weighted round robin scheduler instead of a static priority scheduler since the weighted scheduler guarantees all service classes at least the configured amount of service chances, thus avoiding starvation. If we consider packet length variations to allocate the correct proportion of bandwidth, we can use weighted fair queuing [15] or deficit round robin [16]. The data queue of the class Ci maintains data packets whose number of remaining hops to traverse is i. When the number of remaining hops of a data packet is greater than n (the number of data queues), the data packet is classified as Cn. For example, if the remaining number of hops of a data packet is 2, it belongs to C2. The data queue of the class Ci receives weight Wi ($1 \leq i \leq n$).

In the DSR protocol, each data packet header carries a complete list of nodes through which the packet should travel. In DSR, we thus obtain the remaining hops to traverse from the packet headers. However, in other routing protocols, including Ad Hoc On-demand Distance Vector Routing (AODV) [2], we would obtain this information from the routing table, which stores the remaining hops to destinations.

III.B.2. Weighted-distance Scheduling

We also consider a scheduling algorithm that uses physical distance with GPSR. Using physical distance may be a unique feature of ad hoc wireless networks. In the GPSR protocol, each data packet carries a destination's position. Nodes that are close in physical distance are likely to be close in the network topology (i.e., a small number of hops from each other). As the remaining physical distance to a destination decreases, the remaining hops to a destination in the network topology are likely to decrease.

The weighted-distance scheduler is also a weighted round robin scheduler. It gives higher weight to data packets that have shorter remaining geographic distances to the destinations. The remaining distance (*RemainingDistance*) is defined as the distance between a chosen next hop node and a destination. Each class Ci ($1 \leq i \leq n$) is determined by the virtual hop:

$$VirtualHop = \lceil \frac{RemainingDistance}{QuantizationDistance} \rceil + 1$$

where *QuantizationDistance* is a distance for mapping the physical distance into the class. For simplicity we choose this uniform quantization method. Better quantization methods might be used for further improvement. When the *VirtualHop* of a data packet is greater than n (number of data queues), the data packet is classified as Cn. For example, if n = 8, *QuantizationDistance* is 250m, and *RemainingDistance* is 350m, the *VirtualHop* = 3 and the packet belongs to C3. The data queue of the class Ci receives weight Wi ($1 \leq i \leq n$). The higher weight is assigned to the lower class.

III.B.3. Round Robin Scheduling

Round robin scheduling maintains per-flow queues. We identify each flow by a source and destination (IP address, port number) pair. In Figure 2, each Ci is equal to a flow. In round robin scheduling, each flow queue is allowed to send one packet at a time in round robin fashion. We evaluate round robin scheduling to see the effect on performance of having an equal service chance among flows.

III.B.4. Greedy Scheduling

In the greedy scheduling scheme, each node sends its own data packets (packets it has generated) before forwarding those of other nodes. The other nodes' data

packets are serviced in FIFO order. In Figure 2, there are two classes (n = 2). The queue of C1 keeps its own data packets and the queue of C2 keeps the other nodes' data packets. C1 has strict priority over C2. We assess whether such greediness adversely affects network performance. Although it is uncommon in wired networks for a node to act as a source and a router concurrently, it is commonplace in mobile ad hoc networks.

III.B.5. Other Scheduling Algorithms We Considered

In addition to these four scheduling policies, we studied scheduling algorithms favoring data packets with long distance metrics. We expected that such scheduling might improve average throughput by quickly delivering packets with greater remaining hops or greater remaining distance. However, when we gave high priority to data packets with long distance metrics, the average throughput and delay were degraded. Compared to packets with fewer remaining hops or shorter remaining distance, packets with greater remaining hops or greater remaining distance are more likely to experience route changes, resulting in many retransmissions in the MAC layer. Therefore, data packets with long distance metrics require longer service time overall. We do not consider these algorithms further.

IV. Methodology

In this section we describe our simulation environment and performance metrics.

IV.A. Simulation Environment

For our simulations we used ns-2 [12], a packet-level discrete event simulator. Ns-2 includes the simulation model for mobile ad hoc networks developed by the CMU Monarch project. The model includes a physical layer, an 802.11 MAC layer, and a data link layer [3]. The wireless channel capacity is 2Mb/sec. As mentioned earlier, we performed our study with DSR and GPSR as the routing protocols.

The default overall buffer size of the scheduler of each node is 64 packets. The buffer is shared by multiple queues when the scheduler maintains multiple queues.

The DSR protocol implementation in ns-2 also maintains a send buffer of 64 packets used during route discovery. The maximum waiting time in the send buffer during route discovery is 30 seconds. If a packet remains in the send buffer for over 30 seconds, the packet is dropped.

In GPSR protocol simulations, we set a beacon interval to one second. The beacons are sent proactively in the GPSR protocol. We assume that each node knows the current location of the destination because the original GPSR simulation code does not include a location database for locating destination. Each source annotates its packets with the current position of the destination. Hence, our GPSR simulation results might be better than the GPSR simulation results with a location service.

We use 50 mobile nodes in a rectangular grid of dimensions 1500m x 300m. We ran each simulation for 900 seconds. We use the *random waypoint model* because it is the most widely used mobility model in previous studies [3]. In this model, a node decides to move to a random location within the grid. When it reaches that location, it pauses for a fixed amount of time, possibly zero seconds, and then it moves to another random location. The maximum allowed speed for a node is 20 meters per second.

We use a constant bit rate (CBR) source as the data source for each node. Each source node transmit packets at a certain rate, with a packet size of 512 bytes. We choose source and destination nodes randomly among all nodes. The communication patterns are peer-to-peer, and connections were initiated at random times between 0 and 180 seconds.

We vary the traffic load and the degree of mobility in the simulations. We vary traffic load by changing the number of sources or the packet sending rate. We control the degree of mobility through the pause time. We use pause times of 0, 30, 60, 120, 300, 600, and 900 seconds. A pause time of 0 seconds implies constant movement, whereas 900 seconds implies no movement at all since our simulations run for 900 seconds. A movement scenario arranges the movement and the position of the nodes according to the random waypoint model. Because the simulation results depend on the movement scenarios, we averaged simulation results over four different movement scenarios for each data point.

IV.B. Performance Metrics

We use the following performance metrics to evaluate the effect of each scheduling algorithm:

Average delay: This is the average overall delay for a packet to travel from a source node to a destination node. This includes the route discovery time, the queuing delay at a node, the retransmission delay at the MAC layer, and the propagation and transfer time

Table 1: Average Queue Length Across All The Nodes (DSR)

Number of Sources	Pause Time(s)	Average Queue Length (packets)		
		Min	Median	Max
10	0	0.52	0.55	0.65
	900	0.50	0.50	4.21
20	0	0.56	0.97	6.35
	900	0.50	0.51	0.58
30	0	1.42	9.00	18.23
	900	0.50	0.69	52.61
40	0	1.90	14.00	34.26
	900	0.51	0.94	58.39

in the wireless channel.

Average througput: This is the average number of data packets received by the destination node per second.

We also measured *routing overhead*, defined as the average ratio of routing-related transmissions to data transmissions. The transmission in each hop is counted when a node sends or forwards a packet. ARP packet transmissions are not included in this metric. Since the routing overhead is not affected considerably the choice of scheduling algorithms (the maximum difference of the routing overhead among scheduling algorithms is less than 0.05), we do not present it here. Some of results can be found in [17].

V. Queuing Dynamics

Before evaluating scheduling algorithms, we analyze the queuing dynamics of the nodes in the network. Analyzing the queuing dynamics under different traffic load and mobility conditions helps us understand when and why different scheduling algorithms affect network performance. We use the conventional scheduling algorithm (Section III.A.2) for the experiments in this section. We first examine the average queue lengths for all nodes to find the queue distribution throughout the network, and then we examine how queue length changes in a congested node.

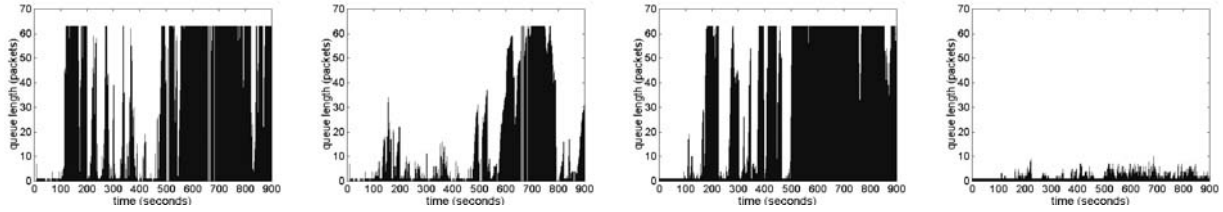
Table 1 shows the minimum, median and maximum of average queue lengths across all the nodes. The packet sending rate was 4 packets/second. Throughout the network, there is significant queuing in the nodes when traffic load is high. When the number of sources is 10 or 20, the nodes have very little queuing; however, queuing becomes more evident as the number of sources increases to 30 or 40. With high mobility queuing in the nodes occurs evenly throughout the network; we infer this from the fact that the maximum is much smaller and the median is bigger than with low mobility. With low mobility, the queuing seems to occur in a small region of the network, because the median value is small but the maximum value is large. We expect that using different scheduling algorithms in such congested nodes will affect the performance. In the paper we present only the results of DSR, because these statistics of average queue lengths were similar for GPSR. Results for GPSR can be found in [17].

We look at the queue traces of the most heavily loaded nodes (i.e., nodes with the highest average queue length) in a movement scenario to examine how the queue lengths change and what sorts of packets the queues contain with DSR and GPSR. In the queue traces, the number of data packets is the number of total packets minus the number of control packets.

The shape of the queue traces differs greatly depending on mobility. When there is constant movement, there is a large and frequent variance in the queue length, from 0 to 64, the maximum queue length (Figure 3). Due to constant movement, however the routes are not stable, and the queue length generally remains short compared to the stationary movement pattern. When there is no movement of nodes, we see that most queue lengths are 64 packets (Figure 4). This is because the routes do not change, and the same routes get used repeatedly.

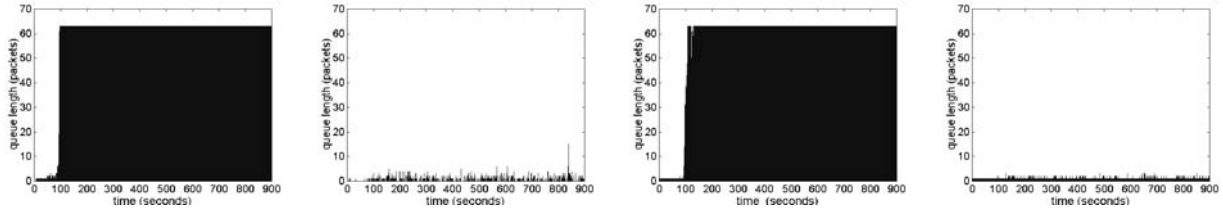
In addition, there is a big difference in the type of packets in the queue depending on mobility. When nodes are static, most of the packets in the queue are data packets with both DSR and GPSR. However, when nodes are highly mobile, the packet composition of the queue is dependent on the routing protocol used. With DSR, the queue is often composed of more routing packets than data packets (Figures 3(a) and 3(b)). Most of the routing packets are Route Reply (RREP) packets. This phenomenon is due to the aggressive use of route caches in the DSR protocol. (If many nodes have requested routes in route caches, RREP packets are generated from multiple nodes.) As a result, within an interval, the reply flood fills up the queue and few data packets are serviced. In contrast to DSR, with GPSR, most of the packets in the queue are data packets (Figures 3(c) and 3(d)). Since GPSR is proactive, it does not incur the flood of routing packets when routes change.

From the analysis of queue traces, we expect that with high mobility giving high priority to control traffic should affect the performance of DSR and GPSR differently. In addition, using different prioritization



(a) Total packets (b) Control packets (c) Total packets (d) Control packets

Figure 3: Queue Traces of the Most Heavily Loaded Node (40 sources, 0 seconds pause time). (a) and (b) for DSR. (c) and (d) for GPSR.



(a) Total packets (b) Control packets (c) Total packets (d) Control packets

Figure 4: Queue Traces of the Most Heavily Loaded Node (40 sources, 900 seconds pause time). (a) and (b) for DSR. (c) and (d) for GPSR.

schemes among data packets will lead to differences in the performance with low mobility and even with high mobility in case of GPSR.

VI. Effects of Giving High Priority to Control Traffic

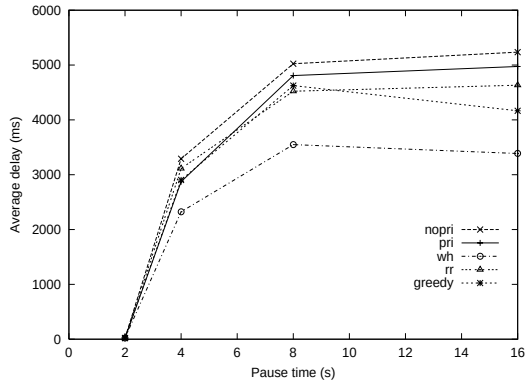
In this section we evaluate the effects of giving high priority to control traffic on average delay and average throughput under various mobility and traffic load conditions. In the graphs presented in Sections VI and VII, we use the following abbreviations: *nopri* for no-priority scheduling, *pri* for priority scheduling, *wh* for weighted-hop scheduling, *wd* for weighted-distance scheduling, *rr* for round robin scheduling, and *greedy* for greedy scheduling. Although we simulated many pause times, in most cases we list results only for the extremes (0 and 900 seconds) due to space constraints. In-between pause times show appropriate in-between results.

VI.A. DSR

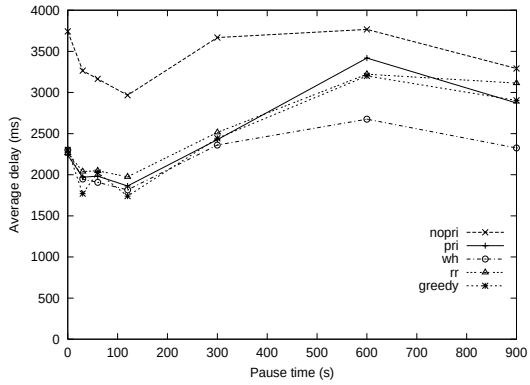
The average delays of the no-priority and priority scheduling algorithms are slightly different when traffic load is high (Figure 5(a)). This prioritization has bigger impact on delay reduction as mobility increases (Figure 5(b)). When the number of sources is 40 and the nodes are highly mobile, priority scheduling decreases the average delay by 40% compared to no-priority scheduling.

To understand how the average delay is reduced by giving high priority to control traffic in detail, we look at the *cumulative distribution functions* (cdf's) of the packet delays. When nodes move without pause, the delay distribution shifts to the left in priority scheduling compared to no-priority scheduling (Figure 6(a)). This is because giving high priority to control packets helps notify the source of the route discovery or route error quickly. With low mobility the cdf's of the two scheduling algorithms are almost same (Figure 6(b)). In the case of low mobility, since most of the packets in a queue are data packets, giving high priority to control packets only improves delay slightly.

Giving high priority to control packets does not improve the average throughput (Figure 9). With low mobility, there is little effect of this prioritization, since control packets are not sent frequently. With high mobility, since the flood of control packets reduces the chance of data packets being serviced, the average throughput does not increase.

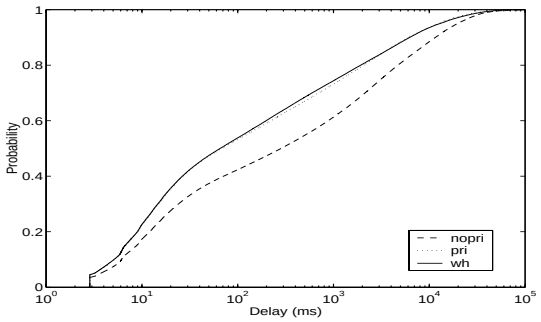


(a) Varying packet sending rate (40 sources, 900 seconds pause time)

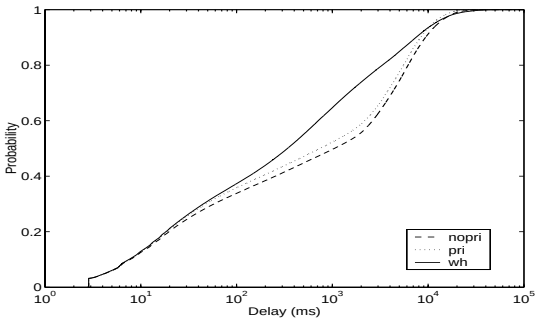


(b) Varying mobility (40 sources, 4 packets/second packet sending rate)

Figure 5: Average Delay (DSR)



(a) 0 seconds pause time



(b) 900 seconds pause time

Figure 6: Cumulative Distribution Functions of Packet Delays (DSR, 40 sources, 4 packets/second packet sending rate)

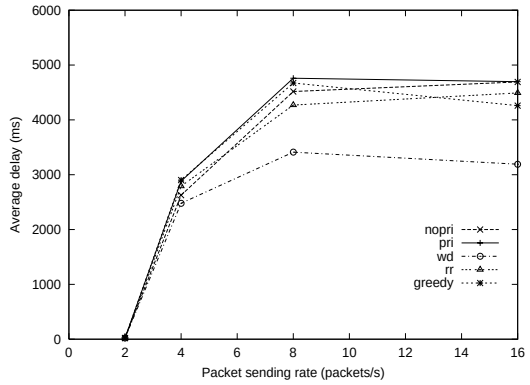
VI.B. GPSR

In the simulations with GPSR, no-priority scheduling shows smaller average delay than priority scheduling in contrast to the simulation results of DSR. When the number of sources is 40 and nodes are highly mobile, the priority scheduling algorithm increases the average delay by 71% compared to the no-priority scheduling algorithm (Figure 7). Since GPSR proactively updates topology changes, there is little benefit from giving high priority to control packets for quickly updating route changes. Instead, as control packets consume the chances of servicing data packets, they increase the delay of data packets. As shown in Figure 8, with high mobility the cdf for the no-priority scheduling is positioned to the left of the cdf for the priority scheduling. With low mobility there is little difference between no-priority scheduling and priority scheduling.

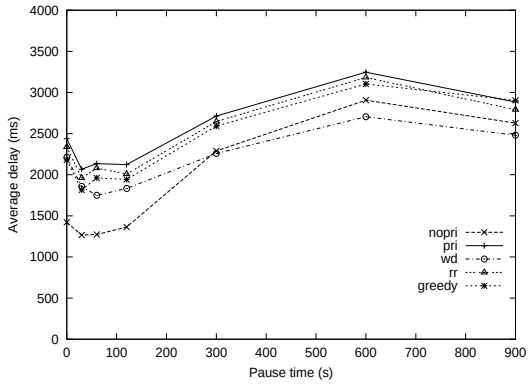
There is slight difference in the average throughput between the no-priority and priority scheduling algorithms (Figure 10). The network topology is constantly updated in the GPSR protocol so that the no-priority scheduler can deliver data packets as successfully as the priority scheduler can.

VII. Effects of Setting Priorities in Data Traffic

In this section we evaluate the effects of setting priorities among data packets under various mobility or traffic load conditions, and various packet buffer sizes. Our goal is to find scheduling algorithms that improve performance most compared to the conventional ones. Since we compare the effects of different scheduling algorithms that choose among data packets, we need to separate out the effects of control packets. Therefore, all the scheduling algorithms presented here con-

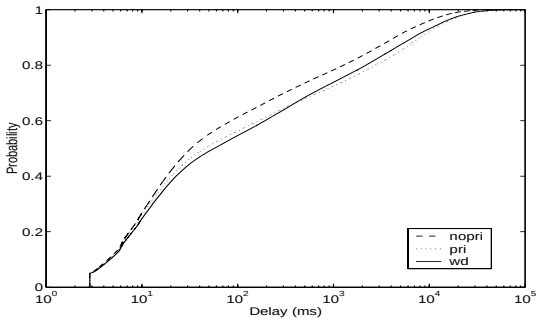


(a) Varying packet sending rate (40 sources, 900 seconds pause time)

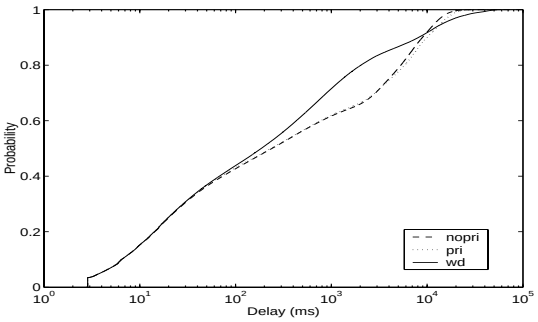


(b) Varying mobility (40 sources, 4 packets/second packet sending rate)

Figure 7: Average Delay (GPSR)



(a) 0 seconds pause time



(b) 900 seconds pause time

Figure 8: Cumulative Distribution Functions of Packet Delays (GPSR, 40 sources, 4 packets/second packet sending rate)

sistently give higher priority to control packets than to data packets.

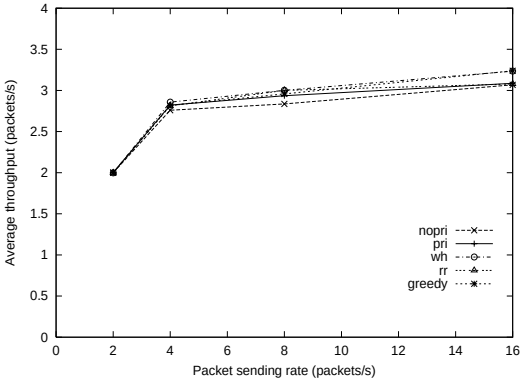
The configurations of the weighted-hop and weighted-distance scheduling algorithms are presented in Tables 2 and 3. The scheduling algorithms have eight classes. The *QuantizationDistance* of the weighted-distance scheduling is 175 meters, which is 70% of the radio distance. We simulated a movement scenario with varying weight assignments in a limited weight value space. The weighted-hop and weighted-distance scheduling algorithms show better performance than the priority scheduling in 217 out of 230 randomly selected weight assignments. We choose one assignment for presentation that shows good performance improvement. In these scheduling algorithms, the weight value represents the number of packets served in a service round.

Table 2: Weighted-hop scheduling configuration

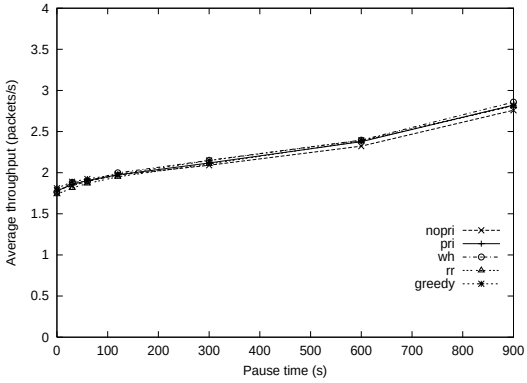
Class	Weight	Remaining-hop(h)
C1	5	h = 1
C2	5	h = 2
C3	3	h = 3
C4	1	h = 4
C5	1	h = 5
C6	1	h = 6
C7	1	h = 7
C8	1	$h \geq 8$

VII.A. Effects of Traffic Load and Mobility

Figures 5 and 7 show the average delay of the scheduling algorithms studied with DSR and GPSR, respectively. The average delay of different scheduling algorithms differs significantly as the packet sending rate

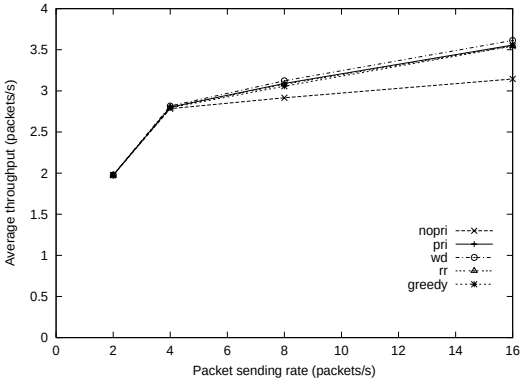


(a) Varying packet sending rate (40 sources, 900 seconds pause time)

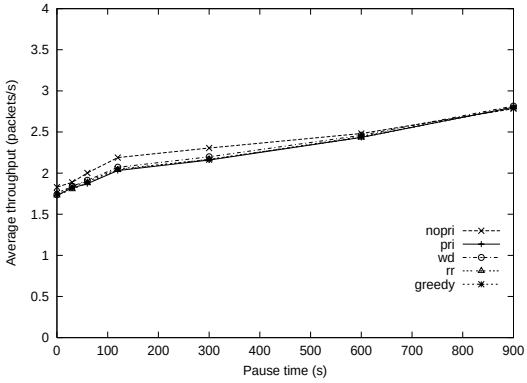


(b) Varying mobility (40 sources, 4 packets/second packet sending rate)

Figure 9: Average Throughput (DSR)



(a) Varying packet sending rate (40 sources, 900 seconds pause time)



(b) Varying mobility (40 sources, 4 packets/second packet sending rate)

Figure 10: Average Throughput (GPSR)

Table 3: Weighted-distance scheduling configuration

Class	Weight	Remaining-distance(d)
C1	5	d = 0
C2	5	$0 < d \leq 175$
C3	3	$175 < d \leq 350$
C4	1	$350 < d \leq 525$
C5	1	$525 < d \leq 700$
C6	1	$700 < d \leq 875$
C7	1	$875 < d \leq 1050$
C8	1	$1050 < d$

increases. With a 2 packets/second sending rate, there is little difference in average delay between the two scheduling algorithms because there are few packets in the queue(s). However, with 4, 8, and 16 packets/second sending rates, there is significant difference. In addition, setting priorities among data packets has a bigger impact as mobility decreases. With low mobility, the reductions in delay with the weighted-hop and weighted-distance scheduling algorithms are most significant. When the packet sending rate is 16 packets/second and mobility is low, the weighted-hop scheduling decreases the average delay by 32% compared to priority scheduling with DSR (Figure 5). The reduction in delay is also big with the weighted-distance scheduling with GPSR. It reduces the average delay by 32% compared to priority scheduling (Figure 7).

With moderate mobility, the reduction in the average delay is still significant. The weighted-hop algorithm with DSR reduces delay by 22% compared to priority scheduling. The weighted-distance scheduling algorithm with GPSR reduces delay by 17% com-

pared to priority scheduling. By giving higher weight to packets with fewer remaining hops or shorter distances, the delay of such packets decreases considerably, while the delay of packets with larger remaining hops or longer distance increases relatively little.

Interestingly, when nodes are highly mobile, the reduction in the delay is negligible in the simulation results with DSR. As shown in Section 5, with high mobility most of the packets in the queue are control packets, so setting priorities in data traffic does not much change the servicing order of the packets in the queue. However, the reduction in the delay with the weighted-distance scheduling algorithm with GPSR is still noticeable because most of the packets in the queue are data packets even with high mobility.

To understand how the average delay is reduced by some of the scheduling algorithms, we look at the cdf's for DSR and GPSR (Figures 6 and 8). With low mobility, the delay distributions shift to the left in the weighted-hop scheduling algorithm compared to the priority scheduling algorithm (Figure 6(b)). In the middle range of the distribution, where common cases are located, we observe the largest reduction in delay. The trend in graphs of Figure 8(b) is similar to that in graphs of Figure 6(b). With high mobility, there is little difference between priority scheduling and weighted-hop scheduling with DSR (Figure 6(a)). In addition, the cdf of the weighted-distance scheduling is similar to that of the priority scheduling. It is interesting to note that the starting slope of the cdf for high mobility is higher than that of the cdf for low mobility. This is because with high mobility, packets with shorter distance metrics are favored.

Greedy scheduling and round robin scheduling show little difference in performance compared to priority scheduling. In the case of greedy scheduling, if we look at the performance of individual flows, some flows are severely penalized, although the overall performance does not change. In the case of round robin scheduling, we believe that the small difference in performance may be due to the source type being CBR. With a bursty source such as TCP, the effect of round robin scheduling might be larger.

Figures 9 and 10 show the average throughput for the studied scheduling algorithms with DSR and GPSR, respectively. The weighted-hop scheduling algorithm with DSR and weighted-distance scheduling algorithm with GPSR consistently display higher average throughput compared to the other scheduling algorithms, but the difference is not significant.

Table 4: Average Delay(DSR) (ms)

Pause time	Buffer size	*pri*	*wh*
0 seconds	48 packets	1589.34	1455.16
	64 packets	2236.78	2299.44
	80 packets	3112.31	2967.82
900 seconds	48 packets	2269.35	1888.79
	64 packets	2872.04	2324.85
	80 packets	3806.46	3063.79

Table 5: Average Throughput(DSR) (packets/second)

Pause time	Buffer size	*pri*	*wh*
0 seconds	48 packets	1.785	1.790
	64 packets	1.778	1.766
	80 packets	1.762	1.789
900 seconds	48 packets	2.842	2.851
	64 packets	2.822	2.858
	80 packets	2.806	2.862

VII.B. Effects of Buffer Size

We also examine the effects of scheduler buffer size on performance under different scheduling algorithms. We show the results of priority scheduling, weighted-hop scheduling, and weighted-distance scheduling algorithms for the two extremes of mobility (0 and 900 seconds pause times.)

Tables 4 and 5 show the average delay and the average throughput respectively with buffer sizes of 48, 64, and 80 packets when the number of sources is 40, packet sending rate is 4 packets/sec, and the routing protocol is DSR. In Table 4 we see that for any given pause time, as the buffer size increases, the average delay increases for both scheduling algorithms due to increased queuing delay in forwarding nodes. However, this increase is less for weighted-hop scheduling, suggesting that this scheduling algorithm is more resilient than priority scheduling to buffer size changes as far as the average delay is concerned. In Table 5 we see that for both scheduling algorithms, regardless of the degree of mobility, the average throughput does not change much as the buffer size varies. For GPSR, the results with the priority and weighted-distance scheduling algorithms show similar trends (Tables 6 and 7).

VIII. Related Work

To the best of our knowledge, no work has been published previously in the area of queuing dynamics or packet scheduling algorithms in the queues of the nodes in mobile ad hoc networks. There are studies of

Table 6: Average Delay(GPSR) (ms)

Pause time	Buffer size	*pri*	*wd*
0 seconds	48 packets	1803.65	1701.23
	64 packets	2437.70	2210.02
	80 packets	3015.11	2884.60
900 seconds	48 packets	2163.69	1882.13
	64 packets	2883.42	2479.83
	80 packets	3617.88	3079.14

Table 7: Average Throughput(GPSR) (packets/second)

Pause time	Buffer size	*pri*	*wd*
0 seconds	48 packets	1.780	1.792
	64 packets	1.731	1.736
	80 packets	1.696	1.710
900 seconds	48 packets	2.802	2.806
	64 packets	2.803	2.817
	80 packets	2.800	2.812

scheduling algorithms in base stations of wireless cellular networks and medium access controls in static multi-hop wireless networks. However, they do not consider mobility, the effects of control traffic on the performance, the use of the distance metrics to setting priorities in data traffic, or the effects of the greediness of nodes in forwarding packets.

In wireless networks, much effort in scheduling research has focused on fairness issues. Scheduling algorithms to support fairness in the presence of link errors have been studied in wireless cellular networks [18, 19]. Medium access scheduling to achieve fairness in static multi-hop wireless networks has also been studied [8, 9]. These studies only consider wireless channel contention. Nantagopal et al. [8] focus on achieving fairness requirements with proper MAC protocol designs. Luo, Lu, and Bharghavan study minimum fairness of flows with maximum spatial reuse with a core-based conflict-free shared multicast tree [9]. Maintaining the tree seems to be a difficult task in a highly mobile environment.

In static multi-hop wireless networks, medium access scheduling algorithms to support bounded delay and guaranteed throughput have also been studied. In a study by Kanodia et al. [10], the priority value of the head-of-line packet is piggybacked onto handshake and data packets, and the next packet to send is determined by this distributed priority information. The authors also propose a scheme to access the medium in a reference order [11]. In comparison, our scheduling algorithms use local information or information that can be acquired from the routing protocol; thus, there is no stale information distributed with high mobility and no additional overhead to exchange priority or ordering information.

There is an interesting similarity between the scheduling algorithms that favor packets with short distance metrics in this paper and the SRPT connection scheduling algorithm in [20, 21]. In these papers, Crovella et al. show that the SRPT connection scheduling algorithm, if employed in web servers, can improve the mean response time for serving static pages over that in web servers employing a size-independent (or no) connection scheduling policy. The size of the connection is the file size requested. The improvement in the mean response time comes at the expense of fairness to long connections; however, the authors show that there is only a marginal penalty to the long connections.

IX. Future Work

There are several areas of future work we would like to explore.

We would like to study more routing protocols and different data sources. Studying more routing protocols, including AODV, will help us to see the broader effect of the scheduling algorithms. We use CBR sources in this paper, but it would be interesting to analyze how the scheduling algorithms affect performance with bursty TCP sources.

We experiment with the random waypoint mobility model in our simulations. In this model, the traffic load is evenly distributed across the network due to randomized source movements. If we use a different mobility model such as grouped mobility, the traffic load would not be distributed as evenly and the scheduling algorithms might perform better even with high mobility. We would like to investigate a larger set of mobility patterns such as those included in [5], since performance results can be very sensitive to the mobility pattern.

With GPSR, it would be interesting to experiment with the integration of the no-priority and weighted-distance scheduling algorithms to see potential performance improvement. One way to integrate these is to maintain the time of the control packets (e.g., 5th packet in the queue) when they arrive at the queue and serve them at the expected service time in no-priority scheduling (e.g., 5th departure from the queue). Meanwhile, data packets are serviced with the weighted-distance scheduling policy.

X. Conclusion

In this paper, we analyze queuing dynamics in mobile ad hoc networks and evaluate the effect of different scheduling algorithms on network performance with DSR and GPSR as the underlying routing protocols.

Queuing dynamics with different degrees of mobility and routing protocols show that the composition of packets in the queue determines the effects of giving priority to control packets or setting priorities among data packets, especially for the average delay. During low mobility, the average delay is dominated by network congestion due to data traffic. During high mobility, it is dominated by route changes in the simulation results.

The effects of giving priority to control packets are different depending on whether the routing protocol is DSR or GPSR. With DSR, giving control packets higher priority reduces the average delay, but it rarely affects the average throughput. As mobility increases, so does the reduction in packet delay. With GPSR, priority scheduling increases the average delay compared to no-priority scheduling. Because GPSR is proactive, there is little advantage in sending control packets quickly to update topological changes.

Our scheduling algorithms that give higher weight to data packets with smaller numbers of hops or shorter geographic distances to their destinations reduce average delay significantly without any additional control packet exchange. The weighted-hop scheduling algorithm with DSR and the weighted-distance scheduling algorithm with GPSR show considerably smaller delay than the other scheduling algorithms. The reduction in the average delay decreases as the mobility of nodes increases.

We also investigate the effects of buffer size on performance and show that larger buffer size increases the average delay. Our results indicate that this increase is noticeably less for weighted-hop and weighted-distance scheduling than for simple priority scheduling. Buffer size does not affect the average throughput, however.

From the simulation results, we find that giving high priority to control traffic should be carefully evaluated for use depending on the routing protocol. We show that on-demand routing protocols are likely to benefit from this arrangement, but proactive routing protocols might not. With scheduling algorithms using short distance metrics, data packets can be delivered much faster in a congested network, without additional control packet exchange for the algorithms. Furthermore, the implementation of these algorithms is simple. Thus, they are easily deployable to improve performance in resource-constrained ad hoc networks.

XI. Acknowledgements

We greatly thank Devendra R. Jaisinghani for his contribution to the initial work of the study. We also thank T. J. Giuli, Armando Fox, and Chan Jean Lee for valuable discussions regarding this work and their comments on drafts of this paper. This work has been supported by a grant from the Stanford Networking Research Center and by MURI award number F49620-00-1-0330.

References

[1] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. IETF Internet draft, *Mobile Ad-hoc Network Working Group*, IETF, February 2002 (work in progress).

[2] Charles E. Perkins, Elizabeth M. Royer, and Samir Das. Ad Hoc On Demand Distance Vector (AODV) Routing. IETF Internet draft, *Mobile Ad-hoc Network Working Group*, IETF, January 2002 (work in progress).

[3] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of ACM/IEEE MOBICOM*, Dallas, TX, October 1998.

[4] Samir R. Das, Charles E. Perkins, and Elizabeth M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proceedings of the IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[5] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of ACM MOBICOM*, Seattle, WA, August 1999.

[6] V.D. Park and M.S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of IEEE INFOCOM*, Kobe, Japan, April 1997.

[7] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In

Proceedings of ACM MOBICOM, Boston, MA, August 2000.

[8] T. Nantagopal, T. Kim, X. Gao, and V. Bharghavan. Achieving MAC layer fairness in wireless packet networks. In *Proceedings of ACM MOBICOM*, Boston, MA, August 2000.

[9] H. Luo, S. Lu, and V. Bharghavan. A New Model for Packet Scheduling in Multihop Wireless Netowrks. In *Proceedings of ACM MOBICOM*, Boston, MA, August 2000.

[10] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. Distributed Multi-Hop Scheduling and Medium Access with Delay and Throughput Constaints. In *Proceedings of ACM MOBICOM*, Rome, Italy, July 2001.

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

[12] K. Fall and K.Varadhan, editors. ns notes and documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, July 1999.

[13] David A. Maltz, Josh Broch, Jorjeta Jetcheva, and David B. Johnson. The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications* special issue on mobile and wireless networks. August 1999.

[14] D. Karger, C. Stein, and J. Wein. Scheduling algorithms. In *CRC Handbook of Computer Science*. 1997.

[15] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair-queuing Algorithm. In *Proceedings of ACM SIGCOMM*, Austin, TX, September 1989.

[16] M. Shreedhar and G. Varghese. Efficient Fair Queuing Using Deficit Round Robin. In *Proceedings of ACM SIGCOMM*, Cambridge, MA, October 1995.

[17] Byung-Gon Chun. Evaluation of Scheduling Algorithms in Mobile Ad Hoc Networks. MS Thesis, Stanford University, May 2002.

[18] S. Lu, V. Bharghavan and R. Srikant. Fair Scheduling in Wireless Packet Networks. In *Proceedings of ACM SIGCOMM*, Cannes, France. September 1997.

[19] S. Lu, T. Nandagopal, and V. Bharghavan. A Wireless Fair Service Algorithm for Packet Cellular Networks. In *Proceedings of ACM/IEEE MOBICOM*, Dallas, TX, October 1998.

[20] Mark Crovella, Bob Frangioso, and Mor Harchol-Balter. Connection Scheduling in Web Servers. In *USENIX Symposium on Internet Technologies and Systems (USITS)*, Boulder, Colorado, October 1999.

[21] Nikhil Bansal and Mor Harchol-Balter. Analysis of SRPT Scheduling: Investigating Unfairness. In *Proceedings of ACM Sigmetrics 2001 Conference on Measurement and Modeling of Computer Systems*, Cambridge, MA, June 2001.

Biographies

Byung-Gon Chun received an MS with Distinction in Research in Computer Science at Stanford University. He will soon join the PhD program in Computer Science at the University of California at Berkeley. He also received BS and MS degrees in the Electronic Engineering Department at Seoul National University, Korea, where he worked as a research assistant in the telecommunications and signal processing laboratory. In addition, before coming to Stanford, he worked as a software engineer at Sundo Automatic Technology Institute, Korea. His interests include networking, distributed systems, mobile/ubiquitous computing, and network security.

Mary Baker is an assistant professor in the Departments of Computer Science and Electrical Engineering at Stanford University. Her interests include mobile systems, distributed systems and networks. Baker received a BA degree in mathematics in 1984 from the University of California at Berkeley, and a PhD in computer science in 1994 also from U. C. Berkeley. Baker is a recipient of an Alfred P. Sloan Research Fellowship, a Terman Fellowship, an NSF Faculty Career Development Award, and an Okawa Foundation grant. She has participated on the technical advisory boards of several companies including DoCoMo USA Labs.