Distributed Coordination and QoS-Aware Fair Queueing in Wireless Ad Hoc Networks*

Muhammad Mahbub Alam, Md. Mamun-or-Rashid, and Choong Seon Hong**

Department of Computer Engineering, Kyung Hee University,
1 Seocheon, Giheung, Yongin, Gyeonggi, Korea, 449-701
{mahbub, mamun}@networking.khu.ac.kr,
cshong@khu.ac.kr

Abstract. Shared channel, multihop wireless ad hoc network has some unique characteristics that make the fair scheduling in such environment challenging. In this paper we propose a new QoS-aware fair queueing model for ad hoc networks. Our proposed algorithm ensures a distributed coordination of fair channel access while maximize the throughput using spatial reuse of bandwidth. We consider the presence of both guaranteed and best effort flows. The goal is to satisfy the minimum bandwidth requirement of guaranteed flows and provide a fair share of residual bandwidth to all flows. We propose a flow weight calculation scheme to both guaranteed and best-effort flows and a distributed, localized mechanism to implement the time-stamp based ad hoc fair queueing model.

1 Introduction

A wireless ad hoc network consists of a group of mobile nodes without the support of any infrastructure. Such a network is expected to support advanced applications such as communications in emergency disaster management, video conferencing in a workshop or seminar, communications in a battlefield. This class of mission-critical applications demands a certain level of quality of services (QoS) for proper operations. Also due to the distributed nature of these networks providing a fair access to multiple contending nodes is an important design issue.

Fairness is an important criterion of resource sharing in the best effort Internet, especially when there is competition for the share among the nodes due to unsatisfied demands. In Fair scheduling each flow f is allowed to share a certain percentage of link capacity based on its flow weight indicated as w_f. Let $W_f\ (t1,\ t2)$ and $W_g\ (t1,\ t2)$ denote the aggregate resource received by flows f and g respectively in time interval $[t1,\ t2]$ and w_f and w_g are the flow weights of the flows f and g respectively . The allocation is ideally fair if it satisfies (1).

$$\left| \frac{W_f(t_1, t_2)}{w_f} - \frac{W_g(t_1, t_2)}{w_g} \right| = 0 \tag{1}$$

* This work was supported by MIC and ITRC Project.
** Corresponding author.

M. Gavrilova et al. (Eds.): ICCSA 2006, LNCS 3981, pp. 651 – 660, 2006.

Adapting fair queueing to an ad hoc network is challenging because of the unique issues in such a network. These issues include spatial contention among transmitting flows in a spatial locality, spatial reuse through concurrent flow transmissions in a partially connected network, location-dependent channel error, the distributed nature of packet scheduling, and user mobility.

Providing QoS in Wireless ad hoc networks is a new area of research. Existing work focuses mainly on QoS routing which finds a path to meet the desired service requirements of a flow. In this paper we consider a mix of guaranteed and best effort flows and investigated fair queueing with QoS support for the network. The goal is to guarantee the minimum bandwidth requirements of guaranteed flows and to ensure a fair share of residual bandwidth to all flows.

This paper is organized as follows. Section 2 describes related works. In Section 3 we explain the design issues of fair queueing in ad hoc networks. Section 4 describes the proposed mechanism and followed by the details of the implementation of the proposed mechanism in section 5. Section 6 presents the simulation and results. We conclude in section 7 by conclusion and future works.

2 Related Works

Fair queueing has been a popular paradigm for providing fairness, minimum throughput assurance and guaranteed delay in wired network [1], and in packet cellular networks [2] – [4]. Recently some papers have been proposed to incorporated fair queueing in shared channel, multihop wireless networks [5] – [7]. Also, providing QoS in wireless ad hoc networks is a new area of research. Some of the research works also incorporated both QoS and fair queueing in ad hoc networks. Both QoS guarantee and fair queueing in ad hoc networks have been proposed in [8] and [9].

Existing works for fair scheduling in ad hoc networks can be classified into two groups, timestamp-based [5, 6, 8] and credit-based [7, 9]. Timestamp-based protocols convert a node graph into flow graph and for each newly arrived packet two timestamps are assigned, namely *start tag* and *finish tag*. The start tag is set either to the system time at which the packet arrives, or to the finish tag of its previous packet, depending on which value is larger. The finish tag is set to the predicted finishing time, which is equal to the start tag plus the estimated packet transmission time. Either timestamp can serve as the service tag. A back-off value is set based on the service tag and it determines when the packet will be sent. The back-off timer is decremented by one at each time slot until it reaches zero. If the node with a zero timer finds that the channel is free, the packet is transmitted. Nodes with zero timers do not coordinate before transmission and thus collision may occur.

Credit-based protocol assumes the network is divided into clusters and for each cluster there is a cluster head. Each flow simply maintains a counter to record the transmission credit, instead of using two tags as in timestamp-based mechanisms. The basic scheduling concept is “the less excess in usage value, the higher the transmission priority.” The clustering approach is used to implement spatial channel reuse.

A time-stamp based protocol has been extended to provide QoS guarantee with fair scheduling in [8] while [9] provides QoS guarantee with fair scheduling in a credit-based protocol. None of the time-stamped based protocols mentioned how to assign weight to flows. A flow weight calculation scheme is proposed in [9] and used (2).

$$w_f = Resv_f + \frac{1 - \sum_{i \in B} Resv_i}{Num} \tag{2}$$

where $Resv_f$ is the minimum bandwidth requirement of flow *f*, *Num* is the total number of flows passing through node N, and B is the set of backlogged flows. If *f* is best-effort flow, its $Resv_f$ value is zero. If flow *f* is a guaranteed flow, its $Resv_f$ is between zero and one. But this assignment is slightly inconsistent, since the value of $Resv_f$ represents the minimum required bandwidth, so a high value and needs to be normalized. Also, this assignment may allocate more bandwidth to guaranteed flows than required if number of QoS-aware flows is less or the network is lightly loaded.

3 Design Issues in QoS Supported Fair Queueing

This section identifies issues unique to fair queueing in ad hoc wireless networks.

1) Distributed Nature of Ad Hoc Fair Queueing: In wired network a switch makes scheduling decision and in packet cellular network the base station does this. But in an ad hoc network contending flows may originate from different sending nodes and no single logical entity for scheduling of these flows is available. The flow information is distributed among these sending nodes, and nodes do not have direct access to flow information at other nodes. Therefore, the ad hoc network fair queueing is distributed.

2) Defining Fairness for spatially contending Flows: In wired and packet cellular network packet transmission takes place locally, so there is no transmission constrain among neighboring links and fairness is a local property. But in shared-medium ad hoc networks spatial collisions introduce spatial domain channel contention. All the neighbors of the source and destination of a flow have to defer transmission. Therefore, fairness model cannot be defined with respect to local flows in a node only.

3) Conflicts between Fairness and Maximal Channel Reuse: The local broadcast nature of multihop wireless networks allows multiple flows to continue simultaneously, if they do not conflict with each other. This makes the goal conflicting; maximizing spatial reuse may provide more chance to certain nodes to transmit and may have a negative impact on fairness.

4) Providing QoS and State Maintenance: Providing QoS to certain flow requires availability and reservation of resources for that flow. Since, the single wireless medium is shared by the contending nodes, a node can ensure the availability of certain resources if it has the flow information of all the contending nodes and if the information is updated regularly. Also if there is break of route the resource of the flow should be released as early as possible so that the resource can be allocated to a new flow.

4 Distributed Coordination Fair Queueing Model

We now describe a new approach to QoS-aware distributed fair queueing model in ad hoc networks. This model is fully distributed, localized and local scheduler at each node has a certain level of coordination with its neighboring nodes. And this does not require any global information propagation and global computation. The mechanism is as follows:

1) Assumptions: In this paper we assume that errors are caused only by collisions. We consider packet-switched multi-hop wireless networks, but do not consider host mobility as in other existing works [5-9].

2) Maintaining Flow Information within One-hop Neighborhood: Each node maintains two flow tables for the proper operation of fair scheduling. One table is to keep the flow information of two-hop neighbors' flow information, say *flow_table*. This table is sorted according to the service tag of the flows. The fields of the table are *node_id*, *flow_id* and *service_tag*. Another table, *local_table*, contains the flow information where this node is either the sender or the receiver of the flow. The fields of the table are *flow_id*, *s_service_tag*, *r_service_tag, service_tag* (i.e, service tag assigned by source and receiver of the flow) and a *flag* indicating whether the node is the sender or the receiver of the flow.

3) Assignment of flow weight: As mentioned, we assume that both guaranteed and best-effort flows exist simultaneously in the network. To support both QoS-aware and best-effort flows we assign flow weight to different flows according to the respective service requirement. QoS-aware flows specify their required bandwidth, Req_f and minimum bandwidth Min_f. The residual Bandwidth, bandwidth left after fulfilling the minimum requirements of all QoS-aware flows, are fairly distributed to all flows, both QoS-aware and best-effort flows. Such an assignment is given in [9], but as mentioned earlier, this may assign more bandwidth to QoS flows than required if number of QoS flows is less or the network is lightly loaded. Instead we fallow the following scheme to assign bandwidth to flows:

Weight of QoS flows, w_g	Weight of best-effort flows, w_b

For i = 1 to n {
w = $Min_f/C + (C-\sum Min_f)/(n + m)$
if w*C > Req_f
$W_g = C/Req_f$
else
W_g = w
}
	For i = 1 to m {
	$w_b = (C - (\sum w_g)*C)/m$
	}

N = number of QoS flows C = Link Bandwidth	m = number of best effort flows

Also the flow weight for a flow is not fixed for a path and every forwarding node will assign a different weight for a flow. And over time, based on the current flows within two-hop neighbors, the weight may change, otherwise either the network utilization or the fairness will be poor. In our proposed mechanism, after certain number of rounds, all the nodes will update their flow weights.

4) Tagging Operations: For each flow *f* we use the SFQ [10] algorithm to assign tags for the arriving packets: a start tag and a finish tag. But this tagging operation is dependent on the system virtual time. However, in a distributed environment, this information is not 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. For the head-of-line packet *k* of flow *f*, which arrives at time $A(t_k^f)$ and packet size is L_p, its start tag S_k^f and finish tag F_k^f are assigned as follows:

(i) Start tag:

If flow *f* is continuously backlogged, then

$$S_k^f = F_{k-1}^f \tag{3}$$

If flow *f* is newly backlogged, then

$$S_k^f = \max_{g \in S}\{V_g(A(t_k^f))\} \tag{4}$$

(ii) Finish tag:

$$F_k^f = S_k^f + L_p / w_f \tag{5}$$

where S consists of all flows stored in the *flow_table* of node *n*, and $V_g(t)$ is the flow *g*'s virtual time at *t*.

The start tag is used to find the transmission order of the packets. Based on the order of the start tag each flow is assigned a backoff value. For flows that have smallest service tags, in their local table, the backoff is zero; for each flow *f* in concurrent transmissions due to channel reuse, its backoff is set to be 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 sender and receiver. To solve this problem, we use a different approach. The service tag will be assigned by both the source and the destination of a flow when the source receives the packet. The source of a flow assigns the s_start and s_finish and receiver of the flow assigns r_start and r_finish. The largest of the start_tag is considered as the overall service tag of the flow and stored in the flow table of all the neighbors of source and destination. Now if the backoff value is set based on the service tag, this assignment is not only based on sender's flow table but based on the receiver's flow table as well.

5) Path Registration: To provide guaranteed service a routing protocol should find a feasible path. AODV [11] is one of the most widely used table-based and reactive routing protocols. But AODV is designed to find a feasible route only. Therefore, to support QoS we need to modify AODV. To ensure QoS, AODV is modified to support two types of schemes: *admission scheme* and *adaptive scheme*. In admission scheme a feasible path should provide the required minimum bandwidth, while in

adaptive feedback scheme the source is informed about the minimum available bandwidth so that the source can adapt its transmission speed.

To initiate QoS-aware routing discovery, the source host sends a RREQ packet whose header is changed to <*model-flag*, *required bandwidth*, *min-bandwidth*, AODV RREQ header>. The model-flag indicates whether the source is using the admission scheme or the adaptive feedback scheme. When an intermediate host receives the RREQ packet, it first calculates its residual bandwidth. If the model-flag is the admission scheme, the host compares its residual bandwidth with the minimum requested bandwidth. If its residual bandwidth is greater than the minimum bandwidth, it forwards this RREQ. Otherwise, it discards this RREQ. If the model-flag is adaptive, the host compares its residual bandwidth with the min-bandwidth field in the RREQ. If its residual bandwidth is greater than the min-bandwidth, it forwards the RREQ. Otherwise, it updates the min-bandwidth value using its residual bandwidth. Finally the forwarding node temporarily stores the flow. When a node forwards a RREP message it assigns a flow-weight to the flow and store the flow information.

6) Distributed Scheduling: Identifying the smallest tag among all backlogged nodes is a global computation. We take a table driven, backoff-based approach in scheduling flows. 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 are ordered so that each node can learn whether that node itself has the minimum tag, a distributed coordination among the neighboring nodes. Since we are also interested in maximizing spatial reuse, we do not confine the transmission to the minimum-tag holders only, because this will create the following problem:

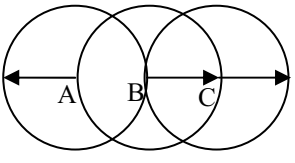

Fig. 1. Scheduling flows based on lowest service tag

As shown in Figure 1, according to nodes A's flow_table node B has the minimum service tag, so node A will not transmit. And according to node B's flow_table node C has the minimum service tag so node B will not transmit. But if node B does not transmit then both nodes A and C can transmit simultaneously and maximize the bandwidth utilization. Therefore if we allow only the minimum tag holders, then number of simultaneous transmission will be less and a less channel reuse. Instead, we set a backoff value to each node and the value is the number of nodes having 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 backoff period B_f in minislots as

$$B_f = \sum_{g \in S} I(T_g < T_f) \tag{6}$$

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 $T_g < T_f$; I(x) = 0, otherwise.

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.

7) Table Update: Whenever a node hears a new service tag for any flow on its table or a new flow, it updates the table entry for that flow or adds this flow information on its table. Whenever any node transmits a head-of-line packet for a flow, it updates that flow's service tag in the table entry.

5 Implementation of the Proposed Mechanism

In this section, we describe a distributed implementation of the proposed model within the framework of CSMA/CA MAC architecture. Our implementation addresses the following practical issues:

1) Message Exchange Sequence: In this mechanism, 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. As mentioned earlier, the node sets a backoff timer to the flow *f*, to be the number of flows with tags smaller than the tag of flow *f*. If the node does not hear any transmission then it decreases backoff value by one in each minislot. If the backoff timer of *f* expires without overhearing any ongoing transmission, it starts RTS 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. 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 [12]). When hosts hear either a CTS or a DS message, they will defer until the DATA-ACK transmission completes.

2) Maintaining Table Information at both the sender and receiver: To schedule a flow, a node should know the flow information of the neighbors of sender and receiver. This information needs to be combined and known by the sender to make the scheduling decision. A straight forward solution would be to broadcast the receiver 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 novel solution for this: both the sender and receiver will assign two separate service tags to the arriving packets of a flow *f* and they exchange the service tags. The larger value of the service tags will be the overall service tag for the packet. Both sender and receiver also distribute the service tag to their neighbors. And every time a packet of this flow is

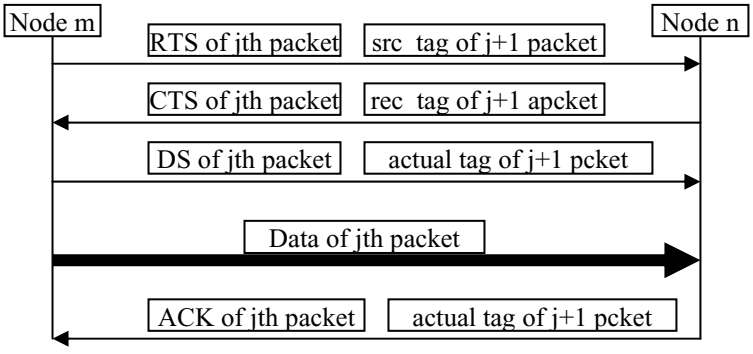

Fig. 2. Maintaining table information at both the sender and receiver

transmitted the service tag of the next packet is updated and distributed. Let P_f^j is the jth packet of flow *f*, and node *m* and *n* are the sender and receiver respectively. When node *m* receives the packet, its ACK is heard by the next node of the packet and the receiver assign a service tag to this packet and store it in local_table. By this way, both source and receiver can assign service tag to a packet. Now when a node transmits jth packet in the RTS frame, sender transmits the service tag of (j+1)th packet. In the following CTS packet, the receiver transmits its service tag for (j+1)th packet. So, both the source and receiver can have the service tag of its counterpart and select the largest value as the service tag for that packet. DS packet transmitted by the source announces the actual service tag of the next packet for flow *f* and all the neighbors of source hear this value. Finally the ACK frame of jth packet from receiver contains the actual service tag for the next packet of flow *f* and all the neighbors of m hear this. The complete operation is shown in figure 2.

3) Propagation of 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 for flow *f* in DS and ACK frames as shown in Figure 2. Since every node within one-hop of the sender and receiver has the updated flow tag, chance of collision is less.

6 Simulation and Results

In this section, we evaluate our proposed algorithm by simulations. Several performance metrics are used to evaluate the algorithm. N_l: Number of transmitted packets of a flow during the simulation time; N_s: Number of transmitted packets of a flow during a short interval. We measured the fairness of our protocol when both guaranteed and best-effort traffics coexist in the network and when only best-effort traffics exist in the network. For simulation we consider the flow graph as shown in figure 3.

Each of our simulations has typical run of 10,000 time units and we assume that physical channel capacity C is one slot per time unit. To obtain measurements over short time windows, we measure the performance over the 10 different time windows, each of which have 100 time units, and averaged the results. For all cases, we consider constant rate of source traffic.

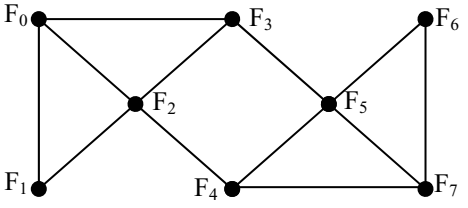

Fig. 3. Example flow graph used in simulation

Table 1. (a) Theoretical fair queueing in example scenario 1 (b) Implemented fair queueing in example scenario 1

Flow	N_1	N_s
0	3333	34
1	3333	34
2	3333	33
3	3333	33
4	3333	34
5	3333	33
6	3333	34
7	3333	33

(a)

Flow	N_1	N_s
0	3298	33
1	3290	32
2	3255	30
3	3305	33
4	3278	31
5	3295	33
6	3288	32
7	3267	31

(b)

Example1: In this example, we evaluate the fairness of our theoretical algorithm where only best-effort traffics are present in the network. Also we measure the efficiency of achieving spatial reuse of bandwidth. The throughput achieved by each flow is given in Table 1(a). Flow F_0, F_4 and F_6 can be transmitted simultaneously and F_1, F_3, F_7 and F_2, F_5 are also two independent sets. The optimal throughput under fairness constraint is 266% for this flow graph. We applied our implemented method to the flow graph in Figure 3. Simulation result is shown in Table 1(b).

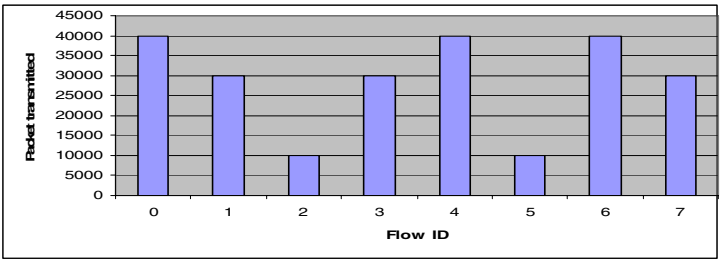

Fig. 4. Example flow graph used in simulation

Example 2: Now we consider the presence of both guaranteed and best-effort flow in the network. In our simulation both F_0 and F_1 are guaranteed flows and all the other flows are best-effort flows. Minimum Bandwidth requirement for F_0 is 0.4C and for

F_1 is 0.3C where C is the link bandwidth. The simulation result is given in Figure 4. The simulation result shows that our protocol fulfills the requirement of guaranteed flows and then extra bandwidth is divided between all the flows, as expected.

7 Conclusions

In this paper, we proposed a distributed fair queueing algorithm for providing scheduling service in an ad hoc network. Our proposed mechanism also provides requested bandwidth to guaranteed flows if available. It assigns flow weight to flows based on flow types, it first assigns bandwidth to guaranteed flows according to their minimum requirements and the remaining bandwidth is divided to all flows. Finally, we describe a distributed algorithm for providing fair scheduling in ad hoc networks. As a future works, we like to apply our proposed algorithm to mobile ad hoc networks.

References

1. A. Demers, S.Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in Proc. ACM SIGCOMM, Aug. 1989, pp. 1–12.
2. S. Lu, V. Bharghavan, and R. Srikant, "Fair scheduling in wireless packet networks," IEEE Trans. Netw., pp. 473–489, Aug. 1999
3. T. S. Ng, I. Stoica, and H. Zhang, "Packet fair queueing algorithms for wireless networks with location-dependent errors," in Proc. IEEE INFOCOM, San Francisco, CA, Mar. 1998, pp. 1103–1111
4. P. Ramanathan and P. Agrawal, "Adapting packet fair queueing algorithms to wireless networks," in Proc. ACM MOBICOM, 1998, pp. 1–9
5. H. Luo and S. Lu, "A self-coordinating approach to distributed fair queueing in ad hoc wireless networks," in Proc. IEEE INFOCOM, Apr. 2001, pp. 1370–1379
6. H. Luo and S. Lu, "A topology-independent fair queueing model in ad hoc wireless networks," in Proc. IEEE Int. Conf. Network Protocols, Nov. 2000, pp. 325–335
7. H. L. Chao and W. Liao, "Credit-based fair scheduling in wireless ad hoc networks," in Proc. IEEE Vehicular Technology Conf., Sept. 2002
8. Jerry Cheng and Songwu Lu, "Achieving Delay and Throughput Decoupling in Distributed Fair Queueing Over Ad Hoc Networks", IEEE ICCCN, 2003
9. H. L. Chao and W. Liao, "Fair Scheduling With QoS Support in Wireless Ad Hoc Networks", IEEE Trans. on Wireless Comm., vol. 3, no. 6, November 2004
10. P. Goyal, H.M. Vin and H. Chen, "Start-time fair queueing: A scheduling algorithm for integrated service access," ACM SIGCOMM'96. August 1996.
11. C. Perkins, E. Belding-Royer and S. Das "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, July 2003
12. V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Medium Access Protocol for Wireless LANs," Proc. ACM Ann. Conf. Special Interest Group on Data Comm. (SIGCOMM), 1994.