ACHIEVING FAIR SERVICE IN LARGE-SCALE MULTIPLE ACCESS NETWORKS

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

Keywords: CSMA/CA, ad hoc wireless networks, distributed fair queueing

Abstract Providing fair share of bandwidth is critical to support communication-intensive applications in a shared-medium, capacity-constrained, large-scale infrastructureless wireless network. The proposed design has to be distributed, scalable and address the issue of location-dependent contention. In this paper, we present a novel packet scheduling solution that achieves distributed fair service in such multihop, multiple access wireless networks. Our design works well with the CSMA/CA MAC framework. We evaluate our proposed design through both analysis and simulations.

1. Introduction

Emerging infrastructureless wireless networking technologies such as MANET, bluetooth and sensor networks will seek to support advanced applications such as collaborative learning, smart environment, zero-configuration conferencing, and mission-critical military operations. This class of applications is communication intensive and requires sustained level of bandwidth support for efficient operation. Therefore, the issue of providing fair services (in terms of throughput and bounded delay channel access) for multiple contending hosts over a scarce and shared wireless channel has come to the fore. Fair queueing has been a popular paradigm to achieve this goal in both wireline [1, 2] and packet cellular networks [3, 4]. However, achieving fair service through fair queueing is non-trivial in a shared-medium, large-scale ad hoc network. The solution has to address issues such as location-dependent contention and errors, user mobility, node failures, and scalable state management in

a large-scale network, etc. Besides, the proposed design has to be fully distributed, and be scalable to a large number of nodes and flows.

The problem of fair queueing in ad hoc networks has been formulated in two recent works [7, 9]. In each work, the authors have proposed ideal centralized fair queueing models (that address design issues raised above) by assuming a centralized scheduler and perfect knowledge of each flow in the entire network topology, and then designed distributed implementations to approximate the idealized model. The proposed implementation has taken a backoff-based approach, in which each flow backs-off for a number of minislots before contending for the channel within the CSMA/CA MAC protocol. The backoff value is set appropriately such that flows with higher precedence for channel access at t (based on the scheduling decision) will be assigned smaller backoff intervals thus having priority for channel access. This way, the implementation is fully distributed and no centralized scheduler is needed. It has been shown through simulations that this implementation can reasonably approximate the ideal centralized model [7, 9]. However, this approach suffers from three limitations: (a) the backoff-based design requires synchronization; (b) large backoff value leads to long durations of idle time, and this happens if flow weights are small; (c) it requires fine-grain timer support for backoff timers.

In this paper, we propose a novel method to achieve fair service through distributed fair queueing within the CSMA/CA MAC framework in ad hoc networks. In our design, each flow maintains local information for its local neighboring flows and makes localized scheduling decisions. We show that the local schedulers self-coordinate their local interactions to collectively achieve global fair services among competing flows. Two key contributions of this work are the following: (a) a *fully distributed* and *localized* fair queueing algorithm, in which local schedulers collectively achieve desired global fair services, and (b) a novel table-driven distributed implementation within the framework of CSMA/CA paradigm. Our simulations and analysis demonstrate that our proposed approach provides a localized, scalable, and efficient solution to distributed fair queueing in ad hoc networks.

The rest of the paper is organized as follows. Section 2 identifies the key design issues of distributed ad hoc fair queueing that we are addressing in this paper. Section 3 describes a new algorithm to achieve distributed fair queueing and a distributed implementation within the CSMA/CA MAC protocol. Section 4 evaluates the proposed design through simulations. Section 5 concludes the paper.

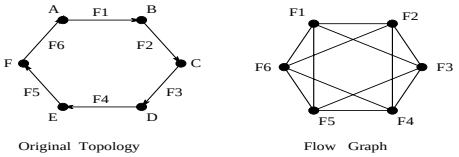

Figure 1 Location-dependent contention

2. Design Issues

In this paper, we consider a CSMA/CA based, packet-switched multihop wireless network in which a single, shared physical channel with capacity C is available for wireless transmissions. Transmissions are locally broadcast and only receivers within the transmission range of a sender can receive its packets. Each link-layer packet flow is a stream of packets being transmitted from the source to the destination, where the source and destination are neighboring nodes that are within transmission range of each other.

We make three assumptions [5, 7, 8]: (a) a collision occurs when a receiver is in the reception range of two simultaneously transmitting nodes, thus unable to cleanly receive signal from either of them; we ignore capture effect, (b) a node cannot transmit and receive packets simultaneously, and (c) neighborhood is a commutative property; hence, flow contention is also commutative. In this work, we do not consider non-collision-related channel errors. For simplicity of presentation, we only consider fixed packet size in this paper, which is a realistic assumption in typical wireless networks.

We focus on two design issues in this work:

(a) Spatial coordinated design in the presence of Location-dependent contention Since wireless transmissions are locally broadcast, collisions and contention for the shared medium, are location dependent. Consider the example shown in Figure 1. Flow F_1 contends with flows F_2, F_3, F_5, F_6, since these four flows are within the transmission range of F_1. Therefore, these four flows should restrain from transmissions when F_1 transmits. Similarly, Flow F_2 contends with flows F_1, F_3, F_4, F_6. Hence, each flow has a different contending flow set depending on its location. Furthermore, since wireless transmissions are locally broadcast, any two flows that are not interfering with each other can potentially transmit data packets over the physical channel simultaneously, thus increasing the effective system throughput. In Figure 1, flows F_1 and F_4 are not interfering with each other and can transmit concurrently.

Location-dependent contention brings new issues to packet scheduling in ad hoc networks. In a wireline or packet cellular network, packets are scheduled *independently* at each link. The scheduler at each link needs

to consider flows contending for that link only [1, 2]. In a shared-medium multihop wireless network, location-dependent contention implies that, any scheduling decision made at a node will have impact on its neighbors and incur domino effects in the entire connected network graph. Therefore, flow scheduling decision cannot be made with respect to each node's "local" flows and independent of its neighbors. Flow scheduling has to be coordinated among neighbors that have contending flows. Spatial coordinated design among neighboring nodes is a must.

(b) A case for distributed fair queueing Distributed fair queueing in ad hoc networks is motivated by two unique characteristics of such networks: (a) No *single* logical entity for scheduling flows in the network graph is available. Contending flows may originate from different sending nodes, and each node needs to implement a local scheduler for its transmitting flows. (b) Flow information is only available at each sending node, and each sender does not have direct access to other flows' information at other senders. Consider Figure 1 again, each of the six senders $A - F$ does not know the packet-level flow information at the other nodes.

3. A New Approach to Distributed Fair Queueing

In this section, we will develop a novel distributed fair queueing algorithm for a large-scale multihop wireless network. To this end, we first generate a *flow contention graph*, which characterizes exact contentions among transmitting flows. In a flow contention graph, each vertex represents a flow, and an edge between two vertices denotes that these are two contending flows. An example is shown in Figure 1.

3.1. Overview of the proposed algorithm

In our algorithm, each local scheduler maintains *updated* local (one-hop neighbor) flow information and performs local computation only. Each scheduler uses the start-time fair queueing (SFQ) algorithm [2] to assign a service tag for each packet and schedule packets based on the service tag. However, in order to achieve global fair services among contending flows, each scheduler does not schedule a flow f for transmission at time t unless f has the smallest service tag in its *local* one-hop neighborhood of the flow contention graph at t. This way, only flows that receive local minimum normalized services (i.e., they have local minimum service tags) in the entire connected flow graph are scheduled for transmission. This is what we called *maximizing local minimum* policy.

Our "maximizing local minimum" scheduling policy is motivated by the following fact: in order to ensure minimum fair share for each flow,

the flow with the global minimum service tag in the entire network must be scheduled for transmission with precedence. However, this is challenging in distributed fair queueing, since identifying the flow with the global minimum service tag requires sorting all flows in the entire network graph and it is a global computation task. In our policy, we identify all flows with local minimum service tags, and schedule all such flows for transmission. Since the global minima must be a local minima (but not vice versa), we know that the flow with the global minimum tag must be among these transmitted flows that have local minimum tags. Hence, the "maximizing local minimum" policy is a superset of the "maximizing global minimum" policy. An additional benefit of our policy is that we schedule multiple non-interfering flows simultaneously thus effectively increasing the system throughput.

In essence, each local scheduler using our fully distributed, localized algorithm coordinates its local scheduling decisions with its neighbors in order to achieve global fair services. This is done without global computation or global information propagation. The solution requires simple computations and is scalable to a large number of nodes.

3.2. Algorithm description

Specifically, in our algorithm, each node maintains a local flow table, and the table records information of all its one-hop neighboring flows in the flow contention graph. Each node is responsible for assigning start tags and finish tags for flows that it serves as the sender using SFQ [2]. In each flow table entry, we record the following information: $[flow\_id,\ \ flow\_tag]$, where the $flow\_tag$ is the *most recent* service tag that the node hears for flow $flow\_id$.

The detailed operations consist of four parts:

1 *Local state maintenance*: At each node n, it maintains a local table T_n, which records each flow's current service tag for all flows $\mathcal{S}$ that the node serves as the sender or receiver and their one-hop neighboring flows in the flow graph. Each table entry has the form of $[f,\ V_f]$, where V_f is the current virtual time of flow f, i.e., the most recent start tag of flow f.

2 *Tagging operations*: For each flow f that node n serves as the sender, we simulate the SFQ algorithm [2] 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 a finish tag F_k^f are assigned as follows:

(a) If f is continually backlogged, then

$$S_k^f = F_{k-1}^f; \quad F_k^f = S_k^f + L_p/r_f.$$

(b) If f is newly backlogged, then

$$S_k^f = \max_{g \in \mathcal{S}}\{V_g(A(t_k^f))\}; \quad F_k^f = S_k^f + L_p/r_f,$$

where $\mathcal{S}$ consists of all flows stored in the table of node n, and $V_g(t)$ is flow g's virtual time at t.

3 *Scheduling loop*: At node n, whenever it hears that the channel is clear, if one of flows in T_n, say f, has the *smallest* service tag in the table T_n, the sender n of f transmits the head-of-line packet of flow f and piggybacks V_f with the packet transmission.

4 *Table updates*: whenever node n hears a new value $V_g^{'}$ for its neighboring flow g on its table T_n, it updates the table entry for flow g to $[g, V_g^{'}]$. Whenever node n transmits a head-of-line packet for flow f, it updates flow f's service tag in the table entry.

3.3. Performance analysis

In the following, we briefly characterize the properties of the proposed algorithm. We omit the proof due to lack of space.

Proposition 1 *(bounded flow unfairness) Given any two backlogged flows f and g in the connected network graph, their received services $W_f(t_1, t_2)$ and $W_g(t_1, t_2)$ during time interval $[t_1, t_2]$ satisfy:*

$$|\frac{W_f(t_1, t_2)}{r_f} - \frac{W_g(t_1, t_2)}{r_g}| < \Delta \qquad (1.1)$$

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

Proposition 2 *(Minimum fair share for each flow) The algorithm guarantees that each continually backlogged flow f receives a minimum fair share of the channel capacity C. That is,*

$$W_f(t_1, t_2) \geq C\frac{r_f}{k\sum_{g \in \mathcal{S}'} r_g}(t_2 - t_1) - \alpha \qquad (1.2)$$

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

3.4. Implementation Within CSMA/CA

We now describe a practical implementation within the CSMA/CA MAC paradigm. Our implementation seeks to address two issues:

(a) Exchange of the table information at a flow's sender and its receiver: In the algorithm of Section 3.2, each node maintains information for flows within one-hop neighborhood in the flow contention graph. However, one-hop neighborhood in a flow contention graph translates to two-hop neighborhood in a node graph. Information on the one-hop neighboring flows in the flow graph is distributed in both the sender and the receiver.

(b) Propagation of each transmitting flow's updated virtual time in its one-hop neighborhood: In our algorithm, the table at each node needs to record the most recent virtual time for each neighboring flow in the flow graph. Whenever a flow transmits, all the senders and receivers of its neighboring flows should update the new virtual time for this flow.

A brief overview of our implementation is as follows:

In our implementation, each data transmission follows a basic sequence of RTS-CTS-DS-DATA-ACK handshake [5], in which DS is introduced to notify nodes in the one-hop neighborhood of a flow's sender to defer their transmissions. When the sender of flow f has the minimum service tag in its table, it initiates a RTS request. When the receiver hears RTS, if flow f also has the minimum service tag in the receiver's flow table, the receiver responds with a CTS message. Flow f's sender then replies with a DS message, and all nodes in the sender's neighborhood defer until f completes its data packet transmission upon hearing DS. After DS, the sender and the receiver complete with the DATA-ACK handshake. Whenever a node hears a collision, it randomly backs off using a standard binary exponential backoff scheme. Whenever a backlogged node has detected the channel to be idle for an extended period of time, it starts to broadcast the virtual times for flows that this node serves as the sender or the receiver.

In order to propagate a flow's updated virtual time to nodes in the one-hop neighborhood of both its sender and its receiver, we piggyback this information in DS and ACK messages. Besides, note that a flow f's sender always has *correct* information (i.e., current service tag) on f and it is responsible to propagate this accurate information to its neighbors. Hence, only *accurate* and correct information will be propagated in each local neighborhood of the network graph.

3.5. Further comments

We have three further comments on our proposed algorithm:

Node mobility In an ad hoc network, each node can be mobile, thus changing the network topology dynamically. In a highly mobile wireless network, any model that requires global topology information or global

computation is not feasible. Note that both our proposed localized algorithm and its implementation require only one-hop flow information (i.e., each flow's ID, and its current virtual time) and simple local computation, this feature makes our design work well in the presence of node mobility. However, if a node is mobile, it does take several packet transmission times to upload its table in the new location. Ongoing work seeks to evaluate this aspect via simulations.

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

Synchronization Our proposed algorithm does not need any synchronization in the system for its operations; this is different from the backoff-based approach [7, 9] that requires accurate synchronization for the backoff timers.

4. Simulation Evaluation

In this sections, we use simulations to evaluate our proposed algorithm (we name it as MAX-MIN MAC), which have been implemented within the CSMA/CA MAC protocol. We compare our algorithm with the IEEE 802.11 MAC protocol.

Our algorithm was implemented within the GloMoSim library [10]. The radio model is based on existing commercial wireless network (e.g. Lucent WaveLAN), with a radio transmission range of 150 meter and channel capacity 2Mbits/sec. The packet size we use in our simulations is 512 bytes. Each simulation lasts for 1000 seconds. We compare the inter-flow fairness and the total aggregate throughput for both protocols. For comparison purpose, we set the flow weight to be the same for all simulations, but flow weights can be set arbitrarily in our algorithms.

In the following, we present three simulation examples.

Example 1 In this example, we consider a simple linear topology as shown in Figure 2. In this example, there are five flows, and the corresponding flow contention graph is shown in Figure 3. We compare the throughput of our implementation within CSMA/CA with an FIFO scheduler within the IEEE 802.11 MAC. The simulation results are shown in Figure 4 and Table 1. From the simulation results, we can see that our protocol achieves fair service for all five flows. However, the

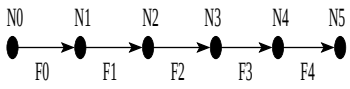

Figure 2 Example 1: Node graph

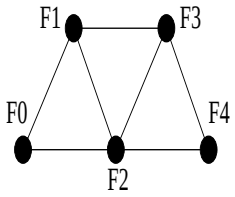

Figure 3 Example 1: Flow contention graph

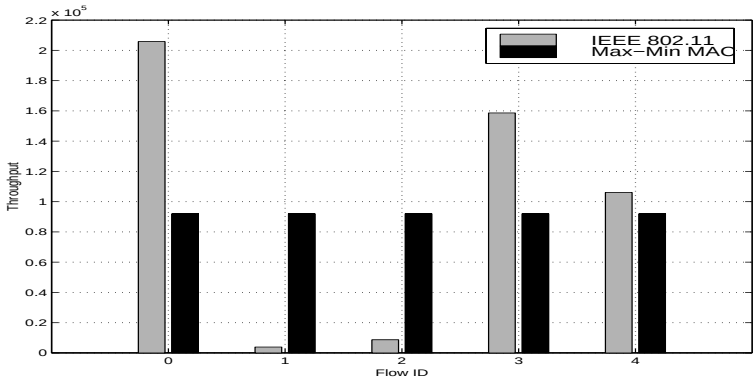

Figure 4 Example 1: Throughput comparison between our algorithm and FIFO scheduler with 802.11 MAC

FIFO scheduler with 802.11 MAC does not achieve fair bandwidth allocation, flows F_1 and F_2 are almost starved and receive much less service than the other three flows. In this example, we also see that our implementation achieves 95.2% aggregate throughput compared with 802.11 MAC. Note that the 4.8% throughput decrease is mainly due to the requirement of fair allocation (since fairness prevents a fixed set of flows transmitting all the time), instead of implementation overheads e.g. the introduction of the DS control message. In fact, we observe in our simulations that our implementation significantly reduces the number of collisions compared to the 802.11 standard.

Example 2 In this example, we compare the performance of our implementation and the 802.11 MAC protocol in an asymmetric topology. In this topology (shown in Figure 5), we have 9 flows and the flows belong to two cliques in the flow contention graph (shown in Figure 6). The simulation results are shown in Figure 7 and Table 2. Again we achieve near perfect inter-flow fairness, but the 802.11 MAC protocol suffers from unfairness among the nine flows. The aggregate throughput of our algorithm is about 99.85% compared to the IEEE 802.11 standard. Even

Flow	802.11 MAC	MAX-MIN MAC
0	205789	91983
1	3917	91982
2	8709	91981
3	158614	91983
4	106058	91982
Total	483087	459911

Table 1 Ex.1: Throughput comparisons

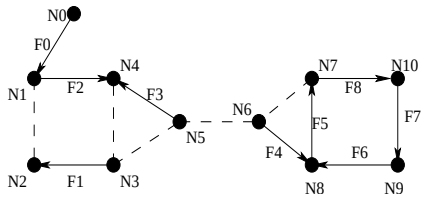

Figure 5 Example 2: Node graph

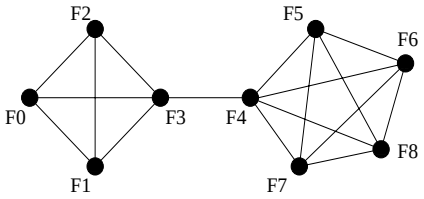

Figure 6 Ex. 2: Flow contention graph

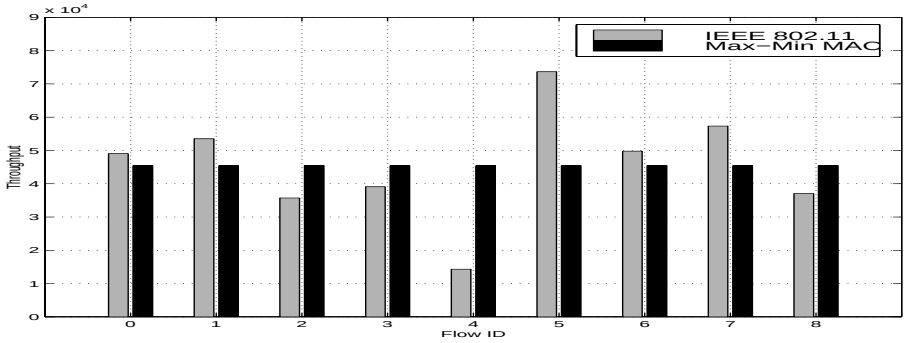

Figure 7 Example 2: Throughput comparison between our algorithm and FIFO with 802.11 MAC

though the DS message and piggybacked virtual time will incur about 6.7% overhead, our design has reduced collisions of the 802.11 standard. Therefore, we only observe negligible decreases in terms of the aggregate throughput.

Example 3 In this simulation, we evaluate a scenario with 33 nodes and 21 flows as shown in Figures 8 and 9. The simulation results are shown in Figure 10. From the figure, we can observe that our protocol achieves perfect fair service for every flow. However, the total aggregate throughput is only 64.5% of the IEEE 802.11 protocol due to the fairness constraint. On the other hand, the IEEE 802.11 protocol provides larger

Flow	802.11 MAC	MAX-MIN MAC
0	49027	45431
1	53504	45430
2	35730	45431
3	39090	45431
4	14337	45432
5	73644	45431
6	49809	45431
7	57328	45432
8	37036	45431
total	409505	408880

Table 2 Example 2: Throughput comparisons

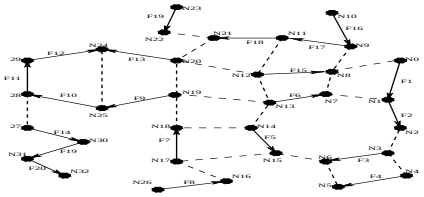

Figure 8 Example 3: Node graph

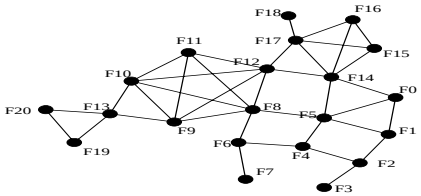

Figure 9 Ex. 3: Flow contention graph

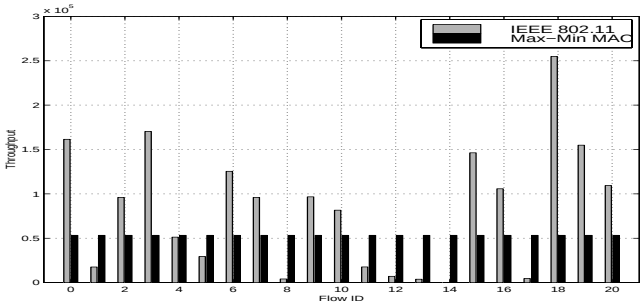

Figure 10 Example 3: Throughput comparisons

aggregate throughput, but at the cost of flow starvation, e.g., flows 8, 12, 13, 14 and 17 are nearly starved. This illustrates the inherent conflict between achieving fairness and maximizing overall system throughput in a generic topology.

5. Conclusion

In this paper, we have proposed a novel localized and fully distributed fair queueing algorithm to achieve fair services in ad hoc wireless networks. Our proposed algorithm seeks to devise a scalable and efficient solution to ad hoc fair queueing problem within the CSMA/CA MAC

framework. Our algorithm relies on local information and local computations only, and multiple localized schedulers coordinate their interactions and collectively achieve desired global fair services. We demonstrate the effectiveness of our proposed design through both simulations and analysis. Ongoing work seeks to improve the design of the distributed implementation, to perform more extensive simulations, and to refine the analytical bounds of the proposed algorithm.

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," *ACM SIGCOMM'97*, September 1997.

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

[6] "IEEE 802.11 Standard Specification," 1997.

[7] N. H. Vaidya, P. Bahl and S. Gupta, "Fair scheduling in broadcast environments," *Tech. Rep. MSR-TR-99-61*, Microsoft Research, August 1999.

[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 and S. Lu, "Fair queueing in ad hoc wireless networks," *to appear in IEEE ICNP'00*, November 2000.

[10] "Global Mobile Information Systems Simulation Library (GLOMOSIM)," *http://pcl.cs.ucla.edu/projects/glomosim/*.