QoS-Aware Fair Scheduling in Multihop Wireless Ad Hoc Networks

Muhammad Mahbub Alam, Md. Abdul Hamid, Choong Seon Hong
Department of Computer Engineering, Kyung Hee University
1 Seochun-ri, Yongin-si, Gyunggi-do, Korea, 449-701
{mahbub, hamid}@networking.khu.ac.kr and cshong@khu.ac.kr

Abstract — **Providing fairness and maintaining a certain quality of service (QoS) in shared medium, multihop wireless networks are challenging, due to the unique characteristics of these networks. In this paper we consider the presence of both guaranteed and best-effort flows in the network. The aim is to ensure minimum required bandwidth to the best-effort flows and provide an equal share of residual bandwidth to all the 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.**

Keywords — **Ad hoc networks, fair scheduling, quality of services.**

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 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 flow *f* and *g* respectively in time interval [t1, t2]. The allocation is ideally fair if it satisfies (1)

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

for all flows *f* and *g*.

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 in ad hoc networks, 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 scheduling 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 for all flows.

This paper is organized as follows. Section 2 describes related works. Section 3 describes the design issues of fair queueing in ad hoc networks. In section 4 we explain our proposed mechanism. Section 5 details the implementation of the proposed mechanism. 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 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. [8], [9] provide both QoS guarantee and fair scheduling in ad hoc networks.

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

* This work was supported by ITRC project of MIC

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 is 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. [9] proposed a method to assign weight to flows and used (2).

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

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 Scheduling

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

1) 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.

2) Conflict 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.

3) Distributed Nature of Ad Hoc Fair Scheduling: 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.

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. The proposed mechanism

We now describe a new approach to QoS-aware distributed fair scheduling 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 a flow table for the proper operation of fair scheduling. The table keeps 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*, *service_tag* and *flow_weight*.

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 use the following scheme to assign bandwidth to flows:

```
Weight of QoS flows, $w_g$
    For i = 1 to n {
        $w = Min_f/C + (C-\sum Min_f)/(n + m)$
        if $w*C > Req_f$
                $Wg = C/Req_f$
        else
                $Wg = w$
    }
Weight of best-effort flows, $w_b$
    For i = 1 to m {
        $w_b = (C - (\sum w_g)*C)/m$
    }
where
  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 Lp, its start tag S_k^f and finish tag F_k^f are assigned as follows:

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

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

Finish tag:

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

where S consists of all flows stored in the *flow_table* of node n, and Vg(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. Every transmitted packet contains the arrival time and service tag of the next packet. Accordingly the receiver checks its own flow table and returns the number of flow having lower service than this packet. So the sender can set the backoff value based on the receiver response.

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 initiate QoS-aware routing discovery, the source host sends a RREQ packet whose header is changed to <model-flag, required bandwidtht, min-bandwidth, AODV RREQ header>. The model-flag indicates whether the source is using the admission scheme or the adaptive feedback scheme. In admission scheme a feasible path should provide the required bandwidth, while in adaptive feedback scheme the source is informed about the minimum available bandwidth so that the source can adapt its transmission speed. 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 requested bandwidth. If its residual bandwidth is greater than the requested 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. And 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) Scheduling of Flow: At any node whenever it sense the channel is free, if one of its flow has the smallest service tag in the table then it immediately transmits the head-of-line packet of this flow. Otherwise it updates the backoff value of this flow. Collision may occur if two nodes have the backoff value zero. This may happen if any node has a new flow with the same service tag and the neighbors do not have the information. To solve the problem we use the RREP and HELO message. Every node hears the RREP message and aware of a new flow. When any node gets the first packet of a new flow, it broadcasts the service tag the first packet of the flow and all of its neighbors update their flow table. Also, the immediate next node (i.e., the receiver of the flow) rebroadcasts this to inform all of its neighbors. So, each node has the knowledge of its local neighborhood. Even then two flows can have the same backoff value but it is known to all the two-hop neighbors. And we break the tie by assigning priority to nodes having higher flow weight and lower node id. For example, if two nodes have the zero backoff value then nodes with the higher weights will transmit first and if they have the same weight then node with lower ID will win.

6) 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-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). When hosts hear either a CTS or a DATA 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. Both sender and receiver also distribute the service tag to their neighbors. And every time a packet of this flow is transmitted the service tag of the next packet is updated and distributed. We use the RTS and CTS packets for transmitting the updated service tag of the next packet of a flow. But the transmission of RTS and CTS packet does not guaranty that all the two-hop neighbors will hear the update because RTS or CTS packet may collide. So the neighbors temporarily store this update. And the neighbors of the sender permanently update this information when they hear the transmission of DATA packet and the neighbors of the receiver do this when they hear the transmission of the ACK packet. By this way, we can avoid the transmission of DS packet.

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 DATA and ACK frames as shown in Figure 1. 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; Ns: 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 1.

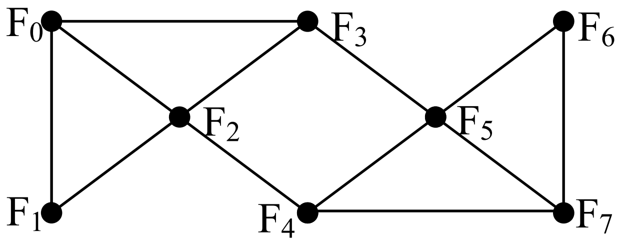

Figure 1. Example Flow Graph Used in Simulation

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.

Example1: In this example, we evaluate the fairness property 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. As flow F0, F4 and F6 can be transmitted simultaneously and also F1, F3, F7 and F2, F5 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 shown in Figure 1 and got similar results.

Table 1: Fairness of Flows for example scenario 1.

Flow	N_l	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

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 2. 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.

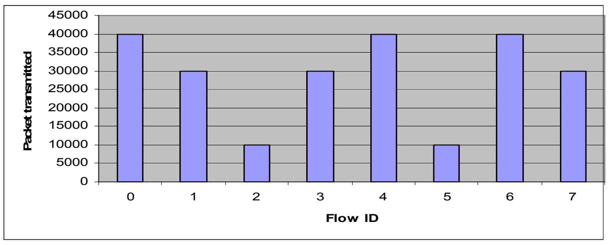

Figure 2: Simulation Result of Fairness and QoS Guarantee.

7. Conclusion

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.