A History-based Scheduling Protocol for Ad Hoc Networks

Yu Du, Ye Bao, and J.J. Garcia-Luna-Aceves
Department of Computer Engineering
University of California
Santa Cruz, CA 95064, U.S.A
duyu, yebao, and jj@cse.ucsc.edu

Abstract – This paper presents the history-based scheduling (HBS) protocol for collision-free channel access in ad hoc networks. In HBS, the channel is scheduled based on the history of activity of each node in order to attain higher channel utilization than traditional distributed scheduling schemes based on node activation. Conflict-free access to the channel is determined at each node based on a priority list of the nodes within two hops of each node that takes into account the activity history of each node. To keep the activity history of each node synchronized, a node that is assigned the channel and has no data packet to transmit simply transmits a "Nothing-to-Transmit" (NT) packet in that time slot. In this way, the exchange of the signal packet should be reduced. The throughput and delay characteristics of HBS are compared analytically and by simulation with those of CSMA/CA and the node activation multiple access (NAMA) protocol.

Keywords – ad hoc networks, history-based scheduling protocol (HBS), nothing-to-transmit packet

I. Introduction

Today's channel access protocols for ad hoc networks are contention-based and schedule-based. Contention-based channel access protocols include ALOHA [1], carrier sense multiple access (CSMA) [6], and CSMA with collision avoidance (CSMA/CA) [8]. CSMA [6] provides a dramatic improvement over ALOHA in the absence of hidden terminals, because the sender transmits its packet only when the channel is free. Because of the propagation delay τ, collisions can still happen when multiple senders transmit within τ seconds of each other. CSMA/CA schemes attempt to remedy the limitations of CSMA in the presence of hidden terminals by establishing handshakes between senders and receivers using short control packets before data packets are sent. However, CSMA/CA schemes degrade rapidly when the number of competing nodes increases, and have been shown to provide unfair channel sharing [9], [10].

Schedule-based protocols establish dynamic or static transmission schedules that allow nodes to transmit without collisions. Among the protocols that provide dynamic transmission schedules, the protocols proposed by Bao and Garcia-Luna-Aceves [2], [3] have been shown to perform very competitively compared to the unified framework for (T/F/C)DMA channel assignment, which is the best-performing heuristic known to date based on full topology information [7], while using very limited information. More specifically, these scheduling protocols operate by means of a distributed election run at each node and based on a priority list computed at each node. The priority list at each node is based on the identifiers of nodes within two hops of the node. To determine which node should be given the opportunity to access the common channel to transmit to all its neighbors, a hash function is applied to the list of two-hop neighbors and a single node is elected as having the highest priority during each contention period, which in our case is a time slot.The hash function ensures that all nodes have an equal likelihood to become the node with the highest priority.

However, while the scheduling approach introduced in [2] supports collision-free channel access using very limited information, nodes may be given the opportunity to transmit when in fact they have no data to transmit, which wastes channel bandwidth. This is the motivation behind the history-based scheduling (HBS) protocol introduced in this paper. HBS is a channel access scheduling protocol with neighborhood awareness. It modifies the neighbor-aware contention resolution (NCR) scheme [2] by associating a weight to each node based on its transmission history. Time is divided into time slots and each time slot is scheduled for one or more nodes to transmit data packets. Section II presents HBS in detail, and shows that it provides collision-free channel access without deadlocks. Section III analyzes the throughput estimated delay of HBS and NAMA [2]. Section IV compares the performance of HBS through simulations with NAMA and the idealized CSMA/CA.

II. History-based Scheduling Protocol (HBS)

A. Neighborhood aware contention

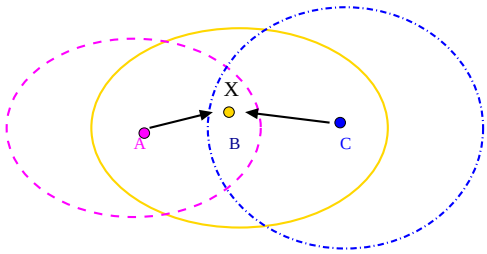

Fig. 1. *Example of Hidden-Terminal Problem*

Fig. 1 shows an example of the hidden-terminal problem. Because nodes A and C cannot sense each other, a collision occurs if each of them sends a data packet within the same packet time. Fig. 2 illustrates an example of the exposed-terminal problem, which is another problem for contention-based media access protocols.

To schedule channel access without the collision due to hidden terminals, each node must have knowledge of its contenders, i.e., its one-hop and two-hop neighbors, and the contention context, i.e., the current time slot number. In the above example, if only one node among nodes A, B and C can be elected to transmit data during a time slot, collisions due to a hidden terminal are avoided. By periodically broadcasting the identifiers of its one-hop neighbors, each node knows

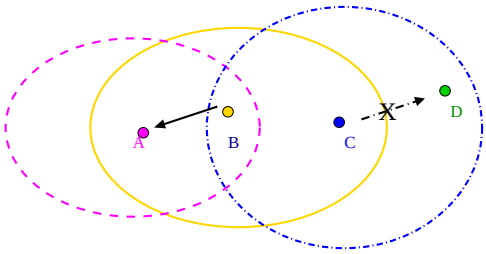

Fig. 2. *Example of Exposed-Terminal Problem*

its contenders. Also, we assume that all the nodes are synchronized, so that the contention context t is the same at all nodes and is known by each node.

B. Scheduling Protocol with Weight

Because applications using a wireless network typically require multiple packets to flow, there is high correlation between a node's activity in the past and its activity in the future. Accordingly, we assign a weight-level to each node that captures the node's history of activity to predict its future activity. We also define the maximum and minimum weights for each node and initialize each node with the maximum weight.

When the ratio of the number of packets sent by a node over the number of the time slots scheduled to that node is smaller than a certain threshold, the node's weight is decreased by one, and this process is repeated until the node's weight reaches the minimum value. In the same way, the weight of a node can be increased by one up to the maximum weight, while the ratio of the number of packets it sent to the number of time slots it won is larger than another threshold.

i: identifier of the node
M_i: set of node i's two-hop neighbors
k: ID of a node within two-hop range of node i, $k \in M_i$
t: time slot
w_i: weight of node i
w_k: weight of node k

Compute the priority of node i
  $p(i,t) = Rand(i,t) \cdot w_i$;
Compute the priority of all i's contenders
  $p(k,t) = Rand(k,t) \cdot w_k, k \in M_i$;
if (the i's priority is higher than all of its contenders',
  i.e., $\forall j \in M_i, p(j,t) < p(i,t)$)
  if (node i has a data packet to send)
    send a data packet;
    else send an NT packet;
  end if;
end if;

Fig. 3. *Algorithm of Channel Access in a Data Section*

Fig. 3 shows the algorithm used to schedule access the channel in a data section. For each node i, k and t are known and consistent, because each node knows its contenders and the contention context. w_i is known because a node knows the actual number of time slots it was assigned and the number of packets it sent in the last time slot and its weight-level is determined by the ratio of these two numbers. But it is not so easy to calculate w_k, the weight-level of a node's one-hop and two-hop neighbors, because a node does not know how many time slots are assigned to its neighbor in a certain time period. We present the method to solve this problem in the following.

We could broadcast the current weight-level of the node to its one-hop neighbors, and then relay to its two-hop neighbors, but this method incurs additional overhead.

A node knows how many packets are sent by its neighbor in a certain time period because all the information is broadcasted. A node can receive a packet from its neighbor (assuming perfect physical layer) even if it is not the receiver of the packet. A node may not know the number of time slots assigned to its neighbor, although it knows the neighbor's priority for each time slot, because it might not know the priorities of its neighbor's two-hop neighbors. Noticing that the channel is wasted when a node is assigned the channel but has nothing to transmit, we can use these wasted time slots to transmit a Nothing-to-Transmit packet, which can be viewed as an empty packet. When a node is assigned to use the channel, all its one-hop neighbors can receive a data packet or a Nothing-to-Transmit packet. Thus, a node can learn the number of time slots assigned to any of its neighbors simply by adding all the data packets and Nothing-to-Transmit packets it receives.

C. Detailed Description

A node exchanges weight-level information with its neighbors by sending Nothing-to-Transmit packets. Fig. 4 shows the time division in our protocol.

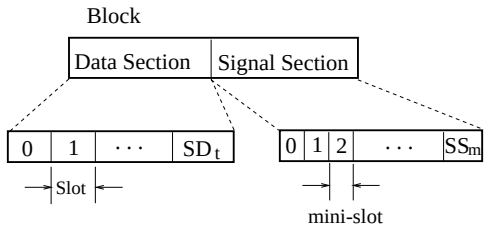

Fig. 4. *Time Division in HBS Protocol*

A slot and a mini-slot are the basic time units in our protocol. A data section contains SD_t time slots and a signal section contains SS_m mini-slots. A block is composed of one data section and one signal section. Fig. 5 shows the structure of the data frame format in a slot of a data section, and Fig. 6 shows the structure of the empty frame format (Nothing-to-Transmit packet) in a slot of a data section. Fig. 7 shows the structure of the signal frame format in a mini-slot of a signal section.

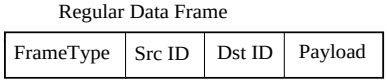

Fig. 5. *Structure of the Data Frame Format in a Slot of a Data Section*

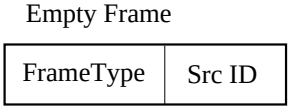

Fig. 6. *Structure of the Empty Frame Format in a Slot of a Data Section*

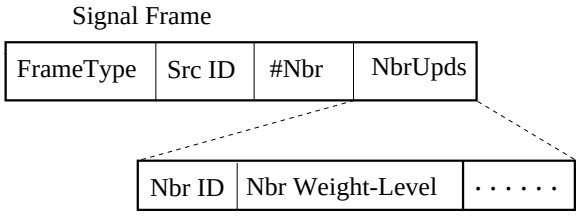

Fig. 7. *Structure of the Signal Frame Format in a Slot of a Data Section*

The rule for deciding who sends what in a signal section is shown in Fig. 8. It can be noticed that the algorithm of signal packet sending is very different from data packet sending. Whether a node is qualified to send a signal packet does not require computing the priority of its contenders, so collisions may happen in signal sections. However, since the same information about a node and its neighbors will be sent several times, a few collisions will not impede the nodes getting correct information of its contenders in time. Fig. 9, and Fig. 10 show the detailed algorithm of the history-based scheduling (HBS) protocol.

```
Compute the priority of node i
      p(i,t) = Rand(i,t);
if (node i is qualified to send a signal,
  i.e., p(i,t)modN = N − 1)
  send a signal packet, containing information
  of node i and its neighbors;
end if;
```

Fig. 8. *Algorithm of Channel Access in a Signal Section*

```
Initialize weight-level and all counters;
while forever
   if (the current time is a data section)
     perform the packet sending in a data section;
     perform the receiver's algorithm for a time slot;
   else
     if (the current time is the first mini-slot)
       update the weight-level of this node and all
       its neighbors;
     perform a packet sending in signal section;
     perform the receiver's algorithm for a mini-slot;
     end if;
   end if;
end while;
```

Fig. 9. *Sender's Algorithm*

D. Correctness

If the nodes in the network have correct and up-to-date knowledge of their two-hop neighbors, HBS achieves the goals of collision-free, safety and liveness.

The contenders have mutual knowledge and their local time t is synchronized. Because the pseudo-random function $Rand(i,t)$ generates unique results with the same i and t, the priority numbers

```
Wait until current slot/mini-slot ends or a packet arrives;
if (a packet arrives)
  case (packet type):
    data packet:
        increase both the time slot got counter and
        packet sent counter for the sender;
        if (it is a unicast packet & the receiver is
          not the current node)
          discard this packet;
        end if;
        pass the packet to the network layer;
    signal packet:
        update the information from this packet;
    NT packet:
        increase the time slot got counter for the
        sender of the packet;
  end case;
end if;
```

Fig. 10. *Receiver's Algorithm*

$p(i,t)$ of the nodes are consistent at every time slot t. When node k has the highest priority among its one-hop and two-hop neighbors at time slot t, all of its neighbors will have the priority information about node k, and k will be the only node to access the channel in its two-hop range during time slot t. Thus, no collision will happen during any time slot. The Nothing-to-Transmit packet is only sent out when the channel is assigned to the node and all its contenders will not send any packets. So the NT packet does not create any collisions.

Because there is a maximum and minimum weight level to limit the node's weight range, no node can occupy the channel all the time (unless there are no other nodes in the node's two-hop range) and all nodes have a channel access probability greater than zero. Hence, no starvation occurs in HBS.

Because a node has a finite number of contenders in the network, HBS can always generate one or more channel winners for each time slot. Due to the collision-free property of HBS, the nodes who win the channel can send data packets without collisions. Thus, HBS allows the live utilization of the channel.

III. HBS Performance Analysis

A. Data Packet Arrival Model

It is difficult to accurately model the data packet arrival of the MAC layer in the real world because the arrival pattern differs very much with different applications. In our model, the nodes in the network have two states, *active and quiet*. The data packets arrive at a high rate λ_1 when the node is in *active* state, and no data packet arrives when the node is *quiet*. When a *job* arrives at a node, the status of the node becomes *active*, and the status switches from *active* to *quiet* when a *job* ends. The *jobs* arrive exponentially at a rate λ_J and the job lengths are exponentially distributed with parameter μ_J.

Given the arrival rate λ_1 of the *active* state packets, λ_J, and μ_J, we have that the overall data packets arrival rate λ equals:

$$\lambda = \lambda_1 \cdot \frac{\frac{1}{\mu_J} - 1}{\frac{1}{\lambda_J} + \frac{1}{\mu_J} - 2} \tag{1}$$

Let q be the average channel access probability of a node. Due to the overall fairness of all the nodes in NAMA and HBS, we have

$$q = \frac{1}{N+1} \tag{2}$$

where N is the number of two hop neighbors of each node.

B. Throughput

Since both NAMA and HBS are collision free, the channel can serve the load up to the maximum channel capacity in a fully-connected network. Thus, the throughput of the channel is the sum of overall arrival rates of the nodes when the arrivals and departures keep equilibrium on all nodes, and the channel throughput remains at the maximum channel capacity when the load exceeds the channel capacity. The throughput of the channel S can be described by the overall arrival rate λ_i and channel access probability q_i of node i:

$$S = \sum_i \min(\lambda_i, q_i) \tag{3}$$

C. Data Packet Delay

A node has a probability q of accessing the channel, and because of the randomness of NAMA and HBS, the period of time that a node needs to wait (the unit for time is a time slot) to get access to the channel is a geometric distribution Y with parameter $1-q$. Therefore, the mean and second moments of queueing time $\bar{Y}$, $\bar{Y^2}$ are given by

$$\bar{Y} = \frac{1}{q} - 1 \tag{4}$$

$$\bar{Y^2} = \frac{2-4q}{q^2} \tag{5}$$

Accordingly, the service time X for sending a data packet is $Y+1$, and its mean $\bar{X}$ and second moments $\bar{X^2}$ equal

$$\bar{X} = \frac{1}{q} \tag{6}$$

$$\bar{X^2} = \frac{q^2-2q+2}{q^2} \tag{7}$$

If the arrival of data packets is Poisson, we can obtain the average queueing time from the extended Pollaczek-Khinchin formula [5], [4],

$$W = \frac{\lambda(q^2-2q+2)}{2q(q-\lambda)} + \frac{1}{2} \tag{8}$$

Here we present how we apply Eq.8 to the analysis. For the first data packet of a *job*, we can approximate the Poisson arrival queueing time W as the queueing time of the first data packet in a *job*. However, for other data packets in a *job*, this approximation does not hold. When the node is in *active* status, the data packet arrival rate is λ_1, which is much higher than the overall data packet arrival rate λ. At this time, the delay for a data packet will be one of the two following cases:

1) If the arrival rate is lower than the service rate, i.e., the node's channel access probability, we can use the extended Pollaczek-Khinchin formula to estimate the data packet average delay D by

$$D = \frac{\lambda_1 q + 2(1+q)}{2(q-\lambda_1)} + \frac{3}{2} \tag{9}$$

2) If the arrival rate λ_1 in *active* state is higher than the service rate q, the Pollaczek-Khinchin formula is not valid. In this case, we assume that the queue is not empty when the packets arrive after the first packet of a *job*. Then the delay of the nth data packet in a *job* equals

$$\begin{aligned} D_1 &= W + \tfrac{1}{q} \\ D_n &= D_{n-1} - \tfrac{1}{\lambda_1} + \tfrac{1}{q}, \quad n > 1 \end{aligned} \tag{10}$$

and the average delay of the first n data packets in a *job* is

$$\bar{D}_n = \frac{n}{2q} - \frac{n}{2\lambda_1} + D_1 \tag{11}$$

Because the length of a *job* follows a geometric distribution with parameter $1-\mu_J$, we have that the data packet average delay D is

$$D = \Sigma_{n=1}^{\infty} \mu_J (1-\mu_J)^{n-1} \cdot D_n \tag{12}$$

Substituting Eq.10 and Eq.11 into Eq.12, the average delay for data packets equals

$$\begin{aligned} D = &(\tfrac{1}{\mu_J} + 1) \cdot (\tfrac{1}{2q} - \tfrac{1}{2\lambda_1}) \\ &+ \tfrac{\lambda q + 2(1+q)}{2(q-\lambda)} + \tfrac{3}{2} \end{aligned} \tag{13}$$

For NAMA, the channel access probability of a node keeps constant, so the average system delay of data packets is the same as Eq. 12

$$\begin{aligned} D_N = &(\tfrac{1}{\mu_J} + 1) \cdot (\tfrac{1}{2q} - \tfrac{1}{2\lambda_1}) \\ &+ \tfrac{\lambda q + 2(1+q)}{2(q-\lambda)} + \tfrac{3}{2} \end{aligned} \tag{14}$$

For HBS, a node's overall probability to access the channel remains the same, so the delay time of the first data packet in a *job* is the same as in NAMA. However, a node in HBS can update its weight, so that it can increase its channel access probability when it is in *active* status. Let qh be the average channel access probability of a node when it is in *active* status. The average system delay of data packets for HBS is

$$\begin{aligned} D_H = &(\tfrac{1}{\mu_J} + 1) \cdot (\tfrac{1}{2\mathbf{qh}} - \tfrac{1}{2\lambda_1}) \\ &+ \tfrac{\lambda q + 2(1+q)}{2(q-\lambda)} + \tfrac{3}{2} \end{aligned} \tag{15}$$

In HBS, **qh** is usually $1.2 \sim 1.5$, depending on the number of two-hop neighbors of a node and the activity of its neighbors. Fig. 11 shows the results of the above analytical model for two different values of qh. As expected, HBS shows smaller average delay than NAMA.

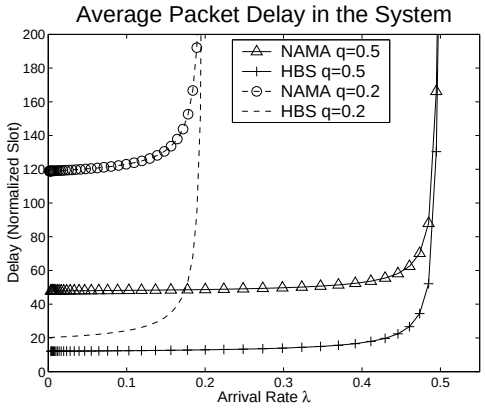

Fig. 11. *Comparison of Average System Delay*

IV. Simulation Results

We study the performance of HBS and compare it against the performance of NAMA and CSMA/CA by simulations using two scenarios: fully-connected networks and multihop networks. To compare the performance of the channel access protocols only, the network topologies are kept static during the simulations. In CSMA/CA, the duration of a time slot is the channel round-trip propagation delay, while in HBS the time slot is the length of a complete data packet. When we do the comparison, we use the normalized time slot in CSMA/CA, which contains several time slots of CSMA/CA and has a length equal to a data packet length. We run the simulations with the following behaviors and parameters:

- Signals propagate in free space and all the nodes have the same radio transmission range in all directions.
- The time unit in simulation is one time slot in the protocol. and the length of one time slot is 8 ms.
- The maximum bandwidth for a radio transmission is 2 Mbps, enough for a 2 Kb packet transmission.
- The traffic generated at each node has two types: telnet model and FTP model. Both models generate transfer requests at random times, and the file length in an FTP transfer is of random size.
- The simulation duration is 2560000 time slots (20480 seconds) for the fully- connected scenario, and 640000 time slots (5120 seconds) for the multihop network scenario. They are long enough to get simulation results in a steady state.
- In HBS, the minimum weight-level is 2 and the maximum weight-level is 4.

A. Fully Connected Scenario

We have four configurations in our fully-connected simulation: networks of 2, 5, 10, and 20 nodes. Fig. 12 compares the delay of CSMA/CA, NAMA and HBS in networks of 2,5,10, and 20 nodes under different arrival loads, while Fig. 13 compares the throughput of CSMA/CA, NAMA and HBS in 2-, 5-, 10- and 20-node netwroks.

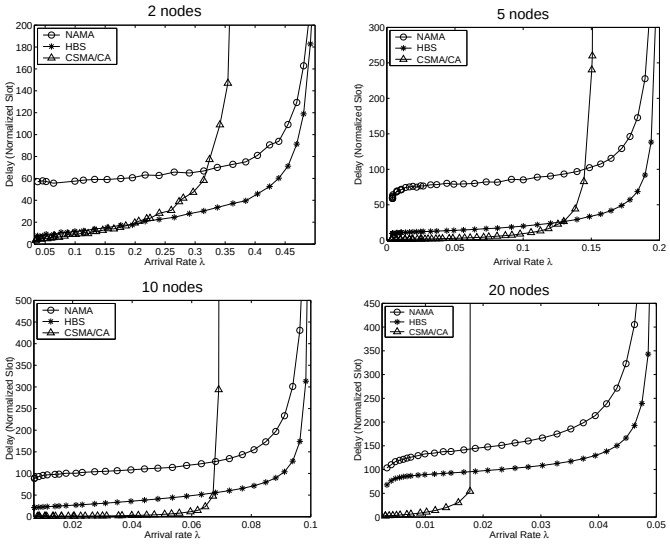

Fig. 12. *Average Packet Delay in Fully-Connected Networks*

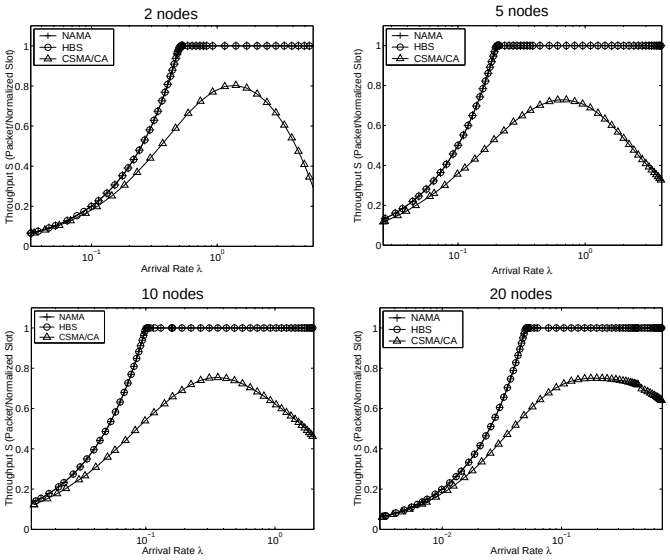

Fig. 13. *Average Throughput in Fully-Connected Networks*

From Fig. 12, we observe that HBS has lower delay than NAMA in fully-connected scenarios, as predicted in Eq. 14 and Eq. 15. When the arrival rate is very low, CSMA/CA has lower delay than NAMA and HBS. However, when the arrival rate λ reaches some point (still low arrival rate), the delay of CSMA/CA increases faster than HBS and NAMA due to the high probability of collisions. Fig. 13 illustrates that HBS has higher throughput than CSMA/CA. Because HBS and NAMA use the activation node scheme, they have the same throughput in fully-connected networks, which is shown in Fig. 13.

B. Multihop Scenario

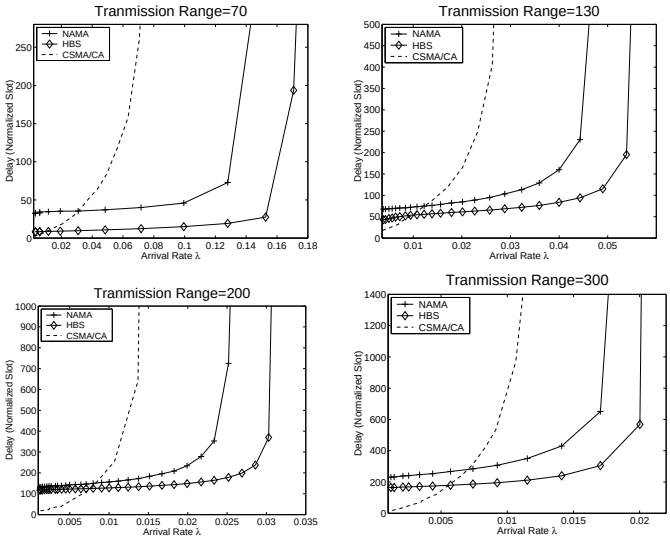

Fig. 14. *Average Packet Delay in Multihop Networks*

In the multihop scenario, 50 nodes are randomly placed in a 700x700 square-meter area. We consider different topologies and contention levels by setting the transmission range of each node to 70, 130, 200, and 300 meters. Fig. 14 shows the delay of NAMA,

HBS and CSMA/CA for the different transmission-range settings. As in the fully-connected network, HBS has lower delay than NAMA. Although CSMA/CA has the lowest delay at very low arrival rates, its delay becomes larger than NAMA and HBS at medium or high loads. Throughput of NAMA, HBS and CSMA/CA in different transmission range settings are shown in Fig. 15. Because of node activation scheme, both HBS and NAMA have higher throughput than CSMA/CA, especially at high loads.

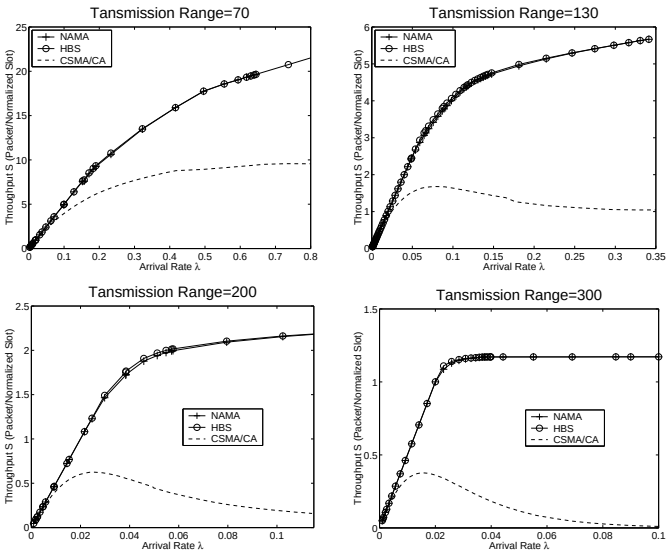

Fig. 15. *Average Throughput in Multihop Networks*

V. Conclusions

In this paper, the history-based scheduling (HBS) protocol was presented, verified and analyzed. Because it changes node weights dynamiccally based on the history of the nodes' transmissions, HBS can achieve lower delays than NAMA. By preventing collisions, HBS has much better performance than CSMA/CA, and the same throughput as NAMA.

References

[1] N. Abramson. The ALOHA system - another alternative for computer communications. In *Proc. of the Fall Joint Computer Conferernce*, pages 281–285, 1970.

[2] L. Bao and J.J. Garcia-Luna-Aceves. A new approach to channel access scheduling for Ad Hoc networks. In *Proceeding ACM MobiCom*, Rome, Italy, July 2001.

[3] L. Bao and J.J. Garcia-Luna-Aceves. Hybrid channel access scheduling in Ad Hoc networks. In *Proc. IEEE Tenth International Conference on Network Protocols*, Paris, France, November 2002.

[4] Vladimir V. Kalashnikov. *Mathematical Methods in Queuing Theory*. Kluwer Academic, Norwell, MA, 1994.

[5] Khinchin and Aleksandr IAkovlevich. *Mathematical Methods in Theory of Queuing*. New York, Hafner Pub. Co., 1960.

[6] L. Kleinrock and F. Tobagi. Packet switching in radio channels. Part I. carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE Transactions on Communications*, COM-23(12):1400–16, December 1975.

[7] R. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wireless Networks*, 5(2):81–94, 1999.

[8] Tanenbaum. *Computer Networks*. Prentice Hall, 1996.

[9] Y. Wang and J. J. Garcia-Luna-Aceves. Performance of collision avoidance protocols in single-channel Ad Hoc networks. In *Proc. of IEEE Tenth International Conference on Network Protocols (ICNP)*, Paris, France, November 2002.

[10] Y. Wang and J. J. Garcia-Luna-Aceves. Channel sharing among competing flows in Ad Hoc networks. In *Proc. of IEEE Symposium on computers and communications (ISCC)*, Kemer - Antalya, Turkey, June 30 - July 3 2003.