Time Division Hashing (TDH): A New Scheduling Scheme for Wireless Ad-Hoc Networks

Winnie Cheng, I-Ting Angelina Lee, Neha Singh

MIT Computer Science and Artificial Intelligent Laboratory

32 Vasser Street, Cambridge, MA 02139

Phone: (617) 253-5851, Fax: (617) 258-8682, Email:{wwcheng, angelee, nsingh }@mit.edu

Abstract

The current 802.11 MAC protocol uses RTS-CTS-DATA-ACK packet exchange and exponential backoffs to prevent collisions at both the sender's side and the receiver's side. While the protocol works well in prevention of collisions, it degrades throughput due to a significant amount of overhead produced by RTS/CTS message exchange. In addition, aggressive backoffs used in current 802.11 MAC protocol can lead to long channel idle time, resulting in the channel under-utilization. In this paper, we propose a new scheduling scheme, TDH, for ad hoc wireless networks which minimizes the chance of collisions by eliminating RTS/CTS message exchange and the use of backoffs. We also study and compare the performance of our proposed scheme with the performance of MAC RTS/CTS scheme under different types of topologies. Our analysis leads to the conclusion that, our scheme has higher throughput in many cases compared to MAC RTS/CTS scheme, however, at the expense of a slightly higher collision rate.

I. INTRODUCTION

Shared media is a fundamental problem in the wireless network. Nodes within certain range can interfere with each other's communication. A substantial amount of research has been done to address this issue, and most working schemes are based on Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA). Current 802.11 Media Access Control protocol, for instance, uses RTS-CTS-DATA-ACK message exchange to establish connection between two nodes and prevent other nodes within range from transmitting and causing interference. When a node tries to establish a connection with another node and fails (either due to busy media or no CTS response from the other node) or when a collision happens, exponential backoff is performed. While it works well in preventing collisions at both the sender's side and the receiver's side, there are downsides to the scheme that can sometimes cause more harm than benefit and decrease overall throughput. For instance, the RTS/CTS message exchange can generate a lot of overhead – ideally, this should not be the case since RTS/CTS packet is only 30 bytes each in length. However, in reality, RTS/CTS packets are sent at much slower rate than data packets (usually 10 times or 11 times slower compared to data packets), and thus, cost a substantial amount of overhead, especially when the data packet sizes are small. Furthermore, while aggressive backoff provides a mechanism for channel contentions, it sometimes causes long channel idle time (when multiple nodes decide to backoff simultaneously) or unfairness with transmitter-based contention (i.e. one transmitter gets considerably higher bandwidth compared to other transmitters).

These fundamental problems within MAC RTS/CTS scheme motivated the design of our proposed scheme – TDH, Time Division Hashing. Our proposed scheme is based on dividing time into fixed-length slots and hashing these slot values to determine whether the node is in 'send' or 'receive' mode. Each node generates its own random but deterministic sending/receiving schedule by hashing with a pseudo random seed (details of the algorithm are described in section III). That means, as long as the random seed is known, a node can deterministically predict when any other node will be sending or receiving. While our proposed scheme does not prevent all types of collisions, it generates better overall throughput in many cases because there is no overhead associated with RTS/CTS message exchange or idle time caused by exponential backoffs. Furthermore, our scheme leads to better fairness within the network traffic, i.e. each node gets approximately fair share of bandwidth for transmission.

This paper is organized in the following manner: in section II, we briefly discuss some related work done previously and explain our motivations for designing the new scheme. In section III, we describe our proposed scheme, including pseudo code and how the algorithm works. In section IV, we present our simulation results of comparisons between the two schemes. Finally, in section V, we summarize and

conclude our results from the simulation, and propose possible future work.

II. MOTIVATIONS AND RELATED WORK

Some of the early work in Media Access Control schemes for wireless networks was based on the Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) technology. An example of such a scheme is Multiple Access with Collision Avoidance (MACA) [6]. Such schemes try to address the issues specific to wireless networks such as congestion at the receiver, collisions due to hidden terminal problems and inefficiency due to exposed terminals. To solve some of these problems, they use the RTS/CTS handshake mechanism to perform 'Virtual Carrier Sensing' and establish a connection between the two communicating nodes. Once the RTS/CTS messages have been exchanged, the communicating nodes cannot participate in any other transmission for the length of time specified in the control messages. In addition, neighboring nodes (within receive range) of the two communicating nodes also cannot transmit during that time.

These early schemes suffer from inherent problems such as unfairness and long delays during traffic congestion. This led to development of improved backoff algorithms and contention window management schemes such as MACAW [3], Estimation-based Backoff [5], Adaptive Backoff [4], Weighted Hierarchical Backoff [9] and Dual Stage Contention Resolution [16]. There are other approaches that borrow ideas from fair queuing such as Distributed Fair Scheduling [15] and fair-share estimates [2]. These schemes often require great implementation complexity to achieve reasonable fairness.

Other than unfairness and congestion, the RTS/CTS mechanism has some more fundamental problems. Firstly, it is ineffective in detecting hidden terminals that are present outside of the receive range, but in each other's interference range. Virtual carrier sensing does not work in more realistic interference models, where the interference range is usually two or more times greater than the receive range . Another reason for performance degradation can be inefficiency caused by exposed terminals not being able to transmit simultaneously. Two neighboring nodes should be able to transmit at the same time as long as their receivers are not neighbors, but RTS/CTS messages prevent this from happening as no node that has heard an RTS from a neighbor will transmit until the first transmission is complete. These and other issues have been analyzed in some recent work that is aimed at evaluating performance of the RTS/CTS mechanism ([10], [13]).

Another problem with RTS/CTS as implemented in the 802.11 standard is the overhead caused by sending these control messages at a much lower bit rate than the rate at which data is sent [12]. This leads to observed capacities being significantly less than optimal even for simple chain and lattice networks [7]. The aggressive backoff mechanism can also cause long idle times [7]. Backoff occurs in 802.11 when two RTS messages collide. It also happens in the event of a data collision, which is detected by the sender upon not receiving acknowledgement for the packet that it transmitted.

To demonstrate some of the weaknesses of the RTS/CTS mechanism, we performed some simulations is *ns*. The first result (Fig. 1) shows the effect of the RTS/CTS control messages on the overall throughput.

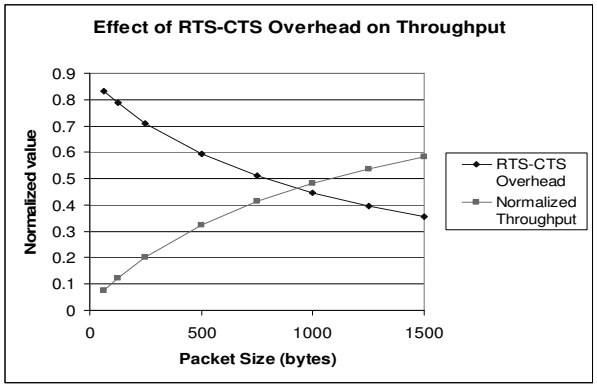

Figure 1. Effect of RTS-CTS Overhead on Throughput

The scenario above consists of two nodes, with one sending to the other. The normalized throughput decreases as the packet size decreases and reaches a value below 10% for a packet of size of around 64 bytes. The overhead is calculated as the fraction of the total transmission time taken up by the RTS/CTS messages. It can be seen from this result that the RTS/CTS overhead is significant, especially for small packet sizes. The normalized throughput is directly affected by this overhead.

The second experiment demonstrates the performance of RTS/CTS in a topology with interference. Each pair of nodes in Fig. 2 indicates a flow. The carrier sensing range is indicated by the circles, so none of the pairs directly hear other. But the interference range of the nodes is approximately twice the carrier sense range.

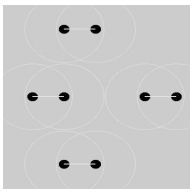

Figure 2. Interference Topology

The results of measuring throughput as more such pairs are added are shown in the graph below (Fig. 3). In this case, RTS/CTS is not helpful in avoiding

collisions at the receiver. The throughput drops to less than 10% when there are 2 or more pairs of interfering nodes. Disabling RTS/CTS does not help much either.

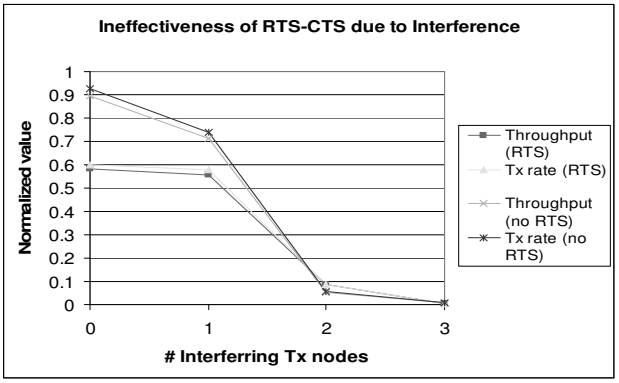

Figure 3. Ineffectiveness of RTS-CTS due to Interference

To address some of the above weaknesses of 802.11 caused by the RTS/CTS mechanism, we propose a new scheduling scheme, Time Division Hashing (TDH), for the MAC layer of 802.11. This scheme is based on the clock exchange scheduling algorithm for Packet Radio Networks proposed by Tim Shepard [11]. Tim's thesis addresses the problem of designing a scalable multihop packet radio network that manages transmissions of spread spectrum radios. He analyzes propagation and interference models for local interference as well as the overall noise in the system, and proposes a scheme for scheduling packet transmissions to avoid collisions caused by interference from nearby stations without the need for global coordination or synchronization. TDH is based on this idea of forming deterministic schedules without the need for global coordination. However, Tim's model cannot be used as such for the wireless networks using 802.11 because he makes some assumptions in deriving his results that are not true for current wireless networks in general. In his scheme, collisions at the receiver are avoided by the use of multiple receive channels and code division multiplexing. Most of today's wireless radios do not use this spread spectrum technology and therefore, we still have to face the problem of collisions at the receiver. Also, his experimental results are derived with the assumption that minimum energy routing is used for routing packets in multihop networks. This assumption also does not hold for today's wireless networks in general.

There has been other work in designing a better MAC scheduling scheme for wireless networks than the RTS/CTS mechanism. [14] proposes a topology-dependent transmission scheduling scheme, called collision-avoidance time allocation (CATA). CATA allows nodes to contend for and reserve time slots by means of a distributed reservation and handshake mechanism. This scheme still suffers from the overhead of the handshake required for data transmission, which TDH is trying to avoid. [1] proposes a scheme called Neighbor-aware Contention Resolution (NCR), which generates a permutation of the contending members, the order of which is decided by the priority of all participants. This scheme relies on the availability of local topology information within at least two hops. Our work is different from these and other similar schemes because it tries to solve the scheduling problem with minimum overhead of control messages by having a deterministic schedule for a node that is known to all its neighbors. It tries to maximize throughput by probabilistically picking the time slots in which a node will be sending and receiving. We also simulate our scheme on a variety of different topologies, including some that are more realistic than the ones used for deriving results in most other schemes.

III. THE TDH ALGORITHM

Time Division Hashing (TDH) is a decentralized media access scheme that is based on each node in the wireless network having a deterministic schedule. Time is divided into fixed-length slots. Each slot is meant for transmitting a single packet. The schedule is represented in terms of the node being in either 'send' state or 'receive' state in each of these slots. Slots of either kind are picked according to some fixed probability. The mechanism for the nodes to come up with a schedule is as follows. Each node in the network maintains a value called the seed, which is randomly picked. Each node's schedule is offset by this random value, and that is how each node has a different schedule. To determine the state of a slot, the value of the time slot is offset by the seed and is then hashed. Depending on the fixed transmit probability value, the value returned from the hash function is either chosen to be a 'send' slot or a 'receive' slot. Fig 4 shows an example schedule for a few nodes as returned from this algorithm.

The power of this scheme lies in the fact that a node can compute its neighbor's schedule by knowing just one value, which is the seed. Each node needs to know the seed values of all its neighbors. Using this seed, it can determine whether its neighbor is in 'send' or 'receive' mode using the method above. The problem of scheduling now reduces to finding a slot in which the sender is in 'send' mode and the receiver is in 'receive' mode. For example, in Fig 4, if node 1 gets a packet destined for node 2 at time 0, the first available slot it would pick is slot 2 which is a 'send' slot for node 1 and a 'receive' slot for node 2.

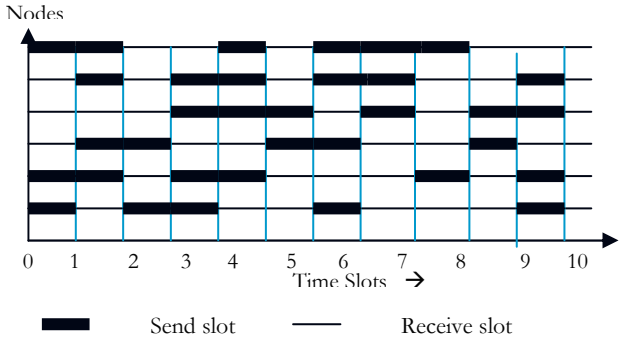

Figure 4. Example of a schedule for 6 nodes

The most important parameter to this algorithm is the 'transmit probability' *p* with which slots are picked to be transmission slots or reception slots. The optimal value of this parameter is dependent on the topology that the algorithm is being used for and on the traffic patterns. For some simple topologies, the optimal value of *p* that will maximize throughput can be derived theoretically. For more complicated scenarios, it is hard to get the best value for *p* theoretically. In such cases, a good value can be arrived at by estimation. In our scheme, all nodes share the same value of *p*. However, it is also possible to envision a scheme in which each node has its own *p* depending on its own traffic pattern. So if a node is going to be sending a lot of traffic, it should pick a high value of *p* so that it will maximize its chances of finding a match for a receive slot.

The second important parameter to the algorithm is the length of a time slot. In the TDH scheme, collisions are still possible at the receiver. This is because two nodes that are in 'send' mode in the same time slot could simultaneously schedule a packet for transmission to a third node that is in 'receive' mode in that slot. This is a side-effect of the fact that the scheme has been designed such that a sender only needs to know the receiver's schedule in order to transmit a packet. Even if a sender checked all its neighbors' schedules, collisions due to hidden terminals could not have been avoided. Therefore, it is required to detect collisions in this scheme and acknowledgements are necessary. The time slot has been designed so that both the packet and the ACK can be transmitted in one slot. The slots are fixed-size and are long enough to transmit a packet of the maximum size. Also since transmission and reception are being done in the same slot, it is necessary to switch the wireless radios between the two modes twice in the slot. So the length of a slot can be calculated with the following formula:

$$\begin{aligned} \textit{slot length} = {} & \textit{time to transmit packet} \\ & + \textit{time to transmit ACK} \\ & + 2 * \textit{(time to switch radio)} \end{aligned} \quad (1)$$

Fig. 5 outlines the TDH algorithm in pseudo-code. The first step in TDH is for each node to pick its seed *s*, which is a random value. The seeds are exchanged between all pairs of neighbors, so that a node knows the seeds of all its neighbors. The scheduler uses a few data structures to keep track of the state of the scheduled packets. *PrevTimes* is a table that keeps track of the time that the last packet for scheduled for each destination that the node has sent to. *ScheduledTimes* is a list that contains all the slots that are 'occupied', meaning that all the time slots beginning from the current time that already have packets scheduled to be sent in them.

Figure 5. TDH Algorithm

```
SchedulePacket (recvID, currrentTime) {
t = beginning of next time slot from currentTime ;
if ( PrevTimes(recvID) > t )
      then t = PrevTimes(recvID) + slotLength ;
s_recv = seed of receiver ;
loop {
            increment t till a slot is found that is not
   in ScheduledTimes;
            senderMode = hash ( s + t ) ;
            destMode = hash ( s_recv + t ) ;
            if ( senderMode <= p and destMode > p) {
                  Schedule packet for t ;
                  Update ScheduledTimes, PrevTimes ;
                  return t ;
            } else
                  t = t + slotLength ;
      }
}
```

The TDH algorithm is called to schedule a packet for a certain receiver at a certain time. The scheduler starts looking for a slot after the current time. It first checks to see if there is a packet already scheduled for that destination at a later time using *PrevTimes*. In this case, it only needs to start looking for a slot after that time. The algorithm then iterates on this time slot until it finds a suitable time. It first checks to see if the current slot has already been used by checking if it is in *ScheduledTimes*. If it is, then the current slot is incremented to the next slot. Once it finds a free slot, it finds out its mode in that slot by adding its seed to the current time and hashing that value. The hash function returns a value between 0 and 1. If the value returned is less than the 'transmit probability' *p*, then the node is in 'send' state in that slot. The same process is repeated for the receiver using its seed. If the hash function returns a value greater than *p* for the receiver, it means

the destination is in 'receive' mode at that time. If both these conditions are met, it is the ideal case and a packet can be scheduled for that time. If the result of the hash function is not less than *p* for the sender or if it is not greater than *p* for the destination, then the time is incremented to the next slot and the algorithm repeats the process until it finds a match.

As an optimization to the algorithm, if a node is in 'send' mode in a particular time slot but does not have any packets scheduled for that slot, it's receiver is turned on to listen to broadcast traffic. Also, for the initial seed exchange phase of the algorithm when all nodes pick their random seeds and transmit it to all their neighbors, a modification is possible that will eliminate the need for this exchange. A unique ID of the node such as its MAC address can be used as the seed value of the node. As long as the hashing function is good, this should work as well as picking random numbers. In this case, there would be no seed exchange phase needed as the nodes could just discover their neighbor's seeds from the MAC address on the packets sent by them.

Theoretical Analysis

As suggested previously, the transmit probability *p* affects the throughput of the network. Though the optimal value of p depends on the underlying topology and network traffic dynamics, the theoretical throughput can be derived analytically for the following multi-node topology to provide an upper bound on the achievable throughput. Assuming a topology in which one sender has the freedom to send to any of its *k* neighboring nodes. The maximum throughput is expressed as the probability of this sender being in 'send' state and the probability that at least one of its neighbors is in 'receive' state. That is,

$$T(p, k) = p\,(1 - p^k) = p - p^{k+1} \qquad (2)$$

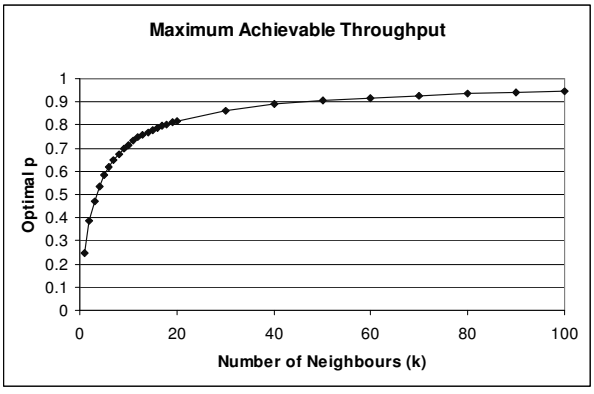

Figure 6. Theoretical Upper bound on Throughput

Implementation

To analyze the TDH scheme, we implemented and simulated it under NS-2. We built our scheme over the CMU Monarch protocol stack [8]. The algorithm described above is implemented as a standalone class, called 'TDHScheduler'. The MAC layer code is replaced with the new scheduling algorithm. RTS/CTS message exchange and use of backoffs is eliminated.

A few implementation issues are discussed below. Firstly, with our scheme, when a packet needs to be scheduled, it is not necessary that the time chosen for it will be later than the time the previous packet was scheduled for. Because of this, the MAC layer needs to get multiple packets from the link layer at once for TDH to work efficiently. For instance, node A might be in the 'send' mode during time slots 2, 3, and 4. Assume it gets a packet from the link layer which should be delivered to node B, and it schedules the packet to be sent at slot 3 because that is the time when node B happens to be in 'receive' mode. If node A gets more packets from the link layer, it is possible to schedule them for the free slots before 3 if they are meant for other destinations. Doing so maximizes the utilization of node A's transmission time. However, this also means that the MAC layer needs to be able to manage multiple outstanding packets at one instant. This includes buffering of the scheduled packets and managing their sending states (actual transmission time, status of outstanding acknowledgements, etc). Secondly, since we are buffering multiple packets at the MAC layer, flow control is required between the link layer and the MAC layer. Ideally, MAC layer would like to get as many packets as possible to maximize the utilization of its transmission time. However, it also needs to limit the amount of buffering since the latency seen by upper layers increases as more packets are buffered. As future work, it would be interesting to look into the relationship between the link layer latency and the amount of buffering at the MAC layer.

IV. SIMULATION RESULTS AND ANALYSIS

To analyze the performance of TDH, we compared the proposed scheme with 802.11 RTS/CTS using the network simulator NS-2. We studied 5 topologies in our simulation: *single sender, access-point, clique, chain* and *random topology*.

In these simulations, we used Constant-Bit-Rate (CBR) traffic with Dynamic Source Routing (DSR) as the underlying routing protocol. The data points are collected for simulation run time of 300s. The results are presented for channel bandwidth of 11Mbps and RTS/CTS transmission rate of 1Mbps. The interference range and carrier sensing range are set to 550m and

250m, respectively. Unless otherwise specified, the data packet size is 1500-bytes.

A. Single Sender Topology

This topology consists of N nodes in which one node is designated as the 'sender'. All nodes are within carrier sensing range of each other. The sender node creates N-1 CBR flows sent at equal rate to each of its neighbors. The aggregate rate of all flows is equal to the channel capacity. The transmit probability p is set to the optimum value according to eqn 2.

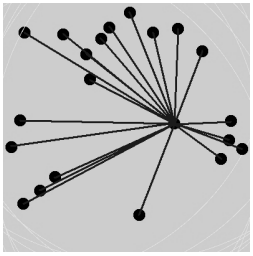

Figure 7. Single Sender Topology

Since there is only one sender, schedule collisions and interference are kept to a minimum. This should result in a throughput close to the theoretical maximum derived in Section III. Fig. 8 plots the observed normalized throughput against the theoretical throughput as a function of the total number of nodes in the topology. The observed throughput is defined as the sum of all received CBR bytes at the (N-1) CBR sink nodes normalized to the product of channel bandwidth and total simulation time. The experimental results are close to the theoretical values with slightly greater deviation as the number of nodes increases. This is due to increase in the overhead of routing messages.

The RTS/CTS throughput is shown in comparison with TDH in Fig 9. In this simulation run, RTS-CTS exchange is enabled. With 1500-byte data packet sizes and an 11:1 ratio in transmission rate, the expected overhead due to RTS/CTS is `11*(40+39)/[11*(40+39) + 1500+39]` ≈ 36% compared with the measured value of 42% (1 – 0.58 from the graph) which also includes the overhead of routing messages and protocol maintenance. The throughput is almost constant regardless of the number of nodes in the topology. The contention scheme of 802.11 is sender-centric where transmitting nodes compete for channel access. With the aggregate rate being constant and only one sender in this topology, the normalized throughput should therefore be constant. For TDH, on the contrary, as the number of nodes increases, there is a higher probability of matching the sender's transmit slot with any of the neighboring nodes' receive slot. This should have an impact on scheduling latency and throughput. When the number of nodes exceeds 10, TDH gives higher throughput than RTS/CTS.

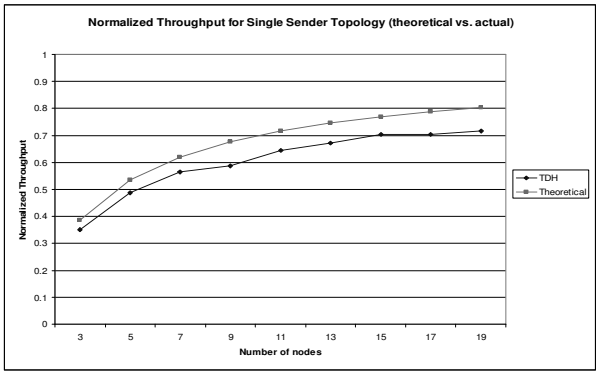

Figure 8. Comparison of Normalized Throughput with Theoretcal value in Single Sender Topology

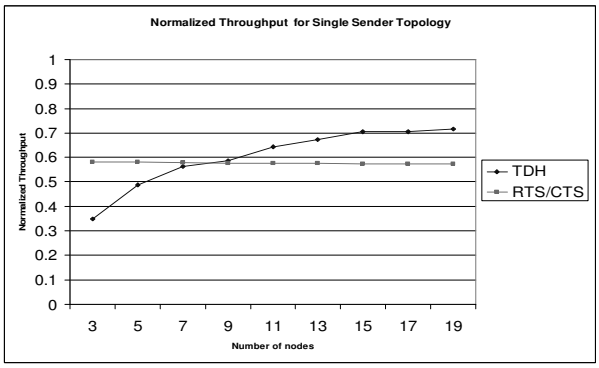

Figure 9. Comparison of Normalized Throughput with RTS/CTS for Single Sender Topology

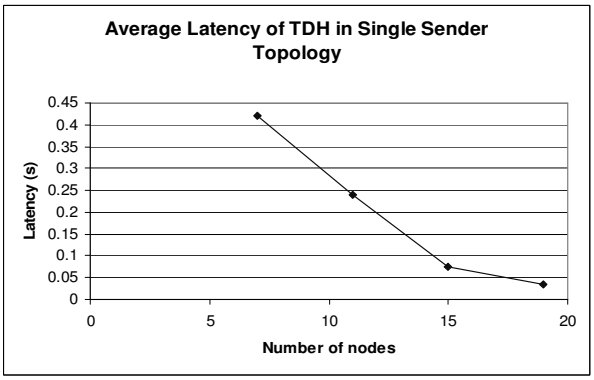

Figure 10. Average Latency in Single Sender Topology

From the latency plot in Fig. 10, it can be seen that average latency is decreasing as the number of nodes increases, though this includes some queuing delay. Here the latency is calculated as the difference between the time that the TDH algorithm is called to schedule a packet and the time that the packet is scheduled to be sent. The latency decreases because the time to find a match reduces when there are more receivers and more slots can be utilized.

B. Access-Point Topology

In this topology, we model an access point configuration where there is one node sending upstream with (N-2) downstream nodes receiving data from the access point. All flows have the same rate, and the aggregate transmission rate does not exceed the channel capacity. We investigate two questions with this topology:

- *How "fair" are the schemes in allocating per-flow access?*
- *How is throughput affected when RTS/CTS is enables/disabled?*

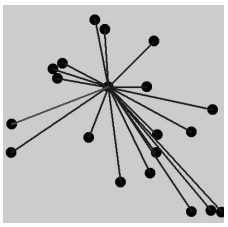

Figure 11. Access Point Topology. (Upstream flows in red. Downstream flows in blue)

In evaluating per-flow fairness, we use the fairness index *f* as our metric. It is defined as follows:

$$f = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

x_1 ,x_2.. x_i refer to the throughput of the i[th] competing flow.

TDH has a high fairness index between 0.9 and 1 as shown in Fig 12. It schedules all transmissions with equal probability, namely, `p*(1-p)`. In RTS/CTS, when there are 2 flows (one upstream and one downstream), the fairness index is 1 as these two senders use RTS to reserve the channel for their respective flows. However, as the number of downstream flows increases, there are still 2 senders. The access point sender is now responsible for more flows but from the channel's perspective, the competition is still between 2 senders. As a result, this causes significant unfairness to the downstream flows, in particular, when the number of downstream flows is small. As this number increases, the RTS sending rate of the access point also increases as a result of a higher aggregate transmission rate and the fairness index improves.

In practice, the RTS/CTS mechanism is sometimes disabled for access point configuration especially when the amount of upstream traffic is small. Fig. 13 illustrates the effect of disabling RTS/CTS on throughput in comparison with TDH and RTS/CTS enabled. The RTS/CTS throughput is almost identical to the throughput in Single Sender topology. This suggests that the extra sender causes very few RTS collisions. Hence, as observed, it is reasonable that the throughput is much higher when RTS/CTS is disabled.

Comparing TDH with 802.11 RTS/CTS, the graph is not too different from the Single Sender topology with the exception of an earlier crossover point. With the extra sender in this topology, when the number of nodes is small, the overall probability of a transmission occurring is increased as it is equal to the sum of the each of the sender transmitting, resulting in higher throughput.

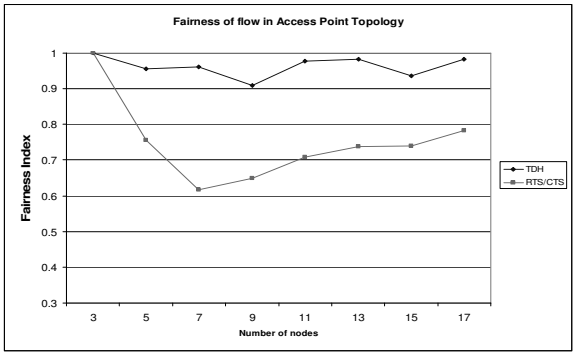

Figure 12. Fairness Indices for TDH and 802.11 RTS-CTS in Access Point Topology.

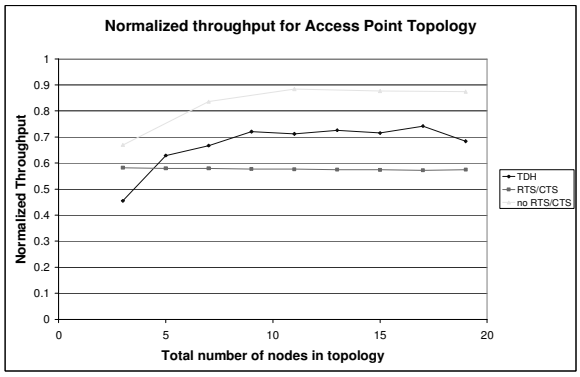

Figure 13. Comparison of Normalized Throughput for Access Point Topology.

C. Clique Topology

One of the drawbacks of TDH is that it does not have an explicit mechanism to avoid collisions from senders that have scheduled transmission in the same time slot. TDH uses a probabilistic approach to reduce the frequency of such events. The clique topology is devised to maximize the chances of such collisions. There are N nodes in carrier sensing range of each other. Each node acts as a sender of (N-1) CBR flows to its neighboring nodes. The aggregate flow rate of all senders is equal to the channel capacity.

The RTS/CTS mechanism avoids many of the collisions resulting in higher throughput than the other two schemes as shown in Fig. 15. Since the aggregate flow rate is set to the channel capacity, as the number of nodes increases, the total transmission rate of each node decreases by (1 / N). The per- flow rate thus decreases approximately by (1 / N^2). The RTS/CTS throughput drops only slightly because the increasing number of senders and associated RTS collisions are offset by the decrease in flow rate.

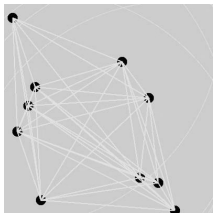

Figure 14. Clique Topology

Under the possibility of multiple senders colliding, TDH shows an average normalized throughput of about 0.45, around 10% less than that of RTS/CTS. One would expect that as the number of nodes increases, the probability of collision increases. Assume that the probability that a node has scheduled transmission in a 'send' slot is g. Then, for a clique with N nodes, the probability of collision in a given time slot is:

$$P(\text{collision}) = 1 - (1-g)^{N-1} \qquad (3)$$

$(1-g)$ is the probability of no transmission in a 'send' slot and therefore, $(1-g)^{N-1}$ is the probability that none of the other (N–1) nodes are also sending in that slot. Or put differently, the probability of successful transmission in a time slot diminishes with the power of (N-1). Therefore, one may expect a much sharper decline in the throughput as the number of nodes increases. However, from Fig. 15, we see that this is not the case. There is no sharp drop due to the following two reasons. First, from previous results, as the number of transmissions increases, the slots are utilized more efficiently. Second, in this topology, the channel is saturated but not oversubscribed as the per-flow rate is set such that the aggregate rate is close to the channel capacity. This means that the per-flow transmission rate decreases by 1 / N^2 as the number of nodes increases and therefore, a high number of collisions is not observed as shown in Fig 16. These two factors lead to a gradual decrease instead of a drastic drop in throughput, in a scenario where a large number of collisions are possible.

D. Chain Topology

It is important to examine multi-hop behavior in MAC protocols. As suggested by [7], interference several hops away can affect node-to-node transmission, and undesirable congestion can occur. The chain topology arranges nodes in a line. Successive nodes are spaced 200m so that only adjacent nodes are within

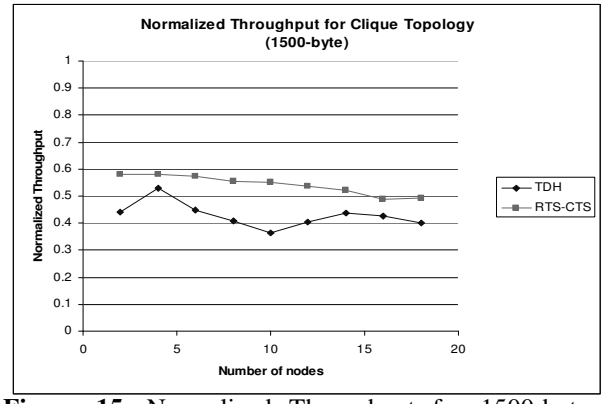

Figure 15. Normalized Throughput for 1500-byte packet size in Clique Topology.

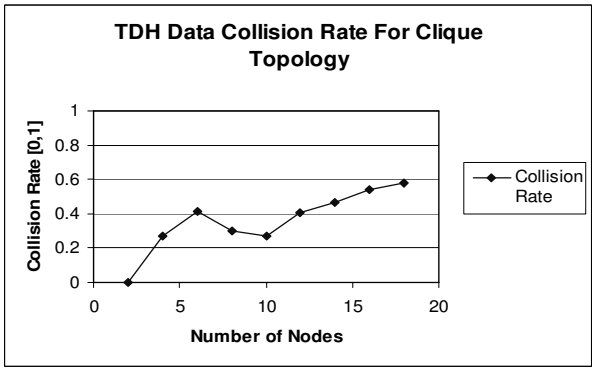

Figure 16. Data Collision Rate of TDH scheme in clique topology

Figure 17. Chain Topology

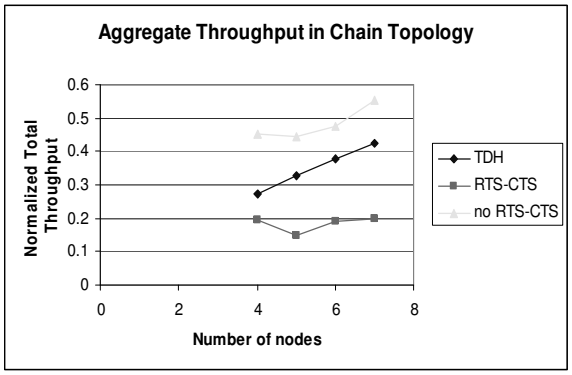

Figure 18. Aggregate Throughput in Chain Topology

carrier sensing range. The interference range, however, extends out to 550m. The transmit probability p is set to 0.5, the optimal value for two-node communication. The leftmost node sends at the maximum channel capacity rate to the rightmost node in the chain. All other nodes are passive participants in the forwarding path.

TDH suffers from having only one neighbor to try to match schedules with, so the node-to-node transmission rate is limited to 25%. Despite this, it is able to achieve a relatively high throughput compared to the RTS/CTS scheme. Fig. 18 shows the total throughput with the received bytes summed at all the nodes. RTS/CTS schedules transmissions over one in every three adjacent links. However, due to the larger interference range, collisions can happen between transmissions from nodes that are even two links apart. To achieve a collision free scenario, only one in four links should be used for transmission. With a maximum of 7 nodes in our topology, we expect no more than (2 * channel capacity). The aggregate throughput is normalized to this value in Fig. 18.

Ideally, in a perfect forwarding chain, the aggregate throughput should increase linearly with the number of nodes. TDH exhibits this trend with a small but steady slope. suboptimal schedules and long backoff in subsequent nodes.

With RTS/CTS disabled, it is able to push traffic through quite aggressively. On the contrary, the aggregate throughput of the RTS/CTS scheme is almost constant. As the number of nodes increases, transmission early on in the chain causes congestion in the middle of the chain which leads to suboptimal schedules and long backoff in subsequent nodes.

The chain throughput in Fig. 19 shows the result of the cumulative effect of interference and collisions at intermediate nodes. It is the observed throughput at the destination (rightmost) node. A quick examination of the TDH curve shows that extrapolating the line to a 2-node chain gives approximately 0.25 which is the throughput limit for node-to-node transmission.

The scheme with RTS/CTS disabled delivers the highest chain throughput. It is able to achieve a throughput of 30% for a 4-nodes 3-links topology, which is very close to 33% that can be obtained by using 1 out of the 3 links. For a 5-nodes 4-links topology, the value is around 22%, close to the expected 25%. However, this is at the expense of a high collision rate. The RTS/CTS disabled chain throughput curve shows the steepest gradient. As the number of nodes increases, the throughput drops significantly since the transmission of earlier nodes interferes with those later in the chain.

The throughput is worsened with RTS/CTS enabled, as it does not actually transmit data until the RTS/CTS messages are successfully exchanged. During the process of RTS/CTS exchange, several tries might be needed to reserve the channel and each unsuccessful try triggers an binary exponential backoff. This results in the channel being greatly under-utilized. Even if the channel is successfully reserved, data collisions can still occur due to interference.

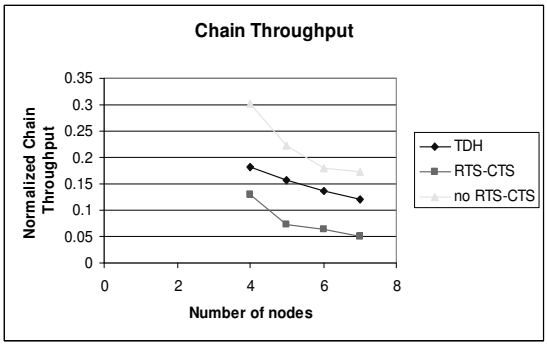

Figure 19. Chain Throughput

E. Random Topology

In an attempt to evaluate more realistic scenarios, the following random topology is generated. An area of 1000m by 1000m is divided into 16 squares. 10 nodes are randomly placed in each square, one of which is chosen as the sender. This sender creates some flows that are intended for nodes within the square and some intended for nodes in different squares. With this configuration, it reflects local traffic patterns as well as multi-hop behavior. The transmit probability p is set at 0.7 for all nodes.

The throughput of TDH turns out to be the highest compared to the other schemes as shown in Fig. 20 without RTS/CTS, many collisions occurred, significantly impacting throughput. However, with RTS/CTS, the situation is worse as RTS collisions triggered the backoff mechanism. About 75% of the transmission time, as illustrated in Fig. 21, is used for RTS/CTS transmissions. RTS collision rate is approximately ((54-19) / 54) ≈ 65% and therefore, the number of transmitted data packets is significantly lower than the other schemes. Time is wasted in leaving the channel idle as a consequence of backoff.

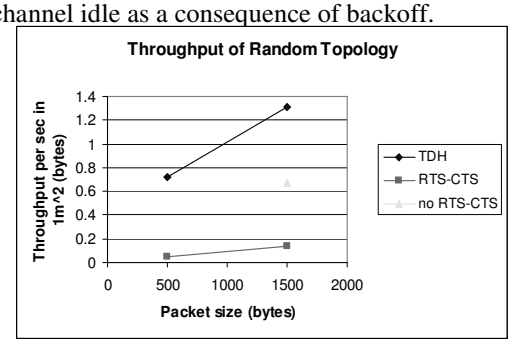

Figure 20. Throughput of Random Topology

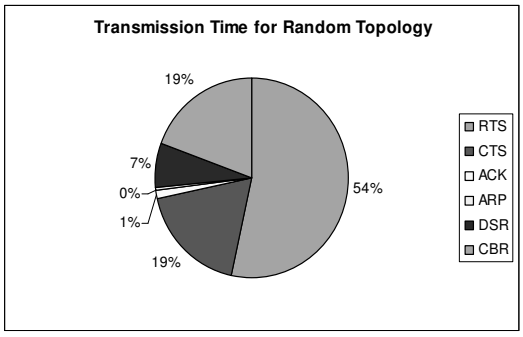

Figure 22. Breakdown of Transmission Time in the RTS/CTS scheme for Random Topology

V. CONCLUSIONS AND FUTURE WORK

The RTS/CTS mechanism in the 802.11 MAC layer incurs a lot of overhead, resulting in significant degradation of overall performance. The gain in throughput from collision avoidance does not always compensate for the harm caused by aggressive backoffs and the RTS/CTS overhead. Our proposed scheme is simple, does not cost additional overhead, and gives better performance in many cases, as the simulation results have shown. In addition, it has better fairness properties.

Some weaknesses of the TDH scheme we implemented are pointed out below. Firstly, the drop rate is high compared to the RTS/CTS scheme since TDH does not have a mechanism to avoid collisions on the receiver side. Secondly, performance degrades as the number of neighbors decreases. This is inherent in the algorithm because it gets harder for the sender to find a matching time slot (when the sender is in 'send' mode, and the receiver is in 'receive' mode) as there are less potential receivers. Also, we use static values for p (the percentage of time a node spends being in 'send' mode) and a fixed slot length in our simulations. This is less flexible and may not work too well with variable packet sizes.

Keeping these weaknesses in mind, there are some interesting topics to investigate and potential improvements that we can make to the TDH scheme. First, it will be interesting to investigate the impact of collisions on the transport layer, especially when the transport layer guarantees reliable and in-order delivery. Second, we can replace the fixed-length slots with multiple sub-slots to achieve more flexible time slot management when dealing with variable packet sizes. Third, we would like to experiment more and see how TDH performs under different network parameters (such as higher bit rate, delay and channel loss rate). Finally, machine learning techniques can be applied to dynamically assess and adjust the p value to optimize performance for different network topologies and traffic patterns.

REFERENCES

[1] L. Bao and J.J. Garcia-Luna-Aceves. "Distributed Dynamic Channel Access Scheduling for Ad Hoc Networks". In JPDC, Special Issue on Wireless and Mobile Ad Hoc Networking and Computing, 2002.

[2] B. Bensaou, Y. Wang, C.Ko. "Fair medium access in 802.11 based wireless ad-hoc networks". International Conference on Mobile Computing and Networking, 2000.

[3] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang,. "MACAW: A Media-Access Protocol for Packet Radio". In Proceedings of ACM SIGCOMM, 1994.

[4] F. Cali, M. Conti, and E. Gregori. "IEEE 802.11 Protocol: Design and Performance Evaluation of an Adaptive Backoff Mechanism". IEEE Journal On Selected Areas In Communications, Vol. 18, NO. 9, September 2000

[5] Z. Fang, B. Bensaou, and Yu Wang. "Performance evaluation of a fair backoff algorithm for IEEE 802.11 DFWMAC". In Mobile Computing and Networking, pages 48-57, 2002.

[6] P. Karn, "MACA – A New Channel Access Method for Packet Radio", ARRL/CRRL Amateur Radio 9th Computer Networking Conference, September 1990.

[7] J. Li, C. Blake, D. De Couto, H. Lee, R. Morris. "Capacity of Ad Hoc Wireless Networks". In the Proceedings of the 7th ACM International Conference on Mobile Computing and Networking, July 2001.

[8] CMU Monarch Group. CMU Monarch extensions to *ns:* http://www.monarch.cs.cmu.edu

[9] T. Ozugur. "Weighted Hierarchical Backoff Algorithm for Wireless Ad Hoc Networks". IEEE Global Telecommunications Conference, 2001.

[10] K. Xu, M. Gerla. "Effectiveness of RTS/CTS Handshake in IEEE 802.11 based Ad Hoc Networks". Mobicom 2003.

[11] T. Shepard. "Decentralized Channel Management in Scalable Multihop Spread Spectrum Packet Radio Networks" MIT PhD Thesis. 1995.

[12] ANSI/IEEE Std 802.11. "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." 1999.

[13] S. Xu, T. Saadawi. "Does the IEEE 802.11 MAC protocol Work Well in Multihop Wireless Ad Hoc Networks?" IEEE Communication Magazine, Vol 39, N. 6, June 2001, pp 130-137.

[14] Z.Tang and J.J.Garcia-Luna-Aceves. "A protocol for topology-dependent transmission scheduling in wireless networks". In Proc. of IEEE Wireless Communications and Networking Conference, 1999.

[15] N.H. Vaidya, P Bahl, and S. Gupta. "Distributed fair scheduling in a wireless LAN. In Mobile Computing and Networking, pages 167-178, 2000.

[16] Xue Yang and Nitin H. Vaidya. "DSCR: A More Stable MAC Protocol for Wireless Networks". Technical Report, University of Illinois at Urbana-Champaign, August 2002.