Token-DCF: An Opportunistic MAC Protocol for Wireless Networks

Ghazale Hosseinabadi and Nitin Vaidya
Department of ECE and Coordinated Science Lab.
University of Illinois at Urbana-Champaign
{ghossei2, nhv}@illinois.edu

***Abstract*—IEEE 802.11 DCF is the MAC protocol currently used in wireless LANs. However, due to idle and collision times, 802.11 DCF performs poorly when it comes to channel utilization, system throughput, and channel access time. To overcome these sources of inefficiency in 802.11 DCF, in this paper, we propose a distributed and dynamically adaptive MAC protocol for wireless networks, called Token-DCF. Main focus of our approach is on reducing idle and collision times by introducing an implicit token passing algorithm. In Token-DCF, a transmitting station schedules one of its neighboring stations for the next transmission epoch using a distributed opportunistic algorithm. Furthermore, packet overhearing is employed to exchange scheduling information across the network. Our simulation results show that Token-DCF achieves more than 2X improvement in system throughput and channel access delay compared to 802.11 DCF for most network configurations.**

I. Introduction

IEEE 802.11 defines the distributed coordination function (DCF) to share the wireless medium among multiple stations. DCF employs CSMA/CA with a binary exponential backoff algorithm to resolve channel contention. DCF specifies random backoff, which forces a station to defer its access to the channel for a random period of time. This backoff period corresponds to the number of idle slots a station has to wait before its transmission attempt. If multiple stations choose the same backoff, they will attempt to transmit at the same time and collisions will occur. Two types of overhead are associated with random access protocols. One is channel idle time (i.e., backoff time) which is the time when contending stations are waiting to transmit. Another is collision which happens when multiple stations transmit simultaneously. If there are few contending stations, idle time is the dominant overhead. If there are many contending stations, collision probability increases and becomes the main source of low channel utilization.

In this paper, we design a distributed MAC protocol, called *Token-DCF*, in which both idle time and collision time are reduced and network throughput is improved significantly. In Token-DCF, when a station transmits on the channel, it might give a privilege (i.e., a token) to one of its neighbors. When a transmission ends, the privileged station, if there is any, starts transmitting after a short period of time, namely SIFS (Short Inter Frame Space). Non-privileged stations follow the backoff procedure of 802.11 to access the channel. In this way, the privileged station does not go through the contention resolution phase and grabs the channel immediately. A distributed scheduling algorithm is used for choosing the privileged stations.

978-1-4673-5494-3/13/$31.00

Token-DCF is fully distributed and does not require any centralized point of coordination. In Token-DCF, a station might schedule one of its neighbors for transmission on the channel. In this way, each network station acts as a scheduler. Token-DCF uses an opportunistic approach based on packet overhearing for exchanging scheduling information as well as token passing. In Token-DCF, queue length of a station is included in the MAC header of the transmitted packets and is overheard by the neighboring stations. Each station keeps track of queue length of its neighbors. Queue length information is used in the scheduling component of the protocol, where a neighbor of the transmitting station is selected as the privileged station. No extra control packet is needed for giving a privilege to a station. Instead, the next privileged station (i.e., the scheduled station) is specified in the MAC header of data packets being transmitted on the channel. The probability of giving a privilege is always less than 1 to cope with newly arrived traffic as well as imperfections in traffic estimation. This probability is adjusted based on the accuracy of the neighbors' traffic estimation. Token-DCF is an opportunistic MAC protocol which behaves similar to 802.11 DCF when packets are not overheard by the neighboring stations. However, when the opportunistic overhearing is feasible, we eliminate the backoff procedure of 802.11 DCF to improve efficiency.

The rest of the paper is organized as follows. We first review some related work in Section II. We then present our protocol, Token-DCF, in Section III. We compare our protocol with IEEE 802.11 in Section IV and finally present concluding remarks in Section V.

II. Related Work

We summarize the prior work into:

1) Distributed MAC protocols to improve the efficiency of 802.11 DCF [1], [2], [3], [4], [5], [6], [7].
2) Token passing MAC protocols [8], [9], [10].
3) Scheduling algorithms of wireless networks [11], [12], [13], [14], [15].

A. Enhancing 802.11 DCF

Various MAC protocols have been proposed to improve the efficiency of DCF. Cali et al. modify the backoff algorithm of the IEEE 802.11 MAC protocol and derive a contention window size that maximizes network throughput [1]. The backoff window size is tuned at run-time to increase the overall throughput. In this protocol, for light and medium load conditions, where the window size defined in 802.11 DCF is sufficient for guaranteeing low collision probabilities, the standard backoff algorithm is adopted. On the other hand, when the network congestion increases, based on the existing load condition, a contention window with the right size is used.

Tay et al. consider a network in which all stations become simultaneously backlogged at some point in time. They proposed *CSMA/p** which finds the optimal backoff distribution for all stations [2]. In *Idle Sense* [3], each host observes the average number of idle slots across its transmission attempts to dynamically control its contention window. Idle Sense enables each host to estimate its frame error rate, which is used for switching to the right bit rate. In *Implicit pipelining* [4], the task of contention resolution and packet transmission is partially paralleled. This technique reduces the channel idle time and collision time.

Our protocol, Token-DCF, reduces idle time and collision time by implementing an opportunistic token passing algorithm. When a transmission ends, the station holding a token, if there is any, may immediately transmit after waiting for an idle duration of SIFS. A distributed scheduling algorithm that considers network status (e.g., links' queue length) is used to choose the next station receiving a token. This results in higher channel utilization and system throughput.

Zeng et al. present *CHAIN* [5], in which clients maintain a precedence relation among one another, and a client can immediately transmit a new packet after overhearing a successful transmission of its predecessor. When the network load is low, CHAIN behaves similar to DCF; However, when the network becomes congested, clients automatically start transmission chains to improve efficiency. CHAIN requires transmission of control packets between an access point and its stations periodically, which adds overhead to the protocol. Furthermore, during each scheduling period, the specified precedence relation is fixed and does not adapt to traffic changes during that period.

IEEE 802.11 itself has been enhanced in a number of ways recently. For instance, IEEE 802.11e [6] has introduced the concept of transmission opportunities (TXOPs). A station that gains access to the channel can transmit multiple of frames separated by a SIFS. Thus, a backoff stage does not occur before each packet transmission; Instead, backoff happens before each TXOP. Also, an exchange mechanism called the *Reverse Direction (RD) protocol* has been introduced by IEEE 802.11n [7]. In this mechanism, once the transmitting station has obtained a TXOP, it may grant permission to another station to send information back during its TXOP. The transmitting station sends its permission to the RD responder using a Reverse Direction Grant (RDG) frame. The responder starts the response burst a SIFS after RDG. Our protocol, Token-DCF, is more general than these mechanisms, because in Token-DCF during each transmission, any station might be chosen as the privileged station, where a privileged station transmits on the channel without going to the backoff procedure.

B. Token passing MAC protocols

Token passing is a medium access method where a short packet called *token* is passed between stations to authorize a station for transmission. In token passing protocols, stations take turns in transmitting by passing the token from one station to another. Stations that have data frames to transmit must first acquire the token before they can transmit them. A station can only send data if it possesses the token; Thus, avoiding collisions. Token passing schemes provide round-robin scheduling method. Their advantage over contention-based medium access is that collisions are eliminated, and the available bandwidth can be fully utilized when there is a high demand. On the flip side, when the demand is light, a station wishing to transmit must wait for the token, increasing latency.

The IEEE 802.4 Token Bus protocol [8] is a well-known example of token passing protocols. This protocol is based on a broadcast medium (e.g., broadband coaxial cable), which connects all nodes to each other. The token is passed among a logical ring of stations attached to the medium. The order in which stations receive the token is determined based on their MAC addresses.

The Wireless Token Ring Protocol (WTRP) [9] is a token bus protocol, derived from IEEE 802.4. WTRP presents a token passing MAC protocol for wireless networks. When token passing is to be used in a WLAN, the characteristics of the wireless medium, such as connectivity loss, network partitioning and token loss, raise additional token management issues. WTRP is capable of recovering from token loss and duplication, and also dealing with changes in network connectivity and membership. The main modifications to 802.4, introduced by WTRP, address the partial connectivity issues in wireless networks.

Johnson et al. design another token passing MAC protocol for wireless networks, called High Frequency Token Protocol (HFTP) [10]. HFTP is based on WTRP, but adds two new mechanisms: token relaying and ring merging. Token relaying deals with a situation where a station attempts to pass a token to its successor, but fails to receive an acknowledgement due to link failure. In such a scenario, HFTP attempts to find an indirect path to its successor rather than reorganizing the ring to exclude that link. This requires new mechanisms to find and use token relay nodes. HFTP also differs from WTRP in how it merges rings that come into each other's range. This can occur after a network that was partitioned regains connectivity.

Our protocol, Token-DCF does not use round-robin scheduling for passing the token among network stations. In round-robin scheduling, when demand is low, the token might be

given to stations with no traffic, which results in under-utilization of the medium. Instead, in Token-DCF, every station estimates queue length of its neighbors and the token is always given to a neighboring station with a non-zero queue length. Furthermore, token passing mechanism of Token-DCF is implicit, in which, no additional token message is transmitted to pass the token from one station to another. Instead, token passing is done via embedding the scheduling information in the header of data packets by the source station and overhearing the packets by the neighboring stations to retrieve such information. When opportunistic overhearing is not feasible, i.e. if token is not received by the privileged station, Token-DCF operates similar to 802.11 DCF. This method eliminates the need for dealing with complicated token management issues in wireless networks such as recovering from connectivity loss, network partitioning and token loss.

C. Scheduling algorithms of wireless networks

Prior work on scheduling algorithms of wireless networks can be largely classified into two main categories:

1) Throughput-optimal scheduling: Here it is assumed that the mean arrival rates of the packets into each queue is within the capacity region, where capacity region is defined as the set of sustainable arrival rates of the channel. The centralized scheduler knows the current queue lengths and the current channel conditions. The first throughput optimal scheduling algorithm was introduced in the seminal work of Tassiulas and Ephremides [11]. The proposed algorithm is a centralized algorithm known as Backpressure. In Backpressure algorithm, the schedule at each time slot t is determined by

$$\vec{r}(t) = \text{argmax}_{\vec{r}\in\mathcal{R}}\left[\sum_{(i,j)}(q_i - q_j)r_{ij}\right] \qquad (1)$$

For each link (i, j) from station i to station j, $(q_i - q_j)$ denotes its queue differential and r_{ij} denotes its rate. $\mathcal{R}$ is the convex hull of the capacity region. In Backpressure, at each time slot, the set of non-conflicting links that maximizes the above sum is activated.

Longest-Queue-First scheduling (a.k.a., greedy maximal scheduling) [12] is another centralized scheduling algorithm, which has been observed to achieve throughput optimality in most practical wireless networks. LQF makes scheduling decisions based on the queue length information as follows. It starts with an empty schedule and first adds the link with the largest queue length to the schedule. It then looks for the link with the largest queue length among the remaining links. This selected link will be added to the schedule only if this addition creates a feasible schedule (i.e., the set of added links satisfies the SINR constraints). This process continues until no more link can be added to the schedule.

Throughput optimal scheduling algorithms are generalized in many different directions [13], [14], [15]. In throughput optimal scheduling algorithms, queues are stable if the arrival rates lie within the capacity region. Throughput-optimal scheduling is suitable for inelastic traffic where the sources do not adapt their transmission rate based on congestion in the network. In this case, admission control is required to ensure that the arrival rates lie within the capacity region of the network.

2) Fair Scheduling: An obvious drawback of throughput optimal policies is that no traffic policing is enforced. For instance, if one or more sources misbehave and increase their arrival rates so that the set of arrival rates lies outside the capacity region, then the system becomes unstable. In other words, all flows will be penalized due to the behavior of a few misbehaving flows. Thus, an alternative is to allocate resources in a fair manner to the various queues. Two examples of fair allocation are weighted proportional fair allocation and max-min fair allocation [16]. Fair scheduling is more suited for elastic traffic sources which can adjust their traffic rates in response to feedback from the network regarding the network conditions.

Various scheduling algorithms can be incorporated in our protocol, Token-DCF, depending on the objective of the scheduling algorithm and type of the arrival traffic.

III. Token-DCF design

In this section, we first provide a high-level overview of Token-DCF and then detail the scheduler signaling and algorithm.

A. Overview

At a high level, the operation of Token-DCF is described as follows. Token-DCF runs an opportunistic token passing protocol, where a token (or a privilege) might be assigned by a transmitting station to one of its neighbors. In this paper, we use the terms privilege and token interchangeably. While transmitting, the transmitting station might select one of its neighbors and give it a higher priority for the next transmission. Various selection mechanisms can be used. When a transmission ends, the station with a higher priority, called `privileged`, starts transmitting after a short period of time (i.e., SIFS), if the channel is sensed idle. Since all other stations should wait for at least a longer DIFS, the transmission of the `privileged` station will not collide with other transmissions.

Token-DCF is implemented in the MAC layer of the protocol stack. Scheduling information is embedded in the MAC header of data packets and is transferred to the neighboring stations via overhearing. Each station maintains queue length of the neighboring stations. These queue lengths are then used in the scheduling phase to select the privileged station for the next transmission. Transmitting station announces the privileged station in the `privileged` field of the MAC header of the data packets it transmits. By overhearing these packets, the privileged station is informed that it has a higher priority for the next transmission. When a transmission ends, the privileged station can start transmitting after SIFS if the

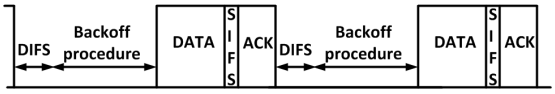

Fig. 1: Access method of IEEE 802.11 DCF

channel is sensed idle. If opportunistic overhearing does not work, i.e., token is not received by the next privileged station, Token-DCF operates similar to 802.11 DCF. But when the next privileged station overhears the token, it can transmit on the channel without going to the backoff procedure.

Signaling mechanism in the scheduling component of Token-DCF is done via embedding the scheduling information in the header of data packets by the source station and overhearing the packets to retrieve such information by the neighboring stations. When a packet is transmitted, the `privileged` station and the queue length of the transmitter are embedded in the MAC header of the packet. Once a packet is received or overheard, the queue length of the source of the packet is retrieved. Furthermore, a neighboring station checks the `privileged` field to find out if it is privileged for the next transmission. In Token-DCF, no extra control messages are transmitted to obtain network status and to pass the privileges. Collecting the information needed for scheduling, assigning a privilege to one of the neighbors and obtaining the privilege by the `privileged` station are all done through overhearing.

Token-DCF has two major components: (1) A method to reduce the idle time of the backoff procedure. (2) A scheduling algorithm to determine which neighbor should be chosen as the privileged station.

B. Reducing idle time

Token-DCF reduces the idle time of the backoff mechanism by assigning privileges to network stations. When a station transmits data packets, it might give a higher priority to one of its neighbors for the next transmission with probability p and with probability $1-p$, no station is privileged. As we will explain in Section III-C, the scheduling algorithm of Token-DCF determines which neighbor is chosen as the privileged station. When a transmission ends, the privileged station starts transmitting after SIFS, if the channel is sensed idle. Non-privileged stations follow the backoff procedure of IEEE 802.11 to access the wireless medium. Backoff mechanism of 802.11 DCF is shown in Figure 1. In this mechanism, after a transmission ends, the station senses the channel after DIFS interval and if the channel is sensed idle, it waits for a random backoff time. It chooses backoff b, an integer distributed uniformly in the window $[0, CW]$, and waits for b time slots before trying to transmit.

Channel access method of our protocol is shown in Figure 2. In Token-DCF, when the channel becomes idle, the privileged station, if there is any, starts transmitting on the channel immediately, and non-privileged stations have to defer backoff count down till when transmission of the privileged

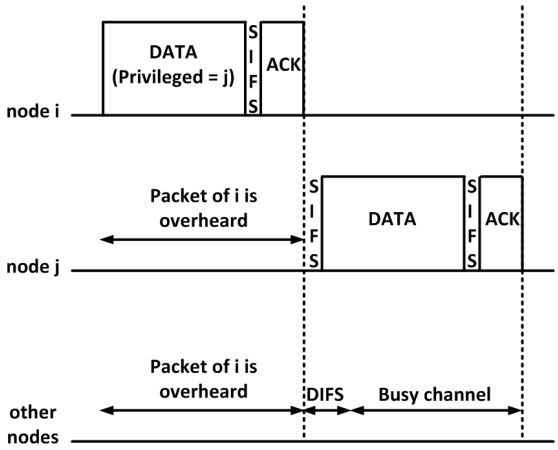

Fig. 2: Access method of Token-DCF protocol

station finishes. This process of giving a privilege to one of the neighbors of the transmitting station repeats in each transmission. Whenever a privileged station transmits on the channel, the idle time of the channel is limited to SIFS. On the other hand, in IEEE 802.11 protocol, the channel idle time between two consecutive transmissions is equal to DIFS plus a random backoff duration. Furthermore, since the privileged station immediately transmits after waiting an idle duration of SIFS, while all other stations should wait for at least a longer DIFS, the transmission of the privileged station will not collide with other transmissions.

C. Scheduling algorithm

The scheduling algorithm of Token-DCF provides a mechanism for choosing the privileged stations. In Token-DCF, when a station transmits, it acts as a scheduler as well and with probability p gives a higher priority for the next transmission to one of its neighbors. This removes the need for a separate scheduler as well as transmission of control messages between the scheduler and network stations. If a privilege is assigned to a station with an empty queue, the privileged station would not take its chance to transmit on the channel without going through the contention phase. This simply results in under-utilization of the underlying wireless channel. As long as the privilege is assigned to a station with backlogged traffic, the privileged station can immediately transmit a new packet without dealing with contention.

Different scheduling algorithms can be used for choosing the next privileged station, depending on the objective of the scheduling algorithm and type of the traffic. Here, we present two example scheduling policies. In the first algorithm, a transmitting station picks the neighbor with the largest q_i as the next privileged station, where q_i is the queue length of station i. In single hop networks, if every station overhears every transmission, this policy implements Longest-Queue-First [12] as the scheduling component of Token-DCF, which guarantees throughput optimality. In the second algorithm, a transmitting station uniformly at random chooses one of its neighbors with

backlogged traffic (i.e., with non-zero queue length). This policy achieves fairness among the network stations, because in this policy all stations have an equal chance to be chosen for transmitting on the channel.

D. Protocol details

The detailed Token-DCF is presented in this section. Procedure III-D.1 sets the initial values of Token-DCF parameters. p, the probability of giving a privilege, is initially set to zero and changes during the execution. *active* denotes the set of neighbors of a station that has transmitted on the channel during the current scheduling period and the transmission is overheard by the station. The station itself, *myId*, is also included in the set *active*. When a station transmits, it might give a privilege to one of the stations in the set *active*. By including *myId* in the set *active*, a station might choose itself as the *privileged*. Each station keeps track of the transmissions on the channel by overhearing of the packets. *nFail* denotes the number of transmissions in which the sender of the packet is not in the set *active*. *nSuccess* denotes the number of transmissions from the set *active*. *flag* is a boolean variable denoting if the station has a privilege for accessing the channel or not. *flag* equals to `true` means that the station has a privilege for transmission on the channel. *flag* is initially set to `false`, meaning that no station is privileged initially. Initially, a station with data for transmission has to go through the backoff process to access the medium. Protocol parameters are reset to initial values after each *period* seconds. Protocol parameters are reset periodically in order to prevent the stale information from degrading the protocol performance.

III-D.1 Initialization at station *myId*

```
1: p = 0
2: active = {myId}
3: nFail = 0
4: nSuccess = 0
5: flag = false
6: call Initialization after period
```

Procedure III-D.2 is executed right before a packet is transmitted on the channel. If the packet is a MAC data packet, the station might give a privilege to one of its neighbors. The mechanism of assigning a privilege or transmitting as the privileged station is not used when control packets are transmitted. In this way, the transmission of non-data packets such as ARP packets or routing packets are not affected by our protocol. The station that is chosen to be the privileged station is called *privileged*.

As explained before, different criteria can be used for choosing the privileged station, *privileged*. One example scheduling algorithm is presented in III-D.2. In III-D.2, *privileged* is the station in the set *active* with the longest queue. Another example of scheduling algorithms is the one in which *privileged* is chosen uniformly at random from the set of stations in *active* with non-zero queue length. This policy achieves fairness among the network stations. Many other queue based scheduling algorithms are presented in the literature that can be incorporated in Token-DCF protocol [11], [13], [14].

If a station chooses itself as the *privileged*, it sets its *flag* to `true`. Otherwise, *flag* is set to `false`. Procedure III-D.6, called Adapt, is then called to update *nSuccess*, *nFail* and p.

III-D.2 Transmit a packet

```
1: if transmitting a MAC data packet then
2:     generate a random number r uniformly distributed
       on [0, 1]
3:     if r < p then
4:         privileged = station with the longest queue
           in active
5:     else
6:         privileged = null
7:     if privileged == myId then
8:         flag = true
9:     else
10:        flag = false
11:    Adapt
12: else
13:    privileged = null
```

Procedure III-D.3 is called when a packet is received or overheard. Since the wireless channel is a shared medium, station i might overhear packets that are not intended for it, i.e., packets with destination address different from i. If the station is chosen to be the *privileged* in the received or overheard packet, it sets its *flag* to `true`. Otherwise, *flag* is set to `false`. The station then calls Adapt (Procedure III-D.6) in which, *nSuccess*, *nFail* and p are updated. The station also saves the queue length of *src* in a local variable *qLen*.

III-D.3 Receiving or Overhearing a packet from station *src*

```
1: if privileged == myId then
2:     flag = true
3: else
4:     flag = false
5: Adapt
6: qLen[src] = queue length of src
```

Procedure III-D.4 is executed when a station starts or resumes its backoff timer. If the station has higher priority (i.e., *flag* == `true`) and the packet is a MAC data packet, the backoff duration is set to SIFS. Otherwise, the backoff duration is chosen to be DIFS plus random number of time slots, similar to 802.11 DCF.

As explained in Procedure III-D.5, when the backoff timer expires, *flag* is reset to `false`. In this way, a privileged station has the privilege to transmit only one packet immediately after the last transmission ends. In case the packet is

III-D.4 Starting or resuming backoff timer

```
1: if flag == true && packet is a MAC data packet then
2:     schedule backoff timer for SIFS
3: else
4:     schedule backoff timer for DIFS + random number of time slots
```

lost, the station does not have the privilege for retransmission of the packet and will follow the backoff procedure to access the channel. When a host detects a failed transmission (it does not receive the ACK of a frame), it executes the exponential backoff algorithm, doubling Contention Window CW (CW may vary between CW_{min} and CW_{max}).

III-D.5 Backoff timer expiration

```
1: flag = false
```

When a packet is transmitted, received or overheard, Adapt (Procedure III-D.6) might be called, in order to update the value of *nSuccess*, *nFail* and p. Station i calls Adapt when it transmits, receives or overhears a packet. If transmitter of the packet, *src*, does not belong to the set *active*, *nFail* is increased by one and *src* is added to the set *active*. In this case, the station that receives or overhears the packet, has not received any transmission from *src* during the current transmission period. Otherwise, if *src* belongs to the set *active*, *nSuccess* is increased by 1. Recall that the set *active* is reset every *period* seconds.

Enough transmissions should happen before adapting p. If so, (i.e., if *nSuccess*+*nFail* >= *maxNum*), ratio of *nSuccess* to *nSuccess*+*nFail* is then recalculated to adapt p. If *ratio* is larger than a threshold, *maxRatio*, p is increased by δ and *nSuccess* and *nFail* are reset to 0. We note that p is increased up to a threshold, *maxP*. It is reasonable to choose *maxP* less than 1 in order to always give a chance to stations not in the set *active* to be able to transmit on the channel. If *ratio* is less than a threshold, *minRatio*, p is decreased by δ and *nSuccess* and *nFail* are reset to 0.

What we have presented in Procedure III-D.6 is an example of dynamically adapting the protocol parameters. There are other alternatives for adapting protocol parameters. For example, different moving average techniques (e.g., weighted, exponential, $\cdots$) can be used to adapt the parameters.

III-D.6 Adapt

```
1: if src ∉ active then
2:     nFail ++
3:     add src to active
4: else
5:     nSuccess ++
6: if (nSuccess+nFail >= maxNum) then
7:     ratio = nSuccess / (nSuccess+nFail)
8:     if (ratio >= maxRatio) then
9:         if (p <= maxP) then
10:            p = p + δ
11:        nSuccess = 0
12:        nFail = 0
13:    if (ratio <= minRatio) then
14:        if (p >= δ) then
15:            p = p − δ
16:        nSuccess = 0
17:        nFail = 0
```

TABLE I: WLAN configuration

SIFS	10 μsec
DIFS	28 μsec
slot time	9 μsec
phy preamble	16 μsec
bit rate	54 Mbps
CWmin	16
CWmax	1024

TABLE II: Token-DCF parameters

Parameter	Value
minRatio	0.2
maxRatio	0.8
maxNum	20
maxP	0.9
period	0.1 sec
δ	0.1

IV. Evaluation

We simulate Token-DCF and 802.11g in ns-2 to measure and compare performance of these two MAC protocols. Table I reports the configuration parameter values of the wireless network analyzed in this section. Table II summarizes the parameter values of Token-DCF chosen in the simulations. The network is a wireless ad hoc network in which transmitting stations are placed uniformly at random in a square area. Flows are single hop, and the receiver of each flow is placed at a distance of $100m$ from the transmitter of the flow. We run the simulations for different network sizes, including single-hop and multi-hop networks. The effective transmission range in the simulations is limited to 250 meters and carrier sense range is limited to 550 meters. IEEE 802.11 RTS/CTS mechanism is turned off. Two-ray ground radio propagation model is assumed. Packet payload size is 1500 bytes. Each simulation lasts for 30 seconds and the presented results are averaged over 20 runs. In each run, a different random network topology is considered. In our simulations, the scheduling algorithm presented in Procedure III-D.2 is used as the scheduling component of Token-DCF. We measure the performance of Token-DCF and 802.11 DCF in terms of aggregate throughput, average access delay, channel idle time and collision frequency.

A. Performance evaluation in saturated single-hop networks

Figures 3 - 6 plot the performance parameters in a single-hop network. The size of the network is $150m$x$150m$ and all

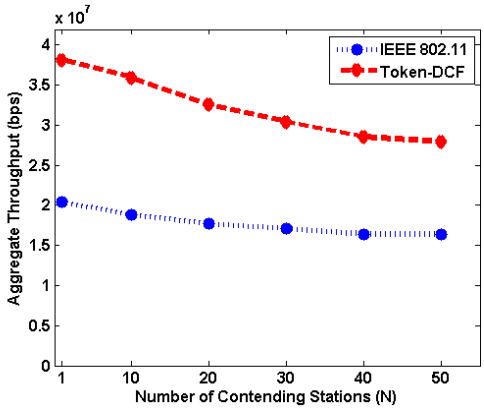

Fig. 3: Aggregate throughput (area=150mx150m)

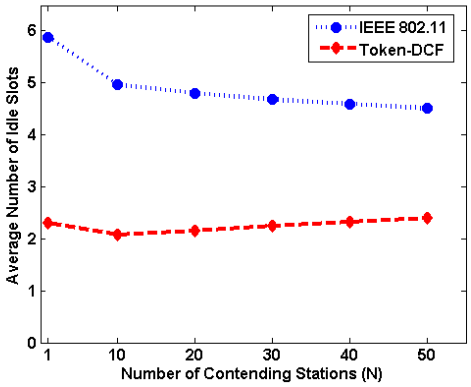

Fig. 5: The average number of idle slots before each media access

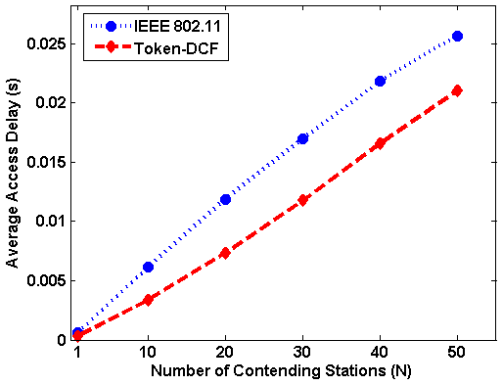

Fig. 4: Average access delay (area=150mx150m)

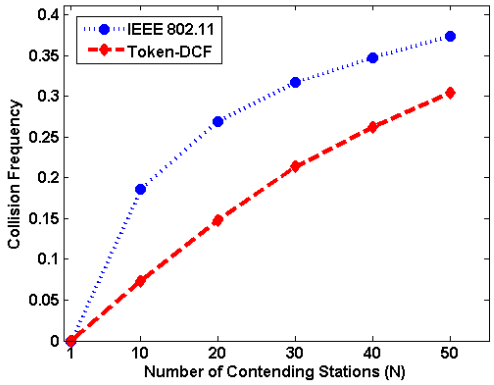

Fig. 6: Collision frequency

flows are single-hop. Traffic is full buffer CBR, meaning that there is always backlogged traffic in the transmission queue of each transmitter.

The aggregate throughput of 802.11 DCF and Token-DCF is presented in Figure 3. As can be seen, throughput gain obtained by Token-DCF compared to IEEE 802.11 in Figure 3 is a factor of $1.7 - 1.9$. Figure 4 shows the average access delay of the two protocols. Access delay is defined as the delay between the time a packet arrives at the MAC layer and the time the source of the packet receives acknowledgment from the destination. Access delay of a packet consists of the waiting time before transmitting on the channel and the time spent in packet retransmissions. As we can see in Figure 4, access delay is smaller in Token-DCF by a factor of $0.53 - 0.81$. As we will explain, the reason is that Token-DCF has a much shorter idle time compared to IEEE 802.11 DCF. Furthermore, many retransmissions are avoided because of reduced collision frequency.

Figure 5 presents the average number of idle slots before each media access. Token-DCF has shorter channel idle time, because in Token-DCF, a privileged station accesses the channel immediately after the latest transmission finishes. In this way, channel stays idle only for SIFS seconds, instead of DIFS plus random backoff duration. We note that the average number of idle slots in Token-DCF is not zero. The reason is that with a non-zero probability, no station is chosen as the privileged station for the next transmission. In such a case, stations follow the backoff mechanism of 802.11 DCF to get an access for transmission on the channel.

Collision frequencies of 802.11 DCF and Token-DCF are shown in Figure 6. Collision frequency is defined as the number of times a transmission fails due to collision normalized by the total number of transmissions (counting retransmissions as well). Figure 6 indicates that Token-DCF has a much lower collision frequency than 802.11 DCF. Recall that when a station transmits, it might choose one of its neighbors as the privileged station. In a single-hop network, at each time instant, at most one station successfully transmits on the media and as a result, there is at most one privileged station at each time instant. Since a privileged station does not follow the backoff mechanism of 802.11 DCF, the transmission by a privileged station does not collide with any other transmission in a single-hop network. This reduces the collision frequency of the protocol. Reducing the idle time and collision time of the channel increases throughput and decreases media access delay. As we can see in Figure 6, with greater number of

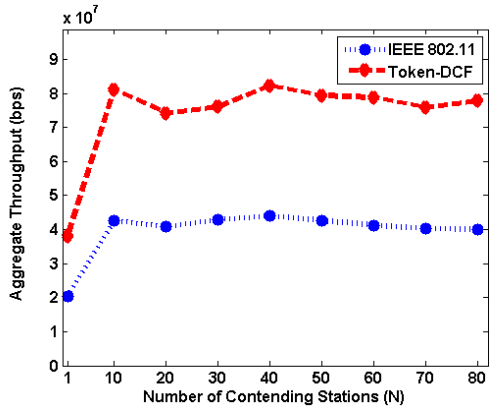

Fig. 7: Aggregate throughput (area=800mx800m)

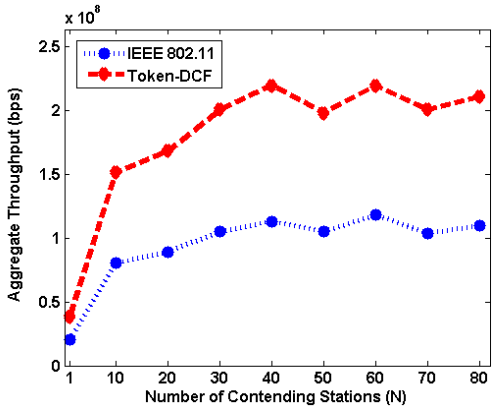

Fig. 9: Aggregate throughput (area=1500mx1500m)

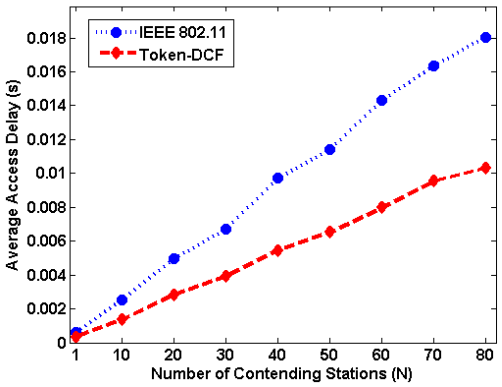

Fig. 8: Average access delay (area=800mx800m)

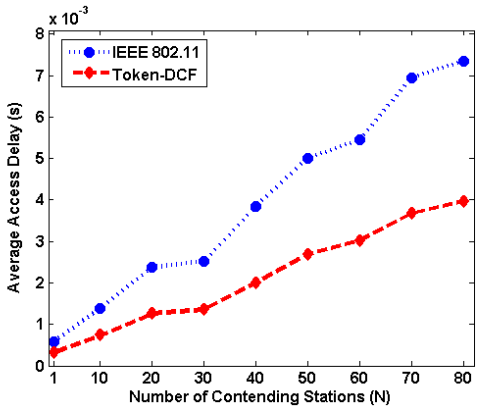

Fig. 10: Average access delay (area=1500mx1500m)

contending stations, the collision frequency in both Token-DCF and 802.11 DCF increases. Token-DCF has non-zero collision frequency, because with probability $1-p$, stations implement backoff mechanism for contention resolution, which might cause collisions.

B. Performance evaluation in saturated multihop wireless networks

In this section, we study performance of Token-DCF in multihop wireless networks. We consider two network sizes; $800m$x$800m$ and $1500m$x$1500m$. Recall that the effective transmission range in the simulations is limited to 250 meters and carrier sense range is limited to 550 meters. Traffic is full buffer CBR and all flows are single hop. Aggregate throughput and average access delay of the networks with size $800m$x$800m$ versus number of contending stations are presented in Figures 7 and 8, respectively. Comparing Token-DCF and 802.11 DCF in these two figures, we can see that throughput gain is a factor of $1.8-2$ and access delay is reduced by a factor of $0.53-0.58$. For the networks of size $1500m$x$1500m$, aggregate throughput and average access delay are presented in Figures 9 and 10, respectively. In this case, throughput gain is a factor of 1.9 and access delay is reduced by a factor of $0.52-0.55$. Considering Figures 3-10, we see that similar performance improvement is obtained by Token-DCF in single hop and multihop networks. The reason is that, in multi-hop networks, Token-DCF improves the channel utilization in each transmission range.

C. Stations with unsaturated traffic

Having shown the performance improvement of Token-DCF over 802.11 for saturated networks, we further identify its performance in networks that have less traffic load. This set of simulations focuses on comparing the performance of Token-DCF with 802.11 when varying the traffic load from low to high. On/Off traffic with burst times and idle times taken from pareto distributions is used. Average on time of the traffic generator is $50ms$. Its average off time is also set to $50ms$.

We perform simulations for randomly generated networks of size $150m$x$150m$. There are a total of 20 one-hop flows. Each source station generates its packets independently. Sending rate during on time, called Rate, is varied between 10^3 bps and 10^8 bps. With Rate $= 10^3$ bps and 1500 bytes packet size, the traffic demand is far below the network capacity. When gradually varying Rate from 10^3 to 10^8 bps, offered load is increased from small to very large. The corresponding

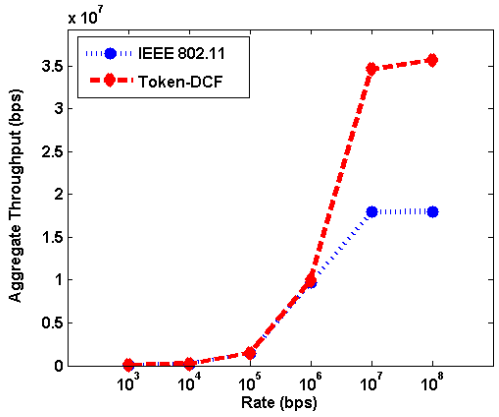

Fig. 11: Aggregate throughput (Pareto traffic)

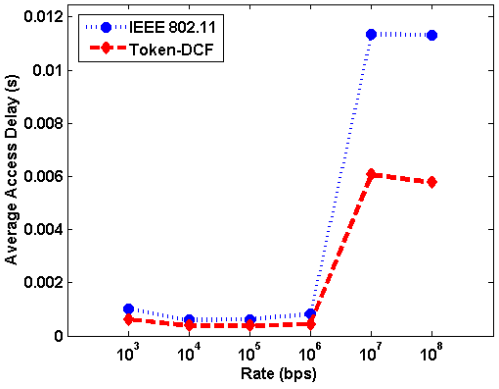

Fig. 12: Average access delay (Pareto traffic)

aggregate throughput and average access delay are presented in Figures 11 and 12, respectively. When the network load is very low, station queues are empty most of the time in which case, no station is chosen as the privileged station. Under low load, Token-DCF behaves very similar to 802.11 DCF. Their performance starts to diverge when the network is loaded more heavily. The saturation throughput of Token-DCF is approximately 2 times of 802.11 DCF.

V. Conclusion

In this paper, we presented the design and performance evaluation of Token-DCF. Token-DCF is a distributed MAC protocol that uses an opportunistic overhearing mechanism to schedule network stations for transmission on the channel. The main design goal of Token-DCF is to reduce both idle time and collision time by introducing an implicit token passing algorithm. Our simulation results show that Token-DCF achieves 2X improvement in system throughput and channel access delay compared to 802.11 DCF for most network configurations.

VI. Acknowledgement

This research is supported in part by National Science Foundation award CNS 11-17539. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

References

[1] F. Cali, M. Conti, and E. Gregori, "Dynamic tuning of the ieee 802.11 protocol to achieve a theoretical throughput limit," *IEEE/ACM Trans. on Networking,* vol. 8, no. 6, December 2000.

[2] Y. C. Tay, K. Jamieson, and H. Balakrishnan, "Collision-minimizing csma and its applications to wireless sensor networks," *IEEE Journal on Selected Areas in Communications,* vol. 22, 2004.

[3] M. Heusse, F. Rousseau, R. Guillier, and A. Duda, "Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless lans," *SIGCOMM Comput. Commun. Rev.,* vol. 35, no. 4, 2005.

[4] X. Yang and N. H. Vaidya, "A wireless mac protocol using implicit pipelining," *IEEE Trans. on Mobile Computing,* vol. 5, no. 3, 2006.

[5] Z. Zeng, Y. Gao, K. Tan and P. R. Kumar, "CHAIN: Introducing minimum controlled coordination into random access MAC," *In Proceedings of INFOCOM 2011,* pp. 2669-2677, 2011.

[6] "Wireless medium access control (MAC) and physical layer (PHY) specifications: Medium access control (MAC) enhancements for quality of service (QoS)," *IEEE Std. 802.11e/Draft 5.0,* July 2003.

[7] "802.11n: Next-generation wireless LAN technology," *Broadcom Corporation,* 2006.

[8] ANSI/IEEE Standard 802.4, "Token-passing bus access method and physical layer specifications," IEEE, 1985.

[9] M. Ergen, D. Lee, A. Puri, P. Varaiya, R. Attias, R. Sengupta, S. Tripakis, "Wireless token ring protocol," *Proceedings of the Eighth IEEE Symposium on Computers and Communications (ISCC 2003)* pp. 710-715, 2003.

[10] E. E. Johnson, Z. Tang, M. Balakrishnan, J. Rubio, H. Zhang, and S. Sreepuram, "Robust token management for unreliable networks," *In Proceedings of the 2003 IEEE Conference on Military Communications, MILCOM'03,* vol. 1, 2003.

[11] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control,* 37(12): pp. 1936-1948, 1992.

[12] A. Dimakis and J. Walrand. "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits." *Advances in Applied Probability,* 38(2):505521, 2006.

[13] S. Shakkottai, R. Srikant, and A. Stolyar. "Pathwise optimality of the exponential scheduling rule for wireless channels," *Advances in Applied Probability,* 36:10211045, 2004.

[14] A. Eryilmaz, R. Srikant, and J. Perkins. "Stable scheduling policies for fading wireless channels," *In Proceedings of IEEE International Symposium on Information Theory,* 2003.

[15] M. Andrews, K. Kumaran, K.Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting. "Providing quality of service over a shared wireless link," *IEEE Communications Magazine,* February 2001.

[16] F. P. Kelly. "Charging and rate control for elastic traffic," *European Transactions on Telecommunications,* 8:3337, 1997.