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

Technical Report (February 2, 2012)

***Abstract*—IEEE 802.11 DCF is the MAC protocol currently used in wireless LANs. 802.11 DCF is inefficient due to two types of overhead; channel idle time and collision time. This paper presents the design and performance evaluation of an efficient MAC protocol for wireless networks, called Token-DCF. Token-DCF decreases both idle time and collision time. In Token-DCF, each station keeps track of neighboring links' queue length by overhearing of transmitted packets on the wireless medium. The result is then used to assign privileges to the network stations. A privileged station does not follow the backoff mechanism and transmits immediately after the channel is sensed idle. Our simulation results show that Token-DCF can significantly improve channel utilization, system throughput and channel access delay over 802.11 DCF.**

I. Introduction

IEEE 802.11 defines the distributed coordination function (DCF) to share the wireless medium among multiple stations. DCF employs CSMA/CA with the 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. Backoff counter 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 overhead is channel idle time (e.g. backoff time) which is the time when contending stations are waiting to transmit. Another is collision 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 reason of low channel utilization. In the literature, many MAC protocols have been proposed to reduce the total overhead caused by idle periods and collisions [1], [2], [3], [4], [5].

In this paper, we design an efficient MAC protocol, called *Token-DCF*, in which both idle time and collision time are reduced significantly. In Token-DCF, when a station transmits on the channel, it might give a privilege to one of its neighbors. When a transmission finishes, the privileged station, if there is any, starts transmission after a short period of time, namely SIFS (Short Inter Frame Space). Non-privileged stations follow the backoff mechanism 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. Since in Token-DCF contention resolution is done via assigning tokens, or privileges, idle time and collision time are decreased significantly.

A scheduling policy is a rule to determine a set of links to be activated simultaneously at each time instant. The scheduling policy of 802.11 DCF is CSMA random access, in which a station has to sense the channel as idle for a random period of time before it transmits on the channel. Depending on network configuration, IEEE 802.11 can operate very far from the throughput capacity. Several centralized and distributed scheduling algorithms were designed for wireless networks [11], [12], [13], [14] which have better throughput characteristics than 802.11 DCF. Centralized scheduling algorithms rely on a centralized coordinator to manage channel access, which is often not available in distributed networks. Distributed scheduling algorithms do not rely on such a coordinator and each station makes the scheduling decision in a distributed way.

Token-DCF is fully distributed and does not require any centralized point of coordination. Furthermore, it works for both single hop and multi hop flows. In Token-DCF, a station might schedule its neighbors for transmission on the channel. In this way, each network station performs as a scheduler. Token-DCF is flexible in the sense that it allows different scheduling mechanisms to be used for assigning privileges to network stations. Token-DCF uses an opportunistic approach based on packet overhearing to exchange scheduling information. In Token-DCF, queue length of a station is included in the MAC header of the packets it transmits 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, which chooses a neighbor of the transmitting station as the privileged station. No extra control packet is transmitted to assign a privilege to a station. Instead, the next privileged station (scheduled station) is specified in the MAC header of data packets being

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.

transmitted on the channel. The probability of assigning a privilege is always less than 1 to allow transmission of newly arrived traffic on the channel as well as imperfections in traffic estimation. This probability is adjusted based on the accuracy of the neighbors' traffic estimation.

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

II. Related Work

We review four categories of existing work,

1) Protocols to decrease the idle time and collision time of IEEE 802.11 DCF [1], [2], [3], [4], [5], [6].
2) Token passing MAC protocols [7], [8], [9], [10].
3) Centralized scheduling algorithms for wireless networks [11], [12].
4) Distributed throughput optimal CSMA protocols [13], [14], [15].

A. Enhancing 802.11 DCF

[1] modifies the backoff algorithm of the IEEE 802.11 MAC protocol and derives the contention window size that maximizes network throughput. The backoff window size is tuned at run time to increase the throughput. In this protocol, in light and medium load conditions, in which the window size defined in 802.11 DCF is sufficient to guarantee low collision probabilities, the standard backoff algorithm is generally adopted. On the other hand, when the network congestion increases, a contention window with the right size for that load condition is used.

Model-based frame scheduling (*MFS*) is presented in [2]. In MFS, each station estimates the current network status by keeping track of the number of collisions it encounters between its two consecutive successful frame transmissions, and, based on the the estimated information, computes the current network utilization. The result is then used to determine a scheduling delay that is introduced, with the objective of avoiding collision, before a station attempts for transmission of its pending frame.

[3] considers a network in which all stations become simultaneously backlogged at some point in time and designs *CSMA/p** to find the optimal backoff distribution according to which every station chooses a contention slot. In *Idle Sense* [4], each host observes the mean number of idle slots between 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. *Implicit pipelining* [5] parallelizes part of the contention resolution time and packet transmission time. It partially hides channel idle overhead and reduces collision overhead.

[6] presents *CHAIN*, in which clients maintain a precedence relation among one another, and a client can immediately transmit a new packet after it overhears a successful transmission of its predecessor, without going through the regular contending process. When the network load is low, CHAIN behaves similar to DCF; But when the network becomes congested, clients automatically start chains of transmissions 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.

B. Token passing MAC protocols

Token passing is a medium access method where a short packet called a *token* is passed between stations that authorizes the station to transmit. In token passing protocols, stations take turns in transmitting by passing the token from station to station. 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. The advantage over contention based medium access is that collisions are eliminated, and that the available bandwidth can be fully utilized without idle time when demand is heavy. The disadvantage is that even when demand is light, a station wishing to transmit must wait for the token, increasing latency.

The IEEE 802.4 Token Bus protocol [7] is a well-known example of token passing protocols. Token bus protocol is based on a broadcast medium (broadband coaxial cable), which connects all nodes to each other. The token is passed among a logical ring of stations attached to the cable. The stations sort themselves for order of token passing by their MAC addresses.

IEEE has standardized another token passing MAC protocol for wired networks, called token ring (IEEE 802.5) [8]. Stations on a token ring LAN are logically organized in a ring topology with data being transmitted sequentially from one ring station to the next with a control token circulating around the ring controlling access. In token ring standard, token is passed around a ring and whichever station holds the token is allowed to transmit before putting the token back on the ring.

The Wireless Token Ring Protocol (WTRP) [10] 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 dealing with changes in network connectivity and membership. The principal modifications of 802.4 that are introduced by WTRP address the partial connectivity issues that arise in wireless networks.

[9] designs another token passing MAC protocol for wireless networks, called high frequency token protocol (HFTP). HFTP is based on WTRP, but adds two new mechanisms: token relaying and a ring merging procedure. Token relaying deals with the situation when a station attempts to pass the token to its successor, but fails to receive acknowledgement

due to a link outage. HFTP will attempt to find an indirect path to its successor rather than reconnecting the ring to exclude that node. This requires new mechanisms to find and to use token relay nodes. HFTP also differs from WTRP in its mechanism for merging rings that come into range of each other. This can occur after a network that was partitioned regains connectivity.

C. Centralized Scheduling Algorithms

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

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

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. Backpressure is a centralized throughput optimal scheme, which is capable of scheduling all feasible flow arrivals while maintaining the network stable. When flows are single hop, i.e., communication is between adjacent stations, then the backpressure algorithm reduces to

$$\vec{r}(t) = \text{argmax}_{\vec{r} \in \mathcal{R}} \left[\sum_{(i,j)} q_i(t) r_{ij} \right] \tag{2}$$

Another important scheduling policy which has been observed to achieve 100% throughput in most practical wireless networks is longest-queue-first scheduling, also called greedy maximal scheduling [12]. LQF makes scheduling decisions based on queue length information as follows. It starts with an empty schedule. It 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 chosen link will be added to the schedule if this addition creates a feasible schedule, i.e. the set of added links satisfies the SINR constraints, or it is discarded otherwise. This process continues until no link is left.

A centralized scheduler requires a central authority to determine the schedule. In a distributed wireless network such a central authority does not necessarily exist. Consequently, various distributed scheduling schemes were designed for wireless networks that might not be throughput optimal but are simpler than a centralized algorithm and can be implemented in a large scale wireless network.

D. Throughput optimal CSMA

Recently, it has been shown that distributed carrier sense multiple access (CSMA) algorithms can achieve throughput optimality under certain network models and assumptions [13], [14], [15]. Jiang and Walrand [13] proposed a distributed adaptive random access CSMA algorithm, in which the backoff time of a link is an exponentially distributed random variable. The mean of this random variable changes over time and its dynamic is determined by the queue length of the link. Their algorithm achieves throughput-optimality under the assumption of continuous-time backoff duration (zero probability of collision) and continuous-time transmission duration.

Ni and Srikant [14] designed a distributed CSMA/CA protocol for achieving maximum throughput in a discrete-time setting. In their work, the model of an idealized CSMA protocol with continuous backoff times, under which collisions cannot occur, is relaxed. Collisions of data packets are avoided through the exchange of control messages, where control messages might collide. The optimality in protocols designed in [13] and [14] is established under the ideal carrier sensing assumption, i.e., each link can precisely sense the presence of other active links in its neighborhood. [15] investigates the achievable throughput of the CSMA algorithm under imperfect carrier sensing. Their main result is that CSMA can achieve an arbitrary fraction of the capacity region if certain access probabilities are set appropriately.

Channel access method in throughput-optimal CSMA protocols is random access in which contention among the stations for accessing the channel is resolved through the backoff mechanism. This results in non-trivial backoff overhead (idle time) in these protocols.

III. Token-DCF design

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

A. Overview

At a high level, the operation of Token-DCF is described as follows. Token-DCF runs a distributed scheduling protocol, where a privilege might be assigned by a transmitting station to one of its neighbors. In each transmission, the transmitting station might select one of its neighbors to have a higher priority for the next transmission. Selection mechanism is based on flow queue lengths. When a transmission finishes, the station with a privilege, called *privileged*, starts transmission after a short period of time, SIFS (Short Inter Frame Space), if the channel is sensed idle.

Token-DCF is implemented in the MAC layer of the protocol stack. Scheduling information is embedded in the MAC header of the data packets and is transferred to the stations via overhearing. Token-DCF reduces signaling overhead in its scheduling component compared to central scheduling algorithms. Each station maintains queue length of the neighboring stations. The queue lengths are used in the scheduling component 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 of these packets, the privileged station is informed that it has a higher priority for the next transmission. When a transmission finishes, the privileged station can start transmission after SIFS, if the channel is sensed idle. Note that in multi-hop networks, at

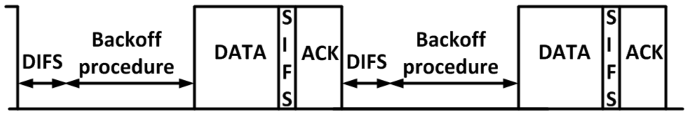

Fig. 1: Access method of IEEE 802.11 DCF

each time instance, several privileged stations might be present in the network, since in multi-hop networks, non-interfering transmitters transmit at the same time and each of them assigns a privilege to one of their neighbors.

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 of the packets to retrieve such information by the neighboring stations. When a packet is transmitted, the station that will have higher priority for the next transmission, 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, queue length of the source of the packet is saved by the receiving or overhearing station. Furthermore, a station that receives or overhears a packet, checks the `privileged` field of the MAC header of the packet to find if it is chosen to be the next privileged station. In Token-DCF, no central scheduler is deployed in the network and no extra control messages are transmitted to find and disseminate a schedule. 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 via receiving or overhearing of data packets.

Token-DCF has two major components: (1) A method to reduce the idle time of the backoff mechanism. (2) A scheduling algorithm to determine which neighbor is 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 higher priority for the next transmission to one of its neighbors. A transmitting station gives a high priority to one of its neighbors with probability p. With probability $1-p$, no station is given a higher priority. As we will explain in Section III-C, the scheduling algorithm of Token-DCF determines which neighbor is chosen as the privileged station, i.e., the station with a higher priority. When a transmission finishes, a station that has a privilege starts transmission after short period of time, SIFS, if the channel is sensed idle. Non-privileged stations follow the backoff mechanism 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 finishes, the station senses the channel after DIFS interval and if the channel is sensed idle, it waits for a random contention time: it chooses backoff b, an integer distributed uniformly in the window $[0, CW]$ and waits for b time slots before attempting to transmit.

Channel access mechanism of our protocol, Token-DCF, is shown in Figure 2. In Token-DCF, when the channel becomes idle, the privileged station, if there is any, starts transmission on the channel immediately, and non-privileged stations have to defer backoff count down till when transmission of the privileged station finishes. The process of giving a privilege by a transmitting station to one of its neighbors repeats in each transmission. Whenever a privileged station transmits on the channel, the idle time of the channel is only SIFS. On the other hand, in IEEE 802.11 protocol, the channel idle time between two consecutive transmissions is equal to DIFS plus random backoff duration.

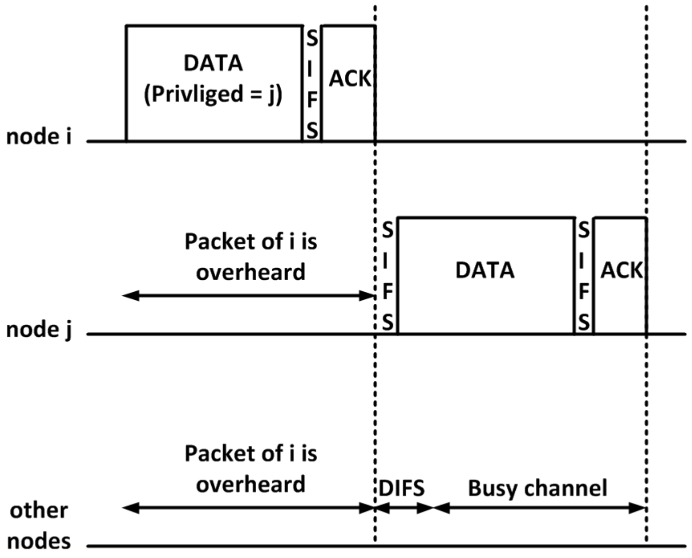

Fig. 2: Access method of Token-DCF protocol

C. Scheduling algorithm

The scheduling algorithm of Token-DCF provides a mechanism for choosing the privileged stations. Information needed in the scheduling component of the protocol is embedded in the MAC header of data packets. Such information includes queue length of transmitter of the packet and the next privileged station. Each station keeps track of neighbor's queue length in order to enable neighbor scheduling. 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 technique removes the need for a separate scheduler as well as transmission of control messages between the scheduler and network stations. In central scheduling algorithms, scheduler component and network hosts must exchange control information to coordinate the schedule. As a trade off, our approach is opportunistic and uses message overhearing to exchange the information needed in the scheduling component.

Different mechanisms can be used to choose the privileged station. In this paper, we presently consider only single hop flows (i.e., sender and receiver are adjacent nodes), but our ideas can be extended to multi-hop flows as well. A station might choose the neighbor with the largest $q_i c_{ij}$ as the next privileged station, where q_i is queue length of transmitter of link (i, j) and c_{ij} is the capacity of link (i, j). If this policy is implemented as the scheduling component of the protocol, a transmitting node should announce its queue length, q_i, as well as its link capacity, c_{ij}, in the packets it transmits. In single hop networks, if every station overhears packets of every other station, this policy implements backpressure scheduler, explained in Section II-C. In single hop networks, data transmission of any two links interferes with each other and as a result, at each time instance, at most one link can be scheduled for transmission. If network is single hop and every station overhears every other transmission, each station knows queue length q_i and capacity c_{ij} of other network links and schedules the link with the largest $q_i c_{ij}$ as the privileged link. In single hop networks, the link with largest $q_i c_{ij}$ is the one that maximizes Equation (2). In practice, c_{ij} may be approximated by the transmission rate used by the MAC-layer rate control algorithm.

Another scheduling policy is to pick the link with the longest queue. In single hop networks, when every station overhears every transmission, this policy implements longest-queue-first (LQF) as the scheduling component of Token-DCF. LQF is throughput optimal if the so called local pooling condition is satisfied [12]. In our simulations, we have used LQF as the scheduling component of Token-DCF, and all transmissions occur at a fixed rate.

D. Protocol details

Procedure III-D.1 sets the initial value of protocol parameters. A station that is going to transmit on the channel, with probability p, chooses one of its neighbors to have a higher priority for transmission. With probability $1-p$, no station is chosen to have a privilege. p is initially set to zero and changes during the protocol execution in order to adapt the probability of giving a privilege to neighbors. `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 station. Each station keeps track of the transmissions on the channel by overhearing of the packets. `success` denotes the number of transmissions from the set `active`. `fail` denotes the number of transmissions in which the sender of the packet is not in the set `active`. Protocol parameters are reset to initial values each `period` seconds. Protocol parameters are reset periodically in order to prevent stale information making the protocol unfair. An alternative to this method (i.e., resetting the initial values) is to use moving average for adapting parameter values during the protocol execution.

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 chooses a `privileged` station with probability p, where `privileged` is the station in the set `active` with the longest queue. With probability $1-p$, no station is given a privilege. If a station chooses itself as the `privileged`, it sets its `flag` to 1. Otherwise, `flag` is set to 0. `flag` equals to 1 means that the station has a privilege for the next transmission on the channel. Procedure III-D.5, called Adapt, is then called to update `success`, `fail` and p.

III-D.1 Initialization at station `myId`

```
1: $p = 0$
2: $active = \{myId\}$
3: $success = 0$
4: $fail = 0$
5: call Initialization after $period$
```

III-D.2 Transmit a packet

```
1: **if** it is a MAC data packet **then**
2:     with probability $p$
3:     $privileged$ = station with the longest queue in $active$
4:     **if** $privileged == myId$ **then**
5:         $flag = 1$
6:     **else**
7:         $flag = 0$
8:     Adapt
9: **else**
10:    $privileged = null$
```

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 1. Otherwise, `flag` is set to 0. The station then calls Adapt, Procedure III-D.5, in which `success`, `fail` and `p` are updated. The station also saves the queue length of `src` in its `qLen`.

III-D.3 Receive or Overhear a packet from station `src`

```
1: **if** $privileged == myId$ **then**
2:     $flag = 1$
3: **else**
4:     $flag = 0$
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` $== 1$, 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. There are alternatives to this approach, for example the privileged station might choose a smaller backoff compared to non-privileged stations. This approach will decrease the probability of collision when there are multiple privileged stations. Recall that in multihop networks, at each time instant more than one privileged node might exist in the network.

III-D.4 Start or resume backoff timer

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

When the backoff timer expires, $\mathtt{flag}$ is reset to zero. In this way, a privileged station has the privilege to transmit only one packet immediately after the last transmission finishes. In case the packet is lost, the station does not have the privilege for retransmission of the packet and will follow the backoff mechanism 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—it doubles contention window CW (CW may vary between CW_{min} and CW_{max}).

As explained before, when a packet is transmitted or received, Adapt (Procedure III-D.5) might be called, in order to update the value of $\mathtt{success}$, $\mathtt{fail}$ and p. Station i calls Adapt when it transmits a packet or when it receives or overhears a transmission. If transmitter of the packet, $\mathtt{src}$, does not belong to the set $\mathtt{active}$, $\mathtt{fail}$ is increased by one and $\mathtt{src}$ is added to the set $\mathtt{active}$. In this case, the station that receives or overhears the packet, has not received any transmission from $\mathtt{src}$ during the current scheduling period. Otherwise, If $\mathtt{src}$ belongs to the set $\mathtt{active}$, $\mathtt{success}$ is increased by 1. Recall that the set $\mathtt{active}$ is reset every $\mathtt{period}$ seconds.

Ratio of $\mathtt{success}$ to $\mathtt{success}$+$\mathtt{fail}$ is then considered to adapt p. If $\mathtt{ratio}$ is larger than a threshold, $\mathtt{maxRatio}$, and enough number of transmissions have happened (i.e., $\mathtt{success}+\mathtt{fail} >= \mathtt{maxNum}$) $\mathtt{p}$ is increased by δ and $\mathtt{success}$ and $\mathtt{fail}$ are reset to 0. We note that $\mathtt{p}$ is increased up to a threshold, $\mathtt{maxP}$. It is reasonable to choose $\mathtt{maxP}$ less than 1 in order to always give a chance to stations not in $\mathtt{active}$ to be able to transmit on the channel. If $\mathtt{ratio}$ is less than a threshold, $\mathtt{minRatio}$, while enough number of transmissions have happened (i.e., $\mathtt{success}+\mathtt{fail} >= \mathtt{maxNum}$), $\mathtt{p}$ is decreased by δ and $\mathtt{success}$ and $\mathtt{fail}$ are reset to 0. 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 protocol parameters.

III-D.5 Adapt

```
if $src \notin active$ then
    $fail$ ++
    add $src$ to $active$
else
    $success$ ++
if ($success + fail >= maxNum$) then
    $ratio = success / (success + fail)$
    if ($ratio >= maxRatio$) then
        if ($p <= maxP$) then
            $p = p + \delta$
        $success = 0$
        $fail = 0$
    if ($ratio <= minRatio$) then
        if ($p >= \delta$) then
            $p = p - \delta$
        $success = 0$
        $fail = 0$
```

IV. Evaluation

We simulate Token-DCF and 802.11g [16] in ns-2 to measure and compare performance of these two MAC protocols. The network is a wireless ad hoc network in which transmitting stations are placed uniformly at random in a square area. Flows might be single hop or multi hop. In the case of single hop flows, the receiver of each flow is placed at a distance of $100m$ from the transmitter of the flow. This means that if transmitter is placed at point (x, y), receiver is placed at point $((x + 100) \bmod d, y)$, where d is area length. In case of multi-hop flows, receiving stations are placed uniformly at random in the area. 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. Each simulation lasts for 30 seconds and the presented results are averaged over 5 runs. In each run, a different random network topology is considered. We measure the results of Token-DCF and 802.11 DCF in terms of aggregate throughput, average access delay, channel idle time and collision frequency. Table I reports the configuration parameter values of the wireless network analyzed in this section. Table II reports the parameter values of Token-DCF chosen in the simulations.

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

TABLE I: WLAN configuration

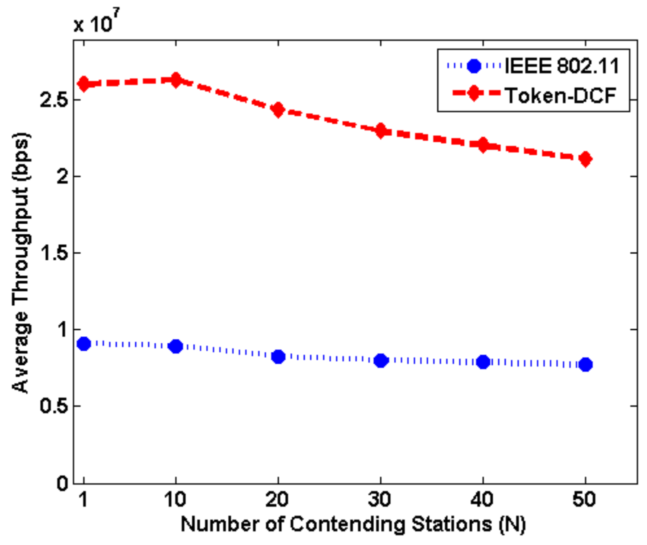

Fig. 3: System throughput (area=150mx150m, packet size=500B)

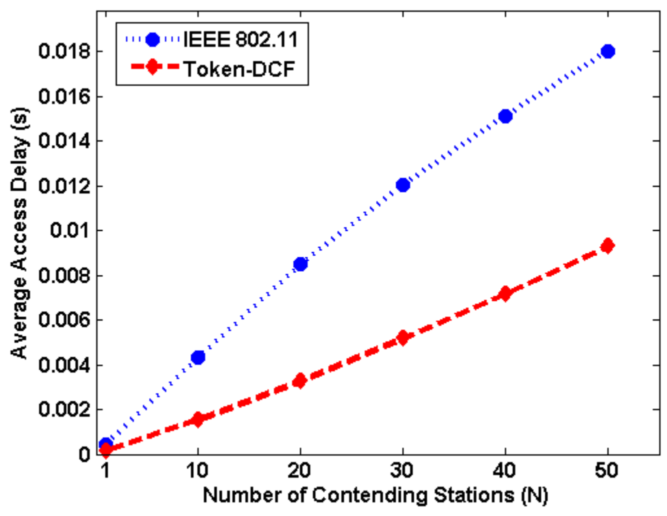

Fig. 4: Average access delay (area=150mx150m, packet size=500B)

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

TABLE II: Token-DCF parameters

A. *Performance evaluation in single-hop networks*

Figures 3-8 plot the performance parameters in a single-hop network. The size of the network is $150m$x$150m$. Traffic is full buffer CBR, meaning that there is always backlogged traffic in the transmission queue of each link. Transmission queue of each link holds up to 50 packets and when the buffer is full, newly arrived packets get dropped. The payload size is 500 bytes and all flows are single-hop.

The aggregate throughput of 802.11 DCF and Token-DCF is presented in Figure 3. As we can see, throughput gain obtained by Token-DCF compared to IEEE 802.11 in Figure 3 is a factor of $2.7-2.9$. 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 $0.35 - 0.51$. 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 medium.

Collision frequency of 802.11 DCF and Token-DCF is 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 much lower collision frequency than 802.11 DCF. 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. Recall that when a station transmits, it might choose one of its neighbors as the privileged station. 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 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.

Figures 7 and 8 show the throughput and access delay versus number of transmitters in a network of size $150m$x$150m$ where the packet size is 1500 bytes. Traffic type is full buffer CBR. As we can see in these figures, throughput gain is about $1.7 - 1.9$ and access delay is reduced by a factor of $0.53 - 0.81$. When packet size is 1500 bytes, the throughput gain is lower since the efficiency of DCF increases with packet

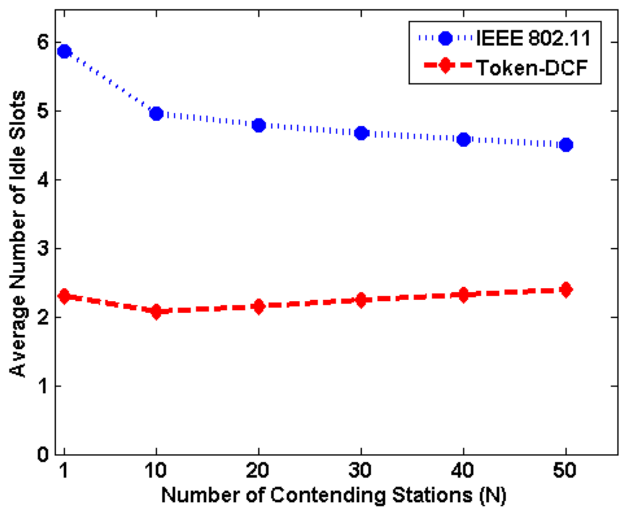

Fig. 5: The average number of idle slots before each media access (area=150mx150m, packet size=500B)

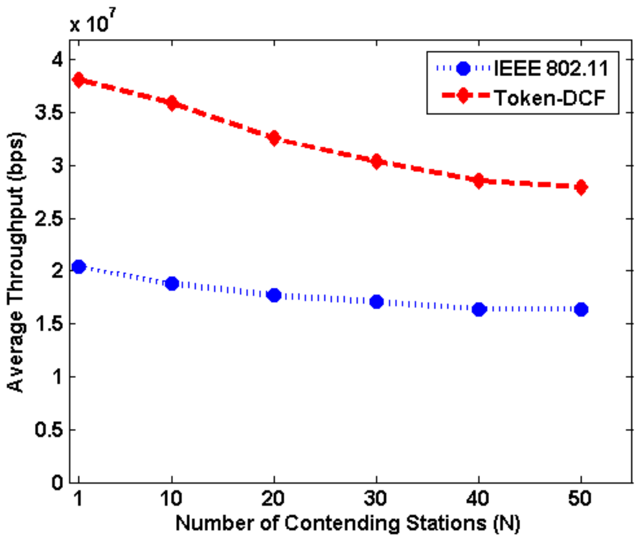

Fig. 7: System throughput (area=150mx150m, packet size=1500B)

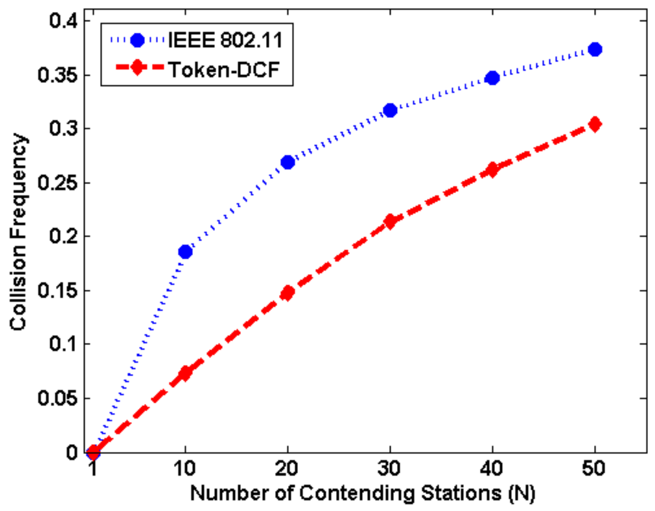

Fig. 6: Collision frequency (area=150mx150m, packet size=500B)

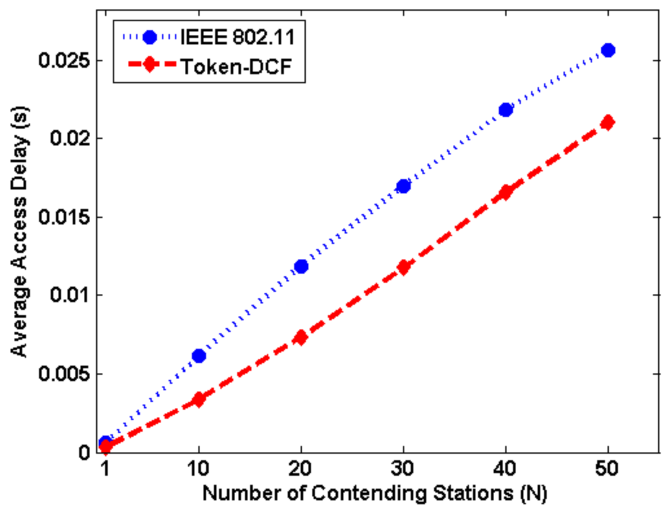

Fig. 8: Average access delay (area=150mx150m, packet size=1500B)

size. The overhead per successful packet transmission, T_{oh}, is equal to the sum of the channel idle time and collision time. We denote the packet transmission time by T_{tr}. T_{tr} is equal to the sum of DIFS, transmission time of the data packet, SIFS and transmission time of the acknowledgement. Then, the efficiency of DCF can be defined as

$$efficiency = \frac{T_{tr}}{T_{oh} + T_{tr}} \qquad (3)$$

T_{oh} is not a function of the packet size and does not change when packet size changes. On the other hand, T_{tr} increases when packet size increases. This results in larger *efficiency* when packet size is 1500 bytes.

B. Performance evaluation in 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. The payload size is 1500 bytes. System throughput and access delay of the networks with size $800m$x$800m$ versus number of contending stations are presented in Figures 9 and 10, 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$1500m$, system throughput and access delay

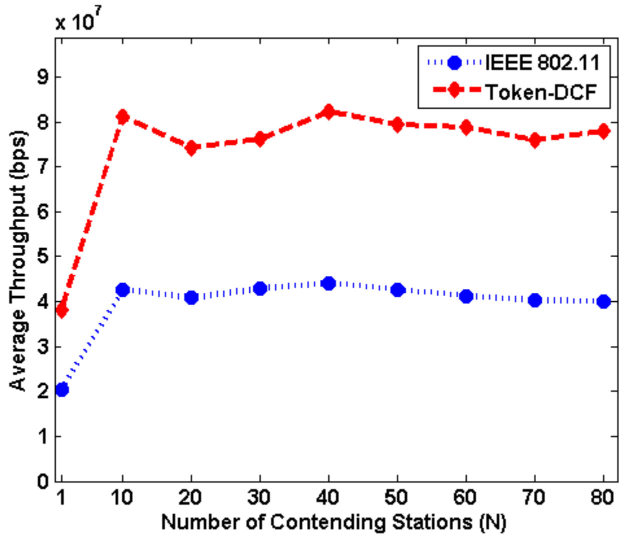

Fig. 9: System throughput (area=800mx800m)

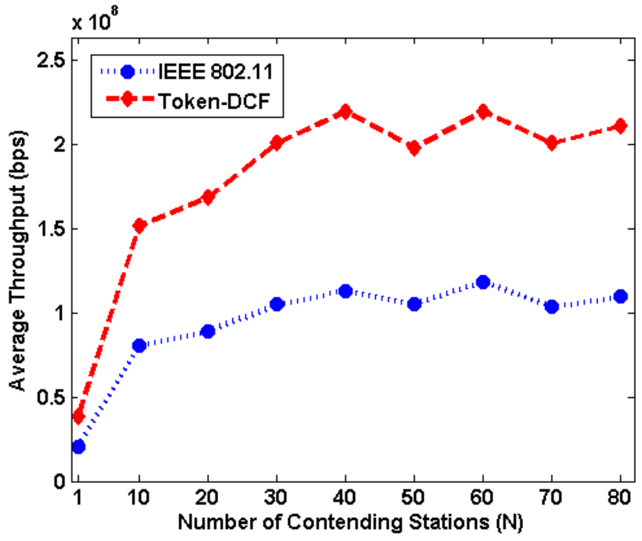

Fig. 11: System throughput (area=1500mx1500m)

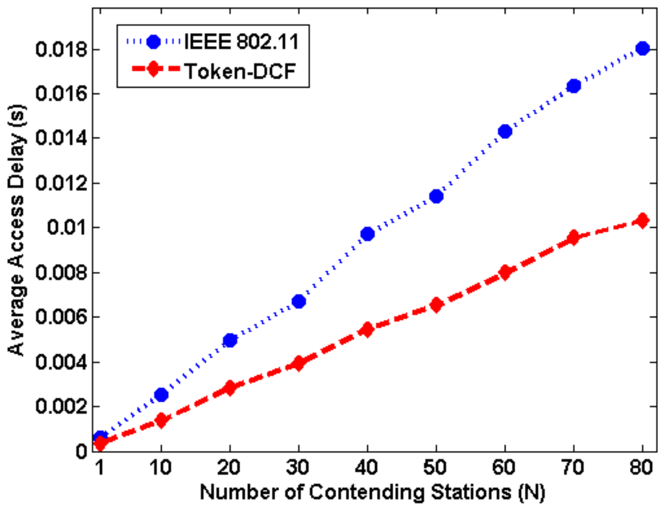

Fig. 10: Average access delay (area=800mx800m)

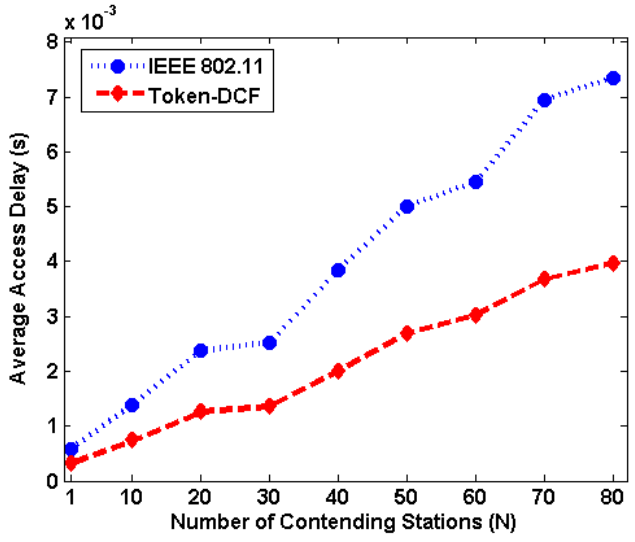

Fig. 12: Average access delay (area=1500mx1500m)

are presented in Figures 11 and 12, 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 7-12, we see that similar performance improvement is obtained by Token-DCF in single hop and multihop networks. In multi-hop networks, Token-DCF improves the channel utilization in each transmission range.

V. 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. The purpose of 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. Configuration parameters are as follows; Packet size is 1500 bytes. Average on time for generator is $50ms$. Average off time for generator is also $50ms$. 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 and the packet arrival rate of each station during on time is Rate. Rate (sending rate during on time) is varied between 10^3 bps and 10^8 bps. With Rate $= 10^3$ bps and 1500 byte 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 aggregate throughput and average access delay are presented in Figures 13 and 14, respectively. When

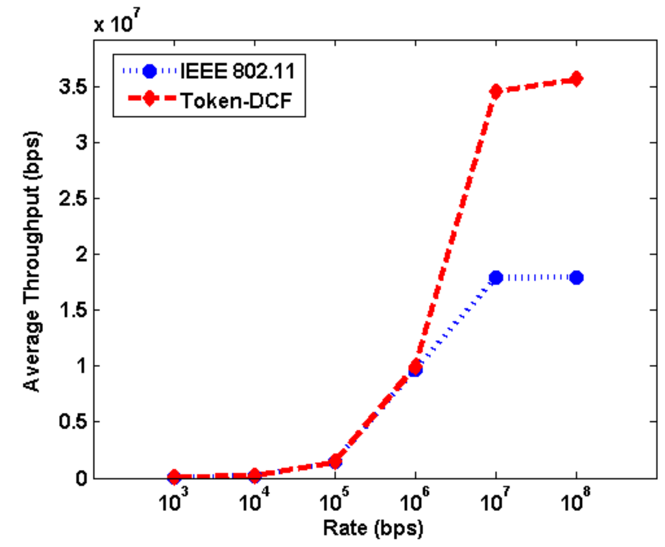

Fig. 13: System throughput (Pareto traffic)

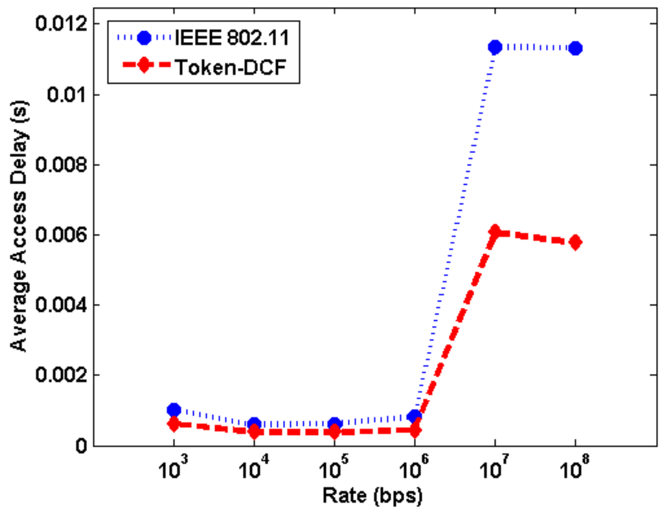

Fig. 14: Average access delay (Pareto traffic)

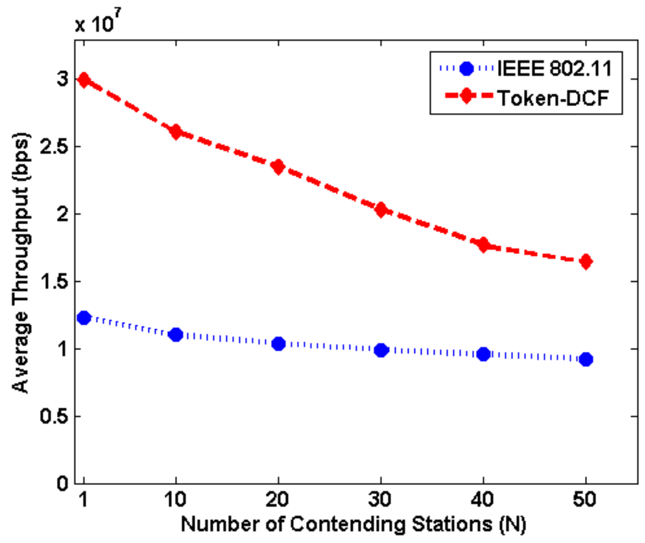

Fig. 15: System throughput (TCP traffic, area=150mx150m)

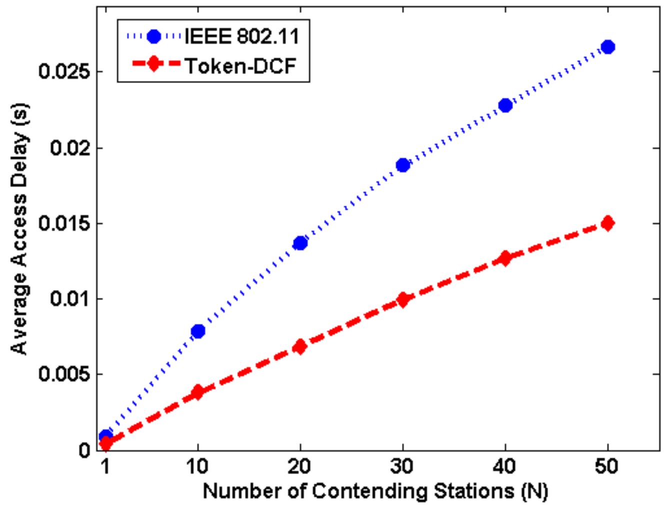

Fig. 16: Average access delay (TCP traffic, area=150mx150m)

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.

VI. Networks with TCP traffic

In this section, we study Token-DCF's performance under TCP traffic. We perform simulations for networks of different sizes, i.e., $150m$x$150m$, $800m$x$800m$ and $1500m$x$1500m$. Packet payload size is 1500 bytes. Transmitting and receiving stations of each flow are placed uniformly at random in the area. As a result, for networks of size $800m$x$800m$ and $1500m$x$1500m$, where network is multi-hop, flows are also multi-hop. DSDV (Destination-Sequenced Distance-Vector Routing) is used as the routing protocol. Figures 15 and 16 show the total throughput and average access delay for single-hop networks of size $150m$x$150m$. As presented in these figures, throughput gain is a factor of $1.8 - 2.4$ and access delay is reduced by a factor of $0.42 - 0.56$. Comparing Figures 7 and 15, we can see that the performance improvement of Token-DCF over 802.11 is similar for both TCP and saturated CBR traffic. When traffic is TCP, although buffer of stations might not be fully backlogged, stations might have few packets backlogged in their transmission queue, in which case privilege can be given to one of the stations. This results in decreasing the idle time and increasing the throughput.

The results for network size of $800m$x$800m$ are shown in

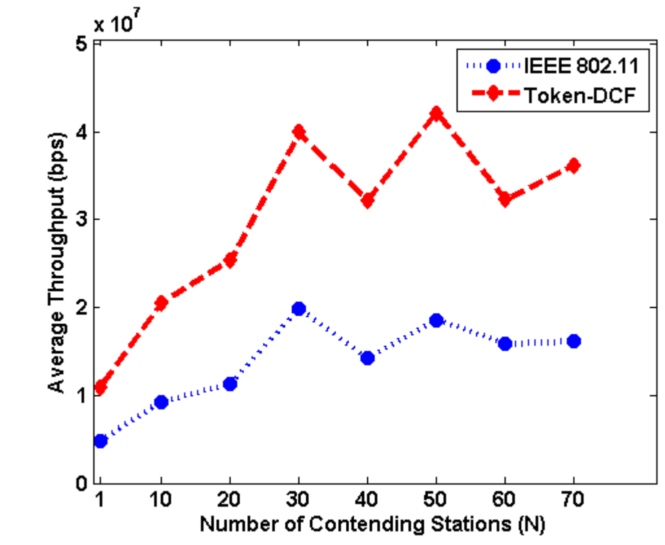

Fig. 17: System throughput (TCP traffic, area=800mx800m)

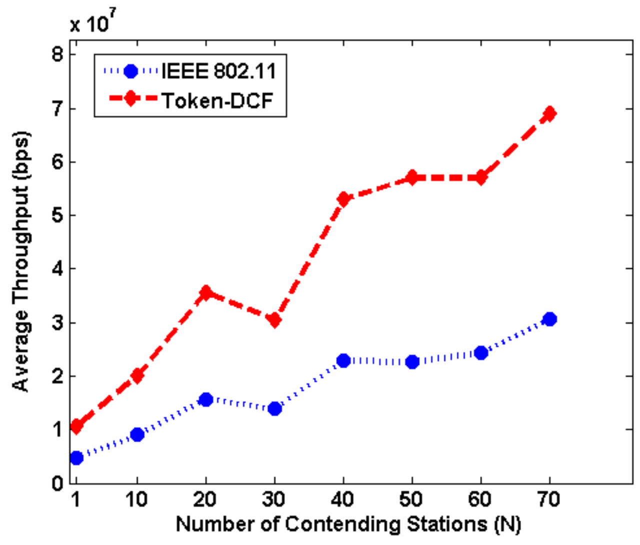

Fig. 19: System throughput (TCP traffic, area=1500mx1500m)

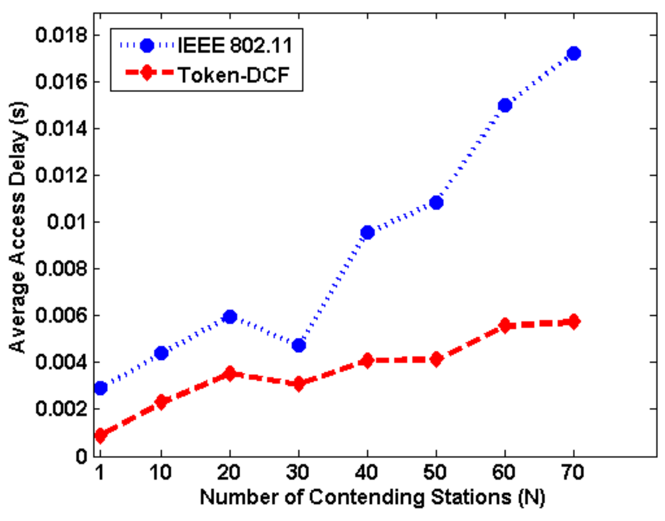

Fig. 18: Average access delay (TCP traffic, area=800mx800m)

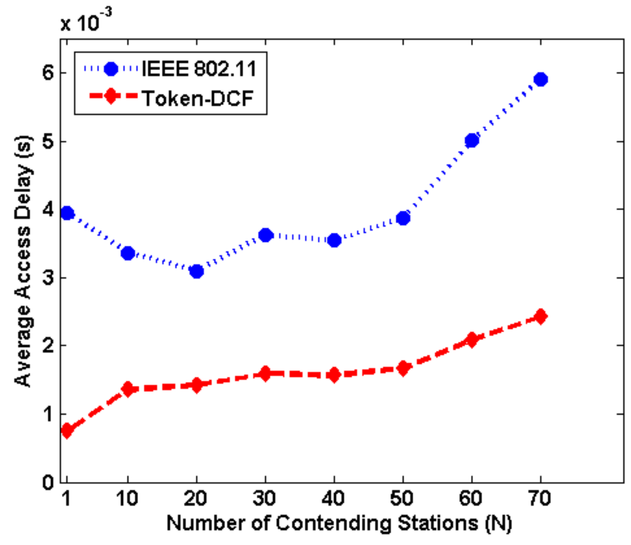

Fig. 20: Average access delay (TCP traffic, area=1500mx1500m)

Figures 17 and 18. In these networks, throughput gain is a factor of $2-2.3$ and access delay is reduced by a factor of $0.3-0.65$. Figures 19 and 20 present total throughput and average access delay for networks of size $1500m$x$1500m$. Throughput gain is a factor of $2.2-2.5$ and access delay is reduced by a factor of $0.18-0.45$. Although flows are multi-hop in these networks, since Token-DCF improves the channel utilization in each transmission range, total throughput gain and delay reduction is similar to single-hop networks.

VII. Conclusion

This paper presents the design and performance evaluation of Token-DCF. Token-DCF is a distributed media access protocol that uses an overhearing mechanism to schedule network stations for transmission on the wireless medium in an efficient manner. The design goal of Token-DCF is to reduce both idle time and collision time. Our simulation results show that Token-DCF significantly improves the performance in terms of system throughput and access delay.

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] H. Kim and J. Hou, "Improving protocol capacity with model-based frame scheduling in ieee 802.11-operated wlans," in *Mobicom,* 2003.

[3] 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.

[4] 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.

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

[6] 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.

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

[8] “IEEE Standards: P802.5 Working Group Area,” *Ieee802.org*.

[9] 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.

[10] 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.

[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] L. Jiang and J. Walrand, “A distributed CSMA algorithm for throughput and utility maximization in wireless networks,” *46th Annual Allerton Conference on Communication, Control, and Computing,* Page(s):1511 - 1519, September 2008.

[14] J. Ni and R. Srikant, “Distributed CSMA/CA algorithms for achieving maximum throughput in wireless networks”, *Information Theory and Applications Workshop,* Page(s):250 - 261, February 2009.

[15] T. H. Kim, J. Ni, R. Srikant and N. H. Vaidya, “On the achievable throughput of CSMA under imperfect carrier sensing”, *In Proc. IEEE INFOCOM,* April 2011.

[16] “IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” *IEEE Computer Society,* 2007.