Overheard ACK With Token Passing: An Optimization to 802.11 MAC Protocol [*]

Shegufta Bakht Ahsan
Department of Computer Science
University of Illinois at Urbana-Champaign
sbahsan2@illinois.edu

Nitin Vaidya
Department of Computer Science
University of Illinois at Urbana-Champaign
nhv@illinois.edu

ABSTRACT

Distributed Coordination Function (DCF) is defined in IEEE 802.11 standard, which is widely used in practice. Despite of its wide use, it has several limitations. Because of the idle and collision times, it suffers from poor channel utilization. Besides, the control packets, particularly, *Acknowledgement (ACK)*, consume non-trivial amount of bandwidth. Though the number of control bits in an ACK frame is small, the added overheads like the *preamble*, *packet header* etc. make the situation worse. In this paper, we propose a scheme called *Overheard ACK* where the *explicit* ACK frame has been eliminated by using the leverage of *packet overhearing.* Also, by incorporating *explicit* and *implicit* token-passing, this protocol attempts to schedule transmissions without having to use random access, dramatically reducing the idle time and collision time. Simulation results using *NS2* confirm that this protocol significantly outperforms the conventional 802.11 DCF.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Protocol architecture (OSI model)

Keywords

Packet Overhearing; Token Passing

1. INTRODUCTION

IEEE 802.11 is a standardized protocol for wireless networks. This protocol uses the Distributed Coordination Function (DCF) as its primary Medium Access Control (MAC) method [1]. To resolve channel contention, DCF employs Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) scheme with binary exponential backoff algorithm. DCF imposes two kinds of overheads; one is the channel idle time and the other is the collision overhead. Also, in DCF, a node transmits an ACK packet in reply to a received DATA packet, which also adds to the overhead (each MAC frame also has the overhead of the *preamble* and *frame header*).

This paper presents a solution that incorporates *Overheard ACK* and *token-passing* to significantly reduce the overhead due to *backoff time*, *packet collision* and *explicit ACK frame.* The key features of our solution are as follows:

- Acknoledgements are piggybacked on data frames when possible (similar to transport layer TCP protocol). Using the leverage of packet overhearing, this approach significantly reduces the overhead of sending explicit ACK packets.
- Each ongoing transmission passes a *token* to another host who will then have a higher priority to transmit after the ongoing transmission completes. Thus, the token-passing mechanism schedules transmissions without having to perform random access. When the host to which a token is passed does not have any pending packet, the protocol falls back on standard 802.11 DCF protocol. Thus, the proposed protocol is resilient to loss of tokens. Additionally, the protocol also gracefully handles situations in which spatial reuse of the channel may result in multiple tokens being present in the system.
- We combine the piggybacked ACK and token-passing into a single mechanism to yield the benefits of both approaches.

In this scheme, wireless nodes can be of two types, station (STA) and access point (AP).

2. RELATED WORK

The protocol presented in this paper is build on our prior work [3], in which we developed *Token-DCF*, an opportunistic MAC protocol for wireless networks, which reduces idle time and collision time by implementing an opportunistic token passing mechanism. In this scheme, with a certain probability p, a transmitting node might select one of its neighbours as a privileged node by mentioning that neighbour's ID in a token. By overhearing, all neighbours receive the token. Only the *privileged* neighbour mentioned in the token starts transmitting after SIFS interval of time. All other

[*]This research is supported in part by National Science Foundation awards CNS-11-17539 and Futurewei Technologies. 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.

S3'14, September 7, 2014, Maui, Hawaii, USA.
Copyright 2014 ACM 978-1-4503-3073-2/14/09 ...$15.00.
http://dx.doi.org/10.1145/2645884.2645886 .

non-privileged neighbours have to wait for *DIFS+backoff* time to access the channel. Thus whenever a privileged node transmits on the channel, the idle time of the channel is reduced to SIFS. The channel access method is shown in Figure 1.

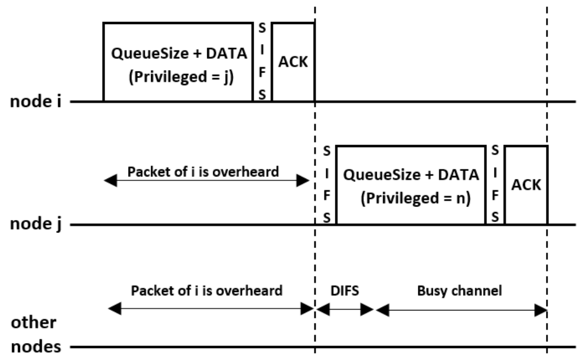


Figure 1: Channel Access Method of Token-DCF protocol

Token-DCF is implemented in the MAC layer, and the token is embedded in the MAC header. Each station continuously adds its current queue length in the MAC header of the outgoing packet. By overhearing, each station gets the latest queue size of its neighbour. Scheduling algorithm uses these information to select one of the neighbours as a privileged station. In the next transmission, privileged station's ID is passed with a token.

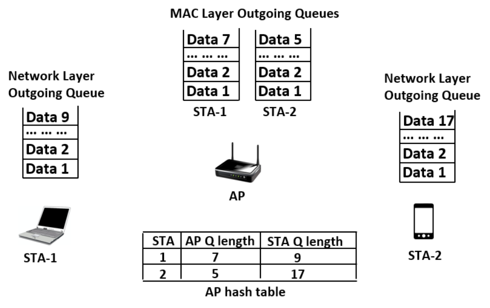
The protocol presented in this paper augments token-passing with piggybacked ACKs, and we develop a unified mechanism that benefits from both the optimizations.

In other related work, Choudhury et al. propose an "implicit MAC acknowledgement" scheme [2] where the explicit ACK frame has been eliminated by piggybacking ACK information in RTS/CTS frame. As a result in this protocol, RTS/CTS frames must precede each data frame, increasing the overhead. In contrast, for our mechanism, RTS/CTS exachange is optional. Also, according to the scheme in [2], to be able to send an implicit-ack, a sender node must have at least two packets destined for the same receiver.

3. PROTOCOL DESCRIPTION

Overheard ACK is implemented in the MAC layer of the protocol stack. In this protocol, each STA embeds its latest network layer packet queue length in the MAC header of the transmitted packet. AP maintains a database where queue length of each neighbour is stored, based on which AP *explicitly* or *implicitly* selects a privileged STA. Instead of waiting for *DIFS+backoff* time, a privileged STA can access the channel just after *SIFS* time. We describe our protocol as a sequence of events at the MAC layer.

Figure 2 presents a simple scenario with an AP and two station STA_1 and STA_2 with queue length 9 and 17 respectively. First we will describe the case where *implicit scheduling* is used. Let's assume that for all STAs, AP has always at least one data packet to send. When AP receives a DATA from STA_1, it schedules an ACK after SIFS interval. The larger queue size of STA_2 implies that it has more data to send to AP. Our simple scheduling algorithm, which selects the neighbour with the maximum queue length as a privi-



STA	AP Q length	STA Q length
1	7	9
2	5	17

Figure 2: Queue design of Overheard ACK

leged node, selects STA_2 as privileged. Therefore AP selects a DATA packet destined for STA_2 and merges it with ACK. The combined packet now contains *ACK*, ACK_{dest}, *DATA* and $DATA_{dest}$. After SIFS interval, instead of sending the ACK to STA_1, AP sends the merged packet to STA_2. STA_1 receives the combined packet by overhearing and parses the desired ACK. STA_2 receives the packet directly and parses the DATA. At this point, after SIFS interval STA_2 schedules an ACK. But it has at least one DATA packet destined to AP, waiting in the network layer queue. Therefore it merges the ACK with the DATA packet. After SIFS interval, it sends the combined packet to AP. Now AP receives the desired ACK and another new DATA from STA_2. As like before, its scheduler module selects a new privileged STA (it may be STA_2 again), merges the ACK and DATA packet, and sends the merged packet to the privileged STA. In this way AP can *implicitly* select a privileged STA. This scheme avoids sending explicit ACK by merging it with the DATA packet. Most of the time the *merged packet* is the only type of packet transmitted in the channel. In this approach, backoff stage does not occur before each packet transmission, rather there is only SIFS interval between two consecutive transmissions, which results in better throughput than the *Token-DCF* scheme.

Now we will consider the second case, where AP *explicitly* schedules a STA. It can be further divided into two situations where the traffic patterns are mainly: *a*) *up-stream* data and *b*) *down-stream* data. In the first situation, AP has the knowledge about each STA's queue size, based on which it can select a privileged STA. As most of the time AP has no data to send, it returns *explicit ACK frame* in response to the incoming DATA from STAs. But this time the *ID* of the privileged station is explicitly embedded with the transmitted ACK frame. As all the STAs overhears the ACK frame, only the privileged STA schedules the next transmission after SIFS interval. Though this scheme cannot eliminate explicit ACK, but here packets are transferred just after SIFS interval. Hence its throughput is similar to the *Token-DCF* scheme, which is still better than the conventional IEEE 802.11 DCF scheme.

In the second situation, STAs send only the MAC layer ACK to AP. In this kind of scenario AP plays the role as a transmitter. Therefore AP can explicitly announce itself as a privileged node. After receiving an ACK from STA, instead of waiting for *DIFS+backoff* interval, AP waits only for SIFS interval and transmits the next data. Its throughput is also similar to the *Token-DCF* scheme.

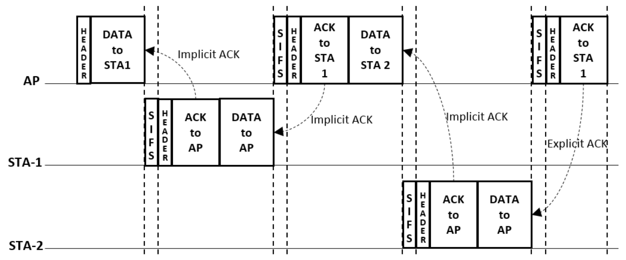


Figure 3: Channel access method of Overheard ACK

In this protocol, network layer queue of the AP has been replaced by a number of dedicated MAC layer queues for each STA. Each queue associated with a STA, holds outgoing packets destined to that STA. AP maintains a hash table (Figure 2) where against each STA, the number of outgoing packet inside AP's dedicated queue and the number of outgoing packets inside that STA's network layer queue are stored. AP uses this information for scheduling the next transmitting station. The channel access mechanism of *Overheard ACK* is shown in Figure 3.

4. EVALUATION

To measure and compare performance, we have simulated *802.11g*, *Token-DCF* and *Overheard ACK* in NS-2. The network is a wireless infrastructure network where nodes have been placed randomly in a $120mX120m$ square area. In our experiment IEEE 802.11 RTS/CTS mechanism is turned off. Channel bandwidth is $54Mbps$ and packet payload size is 1500 bytes. Each simulation has been run for 10 seconds and the presented results have been averaged over 20 runs. In each run, a different random network topology has been created. We have used the default *802.11g* simulation given in NS-2. For *Token-DCF* the value of p has been set to 0.95. In *Overheard ACK* the AP always selects a privilege station, i.e. in that case the value of p is set to 1.0.

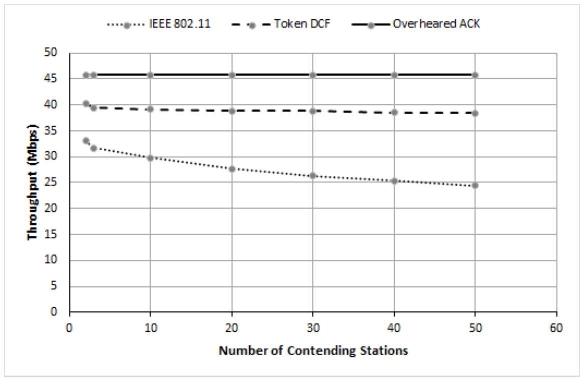


Figure 4: Aggregate throughput, UDP two way traffic

Figure 4 plots the throughput for two way UDP traffic, where both AP and STAs transfer DATA and ACK. Figure 5 plots the throughput for TCP one way traffic. Because of the TCP ack, the actual traffic becomes two way.

5. COMPARISON WITH TOKEN-DCF

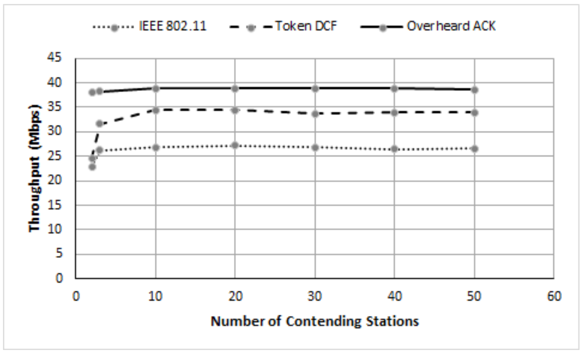


Figure 5: Aggregate throughput, TCP one way traffic

This protocol has several advantages over *Token-DCF*

- Unlike *Token-DCF*, where each station acts as a scheduler, in *Overheard ACK* only the AP takes the scheduling decision, which guarantees more precise scheduling.
- In *Token-DCF*, each node has to continuously listen to the channel to get the queue lengths of its neighbours, which consumes extra power. Whereas In *Overheard ACK*, only AP requires its neighbour stations' queue lengths, which it can get directly.
- In *Token-DCF*, as each node acts as a scheduler, their scheduling module need to continuously analyse the incoming queue length, which consumes extra memory, CPU time and power. Whereas in *Overheard ACK*, only the AP does the complex calculation. Instead of increasing the CPU and memory of each STA, increasing it only in the AP is more practical. Therefore AP can easily overcome the burden of the extra computation.

6. CONCLUSION

In this paper we present *Overheard ACK*, a MAC protocol which by employing packet overhearing and eliminating explicit ACK frames, improves the channel utilization. Primarily we are using a simple scheduling algorithm, which selects each station with equal probability. Our simulation results show that this protocol outperforms the *Token-DCF* protocol. The encouraging results lead us to extend this research for making it more robust and efficient.

7. REFERENCES

[1] *IEEE Std. 802.11-2077, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, June 2007.

[2] R. R. Choudhury, A. Chakravarty, and T. Ueda. Implicit mac acknowledgment: An improvement to 802.11. IEEE/ACM Wireless Telecommunications Symposium (WTS), April 2005.

[3] G. Hosseinabadi and N. H. Vaidya. Token-dcf: An opportunistic mac protocol for wireless networks. In *COMSNETS*, pages 1–9, 2013.