O-ACK: An Adaptive Wireless MAC Protocol Exploiting Opportunistic Token-Passing and Ack Piggybacking

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

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

***Abstract*—Bandwidth is a shared and limited resource of the wireless LAN. The MAC protocols are mainly responsible for efficiently sharing it among all the stations. The IEEE 802.11 standard uses Distributed Coordination Function (DCF) as its default MAC protocol which employs Carrier Sense Multiple Access using Collision Avoidance (CSMA/CA) scheme with binary exponential backoff algorithm. Despite its ubiquitous acceptance, it has two major drawbacks: *i)* channel idle time and *ii)* collision overhead. Besides, this protocol uses *explicit ack* which comes with significant overheads, for example *preamble*, *frame header* etc. In this paper, we propose a scheme called *O-ACK* which reduces the overhead of explicit *ack* and backoff interval by leveraging *piggybacking*, *packet overhearing* and *token based scheduling*. Our *NS2* based simulation results confirm that this protocol significantly outperforms the DCF protocol.**

***Index Terms*—Packet Overhearing, Token Passing, Multiple Access Point, Fairness.**

I. Introduction

IEEE 802.11 is a widely used wireless protocol where Distributed Coordination Function (DCF) is used as the primary Medium Access Control (MAC) method [3]. DCF imposes two major overheads: *i)* channel idle time and *ii)* collision overhead. Besides it uses *explicit ack* which causes significant transmission overhead due to the *preamble*, *frame header* etc.

This paper presents a protocol named Overheard-ACK *(O-ACK)* which incorporates *piggybacking*, *packet overhearing* and *token based scheduling* to significantly reduce the overhead due to *backoff time*, *packet collision* and explicit *ack*. This protocol is primarily designed for the infrastructure mode. In this scheme, wireless nodes can be of two types, station *(STA)* and access point *(AP)*. The key features of our solution are:

- *Use of piggybacked ack and packet overhearing.*
- *Use of implicit/explicit token to schedule transmission.*
- *Dynamically adjusting the chain creation process.*

The rest of the paper is organized as follows. In Section II we first review the related works. Section III describes the *O-ACK* protocol which is followed by the evaluation in Section IV. Finally, Section V concludes the paper.

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

II. Related Work

Researchers have proposed various MAC protocols to improve the efficiency of the 802.11 DCF [3]. In [5] Cali et al. propose a distributed algorithm that tunes the backoff algorithm of each station at run-time and adjusts the contention window such that the throughput gets maximized.

In [9] Tay et al. consider a network where N stations become simultaneously backlogged at some point in time and derive the optimal backoff distribution p^* for a non-persistent CSMA protocol where each station chooses a contention slot according to p^*.

Zeng et al. propose *CHAIN* [10] where each client maintains a *precedence relation* with other clients. After overhearing a successful transmission of the predecessor, a client can immediately access the channel. When the traffic is low, this protocol behaves similar to the conventional DCF. When the network becomes congested, clients automatically start transmitting chains to improve efficiency. This protocol periodically sends control packets between AP and STAs, which adds extra overhead.

IEEE 802.11e [2] introduces a new idea called transmission opportunities *(TXOPs)* where a station that gains access to the channel can transmit multiple frames separated by a *SIFS* interval.

Both *DOMINO* [11] and *CENTAUR* [8] propose a solution for large-scale Enterprise WLANs. *CENTAUR* is optimized only for down-link traffic. *DOMINO* assumes that the *conflict graph* of network links remains same over time, which does not hold in mobility scenarios.

In other related work, Choudhury et al. propose an "implicit MAC acknowledgement" scheme [6] where the explicit *ack* frame has been eliminated by piggybacking *ack* information in RTS/CTS frame.

A lot of proposals incorporate the idea of Token passing to improve performance. Its advantage over contention based medium access is that it eliminates collision by reserving the channel for a particular station. The IEEE 802.4 Token Bus protocol [1] is a well-known example of token passing protocols, which is developed for wired network.

978-1-4673-6770-7/15/$31.00 ©2015 IEEE

The protocol presented in this paper is built on our prior work TOKEN-DCF (*T-DCF*) [7]. *T-DCF* is an opportunistic MAC protocol for wireless networks, which reduces idle time and collision time by implementing an opportunistic token passing mechanism.

III. O-ACK PROTOCOL

The protocol "Overheard ACK with token passing" integrates token-passing with piggybacked *acks*. We presented a preliminary version of this protocol at a student research workshop [4]. Normally in *O-ACK* there is only *SIFS* interval between two consecutive transmission which is called as the *transmission chain*. Though it dramatically increases throughput, but as a side effect a new station finds it difficult to join the network. The situation gets worse in places where the transmission ranges of two or more APs overlap. In our current work, we have introduced several new techniques to handle fairness, to control *transmission chain* and to support multiple *AP* scenario.

Figure 1 illustrate the channel access method of *O-ACK* protocol. It is implemented in the MAC layer of the protocol stack. In this protocol, when *AP* sends *data* (say, $data_1$) to STA_1, it adds a flag $request\_merge$ in the MAC header of the outgoing packet. After successfully receiving $data_1$, STA_1 schedules transmission of an acknowledgement (ack_{ap}) after *SIFS* interval. If the $request\_merge$ flag is turned on and STA_1 has up-link *data* ($data_{ap}$), it merges that *data* and the *ack* into a single packet $(ack_{ap} + data_{ap})$ and transmits it after *SIFS* interval. When *AP* receives the merged packet, it parses ack_{ap}. It then processes $data_{ap}$ and schedules an acknowledgement (ack_1) transmission after *SIFS* interval. But like the previous case, if its outgoing queue is not empty, it selects a *data* packet (say $data_2$ destined for STA_2), merges it with the ack_1 into a single packet ($ack_1 + data_2$), and finally transmits the merged packet to STA_2 after *SIFS* interval. STA_1 overhears the packet and parses the desired ack_1. STA_2 receives the packet, extracts and processes $data_2$. In this protocol, when *AP* transmits a $data_R$ or $(ack + data_R)$ to any station STA_R, with a probability λ, it turns on the flag $request\_merge$ in the MAC header of the outgoing packet. If this flag is turned on and there is backlogged packet both in the *AP* and *STAs*, then there remains only *SIFS* interval in between two consecutive transmission, which forms a *transmission chain*. The protocol proceeds based on several events which are explained below:

A. Station receives a packet from AP

After successfully receiving a $data_R$ or $(ack + data_R)$ packet, if the receiving station STA_R finds the flag $request\_merge$ turned off, it schedules a normal ack_{ap} just after *SIFS* interval. If the flag is turned on, it considers two situations: *i)* It has $data_{ap}$ packet for sending to AP: in that case it merges the ack_{ap} and $data_{ap}$ packet and schedules it for transmission after *SIFS* interval. *ii)* It has no outgoing *data*: in that case it schedules a single ack_{ap} with a flag named $cannot\_merge$ turned on in its MAC header.

Station STA_R can also successfully receive an desired ack_R or any overheard ack packet from *AP*. If the acknowledgement contains an *explicit token* where the mentioned privileged ID is same as this station's ID, it schedules its next data transmission (if any) after *SIFS* interval.

B. AP receives a packet from a station

AP may receive two types of special packets from STA_R: *i)* A single ack_{ap} with $cannot\_merge$ flag turned on: it implies that though *AP* assigned the next time slot to STA_R, it cannot utilize that slot and returns the token to *AP*. Hence *AP* utilizes that time slot by scheduling its next *data* just after *SIFS* interval. *ii)* A merged $(ack_{ap} + data_{ap})$ packet: in that case *AP* receives the desired ack_{ap}. Also, as *AP* receives a new data packet from STA_R, it is obliged to send back an ack_R. When *AP* does not have any *data* in its outgoing queue, just like the conventional protocol, it sends an explicit ack_R to STA_R. Otherwise then with probability λ it selects a $data_V$ from the outgoing queue. *AP* merges $data_V$ with ack_R and finally schedules for transmission after *SIFS* interval. To maintain the *transmission chain*, *AP* turns on the $request\_merge$ flag with probability λ. *AP* takes default action for receiving other types of packets.

C. Handling explicit/implicit tokens

In this scheme, *AP* maximizes the channel utilization by *explicitly* or *implicitly* scheduling a transmitter for the next time slot. The scheduled transmitter is referred to as the *privileged station*. When *AP* sends *data* to STA_V, if the $request\_merge$ flag is turned on, it *implicitly* makes STA_V the *privileged station*. Hence STA_V merges its next *data* with the ack_{ap}. If *AP* has no outgoing data packet, it sends *explicit ack*. In that case, with probability λ, *AP* selects a station as *privileged* station and *explicitly* mentions its ID on the MAC header of the *explicit ack*. All stations inside its transmission range get the information about the *privileged* station. All non-privileged stations wait for *(DIFS + backoff)* interval to get access to the channel. Only the privileged station starts transmission just after *SIFS* interval. If that station does not have any data to transmit, it remains silent.

D. Adapt to the current channel condition

This section briefly explains the process where λ is dynamically adapted based on current channel conditions. In this protocol, each *AP* maintains two sets of stations: *i*) The *first set* contains stations from which *AP* successfully receives frame and are associated with this *AP*. This set is referred to as STA_{own}. *ii*) The *second set* contains stations from which *AP* successfully receives frames (by overhearing) but they are associated with some other *APs*. This set is referred to as $STA_{overheard}$. If AP does not hear from a station $R \in \{STA_{own} \cup STA_{overheard}\}$ for more than $stale\_node\_duration$ time, it removes that station from the corresponding set.

Each AP continuously maintains a ratio $|STA_{own}|/(|STA_{own}| + |STA_{overheard}|)$ also called

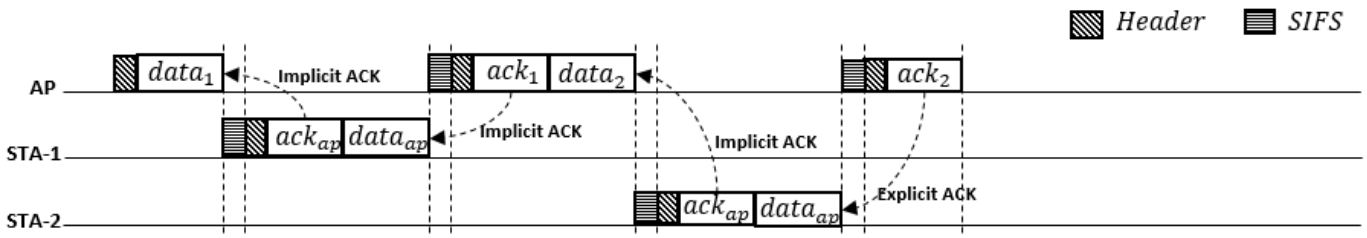

Fig. 1: Channel access method of O-ACK using an example traffic pattern. $data_n$/ack_n is the *data*/*ack* directed to STA-n.

$current\_max\_prob$ which is used as an upper bound of the calculated probability for that particular period of time. This upper bound ensures fair use of the channel. For example, when the set $STA_{overheard}$ is empty and all the stations are associated with the AP_a, $current\_max\_prob$ becomes 1 and AP_a alone utilizes the channel. On a different scenario, say, for example among total 10 *STAs*, $|STA_{own}| = 5$ stations belong to AP_a , and the rest 5 stations are associated with other *APs*, in that case $current\_max\_prob$ becomes 0.5, and AP_a tries to create the *transmission chain* with maximum 0.5 probability.

Based on the probability λ, this protocol swaps between the *DCF* and the *O-ACK*. Depending on the two *sets* described above, $current\_max\_prob$ may become 1 in which case AP remains in *0-ACK* mode for 100% of the time, hence continuously communicates among the current members of the set STA_{own}. As a result, a new station finds the channel always busy and cannot get a chance to transmit. Though in this case the overall throughput is higher, it prevents the new stations from transmitting. $global\_max\_prob$ is introduced to solve this problem. The upper limit of λ is set to $\min(current\_max\_prob, global\_max\_prob)$. In our experiment $global\_max\_prob$ was set to 0.8.

Only an *AP* runs the procedure to adjust the probability λ. Here, one variable $fail$ keeps count of the number of reception from a previously unseen station. Similarly another variable $success$ keeps track of the number of reception from an already seen station. When the total number of reception exceeds $maxNum$, the $ratio$ between number of reception from already seen station vs total number of reception is calculated. If the ratio is greater than $maxRatio$, λ is increased by δ. On the other hand, if the ratio is smaller than $minRatio$, λ is decreased by δ. In both case λ is adjusted so that it does not exceed the global upper or lower probability. The $current\_max\_prob$ is also calculated inside this algorithm.

Overheard ack employs several other techniques to ensure a fair channel utilization among multiple *APs*. *i)* If a station is in the overlapping region of more than one AP, because of the characteristics of the O-ACK protocol it may find the channel almost always busy. In that situation, if the station waits more than $starving\_duration$ time, it turns on a flag called $is\_starving$ and places it in the MAC header of the outgoing packet. Eventually, when it gets a chance to access the channel, it transmits the packet. All the APs in its transmission range get the $is\_starving$ flag and reduce their current λ by multiplying it with α (< 1.0). *ii)* If AP does not get an acknowledgement in time, it reduces λ by multiplying it with β (< 1.0). *iii)* If AP receives a corrupted packet, it reduces λ by multiplying it with γ (< 1.0).

IV. Evaluation

O-ACK protocol has been simulated in *NS-2* and compared with *802.11g DCF* and Token-DCF (*T-DCF*). We have performed several simulation on different scenarios. Due to space constrain here we are briefly explaining two of them.

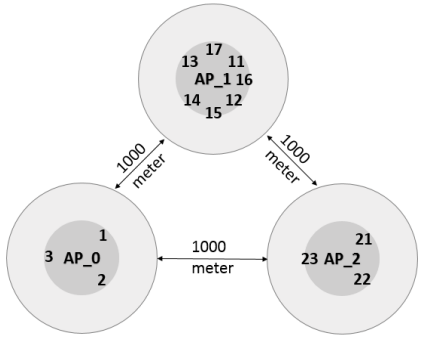

Fig. 2: Non-Overlapping Access Points. Dark circles are the transmission range and light circles represent the carrier sense range. The numbers represent the stations

The first scenario is shown in Figure 2. This scenario consist of three *APs* where each of them are in the same channel: AP_0 has three stations $1, 2, 3$, AP_1 has seven stations $11, 12, 13, 14, 15, 16, 17$ and AP_2 has three stations $21, 22, 23$. This simulation consists of TCP and UDP traffics and all stations except the four stations $14, 15, 16, 17$ start transmitting to their corresponding AP from the 0.0^{th} second. Those four stations are intentionally introduced in the network at 30.0^{th} second to check how efficiently they can get into the network. The second scenario is the same as the first one except here the transmission range of the three *AP's* are overlapped.

The graph presented in Figure 3 compares the throughput among three different protocols in two different scenarios. The transmission rate in a single collision domain is set to 54Mbps. Maximum network capacity is not achieved due to inefficiencies at the MAC and TCP layer, but the result show that, in both scenario *O-ACK* is the most efficient protocol. This happens because *O-ACK* protocol tries to maximize its channel utilization based on the surrounding environment. At

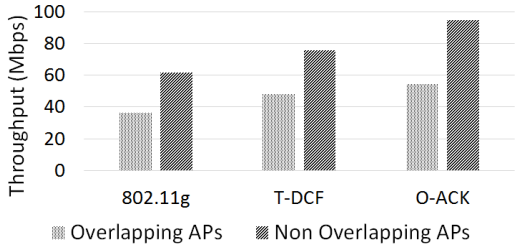

Fig. 3: Throughput of different protocols.

the same time, by carefully controlling the probability λ, it ensures that a newly arrived station finds enough empty slots in the channel.

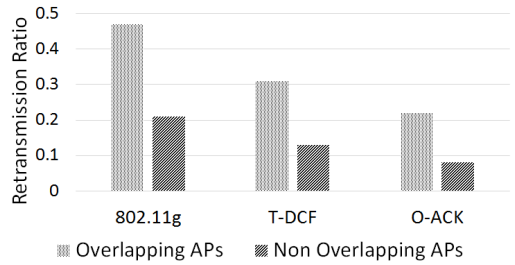

Fig. 4: Retransmission ratio of different protocols.

Figure 4 presents the ratio between re-transmission count and successful transmission count. In O-ACK protocol the next transmitting station is scheduled with probability λ. Hence as λ increases, the number of scheduled transmissions increases which reduces the packet loss due to collision. The 802.11 DCF protocol does not apply the idea of scheduling. Here packets wait for a random time, as the timer expires it starts transmission. This random backoff time is one of the main reasons of increased collision and thus increased re-transmission count.

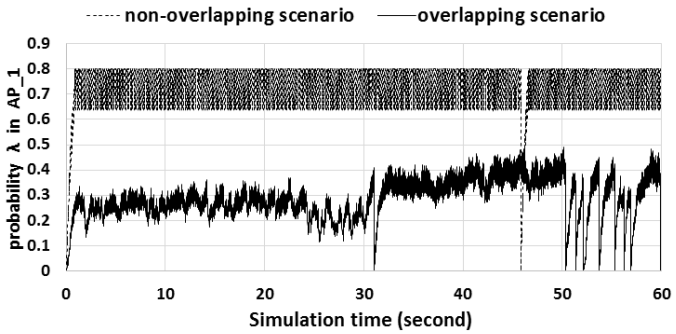

Fig. 5: Comparing the progress of λ in AP_1 between overlapping and non-overlapping scenario.

Figure 5 presents the progress of probability λ of AP_1 for both the overlapping and non-overlapping scenario. We first discuss the non-overlapping scenario where the set $STA_{overheard}$ is always empty, as a result $current\_max\_prob$ becomes 1.0. For this simulation the value of the $global\_max\_prob$ is set to 0.8, as a result the maximum possible value of λ is trimmed to 0.8. Almost all of the time λ fluctuates between 0.64 to 0.8. The reason behind such behavior is that, though λ reaches its maximum possible value, but still our protocol behaves like the conventional 802.11 protocol with 0.2 probability which creates packet loss. As explained earlier, in case of packet loss AP reduces its probability by multiplying it by β, which is set to 0.8, hence λ becomes $0.8 * 0.8 = 0.64$. At around 45.0^{th} second the *AP* starts four other data flows to the newly arrived stations, as a result the channel becomes temporarily congested. The sharp dip at around the 45.0^{th} second happens because of it.

In the case of the overlapping scenario, AP_1's $|STA_{own}| = 3$ for the first 30^{th} second. Station 14, 16, 17 and 18 starts communicating with AP_1 at 30^{th} second which make $|STA_{own}| = 7$ and also results in an increase in $current\_max\_prob$. This explains the rise in the graph after the 30.0^{th} second.

V. Conclusion

In this paper we present *O-ACK*, an efficient MAC protocol which improves the channel utilization by employing packet overhearing and eliminating explicit ACK frames. This protocol adjusts itself based on the surrounding environment. Our thorough simulation results show that this protocol outperforms the *DCF* and *Token-DCF* protocol. One advantage of this protocol is that, here λ is increased opportunistically to get extra gain over the conventional DCF protocol. If $\lambda = 0$, it becomes equivalent to the default *DCF* protocol. Hence the worst case performance of the *O-ACK* is similar to the DCF protocol. In our next step, we are planning to implement it in a test-bed. Right now all transmissions follow the same rate, which we are planning to replace by a rate-adaptive scheme.

References

[1] "Token-passing bus access method and physical layer specifications," *IEEE Standard 802.4-1985*, vol. 238.

[2] *Wireless medium access control (MAC) and physical layer (PHY) specifications: Medium access control (MAC) enhancements for quality of service (QoS)*, July 2003.

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

[4] S. B. Ahsan and N. Vaidya, "Overheard ack with token passing: an optimization to 802.11 mac protocol." ACM S3 Workshop, 2014.

[5] F. Cal, M. Conti, A. Member, E. Gregori, and A. Member, "Dynamic tuning of the ieee 802.11 protocol to achieve a theoretical throughput limit," *IEEE/ACM Transactions on Networking*, 2000.

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

[7] G. Hosseinabadi and N. H. Vaidya, "Token-dcf: An opportunistic mac protocol for wireless networks," in *COMSNETS*, 2013, pp. 1–9.

[8] V. Shrivastava, N. Ahmed, S. K. Rayanchu, S. Banerjee, S. Keshav, K. Papagiannaki, and A. Mishra, "Centaur: realizing the full potential of centralized wlans through a hybrid data path." in *MOBICOM*, 2009.

[9] 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*, 2004.

[10] Z. Zeng, Y. Gao, K. Tan, and P. R. Kumar, "Chain: Introducing minimum controlled coordination into random access mac." in *INFOCOM*. IEEE, 2011.

[11] W. Zhou, D. Li, K. Srinivasan, and P. Sinha, "DOMINO: relative scheduling in enterprise wireless lans," in *CoNEXT*, 2013.