Collision-Free Channel Access Using Transmission Queues with Fast Queue Joining

J.J. Garcia-Luna-Aceves
Electrical and Computer Engineering Department
University of Toronto
Toronto, ON, Canada
jj.garcialunaaceves@utoronto.ca

Sheideh Homayon
Computer Science and Engineering Department
University of California, Santa Cruz
Santa Cruz, CA, USA
shomayon@ucsc.edu

***Abstract*—QSMA-FAST (Queue Sharing Multiple Access with First Attempt with Success in a Turn) is introduced and analyzed. QSMA-FAST consists of establishing and maintaining a distributed transmission queue among nodes sharing a common channel and results in a sequence of queue cycles. Each queue cycle has two or more queue turns for collision-free transmissions from nodes that have joined the transmission queue, followed by a joining period. While a target queue size is not reached, a first-success technique is used during the joining period to expedite the queue-joining process. QSMA-FAST does not need to rely on any physical-layer mechanisms to establish transmission schedules; hence, it can be implemented using very simple transceivers. The performance of QSMA-FAST is compared against the performance of ALOHA with priority ACK's, CSMA with priority ACK's, CSMA/CD with priority ACK's, TDMA with a fixed schedule, and QSMA without the first-success technique.**

I. Introduction

Many Medium Access Control (MAC) protocols have been proposed over the years based on packet-by-packet contention for the channel or the provision of collision-free transmission schedules that are either static or derived dynamically [4].

MAC protocols that provide transmission schedules dynamically organize access to the channel based on transmission frames consisting of multiple time slots. The number of time slots per frame is fixed in many of these protocols and most of them require time slotting supported by synchronization techniques implemented at the physical layer. The dynamic allocation of time slots to transmitting nodes is attained in various ways. Some schemes use contention-based reservations (e.g., [16], [18], [20]), which requires nodes to engage in signalling that can result in long delays for successful reservations to occur. Other schemes are based on distributed elections (e.g., [2], [3], [14], [15]) that can incur long delays for nodes to elect those nodes that should be allowed to transmit in specific time slots. Another type of methods uses codes that determine the time slots in which different nodes may transmit (e.g., [5], [6]). The limitations of this approach is that channel sharing need not be fair and, in some cases, may render low throughput similar to that of slotted ALOHA. In all these cases, the channel is assumed to be organized into transmission frames of a constant number of time slots.

A different type of approach for collision-free channel access without the need for transmission frames of a constant length consists of using distributed queues (e.g., [10], [11], [13], [19]) to allocate transmission times to active nodes. In particular, QSMA [10] and ALOHA-NUI [11] enable channel access in a way that maintains much of the simplicity of ALOHA and CSMA with priority ACK's and attains collision-free transmissions and channel-access delay guarantees, eliminates clock synchronization, and eliminates fixed-length transmission frames.

We adopt the use of a distributed transmission queue in this paper, and introduce a first-success approach to determine the nodes that succeed joining the transmission queue.

Section II describes QSMA-FAST (Queue Sharing Multiple Access with First Attempt with Success in a Turn). QSMA-FAST simplifies the way in which a distributed transmission queue is managed taking advantage of knowing the intended maximum number of nodes that can share a common channel in a wireless local-area network (WLAN). QSMA-FAST also simplifies the queue-joining discipline introduced in [10] by adopting a first-success method to initialize the transmission queue and to allow more nodes into the transmission queue once it is initialized.

QSMA-FAST organizes the channel into cycles, and each cycle consists of two or more queue turns assigned to the nodes that have successfully joined the distributed queue, and these turns are followed by a joining period during which nodes can send requests to join the queue. Data packets carry control information for queue management, and control packets are simply data packets with a zero-length payload. While a target queue size is not reached, a joining period ends with the first successful request to join the queue and lasts at most a maximum number of join-request turns.

Section III derives the throughput attained by QSMA-FAST grow the size of the queue to its target value.

Section IV compares QSMA-FAST with other medium access control (MAC) protocols, namely: ALOHA with priority ACK's, CSMA with priority ACK's, CSMA/CD with priority ACK's, TDMA with a fixed schedule, and the original version of QSMA. The results show that QSMA-FAST provides better throughput than all the other MAC protocols with or without carrier sensing. The results also show that QSMA-FAST incurs shorter delays reaching a target queue size than the original version of QSMA. Section V concludes the paper.

II. QSMA-FAST

Like prior channel-access methods based on distributed transmission queues, the design of QSMA-FAST consists of organizing access to a shared channel as a sequences of *queue cycles*. Each cycle consists of a sequence of *queue turns* with transmissions by nodes that have successfully joined the queue followed by a *queue-joining period* during which nodes can transmit their requests to join the queue. A queue-joining period lasts at most a maximum of m *join turns* and ends with the first successful request. Hence, at most one node can be added to the transmission queue at any given cycle.

A. Information Shared and Locally Maintained

The minimum information required for each node to store locally in order to participate in QSMA-FAST consists of: The target size of the transmission queue (Q), the currently known size of the queue (S), the current turn in the transmission queue (C), and the local position (P) of the node in the schedule determined by the transmission queue.

A packet consists of a header and a payload. The header of a packet states the sender, intended receiver, the SPAN components, an indication of the length of the packet payload, and error checking information. Data packets carry a non-empty data payload, and a packet sent to join the queue or to indicate that a node in the queue has no data to send has a zero-length data payload. To maintain the transmission queue reliably, the header of each packet sent by a node contains four additional components, which we call the SPAN of a packet and allow nodes to track the current queue position that should be transmitting, determine when a new cycle must start, decide when attempts to join the transmission queue are successful, and eliminate empty turns when nodes decide to leave the transmission queue. The four components of a SPAN are:

- S: The size of the queue.
- P: The position of the node in the queue.
- A: An acknowledgment (ACK) stating enough information from the identifier of the last node that sent a packet successfully.
- N: A data-ending notification signal, which consists of a node reducing by one the current size of the queue and stating a position equal to all 0's indicating no position in the queue.

We assume that the wireless LAN can support up to 255 active nodes. With this assumption, S and P can be stated using a byte for each component. The ACK is stated using three bytes, and we assume that they represent the corresponding least-significant bits of the MAC address of a node.

B. Initializing the Transmission Queue

If the transmission queue has not been initialized, the queue-join requests (or join requests) that nodes send help start the transmission queue. A join request consists of a packet with no data payload. When a join request from node x is meant to initialize the transmission queue, its SPAN components equal $(1, 1, x)$, which indicates a queue size of one node, node x occupies the first position in thew queue and does not have another node to acknowledge, and it is remaining in the queue.

The initial join requests used to initialize the transmission queue are sent based on pure ALOHA or non-persistent CSMA, depending on whether or not carrier sensing is used in the WLAN. Once the first node succeeds transmitting a join request, all other nodes receive it and assume the successful node is the head of the transmission queue. However, the head of the queue must be informed by some other node about its success starting the queue. Many different strategies can be used to respond to join requests sent to initialize the transmission queue. In this paper, we adopt an approach similar to the one used in ALOHA-NUI [11].

A node x sends its join request with a SPAN equal to $(1, 1, x)$. If the join request is sent successfully, all other nodes in the WLAN receive it and assume that node x is the first node in the queue. In addition, the successful join request informs the rest of the nodes that there are m transmission turns for other nodes to send their own join requests, where m is a network parameter. A node $y \neq x$ selects one of the m transmission turns to send its own join request with a SPAN equal to $(2, 2, x)$, which acknowledges the original join request from node x. If the request from y is the first to succeed after the join request from x, all nodes in the WLAN, including node x, learn that the queue has been initialized. Accordingly, the joining period ends and the first queue cycle starts with a data packet from node x.

Figure 1 illustrates the two cases in which the initialization of the transmission queue fails assuming that no carrier sensing is used. Figure 1(a) illustrates the case in which successive transmissions of join requests collide with one another, which forces nodes to apply back-off timers for the retransmission of their join requests, which eventually results in node a sending its join request successfully.

Figure 1(b) shows the case in which the queue is not initialized after one node (node a in the figure) succeeds transmitting its join request successfully, because no other node can send a join request acknowledging the join request from node a. As Figure 1(b) illustrates, the head of the queue (node a) must retransmit its join request after a back-off time because it does not know that its request succeeded in the channel.

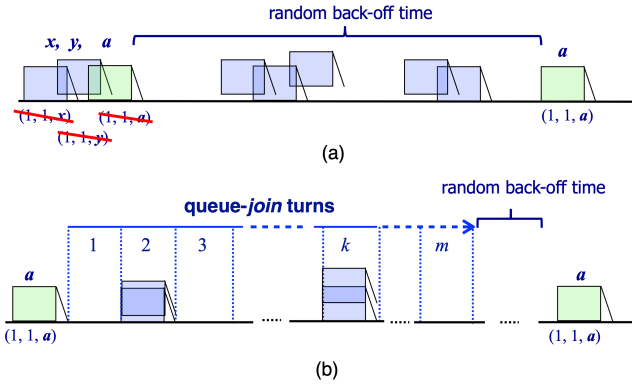

Fig. 1. Unsuccessful initialization of the transmission queue in QSMA-FAST

Fig. 2 shows an example of a successful initialization of the transmission queue in QSMA-FAST. In this example, node a is the first node to send its join request without interference, and node b is the first node that sends a join request acknowledging the join request from node a during join turn k prompted by the join request from node a. As the figure shows, node a sends a join request with a SPAN equal to $(1, 1, a)$; node b sends a join request with a SPAN equal to $(2, 2, a)$, which informs node a that it is the head of the queue; and the joining turns end with the successful transmission from node b. All nodes receive the join request from node b and know that the queue starts next, as shown in the figure. Once the transmissions from node a and node b take place, a new queue-join period starts and the period has up to m join turns, just as in the case shown in Figure 1(b). The new queue-join period ends after either m unsuccessful join turns or the first successful join request.

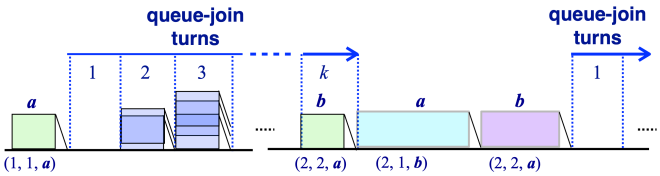

Fig. 2. Successful initialization of the transmission queue in QSMA-FAST

How queue-join requests are sent in QSMA-FAST depends on a limited-persistence policy and whether or not carrier sensing is used. A node that has a queue-join request to send first waits for a *persistence time period* that we denote by α. Once the node has waited for α seconds, it either transmits or backs off for a random amount of time based on the state of the transmission queue. A larger value of α allows more join requests to be sent at the end of a persistence time period, but also introduces a longer delay in the transmission of a join request at very light traffic loads.

Before the first join request is successful, waiting α seconds for the transmission of a queue-join request does not change the way in which such requests may collide in the channel with or without carrier sensing because each node uses its local measure of time.

Once a node sends the first join request without interference and before a second successful join request is sent, all other nodes that had a join request ready to be sent within α seconds of the end of the successful join request packet must select one of the m transmission turns following the successful join request.

After the transmission queue is initialized by the successful transmission of a second join request, nodes that need to be added to the transmission queue transmit their join requests as discussed in the next section.

C. Adding Nodes to an Existing Transmission Queue

The approach used to join an existing transmission queue is similar to the one used to start the queue itself, and has two modes, which depend on whether or not the size of the queue has reached a target value denoted by Q. As long as the queue size is smaller than Q, the queue-joining period of each queue cycle consists of up to m join turns. The queue-joining period ends with the first join request or after the m turns occur without a successful join.

Fig. 3 illustrates this case of the transmission policy in QSMA-FAST. In the example shown in the figure, nodes a, b, c and y have joined the transmission queue, and node z transmits a successful join request during the third join turn of the first queue cycle shown in the example. As a result, the following queue cycle includes node z at the end, and that is followed by a sequence of join turns. The maximum number of join cycles allowed in the example is set to $m = 5$ and no join turn is successful in the second queue cycle in the figure.

Once the queue size reaches its target value of Q, each queue cycle includes only one join turn at the end of the cycle. This is the case whether or not some node in the queue leaves the queue explicitly or due to a failure. This step is taken to maximize the througput of the network while still allowing a few new nodes or nodes that need to rejoin the queue to be added. Fig. 4 illustrates this case.

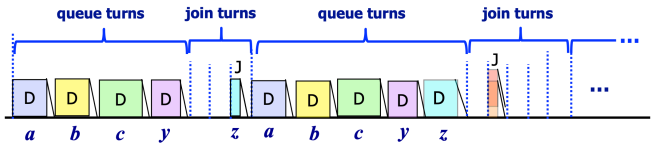

Fig. 3. Queue cycles in QSMA-FAST prior to reaching target queue size

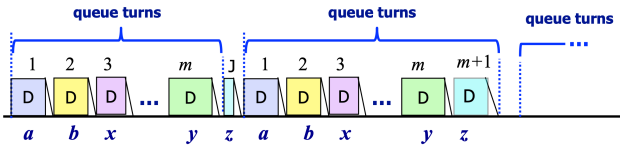

Fig. 4. Queue cycles in QSMA-FAST after target queue size is reached

In the example shown in Fig. 4, m nodes have already joined the queue and node z succeeds sending a join request during the single join turn at the end of the first queue cycle shown in the example. As a result, the following queue cycle has $m+1$ queue turns followed by a single join turn, which is empty. The following queue cycle has $m + 1$ queue turns.

D. Leaving the Transmission Queue

In a typical deployment in which the wireless network is fully connected, a node joining the transmission queue would simply remain in the queue while active and use its turns to transmit data or control packets as needed. In this paper we assume that this is the case, and hence any node in a WLAN is allowed to join and stay in the transmission queue.

The data-ending notification field of a packet header allows a node to explicitly leave the transmission queue. A node that is already in the transmission queue is implicitly assumed to have left it after it does not send a data packet during its transmission turn for a pre-defined number of queue cycles.

E. Using the Transmission Queue

We note again that each packet states a SPAN in its header. The ACK component of the SPAN refers to the last successful packet sent in the channel, which can be a data packet or a control packet. Hence, once the transmission queue has been initialized, a successful join request acknowledges the last data packet sent during a queue turn, the first data packet in a queue cycle acknowledges either a successful join request or the last data packet sent during the previous queue cycle, and a data packet sent in a queue turn other than the first turn acknowledges the prior data packet sent in a queue turn.

The ACK sent as part of the SPAN provides the sender node with fast feedback as to whether or not its transmission succeeded in the channel. In addition, the queue size and queue position stated in the SPAN reinforces the queue state that all nodes must maintain.

A data packet sent during a queue turn can last at most one *maximum channel access time* (MCAT), so that no node can monopolize the channel. Hence, a queue turn lasts at most the receive-to-transmit turn-around time needed for the owner of a queue turn to start transmitting, an MCAT and the maximum propagation delay needed for the transmission to start reaching all nodes. We assume that a node that has no data to send transmits a packet with an empty payload, which helps to reinforce the current state of the queue at every node.

III. Performance Analysis

A. Modeling Assumptions

We use the same traffic model adopted for the analysis of prior MAC protocols based on transmission queues [10], [11]. According to this model, a very large number of stations constitute a Poisson source sending data packets of equal length δ to the channel at an aggregate mean rate of λ packets per unit time.

A retransmission backoff time is random and such that the transmissions of new packets and backlogged packets can be assumed to be independent of one another. The system operates in steady state, with no possibility of collapse.

Any packet propagates to all nodes with the same propagation delay τ, and the only source of errors in transmitted packets is Multiple access interference (MAI). No capture effect exists and hence all transmitted packets that occurred concurrently must be retransmitted.

Requests to join the transmission queue occur with the same aggregate mean rate of λ packets per unit time. Nodes that join the transmission queue stay in the queue.

A fixed turn-around time of ω seconds is assumed for transitions from receive-to-transmit or transmit-to-receive modes.

The probability that a node that has joined the transmission queue has data to send during its queue turn is denoted by ν, where $\nu > 0$. The duration of a packet with a zero-length payload is denoted by γ, which is the length of a queue-join request.

For convenience, we assume that the persistence interval α used in QSMA-FAST equals the length of a join-request turn, that is, $\alpha = \gamma + \tau + \omega$.

B. Throughput of QSMA-FAST

Once the queue size reaches its target value of Q entries, the throughputt of QSMA-FAST is similar to that of ALOHA-NUI [11], because only one turn for joining is allowed in each queue cycle. The following theorem states the value of the throughput attained in QSMA-FAST in this case.

Theorem 1: Once the size of the transmission-queue size of a network is $N \geq Q$, the throughput of QSMA-FAST equals

$$S_{FAST} = \frac{\nu\delta N}{\omega + \tau + \gamma + N[\ \omega + \tau + \nu\delta + (1-\nu)\gamma\ ]} \quad (1)$$

Proof: Once the queue size N is at least equal to Q, the time duration of a queue cycle equals the time spent in N queue turns plus one queue-join period.

The average time U spent by nodes transmitting useful data in a queue cycle with N queue turns is $N\nu\delta$, and the duration of an average queue cycle is $C = T + J$, where T is the average time spent in N queue turns and J is the duration of a queue-join period.

A queue turn lasts $\omega + \delta + \tau$ seconds if the node using the turn has data to send, and lasts only $\omega + \gamma + \tau$ otherwise. Hence, the time T spent in N queue turns of an average queue cycle is

$$\begin{aligned} T &= N\left[\nu(\omega + \delta + \tau) + (1-\nu)(\omega + \gamma + \tau)\right] \quad (2)\\ &= N[\omega + \tau + \nu\delta + (1-\nu)\gamma] \end{aligned}$$

The duration of a join-request turn at the end of each queue cycle is the same as the duration of the persistence interval we have assumed, i.e., $J = \alpha = \omega + \gamma + \tau$.

The throughput of the network is simply the ratio of the time spent sending useful data in an average queue cycle divided by the duration of the cycle, i.e., $U/C = U/(T + J)$. The result follows by substituting the values of U, T and J. ■

Before the queue size reaches its target value, the throughput of the network is a function of the number of join turns used in an average queue cycle. The following theorem states the value of the QSMA-FAST throughput in this case.

Theorem 2: The throughput of QSMA-FAST when $N < Q$ equals

$$S_{FAST} = \frac{\nu\delta N}{J + N[\ \omega + \tau + \nu\delta + (1-\nu)\gamma\ ]} \quad (3)$$

$$\text{where} \quad J = \frac{e^{\lambda\alpha}}{\lambda}\left[1 - (1 + m\lambda\alpha e^{-\lambda\alpha})(1 - \lambda\alpha e^{-\lambda\alpha})^m\right],$$
$$\alpha = \omega + \tau + \gamma$$

Proof: When $N < Q$, the duration of an average queue cycle is $C = T + J$ as in the previous case, and the duration of the queue turns in a cycle is the same as in Eq. (2). However, the time J of the queue-joining period of an average queue cycle in this case equals the time needed for each queue-joining turn times the average number A of queue-joining turns. Given that a queue-joining turn lasts α seconds we have that $J = \alpha \cdot A$.

The probability p that a given queue-joining turn is successful is the probability that a single join request is scheduled for that turn. Given our assumption that the aggregate arrival of queue-join requests can be modelled as a Poisson source with parameter λ, we have that

$$p = \lambda\alpha e^{-\lambda\alpha} = \lambda(\gamma+\tau+\omega)e^{-\lambda(\gamma+\tau+\omega)} \tag{4}$$

When $N < Q$, the queue joining period of each queue cycle ends with either the first successful queue-joining turn or the last of the m turns allowed without a success. Based on our assumption about the Poison arrivals of join requests, the probability that a given join period lasts k joining turns is

$$P_k = p(1-p)^{k-1} \tag{5}$$

Accordingly, the expected number of queue-joining turns in an average queue cycle is:

$$\begin{aligned} A &= \sum_{k=1}^{m} kP_k = \sum_{k=1}^{m} kp(1-p)^{k-1} \\ &= p\left[\frac{1-(m+1)(1-p)^m + m(1-p)^{m+1}}{(1-(1-p))^2}\right] \\ &= \frac{1-(1+mp)(1-p)^m}{p} \end{aligned} \tag{6}$$

Therefore, we have

$$\begin{aligned} J &= \frac{\alpha(1-(1+mp)(1-p)^m)}{p} \\ &= \frac{e^{\lambda\alpha}}{\lambda}\left[1-(1+m\lambda\alpha e^{-\lambda\alpha})(1-\lambda\alpha e^{-\lambda\alpha})^m\right] \end{aligned} \tag{7}$$

The throughput of the network is again $U/(T+J)$ with $U = N\nu\delta$. Eq. (3) then results by substituting the value T in Eq. (2) and the value of J in Eq. (7) in this ratio. ■

C. Average Delay Reaching Target Queue Size

Based on the assumption that the system is in equilibrium, there must be a first join request that is sent without multiple access interference (MAI), which initializes the transmission queue. Once that occurs, all other nodes start transmitting their join requests in one of the queue-join turns allowed by the initial join request or the subsequent queue cycles once a second join request succeeds.

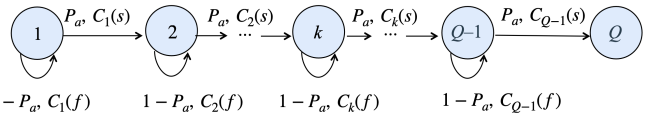

Fig. 5. Random evolution of queue size from 1 to Q entries in QSMA-FAST

Fig. 5 shows the random evolution of the QSMA-FAST queue size from 1 to the target value Q. For each state k of the queue size, the figure shows the probability of adding an entry to the queue (denoted by P_a) and the average length of the queue cycle when an entry is added (denoted by $C_k(s)$). The figure also shows the probability of failing to add an entry to the queue, which equals $1-P_a$, and the average length of the queue cycle when no new queue entry can be added (denoted by $C_k(f)$).

The probability of adding an entry to the transmission queue equals the probability of having a success in any of the join turns available, where up to m join turns can be attempted. While $N < Q$, the size of the transmission queue is not increased when none of the m join turns in a queue cycle result in a success. Using this fact and Eq. (4) for the value of the probability that a success occurs in a join turn we obtain

$$\begin{aligned} P_a &= 1-(1-\lambda\alpha e^{-\lambda\alpha})^m \\ &= 1-(1-\lambda(\gamma+\tau+\omega)e^{-\lambda(\gamma+\tau+\omega)})^m \end{aligned} \tag{8}$$

The average duration of a queue cycle with k queue turns in which no join request succeeds has m join turns and each queue turn has a packet with a non-empty payload with probability ν. Therefore,

$$\begin{aligned} C_k(f) &= k[\ \omega+\tau+\nu\delta+(1-\nu)\gamma\ ] + m[\ \omega+\tau+\gamma\ ] \\ &= (\omega+\tau+\gamma)m + (\omega+\tau+(1-\nu)\gamma+\nu\delta)k \end{aligned} \tag{9}$$

The average duration of a queue cycle with k queue turns in which a join request succeeds has an average time for joins J stated in Eq. (7), and each queue turn has a packet with a non-empty payload with probability ν. Therefore, we have

$$C_k(s) = k[\ \omega+\tau+\nu\delta+(1-\nu)\gamma\ ] + J \tag{10}$$

Because join requests succeed or fail independently of others, the average delay incurred in growing the transmission queue from 1 to Q can be obtained from the following system of equations, where $\bar{D}_i$ denotes the average delay incurred to grow the number of queue entries from i to Q. given by:

$$\begin{aligned} \bar{D}_{Q-1} &= (C_{Q-1}(s))P_a + (C_{Q-1}(f)+\bar{D}_{Q-1})(1-P_a) \\ \bar{D}_k &= (C_k(s)+\bar{D}_{k+1})P_a + (C_k(f)+\bar{D}_k)(1-P_a) \\ &\text{with } k = 1,...,Q-2 \end{aligned} \tag{11}$$

To simplify the task of solving the simultaneous equations in Eq. (11), we observe that $C_k(s) \leq C_k(f)$, $\gamma \ll \delta$, and it is expected that $m < Q$ in a typical network. Therefore, we use $J \approx m(\omega+\tau+\gamma)$, which means that $C_k(s) \approx C_k(f)$. This simplification results in the following system of equations:

$$\begin{aligned} \bar{D}_{Q-1} &= C_{Q-1}(f) + (1-P_a)\bar{D}_{Q-1} \\ \bar{D}_k &= C_k(f) + P_a\bar{D}_{k+1} + (1-P_a)\bar{D}_k \text{ with } k = 1,...,Q-2 \end{aligned} \tag{12}$$

Solving the system of equations in Eq. (12) yields

$$\bar{D}_1 \leq \frac{1}{P_a}\sum_{k=1}^{Q-1} C_k(f) \tag{13}$$

The previous equation is an inequality because of our simplifying assumption about the length of queue-joining periods. Substituting the value of $C_k(f)$ in Eq. (10) we obtain

$$\bar{D}_1 \leq \frac{1}{P_a}\sum_{k=1}^{Q-1}[(\omega+\tau+\gamma)m + (\omega+\tau+(1-\nu)\gamma+\nu\delta)k]$$

$$= \frac{(\omega+\tau+\gamma)(Q-1)m}{P_a} + \left(\frac{\omega+\tau+(1-\nu)\gamma+\nu\delta}{P_a}\right)\frac{(Q-1)Q}{2} \quad (14)$$

where the value of P_a is given in Eq. (8).

IV. Performance Comparison

We compare the throughput of QSMA-FAST with the throughput attained with ALOHA with ACKs, CSMA with ACKs and CSMA/CD with ACKs to highlight the vast performance improvements resulting from the use of transmission queues without the need for carrier sensing. The throughput result for CSMA/CD with ACKs assumes that transceivers are full duplex. We compare QSMA-FAST against TDMA with fixed transmission schedules to show the benefits of scheduled channel access that does not require fixed-length time slots. We also compare QSMA-FAST against the original proposal of QSMA with no carrier sensing, which we denote by QSMA-NCS, to show the improved channel utilization resulting from the new queue sharing approach used in QSMA-FAST. We assume that $Q = N$ for QSMA and QSMA-FAST.

A. Results from Analytical Model

The results are normalized to the length of a data packet by making $\delta = 1$ and use $G = \lambda \times \delta$ as the normalized traffic into the channel, where λ is the arrival rate of all packets. The normalized value of each other variable, which equals its ratio with δ is used as needed.

1) Throughput Results: Fig. 6 shows the numerical results for throughput from the analytical model for ALOHA with ACK's [9], CSMA with ACK's [8], CSMA/CD with ACK's [7], TDMA with Q time slots per TDMA frame, QSMA-NCS [10], and QSMA-FAST. The results assume a channel data rate of 1 Mbps, physical distances of 500 meters, a data packet of 1500 bytes, which renders a normalized propagation delay of 1.7×10^{-6}, and an overhead due to the PLCP (Physical Layer Convergence Procedure) of 24 bytes at 1 Mbps normalized to 240 bytes. The turn-around time ω is assumed to be the same as a propagation delay, and an ACK in ALOHA, CSMA, and CSMA/CD consists of 40 bytes. The relative differences in throughput for the various MAC protocols would be the same for higher data rates in a WLAN.

The arrival of all packets is Poisson with parameter λ. For ALOHA, CSMA, and CSMA/CD this means data-packet arrivals. For QSMA-FAST and QSMA-NCS, a node in the queue transmits during its own turn if it has at least one data-packet arrival during the previous δ seconds. Furthermore, if there is one or multiple arrivals in the last δ seconds prior to the start of a queue turn or a request turn, then there is at least one arrival for the next turn in the cycle. With these simplifying assumptions, the intensity of traffic from nodes in the queue correlates with the total traffic intensity, and means that $\nu = 1 - e^{-\lambda\delta}$ in the throughput equations for QSMA-FAST.

Results for QSMA-FAST and QSMA-NCS are shown for a queue size Q of 2, 10, 50 and 100 nodes, and the results for TDMA assume 100 time slots for 100 nodes.

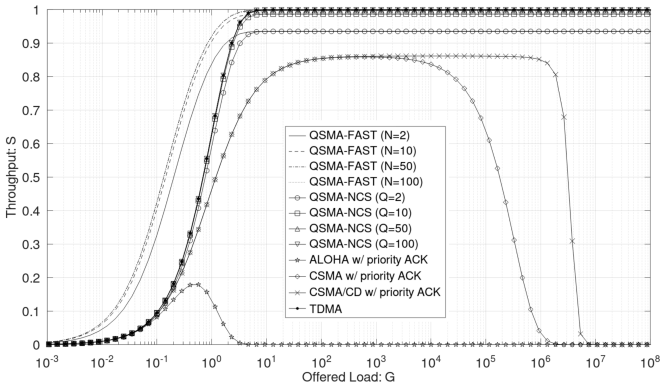

Fig. 6. Throughput of ALOHA with ACK's, CSMA with ACK's, CSMA/CD with ACK's, TDMA, QSMA-NCS, and QSMA-FAST

The results in Figure 6 illustrate the high efficiency and stability of QSMA-FAST, which outperforms all other MAC protocols. QSMA-FAST outperforms TDMA because queue turns with no data packets to be sent are used with very short signalling packets containing only a SPAN and an empty payload. By contrast, QSMA-NCS leaves idle an entire queue turn, just like TDMA. The original design of QSMA can be more efficient than TDMA only by means of carrier sensing, which allows unused time slots to last only as long as needed for a node to detect that there is no packet being transmitted in a queue turn. However, a sequence of queue turns with no data packets complicates the task of maintaining the correct queue status at all nodes based on carrier sensing because of the differences in propagation delays among them. QSMA-FAST avoids this problem by making nodes that do not have any data to send to transmit packets with empty payloads during their queue turns.

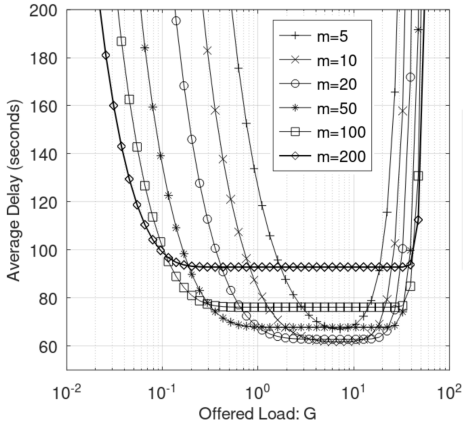

Fig. 7. Delays reaching the target queue size for different network sizes

2) Average Delay Reaching Target Queue Size: Fig. 7 shows the values obtained from Eq. (14) for the upper bound of the delay reaching a queue size of 100 for different values of m when nodes in the queue are in saturation mode, i.e., they always have data to send, and hence $\nu = 1$.

As Fig. 8 shows, there is a tradeoff between the number of join turns allowed per queue cycle and the resulting delay reaching the target queue size. A larger value of m spreads the new join requests over more join turns, which improves the likelihood than one turn is successful; however, it can increase the delay because the successful request may take place after many turns are not used at low loads or no success took place and all the m turns were wasted. The results in the figure indicate that using $m = 50$ is a good tradeoff and renders relatively small delays for realistic join-request traffic loads, mainly when the offered load of join requests ranges from 10^{-1} to 10 requests per packet time.

Based on the above results, we use $m = 50$ in Fig. 8 to show the upper bound of the average delay incurred in QSMA-FAST to reach different target queue sizes ranging from 2 to 100 as a function of the normalized number of join requests.

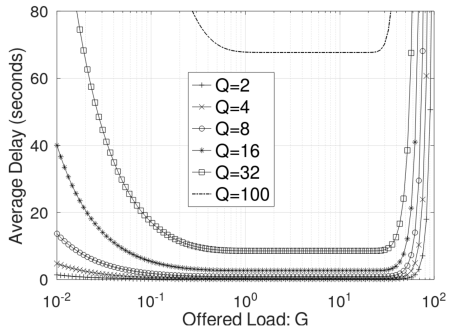

Fig. 8. Upper bound on average delay in reaching target queue value in QSMA-FAST for $m = 50$ and different values of Q

We should point out that the analytical results we show are pessimistic upper bounds, because the model assumes that join requests are always drawn from a large number of sources, independently of how many nodes have joined the queue. In practice, the contention among new join requests is reduced as more and more nodes succeed joining the transmission queue.

Fig. 9 compares the average delay reaching a target queue size of 100 in QSMA with carrier sensing (denoted simply by QSMA), QSMA-NCS, and the upper-bound of delay in QSMA-FAST.

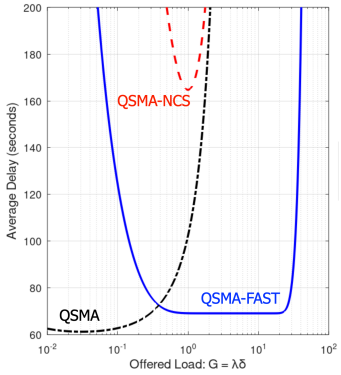

Fig. 9. Average delays in QSMA-FAST, QSMA and QSMA-NCS reaching a target queue value of 100 nodes with $m = 50$ in QSMA-FAST

It is clear from the results that QSMA-FAST is a better choice for fast queue joining than QSMA with or without carrier sensing if the join-request load is at least 10^{-1} requests per data-packet time. This is because the first-success approach of QSMA-FAST results in joining periods that are more likely to have a successful queue join than in QSMA. The figure also shows that carrier sensing is needed in the original design of QSMA to reduce queue-joining delays.

B. Results from Simulation Experiments

1) Throughput Results: We use simulation experiments based on the ns-3 simulator to compare the throughput of QSMA-FAST, QSMA with carrier sensing (QSMA) and without carrier sensing (QSMA-NCS), TDMA, ALOHA with ACK's, and CSMA with ACK's. Table I shows the results of simulation experiments in a network of 20 nodes and 50 nodes with nodes sending packets in saturation mode with data payloads of 1500 bytes and mixed payloads consisting of either 218 bytes or 1500 bytes. The standard deviations were negligible for all MAC protocols.

The simulation experiment for each MAC protocol was run for 10 independent trials with each trial running for 10 minutes of simulated time. The simulation experiments were based on a fully-connected network of 20 and 50 nodes placed randomly over a $212m \times 212m$ grid resulting in a one-way maximum propagation delay τ of 1415 ns. No channel capture or channel errors occur, the data rate is 10 Mbps and a transmission rate for the PLCP (Physical Layer Convergence Procedure) preamble and header of 24 bytes is 1 Mbps, which results in 192 microseconds.

TABLE I
THROUGHPUT OF MAC PROTOCOLS

20-node network		
MAC protocol	**1500-byte payload**	**mixed payload**
ALOHA with ACKs	.208	.3152
CSMA with ACKs	.8719	.8023
TDMA	.991	.626
QSMA	.9985	.9976
QSMA-NCS	.9498	.7541
QSMA-FAST	**.9915**	**.9867**
50-node network		
MAC protocol	**1500-byte payload**	**mixed payload**
ALOHA with ACKs	<.001	.003
CSMA with ACKs	.870	.792
TDMA	.991	.631
QSMA	.9985	.9976
QSMA-NCS	.9777	.7213
QSMA-FAST	**.9957**	**.9932**

The QSMA-FAST header is transmitted as data at a rate of 10 Mbps, which means 4 microseconds for 5 bytes. The persistence interval used in QSMA-FAST is $\alpha = 198830ns$. ALOHA and CSMA use a binary exponential backoff scheme with a maximum backoff time of 1 second. ACK's in ALOHA and CSMA are set equal to the 14 bytes used in IEEE 802.11 ACK's, and time slots in TDMA accommodate the largest packet size, with the TDMA frame being equal to the number of nodes in the network.

The small differences in throughput between TDMA and the QSMA variants for large payloads results from time slots being longer than data packets. The three QSMA variants are more efficient than TDMA when data payloads are mixed because

time slots in TDMA must accommodate the largest possible payload, and all QSMA variants allow a node in the queue to transmit as soon as it determines that the prior queue turn is over. QSMA without carrier sensing attains lower throughput than QSMA and QSMA-FAST, and QSMA-FAST attains very similar throughput than QSMA with carrier sensing for large and mixed loads.

2) Delay Results: Fig. 10 shows the time needed to add all nodes of a WLAN to the distributed queue in networks with 10, 20 and 50 nodes. The simulations used $m = 50$, which is the value that the analytical model showed to be a better tradeoff for low and high loads of queue-join requests.

As expected, the delay incurred for all nodes to join the queue is much smaller than the upper-bound obtained with the analytical model. The total delays for all network sizes are smaller than one second, which is remarkable given the simplicity of transceivers required by QSMA-FAST.

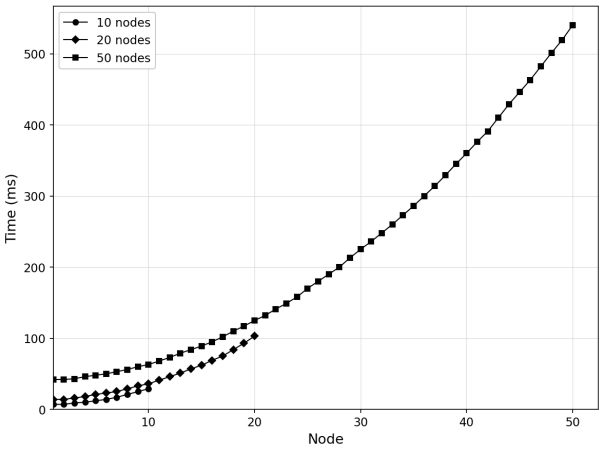

Fig. 10. Delays reaching the target queue size for different network sizes

V. Conclusions and Future Work

We introduced QSMA-FAST, a new MAC protocol based on distributed transmission queues in which channel access is based on the automatic distributed maintenance of a transmission queue that defines the schedule with which nodes can transmit data packets without multiple access interference in a WLAN. The transmission schedule is based on queue cycles in which each cycle consists of collision-free turns to transmit data packets and a joining interval.

QSMA-FAST uses a first-success policy in which, as long as a target queue size has not been reached, the joining interval of a queue cycle consists of up to a maximum of joining turns and ends with the first successful attempt. Once a target queue size is reached, a queue cycle includes a single joining turn. We used analytical modelling and discrete-event simulations to show that QSMA-FAST attains higher throughput than well-known MAC protocols, namely, ALOHA, CSMA, TDMA and the original QSMA proposal. We also showed that the first-success approach to handling requests to join the transmission queue renders much shorter delays than the original QSMA proposal.

The simplicity of QSMA-FAST and its ability to provide collision-free channel access without any manual configuration or special physical-layer support makes it very attractive for fully-connected wireless IoT deployments. Our next steps include studying the performance of QSMA-FAST when its policy allows multiple nodes to join the queue in one queue cycle, exploring how the value of m can be varied automatically based on a target value of Q and the past successes of join turns, determining how QSMA-FAST can be adapted to wireless networks with hidden terminals, and defining how QSMA-FAST can take advantage of multiple channels.

Acknowledgments

This work was supported by the Canada Excellence Research Chair in Intelligent Digital Infrastructures at the University of Toronto, funded by the Tri-agency Institutional Programs Secretariat.

References

[1] N.Abramson, "The ALOHA System–Another Alternative for Computer Communications," *Proc. Fall Joint Computer Conference '70*, 1970.

[2] L. Bao and J.J. Garcia-Luna-Aceves, "A New Approach to Channel Access Scheduling for Ad Hoc Net-works," *Proc. ACM MobiCom '01*, July 2001.

[3] L. Bao and J.J. Garcia-Luna-Aceves, "Hybrid Channel Access Scheduling in Ad Hoc Networks," *Proc. IEEE ICNP '02*, Nov. 2002.

[4] A. Boukersche, et al., *Handbook of Algorithms for Wireless Networking and Mobile Computing*, CRC Press, 2005.

[5] I. Chlamtac and A. Farago, "Making transmission schedules immune to topology changes in multi-hop packet radio networks," *IEEE/ACM Trans. on Networking*, Feb. 1994.

[6] D. Cirimelli-Low and J.J. Garcia-Luna-Aceves, "Key Activation Multiple Access (KAMA)," *Proc. IFIP WMNC 2021*, Oct. 2021.

[7] J.J. Garcia-Luna-Aceves, "Carrier-Resolution Multiple Access," *Proc. ACM PE-WASUN '17*, 2017.

[8] J.J. Garcia-Luna-Aceves, "Carrier-Sense Multiple Access with Collision Avoidance and Detection," *Proc. ACM MSWiM '17*, 2017.

[9] J.J. Garcia-Luna-Aceves, "KALOHA: ike i ke ALOHA," *Proc IEEE MASS 2019*, Nov. 2019.

[10] J. J. Garcia-Luna-Aceves and D. Cirimelli-Low, "Queue-Sharing Multiple Access," *Proc. ACM MSWIM '20*, Nov. 2020.

[11] J.J. Garcia-Luna-Aceves, D. Cirimelli-Low, and S. Homayon, "Making ALOHA Intelligent Taking Advantage of Working Memory Capacity," *Proc. ACM PE-WASUN '22*, Oct. 2022.

[12] L. Kleinrock and F.A.Tobagi, "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," *IEEE Trans. Commun.*, 1975.

[13] A. Muir and J.J. Garcia-Luna-Aceves, "An Efficient Packet-Sensing MAC Protocol for Wireless Networks," *Mobile Networks and Applications*, 1998.

[14] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient Collision-Free Medium Access Control for Wireless Sensor Networks," *Proc. ACM SenSys '03*, 2003.

[15] S. Ramanathan and E.L. Lloyd, "Scheduling Algorithms for Multihop Radio Networks," *IEEE/ACM Trans. Networking*, 1993.

[16] Z. Tang and J.J. Garcia-Luna-Aceves, "A Protocol for Topology-Dependent Transmission Scheduling," *Proc. IEEE WCNC '99*, Sept. 1999.

[17] F. Tobagi and L. Kleinrock, "The Effect of Acknowledgment Traffic on the Capacity of Packet-Switched Radio Channels," *IEEE Transactions on Communications*, June 1978.

[18] D. J. Vergados et al., "Local Voting: Optimal Distributed Node Scheduling Algorithm for Multihop Wireless Networks," *INFOCOM Workshop '17* , 2017.

[19] W. Xu and G. Campbell, "A Distributed Queuing Random Access Protocol for a Broadcast Channel," *Proc. ACM SIGCOMM '93*, Oct. 1993.

[20] C. Zhu and M. S. Corson, "A Five Phase Reservation Protocol (FPRP) for Mobile Ad Hoc Networks," *Proc. IEEE INFOCOM '98*, 1998.