Queue-Sharing Multiple Access

J.J. Garcia-Luna-Aceves
Computer Science and Engineering Department
University of California, Santa Cruz
Santa Cruz, CA, USA
jj@soe.ucsc.edu

Dylan Cirimelli-Low
Computer Science and Engineering Department
University of California, Santa Cruz
Santa Cruz, CA, USA
dcirimel@ucsc.edu

ABSTRACT

Queue-Sharing Multiple Access (QSMA) is introduced and analyzed. The new channel-access method consists of establishing and maintaining a distributed transmission queue among nodes sharing a common channel and results in a sequence of queue cycles, with each cycle having one or multiple queue turns with collision-free transmissions from nodes that have joined the transmission queue, followed by a joining period for the current cycle. Nodes can take advantage of carrier sensing to improve the efficiency with which nodes join and use the shared transmission queue. The throughput of ALOHA with priority ACK's, CSMA with priority ACK's, CSMA/CD with priority ACK's, TDMA with a fixed schedule, and QSMA with and without carrier sensing is compared analytically and by simulation in ns-3. The results show that QSMA is more efficient than TDMA with the simplicity of CSMA or ALOHA.

CCS CONCEPTS

• **Networks** → **Network protocols**; **Link-layer protocols**; **Network performance modeling**.

KEYWORDS

channel access; MAC protocols; ALOHA; CSMA; TDMA

ACM Reference Format:
J.J. Garcia-Luna-Aceves and Dylan Cirimelli-Low. 2020. Queue-Sharing Multiple Access. In *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '20), November 16–20, 2020, Alicante, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3416010.3423230

1 INTRODUCTION

The introduction of the ALOHA channel by Abramson [1] started a revolution on packet switching over wireless multiple-access channels. Its performance was improved dramatically by the introduction of carrier sensing in CSMA (carrier-sense multiple access) [20], and a plethora of channel-access protocols have been developed over the years that improve channel efficiency in different ways. Section 2 provides a summary of prior work on medium-access control (MAC) protocols for wireless networks [5, 15, 19], which can be characterized as contention-based and contention-free. The advantage of contention-based approaches is their simplicity; however, they cannot eliminate multiple-access interference (MAI) entirely or ensure maximum channel-access delays. The advantage of contention-free approaches is that they render high efficiency and delay guarantees; however, they are much more complex than contention-based schemes, and typically rely on time slotting of the channel, which requires clock synchronization among nodes.

The contribution of this paper is introducing a new channel-access method that is simple and attains high efficiency with maximum channel-access delay guarantees. We call this method *Queue-Sharing Multiple Access*, or **QSMA**.

Using the distributed queue method of QSMA transforms CSMA or ALOHA into de facto TDMA (time-division multiple access) with dynamic schedules, but without the need for time slotting at the physical layer, the definition of transmission frames with a fixed number of time slots, the use of pre-defined schedules, or the use of complex signaling to make reservations.

Section 3 describes how QSMA operates. A node with packets to send must first join a transmission queue. Once a node is in the queue it is able to transmit its data packets every time its turn in the queue comes up without any MAI. Channel access occurs in terms of queue cycles, with each cycle consisting of a queue period with one or multiple collision-free queue turns and a request period. Nodes try to join the transmission queue by sending short request packets during request periods. Each request packet and the header of a data packet states the size of the shared queue, the position of the sending node in the queue, a bit indicating the end of transmissions by the transmitting node, and the identifier of the last node that joined the queue. Nodes can use carrier sensing to improve the efficiency with which nodes join the shared transmission queue. A number of MAC protocols have been based on similar notions of distributed transmission queues [10, 21, 31]. QSMA improves over all of them by eliminating the need for a time-slotted channel or signaling between specific sender-receiver pairs.

Section 4 uses a simple analytical model to compute the throughput of QSMA with and without carrier sensing, and the average delay incurred in reaching a target queue size.

Section 5 uses the results from the analytical model and discrete-event simulations in ns-3 [22] to compare the performance of QSMA, TDMA with fixed schedules, ALOHA with priority ACK's, CSMA with priority ACK's, and CSMA/CD (collision detection) with priority ACK's. The results show the enormous benefits of QSMA compared to all the other MAC protocols, and the ability of QSMA to quickly attain collision-free channel access schedules that avoid idle transmission turns. Just like carrier-sensing improves the maximum channel-access efficiency from less than 20% in ALOHA to more than 70% in CSMA in fully-connected networks depending on propagation delays, a distributed queue renders more than 90% channel efficiency with the added benefit of stable channel access

MSWiM '20, November 16–20, 2020, Alicante, Spain
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8117-8/20/11.
https://doi.org/10.1145/3416010.3423230

resulting from collision-free schedules and avoidance of idle transmission turns. The simulation code for QSMA, CSMA and ALOHA is publicly available from GitHub [7], which allows the reader to reproduce the results discussed in this paper, study QSMA in different scenarios, and improve on the basic design of QSMA.

Section 6 presents our conclusions and summarizes a number of directions for future work.

2 RELATED WORK

Carrier sensing has been used since the introduction of CSMA in most contention-based channel-access protocols. Over the years, carrier sensing has been combined with other techniques like collision-avoidance signaling, collision resolution [6, 8], collision detection [5, 10, 12] and other physical-layer mechanisms like coding and time slotting to improve the efficiency of channel access. Today, carrier sensing is an integral part of many channel-access protocol standards (e.g., WiFi and WiMAX). However, the performance of these protocols degrades at high loads and the protocols cannot provide channel-access guarantees. The most efficient contention-based MAC protocol is CSMA/CD, but requires full-duplex operation.

Most of the contention-free MAC protocols proposed to date require the use of transmission frames consisting of a fixed number of time slots, or at least assume the use of a time-slotted channel [5, 25]. The approaches proposed in this context include distributed elections of time slots [2, 3, 24], and the reservation of time slots based on voting or signaling similar to collision-avoidance handshakes [26–28, 30, 32]. A few approaches eliminate the need of fixed-length transmission frames by using lexicographic ordering of the identifiers of transmitting nodes, geographical or virtual coordinates related to the connectivity of nodes (e.g., [11]), or a common tree of periodic schedules of variable periods that are powers of two [18]. All these approaches require time slotting supported at the physical layer, and some even require a central authority to orchestrate the selection of schedules.

In addition to the added complexity incurred by requiring a time-slotted channel in prior schedule-based MAC protocols, MAC protocols based on time slots result in channel-access schedules that are independent of traffic demands. As a result, channel utilization may be low when too many nodes are silent during the time slots they were assigned, or transmit data packets that are much shorter than the duration of time slots.

The prior work that is the most relevant to the design of QSMA consists of viewing the state of the channel-access requests as a distributed queue or a transmission group shared among all nodes accessing the channel.

The Distributed Queue Random Access Protocol (DQRAP) [31] is arguably the first example of this approach, and its design was inspired by the Distributed Queue Dual Bus (DQDB) protocol (IEEE 802.6) [4] and collision-resolution schemes (e.g., [4, 6]). DQRAP assumes that the channel is time slotted and that each time slot consists of a data slots and multiple control mini-slots used for resolving collisions of requests to join the queue. The control mini-slots are used for collision resolution of requests to be added to the distributed data queue and the data mini-slots are used to transmit data packets without interference. Several variants of DQRAP have been reported over the years for applications ranging from satellite networks to the Internet of Things; however, they require the use of time slots and mini-slots (e.g., see [9]).

Group Allocation Multiple Access (GAMA) [21] was the first distributed-queue approach to eliminate the use of time slotting. It organizes channel access into a contention period and a group-transmission period, and uses a collision avoidance handshake in the contention period to add new members to the group-transmission period. CARMA-NTS [10] integrated collision avoidance and resolution in the contention periods of GAMA, which results in each contention period having additions to the group-transmission period. Sync-less Impromptu Time-Divided Access (SITA) [17] used a collision-avoidance handshake similar to the one used in GAMA, but works on the basis of reservations of bandwidth by each node in the form of periodic transmissions by that node. Each node maintains its own version of the state of the queue. The limitation of prior methods based on distributed queues is that they rely on either: (a) time slotting and transmission frames that require clock synchronization; or (b) explicit signaling between specific transmitter-receiver pairs, which requires senders to know whether specific intended receivers are present before the transmission queue can be built.

3 QSMA

QSMA enables channel access in a way that no prior channel-access protocol has been able to provide before, namely: (a) maintaining much of the simplicity of ALOHA and CSMA with priority ACK's; (b) attaining collision-free transmissions and maximum channel-access delay guarantees; (c) eliminating clock synchronization, transmission frames consisting of a fixed number of time slots, or signaling to specific nodes before they become part of the transmission queue; and (d) avoiding empty transmission turns within schedules.

3.1 Transmission Strategy

Just as a node in ALOHA or CSMA with priority ACKs [29] must wait for an ACK to be heard after a successful packet, a node in QSMA must give priority to nodes that have joined the shared transmission queue successfully before trying to join the queue. As a result, channel sharing becomes a sequence of **queue cycles**. Each queue cycle consists of a **queue period** that contains one or more **queue turns**, with each queue turn giving a collision-free transmission opportunity to a node that has joined the queue, followed by a **request period** during which nodes contend to join the queue by transmitting queue-join requests.

Each data packet sent during a queue turn includes the following *SPAN* components in its header: The *size* of the queue (S), the *position* (P) of the node in the queue, an *acknowledgment* (A) stating the identifier of the last node that joined the queue, and a *data-ending* bit (N) set to indicate the last transmission by the node in its queue turn. A request to join the queue is simply a short packet consisting of the same four SPAN elements we have stated.

The information in the SPAN elements allows nodes to establish and maintain a distributed transmission queue by: (a) tracking the current queue position that should be transmitting, (b) determining when a new cycle must start, (c) deciding when attempts to join the transmission queue are successful, and (d) eliminating empty turns when nodes decide to leave the transmission queue.

3.2 Queue-Bootstrapping Strategy

Figure 1 illustrates the approach used to bootstrap the shared transmission queue in QSMA. To start the queue, nodes transmit join requests as in ALOHA or CSMA, depending on whether or not carrier sensing is used. As the figure shows, while the queue size is 0, all nodes transmit requests stating $S = P = 1$, $N = 0$, and their own identifiers for the acknowledgment field A.

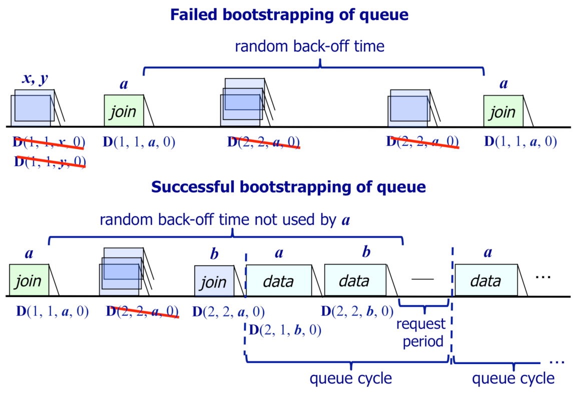

Figure 1: Bootstrapping the transmission queue

To prevent nodes from sending requests that collide persistently after their first transmissions, each node retransmits its request after a random time as long as the queue size is 0. If no carrier sensing is used, such a random time should be longer than the largest packet length allowed.

As Figure 1 indicates, the first successful request from any given node (node a in the example) stating $S = 1$ makes all other nodes adopt the successful node as the head of the queue. Accordingly, all other nodes state $S = P = 2$, $N = 0$, and the identifier of the head of the queue as A in their requests. Nodes other than the head of the queue transmit their own requests in between consecutive requests from the head of the queue. The head of he queue does not know that it was successful until a join request succeeds from a node b stating $S = P = 2$, $N = 0$, and A equal to its own identifier. Accordingly, the head of the queue persists sending its join requests at random times until it hears an acknowledgment from the second node to join the queue through a successful request transmission. The head of the queue (node a in the example) learns that the queue is established from the request it receives from the second node to join the queue (node b in the example). As soon as the queue includes two nodes, all nodes know which node is the head of the queue and nodes start transmitting data packets and request packets according to the queue-joining and management strategies described next.

3.3 Queue-Joining and Departure Strategy

Just like the transmission policy in CSMA may adopt different persistence strategies [20], such as non-persistent or 1-persistent strategies, different *queue-joining strategies* can be implemented in QSMA. A queue-joining strategy determines how many request turns are allowed during the request period of a queue cycle, and how those nodes waiting to join the queue are allowed to persist with the transmission of their requests during the request period of a queue cycle.

In this paper we assume a very simple queue-joining strategy. The request period of a cycle is limited to a single request turn, and nodes waiting to join the queue are allowed to transmit their join requests (i.e., persist) if they become ready to send their requests during the *persistence interval* of the cycle, and are forced to try to join after a random back-off time. To further simplify the strategy, the persistence interval is defined to be the last δ seconds of the queue period of a cycle, where δ is a constant and equals an average packet time. Accordingly, at most one node can be added to the transmission queue at the end of any queue cycle.

A 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. An empty queue turn q lasts only long enough for all nodes to detect that no transmission is taking place before the node occupying queue turn $q + 1$ being allowed to transmit. This time must be longer than a receive-to-transmit turn-around time needed for the owner of a queue turn to start transmitting, the maximum propagation delay needed for a transmission reach all nodes, and the time needed to detect carrier by the next node in the queue.

A request turn with a single request or multiple overlapping requests lasts a receive-to-transmit turn-around time needed for nodes sending their requests to start transmitting, the duration of a request packet, and the maximum propagation delay needed for the transmission to start reaching all nodes. Similarly, an empty request turn lasts only a receive-to-transmit turn-around time needed for any node sending a request to start transmitting, the maximum propagation delay needed for any transmission to start reaching all nodes, and the time needed to detect carrier by the node that occupies the first queue turn.

Figure 2 illustrates the queue-joining strategy in QSMA. Only those requests that occur during the persistence interval of cycle k are allowed to take place during the request turn of the cycle. All other requests to join the queue are scheduled for transmission at random times in the future.

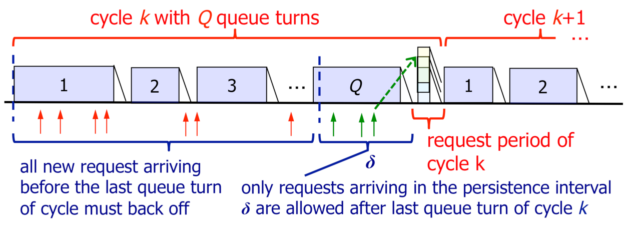
cycle k with Q queue turns
cycle k+1
1
2
3
Q
1
2
all new request arriving before the last queue turn of cycle must back off
δ
only requests arriving in the persistence interval δ are allowed after last queue turn of cycle k
request period of cycle k

Figure 2: Queue-joining strategy in QSMA

Nodes leave the shared transmission queue according to a queue-departure strategy. In most practical networks, 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. The strategy assumed in this paper consists of having nodes that join the transmission queue stay in the queue, until the queue size is one more turn than a target size m. After that, the node that has spent the most time in the queue leaves the queue during a given cycle with some probability.

3.4 Queue-Sharing Strategy

A node maintains several local variables to implement channel access: the queue size q, the current transmission turn c in the queue, the local transmission turn l occupied by the node, the entry turn e proposed by the node when it attempts to join the transmission queue , and the identifier a of the last node that joined the queue.

Stating the *SPAN* components in each packet transmitted makes QSMA more robust in the presence of physical-layer effects like fading, and could also enable the use of such energy-saving steps as allowing nodes that are not in the queue to not monitor the channel. The D bit allows nodes to avoid having idle turns resulting from nodes choosing to leave the queue.

Figure 3 shows the state machine describing how QSMA manages the transmission queue assuming carrier sensing. The state machine assumes that a node monitors the channel independently of whether or not the node has joined the transmission queue. Furthermore, all nodes experience the same channel conditions and no channel capture effects occur. Accordingly, a packet transmitted without MAI is either decoded correctly by all the nodes or by none of them, and no packet subject to MAI is decoded by any node.

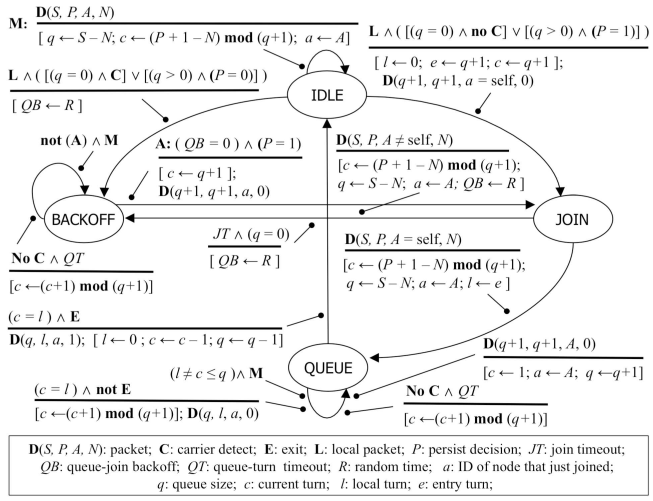
M: D(S, P, A, N) / [q ← S – N; c ← (P + 1 – N) mod (q+1); a ← A]
L ∧ ([(q = 0) ∧ no C] ∨ [(q > 0) ∧ (P = 1)]) / [l ← 0; e ← q+1; c ← q+1]; D(q+1, q+1, a = self, 0)
L ∧ ([(q = 0) ∧ C] ∨ [(q > 0) ∧ (P = 0)]) / [QB ← R]
IDLE
not (A) ∧ M
A: (QB = 0) ∧ (P = 1) / [c ← q+1]; D(q+1, q+1, a, 0)
D(S, P, A ≠ self, N) / [c ← (P + 1 – N) mod (q+1); q ← S – N; a ← A; QB ← R]
BACKOFF
JOIN
JT ∧ (q = 0) / [QB ← R]
D(S, P, A = self, N) / [c ← (P + 1 – N) mod (q+1); q ← S – N; a ← A; l ← e]
No C ∧ QT / [c ←(c+1) mod (q+1)]
(c = l) ∧ E / D(q, l, a, 1); [l ← 0 ; c ← c – 1; q ← q – 1]
D(q+1, q+1, A, 0) / [c ← 1; a ← A; q ←q+1]
QUEUE
(l ≠ c ≤ q)∧ M
(c = l) ∧ not E / [c ←(c+1) mod (q+1)]; D(q, l, a, 0)
No C ∧ QT / [c ←(c+1) mod (q+1)]
D(S, P, A, N): packet; C: carrier detect; E: exit; L: local packet; P: persist decision; JT: join timeout;
QB: queue-join backoff; QT: queue-turn timeout; R: random time; a: ID of node that just joined;
q: queue size; c: current turn; l: local turn; e: entry turn;

Figure 3: QSMA state machine

QSMA has four states: IDLE, JOIN, BACKOFF, and QUEUE. In addition to the reception of packets, a node reacts to other types of input events when it transitions to or remains in one of those states, and events are indicated in bold font. Each state transition specifies: (a) the event that causes the transition and the resulting update to the state of node, if any; and (b) the transmission by the node if there is any.

A packet received or transmitted by a node states the queue size S, the turn of the transmitting node P, the identifier A of the node that occupies the last queue turn, and the data-ending bit N. It is denoted by $\mathbf{D}(S, P, A, N)$. A request to join the transmission queue is simply a packet with no payload. The occurrence of carrier detect by the node is denoted by **C**. The event that a node in the transmission queue is ready to exit the transmission queue is denoted by **E**. The event that a node that is not in the transmission queue needs to join the queue is denoted by **L**.

A node is initialized to start in the IDLE state with the values $q = 0$, $l = 0$, $c = 0$, $e = 0$, $P = 0$, and a set to an invalid identifier. A node remains in the IDLE state as long as it has no need to join the transmission queue and simply monitors the activity in the channel.

A node transitions to the JOIN or BACKOFF state depending on input events. A node is in the JOIN state when it is attempting to join the transmission queue. A node is in the BACKOFF state if it must wait to attempt to join the queue. A node is in the QUEUE state if it succeeded joining the transmission queue. A node that joins the transmission queue can transmit only at the beginning of its queue turn and before it can exit the transmission queue it must transmit a packet with the N bit set to 1.

According to the persistence strategy assumed in this paper, if a node is ready to join the transmission queue during the last δ seconds of the queue period of the current queue cycle, the node is allowed to persist with its request and sets $P = 1$, and sets $P = 0$ otherwise.

The steps taken by a node as a result of receiving a packet in the IDLE state, the BACKOFF state, or the QUEUE state is denoted by **M** in Figure 3. These steps consist of updating the size of the queue q, the value of the current turn c, and the value of the ACK flag a.

IDLE state: A node in the IDLE state that receives a packet carries out the set of steps denoted by **M**. If a node receives a packet with the N bit set, the node eliminates from the transmission queue the turn that just took place by reducing the size of the queue by one turn and by not incrementing the value of the current turn. If the D bit is not set the value of the queue size is unchanged and the current turn is incremented. This is shown in Figure 3 by using the value of the N bit as an integer.

A node in the IDLE state that needs to join the transmission queue transitions to the BACKOFF state if the queue is empty and the node detects carrier or the queue is not empty and the node is not allowed to persist with its join request ($P = 0$). In that case, the node computes a random time R for its queue backoff (QB).

On the other hand, a node in the IDLE state that needs to join the queue transitions to the JOIN state if either the queue is empty and the node detects no carrier, or the queue is not empty and the node is allowed to persist with its join request ($P = 1$). The join request states: $S = q + 1$ to indicate an additional turn, $P = q + 1$ to request the last turn, A = self (its own identifier), and $N = 0$. The node remembers the value of the requested transmission turn by setting $e = q + 1$ and resets its local turn l to 0.

JOIN state: A node in the JOIN state waits for a packet to acknowledge its join request. The node transitions to the BACKOFF state if either the queue is empty and no packet is received within a join timeout (JT) interval, or a packet is received that does not state the identifier of the node in the acknowledgment (A). In this case the node computes a random time R as a queue-join backoff after updating its local variables as needed.

A node transitions to the QUEUE state if it receives a packet that acknowledges its request by having A as the identifier of the node, and updates the queue size q, and the current turn c in the same way as it does while in the IDLE state. In addition, it sets is local turn l to the turn value e it proposed in its attempt to join the queue.

BACKOFF state: A node in the BACKOFF state waits until its QB expires and processes packets and idle periods while in the BACKOFF state. The node processes packets carrying out the set of steps denoted by $\mathbf{M}$ or advances the current queue turn after detecting no carrier for a period of time (queue-turn timeout or QT) that is long enough for nodes to determine that the queue turn is empty. The node transitions to the JOIN state if its queue-join backoff expires and it is allowed to persist with its request ($P = 1$). In that case, the node updates the current queue turn to equal $q + 1$ and sends a join request $\mathbf{D}(q + 1, q + 1, a, 0)$. The node rejoins the BACKOFF state if its queue-join backoff expires and it is not allowed to persist with its request ($P = 0$).

QUEUE state: A node in the QUEUE state remains in that state until it receives a local signal to exit the transmission queue, which is denoted by event $\mathbf{E}$ in Figure 3. While the current queue turn does not correspond to its own queue turn (i.e., $l \neq c \leq q$), the node simply processes packets carrying out the set of steps denoted by $\mathbf{M}$, or advances the current queue turn after detecting no carrier for QT. A node transmits a packet with $S = q$, $P = l$, $A = a$, and $N = 0$ if it remains in the transmission queue, and transmits a packet with $N = 1$ to exit the transmission queue. In the latter case the node transitions to the IDLE state after reducing the queue size, updating teh curent queue turn, and reseting its position in the queue to 0.

4 PERFORMANCE ANALYSIS OF QSMA

We adopt the traffic model first introduced by Abramson [1] to analyze ALOHA, which has been used subsequently for the analysis of CSMA and most other channel-access protocols. In this model, a large number of nodes send requests to join the transmission queue with an aggregate mean rate of λ packets per unit time and constitute a Poisson source in the aggregate. A node backs off for a random time in a way that transmissions for new arrivals and backlogged arrivals can be assumed to be independent of one another, and the system operates in steady state with no possibility of collapse. Processing delays are negligible, the physical layer introduces no errors, and any packet propagates to all nodes with the same propagation delay τ. Hence, transmissions that overlap in time are the only source of errors.

All data packets are of equal length δ and a request packets are of length γ. A fixed turn-around time ω is assumed for nodes to start transmitting or receiving, and the time needed to detect carrier by any given node is ξ. If carrier sensing is used, the time needed for a node to decide that a queue turn or a request turn is empty is simply ξ (i.e., $QT = JT = 0$). Without carrier sensing, QT equals the length of a data packet and JT equals the length of a request packet. The length of the persistence interval of a cycle consists of the last δ seconds of its queue period. We assume that nodes that join the transmission queue stay in the queue waiting for the queue size to reach a *target value m*. Once the queue size is $m + 1$, nodes follow a first-in, first-out (FIFO) discipline in which the node that has spent the most time in the queue leaves the queue during a given cycle with probability q.

4.1 Throughput of QSMA

Given that QSMA establishes transmission cycles consisting of queue turns followed by a request turn, its throughput can be stated as a function of the average size of the transmission queue $\bar{Q}$ and the length of each queue turn and request turn.

THEOREM 4.1. *The throughput of QSMA with carrier sensing is*

$$S_{CS} = \frac{\delta\mu\bar{Q}}{[\omega + \tau + \xi + (\delta - \xi)\mu]\bar{Q} + \omega + \tau + \gamma - (\gamma - \xi)e^{-\lambda\delta}} \qquad (1)$$

where μ is the probability that a node transmits a packet during its queue turn and $\bar{Q}$ is the average size of the transmission queue.

PROOF. The throughput of QSMA is simply the ratio of the time $\bar{U}$ spent transmitting packets without MAI in an average queue cycle divided by the time $\bar{C}$ of an average queue cycle.

Each queue turn of a cycle contains a successful packet with probability μ lasting δ seconds; therefore, $\bar{U}$ equals $\mu\delta\bar{Q}$. The value of $\bar{C}$ depends on the average queue size and the average length of the join turns that occur in each cycle.

Based on the queue-join strategy we described in Section 3.3, a join turn with at least one join request lasts $\omega + \tau + \gamma$, while an empty join turn lasts only $\omega + \tau + \xi$.

The probability of having an empty join turn equals the probability of no requests arriving for transmission in the last δ seconds of the queue period. Given that the arrivals of join requests is Poisson with parameter λ, this probability is $e^{-\lambda\delta}$. Therefore, the average duration of a join turn equals $\bar{J} = \omega + \tau + \xi e^{-\lambda\delta} + \gamma(1 - e^{-\lambda\delta})$.

Similarly, a queue turn has a transmission with probability μ, in which case it lasts lasts $\omega + \tau + \delta$, and lasts only $\omega + \tau + \xi$ when it is empty. Therefore, the average duration of a queue turn is $\bar{T} = \omega + \tau + (1 - \mu)\xi + \mu\delta$.

The duration of an average queue cycle is then $\bar{C} = \bar{Q}\bar{T} + \bar{J}$ and the result follows by taking the ratio $\bar{U}/\bar{C}$. □

THEOREM 4.2. *The throughput of QSMA with no carrier sensing is*

$$S_{NCS} = \frac{\delta\mu\bar{Q}}{\bar{Q}(\omega + \tau + \delta) + \omega + \tau + \gamma} \qquad (2)$$

where μ is the probability that a node transmits a packet during its queue turn and $\bar{Q}$ is the average size of the transmission queue.

PROOF. Given an average queue size $\bar{Q}$, the average time $\bar{U}$ spent transmitting packets without MAI without carrier sensing is the same as in the previous theorem, i.e., $\mu\delta\bar{Q}$.

Given the assumption that all data packets have the same length, the duration of a queue turn is the same whether or not it is used for a transmission, because nodes must defer long enough without the benefit of sensing that transmissions are taking place. The same applies for a join turn.

With the assumption that all data packets last δ seconds and join requests last γ seconds, we thus have that $\bar{T} = \omega + \tau + \delta$ and $\bar{J} = \omega + \tau + \gamma$, which implies that the duration of an average queue cycle is

$$\bar{C} = \bar{Q}\bar{T} + \bar{J} = \bar{Q}(\omega + \tau + \delta) + \omega + \tau + \gamma \qquad (3)$$

The result follows by taking the ratio $\bar{U}/\bar{C}$. □

4.2 Average Size of Transmission Queue

Given that at most one node may join or leave the transmission queue in any given cycle, a cycle of length k must be followed by a cycle whose length can only be $k-1$, k, or $k+1$, depending on whether a node leaves the transmission queue and a node joins the transmission queue. Figure 4 illustrates the nature of channel utilization in QSMA.

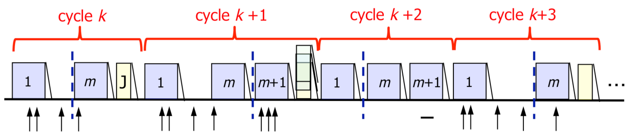

Figure 4: Example of channel utilization in QSMA

The figure shows cycle k having a queue with m turns and a single request to join the queue occurring during the last δ seconds of the queue period, which results in a success and hence cycle $k+1$ has $m+1$ nodes in the queue. Three requests to join the queue are allowed to persist in cycle $k+1$, which results in a collision and cycle $k+2$ has again $m+1$ nodes in the queue. One node leaves the queue after transmitting and no requests to join the queue occur during cycle $k+2$, which results in cycle $k+3$ having m queue turns again. The example also shows a single join request being able to persist in cycle $k+3$, which results in an increase of the queue size.

The following theorem states the average queue size in QSMA as a function of the target value of the queue size.

Theorem 4.3. *The average queue size in QSMA is*

$$\bar{Q} = m + \frac{P_s(1-q)}{q-P_s} \quad \text{with } P_s < q \tag{4}$$

where m is the target queue size, q is the probability that the first node that joined the queue leaves in a given cycle, and P_s is the probability of success during the request turn of a cycle.

Proof. Given that the system is assumed to operate in equilibrium, the size of the queue must drift to m as successes to join the queue take place within a finite period of time. Once the queue size is m, nodes may join and leave the queue with some probability, but the queue size must return to any given size $m+k$ with $k = 0, 1, \ldots$..

The average size of the queue $\bar{Q}$ equals $m+\bar{v}$, where $\bar{v}$ is the average number of queue turns in addition to m. The value of $\bar{v}$ can be obtained using a homogeneous Markov chain whose states represent the number of nodes in the transmission queue once the transmission queue has grown to include at least m nodes.

As Figure 4 illustrates, the probability of growing the queue size from $m+k$ to $m+k+1$ and the probability of reducing the queue size from $m+k$ to $m+k-1$ are independent of the number of nodes in the transmission queue $m+k$, with $k = 0, 1, 2, \ldots$.

Figure 5 shows the resulting Markov chain representing the number of queue turns in a cycle.

We denote by π_k ($k = 0, 1, 2, \ldots$) the stationary probability that there are $m+k$ nodes that have joined the transmission queue in a given queue cycle. The probability of increasing the queue size by one in a cycle is denoted by g and the probability of reducing the queue size by one in a cycle is denoted by r.

Starting with a queue size of m, the addition of a new node to the queue is independent of the departure of a node from the queue during a given cycle.

$$\begin{tikzcd}
m \arrow[r, bend left, "g"] \arrow[loop below, "1-g"] & m+1 \arrow[l, "r"] \arrow[r, bend left, "g"] \arrow[loop below, "1-g-r"] & m+2 \arrow[l, "r"] \arrow[r, bend left, "g"] \arrow[loop below, "1-g-r"] & \cdots \arrow[l, "r"] \arrow[r, bend left, "g"] & m+k-1 \arrow[l, "r"] \arrow[r, bend left, "g"] \arrow[loop below, "1-g-r"] & m+k \arrow[l, "r"] \arrow[r, bend left, "g"] \arrow[loop below, "1-g-r"] & \cdots \arrow[l, "r"]
\end{tikzcd}$$

Figure 5: Markov chain for QSMA with one departure per queue cycle after target queue size m is reached and an unbounded queue size is allowed

With the results stated above and our modeling assumptions we obtain the following balance equations for the Markov chain shown in Figure 5:

$$\begin{aligned} g\,\pi_0 &= r\,\pi_1; \\ (r+g)\pi_k &= g\pi_{k-1} + r\pi_{k+1} \text{ for } k = 1, 2, \ldots \end{aligned} \tag{5}$$

Eq. (5) results in the following by iteration and induction:

$$\pi_k = \pi_0\,(g/r)^k \quad \text{for } \; k = 1, 2, \ldots \tag{6}$$

The transmission queue must have some size of at least m in any given cycle; therefore,

$$\pi_0 + \pi_1 + \ldots + \pi_k + \ldots = 1 \tag{7}$$

Substituting Eq. (6) in Eq. (7) we obtain

$$1 = \pi_0 + \sum_{i=1}^{\infty} \pi_i = \left(1 + \sum_{i=1}^{\infty}\left(\frac{g}{r}\right)^i\right)\pi_0 \tag{8}$$

For the system to operate in equilibrium, it must be true that $g < r$; therefore, $\rho = g/r < 1$. From Eq. (8) we have

$$\pi_0 = \left(1 + \sum_{i=1}^{\infty}\left(\frac{g}{r}\right)^i\right)^{-1} = 1 - \frac{g}{r} = 1 - \rho \tag{9}$$

Substituting Eq. (9) in Eq. (6) we obtain:

$$\pi_k = \left(1 - \frac{g}{r}\right)\left(\frac{g}{r}\right)^k = (1-\rho)\,\rho^k \text{ with } 0 < \rho < 1 \tag{10}$$

The average size of the transmission queue is then:

$$\begin{aligned} \bar{Q} &= m + \sum_{i=0}^{\infty} i\,\pi_i = m + \rho\sum_{i=1}^{\infty} i\,(1-\rho)\rho^{i-1} \\ &= m + \frac{\rho}{1-\rho} \text{ with } 0 < \rho < 1 \end{aligned} \tag{11}$$

With the assumptions in our model and $g < r$ we have

$$g = P_s\,(1-q);\; r = (1-P_s)q;\; P_s < q \tag{12}$$

The result follows by substituting the values of r and g in Eq. (11).

□

Assuming that the arrival of requests to join the transmission queue is Poisson distributed with parameter λ, a join request succeeds with probability $\lambda\delta e^{-\lambda\delta}$. Substituting this result for P_s in Eq. (4) leads to the following result.

Corollary 4.4. *If the arrival of requests to join the transmission queue is Poisson with parameter λ the average queue size in QSMA is*

$$\bar{Q} = m + \frac{(1-q)\lambda\delta e^{-\lambda\delta}}{q - \lambda\delta e^{-\lambda\delta}} \text{ with } \lambda\delta e^{-\lambda\delta} < q \tag{13}$$

The previous results on the average queue size are independent of the use of carrier sensing. This is a direct consequence of the simple approach we chose to use for persistence as part of the queue-join strategy. More specifically, the success of a join turn is determined solely by the arrival of join requests during the last δ seconds of the queue period of a cycle, which is a process that does not depend on carrier sensing.

4.3 Delay Reaching Target Queue Size

Figure 6 illustrates the random evolution incurred in reaching a target queue size of m starting with the first request packet that is transmitted successfully into the channel. Each node in the figure represents the number of queue turns in a given queue cycle. The arrows represent the transition from state k to either state $k+1$ or state k itself for $k = 1, 2, ..., m-1$. The figure shows the transition probabilities and the average delays incurred in transitions.

A transition from state 1 to state 2 or back to state 1 takes place according to the bootstrapping strategy described in Section 3.2. The probability of transitioning from state 1 to state 2 is denoted by P_1. The average time spent in that transition is denoted by $C_1(s)$. On the other hand, the probability of transitioning from state 1 back to state 1 is $1 - P_1$, and the average time spent in that transition is denoted by $C_1(f)$.

The transitions from a state k to $k+1$ or back to state k for $2 \leq k \leq m$ occur according to the queue-joining and sharing strategies presented in Sections 3.3 and 3.4. As such, the probability of transitioning from state k to state $k+1$ is the same for $2 \leq k \leq m$ and is denoted by P_s, and the probability of transitioning from state k back to state k is $1 - P_s$. The average time spent in the transition from state k to state $k+1$ is denoted by $C_k(s)$, and the average time spent in the transition from state k back to state k is denoted by $C_k(f)$.

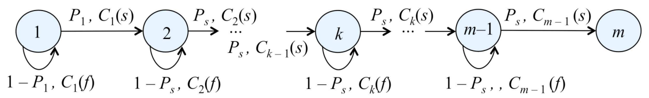

Figure 6: Random evolution reaching target queue size

The following theorem states the average delay reaching a target queue size assuming that carrier sensing is used and the carrier-detect time is negligible ($\xi = 0$).

Theorem 4.5. *The average delay in reaching a target queue size m in QSMA with carrier sensing and a probability μ that a node transmits during its queue turn is*

$$\bar{D}(m) = \frac{1}{\lambda} + \left(e^{\lambda(\omega+\tau)} - 1\right) R + (\omega+\tau+\gamma)\left(e^{\lambda(\omega+\tau)} + 1\right) \tag{14}$$

$$+ (m-2)\left[s + f\left(\frac{e^{\lambda\delta}}{\lambda\delta} - 1\right)\right] + \left(\frac{m(m-1)}{2} - 1\right)\frac{e^{\lambda\delta}}{\lambda\delta}(\omega+\tau+\mu\delta)$$

where $s = \omega + \tau + \gamma\lambda\delta e^{-\lambda\delta}$ and $f = \omega + \tau + (1 - (1+\lambda\delta)e^{-\lambda\delta})\gamma$.

Proof. Given that the system is in equilibrium, there must be a first join request transmitted without MAI with probability 1. To grow the queue to two turns, a second node must succeed transmitting its request without interference from any other node trying to join the queue. Hence, P_1 equals the probability that a request packet is sent successfully. Given that carrier sensing is used and nodes are deaf while they turn from receive to transmit mode, it follows that $P_1 = e^{-\lambda(\omega+\tau)}$.

The average time elapsed between the first and the second successful request packet equals the average interarrival time of requests transmitted by nodes, given that no queue is established until the second request succeeds. By assumption, the arrival process of requests is Poisson with parameter λ, and hence the average elapsed time between two successful requests is $1/\lambda$. Given that each request takes $\omega + \tau + \gamma$ seconds, we have

$$C_1(s) = 1/\lambda + 2(\omega + \tau + \gamma) \tag{15}$$

The average time elapsed in a transition from state 1 back to state 1 is the average time between the transmission of request packets by the head of the queue according to the bootstrapping strategy. Given that an average random time R is used between the retransmissions of requests by the same node until a second request succeeds to acknowledge the head of the queue, we have

$$C_1(f) = R + \omega + \tau + \gamma \tag{16}$$

A transition from state k to state $k+1$ ($2 \leq k \leq m-1$) requires a single request to be transmitted during the request turn of a queue cycle. Given that the persistence interval during a queue cycle is δ seconds, it follows that the transition probability from state k to state $k+1$ for $2 \leq k \leq m-1$ equals $P_s = \lambda\delta e^{-\lambda\delta}$.

A queue cycle in state k incurs k queue turns, and a request turn lasts $\omega + \tau + \gamma$ for a successful request; therefore,

$$C_k(s) = kT + \omega + \tau + \gamma\lambda\delta e^{-\lambda\delta} \tag{17}$$

where T is the average length of a queue turn and for $\xi = 0$ equals

$$T = \omega + \tau + \mu\delta.$$

A transition from state k back to state k ($2 \leq k \leq m-1$) involves k queue turns and occurs if no request is sent or multiple requests are transmitted during the request turn of the cycle. No request is sent in a request turn with probability $e^{-\lambda\delta}$ and the request turn lasts $\omega + \tau + \xi$ seconds in that case. Similarly, multiple requests are sent in a request turn with probability $1 - e^{-\lambda\delta} - \lambda\delta e^{-\lambda\delta}$ and the request turn lasts $\omega + \tau + \gamma$ seconds in that case. Hence,

$$C_k(f) = kT + \omega + \tau + \xi e^{-\lambda\delta} + \gamma(1 - (1+\lambda\delta)e^{-\lambda\delta}) \tag{18}$$

$$= kT + \omega + \tau + \gamma(1 - (1+\lambda\delta)e^{-\lambda\delta})$$

The success of a request to join the queue is independent of any other request. Accordingly, the average delay incurred in growing the queue size to m starting from state 1 can be obtained from the following equations:

$$\bar{D}_1 = P_1(C_1(s) + \bar{D}_2) + (1 - P_1)(C_1(f) + \bar{D}_1) \tag{19}$$

$$\bar{D}_k = P_s((C_k(s) + \bar{D}_{k+1}) + (1 - P_s)(C_k(f) + \bar{D}_k),\ 2 \leq k \leq m-2 \tag{20}$$

$$\bar{D}_{m-1} = P_s C_{m-1}(s) + (1 - P_s)(C_{m-1}(f) + \bar{D}_{m-1}) \tag{21}$$

Solving Eq. (19) for $\bar{D}_1$ and substituting the values of P_1, $C_1(s)$, and $C_1(f)$ we obtain

$$\bar{D}_1 = \frac{1}{\lambda} + \left(e^{\lambda(\omega+\tau)} - 1\right) R + \left(e^{\lambda(\omega+\tau)} + 1\right)(\omega + \tau + \gamma) + \bar{D}_2 \quad (22)$$

Solving Eqs. (20) and (21) for $\bar{D}_2$ we have

$$\bar{D}_2 = (m-2)\left[s + f\left(\frac{e^{\lambda\delta}}{\lambda\delta} - 1\right)\right] + \left(\frac{m(m-1)}{2} - 1\right)\frac{e^{\lambda\delta}}{\lambda\delta}(\omega+\tau+\mu\delta) \quad (23)$$

where $s = \omega + \tau + \gamma\lambda\delta e^{-\lambda\delta}$ and $f = \omega + \tau + (1 - (1 + \lambda\delta)e^{-\lambda\delta})\gamma$.

The result follows by substituting Eq. (23) in Eq. (22). □

5 PERFORMANCE COMPARISON

5.1 Results from Analytical Model

We compare QSMA with the most efficient schedule-based MAC protocol (TDMA with a fixed schedule), the most efficient contention-based MAC protocol (CSMA/CD, which requires full-duplex nodes), and ALOHA and CSMA, which are similar in complexity to QSMA.

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. The normalized value of each other variable, which equals its ratio with δ is used as needed. The carrier-detect time ξ is set to 0 in order to use published results.

5.1.1 Throughput Results: The arrival of all packets is Poisson with parameter λ. For ALOHA, CSMA, and CSMA/CD this means data-packet arrivals. For QSMA, 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, then there is at least one arrival for the next queue turn in the cycle. The average queue size is assumed to be equal to the target queue size (i.e., $\bar{Q} = m$, which implies that $q = 1$ in Eq. (13)).

With these simplifying assumptions, the intensity of traffic from nodes in the queue correlates with the total traffic intensity, and means that $\mu = 1 - e^{-\lambda\delta}$ in Eqs. (1) and (2).

The throughput of fixed-schedule TDMA can be derived from Eq. (2) for QSMA by considering that the only overhead TDMA would incur is due to propagation delays and turn-around times. Assuming the same traffic intensity μ above, this results in:

$$S_{TDMA} = \delta\mu/(\omega + \tau + \delta) = (1 - e^{-\lambda\delta})\delta/(\omega + \tau + \delta) \quad (24)$$

Figure 7 shows the throughput (S) as a function of G for QSMA based on Eqs. (1) and (2), fixed-schedule TDMA (Eq. (24)), ALOHA with priority ACK's (Eq. (20) in [14]), non-persistent CSMA with priority ACK's (Eq. (18) in [12]), and non-persistent CSMA/CD with priority ACK's (Eq. (3) in [13]).

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. No carrier sensing in QSMA is denoted by "NCS."

The results in Figure 7 illustrate the high efficiency and stability of QSMA, which outperforms CSMA and CSMA/CD even when the average queue size is only two turns. This is remarkable, given that only carrier sensing is used to attain collision-free transmissions from all the nodes that join the transmission queue.

It is clear that carrier sensing is very useful in QSMA at light loads by allowing a node in the queue to quickly take over an unused queue turn or request turn after detecting no carrier at the start of the turn. This results in a very effective transmission strategy without collisions that is even better than TDMA with a fixed schedule, which may leave time slots unused. At high loads, carrier sensing does not provide any advantage in QSMA, because most request and queue turns tend to be used. QSMA is slightly less efficient than fixed-schedule TDMA at high loads only if every data packets occupies most of its time slot. Large variances in packet lengths makes QSMA more efficient than TDMA even at high loads, because a queue turn does not have a fixed length.

5.1.2 Delay Results: Figure 8 shows the average delay incurred in QSMA to reach different target queue sizes as a function of the normalized number of join requests $G = \lambda\delta$ when $\mu = 1$ and $R = 0$. It is clear from Eq. (14) and the figure that the average delay incurred in reaching a target queue becomes very large when m is large, the average length of data packets is long, and the average number of requests per request turn ($\lambda\delta$) is much larger than 1. This is because each success would take many cycles to occur and each cycle would last a long time. has many queue turns.

In a finite network with a target queue size that can accommodate all active nodes, the arrival rate of queue-join requests decreases as more nodes join the queue. This makes the results in Figure 8 far more promising, because they suggest that a target queue size that includes all active nodes can be reached in just a few seconds even in networks with a hundred nodes.

5.2 Simulation Comparison

5.2.1 Simulation Setup and Scenarios: We use the ns-3 simulator [22] to verify the average throughput of QSMA and delays incurred in reaching a target queue size predicted by the analytical model. Given the results from the analytical model, we compare QSMA only with ALOHA with ACK's and CSMA with ACK's. TDMA with fixed schedules would result in very similar throughput as QSMA, and CSMA/CD would require full-duplex transceivers.

The results represent the mean and the standard deviation of 10 trials for each experiment. The scenarios assume fully-connected topologies of 10 or 50 nodes that always have data packets to send. Node placement is random with nodes being within 425m of each other, resulting in propagation delays of 1415 ns or less. No channel capture or channel errors occur, the MAC data rate is 10Mbps, and the transmission rate for the PLCP (Physical Layer Convergence Procedure) preamble and header of 24 bytes is 1 Mbps in the three protocols. Each simulation experiment lasts 10 min.

ALOHA and CSMA use a binary exponential backoff scheme with a maximum backoff of 256 epochs, where each epoch lasts 100 μs. ACK's in ALOHA and CSMA are set to 14 bytes used in 802.11 ACK's. Until the queue is successfully bootstrapped, QSMA uses a binary exponential backoff scheme with a maximum backoff of 10ms. Once the queue is established, nodes' backoffs are calculated

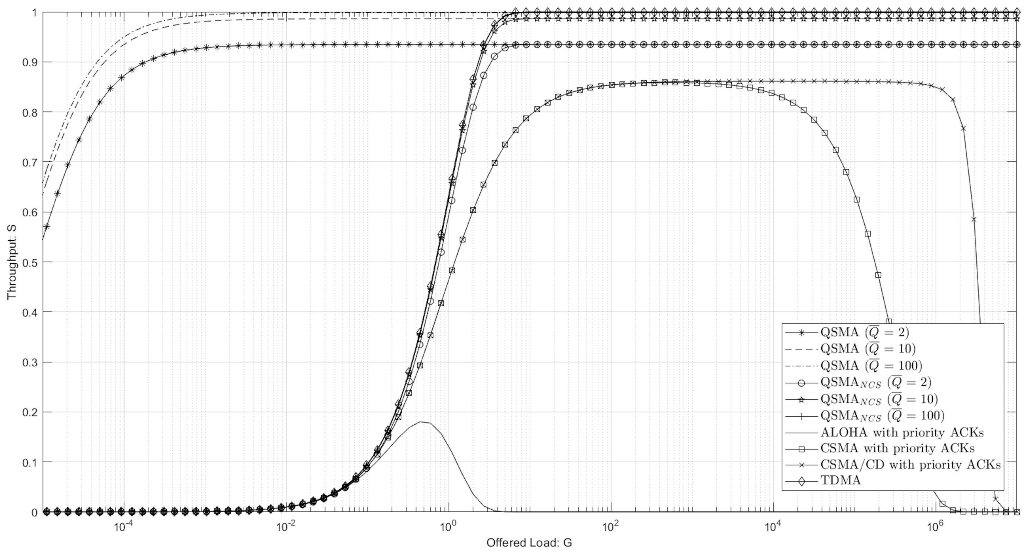

Figure 7: Throughput of ALOHA with ACK's, CSMA with ACK's, CSMA/CD with ACK's, TDMA, and QSMA

in queue turns, with a maximum backoff of 32 turns. Data packets in QSMA add three bytes, which suffices to carry the S, P, A, and N feedback for up to 64 nodes. The target queue size in QSMA is set to accommodate any number of nodes.

5.2.2 *Throughput Results:* Given the results from the analytical model, we do not consider TDMA , which performs much like QSMA, and CSMA/CD, which requires full-duplex operation. Figure 9 shows the normalized throughput for ALOHA, CSMA and QSMA with and without carrier sensing. QSMA w/o CS denotes QSMA without carrier sensing, and QSMA denotes the use of carrier sensing in QSMA. We consider data payloads of 218 bytes, which correspond to a typical VoIP frame [23], and 1500 bytes, which is the typical payload MTU of an IP packet [16], and an even combination of them.

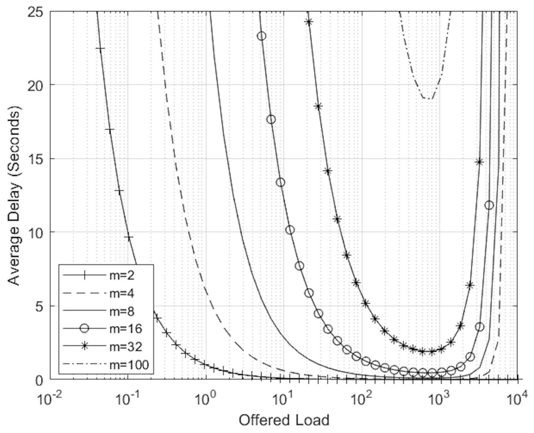

Figure 8: Average delay reaching target queue size

As the figure shows, QSMA attains far better throughput than ALOHA and CSMA independently of the network size or payload type, with better than 90% throughput even when no carrier sensing is used. The small throughput degradation with small payloads in QSMA results mainly from the relatively lager overhead of propagation delays in queue turns with short packets. ALOHA performs much worse with large and mixed payloads because because the average vulnerability period of a data packet is larger. CSMA attains throughput values above 60% and below 90%; it performs better with large payloads because the overhead of priority ACK's is comparatively smaller than with small data packets.

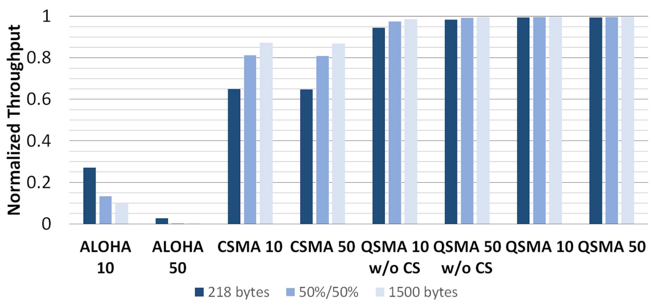

Figure 9: Throughput of ALOHA, CSMA and QSMA

5.2.3 *Delay Results:* Figure 10 shows the average delay incurred by each node to join the distributed queue in QSMA in the order in which each node joins the queue when data packets payloads have 218 and 1500 bytes. It should be noted that nodes do no reset their backoff exponent as in ALOHA or CSMA, and nodes that fail repeatedly can face long delays joining the queue, even when the network has a few nodes attempting to join. Furthermore, as the shared queue grows in size, each queue cycle becomes longer, which increases the time a node must wait before re-transmitting a join request, and decreases the probability that a node will come out of backoff within the persistence interval. Fortunately, each node needs only one success to join the queue, which results in all the nodes joining the queue in a very short period of time for practical purposes.

The simulation experiments show that, even with no carrier sensing, all nodes join the queue well within 9 seconds when data packets have 1500 bytes, and well within 3 seconds when data packets have 218 bytes. The delay results shown in Figure 10 would be more than adequate in most real networks. However, the simplistic queue-joining strategy assumed in this paper can and should be improved to make QSMA more effective in the presence of long propagation delays resulting from longer distances.

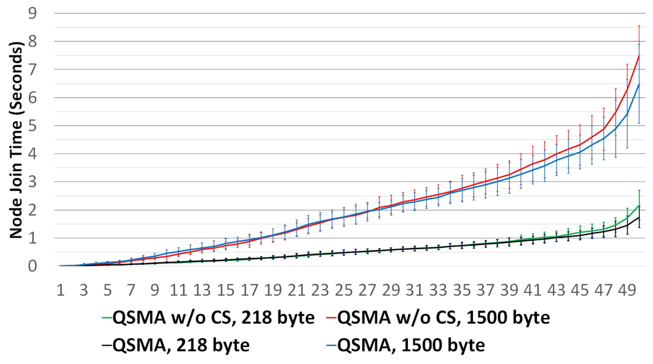

Figure 10: Delay joining the queue in QSMA

6 CONCLUSIONS AND FUTURE WORK

We introduced QSMA, a new family of channel-access protocols in which collision-free transmissions and maximum channel-access delay guarantees are quickly attained through the sharing of a transmission queue without the need for time slotting at the physical layer or the use of explicit handshakes requiring transmitters to know the identity of intended receivers before such nodes have joined the transmission queue.

The signaling overhead in QSMA is very small. Each packet header states the queue size, a position in the queue, a bit informing whether the transmitter is leaving the queue, and the identifier of the last node that joined the transmission queue. A request packet used to join the shared transmission queue simply specifies this same information, and is much smaller than a data packet.

Our results show that QSMA is more efficient than even TDMA with a fixed schedule, and yet it maintains much of the simplicity of ALOHA and CSMA.

Several QSMA optimizations can be made and deserve further study. The delays incurred in reaching target queue sizes could be reduced by either allowing multiple successes during a requests turn, or by increasing the probability of having a successful request in a request turn. In addition, the efficiency of QSMA could be further improved by making the queue-joining mechanism more aggressive when the queue size is small and recent join requests are successful, and less aggressive otherwise.

Our results on QSMA open up many research avenues on the design of channel-access protocols based on distributed-queues. Of particular interest are: (a) designing signaling to cope with hidden terminals while attaining collision-free transmissions; (b) taking advantage of multiple channels; (d) having nodes occupy multiple turns per queue cycle to support quality-of-service guarantees; and (e) allowing nodes in the queue to save energy by being inactive in some queue cycles.

ACKNOWLEDGMENTS

This material is based upon work sponsored by the National Science Foundation (NSF) under Grant CCF-1733884, and by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL). Any opinions, findings, conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of NSF, DARPA, AFRL, the U.S. Department of Defense, or the U.S. government.

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] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, 1992.
[5] A. Boukersche, et al., *Handbook of Algorithms for Wireless Networking and Mobile Computing*, CRC Press, 2005.
[6] J. Capetanakis, "Tree Algorithm for Packet Broadcasting Channel," *IEEE Trans. Info. Theory*, 1979.
[7] D. Cirimelli-Low and J.J. Garcia-Luna-Aceves, "ns-3 Simulation of QSMA." [Online] Available at: https://github.com/DylanCirimelli-Low/QSMA-Sim
[8] F. Clazzer et al., "Enhancing Contention Resolution ALOHA Using Combining Techniques," *IEEE Trans. Commun.*, 2018.
[9] A. Laya et al., "Goodbye, ALOHA!," *IEEE Access*, April 2016.
[10] R. Garces and J.J. Garcia-Luna-Aceves, "Collision Avoidance and Resolution Multiple Access with Transmission Groups," *Proc. IEEE INFOCOM '97*, April 1997.
[11] J.J. Garcia-Luna-Aceves and A.N. Masilamani, "NOMAD: Deterministic Collision-Free Channel Access with Channel Reuse in Wireless Networks," *Proc. IEEE SECON '11*, June 2011.
[12] J.J. Garcia-Luna-Aceves, "Carrier-Sense Multiple Access with Collision Avoidance and Detection," *Proc. ACM MSWiM '17*, 2017.
[13] J.J. Garcia-Luna-Aceves, "Carrier Resolution Multiple Access," *Proc. ACM PE-WASUN*, 2017.
[14] J.J. Garcia-Luna-Aceves, "KALOHA: ike i ke ALOHA," *Proc IEEE MASS 2019*, Nov. 2019.
[15] A.C.V. Gummalla and J.O. Limb, "Wireless Medium Access Control Protocols," *IEEE Communications Surveys & Tutorials*, 2000.
[16] C. Horning, "A Standard for the Transmission of IP Datagrams over Ethernet Networks," RFC 894, IETF, April 1984.
[17] G. Jakllari and R. Ramanathan, "A Sync-less Time-Divided MAC Protocol for Mobile Ad-hoc Networks," *IEEE MILCOM '09*, Oct. 2009.
[18] G. Jakllari, M. Neufeld and R. Ramanathan, "A Framework for Frameless TDMA Using Slot Chains," *Proc. IEEE MASS 2012*, Oct. 2012.
[19] R. Jurdak et al., "A Survey, Classification and Comparative Analysis of Medium Access Control Protocols for Ad Hoc Networks," *IEEE Communications Surveys & Tutorials*, 2004.
[20] 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.
[21] A. Muir and J.J. Garcia-Luna-Aceves, "An Efficient Packet-Sensing MAC Protocol for Wireless Networks," *Mobile Networks and Applications*, 1998.
[22] ns3 Network Simulator. On-line: https://www.nsnam.org
[23] M. Ramalho et al., "RTP Payload Format for G.711.0," RFC 7655, IETF, Nov. 2015.
[24] S. Ramanathan and E.L. Lloyd, "Scheduling Algorithms for Multihop Radio Networks," *IEEE/ACM Trans. Networking*, 1993.
[25] A. Sgora et al., ,"A survey of TDMA Scheduling Schemes in Wireless Multihop Networks," *ACM Computing Surveys*, 2015.
[26] Z. Tang and J.J. Garcia-Luna-Aceves, "A Protocol for Topology-Dependent Transmission Scheduling," *Proc. IEEE WCNC '99*, Sept. 1999.
[27] Z. Tang and J.J. Garcia-Luna-Aceves, "Hop Reservation Multiple Access (HRMA) for Ad-Hoc Networks," *Proc. IEEE INFOCOM '99*, 1999.
[28] A. Tzamaloukas, J.J. Garcia-Luna-Aceves, "Channel Hopping Multiple Access with Packet Trains for Ad Hoc Networks," *Proc. IEEE MoMuC '00*, 2000.
[29] F. Tobagi and L. Kleinrock, "The Effect of Acknowledgment Traffic on the Capacity of Packet-Switched Radio Channels," *IEEE Trans. Commun.*, June 1978.
[30] D. J. Vergados et al., "Local Voting: Optimal Distributed Node Scheduling Algorithm for Multihop Wireless Networks," *INFOCOM Workshop '17* , 2017.
[31] W. Xu and G. Campbell, "A Distributed Queuing Random Access Protocol for a Broadcast Channel," *Proc. ACM SIGCOMM '93*, Oct. 1993.
[32] C. Zhu and M. S. Corson, "A Five Phase Reservation Protocol (FPRP) for Mobile Ad Hoc Networks," *Proc. IEEE INFOCOM '98*, 1998.