UC Santa Cruz

UC Santa Cruz Previously Published Works

Title

ALOHA with Queue Sharing

Permalink

https://escholarship.org/uc/item/5544h1p7

Authors

Garcia-Luna-Aceves, J.J.
Cirimelli-Low, Dylan
Mashhadi, Najmeh

Publication Date

2020

Data Availability

The data associated with this publication are within the manuscript.

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

ALOHA with Queue Sharing

J.J. Garcia-Luna-Aceves, Dylan Cirimelli-Low, Najmeh Mashhadi
Computer Science and Engineering Department
University of California
Santa Cruz, CA 95064, USA
jj@soe.ucsc.edu, dcirimel@ucsc.edu, nmashhad@ucsc.edu

***Abstract*—ALOHA with Queue Sharing (ALOHA-QS) maintains most of the simplicity of ALOHA with priority acknowledgments (ACK) and attains the high throughput of transmission scheduling methods that require clock synchronization. Channel access with ALOHA-QS consists of a sequence of queue cycles, with each cycle having one or multiple collision-free transmissions by nodes that have joined the transmission queue and a single request turn to join the queue. The signaling of ALOHA-QS entails adding to packet headers 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 a bit stating whether or not a new node joined the queue successfully. The throughput of ALOHA-QS is compared with the throughput of TDMA with a fixed transmission schedule, ALOHA with priority ACK's, and CSMA with priority ACK's analytically and by simulation.**

***Keywords*-channel access, MAC protocols, ALOHA, CSMA**

I. Introduction

The beauty of the ALOHA channel introduced by Abramson [1] is the simplicity of its transmission strategy: Each node transmits-at-will and then retransmits at some future random time if the transmission is unsuccessful. This simplicity launched a revolution on packet switching over shared multiple-access links that has led to today's standards for WiFi and WiMAX, among many other notable developments in medium access control (MAC) protocols.

It is well known that the simplicity of ALOHA comes at a price. The maximum channel utilization of the basic ALOHA protocol is 18% of the channel capacity, which occurs when one packet is offered to the channel every two packet times. As a result, starting with slotted ALOHA [29], a plethora of MAC protocols has been proposed and implemented over the years focusing mainly on improving the efficiency of channel access. Section II summarizes prior work on MAC protocols intended for single-channel networks [6], [16], [19]. These MAC protocols can be categorized as contention-based, such as ALOHA and CSMA (carrier-sense multiple access) [22]; and contention-free, such as using reservations or elections of time slots in TDMA transmission frames [6].

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.

Many contention-based MAC protocols are relatively simple and improve over the performance of ALOHA, but cannot eliminate multiple-access interference (MAI) completely or provide maximum channel-access delay guarantees [6], [19]. On the other hand, contention-free MAC protocols can provide high channel efficiency and delay guarantees, but are far more complex than ALOHA and other contention-based schemes. Attaining high channel utilization that approaches that of collision-free transmission scheduling while maintaining most of the simplicity of pure ALOHA has proven to be a baffling and very difficult problem. The key contribution of this paper is presenting the first practical solution to this problem; we call this solution *ALOHA with Queue Sharing*, or **ALOHA-QS**.

Section III describes how ALOHA-QS operates. A node with packets to send must first join a transmission queue. A node in the queue is able to transmit its data packets without any MAI every time its turn in the queue comes up. Many approaches have been reported in the past based on similar notions of distributed transmission queues [12], [23], [37]. The novelty of ALOHA-QS is that its transmission strategy simply augments the one used in ALOHA with priority acknowledgments (ACK), and yet it provides collision-free transmissions and maximum channel-access delay guarantees. ALOHA-QS does not require any time slotting or fixed-length transmission frames, and does not need a control handshake between a transmitter and a specific receiver. The signaling in ALOHA-QS consists of including a queue-status update of just a couple of bytes in each packet header by stating the queue size, the position of the sending node in the shared queue, a bit used to indicate that the transmitting node is leaving the queue, and a bit used to indicate whether a new node succeeded in joining the queue.

Section IV shows that ALOHA-QS enables nodes in the queue to transmit during their turn without causing MAI, and allows nodes wishing to be added to the queue to try to join the queue without disrupting in-queue transmissions.

Section V uses a simple analytical model to compute the throughput attained with ALOHA-QS, the average length of the transmission queue, and the average delay incurred in reaching a target queue size. This model takes into account the effect of the number of nodes in the transmission queue and the impact of the probability with which a node in the queue decides to leave.

Section VI compares the performance of ALOHA-QS against the performance of TDMA, ALOHA, and CSMA

using the numerical results from the analytical model and discrete-event simulations in ns-3 [24]. The results show that ALOHA-QS renders very efficient and stable channel access using a transmission strategy that is almost as simple as that of ALOHA with priority ACK's. Section VII presents our conclusions.

II. Related Work

Contention-based MAC protocols developed over the years have taken advantage of physical-layer mechanisms to improve channel utilization. Time slotting was the first physical-layer mechanisms proposed to improve the performance of ALOHA [29] and has been used in many subsequent variants of slotted ALOHA and framed slotted ALOHA [25], in which time slots are organized into frames and users randomly select different time slots for their transmissions. These variants of slotted ALOHA and framed slotted ALOHA include: controlling the probabilities with which nodes transmit in order to offer at most one data packet to the channel per time slot [5], [39]; using machine-learning approaches [9], [10], [38], [36] for nodes to learn which time slots are less utilized; collision-resolution approaches that resolve collisions before allowing new transmissions to occur (e.g., [7]); and using repetition strategies with which each node transmits the same packet multiple times together with physical-layer techniques like code division multiple access (CDMA) or successive interference cancellation (SIC) to attain multi-packet reception (e.g., [20], [21], [26], [30]).

Carrier sensing [22] has been used in many contention-based MAC protocols since the introduction of CSMA and has been used together with additional signaling for collision avoidance, collision resolution, and collision detection [6], [12], [13]. Today, carrier sensing with collision avoidance is an integral part of widely used MAC protocol standards (e.g., WiFi and WiMAX).

Many contention-free MAC protocols have been proposed, and most of them assume the use of transmission frames consisting of a fixed number of time slots, or at least assume the existence of a time-slotted channel [6], [31]. The signaling mechanisms that have been used for scheduling include distributed elections of time slots for broadcast or unicast transmissions [3], [4], [27], and the reservation of time slots based on voting or node-to-node handshakes. [32], [33], [35], [40].

The MAC protocols most relevant to the design of ALOHA-QS are based on the use of distributed queue or a transmission group shared among all nodes accessing the channel.

The Distributed Queue Random Access Protocol (DQRAP) [37] is arguably the first example of this approach, and its design was inspired by the Distributed Queue Dual Bus (DQDB) protocol (IEEE 802.6) [5] and collision-resolution schemes [5], [7]. DQRAP assumes that the channel is time slotted and that each time slot consists of a data slots and multiple control mini-slots. 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 [11]).

Group Allocation Multiple Access (GAMA) [23] improves on DQRAP by eliminating the need for time slotting and the use of control mini-slots. GAMA organizes channel access into cycles, with each consisting of a contention period and a group-transmission period. Maximum channel-access delay guarantees can be attained by limiting the number of transmissions per cycle. A collision avoidance handshake is used in the contention period of a cycle to add new members to the group-transmission period. GAMA uses two signaling packets to manage cycles. A begin-transmission-period packet is sent by each node stating, as a minimum, the position of the sending node in the transmission group and the size of the group; and a transmit-requests packet is sent by the last member of the transmission group after it has transmitted its data.

CARMA-NTS [12] integrates 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) [18] uses a collision-avoidance handshake as 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 limitations of prior protocols based on distributed queues is that they rely on either: (a) time slotting and transmission frames, or (b) explicit signaling between specific transmitter-receiver pairs that requires senders to know whether specific intended receivers are present before the transmission queue can be built.

III. ALOHA-QS

The objectives in ALOHA-QS are to: (a) maintain much of the simplicity of pure ALOHA with priority ACK's; (b) attain collision-free transmissions and delay guarantees accessing the channel; (c) eliminate the need for carrier sensing, clock synchronization, transmission frames consisting of a fixed number of time slots, the ability to distinguish between time periods when the channel is idle or has collisions, or signaling addressed to specific nodes before they become part of the transmission queue.

The design of ALOHA-QS is based on: (a) establishing shared transmission queues by making minor modifications to the transmission strategy used in ALOHA with priority ACK's, and (b) sharing the state of the queue with each packet transmitted.

A. Transmission Strategy

The "transmit-at-will" transmission strategy of ALOHA is slightly modified when priority ACK's are used, so that nodes with packets to send that have received a data packet correctly back off to give the impending ACK priority over their own transmission attempts [34]. The simplest transmission strategy

that can be adopted for this purpose is a non-persistent strategy in which a node that obtains a local packet to send while it is waiting for an ACK for another packet to be heard simply backs off. ALOHA-QS augments this strategy to establish a transmission queue in which transmissions take place without any MAI.

A node in ALOHA-QS trying to join the shared queue transmits after the last turn of the transmission queue. If the attempt is not successful, the node retransmits after the last turn of the transmission queue sometime in the future. Accordingly, channel sharing in ALOHA-QS is a sequence of **queue cycles**. Each queue cycle consists of a sequence of zero or more **queue turns** with transmissions by nodes that have joined the queue, followed by a queue-joining period during which nodes can transmit their requests to join the queue. A request to join the queue is simply a packet. The joining period of a cycle is limited to a single **request turn**. Accordingly, at most one node can be added to the transmission queue at the end of a given queue cycle.

Nodes wishing to join the transmission queue give priority to transmissions by the nodes that have successfully joined the queue. Requests to join the queue can be transmitted only during the *persistence interval* of the current cycle, and any request outside the interval must back off for a randomly chosen number of queue cycles. The persistence interval is defined to be one or multiple queue turns, and for the sake of simplicity we assume that it consists of the last queue turn of a cycle. Any request to join the queue is transmitted at the beginning of the request turn in a cycle. A queue turn and the request turn can last at most one *maximum channel access time* (MCAT), so that no node can monopolize the channel. An MCAT includes: (a) a fixed transmission delay equal to a maximum propagation delay to allow all nodes to hear the transmission in the previous turn, (b) the maximum receive-to-transmit turn-around time, (c) the time needed to transmit the largest packet allowed, and (d) a maximum propagation time of a packet. A turn with a successfully transmitted packet lasts at most one MCAT, and an empty turn and a turn with collisions is assumed to last one MCAT.

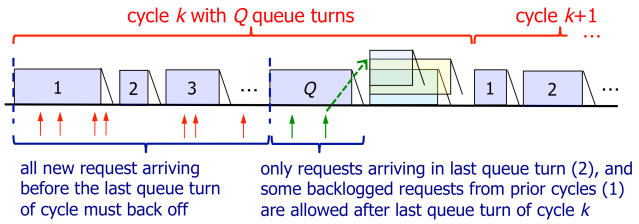

Fig. 1. Transmission strategy in ALOHA-QS

Fig. 1 illustrates the transmission strategy of ALOHA-QS with an example in which a cycle k has Q queue turns and a request turn. Only those requests arriving during the last queue turn of the queue cycle are allowed to take place during the request turn of the cycle together with backlogged requests from some prior cycles. All other requests to join the queue are scheduled for transmission at the end of randomly chosen queue cycles taking place in the future.

B. Queue-Departure Strategy

Nodes leave the transmission queue according to a queue-departure discipline, which can be very simple. In fact, 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. To ensure that transmissions occur only at the beginning of queue turns or request turns once the queue is started, a node that perceives itself as the first node in the queue continues to transmit a packet after one MCAT (corresponding to a request turn), until the second node joins the queue successfully.

C. Queue Sharing Strategy

The only physical-layer feedback used in ALOHA-QS consists of the reception of packets received without interference or errors caused by physical-layer effects. Hence, a node is unable to distinguish between an idle channel or the presence of collisions. Each packet transmitted in ALOHA-QS states the *size* of the queue (Q), the *entry turn* indicating the position of the node in the queue (E), a *data-ending* bit (D) that is set to indicate the last transmission by the node in turn E in the queue, and an *acknowledgment bit* (A) that is set to indicate that a node succeeded joining the queue at the end of the prior queue cycle. This added information would represent only a few extra bytes of added packet-header overhead in networks of even hundreds of nodes, and allows nodes to track the current queue position that should be transmitting, determine when a new cycle must start, eliminate empty turns when nodes decide to leave the transmission queue, and decide when attempts to join the transmission queue are successful.

A node maintains five local variables to implement channel access based on a distributed transmission queue: 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 an ACK flag a stating whether or not there was a successful attempt to join the queue at the end of the previous queue cycle. Nodes share information about their state in the queue with each packet they transmit. More specifically, the header of each packet transmitted includes the queue size Q, the entry turn E of the transmitting node, the data-ending bit D, and the ACK bit A.

Fig. 2 shows the state machine describing the operation of ALOHA-QS. 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. ALOHA-QS 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 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 can be decoded.

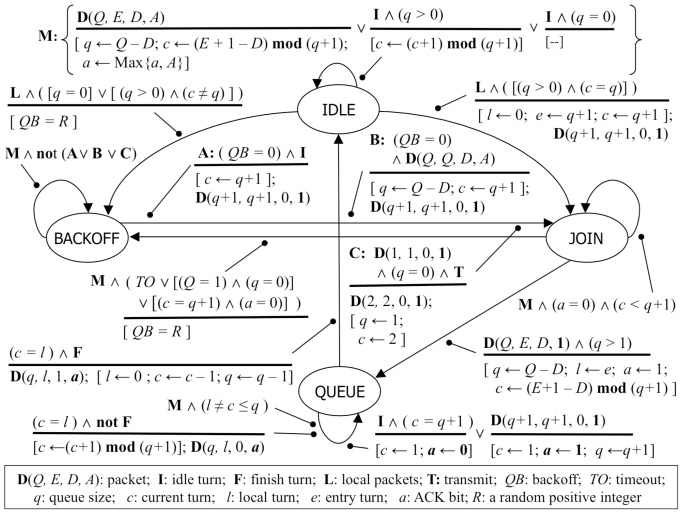

Fig. 2. ALOHA-QS state machine

A packet received or transmitted by a node is denoted by $\mathbf{D}(Q, E, D, A)$ and states the queue size Q, the turn of the transmitting node E, the data-ending bit D, and the ACK bit A as perceived by the node that sends the packet.

The occurrence of a perceived idle period is denoted by **I**. If the transmission queue is not empty, such an idle period lasts one MCAT and may occur due to: a node in the transmission queue not transmitting during its turn, physical-layer effects affecting the reception of a transmitted packet, the occurrence of MAI on packets sent to join the queue at the end of a queue cycle, or the absence of requests at the end of a queue cycle. If the transmission queue is empty, the perceived idle period ends with the first packet received without MAI. The event that a node in the transmission queue is ready to finish its turn in the queue is denoted by **F**. The event that a node that is not in the transmission queue has obtained local packets ready for transmission is denoted by **L**. The event that a node in the BACKOFF state decides to transmit is denoted by **T**.

A node is initialized with the values $q = 0$, $l = 0$, $c = 0$, $e = 0$ and $a = 0$; starts in the IDLE state; and remains in that state until it has local packets to transmit. A node that joins the transmission queue can transmit only at the beginning of its queue turn and must transmit at least one more packet stating that it is leaving the queue by setting the D bit to 1. A node that is not in the queue and has local packets to transmit can access the channel only after the last queue turn of the current cycle. All packets sent to join the transmission queue state $A = 1$, and packets sent during a queue turn state $A = a$. The ACK flag is reset $a = 0$ by nodes in the queue (i.e., in the QUEUE state) when no request is received correctly during the request turn of a cycle.

A node in the IDLE state monitors the activity in the channel and 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. The node activity that takes place in the four states in ALOHA-QS is somewhat similar to what occurs in ALOHA with priority ACK's, which can be viewed as ALOHA-QS with a zero-length transmission queue. More specifically: A node with nothing to send is in an idle state waiting for a packet to send; a node that transmits a packet waits for an ACK (as in the JOIN state); nodes that hear a data packet wait for the ACK to be transmitted (similar to the QUEUE state but without transmissions); and nodes that fail to receive ACK's for their transmitted packets enter the BACKOFF state.

IDLE state: The set of steps taken by the node as a result of monitoring the reception of packets or perception of idle turns in the IDLE state is denoted by **M** in Fig. 2. These monitoring 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 with each packet received. The node advances the current turn modulo $q + 1$ even when idle turns occur. The same monitoring steps are also carried out by a node while in the other three states; this is indicated by **M** next to those states for brevity.

The ACK flag is set to 1 with the first packet received with $A = 1$; hence, a node trying to join the transmission queue receives an ACK if its transmission succeeds and at least one transmission from nodes already in the queue is received in spite of physical-layer effects. If a node receives a packet with the D 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 Fig. 2 by using the value of the D bit as an integer.

A node in the IDLE state that receives local packets to send when the transmission queue is empty or during a turn of a non-empty queue that is not the last transmission turn must transition to the BACKOFF state. In that case, the node computes a random integer R corresponding to the number of queue cycles for its queue backoff (QB). On the other hand, a node in the IDLE state transitions to the JOIN state if it receives local packets to send during the last turn of a non-empty transmission queue. Its packet states: $Q = q + 1$ to indicate an additional turn, $E = q+1$ to request the last turn, $D = 0$, and $A = 1$. The node remembers the value of the requested transmission turn by setting $e = q+1$ and resets its local turn l to 0. Fig. 1 illustrates this aspect of the operation of ALOHA-QS. Only those nodes in the IDLE state that have arrivals in the last turn of the transmission queue can transmit packets after the last turn ends in order to join the queue. If the queue is empty, there is no last transmission turn in the

queue and any node in IDLE state that obtains local packets to send transitions to the BACKOFF state. Requiring a backoff when the transmission queue is empty forces nodes to listen to the channel for a period of time that is long enough to detect packets transmitted without MAI.

JOIN state: A node in the JOIN state remains in the JOIN state until it can transition to the QUEUE state or must transition to the BACKOFF state. If he node is attempting to start the transmission queue, it waits for positive feedback for a timeout interval TO that is longer than an MCAT. The node transitions to the QUEUE state if it receives a packet stating $Q > 1$ and $A = 1$, which indicates that its request packet was sent without MAI. The node transitions to the BACKOFF state if it obtains any of the following indications that its request was unsuccessful: (a) no node in the transmission queue transmits a packet stating $A = 1$; (b) the node attempted to start the transmission queue and the TO to receive a packet from any other node expires; or (c) the node attempted to start the transmission queue and receives a packet stating $Q = E = 1$, which indicates that its own packet was unsuccessful.

If a node transitions to the QUEUE state, it 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. If the node transitions from the JOIN state to the BACKOFF state, it sets its queue backoff QB to a positive random integer stating the number of queue cycles that the node will wait before attempting to join the queue again.

BACKOFF state: A node in the BACKOFF state decrements the value of QB after each complete queue cycle occurs, and processes an input event carrying out the set of monitoring steps **M** while in the BACKOFF state.

A node remains in the BACKOFF state while $QB > 0$ or $QB = 0$ and the request turn of the current cycle has not been reached. The node transitions to the JOIN state when either: (a) $QB = 0$ and the request turn of the current queue cycle is reached ($c = q + 1$); or (b) the queue is empty, the node receives a packet $\mathbf{D}(1, 1, 0, 1)$ starting the queue, and the node decides with some probability to try to join the queue (indicated by **T** in Fig. 2). In the first case, the node updates the queue size q if needed, sets $c = q + 1$ and transmits a packet indicating that it wants to occupy turn $q + 1 > 0$ of the queue by stating $\mathbf{D}(q + 1, q + 1, 0, 1)$. In the second case, the node updates $q = 1$ and $c = 2$, and transmits a packet $\mathbf{D}(2, 2, 0, 1)$.

QUEUE state: A node in the QUEUE state remains in that state until it receives a local signal to end its transmissions, which is denoted by event **F** in Fig. 2. The node simply carries out the set of monitoring steps **M** after an input event when the current turn is not the last turn of the cycle and is not the queue turn of the node ($l \neq c \leq q$).

A node in the QUEUE state that does not receive a packet during the request turn of the current cycle sets c to 1 and resets its ACK flag a to 0. This is done to account for the start of a new queue cycle without a successful join request in the previous cycle. If the node receives a packet correctly during the request turn of the current cycle, the node increases the queue size by one, sets c to 1 and sets its ACK flag a to 1 to account for the start of a new cycle with a successful join request in the previous cycle. The handling of the ACK flag at the end of a queue cycle is illustrated in the example shown in Fig. 3. Nodes a to x in Fig. 3 reset $a = 0$ at the end of cycle k, because collisions took place, and the same would occur if either no packet were transmitted or physical-layer effects prevented the reception of the packet transmitted during the request turn of cycle k.

While a node does not need to end its turn (event **F** is not true) and the current turn corresponds to the turn of the node ($c = l$), then the node increments the value of the current turn by one and transmits a data packet stating $Q = q$, $E = l$, $D = 0$, and $A = a$. If a node needs to transmit its last packet (event **F** is true) and the queue turn of the node corresponds to the current turn ($c = l$), the node transmits its last packet with the current values of the queue size and queue turn, ACK flag, and $D = 1$ to announce its departure. The node then decrements by 1 the values of queue size and current queue turn and sets $l = 0$ to account for its own departure.

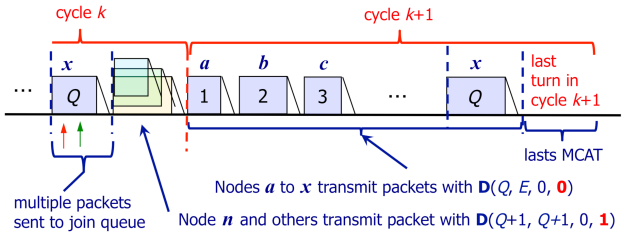

Fig. 3. Example of failed attempts to join transmission queue

IV. Correctness of ALOHA-QS

The service provided by ALOHA-QS consists of allowing any node that joins the transmission queue to transmit packets without MAI and with maximum channel-access delay guarantees. The following theorem proves that this is true under the following assumptions: (a) The channel introduces no errors; (b) the maximum propagation delay between any two nodes is finite; (c) all nodes have direct radio connectivity with one another; and (d) one MCAT is longer than the time needed for a request packet to be received and processed by all nodes.

Theorem 1: ALOHA-QS guarantees that data packets transmitted by nodes that have joined the transmission queue are received without MAI.

Proof: The proof is by induction on the number of nodes Q in the transmission queue.

Basis Case: Nodes must enter the BACKOFF state when they have packets to send and find an empty transmission queue. Accordingly, when the queue is empty, each node with packets to send must wait for a random number of turns, each lasting one MCAT, before it transitions to the JOIN state and transmits its request to start the queue. Once a first node n_1 successfully transmits a request packet stating $Q = E = A = 1$, all nodes must update $q = Q = 1$, $c = 2$, and

$a = A = 1$. As a result, nodes other than n_1 must be on the IDLE or BACKOFF state, and can transmit their request packets only during the request turn following the reception of a packet from n_1. By design, n_1 keeps transmitting a packet stating $Q = E = A = 1$ after waiting a multiple of MCATs, until another node n_2 succeeds transmitting a request packet stating $Q = E = 2$ and $A = 1$. Packets sent in the request turn cannot cause MAI on packets from n_1 when $Q = 1$ because request packets can start only after a packet from n_1 is received, and must end before an MCAT elapses.

If n_2 succeeds, n_1 transitions to the QUEUE state and updates $q = Q = 2$, $c = 3$, and $a = A = 1$, and n_2 must receive the packet from n_1 and transitions to the QUEUE state also setting $q = Q = 2$ and $a = A = 1$. MAI cannot occur on packets from n_1 or n_2 when $Q = 2$ because: (a) n_2 sends a packet in its queue turn, which starts after any packet from n_1 is sent; (b) nodes other than n_1 and n_2 can transmit request packets only in a request turn, which starts after any packet from n_2 is sent; and (c) n_1 cannot start transmitting a packet until a request turn elapses, which lasts longer than any request packet that may be sent during that turn.

Inductive Step: Assume that the result is true for a transmission queue of size Q. Nodes n_1 to n_Q are in the QUEUE state and other nodes are in the IDLE, JOIN, or BACKOFF state. Therefore, they can only transmit their requests in the request turn starting after a packet from n_Q is received or one MCAT elapses for the queue turn of n_Q. Packets sent in a queue turn cannot cause MAI because each queue turn lasts at least as long as a packet sent during the turn and only one node can transmit in a queue turn. No request packet sent can cause MAI on packets sent by n_1, because n_1 waits a request turn after receiving the last packet from n_Q, and a request turn either results in a request with no MAI or lasts is longer than the collisions of any request packets. Node n_{Q+1} joins the transmission queue when its request, sent after the packet from n_Q, does not have any MAI. Hence, the result is true. ■

V. Performance Analysis of ALOHA-QS

A. Model and Assumptions

We assume the same traffic model first introduced by Abramson [1] to analyze ALOHA. According to this model, a very large number of stations constitute a Poisson source sending data packets of equal length δ to the the channel with an aggregate mean rate of λ packets per unit time. A node that is forced to backoff does so for a random amount of time that is much larger than the time needed for a successful packet transmission and such that packet transmissions for new arrivals and backlogged arrivals can be assumed to be independent of one another. Multiple access interference (MAI) is the only source of errors, multiple concurrent transmissions to the common channel must all be retransmitted and any packet propagates to all nodes with the same propagation delay. The only physical-layer feedback is the decoding of packets received without MAI. The system operates in steady state, with no possibility of collapse.

Requests to join the transmission queue occur with an aggregate mean rate of λ packets per unit time. 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. A fixed turn-around time of ω seconds is assumed for transitions from receive-to-transmit or transmit-to-receive modes.

B. Throughput of ALOHA-QS

The throughput of a MAC protocols is the percentage of time that the channel is used for the transmission of packets without MAI. Given that ALOHA-QS 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. The following theorem states this result assuming that each queue turn or request turn lasts T seconds.

Theorem 2: The throughput of ALOHA-QS is

$$S = \frac{\delta\left(\mu\bar{Q} + P_s\right)}{T(\bar{Q} + 1)} \tag{1}$$

where μ is the probability that a node transmits a packet during its queue turn, $\bar{Q}$ is the average size of the transmission queue, T is the duration of a transmission turn, and P_s is the probability of success during the request turn of a cycle.

Proof: The throughput of ALOHA-QS is simply the ratio of the time $\bar{U}$ spent transmitting packets without MAI in an average queue cycle divided by the time $\bar{C}$ that such a cycle lasts. $\bar{C}$ equals $T(\bar{Q} + 1)$ given that the average queue cycle has $\bar{Q} + 1$ turns and each turn lasts T seconds. Each queue turn of a cycle contains a successful packet with probability μ lasting δ seconds and the request turn of a cycle contains a successful packet with probability P_s. Therefore, $\bar{U}$ equals $\delta(\mu\bar{Q} + P_s)$. The result follows by taking the ratio $\bar{U}/\bar{C}$. ■

Taking into account our assumption that the arrival of requests to join the transmission queue is Poisson with parameter λ leads to the following result.

Corollary 1: The throughput of ALOHA-QS is

$$S = \frac{\delta\left(\mu\bar{Q} + \lambda T e^{-\lambda T}\right)}{T(\bar{Q} + 1)} \tag{2}$$

Proof: The result follows from Theorem 1 and the fact that P_s equals the probability that only one request occurs during the last queue turn of an average cycle, which is $\lambda T e^{-\lambda T}$. ■

We observe that making $\bar{Q} = 0$ and $T = \delta$ in Eq. (2) results in the known throughput result of slotted ALOHA [29] with $\lambda T = G$. This should be expected, given that slotted ALOHA can be viewed as ALOHA-QS when the size of the transmission queue is kept empty, MAC-level ACK's in ALOHA-QS are sent in packet headers, and nodes establish virtual time slotting.

C. Average Size of Transmission Queue in ALOHA-QS

Given that at most one node may join or leave the transmission queue in any 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.

Fig. 4 illustrates channel utilization in ALOHA-QS. The figure shows cycle k having a queue with m turns and a single request to join the queue, which results in a success and hence cycle $k+1$ has $m+1$ nodes in the queue. Three requests to join the queue take place in cycle $k+1$, which results in a failure 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 two join requests occurring in cycle $k+3$, which results in a failure to increase the queue size.

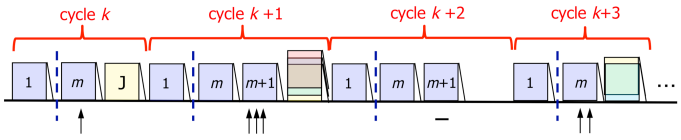

Fig. 4. Channel utilization in ALOHA-QS

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

Theorem 3: The average queue size in ALOHA-QS is

$$\bar{Q} = m + \frac{P_s(1-q)}{q - P_s} \text{ with } P_s < q \tag{3}$$

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: The proof of this theorem is presented in [15] in the context of QSMA, which is a MAC protocol that uses a queue-sharing approach that is very similar to the design of ALOHA-QS. ■

Assuming that the arrival of requests to join the transmission queue is Poisson distributed with parameter λ leads to the following result by substituting $\lambda T e^{-\lambda T}$ for P_s in Eq. (3).

Corollary 2: If the arrival of requests to join the transmission queue is Poisson with parameter λ the average queue size in ALOHA-QS is

$$\bar{Q} = m + \frac{(1-q)\lambda T e^{-\lambda T}}{q - \lambda T e^{-\lambda T}} \text{ with } \lambda T e^{-\lambda T} < q \tag{4}$$

D. Average Delay Reaching Target Queue Size

Fig. 5 illustrates the random evolution of reaching a target queue size of m starting with the first 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 $k = 1, 2, ..., m-1$. Given that the system is in equilibrium, there must be a first packet transmitted without MAI with probability 1. Once that first packet is transmitted, all nodes must transmit their own packets during the request turn that follows the first packet, and in general by the last packet transmitted during a queue turn. A packet transmitted during a request turn succeeds with probability P_s. Therefore, for any given k between 1 and $m-1$ the queue size k increases by one with probability P_s and remains the same with probability $1-P_s$.

The node that starts the transmission queue continues to transmit its request packet after a random backoff number of MCATs elapses. If the average number of backoff MCATs incurred between consecutive transmissions of request packets is R when $k=1$, then the average number of transmission turns to advance to state $k=2$ or remain in state $k=1$ is $R+2$ to account for two successful transmission turns or one successful turn followed by an idle or unsuccessful turn. For $k>1$, the number of transmission turns incurred in the attempt is $k+1$.

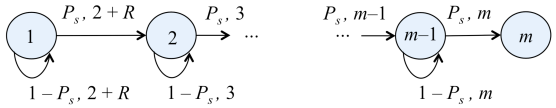

Fig. 5. Random evolution reaching target queue size m in ALOHA-QS

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:

$$\bar{D}_1 = P_s((R+2)T + \bar{D}_2) + (1-P_s)((R+2)T + \bar{D}_1) \tag{5}$$

$$\bar{D}_k = P_s((k+1)T + \bar{D}_{k+1}) + (1-P_s)((k+1)T + \bar{D}_k)$$

$$\text{with } k = 2, ..., m-2 \tag{6}$$

$$\bar{D}_{m-1} = P_s(m\,T) + (1-P_s)(m\,T + \bar{D}_{m-1}) \tag{7}$$

Solving Eqs. (5) to (7) for $\bar{D}_1$ provides the desired average delay in reaching the target queue size, which is

$$\bar{D}(m) = \left(\frac{m(m+1)}{2} + R - 1\right)\frac{T}{P_s} \tag{8}$$

The above result is intuitive. On average, it takes P_s^{-1} queue cycles to increase the queue size in the current state, and each queue cycle at state k has $k+1$ turns. If we only consider the number of cycles needed to reach the target queue size of m from 1 and set $R=0$ we obtain $\bar{C}(m) = (m-1)P_s^{-1}$. Furthermore, given an average queue size $\bar{Q}$, the average delay $\bar{D}_{\bar{Q}}$ incurred in adding a new node to the transmission queue can be obtained as in Eq. (7), that is, $\bar{D}_{\bar{Q}} = P_s(\bar{Q}\,T) + (1-P_s)(\bar{Q}\,T + \bar{D}_{\bar{Q}})$, which results in $\bar{D}_{\bar{Q}} = \bar{Q}TP_s^{-1}$.

If the arrival of new and retransmitted queue-join requests is Poisson with parameter λ, then $P_S = \lambda T e^{-\lambda T}$. Substituting in Eq. (8) we have

$$\bar{D}(m) = [m(m+1) + 2(R-1)]e^{\lambda T}/2\lambda \tag{9}$$

E. Impact of Limited Signaling Overhead on Throughput

A transmission turn lasts $T = \delta + \omega + \tau$, which accounts for the turn-around time and the time to transmit and propagate a packet. As Eq. (2) indicates, the throughput of ALOHA-QS quickly approaches δ/T as the average queue size $\bar{Q}$ grows. This constitutes very high throughput, because $\delta \gg \omega + \tau$.

VI. Performance Comparison

We compare the performance of ALOHA-QS with the performance of three other MAC protocols that are representative of contention-based and schedule-based channel access using numerical results from our analytical model and simulations carried out using ns-3 [24]. We consider CSMA with ACK's, and ALOHA with ACK's because the design of ALOHA-QS seeks to maintain most of their simplicity. We consider TDMA with a fixed transmission schedule because it attains the highest throughput of a schedule-based MAC protocol using time slots. The source code of the ns-3 simulations we discuss is publicly available [8].

A. Numerical Results from Analytical Model

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

1) Throughput Results: We make some simplifying assumptions about packet arrivals to compare the throughput of ALOHA-QS, ALOHA with ACK's, and CSMA with ACK's, and TDMA with a fixed transmission schedule. The combined arrival of all packets is Poisson with parameter λ. For ALOHA and CSMA this means data-packet arrivals. For ALOHA-QS, a node in the queue is assumed to transmit during its own queue turn, provided that it has a data-packet arrival during the previous queue turn, or request turn if the node has the first queue turn. Furthermore, if there is one or multiple arrivals in a given turn (a queue turn or a request turn), then there is at least one arrival for the next turn in the cycle. A similar simplifying assumption is made for TDMA.

With these simplifying assumptions, the intensity of traffic from nodes in the queue in ALOHA-QS and active nodes in TDMA correlates with the total traffic intensity, and also means that $\mu = 1 - e^{-\lambda T}$ in Eq. (1).

Given that the only overhead incurred in TDMA with a fixed schedule is the extra time per time slot needed to account for turn-around times and propagation delays, the throughput of TDMA is simply

$$S_{tdma} = \delta\mu/(\delta + \omega + \tau) \tag{10}$$

Fig. 6 shows the throughput of TDMA (Eqs. (10)), ALOHA-QS (Eqs. (3) and (4)), ALOHA with priority ACK's (Eq. (20) in [14]), and CSMA with priority ACK's (Eq. (18) in [13], which is similar to Eq. (26) in [34]) assuming a channel data rate of 1 Mbps, physical distances of 500 meters, and data packet of 1500 bytes, which renders a normalized propagation delay of 1×10^{-4}. The turn-around time ω is assumed to be the same as a propagation delay, and an ACK in ALOHA and CSMA consists of 40 bytes. Making $\omega = \tau$ results in the length of a transmission turn in ALOHA-QS being $T = \delta + 3\tau$ (see Section V-E), and the length of a time slot in TDMA being $\delta + 2\tau$.

The results illustrate the high efficiency and stability of ALOHA-QS. Even when the queue size is very small (e.g., just five turns as Fig. 6 illustrates), ALOHA-QS is far more efficient than ALOHA and CSMA. In addition, ALOHA-QS is stable at any load and quickly approaches the throughput attained with TDMA as the size of the transmission queue increases. This is remarkable, given that no special physical-layer support is required in ALOHA-QS other than the decoding of received packets.

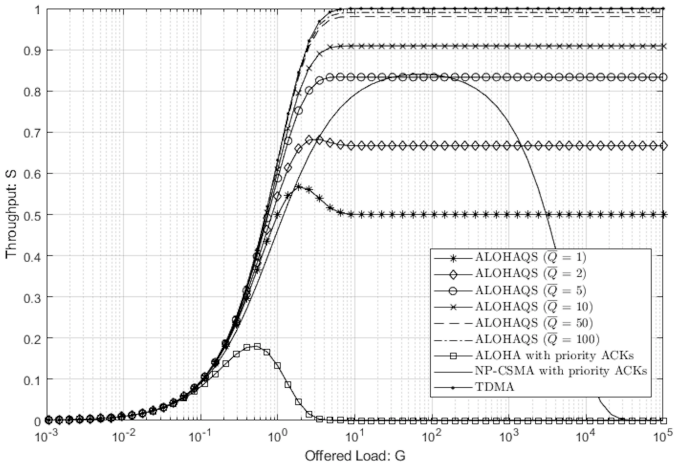

Fig. 6. Throughout of TDMA with a fixed schedule, ALOHA-QS, ALOHA with ACK's, and CSMA with ACK's

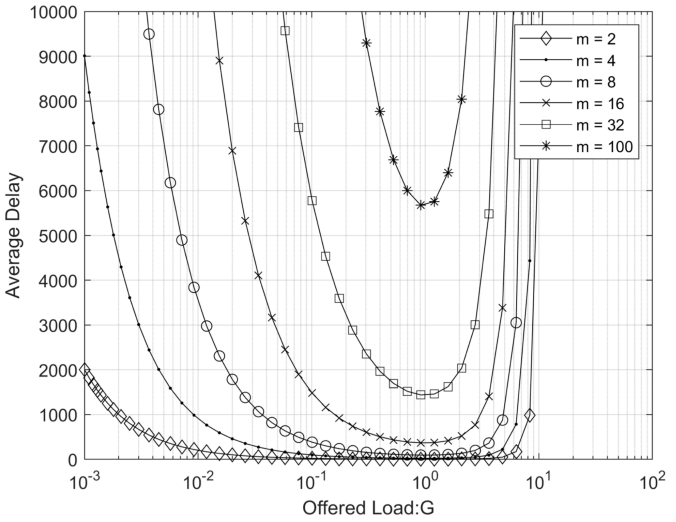

Fig. 7. Average delay in transmission turns reaching target queue size

2) Delay Results: Fig. 7 shows the average delay in terms of transmission turns incurred in ALOHA-QS to reach differ-

ent target queue sizes as a function of the normalized number of join requests, which is set to $G = \lambda T$ with $T = \delta$ when $p = 1$ and $R = 0$. It is clear from Eq. (9) and the figure that the average delay incurred in reaching a target queue size would become very large when m is large and the average number of requests per request turn is much larger than 1. This is because each success takes many cycles to occur and each cycle has many queue turns. In a finite network, the arrival rate of queue-join requests decreases as more nodes join the queue. This makes the results more promising, because they indicate that a queue size that includes all active nodes can be easily reached in just a few seconds in networks with hundreds of nodes.

B. Results from Simulation Experiments

1) Simulation Setup and Scenario: We compare TDMA with a fixed schedule, ALOHA-QS, ALOHA with ACK's, and CSMA with ACK's using the ns-3 simulator [24]. The scenario assumes fully-connected topologies of 10 or 50 nodes that always have data packets to send. The experiment uses a $300m \times 300m$ grid for random placement of nodes, 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. We measure the average throughput of the four protocols and the time that a node takes in ALOHA-QS to transition to the QUEUE state from the time when a it first receives a packet to send.

All contention-based protocols 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. Data packets in ALOHA-QS add two bytes, which suffices to carry the Q, E, D, and A feedback for up to 128 nodes. The target queue size in ALOHA-QS is set to accommodate any number of nodes, and the time slots in fixed-schedule TDMA accommodate the largest packet size.

2) Throughput Results: Table I shows the normalized throughput for TDMA, ALOHA, CSMA and ALOHA-QS.

TABLE I
NORMALIZED THROUGHPUT OF MAC PROTOCOLS

Network Size	Protocol	Packet Size (bytes)		
		218	**50% / 50%**	**1500**
10 nodes	ALOHA	.271	.132	.096
	CSMA	.649	.813	.872
	TDMA	.263	.631	.999
	ALOHA-QS	**.941**	**.975**	**.984**
50 nodes	ALOHA	.028	.003	<.001
	CSMA	.647	.811	.870
	TDMA	.263	.631	.999
	ALOHA-QS	**.984**	**.993**	**.995**

We consider data payloads of 218 bytes, which correspond to a typical VoIP frame [28];1500 bytes, which is the typical payload MTU of an IP packet [17]; and an even combination of them. Results are shown for networks with 10 and 50 nodes. The table shows the mean of 10 trials for each experiment lasting 10 min. Standard deviations were much smaller than .01% for all MAC protocols because of the long duration of each experiment. As the results show, ALOHA-QS attains far better throughput than ALOHA and CSMA independently of the network size or payload type, with better than 90% throughput in all cases. The small throughput degradation with small payloads in ALOHA-QS results from the relatively lager overhead of propagation delays and turn-around times in queue turns with short packets.

CSMA performs better with large payloads, because the overhead of priority ACK's is comparatively smaller than with small data packets.

ALOHA performs much worse with large and mixed payloads because the average vulnerability period of a data packet is larger. As predicted in [2], the throughput degradation for the case of mixed payloads results from small data packets interfering with large packets that occupy the channel for large portions of time. The throughput of ALOHA vanishes as the network size increases due to the amount of multiple access interference they create.

TDMA with a fixed schedule performs better than ALOHA-QS only when large data packets are transmitted most of the time, each occupying most of a time-slot time. ALOHA-QS outperforms TDMA when either small packets are transmitted or data traffic involves a heterogeneous mix of packet lengths. The small throughput in TDMA with small data packets and mixed payloads can be addressed either by using short time-slot times or packing multiple packets in the same time slot. However, using short time slots requires large data packets to be fragmented and transmitted in multiple time slots, which adds considerable overhead and results in much longer delays in the delivery of large data packets. On the other hand, packing multiple data packets in each time slot makes the channel access protocol more complex.

3) Delay Results: Fig. 8 shows the average delay incurred by each node to join the distributed queue in ALOHA-QS in the order in which each node joins the queue when data packets payloads have 1500 bytes.

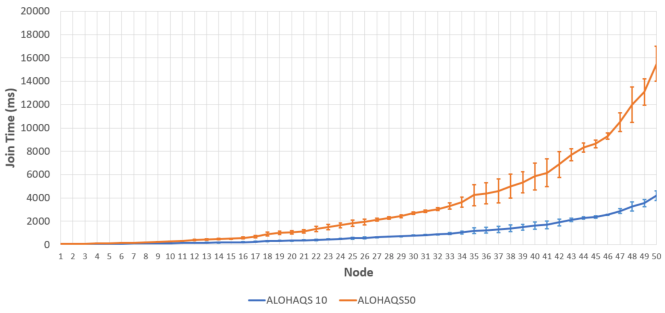

Fig. 8. Normalized delay joining the queue per node in ALOHA-QS

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. 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 all nodes join the queue well within 18 seconds, which would be more than adequate in most network deployments. However, the simplistic queue-joining approach assumed in this paper can and should be improved to make ALOHA-QS more effective in the presence of long propagation delays.

VII. Conclusions and Future Work

We introduced ALOHA-QS, a MAC protocol that quickly attains collision-free transmissions and provides maximum channel-access delay guarantees in fully-connected wireless networks without the need for time slotting at the physical layer or the use of explicit handshakes that require transmitters to know the identity of intended receivers before such nodes have joined the transmission queue.

The signaling overhead in ALOHA-QS is very small. Each packet header simply states the queue size, a position in the queue, a bit informing whether the transmitter is leaving the queue, and a bit serving as an ACK to a request to join the queue. In addition, request packets are much smaller than average-length data packets

The performance-comparison results show that ALOHA-QS provides the best features of contention-based and schedule-based MAC protocols. ALOHA-QS is almost as simple as ALOHA with priority ACK's, and renders channel efficiency comparable to or better than what TDMA provides.

Several ALOHA-QS optimizations can be made and deserve further study. In particular, the efficiency of ALOHA-QS 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. Multiple requests turns could be allowed in a transmission cycle to reduce the delays incurred in reaching target queue sizes. Lastly, carrier sensing could be used in the context of shared transmission queues; an approach that does this is presented in [15].

References

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

[2] N. Abramson, "The Throughput of Packet Broadcasting Channels," *IEEE Transactions on Communications*, Jan. 1977.

[3] 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.

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

[5] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, 1992.

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

[7] J. Capetanakis, "Tree Algorithm for Packet Broadcasting Channel," *IEEE Transactions on Information Theory*, 1979.

[8] D. Cirimelli-Low and J.J. Garcia-Luna-Aceves, "ns-3 Simulation of QSMA." Online: https://github.com/DylanCirimelli-Low/QSMA-Sim

[9] Y. Chu et al., "ALOHA and Q-Learning based Medium Access Control for Wireless Sensor Networks," *Proc. IEEE ISWCS '12*, 2012.

[10] Y. Chu et al, "Application of Reinforcement Learning to Medium Access Control for Wireless Sensor Networks," *Engineering Applications of Artificial Intelligence*, 2015.

[11] A. Laya et al., "Goodbye, ALOHA!," *IEEE Access*, April 2016.

[12] R. Garces and J.J. Garcia-Luna-Aceves, "Collision Avoidance and Resolution Multiple Access with Transmission Groups," *Proc. IEEE INFOCOM '97*, April 1997.

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

[14] J.J. Garcia-Luna-Aceves, "KALOHA: ike i ke ALOHA," *Proc IEEE MASS '19*, Nov. 2019.

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

[16] A.C.V. Gummalla and J.O. Limb, "Wireless Medium Access Control Protocols," *IEEE Communications Surveys & Tutorials*, 2000.

[17] C. Horning, "A Standard for the Transmission of IP Datagrams over Ethernet Networks," RFC 894, IETF, April 1984.

[18] G. Jakllari and R. Ramanathan, "A Sync-less Time-Divided MAC Protocol for Mobile Ad-hoc Networks," *IEEE MILCOM '09*, Oct. 2009.

[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] E. Khaleghi et al., "Near-Far Effect on Coded Slotted ALOHA," *Proc. IEEE PIMRC 2017 Workshop on The Internet of Things (IoT)*, Oct. 2017.

[21] C. Kissling, "Performance Enhancements for Asynchronous Random Access Protocols over Satellite," *Proc. IEEE ICC '11*, June 2011.

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

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

[24] ns3 Network Simulator. On-line: https://www.nsnam.org

[25] H. Okada et al., "Analysis and Application of Framed ALOHA Channel in Satellite Packet Switching Networks - FADRA Method," *Electron. Commun. in Japan*, 1977

[26] E. Paolini, G. Liva, and M. Chiani, "Coded Slotted ALOHA: A Graph-Based Method for Uncoordinated Multiple Access," *IEEE Transactions on Information Theory*, Dec. 2015.

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

[28] M. Ramalho et al., "RTP Payload Format for G.711.0," RFC 7655, IETF, Nov. 2015.

[29] L.G. Roberts, "ALOHA Packet System with and without Slots and Capture," *ACM SIGCOMM CCR*, April 1975.

[30] F. C. Schoute, "Dynamic Frame Length ALOHA," *IEEE Transactions on Communications*, 1983.

[31] A. Sgora et al., ,"A survey of TDMA Scheduling Schemes in Wireless Multihop Networks," *ACM Computing Surveys*, 2015.

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

[33] Z. Tang and J.J. Garcia-Luna-Aceves, "Hop Reservation Multiple Access (HRMA) for Ad-Hoc Networks," *Proc. IEEE INFOCOM '99*, 1999.

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

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

[36] S. Wang et al., , "Deep Reinforcement Learning for Dynamic Multichannel Access in Wireless Networks," *IEEE Transactions on Cognitive Communications and Networking*, 2018.

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

[38] Y. Yan et al., "Adaptation of the ALOHA-Q Protocol to Multi-Hop Wireless Sensor Networks," *Proc. IEEE European Wireless '14*, May 2014.

[39] J. Yu and L. Chen, 'Stability Analysis of Frame Slotted Aloha Protocol," *IEEE Transactions on Mobile Computing*, May 2017.

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