# An Adaptive Tree Algorithm to Approach Collision-Free Transmission in Slotted ALOHA

Molly Zhang
University of California, Santa Cruz
mollyzhang@ucsc.edu

Luca de Alfaro
University of California, Santa Cruz
luca@ucsc.edu

J.J. Garcia-Luna-Aceves
University of California, Santa Cruz
jj@soe.ucsc.edu

## ABSTRACT

A new reinforcement-learning approach is introduced to improve the performance of the slotted ALOHA protocol. Nodes use known periodic schedules as base policies with which they can collaboratively learn how to transmit periodically in different time slots to limit packet collisions. The Adaptive Tree (AT) algorithm is introduced for this purpose, which results in AT-ALOHA. It is shown that nodes using AT-ALOHA quickly converge to transmission schedules that are virtually collision-free, and that the throughput of AT-ALOHA resembles that of TDMA, but without the need to define transmission frames with a given number of time slots. AT-ALOHA is shown to attain better throughput and fairness than slotted ALOHA with exponential back offs and ALOHA-Q (framed slotted ALOHA with Q learning).

## CCS CONCEPTS

• **Networks** → **Network protocol design**; • **Computing methodologies** → **Reinforcement learning**.

**ACM Reference Format:**
Molly Zhang, Luca de Alfaro, and J.J. Garcia-Luna-Aceves. 2020. An Adaptive Tree Algorithm to Approach Collision-Free Transmission in Slotted ALOHA. In *Workshop on Network Meets AI ML (NetAI'20), August 14, 2020, Virtual Event, NY, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3405671.3405817

## 1 INTRODUCTION

The ALOHA protocol [1] was the first medium access control (MAC) protocol for packet switching over wireless networks. Its inherent simplicity makes ALOHA and its variants very attractive in various types of untethered networks today. Of particular interest are underwater acoustic networks, satellite networks, space networks, wireless networks in which hidden-terminal interference is prevalent, and IoT deployments consisting of very simple devices. However, this simplicity comes at the price of performance, with a maximum throughput of only 18% of the available bandwidth.

As a result, as Section 2 summarizes, many attempts have been made to improve the performance of ALOHA by using time slots and transmission frames consisting of a fixed number of time slots. Machine learning has been used more recently in the selection of time slots by nodes with packets to send in a way that reduces the likelihood of packet collisions and hence attains much higher throughput.

The limitations of this prior work include: requiring network nodes to agree on the number of slots per transmission frame a priori (frame slotted ALOHA), requiring the use of transmission frames and slow learning in quickly changing channel conditions.

This paper introduces a new reinforcement-learning approach to improve the performance of slotted ALOHA without the need for transmission frames. Nodes use known periodic schedules as base policies used for reinforcement learning. Nodes carry out collaboratively learning using the policies to determine how to transmit periodically in a way that packet collisions are minimized. The Adaptive Tree (AT) algorithm is introduced for this purpose, which results in AT-ALOHA. The intuition behind the design of AT-ALOHA is to requires nodes to learn how to take turns by using simple policies. Node with a packet to send decided to either transmit or wait according to their policy, and node adopts and changes its transmission policy according to the AT algorithm described in Section 3.

Section 4 compares the performance of AT-ALOHA against the performance of the traditional approach used in framed slotted ALOHA, which consists of having nodes that experience collisions back off exponentially in order to reduce congestion, as well ALOHA-Q [6, 7], which uses Q-learning in the context of framed slotted ALOHA. The results of simulation experiments illustrate the fact that AT-ALOHA attains very high throughput and fairness, compared to ALOHA-Q or slotted ALOHA with exponential backoffs. AT-ALOHA also learns much faster than ALOHA-Q on which time slots to use to avoid collisions, without the need to define transmission schedules consisting of a fixed number of time slots.

## 2 RELATED WORK

Several variants of ALOHA have evolved over the years to allow more efficient sharing of common channels in untethered networks. Slotted ALOHA [20] forces transmissions to occur at the beginning of time slots defined at the physical layer and reduces the time during which transmissions are vulnerable to multiple-access interference (MAI) by half and hence doubles the maximum throughput attainable with pure ALOHA. Framed slotted ALOHA [17], based on slotted ALOHA, organizes the channel into transmission frames consisting of a fixed number of time slots and lets each user select which time slots to use for its transmissions. A few more schemes based on framed slotted ALOHA consist of using repetition strategies with which each node transmits the same packet multiple times, and relying on physical-layer techniques (e.g., code division multiple access and successive interference cancellation) to improve throughput [14, 15, 18, 21].


*NetAI'20, August 14, 2020, Virtual Event, NY, USA*

ACM ISBN 978-1-4503-8043-0/20/08…$15.00
https://doi.org/10.1145/3405671.3405817

The ALOHA protocol and its time-slotted variants achieve bandwidth sharing by transmitting greedily, and then adopting a backoff policy in case of collisions. This, however, leads to poor channel utilization. To address this limitation, a deeper level of coordination is needed in which nodes adapt to each other's behavior so that most transmission slots can be utilized without collisions or only a few.

Many shcedule-based MAC protocols have been proposed in the past in which distributed algorithms are used assuming transmission frames consisting of a fixed number of time slots such that nodes select time slots in a way that eliminates multiple-access interference. The algorithms that have been proposed in this context include distributed elections of time slots for broadcast or unicast transmissions [2, 3, 16, 19], and the reservation of time slots based on voting and signaling similar to collision avoidance handshakes [23–25].

Some approaches allow the use of variable-length transmission frames by using lexicographic ordering of the identifiers of transmitting nodes, geographical or virtual coordinates related to the connectivity of nodes [8, 9, 16], or a common tree of periodic schedules of variable periods that are powers of two [13].

The disadvantage of all these approahes is the added signaling complexity required to establsh internodal coordination in order to attain TDMA scheudles in a distributed manner. An alternative to achieving the desired coordination without complex signaling is to apply reinforcement learning (RL). A well-known family of RL methods is called the *expert method* and involves choosing which of a small number of policies (experts) to follow [5, 10]. The advantage of expert-based learning lies in its relative simplicity, and fast adaptation. This is the approach followed in the ALOHA-Q protocol (framed slotted ALOHA with Q-learnjng) by Chu et al. [6, 7].

In ALOHA-Q, the time-slots are grouped in frames of fixed length $N$. Each node has $N$ policies, where policy $k \in [1, \ldots, N]$ prescribes to transmit in the $k$-th time slot of a frame. The nodes continually tune a quality value associated with each policy, and at every frame, each node transmits according to its highest-quality policy. When the number of active nodes approaches $N$, the overall network throughput after adaptation can approach 1. Unfortunately, when the number of active nodes is below $N$, some bandwidth goes unused. Furthermore, when the number of active nodes exceeds $N$, collisions are bound to occur, and ALOHA-Q reverts to backoff strategies that ultimately degrade the performance to the $1/e$ limit as the number of active nodes grows.

The limitations of ALOHA-Q stem from relying on a small, fixed set of policies. To overcome this limitation, the use of *deep reinforcement learning* (deep RL, in sort) has been recently proposed by Yu el al. [26]. The difficulty in applying deep reinforcement learning to network protocols is that the state-space can be very large. In a network with $n$ nodes, the history of the past $m$ transmission slots gives rise to a state space of size at least $n^m$; large values of $m$ are needed to enable nodes to base their decisions on the transmission schedules of other nodes. The limitations of deep RL approach includes lengthy adaptation time, the complexity and computational requirements of each node, and the lack of guarantees. In [26], only networks with at most two deep RL nodes have been demonstrated, and even so, the adaptation time is to the order of 10,000 time slots.

AT-ALOHA is similar to ALOHA-Q in being rooted in expert-based RL. Unlike ALOHA-Q, AT-ALOHA eliminates the use of the fixed-length transmission frames that are assumed in all prior approaches aimed at improving slotted ALOHA. Rather, the "experts" of AT-ALOHA consist in a tree of periodic schedules. The schedules have periods that are powers of two: the root schedule transmits always (period 1), and the two children of a schedule each transmit with double period, and half of the transmissions, as the parent schedule. By foregoing a fixed frame, and by being able to transmit according to multiple schedules simultaneously, AT-ALOHA is more flexible in the use of bandwidth than ALOHA-Q, and achieves high network utilization under a wide range on network loads. Differently however from classical expert-based RL, and from ALOHA-Q, AT-ALOHA does not associate a quantitative value to each schedule: rather, it just remembers which schedules are good enough to be played. This considerably reduces the memory requirements of the protocol, and simplifies its implementation.

The schedule trees used in AT-ALOHA directly recall the conflict resolution scheme of Capetanakis et al. [4]. However, while the scheme of Capetanakis et al. aimed at resolving each conflict as it arose, our tree schedule is used to let nodes learn tranmission policies that avoid conflicts to begin with. Similar schedule trees have also been used in akllari et al. [13], where however the allocation of schedules to nodes is performed via a central authority to which the nodes need to send bandwidth requests.

# 3 AT-ALOHA

## 3.1 Overview

*Adaptive Tree* ALOHA (AT-ALOHA) is an approach that combines the fast adaptation of ALOHA-Q with the generality that stems from using a very large set of policies. AT-ALOHA nodes share a time-slotted transmission channel. Its operation is very much the same as that of slotted ALOHA, except for the transmission strategy used by a node with a packet to send during a given time slot. Policies in AT-ALOHA consist of the union of simple periodic schedules. The periodic schedules are chosen as base policies because a natural way to achieve coordination is to take turns. The update mechanism of AT-ALOHA policies is loosely based on regret learning, except that rather than storing a quantitative measure of each policy's quality, the protocol just stores which schedules it is following at any particular moment.

In AT-ALOHA a node has the choice of either transmitting or waiting. We denote these two actions by $T$ and $W$, respectively. If all nodes with packets to send wait, then the time slot is empty; if exactly one node transmits, then the slot has a successful transmission; if more than one node transmits, then a collision occurs. We denote these three outcomes by $E$, $S$, and $C$, respectively.

## 3.2 Schedules and Policies

We assume that every node has a clock $t$ that counts the number of time slots. A (periodic) *schedule* $(i, m)$ prescribes sending at all times $t$ such that $t \mod 2^m = i$; we let $T(i, m) = \{t \mid t \mod 2^m = i\}$ be the set of transmission times of $(i, m)$. Let $S = \{(i, m) \mid m > 0, 0 \leq i < 2^m\}$ be the set of all such periodic schedules. The schedules in $S$ can be arranged in a tree, where the schedule $(i, m)$ has $(i, m + 1)$

and $(i + 2^m, m + 1)$ as children: a child schedule transmits in only half the time slots as its parent.

Like [13], AT-ALOHA does not constrain nodes to transmit according to a single schedule. Doing so would result in the bandwidth of individual nodes assuming only values corresponding to the fractional powers of 2: $1, 1/2, 1/4, 1/8, \ldots$. Rather, AT-ALOHA uses a *policy* $\pi \subseteq S$ consisting of a set of schedules; the transmit times of $\pi$ are the union of the transmit times of the individual schedules in $\pi$, or $T(\pi) = \cup_{s\in\pi} T(s)$. Figure 1 depicts the schedule tree, along with a policy consisting of two schedules.

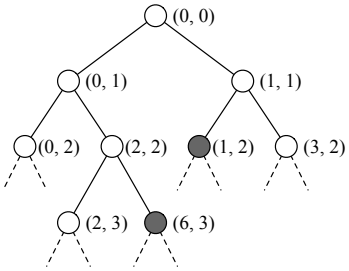


**Figure 1: A (partial) depiction of the schedule tree. The dark nodes correspond to policy $\pi = \{(1, 2), (6, 3)\}$.**

AT-ALOHA policies are required to be in *normal form:*

- *No descendants:* if $s, s' \in \pi$, then $s'$ is not a descendant of $s$ in the policy tree.
- *No siblings:* for all $m > 0$, and $0 < i < 2^m$, it is never the case that both $(i, m + 1) \in \pi$ and $(i + 2^m, m + 1) \in \pi$.

The policy in Figure 1 is in normal form. Every policy can be put in normal form without affecting its transmission times by first eliminating descendant schedules, and then by merging all sibling schedules $(i, m + 1)$ and $(i + 2^m, m + 1)$ into $(i, m)$ (repeating the merging until no siblings remain). The set of AT policies consists of all finite sets of schedules that are in normal form; we denote it by $P$.

## 3.3 The AT Algorithm

Algorithm 1 specifies the AT algorithm. AT updates the time-slot counter $t$, the policy $\pi \in P$, and two probabilities $p_b$ and $p_k$, known as the *barge-in* and *kindness* probabilities, whose role we describe in the following. The policy, time-slot counter, and probabilities are local to each node; in particular, nodes do not need to agree on the numbering of time slots, and simply start counting time slots when joining the protocol. In fact, policy $\pi$ associated with a time-slot counter $t$ is equivalent to a policy $\text{shift}(\pi, \Delta) = \{((i + \Delta) \bmod 2^m, m) \mid (i, m) \in \pi\}$ associated with counter $t + \Delta$.

The algorithm can be in two states, *active* and *inactive,* depending on whether the nodes has packets that need sending or not. A node transmits at time $t$ if it is active and $t \in T(\pi)$. The node then receives the channel state $c\{E, C, S\}$, and uses it to update $\pi$, $p_b$, and $p_q$:

- If $c = C$, the schedule in $\pi$ that caused the collision is either eliminated or replaced by one of its two children in the tree, chosen at random, reducing its bandwidth by at least half. This is implemented in the *demote* procedure.
- If $c = S$, with probability $p_k$ (the kindness probability) the node demotes the policy responsible for a transmission, to allow for the rotation of slot use among nodes.
- If $c = E$, with probability $p_b$ the node adds a schedule to the policy $\pi$ to make use of the free time slot. This is implemented in the *barge-in* procedure.

After this update, the policy $\pi$ is pruned and brought back into normal form via the *normalize* procedure, also described in detail later.

**Constants:**
$\alpha_k = 0.98$: kindness inertia;
$\alpha_b = 0.99$: barge-in inertia;
$q_k = 10^{-2}$: kindness probability lower bound;
$q_b = 10^{-3}$: barge-in probability lower bound;
$\kappa = 0.05$: target fraction of empty slots;
$e$: base of natural logarithm;
$M = 10$: maximum number of schedules in a policy;
$\Delta = 4$: maximum schedule level difference;
$\Delta_{new} = 2$: schedule insertion delta;
**State Variables:**
*active*: True if the node is active; false otherwise;
$t$: time slot counter;
$\pi$: AT policy;
$p_b, p_k$: burst-in and kindness probabilities;
**Channel Variables:**
$T$ : transmit; $W$ : wait;
$d \in \{T, W\}$: decision;
$S$ : successful time slot;
$E$ : empty time slot;
$C$ : time slot with collisions;
$c \in \{S, E, C\}$: channel state;
**Initialization:**
$t := 0$; $p_b := 0.1$; $p_k := 0.05$;
$\pi := choice\{(0, 1), (1, 1)\}$;
**At every time slot:**
$t := t + 1$;
// Decision, and outcome
**if** $t \in T(\pi)$ *and active* **then** $d := T$ **else** $d := W$;
$h :=$ *channel outcome in* $\{E, C, S\}$;
// Policy update
**if** $d = T$ **then**
  **if** $h = S$ **then**
    **with probability** $p_k$**:**
      $\pi := demote(\pi, t)$
  **if** $h = C$ **then** $\pi := demote(\pi, t)$;
**if** $d = W$ *and* $h = E$ **then**
  **with probability** $p_b$**:**
    $\pi = bargein(\pi, t, \Delta_{new})$
$\pi := normalize(\pi, M, \Delta)$;
// Probability update
**if** $h = E$ **then** $p_k := p_k \cdot \alpha_k^{1/\kappa}$ **else** $p_k := p_k/\alpha_k$ ;
**if** $h = E$ **then** $p_b := p_b/\alpha_b$ ;
**if** $h = C$ **then** $p_b := p_b \cdot \alpha_b^{1/(e-2)}$;
$p_k := \min(0.5, \max(q_k, p_k))$; $p_b := \min(0.5, \max(q_b, p_b))$

**Algorithm 1:** AT Algorithm.

### 3.4 Policy Update

AT-ALOHA policies are updated via three operations: *demote, barge-in,* and *normalize.*

***demote***$(\pi, t)$ (Figure 2). The procedure *demote*$(\pi, t)$ removes from $\pi$ the (unique) schedule $(i, m)$ such that $t \in T(i, m)$. Further, if $\{(j, k) \in \pi \mid k \leq m\} = \emptyset$, then the procedure adds to $\pi$ one of the two children $(i, m+1)$ or $(i + 2^m, m+1)$ of the removed schedule, chosen uniformly at random.

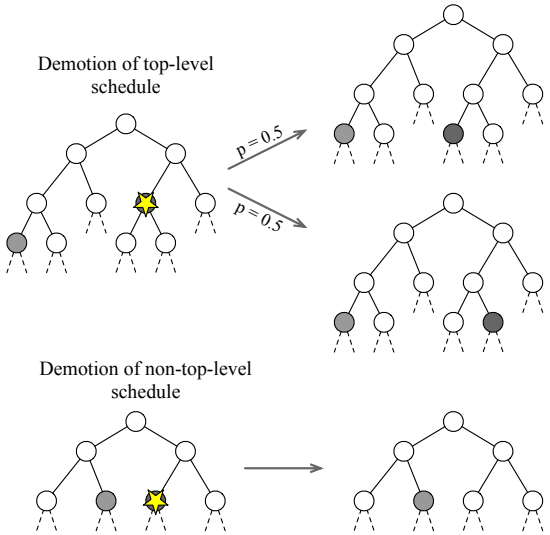


**Figure 2: Effect of demoting the starred schedule in the policy, according to whether the demoted schedule is at top level in the policy, or not.**

***bargein***$(\pi, t, \Delta_{new})$. The procedure *bargein*$(\pi, t, \Delta_{new})$ adds to $\pi$ a schedule $(i, m)$ such that $t \in T(i, m)$. Let $b = \sum_{(i,k)\in\pi} 2^{-k}$ be the bandwidth of the node's policy. We let

$$m = \left\lceil \log_2(1/p_b) + [\log_2(b/p_b)]_{-1}^{1} + \Delta_{new} \right\rceil , \qquad (1)$$

and $i = t \mod m$; the choice of $m$ is justified in the following.

***normalize***$(\pi, M, \Delta)$***:*** To normalize a policy $\pi$, the following steps are repeated until they can no longer be taken:

- *Descendant elimination:* If there are $(i, m), (j, k) \in \pi$ with $k > m$ and $j \mod 2^m = i$, remove $(j, k)$ from $\pi$.
- *Siblings merging:* If there are $(i, m), (j, m) \in \pi$ with $j = i + 2^{m-1}$, then replace both $(i, m)$ and $(j, m)$ in $\pi$ with $(i, m-1)$.

Once $\pi$ is in normal form, we prune it in two steps, first limiting the tree depth, then the number of selected schedule nodes in it. These pruning operations, illustrated in Figure 3, are performed as follows:

- Let $k = \min\{m \mid (i, m) \in \pi\}$ be the minimum level of a schedule in $\pi$. AT prunes all schedules of level below $k + \Delta$, letting $\pi := \{(i, m) \mid (i, m) \in \pi \wedge m \leq k + \Delta\}$.
- AT then prunes $\pi$ to ensure it contains at most $M$ schedules. If $|\pi| \leq M$, AT leaves $pi$ unchanged. Otherwise, let $n_k = |\{(i, m) \in \pi \mid m \leq k\}|$, and let $k$ be the largest integer such that $n_k \leq m$. Then, AT removes from $\pi$ all schedules $(i, m)$ with $m > k + 1$, and we randomly select $M - n_k$ of the schedules at level $k + 1$, that is, of the form $(j, k + 1)$ for some $j$.

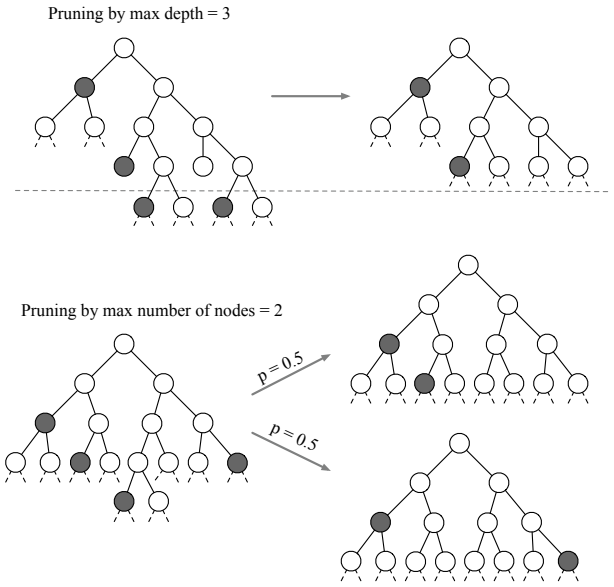


**Figure 3: Pruning a policy by limiting the depth (top), and limiting the number of nodes (bottom).**

### 3.5 Fairness, Kindness, and Barge-in

The kindness and barge-in probabilities $p_k$ and $p_b$, together, ensure that every active node receives a fair share of the total bandwidth. The kindness probability ensures that a node has a non-zero probability of relinquishing any transmission slots it holds. Nodes that transmit in more slots relinquish proportionately more bandwidth than nodes using fewer slots, and every freed slot has the same probability of being captured by any node. Together, this ensures that the bandwidth tends to be uniformly distributed among the nodes participating in the protocol. The probabilities $p_k$ and $p_b$ are tuned dynamically as follows.

*Tuning of Kindness Probability.* The kindness probability $p_k$ is tuned so that for each $n$ successful slots, we have $\kappa n$ of free slots; we choose $\kappa = 0.05$ or 5%. Initially we arbitrarily set $p_k = 0.05$. Thereupon, nodes update $p_k$ according to the channel outcomes $E$, $S$, $C$:

$$E: \ p_k := p_k \cdot \alpha_k^{1/\kappa} \qquad S, C: \ p_k := p_k / \alpha_k$$

where $\alpha_k = 0.98$ is a coefficient determining the adaptation speed. Thus, $\alpha_k$ increases at each success slot, and decreases at each empty slot. Equilibrium is reached when these updates balance, that is, when there are $\kappa$ empty slots for every successful one, so that $(\alpha_k^{1/\kappa})^\kappa = \alpha_k$.

*Tuning of Barge-in Probability.* The barge-in probability is tuned to optimize the probability that when a slot is empty, one node, and only one node, will add a schedule to use the empty slot in the future. The analysis follows the lines of the slotted ALOHA analysis in [22]. If there are $n$ active nodes and each of them barges-in

with probability $q$, then a time slot remains empty with probability $(1-q)^n$, it is used successfully with probability $nq(1-q)^{n-1}$, and collision happens with probability $1-(1-q)^n - nq(1-q)^{n-1}$. The probability of successful transmission is maximized for $q = 1/n$ when $n$ nodes are active. Under this optimal choice of $q$, as $n \to \infty$, the probability of the slot remaining free tends to $1/e$, and the collision probability tends to $2/e$, where $e$ is the basis of the natural logarithm. The optimal ratio of free to collision slots is then $(1/e)/(1-2/e) = 1/(e-2)$. Thus, the nodes tune $p_b$ so that the ratio of free to collision slots is $1/(e-2)$, updating $p_b$ according to channel outcomes:

$$E:\ p_b := p_b/\alpha_b \qquad C:\ p_b := p_b \cdot \alpha_b^{1/(e-2)}\ .$$

For the adaptation coefficient, we take $\alpha_b = 0.99$.

We have seen that the barge-in probability $p_b$ provides an estimate $\tilde{n} = 1/p_b$ of the number of active nodes in the protocol. After experimentation, we choose to insert new schedules at level $m = \log_2(\tilde{n}) + \Delta_{new}$ with $\Delta_{new} = 2$.

# 4 PERFORMANCE EVALUATION

We compare AT-ALOHA with framed slotted ALOHA and ALOHA-Q via simulations of a fully-connected single-channel time-slotted wireless network. The three protocols are compared in terms of their bandwidth utilization and fairness.

*Network utilization.* The network utilization is the fraction of successful transmission slots; we measure this fraction by aggregating time slots in *blocks* of 100. Similarly, we measure the fraction of empty and collision time slots in each block. Using 100-slots blocks offers a compromise between measuring with fine time resolution, and computing meaningful statistics on each block.

*Fairness.* We measure the fairness of the protocols via *Jain's index* [11, 12]. For a block in which $n$ nodes are active, let $b_i$ be the number of successful transmission by node $i$, for $i \le i \le n$. Jain's index is computed as

$$J = \left(\sum_{i=1}^n b_i\right)^2 / \left(n \sum_{i=1}^n b_i^2\right)\ .$$

We have $1/n \le J \le 1$; $J = 1$ for a perfectly fair distribution of the channel, and $J = 1/n$ if only one node gets to use the channel.

## 4.1 Comparison Protocols

We compare the performance of AT-ALOHA with that of two versions of exponential-backoff ALOHA, and with the performance of the ALOHA-Q protocol proposed by Chu et al. [6, 7].

*EB-ALOHA and EB-ALL-ALOHA.* EB-ALOHA is the standard slotted ALOHA with exponential-backoff. In EB-ALOHA, every node, when becoming active, has an initial transmission probability $p = 1/2$. Whenever the node transmits, it updates the transmission probability, setting $p := \alpha p$ in case of collision, and $p := \min(1, p/\alpha)$ in case of success, where $\alpha$ is a constant that determines adaptation speed; in our simulations we use $\alpha = 0.9$. The EB-ALL-ALOHA protocol is similar to EB-ALOHA, except that nodes update their transmission probabilities following *all* successful transmissions or collisions, rather than only those in which they took part.

*ALOHA-Q.* ALOHA-Q is the Q-learning version of ALOHA proposed in [6, 7]. The ALOHA-Q is based on a periodic frame of fixed length $n$. Each node stores $q$-*values* $q_1, q_2, \ldots, q_n$, where $q_i$ represents the quality of the decision of transmitting in the $i$-th slot of the frame. At every frame, the protocol transmits in a slot $i$ with maximal $q_i$; if the transmission is successful, it increases $q_i$; if a collision occurs, it decreases $q_i$ and it follows a randomized backoff before retrying. The bandwidth utilization of ALOHA-Q increases with the number $m$ of active nodes, approaching $m/n$, as long as $m \le n$; when $m \gg n$, the protocol behaves in a similar fashion to EB-ALOHA. In our simulations, we consider frames of $n = 64$ time slots; as our maximum value for $m$ is 50, this ensures that the protocol works close to optimality.

## 4.2 Results

Figure 4 compares the performance of AT-ALOHA, EB-ALOHA, EB-ALL-ALOHA, and ALOHA-Q when the number of active nodes is initially 10, then ramps up to 50, and finally ramps down to 30.

For AT-ALOHA, the throughput remains in the 85% to 90% range in the steady-state periods when nodes neither join nor leave; during the transients, the utilization is still above 75% when ramping up, and above 60% when ramping down. The Jain fairness index of AT-ALOHA is also close to 1.

The only protocol that is competitive with AT-ALOHA in terms of utilization is EB-ALOHA. The problem is that EB-ALOHA achieves its high network utilization via an extremely unfair allocation of bandwidth, leading to a Jain index close to 0. In EB-ALOHA, nodes that are successful in transmitting will increase their transmission probability, while nodes whose transmissions are unsuccessful due to collisions will reduce their transmission probability. This amplifies any initial random difference in transmission success, leading to a winner-takes-all situation in which one node uses most of the bandwidth, transmitting with very high probability, while other nodes are mostly silent.

The EB-ALL-ALOHA protocol manages to achieve the optimal network utilization of $1/e \approx 0.37$ that is the maximum attainable under symmetrical transmission probability (and thus fairness) for ALOHA. Its fairness is uniformly very high, since all nodes transmit with the same probability.

Finally, the bandwidth utilization of ALOHA-Q is dependent on the number of active nodes, increasing as the number of active nodes approaches the frame length of 64. Even when the number of active nodes is 50, as around time block 150 of Figure 4, the utilization is below 0.6. This is well below the theoretical maximum of $50/64 \approx 0.78$, likely because the active nodes have not had time to adapt to the network conditions. ALOHA-Q also allocates bandwidth fairly, as each node can transmit at most once per frame.

# 5 CONCLUSIONS

We introduced a new collaborative learning algorithm, the Adaptive Tree (AT) algorithm, to enable nodes sharing a common channel to quickly approach collision-free transmissions while maintaining fairness. In contrast to prior approaches that use machine learning to improve the performance of slotted ALOHA, the resulting protocol, AT-ALOHA, only requires nodes to agree on the beginning of time slots, and does not require the definition of transmission

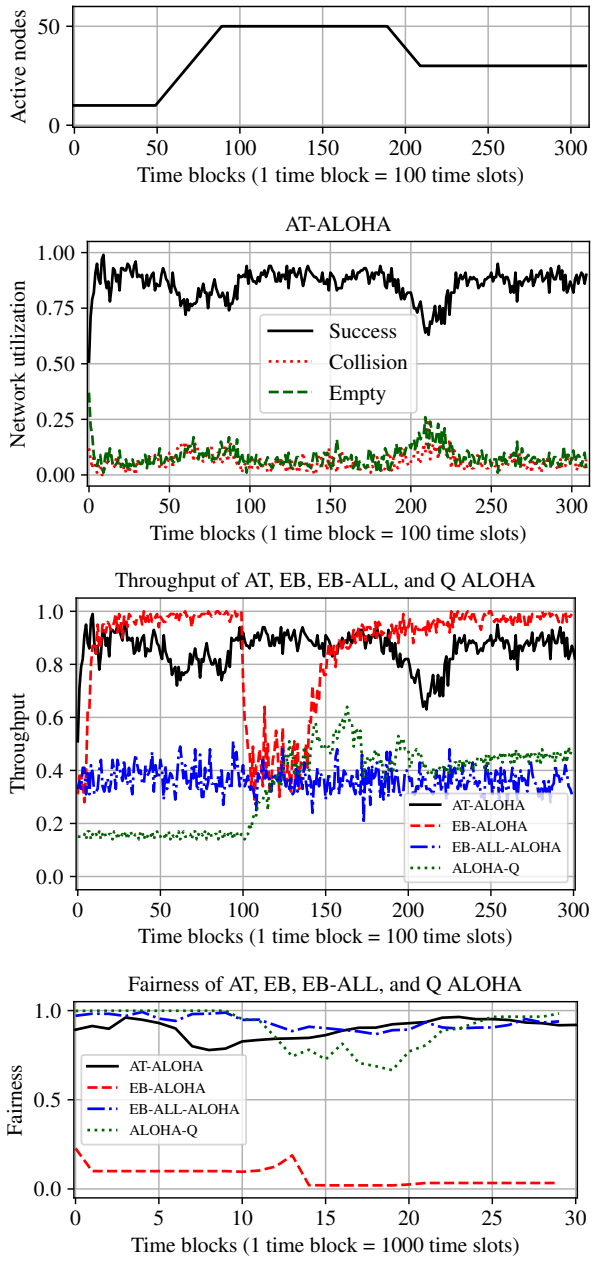


**Figure 4: Protocol comparison when the number of active nodes is fist 10, then ramps up to 50, and finally ramps down to 30.**

frames with a fixed number of time slots per frame or the numbering of time slots. Simulation experiments were used to illustrate that AT-ALOHA attains better throughput and fairness than slotted ALOHA with exponential backoffs and ALOHA-Q, which is framed slotted ALOHA with Q learning.

## REFERENCES


[1] Abramson, N. The throughput of packet broadcasting channels. *IEEE Transactions on Communications 25*, 1 (1977), 117–128.

[2] Bao, L., and Garcia-Luna-Aceves, J. A new approach to channel access scheduling for ad hoc networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking* (2001), pp. 210–221.

[3] Bao, L., and Garcia-Luna-Aceves, J. Hybrid channel access scheduling in ad hoc networks. In *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* (2002), IEEE, pp. 46–57.

[4] Capetanakis, J. Tree algorithms for packet broadcast channels. *IEEE transactions on information theory 25*, 5 (1979), 505–515.

[5] Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., and Warmuth, M. K. How to use expert advice. *Journal of the ACM (JACM) 44*, 3 (1997), 427–485.

[6] Chu, Y., Kosunalp, S., Mitchell, P. D., Grace, D., and Clarke, T. Application of reinforcement learning to medium access control for wireless sensor networks. *Engineering Applications of Artificial Intelligence 46* (2015), 23–32.

[7] Chu, Y., Mitchell, P. D., and Grace, D. ALOHA and q-learning based medium access control for wireless sensor networks. In *2012 International Symposium on Wireless Communication Systems (ISWCS)* (2012), IEEE, pp. 511–515.

[8] Garcia-Luna-Aceves, J., and Masilamani, A. N. Nomad: Deterministic collision-free channel access with channel reuse in wireless networks. In *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks* (2011), IEEE, pp. 1–9.

[9] Garcia-Luna-Aceves, J., and Masilamani, A. N. Using radio connectivity to define transmission schedules in multihop wireless networks. In *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems* (2014), IEEE, pp. 434–442.

[10] Herbster, M., and Warmuth, M. K. Tracking the best expert. *Machine learning 32*, 2 (1998), 151–178.

[11] Huaizhou Shi, Prasad, R. V., Onur, E., and Niemegeers, I. G. M. M. Fairness in Wireless Networks:Issues, Measures and Challenges. *IEEE Communications Surveys & Tutorials 16*, 1 (2014), 5–24.

[12] Jain, R. K., Chiu, D.-M. W., and Hawe, W. R. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA* (1984).

[13] Jakllari, G., Neufeld, M., and Ramanathan, R. A framework for frameless tdma using slot chains. In *2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)* (2012), IEEE, pp. 56–64.

[14] Khaleghi, E. E., Adjih, C., Alloum, A., and Mühlethaler, P. Near-far effect on coded slotted aloha. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* (2017), IEEE, pp. 1–7.

[15] Liva, G. Graph-based analysis and optimization of contention resolution diversity slotted aloha. *IEEE Transactions on Communications 59*, 2 (2010), 477–487.

[16] Masilamani, A. N., and Garcia-Luna-Aceves, J. Scheduled channel access using geographical classification. In *2013 International Conference on Computing, Networking and Communications (ICNC)* (2013), IEEE, pp. 790–796.

[17] Okada, H., Igarashi, Y., and Nakanishi, Y. Analysis and application of framed aloha channel in satellite packet switching networks-fadra method. *Electronics Communications of Japan 60* (1977), 72–80.

[18] Paolini, E., Liva, G., and Chiani, M. Coded slotted aloha: A graph-based method for uncoordinated multiple access. *IEEE Transactions on Information Theory 61*, 12 (2015), 6815–6832.

[19] Ramanathan, S., and Lloyd, E. L. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transactions on networking 1*, 2 (1993), 166–177.

[20] Roberts, L. G. ALOHA packet system with and without slots and capture. *ACM SIGCOMM Computer Communication Review 5*, 2 (1975), 28–42.

[21] Schoute, F. Dynamic frame length aloha. *IEEE Transactions on communications 31*, 4 (1983), 565–568.

[22] Tanenbaum, A. S., and Wetherall, D. *Computer networks*. Prentice hall, 1996.

[23] Tang, Z., and Garcia-Luna-Aceves, J. A protocol for topology-dependent transmission scheduling in wireless networks. In *WCNC. 1999 IEEE Wireless Communications and Networking Conference (Cat. No. 99TH8466)* (1999), vol. 3, IEEE, pp. 1333–1337.

[24] Vergados, D. J., Amelina, N., Jiang, Y., Kralevska, K., and Granichin, O. Local voting: Optimal distributed node scheduling algorithm for multihop wireless networks. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2017), IEEE, pp. 1014–1015.

[25] Yang, Z., and Garcia-Luna-Aceves, J. Hop-reservation multiple access (hrma) for ad-hoc networks. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)* (1999), vol. 1, IEEE, pp. 194–201.

[26] Yu, Y., Wang, T., and Liew, S. C. Deep-reinforcement learning multiple access for heterogeneous wireless networks. *IEEE Journal on Selected Areas in Communications* (2019).