Title

Approaching Fair Collision-Free Channel Access with Slotted ALOHA Using Collaborative Policy-Based Reinforcement Learning

Authors

de Alfaro, Luca
Zhang, Molly
Garcia-Luna-Aceves, J.J.

Publication Date

2020

Peer reviewed

Approaching Fair Collision-Free Channel Access with Slotted ALOHA Using Collaborative Policy-Based Reinforcement Learning

Luca de Alfaro
Computer Science and Engineering
University of California
Santa Cruz, CA
luca@ucsc.edu

Molly Zhang
Computer Science and Engineering
University of California
Santa Cruz, CA
mollyzhang@ucsc.edu

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

***Abstract*—A novel expert-based approach to reinforcement learning is applied to slotted ALOHA in order to approach fair collision-free transmissions. Active nodes use known periodic transmission schedules as policies and assign weights to them based on the Quantitative Tree (QT) algorithm introduced in this paper. Nodes learn to transmit following the policies with the highest weights to minimize packet collisions. This results in two variants of slotted ALOHA, which we call ALOHA-QT and ALOHA-QTF, that converge to transmission schedules that are almost free of collisions within a short period of time, and that attain near perfect transmission throughput even when node additions and departures occur frequently. In addition, ALOHA-QTF attains very fair bandwidth distribution among nodes, where the nodes with bottom ten percent of bandwidth still fare reasonably well. ALOHA-QTF is shown to be better than slotted ALOHA with exponential back off and framed slotted ALOHA with Q learning (ALOHA-Q) in terms of throughput and fairness.**

I. Introduction

Simplicity is the most salient feature of the ALOHA protocol [1]; a node with a packet to send simply transmits. This simplicity makes ALOHA and its variants an attractive choice for channel access in such untethered networks as underwater acoustic networks, satellite networks, space networks, wireless networks in which hidden-terminal interference is prevalent, and IoT deployments consisting of very simple nodes. However, the simplicity of ALOHA comes at the price of poor performance, with a maximum throughput of only 18% of the available bandwidth. As a result, several variants of ALOHA have evolved over the years to allow more efficient sharing of common channels in untethered networks.

Section II summarizes relevant prior work. The first major improvement on ALOHA was slotted ALOHA [19], which doubles the maximum throughput attainable with pure ALOHA by forcing noes to transmit at the beginning of time slots defined at the physical layer. As the review of related work presented in Section II describes, many approaches have been proposed over the years to improve the performance of ALOHA have been based on framed slotted ALOHA [17], which organizes the channel into transmission frames consisting of a fixed number of time slots. A major limitation with these approaches is that they require network nodes to agree on the number of slots per transmission frame a priori. This is a problem because the optimum length of a transmission frame is a parameter whose value depends on the number of nodes, their connectivity, and traffic patterns.

Some recent proposals based on framed slotted ALOHA have adopted the *expert-based* approach to reinforcement learning (RL), also known as *regret learning* [7], [8], [2]. The goal is for nodes to learn which time slots to use in a way that reduces the likelihood of packet collisions and hence attains much higher throughput, and without need for any centralized coordination, pre-agreement, or out-of-band communication. The ALOHA-Q protocol [6], [5] is arguably the first example of this approach. Each node in ALOHA-Q learns in which time slot out of the M time slots of a fixed-length time-frame to transmit by tracking the success of M policies (i.e., "experts"), with each policy consisting of transmitting in a given time slot. Another approach [23] consists of using deep neural networks for deep reinforcement learning (DRL) in the context of slotted ALOHA.

To achieve high utilization, the nodes must learn how to coordinate. In ALOHA-Q, the coordination is guided by a fixed transmission frame, limiting the achievable utilization under variable network conditions. Furthermore, nodes take a long time to learn which slot in a large frame to use. In the DRL-based approach, the nodes can learn very general coordination schemes using deep neural networks, but the learning can take very long, even when only a couple of network nodes are involved. Furthermore, the DRL approach incurs a considerable computational cost at each node for training the neural networks.

This paper introduces a new approach to the use of RL in the context of slotted ALOHA. The key insight in our proposed RL approach consists in organizing the set of known policies into a policy tree, with each policy $(i, 2^m)$ consisting

ISBN 978-3-903176-28-7

of transmitting at every time t with $t \mod 2^m = i$. Nodes learn which policies render higher throughput by assigning quantitative weights to all policies. The weight tracks the policy success in predicting available network time-slots, and the nodes schedule their transmissions according to the best policies. This results in fast convergence to transmission schedules that are fair and almost free of collisions, without the need for predefined transmission frames, sophisticated physical-layer mechanisms, or deep neural networks. As in the original slotted ALOHA protocol, the only assumptions made are that nodes are synchronized so that transmissions start at the beginning of time slots, and a node can detect whether or not its transmission succeeds before the time slot ends.

We call the proposed approach *ALOHA-QT*, where QT stands for *quantitative tree*, referring to the way in which the policies are organized into a tree, according to the policies' period, with quantitative weights tracking the success of each policy. Sections III and IV describe this approach, and Section V introduces ALOHA-QTF, a more sophisticated variant of the same approach that exhibits improved fairness.

Section VI addresses the performance of the proposed approach, and compares ALOHA-QT and ALOHA-QTF with traditional slotted ALOHA with exponential backoff and ALOHA-Q. The results show that nodes using ALOHA-QT and ALOHA-QTF learn to coordinate and achieve utilization above 75% in a few hundred time-slots, which is the time in which each node transmits only 10 to 20 times, even when 50 nodes join the network simultaneously; furthermore, the utilization ultimately settles to about 90%. We show that this speed of convergence is markedly superior to what can be achieved with either ALOHA-Q or the DRL approach. The allocation of bandwidth is markedly fair, with the 10% of nodes in the lowest bandwidth percentile each receiving bandwidth that is 75% of the average one. We present results showing the protocol robustness and speed of adaptation in settings where the number of active nodes varies, and in settings where nodes continuously join and leave the network. In all these settings, ALOHA-QTF reaches very high utilization, always above 75% except in short transients, with fair bandwidth sharing. This high and fair utilization is reached without any centralized coordination, packet content modification, or out-of-band communication to reach agreement.

II. Related Work

Most of the approaches that have been proposed over the years to improve the performance of ALOHA and slotted ALOHA have been based on framed slotted ALOHA. A number of schemes 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 [15], [12], [18], [20].

A number of recent attempts have been made to improve the performance of slotted ALOHA and framed slotted ALOHA by using machine learning in the coordination of time-slot selection among nodes with packets to send.

Two diametrically opposed approaches to learning coordination have been proposed so far. In ALOHA-Q, [6], [5], nodes used reinforcement learning to choose in which slot of a fixed-length transmission frame to transmit. In [23], nodes use general deep-reinforcement learning to learn how to coordinate, without reference to any underlying frame or set of alternative policies.

The ALOHA-Q protocol (framed slotted ALOHA with Q-learning) was proposed by Chu et al. in [6], [5] for coordinating transmissions in framed slotted ALOHA. The protocol assumes a fixed frame length M. Each node has the set of policies $\{(i, M) \mid 0 \leq i < M\}$, where policy (i, M) prescribes transmitting in the i-th time-slot of the frame, that is, at all time slots t where $t \mod M = i$. Whenever a node transmits a packet in time slot t, it updates the weight of policy $(t \mod M, M)$, increasing it if the transmission succeeds, and decreasing it otherwise. At each frame, each node transmits according to the policy of highest weight, with a back-off procedure if a collision occurs.

ALOHA-Q is closely based on the "expert-based" approach to reinforcement learning [7], [8], [2], in which an agent has several experts at its disposal. At each stage, the agent chooses dynamically the best expert to follow, according to the past quality of their advice. In ALOHA-Q, the agents are the nodes, and the experts are the periodic policies. The nodes are thus constrained to transmitting according to frame-based periodic policies: this speeds up the learning step, at the expense of flexibility. The nodes learn to coordinate in a few tens of frames, but the fixed frame length limits the network utilization that can be obtained under variable network traffic.

A reinforcement-learning approach that allows nodes to transmit according to any schedule has been presented in [23]. The nodes use deep-reinforcement learning to learn when it is best to transmit. The approach is based on Q-learning, where the value of actions (transmit, or wait) is learned as a function of the state of the system (the recent network history) [22]. The work relies on *deep* reinforcement learning, where the values are computed with the help of a trainable neural network [14]. In particular, the approach relies on LSTMs [9], a type of recurrent neural network whose output depends on a sequence of inputs. In this way, the value of an action can depend on the recent network history, and nodes can potentially learn arbitrary transmission schedules. The drawback of this generality is that learning, unguided by fixed policies, takes long time: the approach has been demonstrated only for networks of two adaptive nodes, and even for such small networks, it takes tens of thousands of time-slots to converge.

Deep reinforcement learning (DRL) has been used in other approaches to channel access in networks. In [16], DRL is used to choose which of N orthogonal channels to access using a MAC protocol, and in [21], it is used to choose a frequency channel in presence of interference. In [4], DRL is used for channel selection and access in LTE-U networks.

The policy trees introduced in this paper are related to the binary trees proposed by Capetanakis for conflict resolution

following collisions [3]. However, there is a deep difference on how such trees are used in the two approaches. In Capetanakis' approach, the trees guide the resolution of each conflict as it arises. By contrast, policy trees are used in ALOHA-QT and ALOHA-QTF to guide both the resolution of the conflicts, and the periodic transmissions of the nodes, so that once the nodes learn how to resolve a conflict, they continue to transmit according to schedules that avoid conflicts.

III. Preliminaries

We consider a fully-connected network in which the channel is time slotted. At each time slot a node can either transmit (T) or wait (W), and the channel can be in one of three states: empty (E), if no node transmits; success (S), if exactly one node transmits; and collision (C), if two or more nodes transmit. We assume that the nodes can detect the state of the time slot, and thus, the outcome of their transmissions. This assumption can be brought to practice in several ways. A central node (such as a satellite transponder or a base station) can use a downlink to repeat the transmissions sent to it in each slot over an uplink, as in the original slotted ALOHA protocol. Alternatively, each time slot can be divided into a portion for packet transmission and a portion for an acknowledgement.

It is well known that the throughput of slotted ALOHA in a fully-connected network tends to $1/e \approx 0.37$ as the number of nodes grows. Achieving higher throughput while giving each node a fair share of the bandwidth requires coordinating the node's transmission schedules. ALOHA-QT and ALOHA-QTF attempt to do this by allowing node to *learn to coordinate* via reinforcement learning, without need for any centralized coordination, pre-agreement, or out-of-band communication.

A. From ALOHA-Q to ALOHA-QT and ALOHA-QTF

The starting point in our design is ALOHA-Q [6], [5], a protocol based on a fixed-length frame, in which each node learns in which slot of the frame the transmissions can be most successful. ALOHA-Q is based on a transmission frame of fixed length M. An ALOHA-Q node keeps a time-slot counter t, and relies on M policies $(0, M)$, $(1, M)$, …, $(M-1, M)$; a policy (i, M) prescribes sending at all times t such that $t \mod M = i$. The policies play the role of the experts in reinforcement learning: each node tracks the success of each policy, and so eventually settles into transmitting always in the same time-slot of each frame.

ALOHA-Q has two limitations. One is its reliance on a fixed-length frame. When the number of nodes N is smaller than the frame length M, the network utilization can approach N/M in the long term. However, when $N \ll M$ the utilization can be very low, and when $N > M$, the utilization is quickly degraded, as there are not enough slots for each node in the fixed-length period. The second limitation of ALOHA-Q is the adoption of a non-standard version of weight update for the policies, which slows down adaptation.

Our approach improves on ALOHA-Q in four main ways. The first, and most important, is to abandon the use of a fixed-length frame, and adopt instead a *policy tree*, where policies of different periods are arranged in a tree. The tree guides conflict resolution and helps the nodes settle on non-conflicting policies. The availability of policies of different periods enables the nodes to vary their transmission rate and adapt to the number of active nodes: there is no longer a fixed frame that can remain mostly empty, or be of insufficient length to accommodate all nodes. The second, related, improvement is to allow nodes to follow the "advice" of more than one expert simultaneously; this lets nodes mix policies of different transmission bandwidth, enabling the nodes to fine-tune their overall transmissions. The third improvement consists in a new method for updating the policy weights following network outcomes. Our method is closer to the standard methods in reinforcement learning, and enables a faster convergence to an efficient schedule. The last improvement consists in tuning the weight updates and policy selection to ensure a fair distribution of the network bandwidth to the participating nodes.

B. The Policy Tree

Each node keeps a time-slot counter t. A (periodic) *policy* consists in a pair (i, m) with $0 \le i < m$; the policy (i, m) prescribes transmitting at all times t such that $t \mod m = i$. For instance, the policy $(3, 8)$ prescribes transmitting at times $t = 3, 11, 19, 27, \ldots$. Importantly, our design does not require the nodes to synchronize their time-slot counters: a policy (i, m) for a node with counter t is equivalent to a policy $((i + k) \mod m, m)$ for a node with counter $t' = t + k$.

The reinforcement learning at each node is based on the set of policies $\mathcal{P} = \{(i, 2^m) \mid 0 \le i < 2^m, 0 \le m \le n\}$. The maximum periodicity 2^n is chosen so that it is larger than the maximum number of nodes that can be present on the network. Since nodes can transmit with periods that are smaller than 2^n (for example, nodes can transmit every fourth time-slot using a policy of period 2^2), there is no performance penalty in ALOHA-QT in choosing a value of n that is larger than necessary.

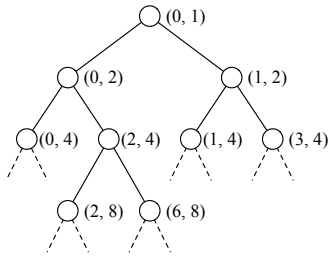

Fig. 1. Policy tree in ALOHA-QT and ALOHA-QTF

In principle, reinforcement learning could be applied to arbitrary sets of policies. However, constraining the periods to be powers of 2 facilitates the coordination between the nodes. To illustrate this, it is useful to depict the policies of ALOHA-QT as arranged into a binary *policy tree*, where the root policy is the policy that transmits at all times, and where the two children of a policy transmit each in half of the time slots of the parent, as depicted in Figure 1. The policy

$(0,2)$ in the tree prescribes transmitting at even time slots, and has as children the two policies $(0,4)$ and $(2,4)$ which both prescribe transmission once every four time slots; the union of the transmission schedule of these two children policies is exactly the set of transmission schedule of the parent $(0,2)$. In this tree, two policies prescribe conflicting transmissions only if one policy is an ancestor of the another. Thus, as long as nodes settle on policies that are not the ancestor of the other, the nodes can transmit on the network avoiding conflicts.

If we allowed policies with arbitrary periods, rather than policies in the tree with periods that are powers of 2, conflicts would be common: for instance, if two policies (i,m), (i',m') have mutually prime periods m, m', then there would be a collision every mm' time-slots.

IV. ALOHA-QT

Algorithm 1 describes how each node selects time slots for transmission in ALOHA-QT. Each node keeps a time-slot counter t, and it stores the weight $w_\sigma \in [0,1]$ of each policy σ in the policy tree. At each time slot, each node performs the following actions:

- *Policy selection.* The node selects a subset $\mathcal{A}_t$ of *active* policies to follow in the time slot t, based on the weights of the policies.
- *Decision.* If $t \mod m = i$ for some active policy $(i,m) \in \mathcal{A}_t$, and the node is active (it has some packet to send), the node transmits; otherwise, it waits.
- *Policy weight update.* Based on the resulting channel state (Successful, Empty, Collision) of the time slot, the node updates the weights of all policies.
- *Policy weight normalization.* Once the weights of the policies have been individually updated, the values for all policies are normalized, redistributing some of the "lost weight" randomly across all policies.

A. Policy-Weight Initialization

Let n be the depth of the policy tree. For $k = 1, 2, \ldots, n$ and $0 \le i < 2^k$, we initialize the weight of policy $(i, 2^k) \in \mathcal{P}$ by:

$$w_{(i,2^k)} = 0.2 \cdot \frac{0.9 + 0.1 \cdot Z_{(i,2^k)}}{1.2^k},$$

where $\{Z_\sigma\}_{\sigma \in \mathcal{P}}$ is a set of random variables independently sampled from the uniform distribution over $[0,1]$. Thus, policies have an initial weight of approximately 0.2, with a small amount of noise added to break ties between policies and to ensure that the initial behavior of nodes is not synchronized. The denominator 1.2^k causes policies with shorter periods to have higher probability of being initially active. In this fashion, nodes are initially likely to adopt policies that transmit frequently, falling back on policies that transmit more rarely only as needed to avoid collisions.

B. Policy Selection

We denote by $\mathcal{A}_t$ the set of policies that have been selected as active at time t. A policy $\sigma \in \mathcal{P}$ is selected as active in a round if:

Constants:
- $n = 8$: depth of policy tree;
- $w_{init} = 0.25$: weight initialization factor;
- $\alpha^+ = 0.2$: multiplicative increment factor;
- $\alpha^- = -0.5$: multiplicative decrement factor;
- $\gamma_0 = 0.1$: policy initialization noise;
- $\gamma_1 = 1.2$: initial bias for high-bandwidth policies;
- $\eta = 0.95$: weight selection threshold;
- $\epsilon_r = 0.02$: probability of relinquishing a time-slot;

State Variables:
- $\mathcal{P} = \{(i,m) \mid 0 \le i < m, m = 2^k, 0 \le k \le n\}$: policies;
- $\{w_\sigma\}_{\sigma \in \mathcal{P}}$: policy weights;
- *active*: True if the node is active; false otherwise;
- $t \in \mathbb{N}$: time slot counter;

Channel Variables:
- $d \in \{T, W\}$: decision (T : transmit; W : wait);
- S : successful time slot;
- E : empty time slot;
- C : time slot with collisions;
- $c \in \{S, E, C\}$: channel state (S : successful transmission, E : empty time slot, C : collisions);
- X: each occurrence of X is independently sampled from the uniform distribution over $[0,1]$;

Initialization:

$t := 0$;
for $0 \le k \le n$, $0 \le i < 2^k$ **do**
  $w_{(i,2^k)} = w_{init} \cdot \gamma_1^{-k} \cdot (1 - \gamma_0 + \gamma_0 \cdot X_{ik})$;

At every time slot:

// Policy selection
$\mathcal{E}_t = \{(i,m) \in \mathcal{P} \mid i \mod m = t\}$;
$\mathcal{A}_t = \text{argmax}_{\sigma \in \mathcal{P}} w_\sigma \cup \{\sigma \in \mathcal{P} \mid w_\sigma > \eta\}$;
// Decision
if $\mathcal{E}_t \cap \mathcal{A}_t \neq \emptyset$ *and active* **then** $d := T$ **else** $d := W$;
$h :=$ *channel outcome in* $\{E, C, S\}$;
// Policy weight update
if $(d,h) \in \{(W,E), (T,S)\}$ **then** $\alpha := \alpha^+$ **else** $\alpha := \alpha^-$;
for $\sigma \in \mathcal{P}$ **do**
  if $\sigma \in \mathcal{E}_t$ **then** $w'_\sigma := w_\sigma \cdot e^{\alpha X_\sigma}$ **else** $w'_\sigma = w_\sigma$;
// Voluntary slot relinquishment
if $\text{Rand}[0,1] < \epsilon_r$ **then**
  for $\sigma \in \mathcal{E}_t$ **do** $q'_\sigma = 0$;
// Policy weight normalization
$W := \sum_{\sigma \in \mathcal{P}} w_\sigma$, $W' := \sum_{\sigma \in \mathcal{P}} w'_\sigma$, $\Delta = W - W'$;
if $\Delta > 0$ *and* $W' < W_{init} \cdot |\mathcal{P}|$ **then**
  for $\sigma \in \mathcal{P}$ **do** $X_\sigma := \text{Rand}[0,1]$;
  for $\sigma \in \mathcal{P}$ **do** $w_\sigma := w'_\sigma + \Delta * (X_\sigma / \sum_\sigma X_\sigma)$;
else
  for $\sigma \in \mathcal{P}$ **do** $w_\sigma := w'_\sigma$;
// Bound enforcement
for $\sigma \in \mathcal{P}$ **do** $w_\sigma := \min(1, w_\sigma)$;
// Increment time
$t := t + 1$;

Algorithm 1: ALOHA-QT Algorithm. $\text{Rand}[0,1]$ is a random number generator that returns independent samples from the uniform distribution over $[0,1]$.

1) either σ is the policy with the maximal weight w_σ among all policies in a node;
2) or $w_\sigma \geq w_h$, where w_h is a pre-determined weight threshold; we use $w_h = 0.95$ in our implementation.

The first criterion ensures that a node always follows its best policy: this guarantees that every node will transmit at least once every 2^n time-slots. The second criterion allows a node to follow any additional policy that has been successful in predicting available slots. The ability of nodes to follow more than one policy is instrumental in enabling nodes to utilize a flexible amount of network bandwidth.

We experimented with alternative selection schemes to the one mentioned here; for instance, we experimented with selecting the highest-weight policy σ, along with any other policy σ' such that $w_{\sigma'} > \alpha w_\sigma$, for a weight fraction $\alpha < 1$. These schemes did not work as well as the one we presented above.

C. Decision

We say that a policy (i, m) is *enabled* in a time slot t if $t \mod m = i$: thus, the policies enabled at a time slots are those that prescribe transmitting in time slot. We let $\mathcal{E}_t = \{(i, m) \in \mathcal{P} \mid t \mod m = i\}$ be the set of enabled policies at time t. A node transmits (takes decision T) if $\mathcal{A}_t \cap \mathcal{E}_t \neq \emptyset$, that is, if one of the active policies at t is enabled; otherwise, it waits (decision W). At the end of the time slot, the node receives the network state for the slot, which can be E (empty slot), S (successful transmission), or C (collision). The pair (d, h), consisting of the decision $d \in \{T, W\}$ and the network state $h \in \{E, S, C\}$, is called the *outcome* of the slot for the node.

D. Policy Weight Update

At the end of a time slot, we apply a multiplicative update to the policies that are enabled in the time slot, increasing their weight if transmitting does not lead to collisions, and decreasing if it does. The multiplicative update to the weights w_σ of all enabled policies $\sigma \in \mathcal{E}_t$ takes the following form:

$$w'_\sigma = w_\sigma \cdot e^{X_\sigma \alpha} , \tag{1}$$

where:

- $\{X_\sigma\}$ is a set of random variables independently sampled from the uniform distribution $[0, 1]$;
- α is an update constant that depends on the slot outcome; we use:
  - $\alpha = 0.2$ for $(d, h) \in \{(W, E), (T, S)\}$;
  - $\alpha = -0.5$ for $(d, h) \in \{(W, S), (W, C), (T, C)\}$;
- w'_σ is the policy weight after the update.

Due to X_σ, the multiplicative update factors are randomized. This is different from the common case for expert-based reinforcement learning [22], [8]. Randomization helps break the ties between nodes that lay claims on the same transmission slots. To understand this, consider the case of nodes transmitting in the same time slot, leading to a collision. If a deterministic update was used, the nodes would update the weights of the policies responsible for the conflict in a synchronized manner, multiplying them by the same factor smaller than one. Eventually, the nodes involved in the collision would stop using the policies and cease transmitting in the slot. As the slot became empty, the nodes would reverse course, and increase the weights of the conflicting policies, likely reintroducing the conflict. This oscillatory behavior, from collisions to empty slots and back again, would slow down convergence to collision-free transmissions. Randomized updates break the symmetry and facilitate the emergence of a "winning" node that claims a slot; once a slot is claimed, the weight update mechanism reinforces the exclusive use of the slot by the winning node.

E. Policy-Weight Normalization

After the multiplicative update of the policy weights, we perform a three-step normalization of the weights.

a) Relinquishing the slot: First, with a small, constant probability ϵ_r, the weight of every policy in $\mathcal{E}_t$ is set to 0, forcing the node to relinquish transmission at a time slot, if it were holding it. This ensures that no node can hold a slot forever, ensuring some amount of fairness in the bandwidth allocation to the nodes.

b) Redistributing lost weights: In expert-based reinforcement learning, some of the weights lost by the policies that are downgraded is redistributed across all policies. In this way, if policies once successful become unsuccessful, the nodes will explore alternative policies [7], [8]. To this end, let w_σ, w'_σ be the weights of policy σ before and after the multiplicative update step, let $W = \sum_{\sigma \in \mathcal{P}} w_\sigma$ and $W' = \sum_{\sigma \in \mathcal{P}} w'_\sigma$, and let $\Delta = W - W'$ be the decrease in total weight. If $\Delta > 0$ and $W' < w_{init} \cdot |\mathcal{P}|$, where w_{init} is the initial reputation given to each policy, we redistribute the lost weight via:

$$w'_\sigma := w'_\sigma + \Delta \frac{X_\sigma}{\sum_\sigma X_\sigma} ,$$

where $\{X_\sigma\}_{\sigma \in \mathcal{P}}$ is a set of random variables independently sampled from the uniform distribution over $[0, 1]$. Thus, the redistribution of the lost weight is randomized, again to break the symmetry between the updates at different nodes.

c) Bound enforcement: Finally, the weights of all policies is bound to the $[0, 1]$ interval, setting $w_\sigma = \min(1, w_\sigma)$. Bounding the weights of policies that have been successful for a long time ensures that the weights can be reduced quickly, and the policies abandoned, should the policies become unsuccessful (that is, prescribe transmissions that cause collisions).

V. ALOHA-QTF

The bandwidth allocation of active nodes using ALOHA-QT is fair in the long term due to two reasons. First, every node has at least $1/2^n$ bandwidth, because a node always selects at least one policy. Second, the more frequently a node transmits, the more frequently it will relinquish a time-slot, and once a time slot is relinquished, all nodes can lay a claim to it. Thus, in the long term the time slots will rotate the node to which they are allocated, and the overall allocation of bandwidth to the nodes will be fair.

This long-term fairness guarantee; however, it is not useful for nodes that are only active during short intervals of time in which they have data to send. We describe here a variation of the ALOHA-QT protocol, which we call ALOHA-QTF and achieves fairness in short time intervals. ALOHA-QTF differs from ALOHA-QT in two respects:

- nodes randomly relinquish time slots only if they use more than their fair share of bandwidth;
- the policy-weight update is made sensitive to the fraction of bandwidth used by each node.

To implement these refinements, ALOHA-QTF keeps track of the number of recently active nodes in the network via a *participant counter*.

A. Counting Active Nodes and Estimating Fair Bandwidth

The ALOHA-QTF protocol achieves fairness by estimating the number of active network nodes, and by using the estimate to tune protocol parameters, among which the policy weight updates. We assume that each node transmits its node ID (such as its MAC address) as part of each packet.

a) Counting active nodes: To estimate the number of active network nodes, each node keeps a sliding window consisting of 2^n slots, where n is the depth of the policy tree. Each slot in the sliding window can contain either a node ID, or a placeholder $\perp$, which indicates no ID. At each network time-slot, the node deletes the left-most slot in the sliding window, corresponding to the oldest information, and adds to the right of the sliding window a new slot containing new information, according to the channel state $c \in \{S, E, C\}$:

- S: the ID of the node that transmitted successfully is added;
- E: $\perp$ is added;
- C: a random ID, taken uniformly at random from the space of node IDs, is added.

For example, if the sliding window contains $[\beta_1, \beta_2, \ldots, \beta_{2^n}]$, and a successful transmission by ID γ is received, the content of the sliding window is updated to $[\beta_2, \ldots, \beta_{2^n}, \gamma]$. Once this is done, the node produces the estimated $\hat{N}$ of the number of active network nodes by counting the number of distinct IDs (excluding $\perp$) in the sliding window. The rule for collisions ensures that, if there are network collisions, $\hat{N}$ tends to over-estimate the number of active nodes, as each collision is counted as a new participant (assuming the space of node IDs is much larger than the actual number of participating nodes, as is usually the case). This over-estimation works in the right direction, as it causes each node to trim down its transmissions, thus reducing the frequency of collisions.

b) Fair and requested bandwidth: From the estimate $\hat{N}$ for the number of active nodes, the node computes the *fair* bandwidth $b_f = 1/\max(1, \hat{N})$: this is the bandwidth that each network node would receive under perfectly fair allocation. The node also computes *requested* bandwidth b_r, which is the fraction of network slots during which the node will transmit. To compute b_r, let $\delta(t, t') = 1$ if $\mathcal{A}_t \cap \mathcal{E}_{t'} \neq \emptyset$ and $\delta(t, t') = 0$ otherwise. In other words, $\delta(t, t') = 1$ if there is a policy active at t which is scheduled to transmit at time t'. At the current time t, the requested bandwidth b_r is defined as

$$b_r = \frac{1}{2^n} \sum_{t'=0}^{2^n-1} \delta(t, t') \; .$$

The requested bandwidth can be computed efficiently by letting $\mathcal{B} := \mathcal{A}_{\sqcup}$, and by then removing from $\mathcal{B}$ all policies that are the descendants, in the policy tree, of policies already in $\mathcal{B}$. Precisely, we remove from $\mathcal{B}$ every policy (i, m) such that there is $(i', m') \in \mathcal{B}$ with $m' < m$ and $i \mod m' = i'$. For example, if $\mathcal{B} = \{(0, 2), (0, 4), (2, 8), (3, 4)\}$, the policies $(0, 4)$ and $(2, 8)$ would be removed from $\mathcal{B}$, as they are descendants of $(0, 2) \in \mathcal{B}$, leading to $\mathcal{B} = \{(\prime, \in), (\ni, \triangle)\}$. Once the set $\mathcal{B}$ is thus minimized, we have $b_r = \sum_{(i,m)\in\mathcal{B}} 1/m$.

B. Fair Policy Update

Once the fair and requested bandwidths b_f, b_r are computed, we modify the policy weight update in two ways.

- First, we apply the *Relinquish* step of Algorithm 1 only if $b_r > b_f$, that is, only if a node uses more than its fair share of bandwidth would it give up its claim on slots.
- Second, we modify the multiplicative update (1) by distinguishing the two cases of policy demotion ($\alpha < 0$) and policy promotion ($\alpha > 0$). For every $\sigma \in \mathcal{E}_t$, the update becomes, for $\alpha < 0$:

$$w'_\sigma = w_\sigma \cdot \exp\big(X_\sigma \cdot \alpha \cdot \min(1, (b_r/b_f)^{1/2})\big) \; , \qquad (2)$$

and for $\alpha > 0$:

$$w'_\sigma = w_\sigma \cdot \exp\big(X_\sigma \cdot \alpha \cdot \max(0, 1 - (b_r/b_f)^2)\big) \; . \qquad (3)$$

The modified weight updates (2) and (3) can be interpreted as follows. In the case of a policy demotion, or $\alpha < 0$, if $b_r < b_f$, that is, if the node is using less than its fair share of bandwidth, then the demotion is scaled down from 1 to $(b_r/b_f)^{1/2} < 1$. In this way, nodes that use less than their fair share see their policies weights reduced less forcefully. Thus, when a node A that uses less than the fair share of bandwidth vies for the use of a time-slot with a node B that uses more than the fair share, the policy weights of B will be reduced more, and A will tend to prevail in the use of the slot.

Conversely, in the case of policy promotion, or $\alpha > 0$, only nodes that request less than their fair share (that is, with $b_r < b_f$) will see their policy promoted. Thus, empty network slots will be more likely to be allocated to nodes whose active policies request less than the fair share.

VI. Performance Evaluation

We compare the performance of ALOHA-QT and ALOHA-QTF with the performance of ALOHA-Q [6], [5], and slotted ALOHA with exponential backoff (ALOHA-EB), by means of simulations. We consider a fully-connected single-channel wireless network in our comparisons. The channel is time slotted, and time slots are organized into transmission frames of 64 time slots each for the case of framed slotted ALOHA-Q. The length of a time slot equals a packet length, which

is assumed to be a constant. The number of active nodes is changed for different scenarios. We compare the performance of the protocols in terms of their network utilization, and of their fairness. For simplicity, the simulations assume that a node knows the fate of any transmission within the same time slot that it took place.

Network utilization: A time slot is either empty or contains a successful transmission or a collision. To show how the network utilization evolves over time, we aggregate time slots in *blocks* of 100, and for each block we can compute the utilization as a fraction of individual slots that contains a successful transmission. Similarly, we can measure the fraction of empty and collision time slots in each block. Using blocks of length 100 offers a compromise between having a fine time resolution, and computing meaningful statistics on each block.

Fairness: The *fairness* of a protocol indicates how equitably the bandwidth of the protocol is distributed among the nodes. We provide two measurements of fairness. The first is the *Jain's index* [11], [10]. Assume that n nodes are active in a time block and let b_i be the number of successful transmissions in the time block by node $i \in [1, \ldots, n]$. Let $B = \sum_{i=1}^{n} b_i$ be the bandwidth in the time block. Jain's index is computed as

$$J = \frac{B^2}{n \sum_{i=1}^{n} b_i^2} \ .$$

Jain's index is a quantity between $1/n$ and 1; it is 1 for a perfectly fair distribution of the channel ($b_i = B/n$ for all i), and it is $1/n$ if only one node gets to use the channel.

The other measure we use is the *bottom-10% fair share.* To compute it, sort the nodes in order of bandwidth, so that $b_1 \leq b_2 \leq \cdots \leq b_n$, and let $m = \lceil n/10 \rceil$. Then, $B_{10} = \sum_{i=1}^{m} b_i$ is the cumulative bandwidth of the bottom 10% of the nodes, and

$$F_{10\%} = \frac{nB_{10}}{mB}$$

is the ratio between the actual bandwidth for the bottom 10%, and the bandwidth the bottom 10% would receive under fair allocation. The $F_{10\%}$ measure is a number between 0 and 1, just like Jain's index. While Jain's index captures the fairness of the overall allocation, the $F_{10\%}$ measure captures how the most "unfortunate" nodes fare in the protocol.

A. ALOHA-Q and ALOHA With Exponential Back-off

a) ALOHA-Q: We implemented ALOHA-Q, the Q-learning version of slotted ALOHA proposed in [6], [5], using a frame length of $M = 64$. Each node stores *q-values* $q_1, q_2, \ldots, q_{64}$, where q_i represents the quality of the decision of transmitting in the i-th slot of the frame. In every frame, a node transmits in the 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 back-off before retrying. As long as the number m of active nodes is no larger than $M = 64$, the performance of ALOHA-Q converges to m/M in the long run; when $m > M$, conflicts for the use of the time-slots in the fixed-length frames arise, and the performance degrades. In our simulations, the number of active nodes is at most about 50, so as to use ALOHA-Q close to its optimum performance.

b) Slotted ALOHA-EB: In slotted ALOHA with exponential back-off, which we denote as ALOHA-EB, every node has an initial transmission probability $p = 1/2$ when it becomes active. The node then updates the probability p whenever a collision, or an empty slot, is detected on the network, setting $p := \alpha p$ in case of collisions, and $p := \min(1, p/\alpha)$ in case of empty slots, where α is a constant that determines adaptation speed; in our simulations we use $\alpha = 0.9$.

This is the *symmetrical* version of ALOHA-EB, in which all nodes transmit with similar probability. For large numbers of nodes, the bandwidth utilization reaches the optimal value of $1/e$, or about 37% [13]. In another version of ALOHA with exponential back-off, each node adjusts its transmission probability as a function of the success, or failure (collision), of its own transmissions only. For this "individual" version of ALOHA with exponential-backoff, it is known that one node will soon dominate and transmit all the time, while the other nodes reduce their transmission probability indefinitely. The network utilization approaches 100%, but the bandwidth is used by one node only. We do not provide comparison graphs for the "individual" version of ALOHA with exponential backoff, as its behavior is well known, and as we are interested in protocols that allow nodes to share the bandwidth.

B. Simulation Results

1) Network With 50 Active Nodes: In Figure 2 we compare the network utilization and fairness of the protocols for a network consisting of 50 active nodes. As the protocols include randomization, we report the average and standard deviation computed over 25 simulations. We aggregate fairness over time blocks that contain on average 20 time slots for each network node: this ensures that the relative number of transmissions by the node are not unduly affected by statistical noise.

Both ALOHA-QT and ALOHA-QTF achieve a channel utilization above 75%. ALOHA-QT reaches 75% utilization in about 500 time slots (that is, in only 10 time-slots per node); the ramp-up of ALOHA-QTF is somewhat slower, and 1000 time-slots (or 20 slots per node) are required to achieve 75% utilization. It is remarkable that 50 nodes can coordinate their transmissions with just a few transmissions per node. The slower ramp-up of ALOHA-QTF is due to the modified weight update (3), which makes the nodes slightly slower in exploiting available network slots. This is the price to pay for the fairness of ALOHA-QTF: the protocol exhibits the highest fairness of the protocols considered, guaranteeing 75% of the average node bandwidth even to the nodes in the 10% lowest bandwidth percentile.

Once the nodes have time to fully adapt, ALOHA-Q should lead to a utilization of $50/64 \approx 78\%$. However, in the 4,000 time-slots spanned by our simulation, the adaptation has not occurred. We note that ALOHA-Q has a $F_{10\%}$ fairness very close to zero, indicating that there is a group of nodes that consistently fails to be able to transmit successfully.

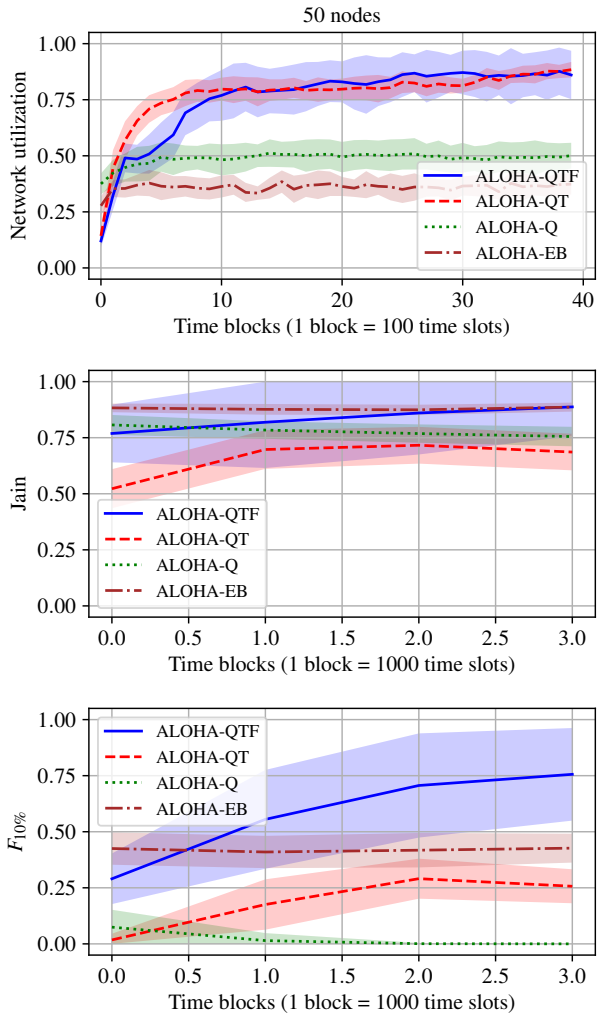

Fig. 2. Network utilization and fairness in a network with 50 active nodes. The results are the average of 25 simulations; the colored bands are plus or minus one standard deviation.

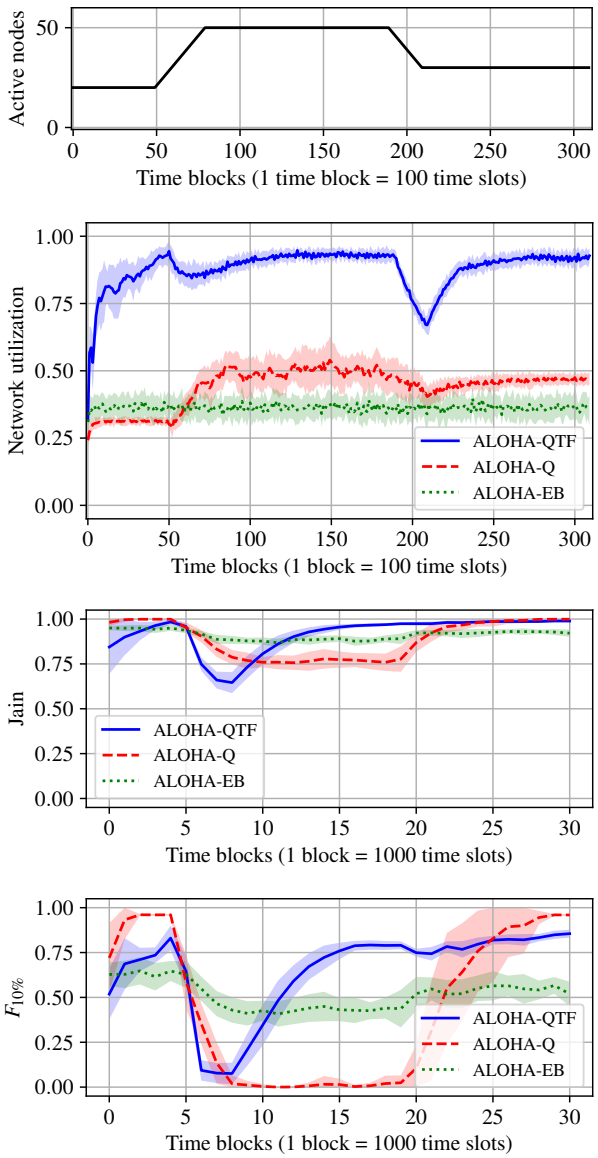

Fig. 3. Network utilization and fairness for ALOHA-QTF, ALOHA-Q, and ALOHA-EB under variable number of active nodes. The results are the average of 20 simulations; the colored bands are plus and minus one standard deviation.

2) Variable Number of Active Nodes: To gain a better understanding of the performance of the protocols when nodes join and leave active transmission, we consider a scenario in which the number of active nodes is initially 20, then increases to 50, and finally decreases to 30, as indicated in Figure 3.

We give results for the average of 20 simulations for each protocol. We see that the bandwidth utilization of ALOHA-QTF is markedly superior to the utilization resulting under both ALOHA-Q and ALOHA-EB. The utilization of ALOHA-QTF declines only during and immediately after the ramp-down from 50 to 30 active nodes: as nodes become inactive, some time-slots are left empty, and the remaining nodes need some time to adapt and utilize these newly-available time-slots. ALOHA-EB is once again close to its optimal utilization of $1/e \approx 0.37$ throughout. The utilization of ALOHA-Q reaches 50% for 50 active nodes, again short of its theoretical limit of $50/64 \approx 78\%$.

The fairness of ALOHA-FQT dips temporarily when the number of active nodes rises from 20 to 50: the 30 newly active nodes need some time to gain a bandwidth comparable to the one of the 20 incumbents. In particular, the $F_{10\%}$ index for ALOHA-FQT dips to close to 0 for a few thousand time-slots. The dip in $F_{10\%}$ is much more pronounced and long-lasting for ALOHA-Q.

3) Nodes Randomly Becoming Active and Inactive: Finally, we considered the case of nodes becoming active or turning inactive at random. We simulated a network with 100 nodes, of which only one is initially active. At each time block (where 1 time block = 100 time slots), each node has probability $1/100$ of switching state, from inactive to active, or vice versa. Thus, on average, in each time block one node changes state. The

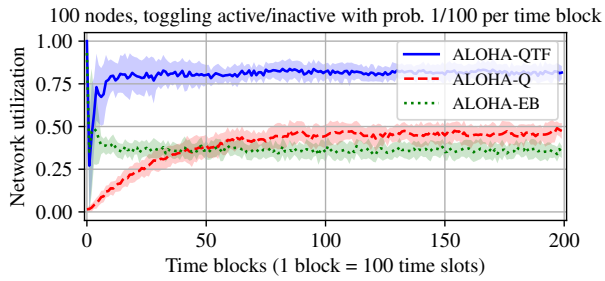

Fig. 4. Network utilization in a network of 100 nodes in which every node has probability $1/100$ of toggling the active/inactive state every 100-time units block. The results are the average of 20 simulations; the colored bands are plus and minus one standard deviation.

utilization is reported in Figure 4. We see that ALOHA-FQT maintains its high level of utilization, above 75%, in spite of the nodes continually joining or leaving the set of active nodes. The utilization of ALOHA-Q grows with the number of active nodes, and then settles at about 50%. Again, the utilization of ALOHA-EB is close to its 37% theoretical optimum.

VII. Conclusions

We presented a new approach to the use of reinforcement learning in the context of slotted ALOHA that dramatically improves channel throughput. The proposed approach is based on the concept of policy trees, and strikes a balance between the two diametrically opposite approaches followed in ALOHA-Q and in the DRL-based approach. As in ALOHA-Q, the learning is based on a fixed set of policies, carefully chosen to guide the nodes towards collaboration. As in the DRL-based approach of [23], there are no fixed transmission frames. This yields protocols that can quickly adapt to changing network conditions, achieving high and fair utilization under a wide range of number of active nodes and network traffic conditions, with none of the computational load required by training neural networks.

We presented two examples of the use of policy trees to access a shared time-slotted channel. The simpler ALOHA-QT is based on reinforcement learning applied to the policy tree that defines periodic policies. Its refinement ALOHA-QTF modifies the policy weight update rules in order to improve the fairness of the bandwidth distribution among active nodes at the price of additional computation and storage requirements. These computational and storage costs are relatively modest, and consequently, ALOHA-QTF can be easily implemented on top of low-power embedded CPUs, or even in custom hardware. Simulation experiments illustrate the marked performance improvements attained with ALOHA-QT and ALOHA-QTF compared to ALOHA-Q and slotted ALOHA with exponential back offs.

Acknowledgments

This material is based upon work sponsored 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 authors and do not necessarily reflect the views of DARPA or AFRL.

References

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

[2] O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3(Nov):363–396, 2002.

[3] J. Capetanakis. Generalized tdma: The multi-accessing tree protocol. *IEEE Transactions on Communications*, 27(10):1476–1484, 1979.

[4] U. Challita, L. Dong, and W. Saad. Deep learning for proactive resource allocation in lte-u networks. In *European wireless technology conference*, 2017.

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

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

[7] D. P. Helmbold, D. D. Long, and B. Sherrod. A dynamic disk spin-down technique for mobile computing. In *Proceedings of the 2nd annual international conference on Mobile computing and networking*, pages 130–142. ACM, 1996.

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

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

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

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

[13] L. Kleinrock. *Queueing systems. Volume I: theory*. wiley New York, 1975.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *Computer Science*, 8(6):A187, 2015.

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

[16] O. Naparstek and K. Cohen. Deep multi-user reinforcement learning for dynamic spectrum access in multichannel wireless networks. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–7. IEEE, 2017.

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

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

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

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

[21] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari. Deep reinforcement learning for dynamic multichannel access in wireless networks. *IEEE Transactions on Cognitive Communications and Networking*, 4(2):257–265, 2018.

[22] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

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