Repeated Games, Optimal Channel Capture, and Open Problems for Slotted Multiple Access

Michael J. Neely
University of Southern California

***Abstract*— This paper revisits a classical problem of slotted multiple access with success, idle, and collision events on each slot. First, results of a 2-user multiple access game are reported. The game was conducted at the University of Southern California over multiple semesters and involved competitions between student-designed algorithms. An algorithm called 4-State was a consistent winner. This algorithm is analyzed and shown to have an optimal expected score when competing against an independent version of itself. The structure of 4-State motivates exploration of the open question of how to minimize the expected time to capture the channel for a n-user situation. It is assumed that the system delivers perfect feedback on the number of users who transmitted at the end of each slot. An efficient algorithm is developed and conjectured to have an optimal expected capture time for all positive integers n. Optimality is proven in the special cases $n \in \{1, 2, 3, 4, 6\}$ using a novel analytical technique that introduces virtual users with enhanced capabilities.**

I. Introduction

This paper studies simple uncoordinated schemes for human users to share a multiple access channel. Such schemes are useful, for example, in internet-of-things situations where distributed users send bursts of data and must learn an efficient channel sharing rule based on feedback. In this direction, the first thrust of this paper considers a series of 2-user slotted multiple access competitions that were conducted at the University of Southern California (USC) over multiple semesters. Students were asked to develop their own algorithms for choosing to transmit, or not transmit, over 100 slots. All pairs of algorithms competed and the one with the best accumulated score was declared the winner. An algorithm called 4-State consistently outperformed the others. It is mathematically shown that 4-State has an optimal expected score over the class of all algorithms that compete against independent versions of themselves. This competition motivates the second thrust of this paper: Investigating the open question of how to minimize the expected time to capture the channel for a n-user situation. This question is of fundamental interest because it explores the learning times and algorithmic protocols required for a collection of indistinguishable users to identify a single user via distributed means.

Algorithms that minimize the first capture time can be used in a variety of contexts. For example, to maximize throughput, an extended algorithm might give the first-capturer unhindered access to the channel for k additional slots (this amortizes the slots that were spent trying to establish the first success). Alternatively, to provide fairness, an algorithm that minimizes the time to the first success given an initial collection of n users might be recursively repeated for $n-1$ users, then $n-2$ users, and so on, in order to construct a fair transmission schedule (such as a round-robin schedule).

The paper treats a classical multiple access scenario with success, collision, or idle on every slot. However, for the case with more than two users, it is additionally assumed that the receiver gives feedback on the number of transmitters at the end of every slot. This is more detailed feedback than collision, idle, or success. The number of transmitters can be determined by various physical estimation techniques, such as measuring the combined energy in the collisions and/or by using the bit signature technique of [1] to count the number of received signatures. The paper develops an algorithm that can be implemented for any number of users n. The algorithm is optimal over the structured class of "divide and conquer" algorithms that throw away a less desirable group once the n indistinguishable users can be discerned into 2 groups. For the special case $n \in \{1, 2, 3, 4, 6\}$ we show there is no loss of optimality by considering this structured class. The proof uses a novel technique that introduces virtual users with enhanced capabilities. Remarkably, the optimal expected capture time for the case of 3 users is strictly smaller than the optimal expected capture time for the case of 2 users. For $n \notin \{1, 2, 3, 4, 6\}$ it is not clear if multiple groups should coexist and transmit simultaneously. This identifies two open questions: (i) Is the algorithm of this paper optimal for all values of n? (ii) What algorithms are optimal when more limited feedback information is used?

A. Decentralized control

The problem of distributed minimization of expected capture time is conceptually similar to the stochastic decision problem for multiple distributed agents described in [2]. The work [2] provides an optimal strategy for $n = 3$ agents but leaves the case $n > 3$ open. To date, there are no known polynomial time solutions for general n. Optimal distributed strategies for extended problems are developed in [3], although complexity can grow exponentially in the number of agents. Unlike [2][3], the agents in the current paper are not labeled and cannot distinguish themselves. Further, the expected capture time problem treated in Section IV-A of this paper (seemingly) suffers from an even worse complexity explosion than the problems in [2][3].

The problem of this paper is similar to the class of *sequential team problems* treated in [4] using concepts of *common*

This work was supported by NSF SpecEES 1824418.

information (see also [5]). In the current paper, common information arises from the feedback that is commonly given to all users. It may prove fruitful to cast the current paper in the framework of [4][5]. However, that framework does not necessarily provide low complexity solutions.

The works [6][7][8] treat distributed channel selection for multi-access problems using a multi-armed bandit framework and using the criterion of asymptotic regret. Like the current paper, [6][7][8] assume the users are not labeled, cannot distinguish themselves, and experience collisions if multiple users pick the same channel. The work [6] treats two users and three stochastic channels, [8] treats multiple users, [7] treats multiple users in a non-stochastic setting. The channel model of the current paper is simpler than [6][7][8] and can easily be shown to yield a constant regret on expected throughput. However, rather than seeking to minimize regret in an asymptotic sense, this paper seeks a more stringent form of optimality: Maximizing throughput over a finite horizon, and minimizing expected capture time over an infinite horizon. The resulting control decision structure is different from [6][7][8] and gives rise to a number of open questions that we partially resolve in this work.

B. Distributed MAC and repeated games

For randomly arriving data, the classical slotted Aloha protocol is well known to achieve stability with throughput close to $1/e$. Splitting and tree-based algorithms that are optimized for Poisson arrivals are treated in [9][10][11][12][13][14] and shown to increase throughput. Algorithms of this type that achieve throughput of 0.4878 are developed in [10][11]; The maximum stable throughput under these assumptions is unknown but an upper bound of 0.587 is developed in [15]. The current paper treats a fixed number of users with an infinite number of packets to send, rather than randomly arriving users. Thus, there is no stability concern and the corresponding distributed implementation issues are different.

The problem of minimizing the expected time to capture the channel in an asymptotic sense and with more limited feedback is considered in [16][17][18][19]. For n users, the work [16] shows the time is $\Omega(\log(n))$ with only success/fail feedback; the work [17] shows the time is $\Theta(\log(\log(n))$ with success/idle/collision feedback. In Section IV-A we consider a problem with more detailed feedback that gives the full number of colliders. In this case it is trivial to show a constant upper bound that does not depend on n. Thus, rather than considering asymptotics with n, we consider the challenging problem of fully minimizing the expected capture time.

The multi-access game treated in thrust 1 of this paper is a *repeated game* and is inspired by the repeated prisoner dilemma games in [20][21] (see also, for example, [22][23][24]). It has been observed that competitions involving repeated prisoner dilemma games are often won by the simple *Tit-for-Tat* strategy that mirrors the opponent decision [20][21]. This is not the case for the multi-access game treated in the current paper. While a Tit-for-Tat strategy can be used in the multi-access competitions, and in several semesters students submitted such algorithms, these algorithms did not win because: (i) Tit-for-Tat is deterministic and so it necessarily scores zero points when competing against itself; (ii) Tit-for-Tat performs poorly when it competes against an algorithm that never transmits. An algorithm called 4-State consistently wins the competitions. This algorithm has an initial randomization phase to capture the channel. It also has a punishing mechanism that seeks to drive the opposing algorithm to fairly take turns. It is shown that 4-State achieves an optimal expected score when competing against an independent version of itself.

C. Feedback details and physical layer

This paper assumes slotted time with fixed length packets. Let $F[t]$ be the number of users who transmit on slot $t \in \{1, 2, 3, \ldots\}$. A success occurs if and only if $F[t] = 1$. For the n-user situation it holds that $F[t] \in \{0, 1, 2, \ldots, n\}$ and

- $F[t] = 0 \iff$ idle.
- $F[t] = 1 \iff$ success.
- $F[t] \geq 2 \iff$ collision.

The value of $F[t]$ is assumed to be given as feedback at the end of slot t. If $n = 2$ then $F[t]$ is equivalent to the idle, success, collision feedback of classical slotted Aloha. If $n > 2$ then the $F[t]$ feedback is more detailed. It is assumed that $F[t]$ can be inferred by the receiver even in the case of a collision. This can be done, for example, by measuring the energy in the combined interfering signals and assuming that this energy is proportional to $F[t]$. Alternatively, the value of $F[t]$ can be inferred by installing a short signature bit pattern in every packet transmission and using a filter to count the number of patterns that arise. This method for counting the number of transmitters is used in the ZigZag multiple access scheme of [1], which uses timing misalignments to perform interference stripping. A soft decision decoding version called SigSag is treated in [25]. Like [1][25], the current paper assumes $F[t]$ can be accurately counted. However, it does not consider interference decoding and treats two or more transmissions on the same slot as a collision from which no information is obtained (other than the number of packets that collided). It is worth noting that any realistic scheme for reporting $F[t]$ will have some probability of reporting error. Nevertheless, for simplicity, it is assumed that $F[t]$ is reported without error.

II. 2-PLAYER MAC GAME

Consider the following 2-player multiple access (MAC) game: Fix T as a positive integer. The game lasts over T consecutive slots. Two players compete to send fixed-length packets over a single channel during this time. A single packet transmission takes one time slot. On each slot $t \in \{1, 2, \ldots, T\}$, each player makes a decision about whether or not to send a packet. For each player $i \in \{1, 2\}$ and each slot $t \in \{1, 2, \ldots, T\}$, define $X_i[t] \in \{0, 1\}$ as the binary decision variable that is 1 if player i decides to send on slot t, and 0 else. There are three possible outcomes on slot t:

- Idle: $(X_1[t], X_2[t]) = (0, 0)$.
- Success: $(X_1[t], X_2[t]) \in \{(0, 1), (1, 0)\}$.
- Collision: $(X_1[t], X_2[t]) = (1, 1)$.

Player $i \in \{1,2\}$ scores a point on slot t if and only if it is the only player to transmit on that slot. Let $F[t] \in \{0,1,2\}$ denote the number of transmissions on slot t, which is equivalent to idle/success/collision feedback. Let (S_1, S_2) be the score of each player at the end of the game:

$$S_1 = \sum_{t=1}^{T} X_1[t](1 - X_2[t])$$
$$S_2 = \sum_{t=1}^{T} X_2[t](1 - X_1[t])$$

Players 1 and 2 know the value of T and the idle/success/collision structure of the game. However, they cannot distinguish themselves as Player 1 or Player 2. Therefore, even if both players have the desire to fairly share the slots, so that one player transmits only on odd slots and the other transmits only on evens, there is no a-priori way to decide who takes the odds and who takes the evens. Each player knows its own decision on slot t. From the feedback $F[t]$ it can infer the decision of its opponent. For each player and each slot $t \geq 2$ define $H_{self}[t]$ and $H_{opponent}[t]$ as the history of decisions up to but not including slot t as seen from the perspective of that player. For example, defining $H_{i,self}[t]$ and $H_{i,opponent}[t]$ for each player $i \in \{1,2\}$ means that

$$\begin{aligned} H_{1,self}[t] &= H_{2,opponent}[t] \\ &= (X_1[1], X_1[2], \ldots, X_1[t-1]) \end{aligned} \quad (1)$$

$$\begin{aligned} H_{1,opponent}[t] &= H_{2,self}[t] \\ &= (X_2[1], X_2[2], \ldots, X_2[t-1]) \end{aligned} \quad (2)$$

A. Random and deterministic algorithms

Algorithms are allowed to make any desired decisions based on the feedback, including randomized decisions. A general algorithm can be mathematically represented by a sequence of Borel measurable functions $f_1, f_2, \ldots, f_T$ such that

$$f_1 : [0,1) \to \{0,1\}$$
$$f_t : [0,1) \times \{0,1\}^{t-1} \times \{0,1\}^{t-1} \to \{0,1\} \quad \forall t \in \{2,\ldots,T\}$$

where the decisions $X[t]$ are given by

$$X[1] = f_1(U)$$
$$X[t] = f_t(U, H_{self}[t], H_{opponent}[t]) \quad \forall t \in \{2,\ldots,T\}$$

where U is a randomization variable that is uniformly distributed over $[0,1)$. The randomness of U can be used to facilitate randomized decisions.[1] Players are assumed to use independent randomization variables. In particular, if Players 1 and 2 implement independent versions of the same algorithm, they use the same functions $f_1, \ldots, f_T$ but they use independent random variables $U_1 \sim U[0,1)$ and $U_2 \sim U[0,1)$.

A *deterministic algorithm* is one that contains no randomization calls. Such an algorithm can be characterized by a sequence of functions $\{g_t\}_{t=1}^{T}$ that only use $H_{self}[t]$ and $H_{opponent}[t]$ as inputs (with no randomization variable)

$$g_1 \in \{0,1\}$$
$$g_t : \{0,1\}^{t-1} \times \{0,1\}^{t-1} \to \{0,1\} \quad \forall t \in \{2,\ldots,T\}$$

[1]A random variable $U \sim U[0,1)$ has binary expansion $U = \sum_{m=1}^{\infty} B_m 2^{-m}$ with $\{B_m\}_{m=1}^{\infty}$ i.i.d. equally likely bits that can be used to make sequences of randomized decisions.

So $X[1] = g_1$ is the deterministic decision on slot $t = 1$ and

$$X[t] = g_t(H_{self}[t], H_{opponent}[t]) \quad \forall t \in \{2,\ldots,T\} \quad (3)$$

Lemma 1: A deterministic algorithm scores zero points against itself.

Proof: See [26]. □

B. Tournament structure

Competitions of these 2-player MAC games were conducted over 7 semesters amongst students in the EE 550 Data Networks class at the University of Southern California. If there were k algorithms competing in a given semester, then all algorithms $i \in \{1,\ldots,k\}$ were paired against all other algorithms $j \in \{1,\ldots,k\}$. This included a pairing (i,i) where algorithm i plays against an independent version of itself. For each algorithm pair (i,j), the scores of 1000 independent simulations of 100-slot games were averaged to provide an estimate of the expected score $(\mathbb{E}[S_i], \mathbb{E}[S_j])$ associated with algorithm i playing algorithm j. The total score of an algorithm is the sum of its scores accumulated over all other algorithms that it played (including itself). The algorithm with the largest accumulated score was declared the winner of the competition for that semester.

The competing algorithms included algorithms that students designed, an instructor-designed algorithm called 4-state, and two special algorithms called AlwaysTransmit and NeverTransmit. AlwaysTransmit transmits on every slot regardless of history. No opponent can score against AlwaysTransmit. This algorithm is maximally greedy and was entered into the competition in order to view its accumulated score in comparison with the other algorithms. In contrast, NeverTransmit never transmits and never scores any points. It tests the ability of other algorithms to adapt to the situation where they are the only ones who want channel access.

In addition to the overall score in the competition, the quality of an algorithm can be understood in terms of figures of merit α and β defined below:

- Self-competition score α: This is the expected score $\mathbb{E}[S_1]$ when an algorithm plays an independent copy of itself. This figure of merit is useful because a good MAC algorithm will be used by others and hence must perform well against itself.
- No-competition score β: This is the expected score $\mathbb{E}[S_1]$ when an algorithm plays NeverTransmit. This figure of merit is useful because a good MAC algorithm should adapt when nobody else is using the channel.

C. Special algorithms

The following algorithms are of key interest:

1) Tit-for-Tat-0 (TFT-0): This is a deterministic policy that operates as follows:
   - $X[1] = 0$.
   - For $t \in \{2,3,\ldots,T\}$: $X[t] = X_{opponent}[t-1]$.

   A variation called Tit-for-Tat-1 (TFT-1) differs only by having $X[1] = 1$. Both Tit-for-Tat-0 and Tit-for-Tat-1 mirror the opponent decisions with one slot delay. This

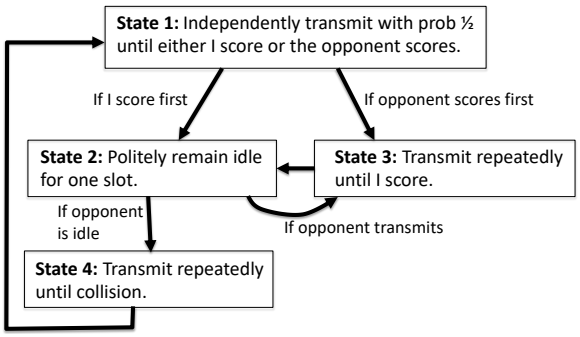

Fig. 1. Algorithm 4-State. The algorithm starts in state 1.

is similar in spirit to the Tit-for-Tat algorithm considered for prisoner dilemma games [20][21]. It can be shown that neither version ever looses a game by more than 1 point (regardless of the opponent). However, both Tit-for-Tat-0 and Tit-for-Tat-1 are deterministic and so they both have a Self-Competition score of 0. They also have poor No-Competition scores (0 and 1 for Tit-for-Tat-0 and Tit-for-Tat-1, respectively).

2) 4-State: This algorithm is given by the diagram in Fig. 1. The idea is to start by randomly transmitting until the first success, then move to a turn-based policy that oscillates between states 2 and 3. On its turn (state 3) it repeatedly transmits until it scores. This is a punishing mechanism designed to force the opponent to acknowledge its turn. Like Tit-for-Tat, it is not difficult to show that 4-State cannot loose a game by more than 1 point. When competing against an independent version of itself, 4-state uses only the first 3 states. The fourth state is added to detect whether or not it is playing against NeverTransmit: If the opponent does not take its turn in state 2, the algorithm moves to state 4 and repeatedly transmits on all remaining slots (unless there is a collision).

D. Fall 2021 competition

A closeup look at the Fall 2021 competition is given in Fig. 2. That was a small competition with 4-State, AlwaysTransmit, NeverTransmit, and only 7 student algorithms. Each row $i \in \{1, \ldots, 10\}$ of Fig. 2 shows the score when algorithm i played against each other algorithm (averaged over 1000 independent 100-slot games). Key algorithms of the 10 are:

- A1: NeverTransmit
- A2: AlwaysTransmit
- A3: 4-State
- A6: This student used Tit-for-Tat-1

The 4-State algorithm (A3) dominated the competition by earning 324.637 points. Its simulated Self-Competition score of 49.48 is shown in cell $(A3, A3)$ and this is consistent with the analytical result of the next section. The AlwaysTransmit algorithm (A2) earned 228.986 points. Tit-for-Tat-1 (A6) earned only 127.597 points. Its poor performance was due to a Self-Competition score of 0 and a No-Competition score of only 1. However, it learns to fairly share with 4-State (earning 49.69 points in that game, see the (A6, A3) cell). In tournaments with a larger number of students, there are more student algorithms that learn to share and Tit-for-Tat-1 performs better (see Section II-E and [26]). The 4-State algorithm receives an average score per game of 32.46. It is interesting to note the following data which is not in the table: When 4-State is replaced by Tit-for-Tat-0 the average score per game for Tit-for-Tat-0 is 12.18; that for Tit-for-Tat-1 is 7.76. Thus, the greedy version of Tit-for-Tat does worse. The performance of 4-State in the Fall 2021 semester is qualitatively similar to its performance over 7 different semesters (4-State winning each time) as reported in [26].

E. Figures of merit

The first two rows of Fig. 3 provide analytical values for the Self-Competition score α and the No-Competition score β of 4-State, Tit-for-Tat-0, Tit-for-Tat-1, and AlwaysTransmit (values of α and β for 4-State are derived in Section III). The analytical values in the first two rows are consistent with simulation results. The third row in Fig. 3 provides a "Tournament Score" γ which is a simulation result on the average score per game, considering all games played, in a large tournament with 135 algorithms (including these five algorithms together with all student algorithms that were created over 7 semesters). Each algorithm was paired against all 135 other algorithms (including an independent version of itself) in 1000 independent runs of 100-slot games. For this tournament 4-State performs best; the greedy version of Tit-for-Tat does significantly worse than the non-greedy version; the AlwaysTransmit algorithm does significantly worse than 4-State, Tit-for-Tat-0, and Tit-for-Tat-1.

III. Analysis of the 2-player MAC game

Theorem 1: Fix T as a positive integer. In a T-slot game, the Self-Competition score for 4-State is:

$$\alpha = \frac{T-1}{2} + \frac{1}{2^{T+1}} \tag{4}$$

Further, no other algorithm can achieve a larger Self-Competition score.

Proof: (Theorem 1 achievability) Consider two independent versions of 4-State that compete over T slots. Let S_1 and S_2 be the resulting scores. The algorithms are identical so

$$\alpha = \mathbb{E}[S_1] = \mathbb{E}[S_2] \tag{5}$$

Let $Y \in \{0, 1, \ldots, T\}$ denote the random number of initial slots in which nobody scores ($Y = T$ if nobody ever scores). When 4-State plays against itself, once the first player scores, exactly one player will score on each slot thereafter. Thus

$$T = Y + S_1 + S_2$$

Taking expectations of both sides and using (5) gives

$$T = \mathbb{E}[Y] + 2\alpha$$

Thus

$$\alpha = \frac{T - \mathbb{E}[Y]}{2} \tag{6}$$

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	Totals
A1	0	0	0	0	0	0	0	0	0	0	0
A2	100	0	1	0	47	0	19.86	10.79	0	50.34	228.986
A3	98.04	0	49.48	0	48.43	49.36	21.15	25.13	0	33.06	324.637
A4	40.13	0	1	0	19.78	0	20.49	10.74	0	36.4	128.531
A5	50.4	0	0.51	0	24.6	0	9.68	5.39	0	25.33	115.909
A6	1	0	49.69	0	0.5	0	19.29	31.87	0	25.24	127.597
A7	100	0	19	0.73	50.8	19.04	16.48	19.05	0.48	34.57	260.159
A8	50.54	0	24.31	0.02	24.9	31.52	18.52	24.72	0.1	27.17	201.788
A9	100	0	1	0	51.3	0	19.81	10.78	0	38.15	221.048
A10	49.92	0	17.1	13.83	24.65	24.7	15.45	22.67	11.94	24.76	205.021

Fig. 2. Results of multi-access game in Fall 2021 semester. Each game consists of 100 slots. Results are averaged over 1000 independent games. The score of Algorithm $i \in \{1, ..., 10\}$ is given in row i. A1=NeverTransmit; A2=AlwaysTransmit; A3=4-State; A6=TFT-1.

	4-State	TFT-0	TFT-1	AlwaysTran
α	49.500	0	0	0
β	98.000	0	1	100
γ	24.613	20.410	15.326	10.714

Fig. 3. A table of scores (Self-Competition α; No-Competition β; Tournament γ) for 4-State, Tit-for-Tat-0, Tit-for-Tat-1, and AlwaysTransmit.

The random variable Y has the following distribution

$$P[Y = i] = (1/2)^{i+1} \quad \forall i \in \{0, 1, \ldots, T-1\} \tag{7}$$
$$P[Y = T] = (1/2)^T \tag{8}$$

Hence

$$\mathbb{E}[Y] = \sum_{i=0}^{T} iP[Y = i] = 1 - 2^{-T}$$

Substituting this expression into (6) proves (4). □

Proof: (Theorem 1 converse) Consider any algorithm that is independently used for both players. For convenience, assume the algorithm is designed to run over an infinite sequence of slots $t \in \{1, 2, 3, \ldots\}$ (an algorithm designed for T slots can be extended to run over an infinite horizon by choosing to not transmit after time T). Arbitrarily assign one of the algorithms as Player 1 and assign its counterpart (identical) algorithm as Player 2. For each positive integer T, define $V[T]$ as the random sum of scores of both players over the first T slots:

$$V[T] = \sum_{t=1}^{T} [X_1[t](1 - X_2[t]) + X_2[t](1 - X_1[t])]$$

By symmetry, it follows that the Self-Competition score (over T slots) is $\mathbb{E}[V[T]]/2$. For each positive integer T, define $s[T]$ as the supremum value of $\mathbb{E}[V[T]]$ over all algorithms that independently compete against themselves. Define $s[0] = 0$. We want to show $s[T]/2 \leq \alpha$ for all nonnegative integers T, where α is the value in (4). Specifically, we want to show

$$s[T] \leq T - 1 + (1/2)^T \quad \forall T \in \{0, 1, 2, 3, ...\} \tag{9}$$

We use induction: Suppose (9) holds for $T = k$ for some nonnegative integer k (it holds for $k = 0$ since $s[0] = 0$). We show it also holds for $T = k + 1$. Since at most one player can score on each slot, we surely have

$$V[k + 1] \leq k + 1 \tag{10}$$

Let A be the event that there is a success by one of the players on slot 1, so that A^c is the event that the first slot results in either an Idle or a Collision. A key observation is

$$\mathbb{E}[V[k + 1]|A^c] \leq s[k] \tag{11}$$

since the event A^c means that neither player scored on the first slot, there are k slots remaining to accumulate the total score $V[k+1]$, and no information has been conveyed to either player on this first slot that would make the expected score over the remaining k slots larger than $s[k]$.[2]

Let q be the probability that the algorithm transmits on the very first slot. Then

$$P[A] = 2q(1 - q) \leq \sup_{q \in [0,1]} 2q(1 - q) = 1/2 \tag{12}$$

We have

$$\begin{aligned}
&\mathbb{E}[V[k + 1]] \\
&= \mathbb{E}[V[k + 1]|A] P[A] + \mathbb{E}[V[k + 1]|A^c](1 - P[A]) \\
&\overset{(a)}{\leq} (k + 1)P[A] + s[k](1 - P[A]) \\
&= s[k] + P[A](k + 1 - s[k]) \\
&\overset{(b)}{\leq} s[k] + (1/2)(k + 1 - s[k]) \\
&= (1/2)(k + 1) + (1/2)s[k] \\
&\overset{(c)}{\leq} (1/2)(k + 1) + (1/2)(k - 1 + (1/2)^k) \\
&= k + (1/2)^{k+1}
\end{aligned}$$

where (a) holds by (10) and (11); (b) holds by (12) and the fact $k + 1 - s[k] \geq 0$ (observe that $s[k] \leq k$ since at most one point can be scored per slot); (c) holds by the induction assumption that (9) holds for $T = k$. Thus

$$\mathbb{E}[V[k + 1]] \leq k + (1/2)^{k+1}$$

This holds for all algorithms. Taking the supremum value of $\mathbb{E}[V[k + 1]]$ over all possible algorithms gives

$$s[k + 1] \leq k + (1/2)^{k+1}$$

[2]If a non-success on the first slot made the expected score on the remaining slots *more* than $s[k]$, one could use an algorithm that starts under the assumption that a non-existent preliminary slot just had a non-success. That would achieve an expected k-slot score that is larger than $s[k]$, a contradiction.

which proves that (9) holds for $T = k + 1$. □

Lemma 2: Fix T as a positive integer. In a T-slot game, the No-Competition score β for 4-State is

$$\beta_4 = T - 2 + \frac{3}{2^T} \tag{13}$$

Proof: See [26]. □

IV. Multi-user MAC

Now consider the MAC problem with n users. In this version of the problem, we assume the receiver can detect the number of users who transmitted on each slot (as discussed in Section I-C). The feedback $F[t]$ given to the users at the end of each slot t is equal to the number of users who transmitted:

$$F[t] = 0 \implies Idle$$
$$F[t] = 1 \implies Success$$
$$F[t] \in \{2, \ldots, n\} \implies Collision$$

Note that the feedback value $F[t]$ specifies the number of transmitters on slot t, but does not indicate *which* users transmitted. If $n = 2$ then this feedback is equivalent to success/idle/collision feedback. However, if $n > 2$ this feedback is more detailed. It is assumed that all users know the value n at the start, and this value does not change for the timescale of interest. For knowledge of n, one can imagine a "slot $t = 0$" where all users agree to transmit, so that the feedback signal $F[0] = n$ communicates n to all users.

The users have no way to distinguish themselves at the start of slot $t = 1$, so there are no labels $\{1, 2, \ldots, n\}$ that the users can identify with. The central goal of this section and the next is to develop an algorithm that all users can independently implement that minimizes the expected time to the first success. Minimizing the time to the first success is also useful for either maximizing throughput or for quickly assigning labels to users. Define z_n^* as the infimum expected time to the first success in a system with n users.

A. Minimizing the expected time to the first success

For each positive integer n, this subsection develops an algorithm that is independently implemented by n users and seeks to minimize the expected time to the first success. There is no deadline, and so this is an infinite horizon problem. Let Z_n denote the random time until the first success under a given n-user algorithm. Define

$$z_n = \mathbb{E}[Z_n]$$

Let p_n denote the transmission probability on the first slot. The idea is to have the n users independently transmit with probability p_n on slot 1 and then receive feedback $F[1]$ that specifies the number of transmitters. If $F[1] = 1$ there was a single success and the algorithm terminates. If $F[1] \in \{0, n\}$ then no information that can distinguish the users is gained and we repeat. If $F[1] = i$ with $i \in \{2, \ldots, n-1\}$ then the n users are partitioned into a group of size i and a group of size $n - i$, the values of z_i and z_{n-i} are compared, the least desirable group is thrown away and the problem is recursively solved on the remaining group with a residual expected time $\min\{z_i, z_{n-i}\}$. This section optimizes over this structured class of algorithms. It should be noted that there is no general proof that throwing away a group is optimal: Section V proves this is optimal for the special case $n \in \{1, 2, 3, 4, 6\}$.

1) Case $n = 1$: If $n = 1$ the algorithm is to transmit with probability $p_1 = 1$, which yields $z_1 = 1$.

2) Case $n = 2$: If $n = 2$ the algorithm is for both users to independently transmit each slot with probability $p_2 = 1/2$, so that $z_2 = 2$.

3) Case $n = 3$: If $n = 3$, the users independently transmit with probability $p \in (0, 1)$ on the first slot (the value of p shall be optimized later). The feedback after slot 1 satisfies $F[1] \in \{0, 1, 2, 3\}$. Based on the value of $F[1]$ do the following:

- $F[1] = 0$ (Idle): Repeat.
- $F[1] = 1$ (Success): Done.
- $F[1] = 2$ (Collision between 2 users): The third user that did not transmit on slot 1 will transmit alone on slot 2; the other two users are silent on slot 2.
- $F[1] = 3$ (Collision between 3 users): Repeat.

The expected time to the first success is:

$$z_3 = \sum_{i=0}^{3} \mathbb{E}[Z_3 | F[1] = i] \binom{3}{i} p^i (1-p)^{3-i} \tag{14}$$

Under this scheme we have

$$\mathbb{E}[Z_3 | F[1] = 0] = 1 + z_3$$
$$\mathbb{E}[Z_3 | F[1] = 1] = 1$$
$$\mathbb{E}[Z_3 | F[1] = 2] = 2$$
$$\mathbb{E}[Z_3 | F[1] = 3] = 1 + z_3$$

Substituting these into (14) gives

$$z_3 = \frac{1 + 3p^2(1-p)}{1 - p^3 - (1-p)^3}$$

The value of $p \in (0, 1)$ that minimizes the right-hand-side is

$$p_3 \approx 0.411972$$

and so

$$z_3 = \inf_{p \in (0,1)} \left\{ \frac{1 + 3p^2(1-p)}{1 - p^3 - (1-p)^3} \right\} \approx 1.78795 \tag{15}$$

It is surprising that $z_3 < z_2$, meaning that it is more efficient to start with 3 users than to start with 2 users.

4) Case $n \in \{4, 5, 6, \ldots\}$: Fix n as the number of users and assume $n \geq 4$. On the first slot the users independently transmit with probability p (to be optimized later) and react to the feedback $F[1] \in \{0, 1, \ldots, n\}$ as follows:

- $F[1] = 0$ (idle): Repeat.
- $F[1] = 1$ (succes): Done.
- $F[1] = i \in \{2, \ldots, n-1\}$ (collision of i users): This partitions the users into two groups, one of size i (consisting of those users who transmitted on the first slot) and one of size $n - i$ (the others). If $z_i \leq z_{n-i}$ then follow the optimal algorithm for minimizing the time to the first success for i users, utilizing only the i users who transmitted (the remaining $n - i$ users stay silent forever). Else, follow the optimal algorithm for minimizing the time to the first success for $n - i$ users, utilizing only the

$n-i$ users who did not transmit in the first slot while the remaining users are silent.

- $F[1] = n$ (collision of n users): Repeat.

The expected time to the first success is then

$$\begin{aligned} z_n &= \sum_{i=0}^{n} \mathbb{E}\left[Z_n | F[1]=i\right] \binom{n}{i} p^i (1-p)^{n-i} \\ &= (1+z_n)(p^n + (1-p)^n) + np(1-p)^{n-1} \\ &\quad + \sum_{i=2}^{n-1} (1+\min\{z_i, z_{n-i}\}) \binom{n}{i} p^i (1-p)^{n-i} \end{aligned}$$

Thus

$$z_n = \inf_{p\in(0,1)} \left\{ \frac{1+\sum_{i=2}^{n-1} \min\{z_i, z_{n-i}\} \binom{n}{i} p^i (1-p)^{n-i}}{1-p^n-(1-p)^n} \right\} \tag{16}$$

The values of p_n and z_n can be recursively computed in terms of $z_1, \ldots, z_{n-1}$. The first several values are given below:

n	p_n	z_n
1	1	1
2	0.5	2
3	0.411972	1.78795
4	0.302995	2.13454
5	0.238640	2.15575
6	0.191461	2.26246
7	0.166629	2.27543

V. Converse

The z_n and p_n values of the previous section are optimized over algorithms that partition users into two groups and throw away the least desirable group. It is not clear if gains can be achieved by keeping track of an ever-increasing number of groups and having multiple groups probabilistically transmit at the same time. The information state of the problem is remarkably complex. For each positive integer n, define z_n^* as the infimum expected time for the first success, considering all possible algorithms that can be independently implemented by n users. Clearly $1 \le z_n^* \le z_n$, where z_n are the values associated with the proposed algorithm of the previous section. The author conjectures that $z_n = z_n^*$ for all n. This section proves the conjecture for the special cases $n \in \{1,2,3,4,6\}$. The cases $n=4$ and $n=6$ are particularly challenging. The following theorem is proven in the next subsections.

Theorem 2: For $n \in \{1,2,3,4,6\}$ we have $z_n^* = z_n$.

A. Preliminaries

By definition of z_n^*, for every $\epsilon > 0$ there is an algorithm that gets arbitrarily close to z_n^*, so that

$$z_n^* \le \mathbb{E}\left[Z\right] \le z_n^* + \epsilon \tag{17}$$

where Z is the random time to the first success. If $\epsilon \in (0,1/2)$ and $n \ge 2$ it can be shown that any algorithm that satisfies (17) uses a probability p for transmitting on the first slot that satisfies $p \in [a_n, b_n]$ for certain values a_n, b_n that satisfy $0 < a_n < b_n < 1$ (see [26]).

Lemma 3: Fix $n \in \{2,3,4,\ldots\}$ and $\epsilon \in (0,1/2)$. Consider an algorithm that satisfies (17) and let $p \in [a_n, b_n]$ be its probability of transmitting on the first slot. Then

$$z_n^* \ge \frac{1-\epsilon+\sum_{i=2}^{n-1} \mathbb{E}\left[Z-1|F[1]=i\right] \binom{n}{i} p^i (1-p)^{n-i}}{1-p^n-(1-p)^n} \tag{18}$$

Proof: See [26]. □

B. Cases $n \in \{2,3\}$

For the case $n=2$, see [26] for a proof that $z_2^* \ge z_2$. Suppose $n=3$. Fix $\epsilon \in (0,1/2)$. From (18):

$$\begin{aligned} z_3^* &\ge \frac{1-\epsilon+\mathbb{E}\left[Z-1|F[1]=2\right] 3p^2(1-p)}{1-p^3-(1-p)^3} \\ &\overset{(a)}{\ge} \frac{1-\epsilon+3p^2(1-p)}{1-p^3-(1-p)^3} \\ &\overset{(b)}{\ge} \inf_{q\in[a_3,b_3]} \left\{ \frac{1-\epsilon+3q^2(1-q)}{1-q^3-(1-q)^3} \right\} \end{aligned}$$

where (a) holds because if $F[1]=2$ then $Z \ge 2$; (b) holds because $p \in [a_3, b_3]$. This holds for all $\epsilon \in (0,1/2)$. Since $q \in [a_3, b_3]$ bounds the denominator of the last expression away from 0, we can take $\epsilon \to 0$ to obtain

$$\begin{aligned} z_3^* &\ge \inf_{q\in[a_3,b_3]} \left\{ \frac{1+3q^2(1-q)}{1-q^3-(1-q)^3} \right\} \\ &\ge \inf_{q\in(0,1)} \left\{ \frac{1+3q^2(1-q)}{1-q^3-(1-q)^3} \right\} = z_3 \end{aligned}$$

where z_3 is defined in (15).

C. Case $n=4$

Suppose there are 4 users. From (18) and the fact $\mathbb{E}\left[Z-1|F[1]=3\right] \ge 1$ we have

$$z_4^* \ge \frac{1-\epsilon+\mathbb{E}\left[Z-1|F[1]=2\right] 6p^2(1-p)^2 + 4p^3(1-p)}{1-p^4-(1-p)^4}$$

The main challenge is to show the following inequality:

$$\mathbb{E}\left[Z-1|F[1]=2\right] \ge 2 \tag{19}$$

Once (19) is established, the proof proceeds as in the $n=3$ case to show $z_4^* \ge z_4$ (see [26] for details).

D. Proving (19)

We have $n=4$ users. Consider the situation at the end of slot 1 given that $F[1]=2$. Let $Z-1$ denote the remaining time until the first success. We shall call this the *2-group situation*: Group A consists of the 2 users who transmitted on the first slot and Group B consists of the 2 who did not. Let ALG_A be an algorithm that the two users in group A independently implement for the remaining slots; let ALG_B be an algorithm that the two users in group B independently implement for the remaining slots. Define $Time(ALG_A, ALG_B)$ as the expected remaining time (not including the first slot $t=1$) to the first success (from either group). We want to show

$$Time(ALG_A, ALG_B) \ge 2$$

We construct a new system with only two devices, called *virtual devices*, with enhanced capabilities. We show:

1) The new system can emulate the 2-group situation.
2) Any such emulation in the new system must take at least 2 slots on average.

For simplicity we shift the timeline so that the current slot 2 is now called slot 1. The two virtual devices act independently as follows: On each slot $t \in \{1, 2, 3, \ldots\}$, each virtual device can choose to send any integer number of packets. The feedback at the end of slot t is $F[t] \in \{0, 1, 2, \ldots\}$, which is the sum number of packets sent by both virtual devices. Consider the constraint that both devices must independently implement the same algorithm. Let R be the random time to see the first success, being a slot where exactly one of the devices sends exactly one packet. Let r^* be the infimum value of $\mathbb{E}[R]$ over all possible algorithms with this structure.

Lemma 4: With two enhanced virtual devices, the minimum expected time to the first success is $r^* = 2$.

Proof: See [26]. □

We now show these two virtual devices can emulate the $k = 4$ user scenario with two groups A and B running ALG_A and ALG_B, respectively. The first virtual device runs two separate programs: One that emulates an independent user implementing ALG_A, the second independently implementing ALG_B as if it is a separate user. If both ALG_A and ALG_B at this device decide to transmit on the current slot, the device sends two packets. If only one of ALG_A and ALG_B decide to transmit, the device sends 1 packet. If neither ALG_A nor ALG_B decide to transmit, the device sends zero packets. The second device does a similar emulation independently. The feedback signaling $F[t]$ on each slot t is the same as if there were 4 users that were initially configured in the 2-group scenario. Hence, the expected time to achieve the first success is the same as $Time(ALG_A, ALG_B)$. Since this is also a situation where two enhanced devices independently implement the same algorithm, we have

$$Time(ALG_A, ALG_B) \geq r^* = 2$$

E. The case $n = 6$ and beyond

The case $n = 6$ is proven in [26]. A brief exploration of this problem for multiple channels is also in [26].

VI. Conclusion

A MAC game was introduced. Unlike related prisoner dilemma games where Tit-for-Tat policies tend to win, the winning algorithm is a 4-State policy with a randomized initial phase. Randomization is fundamental because deterministic algorithms have a zero self-competition score. The policy 4-State was mathematically shown to maximize the expected number of points when competing against an independent version of itself. A closely related problem of minimizing the expected time required to first capture the channel was explored. An efficient algorithm was developed and shown to be optimal when the number of users is in the set $\{1, 2, 3, 4, 6\}$. The optimality proof uses a technique that introduces virtual users with enhanced capabilities.

References

[1] S. Gollakota and D. Katabi. Zigzag decoding: combating hidden terminals in wireless networks. *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, 2008.

[2] H. S. Witsenhausen. A stochastic decision problem. In T. M. Cover and B. Gopinath, editors, *Open Problems in Communication and Computation*, chapter 3.8, pages 49–50. Springer-Verlag, New York, 1987.

[3] M. J. Neely. Distributed stochastic optimization via correlated scheduling. *IEEE/ACM Transactions on Networking*, 24(2):759–772, April 2016.

[4] A. Nayyar and D. Teneketzis. Common knowledge and sequential team problems. *IEEE Transactions on Automatic Control*, 64(12):5108–5115, 2019.

[5] H. S. Witsenhausen. A standard form for sequential stochastic control. *Mathematical systems theory*, 7(1):5–11, 1973.

[6] S. Bubeck and T. Budzinski. Coordination without communication: optimal regret in two players multi-armed bandits. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 916–939. PMLR, 09–12 Jul 2020.

[7] S. Bubeck, Y. Li, Y. Peres, and M. Sellke. Non-stochastic multi-player multi-armed bandits: Optimal rate with collision information, sublinear without. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 961–987. PMLR, 09–12 Jul 2020.

[8] D. Kalathil, N. Nayyar, and R. Jain. Decentralized learning for multiplayer multiarmed bandits. *IEEE Transactions on Information Theory*, 60(4):2331–2345, 2014.

[9] D. P. Bertsekas and R. Gallager. *Data Networks*. New Jersey: Prentice-Hall, Inc., 1992.

[10] J. Mosely and P. A. Humblet. A class of efficient contention resolution algorithms for multiple access channels. *IEEE Trans. Comm.*, COM-33:145–151, 1985.

[11] B. S. Tsybakov and V. A. Mikhailov. Random multiple access of packets: Part and try algorithm. *Problemy Peredachi Inform. (USSR)*, 16:65–79, 1980.

[12] J. F. Hayes. An adaptive technique for local distribution. *IEEE Trans. Comm.*, COM-26:1178–1186, 1978.

[13] J. I. Capetanakis. The multiple access broadcast channel: Protocol and capacity considerations. *IEEE Trans. Inform. Theory*, IT-25:505–515, 1979.

[14] B. S. Tsybakov and V. A. Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Problemy Peredachi Inform. (USSR)*, 14(4):32–59, 1978.

[15] V. A. Mikhailov and B. S. Tsybakov. Upper bound for the capacity of a random multiple access system. *Problemy Peredachi Inform. (USSR)*, 17:90–95, 1981.

[16] E. Kushilevitz and Y. Mansour. An $\omega(d \log(n/d))$ lower bound for broadcast in radio networks. *SIAM J. Comput*, 27(3):702–712, June 1998.

[17] D. E. Willard. Log-logarithmic selection resolution protocols in a multiple access channel. *SIAM J. Comput.*, 15:468–477, 1986.

[18] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *J. Comput. System Sci.*, 45:104–126, 1992.

[19] T. Jurdzinski and G. Stachowiak. The cost of synchronizing multiple-access channels. *Proc. ACM Symposium on Principles of Distributed Computing*, pages 421–430, 2015.

[20] R. Axelrod. More effective choice in the prisoner's dilemma. *The Journal of Conflict Resolution*, 24(3):379–403, 1980.

[21] R. Axelrod and W. D. Hamilton. The evolution of cooperation. *Science*, 211(4489):1390–1396, 1981.

[22] W. Poundstone. *Prisoner's Dilemma: John von Neumann, Game Theory, and the Puzzle of the Bomb*. First Anchor Books Ed., 1993.

[23] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, MA, 1994.

[24] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, 2007.

[25] A. S. Tehrani, A. G. Dimakis, and M. J. Neely. Sigsag: Iterative detection through soft message-passing. *IEEE Journal of Selected Topics in Signal Processing*, 5(8):1512 – 1523, Dec. 2011.

[26] M. J. Neely. Repeated games, optimal channel capture, and open problems for slotted multiple access. *arXiv:2110.09638v3*, Dec. 2021.