Random Access Algorithms without Feedback using Finite Geometry

G. Peeters, R. Bocklandt and B. Van Houdt
University of Antwerp - IBBT, Middelheimlaan 1, B-2020 Antwerp, Belgium

Abstract

A new class of random access algorithms for systems without feedback is introduced and analyzed. A finite population of users is assumed, where each user transmits a packet R times within the next N time slots (and all packets have an equal length of one slot). To improve the performance achieved by randomly selecting these R slots, user codes are designed such that any two users will only transmit simultaneously in at most one slot.

Using finite affine and projective spaces in which points correspond to slots and lines to user codes, we develop an algorithm that generates a maximum set $\mathcal{S}_{N,R}$ of user codes for specific values of N and derive a closed formula for the success probability of a single user assuming we have $|\mathcal{S}_{N,R}|$ users.

We also provide a method to select T user codes from the set $\mathcal{S}_{N,R}$ in case the user population consists of only $T \leq |\mathcal{S}_{N,R}|$ users. For the affine spaces, this selection is based on difference vectors, while for the projective spaces (with an odd dimension) we introduce an algorithm to generate a spread of spreads. We further demonstrate how large populations, with $T > |\mathcal{S}_{N,R}|$, can still benefit from these user codes.

Closed formulas that express the success probability of a packet are provided for both the smaller ($T < |\mathcal{S}_{N,R}|$) and larger ($T > |\mathcal{S}_{N,R}|$) population setup. The performance gain realized by this new class of random access algorithms is demonstrated via a comparison with the random selection strategy.

Random access algorithms without feedback were first developed during the early 1980s by Massey [**?**]. In this setting, a set of M users shares a time-slotted random access channel. The idea was to assign a protocol sequence (or code) to each user (of length N) such that, irrespective of how these sequences were synchronized to one another, a guaranteed throughput could be achieved, provided that all the users make use of their protocol sequence. For instance, for $M = 2$ users the codes were $[1010]$ and $[1100]$ (in this case each packet is transmitted twice per period). The capacity of such a channel turned out to be $1/e$ for M large—even when the users are not slot synchronized—and a protocol sequence generator that realized this throughput was developed [**?**].

The problem of having only T users with data in a population of M was also considered [**?**], where it is unknown which users are active. Again, the channel capacity was shown to equal $1/e$.

The problem addressed in this paper is of a somewhat different nature, in the sense that we do not require that all packets are transmitted successfully with probability one. We allow for a loss tolerance, say of at most 1%, as delay critical data in communication networks can typically cope with some degree of packet loss. The no feedback scenario applies in networks where the round-trip time of the random access channel is so large that any feedback received is useless, as the maximum delay tolerated by this type of data has already expired. Typical networks that suffer such feedback delays are satellite networks (e.g., DVB-RCS networks [**?**]).

Assume that the maximum allowed delay is denoted as N time slots. This implies that we wish to transmit a new packet within the next N time slots. The performance of immediately transmitting this new packet a single time is rather low. On can improve this scheme by transmitting the packet R times in the next N time slots. The most natural way to do this, is by selecting these R slots at random [**?**, **?**]. However, as the user population is finite, one may expect further performance gains by assigning a user code (or pattern) to each of the users that dictates in which R of the next N slots a transmission should occur. The design of such user codes is one of the main contributions of this paper. Recently, it was shown that the random selection scheme can also be improved significantly by implementing an iterative Interference Cancellation (IC) approach [**?**]. This IC approach can potentially be used to further improve our user code based algorithms.

The user codes are designed such that any two user codes share at most one slot. These codes correspond to binary constant weight codes with weight R and minimum distance $2(R-1)$. A simple upper bound $U_{N,R}$ for the maximum number of such user codes is presented and for specific values of N we rely on finite geometries to generate sets $\mathcal{S}_{N,R}$ of user codes with $|\mathcal{S}_{N,R}| = U_{N,R}$. To the best of our knowledge, it is the first time that finite geometry is used in the design of random access algorithms. These finite geometries will contain N points (slots) and each line in such a geometry holds exactly R points. As any two lines can only have a single point in common, lines naturally correspond to user codes. Moreover, the number of lines in the space matches $U_{N,R}$ exactly. This implies that we can generate a maximum set of user codes $\mathcal{S}_{N,R}$ by developing a (fast) algorithm that lists all the lines in the finite geometry under consideration.

Provided that we have a population of $|\mathcal{S}_{N,R}|$ users, we present a closed formula for the success probability of a packet. A packet is successful if any of its R transmission attempts succeeds (meaning, none of the other users used the same slot). Next, we address the problem if the size of user population T is smaller than $|\mathcal{S}_{N,R}|$. Clearly, one could simply select T user codes, however, some choices result in a better performance than others. A selection method that will result in a better performance for smaller populations is presented. The idea is to partition the set $\mathcal{S}_{N,R}$ such that any two user codes in the same partition do not share any slot. To select the T user codes, we make use of the codes in the first partition, followed by the codes in the second partition and so on.

In affine geometries these partitions correspond to lines with the same difference vector, as a result, the partitioning is easy to achieve. For the finite projective geometries, such a partitioning corresponds to generating a spread of spreads, which is only possible if the dimension n of the space is odd. An algorithm based on a technique developed in [] for $n = 3$ is discussed. As shall become apparent further on $n = 3$ covers most of the practical cases, as we are mainly interested in $R = 3$ to 5 (as $R = 1$ and 2 are trivial) and N below 100. A closed formula that expresses the success probability for a population of T users, where the user codes are selected according to the partitioning, is also presented.

In principle, the use of user codes imposes a strict bound on the user population. For larger populations, codes can be reused by some terminals, or those extra users can simply perform random selection. We will derive the (approximated) success probability for both possibilities, indicating that the second option offers the best performance for large populations (i.e., $T > |\mathcal{S}_{N,R}|$).

Finally, we also demonstrate the effectiveness of these novel random access algorithms by comparing them with the random selection approach for a wide range of N values (and R between 1 and 5).

1 A user code based random access algorithm

Consider a random access channel without feedback shared by a set of users. Packets generated by a user can withstand a maximum delay of N time slots. When two or more packets are transmitted simultaneously, all transmissions in this slot are assumed to be lost. A user can typically cope with a small loss rate, e.g., 1%.

Instead of transmitting a packet just once, each user transmits a packet R times within the next N time slots. The most natural way is to select R slots out of the next N slots in a random manner. It is well known that such a repeated randomized transmission can significantly reduce the packet loss rate, compared to a single transmission [**?**, **?**]. Notice, a packet is only lost if all R instances were involved in a simultaneous transmission.

Some formulas to assess the performance of this random approach are giving in the Appendix. Typically, when analyzing such a scheme, the time slots are assumed to be *grouped* in sets of N slots. When a user generates a new packet it will attempt its R transmissions in the next group of N slots. Grouping also seems necessary when we wish to rely on user codes. After all, when we design a set of user codes, such that they share at most one slot, it is essential that the sequences of N slots are synchronized among one another. For frame-based networks, with a (small) contention window per frame, this approach can be naturally applied. Consider F contention slots per frame, then N can be chosen to equal F (or as a multiple thereof). Grouping occurs naturally as the contention based data has to wait for the next set of F contention slots. On the other hand, grouping may violate the delay constraints as packet delays up to $2N - 1$ slots are now possible.

One solution to meet the packet deadline of N slots, is to group the slots in sets of $N/2$ slots and to select R slots out of the next $N/2$ slots. However, decreasing the transmission window by a factor two is not beneficial for the overall success probability, therefore, we propose a solution that allows us to stick with a window of N slots. Suppose that a user code is represented by a bit vector of size n and weight R, where bit number i is set if the user selects slot i as one of his R slots. The idea is that when a packet is ready for transmission at the end of the k-th time slot of a group of N slots, it will change its original user code by moving the first k bits of to the back of its user code. This *shifted* bit vector is subsequently used for the packet transmission and may commence in the very next slot (that is, slot $k + 1$). In this way, we guarantee that any two packets still interfere in at most one slot, even though the transmissions are no longer synchronized to the start of a group.

Finally, as with most random access algorithms, it is assumed that per user there is at most one packet ready for transmission at any given time. Hence, packets from the same user will never compete with each other.

2 Generating $U_{N,R}$ user codes

In this section we explain how to generate a maximum set $S_{N,R}$ of user codes for particular values of N using a finite geometry. We start by repeating a well known argument that states that there can be at most

$$U_{N,R} = \left\lfloor \frac{N}{R} \left\lfloor \frac{N-1}{R-1} \right\rfloor \right\rfloor$$

user codes, such that each two codes have at most one slot in common. Let $i \in \{1, \ldots, N\}$, then there are at most $\lfloor (N-1)/(R-1) \rfloor$ codes that make use of slot i. Also, as each code contains R slots, we may conclude that there can be at most $U_{N,R}$ user codes. It has been proven that this upper bound can be attained in many particular cases [**?**, **?**].

For $R = 2$, all the 2-element subsets of $\{1, \ldots, N\}$ form a set $S_{N,R}$ with $|S_{N,R}| = U_{N,R}$, so we may restrict ourselves to the case where $R \geq 3$. We will show that finite affine and projective geometries allow us to generate a set of $U_{N,R}$ user codes for various choices of N and R. In the last part of this section we also indicate how to generate user codes for N and R values that do not fit within the finite geometry setup.

2.1 Finite affine spaces

A finite affine space $AG(n, q)$ of dimension $n \geq 2$ over a finite field $K = GF(q)$, with $q = p^k$ for some p prime and $k \geq 1$, consists of a set V of $N = q^n$ points having coordinates of the form $(x_1, \ldots, x_n)$ with $x_i \in K$, for $i = 1, \ldots, n$. Through every two points in such a space, there exists exactly one line and every line holds exactly $R = q$ points. This implies that there are exactly $C = N(N-1)/R(R-1) = q^{n-1}(q^n-1)/(q-1)$ different lines in $AG(n, q)$. Further, every two lines intersect in at most one point.

Given these properties, we can make use of such geometries to generate $U_{N,R}$ user codes, provided that N and R can be written as $N = R^n$ and $R = q$, for some $n \geq 2$, $q = p^k$ for p prime and $k \geq 1$. Table 1 lists some of the N values for $R = 3, 4$ and 5.

	$n=2$	$n=3$	$n=4$
$R=3$	9(24)	27(117)	81(1080)
$R=4$	16(20)	64(336)	256(5440)
$R=5$	25(30)	125(775)	625(19500)

Table 1. Feasible N and (C) values for $R = 3, 4$ and 5 using affine spaces

To generate the user codes, it suffices to list the C lines of the affine space $AG(n, q)$. Each line is characterized by the R points that it holds. We can produce this list by iterating over all 2-element subsets of V and determining the coordinates of the remaining $R - 2 = q - 2$ points. Given two points $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$, we first compute the difference vector $b-a$ as $(b_1-a_1, \ldots, b_n-a_n)$, where $a_i, b_i \in K$ and $b_i - a_i \in K$ is determined via the subtraction operation of the finite field K. The $R = q$ points of the line through a and b are then given by $\{a + c*(b-a) | c \in K\}$, where $*$ denotes the product in K. Clearly, setting $c = 0$ and 1 simply reproduces the points a and b.

If we iterate this simple procedure over all 2-element subsets of V, we would produce $R(R-1)/2$ instances of each line. However, this iteration can be adapted easily such that every line is produced just once.

2.2 Finite projective spaces

A finite projective space $PG(n, q)$ of dimension $n \geq 2$ over a finite field $K = GF(q)$, with $q = p^k$ for some p prime and $k \geq 1$, consists of a set V of $N = (q^{n+1} - 1)/(q-1)$ points. The projective space $PG(n, q)$ can be constructed from the affine space $AG(n, q)$ by adding *points at infinity*. More specifically, for each difference vector $b - a$ we add a single point to $AG(n, q)$ and we also add this point to the q^{n-1} lines that correspond to this difference vector, such that each line now carries $q + 1$ points. In this way, a total of $(q^n - 1)/(q - 1)$ points are added. We also add lines at infinity such that these points and lines at infinity also form a projective space $PG(n - 1, q)$ on their own.

Using this construction we can also add coordinates to the projective plane by assigning $(1, x_1, \ldots, x_n)$ to the affine point with coordinate $(x_1, \ldots, x_n)$ and $(0, y_1, \ldots, y_n)$ to the point at infinity that corresponded to the difference vector $(y_1, \ldots, y_n)$. As in the affine case, there exists exactly one line through every two points, but now every line holds exactly $R = q + 1$ points. Hence, there are exactly $C = N(N-1)/R(R-1)$ different lines in $PG(n, q)$. Further, every two lines also intersect in at most one point.

Projective geometries thus allow us to generate $U_{N,R}$ user codes, provided that N and R can be written as $N = (q^{n+1}-1)/(q-1)$ and $R = q+1$, for some $n \geq 2$, $q = p^k$ for p prime and $k \geq 1$. Table 2 lists some of the N values for $R = 3, 4$ and 5.

As with the affine space, we can generate a maximum set of user codes by listing all the lines in the projective space $PG(n, q)$. We first list all the lines in $AG(n, q)$ and add a 1 to the front of each affine coordinate to get the corresponding projective coordinate. This gives us the q coordi-

	$n=2$	$n=3$	$n=4$
$R=3$	7(7)	15(35)	31(155)
$R=4$	13(13)	40(130)	121(1210)
$R=5$	21(21)	85(357)	341(5797)

Table 2. Feasible N and (C) values for $R = 3, 4$ and 5 using projective spaces

nates of the affine points on these lines. By adding the point $(0, y_1, \ldots, y_n)$ to each line, where $y = (y_1, \ldots, y_n)$ is its difference vector, we end up with all the lines that contain affine points. The lines that are not listed thus far, are all located at infinity. As the space at infinity forms a projective space $PG(n-1, q)$, we simply use this procedure in a recursive manner until $n = 1$ (where $PG(1, q)$ has $q + 1$ points all located on a single line).

2.3 User codes for all N and R values

Even though the finite geometry approach taken in the previous two subsections is only applicable for specific choices of N and R, we can also use this approach to generate a set $S_{N,R}$ of user codes for any N and R value. However, these sets will, in general, no longer be of maximum cardinality. To generate a set for an arbitrary N and R value, we can start by selecting the smallest $\bar{N}$ value that corresponds to some space $AG(n, R)$ such that $\bar{N} \geq N$. Next, we generate the user codes of this particular space. Finally, we remove all the user codes from the set $S_{\bar{N},R}$ that make use of one of the slots in $\{N + 1, \ldots, \bar{N}\}$. This procedure can be also be used in the projective case, but with some space $PG(n, R - 1)$. Depending on the value of N, either the affine or the projective case will produce the largest set. Figure 1 depicts the cardinality of this largest set for $R = 3, 4$ and 5. In the remainder of this paper we will only focus on the N and R combinations that correspond to a finite geometric space.

3 Performance in a $U_{N,R}$ user population

3.1 Analysis

In this section we demonstrate that, using the highly symmetric structure of the spaces $AG(n, q)$ and $PG(n, q)$, we can quite easily establish an expression for the success probability of an arbitrary packet. To do so, we assume that we have a user population of $C = U_{N,R} = N(N-1)/R(R-1)$ users and each user is assigned a single user code that is used to transmit a packet. We will address the problem of having a population with fewer (or

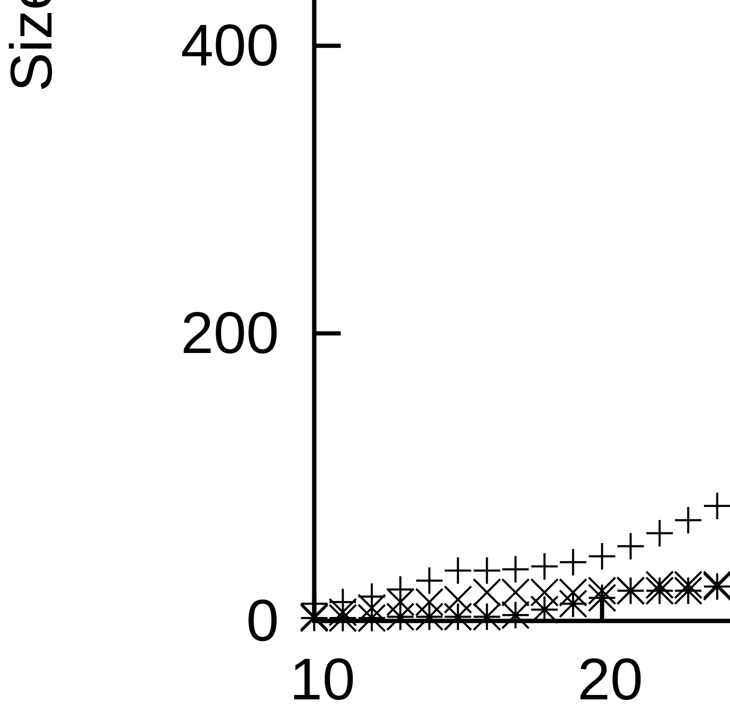

Figure 1. Largest sets for $R = 3, 4$ and 5, and all values of N between 10 and 100, found by the elimination procedure.

more) than C users in Section 4 (or Section 6). For the performance analysis we assume that users wait until the start of the next group of N slots. The same assumption was made when analyzing the algorithm that selects R slots in a random manner. Furthermore, for the user code based algorithm, we will show by simulation that the results obtained from the grouping scenario nearly coincide with those using the shifted bit vectors discussed at the end of Section 1.

For the analysis of our user code based algorithm, it is important to notice that any point that is part of some line ℓ in either $AG(n, q)$ or $PG(n, q)$ will also be part of exactly $S = (N - R)/(R - 1)$ other lines, because any two points characterize a line and all lines hold R points. Further, every line intersects ℓ in at most one point, making the sets of lines passing through different points on ℓ disjoint.

Assume $U \leq C$ users each transmit R times according to their user code and we have a total population of C users. Further let us tag the R transmission attempts by a particular user. To know the probability that the tagged user is successful, it suffices to compute the probability that at least one point on a particular line ℓ is not intersected by one of the other $U - 1$ lines. The probability that a specific set of i points on ℓ is not intersected by any of the other $U - 1$ lines equals

$$\frac{\binom{(C-1)-iS}{U-1}}{\binom{C-1}{U-1}},$$

because there are C lines in total (including line ℓ) and

iS of them intersect with ℓ. To get the success probability $p_{suc}(U)$ of a tagged user, we can use the inclusion-exclusion principle such that we do not count too many successes, as follows:

$$p_{suc}(U) = \sum_{i=1}^{\min(R,\lfloor (C-U)/S \rfloor)} (-1)^{i+1} \binom{R}{i} \frac{\binom{(C-1)-iS}{U-1}}{\binom{C-1}{U-1}}.$$

We further assume that each user generates packets according to a Poisson process with rate λ. If multiple packets are generated by a single user in a length N interval, they are combined into one message that is transmitted R times in the next interval. Thus, with probability $p = 1 - e^{-\lambda N}$, a user will participate in a length N interval. The total load on the contention channel therefore matches $\rho = pC/N$. Hence, the overall success probability under Poisson arrivals matches

$$p_{suc} = \sum_{U=1}^{C} \frac{U}{\rho N} \binom{C}{U} p^U (1-p)^{C-U} p_{suc}(U), \qquad (1)$$

where the $\frac{U}{\rho N}$ deals with the fact that a tagged user is more likely to be part of a larger group of users.

3.2 Numerical Results

Figures 3, 4 and 5 illustrate the error probability for arrivals following a Poisson process, as defined in previous section, for the cases where $R = 3, 4$ and 5 respectively. We see that the use of user codes reduces the error probability significantly compared to random selection, where the reduction becomes more pronounced as the population size and the load diminishes. Also notice that, given a fixed load ρ, increased user populations cause more packet losses for the user code based algorithm, as opposed to the random selection that seems to benefit from more users.

A comparison with a time driven simulation is provided. As the closed formulas are exact for the grouped setup, there was a perfect agreement with the simulated grouped scenario. Figures 3 to 5 also depict the simulated scenario without grouping, where we use the shifted bit vector approach for the user codes. A remarkable observation can be made with respect to the grouping mechanism. For the random selection, grouping (or synchronization) has a negative influence on the packet loss. This is in contrast with many other random access schemes (e.g., slotted vs. unslotted ALOHA), because here a packet is saved if one of its R instances survives transmission, whereas in a classic setting losing a part of the transmission corrupts the entire transmission attempt. This synchronization penalty is however *not* observed for the user code based results. So it seems that our user codes do not suffer a grouping penalty, which is very useful for frame-based networks.

We must remark that to match the arrival pattern of the theoretical grouped analysis and the simulated case without grouping, a minor modification to the Poisson process is required, as Figure 2 illustrates. This modification is needed as multiple arrivals that occur in the same group where merged into one arrival in the grouping setup. Hence, in order to consider exactly the same arrivals in both scenarios, some arrivals are ignored, while others are slightly shifted to avoid contention between two packet of the same user. We refer to Section 7.3 for more details.

Figure 2. Illustration of the arrival process of a single user, as used by the simulation. Poisson arrivals 3 and 5 are not considered in the simulation, since they are both the second arrival within the same group. Although arrival 2 occurs shortly after the first one, it is used, but assumed to arrive exactly N slots after the first one. Thus, both simulations consider the same arrivals as used in the theoretical analysis.

Figure 3. Performance results in a $U_{N,R}$ user population. In this case $R = 3$, which results in $N = 27$, $C = 117$ using affine spaces, and in $N = 15$, $C = 35$ using projective spaces.

Figure 4. Performance results in a $U_{N,R}$ user population. In this case $R = 4$, which results in $N = 64$, $C = 336$ using affine spaces, and in $N = 40$, $C = 130$ using projective spaces.

4 Selecting T of the $U_{N,R}$ user codes

In this section we consider a population of $T < U_{N,R}$ users and address the problem of selecting T user codes from the set of $U_{N,R}$. We could select T codes at random, however, if we are *unlucky* in our choice, the performance might reduce, even though we have fewer users. To remedy this problem, we propose a method that orders the $U_{N,R}$ users codes such that a population of T users will make use of the first T user codes. Although, one easily shows that this choice does not maximize p_{suc} for many T values, we will demonstrate that it significantly improves the average performance of a random selection of T codes. The

Figure 5. Performance results in a $U_{N,R}$ user population. In this case $R = 5$, which results in $N = 125$, $C = 775$ using affine spaces, and in $N = 85$, $C = 357$ using projective spaces.

advantage of this approach is also that we can simply add new users (and their codes) at runtime without the need to change the user codes of the existing population, which is in general not the case for an optimal selection procedure. Finally, this order also allows us to establish a closed expression for the success probability p_{suc}.

Our design goal is to partition the set of all lines $\mathcal{S}_{N,R}$ in $AG(n,q)$ and $PG(n,q)$ such that none of the lines part of the same partition intersect. Further, within every partition, we also demand that every point lies on some line. We start with $AG(n,q)$.

4.1 Finite affine spaces

In the affine case, it is not hard to devise such a partitioning by making use of difference vectors. In $AG(n,q)$ there are $C = q^{n-1}(q^n-1)/(q-1)$ lines and to each of the $(q^n-1)/(q-1)$ difference vectors correspond exactly $L = N/R = q^{n-1}$ lines. By definition, lines with the same difference vector cannot intersect, so by associating a partition to each difference vector, we end up with the required partitioning of $\mathcal{S}_{N,R}$.

To get a total ordering of $\mathcal{S}_{N,R}$, we list the lines partition by partition. From the results presented in Section 5 one deduces that the order of the partitions is irrelevant. The order of the lines within a partition does have a minor impact on the performance (when T is not a multiple of L), but is chosen at random.

4.2 Finite projective spaces

5 Performance in a $T < U_{N,R}$ user population

5.1 Analysis

In this section we derive a new expression for p_{suc} taking into account that we have only $T < U_{N,R} = C$ users. To obtain an elegant closed expression, we restrict ourselves to the case where T is a multiple of L, the number of lines in a single partition. For other values of T, we can get a useful approximation by considering the closest multiple of L. Suppose $T = kL$, for $1 \leq k < C/L$. Due to the design of the selection algorithm, each point in our space is intersected by exactly k lines. Thus, if we tag a user, each point on its line ℓ will intersect with exactly $k-1$ other lines. Also, a set of lines going through one point of ℓ will be disjoint with a set going through any other point on ℓ. Hence, analogue to Section 3.1, where S is now replaced by $k-1$ and C by T, we find

$$p_{suc}(U) = \sum_{i=1}^{\min(R,\lfloor (T-U)/(k-1)\rfloor)} (-1)^{i+1} \binom{R}{i} \frac{\binom{(T-1)-i(k-1)}{U-1}}{\binom{T-1}{U-1}}.$$

For $k = 1$, this expression reduces to $p_{suc}(U) = 1$. Next, we can use formula (1) to determine the success probability under Poisson arrivals.

5.2 Numerical Results

Figure 6 illustrates the loss probability for the case where $R = 4$, for other scenarios we refer to Section 7.3. We first observe that the loss rate reduces as the population size diminishes, where the loss rate drops to zero when the number of users $T = L$. Furthermore, the gain obtained by having a size $T < C$ user population is much more pronounced for the code based algorithm, when compared to the random selection. Finally, we also note that the grouping or synchronization penalty of the random selection algorithm remains absent for the user code based scheme for all $T < C$.

Figure 6. Performance results in a $T \leq U_{N,R}$ user population, with $\rho = 0.1$ and $R = 4$. Note that for $N = 64$, $C = 336$ and for $N = 40$, $C = 130$.

6 Dealing with more than $U_{N,R}$ users

Eventhough the designed user codes are mostly effective when the user population T is bounded by $U_{N,R}$, we will demonstrate that these codes still have their merits even when T exceeds $U_{N,R}$. We discuss two simple possibilities for supporting larger populations that show how to exploit the $U_{N,R}$ user codes.

A first approach is to reuse existing codes for the additional users. Hence, code i is used by the set of users with ids $\{kU_{N,R} + i | k \geq 0\}$. The main disadvantage of this approach is that as soon as two users with the same code become active, they will eliminate all of the R transmissions of one another. Code reuse therefore seems mostly useful when T is only marginally larger than $U_{N,R}$.

A second, better alternative is to assign codes to the first $U_{N,R}$ users and to let the remaining $T - U_{N,R}$ perform a random selection. The main disadvantage of such an approach is that some unfairness between *coded* and *random* can be expected. We will comment more on this unfairness issue in Section 7.3.

We finally note that it might be best to deal with populations of size $T > U_{N,R}$ by designing a different, larger set of user codes, where we allow two codes to share more than one slot. We plan to address this possibility in some future work.

7 Performance in a $T > U_{N,R}$ user population

7.1 Analysis of code reuse

Consider a population of $T > U_{N,R} = C$ users, where user j uses code $j \ mod \, U_{N,R}$. Now each code is used by at least $\alpha = \lfloor T/C \rfloor$ users, while some codes are used as many as $\alpha + 1$ times. The probability that a given user uses a code which is used α times, is given by:

$$p_m = \frac{\alpha\left((\alpha+1)C - T\right)}{T}.$$

This allows us to establish the success probability, given U users are active in an interval of N slots, where we will distinguish between the case where the tagged user code is used α or $\alpha + 1$ times. For simplicity, we assume that T is of the form $T = kL$, meaning we distribute the C codes α times among the first αC users and the remaining $kL - \alpha C$ users are giving the codes in the first $k - \alpha C/L$ partitions as explained in Section 5. By noticing that each point lines on exactly k lines (of which are identical due to the reuse) and by applying similar arguments as before, one establishes

$$p_{suc}^{(d)}(U) = p_m \sum_{i=1}^{\min(R, \lfloor \frac{1+T-U-\alpha}{k-\alpha} \rfloor)} (-1)^{i+1} \binom{R}{i} \frac{\binom{(T-\alpha)-i(k-\alpha)}{U-1}}{\binom{T-1}{U-1}}$$
$$+ (1-p_m) \sum_{i=1}^{\min(R, \lfloor \frac{T-U-\alpha}{k-1-\alpha} \rfloor)} (-1)^{i+1} \binom{R}{i} \frac{\binom{(T-1-\alpha)-i(k-1-\alpha)}{U-1}}{\binom{T-1}{U-1}}.$$

7.2 Analysis of user codes combined with random selection

Consider the same population of $T > U_{N,R} = C$ users, where C users make use of a code, whereas the remaining $T - C$ users transmit at random. Assume that $U = U^{(c)} + U^{(r)}$ users are active in an interval of length N. With probability

$$p(U^{(c)}, U^{(r)}) = \frac{\binom{C}{U^{(c)}}\binom{T-C}{U^{(r)}}}{\binom{T}{U^{(c)}+U^{(r)}}},$$

$U^{(c)}$ of them have a user code and $U^{(r)}$ do not. Given that $U^{(c)}$ users have a code and assuming the tagged user has a code, we find that the probability that the tagged user is successful is given by

$$p_{suc}^{(c)}(U^{(c)}, U^{(r)}) =$$
$$\sum_{i=1}^{m} (-1)^{i+1} \binom{R}{i} \frac{\binom{(C-1)-iS}{U^{(c)}-1}}{\binom{C-1}{U^{(c)}-1}} \left(\frac{\binom{N-i}{R}}{\binom{N}{R}} \right)^{U^{(r)}},$$

where $m = \min(R, N-R, \lfloor (C - U^{(c)})/S \rfloor)$.

Deriving a closed expression for the success probability for remaining $U^{(r)}$ users that transmit in a random manner is more problematic as a random selection can intersect with user codes in a multitude of manners. However, for a tagged user without a code, it turns out that we can make an excellent approximation by assuming that all $U - 1$ other users (including the $U^{(c)}$ that have a code) appear to chose their slots randomly. Hence, from the perspective of a random user, it seems that everyone is transmitting at random. Numerical evidence of the close resemblence between the actual simulated success probability and this approximation is given in Section 7.3. Given this approximation, the resulting success probability of an arbitrary active user becomes:

$$p_{suc}^{(c)}(U) = \sum_{U^{(c)}=U-U^{(r)}=0}^{min(U,C)} \Big(\frac{p(U^{(c)}, U^{(r)})}{U} \cdot$$
$$(U^{(c)} p_{suc}^{(c)}(U^{(c)}, U^{(r)}) + U^{(p)} p_{suc}^{(r)}(U) \Big),$$

with $p_{suc}^{(r)}(U)$ the success probability for U users performing a random selection, as defined in the Appendix. To obtain the success probabilities $p_{suc}^{(c)}$ and $p_{suc}^{(d)}$ for Poisson arrivals, we refer to Equation (1).

7.3 Numerical results

Figures 7, 8 and 9 show the results for various user populations. We compare both the reuse of user codes and the combination of user codes with random selection against completely random selection for a load of 10 percent. As expected, the combination of user codes with random selection ourperforms the other two setups for all scenarios, while the reuse of codes becomes inferior to a standard random selection when the population becomes large enough.

The simulation results for the grouped scenario were matched perfectly by the closed formulas for the random selection and reused codes. For the combined setup, we see that the approximation formula suggested for the *random* users turns out to be very effective. In Section 3.2 we noticed that there is a grouping penalty associated with the random selection, while the user code scheme did not experience such a penalty for $T \leq U_{N,R}$. When the population T becomes larger than $U_{N,R}$, this penalty reappears for both the reuse scenario and the combined scheme. Intuitively, we can expect a gain when two users sharing the same code become desynchronized, meaning the shifted bit vectors will prevail.

The formula for $p_{suc}^{(c)}(U)$, combined with the numerical results, also suggests that the combination of user codes with random selection offers a higher success probability to users with a user code; the loss probability of the remaining users corresponds to the standard random selection scenario. This clearly introduces some unfairness. However, the alternative of using no user codes only offers a disadvantage to the *coded* users and no advantages for the *random* users, so there is no harm in introducing codes in part of the population.

Figure 7. Performance results in a $T > U_{N,R}$ user population, with $\rho = 0.1$ for $R = 3$. The results are for $N = 15$, $C = 35$ and $N = 27$, $C = 117$ respectively.

Figure 8. Performance results in a $T > U_{N,R}$ user population, with $\rho = 0.1$ for $R = 4$. The results are for $N = 40$, $C = 130$ and $N = 64$, $C = 336$ respectively.

Figure 9. Performance results in a $T > U_{N,R}$ user population, with $\rho = 0.1$ for $R = 5$. The results are for$N = 85$, $C = 357$ and $N = 125$, $C = 775$ respectively.

Appendix

A Performance of Random Selection

This section indicates how to assess the success probability $p_{suc}^{(r)}$ of the random selection algorithm for a population of C users. Slots are grouped into sets of N slots and a user who generates $k \geq 1$ packets in a set of N slots, will transmit R instances of a single packet (that contains the combined information of the k packets) by selecting R of the N time slots within the next group of N slots.

Assume that U users attempt to transmit their packet during an interval of N time slots. The probability that a specific set of i slots, selected by a tagged user, remains unused by the remaining $U-1$ users equals

$$\left(\frac{\binom{N-i}{R}}{\binom{N}{R}}\right)^{U-1}.$$

Using an inclusion-exclusion argument, we obtain an expression for $p_{suc}^{(r)}(U)$, the probability that a tagged user is successful given that $U-1$ other users where active

$$p_{suc}^{(r)}(U) = \sum_{i=1}^{\min(R,N-R)} (-1)^{i+1}\binom{R}{i}\left(\frac{\binom{N-i}{R}}{\binom{N}{R}}\right)^{U-1}.$$

By replacing $p_{suc}(U)$ with $p_{suc}^{(r)}(U)$ in Equation (1), we obtain the success probability $p_{suc}^{(r)}$ for the random selection algorithm under Poisson arrivals.

B Packings in Projective Spaces

Theoretical Background First we review some basic facts about finite fields. Every finite field has a generator for the multiplication, i.e. there exists an $\alpha \in GF(q^n)$ such that every nonzero element can be expressed as a power of α. If $m|n$ we can embed $GF(q^m)$ in $GF(q^n)$ as the set containing the zero and all powers of $\alpha^{\frac{q^n-1}{q^m-1}}$. We can view $GF(q^n)$ as vector space over $GF(q^m)$ and this vector space has a basis consisting of the elements $\alpha^0, \ldots, \alpha^{\frac{n}{m}-1}$. So taking $m = 1$ we can express every nonzero element of $GF(q^n)$ in two ways: as a power of the generator or as a $GF(q)$-linear combination of the first n powers of the generator. The first representation ideal to compute products while the second is used to compute sums.

The points in the projective space $PG(n, q)$ can be seen as elements in $GF(q^{n+1})$ by mapping $(x_0, \ldots x_n)$ to $x_0 + x_1\alpha + \ldots x_n\alpha^n$. These elements are determined up to multiplication with a nonzero element of $GF(q)$. If $\xi \in GF(q^{n+1})$ we will denote the corresponding element of $PG(n, q)$ by $[\xi]$. Using the exponential representation we can see that every point is of the form $[\alpha^k]$ and two powers k_1, k_2 give the same point if $k_1 = k_2 \mod \frac{q^{n+1}-1}{q-1}$.

A set of lines $\mathcal{S}$ in the projective space $PG(n, q)$ is called a spread if every point in $PG(n, q)$ is contained in exactly one line of $\mathcal{S}$. The existence of speads implies that n must be odd because the number of points in $PG(n, q)$ must divide the number of points on a line.

$$\#\mathcal{S} = \frac{\#PG(n, q)}{q+1} = \frac{q^{n+1}-1}{q^2-1}$$

If n is odd one can always construct a spread using finite fields. We briefly describe this construction as outlined in [?]. For every $0 \leq i < \frac{q^{n+1}-1}{q^2-1}$ we define the line

$$\ell_i = \{[\alpha^{i+\nu\frac{q^{n+1}-1}{q^2-1}}] | 0 \leq \nu < q+1\}$$

It can be checked that this is indeed a line and $\ell_i \cap \ell_j = \emptyset$ if $i \neq j$. So $\mathcal{S} = \{\ell_i | 0 \leq i \leq \frac{q^{n+1}-1}{q^2-1}\}$ is a spread and it is called the standard spread.

Example. If $q = 2$ and $n = 3$ we have to work over $GF(2^4)$. As generator we will use the α that satisfies $\alpha^4 = \alpha + 1$. This generates the following spread

$$\begin{aligned}\ell_0 &= \{[1], [\alpha^5], [\alpha^{10}]\} = \{[1], [\alpha + \alpha^2], [1 + \alpha + \alpha^2]\} \\ &= \{(1,0,0,0), (0,1,1,0), (1,1,1,0)\} \\ \ell_1 &= \{[\alpha], [\alpha^6], [\alpha^{11}]\} = \{[1], [\alpha^2 + \alpha^3], [\alpha + \alpha^2 + \alpha^3]\} \\ &= \{(0,1,0,0), (0,0,1,1), (0,1,1,1)\} \\ &\cdots\end{aligned}$$

which has 5 lines with 3 points covering all 15 points of the projective space.

A set of spreads $\mathcal{P}$ of the projective space $PG(n,q)$ is called a packing if every line in $PG(n,q)$ is contained in exactly one spread of $\mathcal{P}$. The existence of packings is proven if $n = 3$ and for this case we will give the construction of a packing.

If $q = 2$ one can consider the following automorphism of $PG(3,2)$

$$\phi : (x_0, \ldots, x_3) \to A \cdot (x_0, \ldots, x_3) \text{ with } A = \begin{pmatrix} 1&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \\ 0&1&1&0 \end{pmatrix}.$$

If we start with the standard spread from the example and apply the automorphism to all the points we get a new spread $\mathcal{S}^\phi$. As the matrix A has order 7, we get 7 different spreads: $\mathcal{S}, \mathcal{S}^\phi, \mathcal{S}^{\phi^2}, \ldots, \mathcal{S}^{\phi^6}$. One can check that these 7 spreads form a packing. Note that this construction depends highly on the form of A and it cannot be generalized to other q.

If $q > 2$ we will use a construction by Beutelspracher [?] that is a little more involved. Let β be the generator of $GF(q^2)$ so every element in this field can be written as $x + y\beta$ with $x, y \in GF(q)$. We embed $PG(3,q)$ inside $PG(3,q^2)$ using the map

$$(x_0, \ldots, x_3) \to (x_0 + 0\beta, \ldots, x_3 + 0\beta)$$

A line ℓ in $PG(3,q)$ will induce a line $\tilde{\ell}$ in $PG(3,q^2)$, which contains apart from the points in ℓ an extra $q^2 - q$ points.

Take a line $\ell \subset PG(3,q)$ and chose 2 points $p_0, p_1 \in \tilde{\ell}$ and a third point $p_2 \in PG(3,q^2) \setminus \tilde{\ell}$ We can define the subset $\pi_0 := \{\lambda_0 p_0 + \lambda_1 p_1 + \lambda_2 p_2 | \lambda_i \in GF(q)\} \subset PG(3,q^2)$. This subset is called a Baer subplane and it is a set of $q^2 + q + 1$ points sitting inside a plane in $PG(3,q^2)$. We chose p_0, p_1, p_2 is such a way that π_0 has no points in common with $PG(3,q)$. It is not difficult to show that such a point can always be found.

There are $q^2 + q + 1$ lines in $PG(3,q^2)$ that connect pairs of points of the Baer subplane. We will denote them by g_i with $0 \leq i \leq q^2 + q$ and we suppose that $g_0 = \tilde{\ell}_0$. Every line g_i is the union of 2 sets of points: $g_i^0 := g_i \cap \pi_0$ and $g_i^1 := g_i \setminus \pi_0$. For both sets we define a set of lines in $PG(3,q)$:

$$\begin{aligned}\mathcal{G}_i^0 &:= \{\ell \subset PG(3,q) : \tilde{\ell} \cap g_i^0 \neq \emptyset\} \\ \mathcal{G}_i^1 &:= \{\ell \subset PG(3,q) : \tilde{\ell} \cap g_i^1 \neq \emptyset\}.\end{aligned}$$

To state the final result we have to introduce extra notation. If $\mathcal{L}$ is a set of lines in $PG(3,q)$ then $\mathcal{L}^\top$ is defined as the set of all lines that intersect all lines in $\mathcal{L}$.

$$\mathcal{L}^\top := \{\ell \subset PG(3,q) | \forall l \in \mathcal{L} : \ell \cap l \neq \emptyset\}.$$

Theorem[**?**] For each $1 \leq i \leq q^2 + q$ the set

$$\mathcal{S}_i := {\mathcal{G}_i^0}^\top \cap \mathcal{G}_i^1$$

is a spread. The set of remaining lines $\mathcal{S}_0 := \{\ell \subset PG(3,q) | \ell \notin \cup_{1 \leq i \leq q^2+q} \mathcal{S}_i\}$ is also a spread and together they form a packing.

Remark. Because of the special form of the $\mathcal{G}_i^0$, one can show that ${\mathcal{G}_i^0}^\top = \{l_1, l_2, l_3\}^\top$ for any subset of $\mathcal{G}_i^0$ that has 3 elements. This simplifies the calculation of ${\mathcal{G}_i^0}^\top$ a lot.

B.1 The Algorithm

In this subsection we transform the theorem to an algorithm that constructs the packing. We suppose that there exist subroutines that perform the addition and multiplication in $GF(q)$ and $GF(q^2)$. Points in $PG(n,q^2)$ are represented as arrays of $n+1$ elements of $GF(q^2)$. We need a function `STD` that brings such a point in its standard form i.e. multiplies it with a scalar such that the first nonzero element equals 1. Also let `PG(n, q)` be a function that generates a list of the points in $PG(n,q)$.

1. **Find the $p_0, \ldots, p_2$ for a good Baer Subplane:**
   Define `p0` $:= [1, \alpha, 0, 0]$ and `p2` $:= [0, 0, 1, \alpha]$. Vary `i` from 2 to $q^2 - 1$ and let `p1` $:= [1, \alpha^{i}, 0, 0]$. Check for every `j` from 1 to $q^2 - 1$ whether the elements of `STD(p0` $+ \alpha^{j} *$ `p1)` are not in $GF(q)$. If this holds for all `j` we fix `p1`.

2. **Construct the points and lines of the Baer Subplane:**
   The points are `STD(b[0] * p0 + b[1] * p1 + b[2] * p2)`, where we vary `b` in the list `PG(2, q)`. Put these points in the list `BPOINTS`. Make a list `BLINES` containing the lines in $PG(3,q^2)$ between pairs of points in `BPOINTS` except the line between `p0` and `p1`.

3. **Construct the $\mathcal{G}$'s:**
   Make a list `LINES` containing the lines in $PG(3,q)$.

Vary `i` in `LINES` and `j` in `BLINES`. Look at the intersection between `i` and `j`. If it is nonempty we break the `j`-loop. We add `i` to the list `G0[j]` if the unique element of the intersection sits in `BPOINTS` If it does not sit in `BPOINTS`, we add it to the list `G1[j]`.

4. **Construct the $\mathcal{G}_i^{0\top}$:**
   Vary `i` from 1 to `Length(BLINES)`, vary `x` in `G0[i][1]` and `y` in `G0[i][2]`. Let `L` be the line in $PG(3,q)$ between `x` and `y`. If `L` intersects `G0[i][3]`, we add `L` to `G1[i]`.

5. **Construct the final spread:**
   The `G1[i]` are all disjoint spreads. To construct the final spread we select all lines that are not contained in some `G1[i]`.

The algorithm generates the packing quite fast, but we did not optimize its efficiency in order to keep the connection with the theorem clear.