Multiple access algorithms without feedback using combinatorial designs

Multiple Access Algorithms without Feedback using Combinatorial Designs

G. T. Peeters, R. Bocklandt and B. Van Houdt

University of Antwerp - IBBT, Middelheimlaan 1, B-2020 Antwerp, Belgium

Abstract

A new class of multiple 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 invoked such that any two users will only transmit simultaneously in at most one slot, i.e., 2-$(N, R, 1)$ designs.

We argue that in most cases, the set of user codes can be generated easily using cyclic designs and provide a method to select T user codes from the set of user codes $S_{N,R}$ in case the user population consists of $T < |S_{N,R}|$ users. We further demonstrate how larger populations, with $T > |S_{N,R}|$, can still benefit from these user codes in two different manners. Closed formulas that express the success probability of a packet are provided for all population setups.

Finally, a comparison with the random selection strategy demonstrates the performance gain realized by the new multiple access algorithms and some engineering rules to optimize the performance are provided.

Multiple access algorithms without feedback were first developed during the early 1980s by Massey [1]. 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 [2].

The problem of having only T users with data in a population of M was also considered [3], where it is unknown which users are active. Again, the channel capacity was shown to equal $1/e$.

A number of wireless multiple access algorithms have been introduced [4], [5], [6] that are capable of resolving a conflict of K users through source separation techniques, without the need for any feedback

during the resolution period. More specifically, each of the K users retransmits its packet in every subsequent slot as long as the base station does not announce the end of the current conflict resolution period. The NDMA algorithm of [4] resolves the conflict in K slots, by detecting the conflict multiplicity during the very first transmission via orthogonal identification codes. Next, it waits for another K-1 retransmissions of the same K packets and retrieves the packets from the K transmissions using source separation techniques. The limiting use of the orthogonal codes for larger populations is avoided in [5], by a tight phase control of the retransmitted packets, such that the channel-mixing matrix has a Vandermonde structure (given a static channel during the resolution period). Using this structure, the conflict multiplicity is detected after K+1 transmissions (or more, in the presence of noise) and the packets are resolved by employing a parallel factor analysis. Finally, in [6] the multiplicity detection method of [5] is further improved by eliminating the need for a tight phase control and allowing a quasi-static channel (at the price of using K+2 slots). The collision resolution is accomplished through an independent component analysis. Each of these solutions however still requires feedback from the base station at the end of each conflict resolution period to halt the retransmission process and to announce the start of a new conflict resolution period, while no such feedback is present for the problem considered in this paper.

Furthermore, the problem addressed in this paper is also 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 caused by contention conflicts, e.g., of at most $\epsilon = 1\%$, 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. For example, DVB-RCS networks [7] are deployed with the goal of supporting a wide range of customers, providing both trunking services for connecting proprietary networks as well as setting up a return link for home networking end-users in two-way satellite networks. As such, different population sizes can be supported, ranging from a few tens up to several hundred users respectively. To allocate bandwidth, the DVB-RCS standard provides not only mid-term and long-term reservation schemes (for example, Volume-Based Dynamic Capacity (VDBC) and Constant Rate Assignment (CRA)), but also contention access slots to reduce the delay to set up a connection, for example for Voice over IP applications. As contention channels are typically suited for low loads, bandwidth reservation schemes may take over after exchanging some initial data via the contention channel, resulting

in a good trade off between low delay and bandwidth efficiency.

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. One 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 [8], [9]. 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. Recently, it was shown that the random selection scheme can also be improved significantly by implementing an iterative Interference Cancellation (IC) approach [10]. This IC approach can potentially be used to further improve our user code based algorithms. In the context of satellite networks, such as DVB-RCS networks, centralized code assignment can be easily implemented. In these networks, a connection is initiated using a log on procedure in which a terminal receives the network parameters (for example frequency and timing information), using a forward link (i.e., DVB-S2). At the same time, an identification ID is assigned, which can also be used to designate a user code.

The user codes considered are 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 $2(R-1)$. Moreover, for any 2 slots, there should also be a user code using both slots. Hence, we are looking for sets of user codes such that every two slots are part of exactly one user code. In combinatorial design [11], such codes are known as 2-$(N, R, 1)$ designs (or $(2, N, R)$ Steiner systems). We focus on this type of user codes as it creates as little overlap between two user codes as possible, without having an extremely small number of codes (which would be the case if we allowed no overlap).

Using various results from the combinatorial design literature, we identify the (N, R) combinations for which such codes exist and present a simple way to generate the set of user codes $\mathcal{S}_{N,R}$ for, among others, all feasible combinations with $R \leq 5$ and $N < 85$. Provided that we have a population of $T = |\mathcal{S}_{N,R}|$ 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). Moreover, the closed formula applies to any 2-$(N, R, 1)$ design. Another important property of our algorithms is that the success probability is identical for all users, irrespective of their assigned user code; hence, the set of codes $\mathcal{S}_{N,R}$ is *fair* as all codes are equally good.

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 all the slots appear equally often in a single partition. 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. A closed formula that expresses the success probability for a population of T users is also presented.

In principle, the use of user codes imposes a strict bound on the user population, as any 2-$(N, R, 1)$ is of maximal cardinality. For larger populations, codes can be reused by some terminals, or the 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 somewhat larger populations (i.e., $T > |\mathcal{S}_{N,R}|$). Finally, we also demonstrate the effectiveness of these novel multiple access algorithms by comparing them with the random selection approach for a wide range of N values and provide some engineering rules on how to select the number of transmission attempts R as a function of the number of slots N and the population size T.

I. A USER CODE BASED MULTIPLE 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., $\epsilon = 1\%$.

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 [8], [9]. Notice, a packet is only lost if all R instances were involved in a simultaneous transmission. Instead of performing random transmissions we propose to assign a user code to each user. This weight R and length N user code identifies the R slots in which a user must transmit, when a packet becomes available.

We consider two types of systems:

- *Synchronous transmissions*: the data 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. The deadline of a packet is such that it needs to be successful in the next set of N slots.

- *Asynchronous transmissions*: the data transmissions are not synchronized to group boundaries and a packet has to be successful within the set of N slots following its generation time. In other words, a user can start his transmission much faster.

The first type of systems corresponds to frame-based networks where the channel is time divised into frames of a fixed length and a (small) window of N slots is present to support the random access channel in every frame. Grouping therefore occurs naturally as the random access data has to wait for the next set of N contention slots.

Synchronization may also seem necessary when we wish to rely on user codes. After all, when a set of user codes is said to share at most one slot, it seems essential that the sequences of N slots are synchronized among one another. However, for asynchronous systems we can easily apply the following procedure. Suppose a user code is represented by a bit vector of size N and weight R, where bit number i is set if the user must use slot i as one of his R slots. When a new packet becomes 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 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. Notice, a user needs to know the number of the current time slot (modulo N).

Some formulas to assess the performance of the random selection algorithm are giving in the Appendix. Typically, when analyzing such a scheme analytically, one focuses on the synchronous transmission model. For the closed formulas presented for the user code based algorithm we also restrict ourselves to the synchronous setup. One may expect a significant difference in performance, as in the asynchronous scenario a user can conflict with users who started transmitting as early as $N-1$ slots before as well as $N-1$ slots later, nearly doubling the potential collision window of each user. However, as demonstrated further on by means of simulation, both system types result in a nearly identical performance for the user code based algorithm. 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.

II. Generating sets of user codes

The user code based algorithm presented above can be used in combination with any 2-$(N, R, 1)$ design (or even with any t-(N, R, λ) design). In this section, we identify the (N, R) combinations for which 2-

$(N, R, 1)$ designs exist and explain how to generate the set $S_{N,R}$ in a very simple manner. For $R = 2$, a 2-$(N, 2, 1)$ design consists of all the two-element subsets of the N slots; therefore, we will focus on how to generate designs for $R = 3, 4$ and 5, as small values of R are the most relevant from a practical point of view.

It is easy to verify [11] that a 2-$(N, R, 1)$ design can only exist if $R-1$ divides $N-1$ and $R(R-1)$ divides $N(N-1)$, which can be reformulated as $N = 1 \mod R-1$ and $N^2 = N \mod R(R-1)$. This necessary condition was also proven to be sufficient for N large [12]. Furthermore, for $R = 3, 4$ and 5 this condition was also shown to be sufficient by Hanani [13] for all N, meaning after rewriting this condition, it suffices that

$$\begin{aligned} N = 1 \textit{ or } 3 \mod 6 \quad & \text{for } R = 3, \\ N = 1 \textit{ or } 4 \mod 12 \quad & \text{for } R = 4, \\ N = 1 \textit{ or } 5 \mod 20 \quad & \text{for } R = 5. \end{aligned}$$

Even though the proof of existence given by Hanani is by construction, these constructions are very cumbersome and not suited to generate the set of user codes $S_{N,R}$ in an efficient manner.

To generate the user codes, we will rely on cyclic 2-$(N, R, 1)$ designs, which are also known as cyclic $(N, R, 1)$-balanced incomplete block designs (CBIBDs). CBIBDs form a subclass of the 2-$(N, R, 1)$ designs. To specify a CBIBD, define the orbit of a block, i.e., user code, $B = \{b_1, \ldots, b_R\}$ as the set of distinct blocks/codes

$$B + i = \{b_1 + i \mod N, \ldots, b_R + i \mod N\},$$

for $i \in \{0, 1, \ldots, N-1\}$. Any element in the orbit is called a base block and specifies the entire orbit. If the orbit contains N elements, it is said to be full, otherwise it is termed short. The orbit that contains the block

$$\{0, N/R, 2N/R, \ldots, (R-1)N/R\}$$

is called the regular short block. A CBIBD with $N = 1 \mod R(R-1)$ is a 2-$(N, R, 1)$ design and its $N(N-1)/R(R-1)$ user codes consist of $(N-1)/R(R-1)$ full orbits. A CBIBD with $N = R \mod R(R-1)$ its $N(N-1)/R(R-1)$ user codes on the other hand consist of $(N-R)/R(R-1)$ full orbits and a single short orbit of size N/R which corresponds to the regular short orbit. Hence, a CBIBD can be specified completely by either $(N-1)/R(R-1)$ base blocks (if $N = 1 \mod R(R-1)$) or $(N-R)/R(R-1)$ base blocks for the full orbits and the base block $\{0, N/R, \ldots, (R-1)N/R\}$ for

the short block (if $N = R \mod R(R-1)$). The set of codes $S_{N,R}$ is straightforward to generate from the set of base blocks in a cyclic manner.

For $R = 3$, it has been shown [14] that for any $N = 1$ *or* $3 \mod 6$ there exists a CBIBD, except for $N = 9$. For $R \geq 4$, the existence of cyclic BIBDs is an unresolved and difficult problem, however, a CBIBD with $R = 4$ exists for $N = 1$ *or* $4 \mod 12$ for all $N \leq 600$, except for $N = 16, 25$ and 28 [15]. For $N = 5$ one can often find a CBIBD when $N = 1$ *or* $5 \mod 20$. For instance, when $N < 85$, the only N for which there is no CBIBD is 25 and 45. A table containing the base blocks for all the (N, R) combinations used to generate the required codes via a CBIBD for $N < 85$ is provided in the Appendix. For the remaining six (N, R) cases, it is not hard to generate a set of user codes. For instance, $(N, R) = (9, 3), (16, 4)$ or $(25, 5)$ corresponds to an affine geometry of dimension 2 over $GF(3), GF(4)$ or $GF(5)$, respectively, meaning it suffices to list the sets of points that form lines in these geometries to get the set $S_{N,R}$ of user codes.

III. Performance in a $T = |S_{N,R}|$ user population

A. Analysis

In this section we demonstrate that, using the highly symmetric structure of a 2-$(N, R, 1)$ design, we can quite easily establish an expression for the success probability of an arbitrary packet. Notice, the success probability is valid for any 2-$(N, R, 1)$ design and not merely for the CBIBDs discussed in the previous section.

We start by assuming that we have a user population of $C = |S_{N,R}| = N(N-1)/R(R-1)$ 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 more) than C users in Section IV (or Section VI). For the performance analysis we consider a synchronous system, as was done when analyzing the algorithm that selects R slots in a random manner [8], [9]. Furthermore, for the user code based algorithm, we will show by simulation that the results obtained from the synchronous scenario nearly coincide with those in the asynchronous setup.

For the analysis of our user code based algorithm, it is important to notice that a slot that is part of some user code c will also be part of exactly $S = (N-R)/(R-1)$ other user codes, because any two slots uniquely characterize a user code and all codes consist of exactly R slots. Further, every code $c' \neq c$ shares at most one slot with c, making the sets of user codes that share one of the R slots of c disjoint.

Assume $W \leq C$ 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 slot of a particular user code c is not shared by one of the other $W-1$ user codes. The probability that a specific set of i slots belonging to c is not used by any of the other $W-1$ codes equals

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

because there are C codes in total (including code c) and iS of them share a slot with the specific set of i slots on c. To get the success probability $p_{suc}(W)$ of a tagged user, we can use the inclusion-exclusion principle such that we do not count too many successes, as follows:

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

Remark, the success probability does not depend on the specific user code assigned to the tagged user, implying that the user codes are *fair*. 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

$$p_{suc} = \sum_{W=1}^{C} \binom{C-1}{W-1} p^{W-1}(1-p)^{C-W} p_{suc}(W) \quad (1)$$

$$= \sum_{W=1}^{C} \frac{W}{\rho N} \binom{C}{W} p^{W}(1-p)^{C-W} p_{suc}(W), \quad (2)$$

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

B. Numerical Results

Figure 1 illustrates the error probability for arrivals following a Poisson process, as defined in previous section, for the case where $R = 4$. The values of N were chosen such that this figure holds an example with a CBIBD with $N = 1 \mod R(R-1)$ and $N = R \mod R(R-1)$. 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. This gain can be understood as the

specific construction of the user codes, that is two codes share at most one common slot, significantly increases the chances of retaining at least one successful packet per user. Having a loss tolerance of about 1% thus means that we can support substantially higher loads using the user code based approach. Further notice that, given a fixed load ρ, increased user populations (and correspondingly more slots N to chose from) cause more packet losses for the user code based algorithm, as opposed to the random selection that seems to benefit in the more slots and users scenario. Similar figures can also be constructed for other values of R, indicating that the gain provided by the user code approach increases as R increases.

A comparison with a time driven simulation is provided. As the closed formulas are exact for the synchronous setup, there was a perfect agreement with the simulated synchronous scenario. Figure 1 also depicts the simulated asynchronous scenario, where we use the shifted bit vector approach for the user codes as explained in Section I. A remarkable observation can be made with respect to the synchronization mechanism. For the random selection, synchronization (or grouping) 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 in 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 synchronous analysis and the simulated asynchronous case, 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 were merged into one arrival in the synchronous 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 packets of the same user. We refer to Section VII-C for more details.

IV. Selecting T of the $|S_{N,R}|$ user codes

In this section we consider a population of $T < |S_{N,R}|$ users and address the problem of selecting T user codes from the set of $|S_{N,R}|$. 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 $|S_{N,R}|$ 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

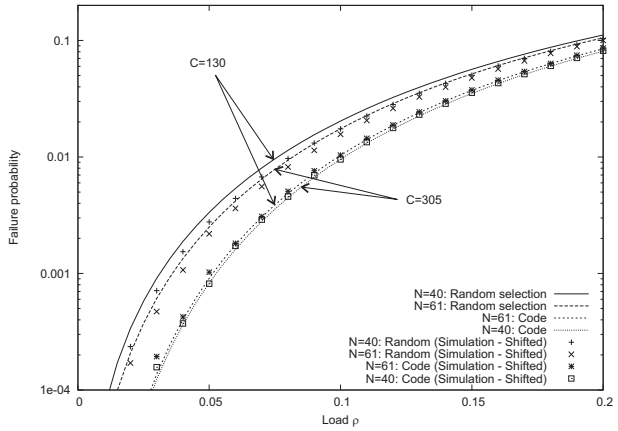

Fig. 1. Performance results in a $|S_{N,R}|$ user population. In this case $R = 4$ and $N = 40$ and 61, which results in $C = 130$ and 305 users, respectively

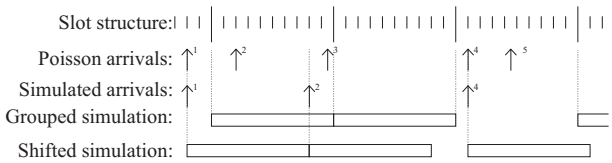

Fig. 2. Illustration of the arrival process of a single user, as used by the simulation. Poisson arrivals 3 and 5 are not considered in the simulation, since they are both the second arrival within the same group. Although arrival 2 occurs shortly after the first one, it is used, but assumed to arrive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

of T codes. The 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}.

The idea is to partition the set of all user codes $\mathcal{S}_{N,R}$ into two-by-two disjoint sets $\mathcal{S}_1, \ldots, \mathcal{S}_n$, for some $n > 1$, where $\mathcal{S}_i$ contains $s_i N/R$ user codes such that each slot appears exactly s_i times in $\mathcal{S}_i$. Next, we list all the user codes by first listing $\mathcal{S}_1$ in some order, followed by $\mathcal{S}_2$, etc. Ideally, we would like to have $s_i = 1$ for all $i = 1$ to n, meaning each set consists of N/R codes and the union of these codes results in the complete set of N slots. Designs that allow such a partitioning are known as resolvable designs [11]. However, resolvability is a rather strong property and many designs cannot be resolved.

The CBIBDs introduced earlier naturally lead to the following sets. If $N = 1 \mod R(R-1)$, we can partition $\mathcal{S}_{N,R}$ into $\mathcal{S}_1, \ldots, \mathcal{S}_n$, with $s_i = R$ and $n = (N-1)/R(R-1)$ by assigning the orbit of the i-th base block, which is full and holds N user codes, to $\mathcal{S}_i$. When $N = R \mod R(R-1)$, we define $s_1 = 1$ and associate the regular short block, containing N/R codes, to $\mathcal{S}_1$, while $s_2 = \ldots = s_n = R$, with $n = (N-R)/R(R-1)+1$ and $\mathcal{S}_i$ holds the orbit of the $(i-1)$-th full base block, for $i > 1$. As will become apparent in the next section, the order of the full base blocks is irrelevant for the performance of

the resulting multiple access scheme.

V. Performance in a $T < |S_{N,R}|$ user population

A. Analysis

In this section we derive a new expression for p_{suc} taking into account that we have only $T < |S_{N,R}| = C$ users. The closed expressions presented apply to any user code based algorithm making use of a 2-$(N, R, 1)$ design where the set of user codes is partitioned into $\mathcal{S}_1, \ldots, \mathcal{S}_n$ as indicated in the previous section and $T = \left(\sum_{i=1}^{t} s_i\right) N/R$ for some $1 \leq t \leq n$. Notice, for resolvable designs we therefore cover all population sizes T that are a multiple of N/R, for the CBIBDs the successive population sizes T covered differ by N users. For other values of T, we can get a useful approximation by considering the closest T value of this form.

Denote $T = kL$, with $L = N/R$ and $k = \sum_{i=1}^{t} s_i$. Due to the design of the selection algorithm, each slot is shared by exactly k user codes. Thus, if we tag a user, each slot belonging to its user code c will be shared by exactly $k-1$ other users. Also, the set of codes that contain one slot of c will be disjoint with a code that shares any other slot with c. Hence, analogue to Section III-A, where S is now replaced by $k-1$ and C by T, we find

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

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

B. Numerical Results

Figure 3 illustrates the loss probability for the case where $R = 3$, 4 and 5, for more scenarios we refer to Section VII-C. The values of N were chosen as $N = 45$, 64 and 81, such that for all three scenarios the number of available user codes C is close to 330. 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 $T = N/R$. 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 $T < C$.

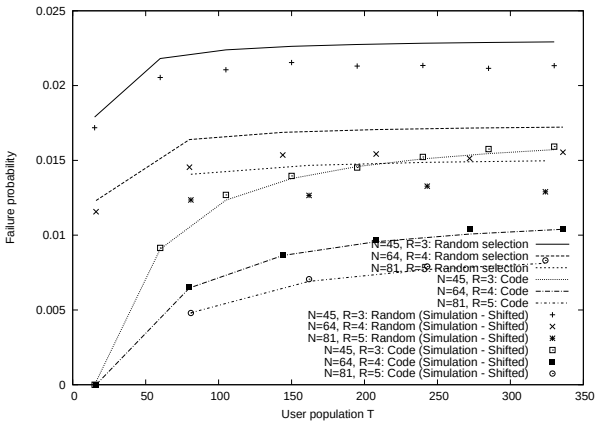

Fig. 3. Performance results in a $T \leq |S_{N,R}|$ user population, with $\rho = 0.1$ and $R = 3, 4$ and 5. Note for $N = 45, 64$ and 81, we have $C = 330, 336$ and 324, respectively.

VI. Dealing with more than $|S_{N,R}|$ users

Eventhough the user codes are mostly effective when the user population T is bounded by $|S_{N,R}|$, we will demonstrate that these codes still have their merits even when T exceeds $|S_{N,R}|$. We discuss two simple possibilities for supporting larger populations that show how to exploit the $|S_{N,R}|$ 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 $\{k|S_{N,R}| + i | k \geq 0\}$. 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 $|S_{N,R}|$.

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

We finally note that it might be useful to consider other t-(N, R, λ) designs when the user populations is of size $T > |S_{N,R}|$. For instance, setting $t = 3$ and $\lambda = 1$, would allow two codes to share at most two slots. Notice, eventhough the code reuse solution mentioned above is a 2-(N, R, λ) design if all codes are used exactly λ times, other designs of this type should result in a better performance. We plan to address these possibilities in some future work.

VII. Performance in a $T > |S_{N,R}|$ user population

A. Analysis of code reuse

The analysis presented in this section applies to any 2-(N, R, λ) design that is obtained from a 2-$(N, R, 1)$ design by code reuse, but does not necessarily apply to other 2-(N, R, λ) designs. Consider a population of $T > |S_{N,R}| = C$ users, where user j uses code $j \ mod \ |S_{N,R}|$. 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 that W 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 $T = kL$, where L was defined as N/R and $k = \alpha C/L + \sum_{i=1}^{t} s_i$ similar to Section V, meaning we distribute the C codes α times among the first αC users and the remaining $kL - \alpha C$ users are given the codes in the first t partitions $\mathcal{S}_1, \ldots, \mathcal{S}_t$. By noticing that each slot is part of exactly k user codes (of which some are identical due to the reuse) and by applying similar arguments as before, one establishes

$$\begin{aligned} p_{suc}(W) = p_m & \sum_{i=1}^{\min(R, \lfloor \frac{1+T-W-\alpha}{k-\alpha} \rfloor)} (-1)^{i+1} \binom{R}{i} \frac{\binom{(T-\alpha)-i(k-\alpha)}{W-1}}{\binom{T-1}{W-1}} \\ +(1-p_m) & \sum_{i=1}^{\min(R, \lfloor \frac{T-W-\alpha}{k-1-\alpha} \rfloor)} (-1)^{i+1} \binom{R}{i} \frac{\binom{(T-1-\alpha)-i(k-1-\alpha)}{W-1}}{\binom{T-1}{W-1}}. \end{aligned}$$

To obtain the success probability p_{suc} for Poisson arrivals, we refer to Equation (1).

B. Analysis of user codes combined with random selection

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

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

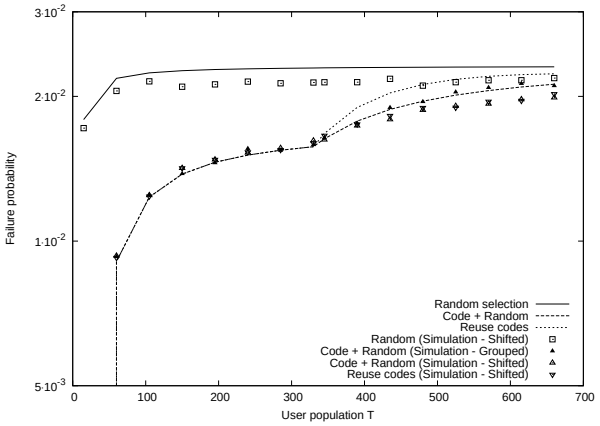

Fig. 4. Performance results in a $T > |S_{N,R}|$ user population, with $\rho = 0.1$ for $R = 3$ and $N = 45$, meaning $C = 330$

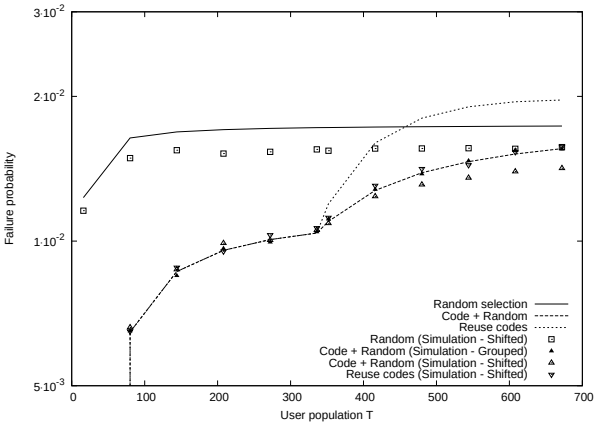

Fig. 5. Performance results in a $T > |S_{N,R}|$ user population, with $\rho = 0.1$ for $R = 4$ and $N = 64$, meaning $C = 336$

$W^{(c)}$ of them have a user code and $W^{(r)}$ do not. Given that $W^{(c)}$ 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

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

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

Deriving a closed expression for the success probability when the tagged user belongs to the set of the remaining $W^{(r)}$ users, who 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 $W - 1$ other users (including the $W^{(c)}$ 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 resemblance between the actual simulated success probability and this approximation is given in Section VII-C. Given this approximation, the resulting success probability of an arbitrary active user becomes:

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

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

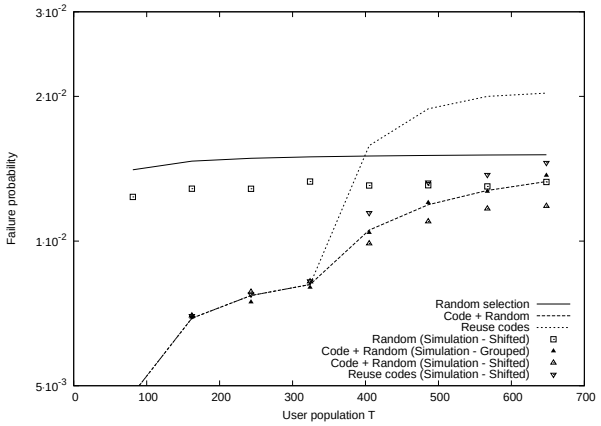

Fig. 6. Performance results in a $T > |S_{N,R}|$ user population, with $\rho = 0.1$ for $R = 5$ and $N = 81$, meaning $C = 324$

C. Numerical results

Figures 4, 5 and 6 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 outperforms 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. We can also remark that in this case, while the population size is more or less held constant, transmitting the packet more often (when more delay is allowed) seems to have a positive effect on the success probability. We refer to Section VIII, where we will investigate further on the optimal choice of R.

The simulation results for the synchronous 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 III-B we noticed that there is a synchronization penalty associated with the random selection, while the user code scheme did not experience such a penalty for $T \leq |S_{N,R}|$. When the population T becomes larger than $|S_{N,R}|$, this penalty does surface 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 $p_{suc}(W)$, 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

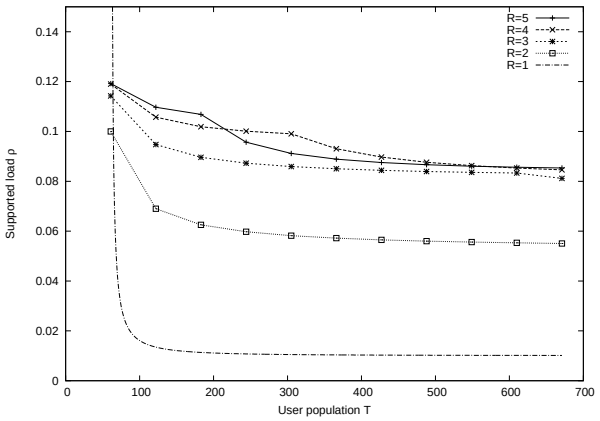

Fig. 7. Maximum supportable load ρ for $N = 61$ and a loss tolerance of $\epsilon = 0.01$ for $R = 1$ to 5

of the population. Recall, among the users who transmit using a user code there is no unfairness (as there is no unfairness among the users who transmit at random), as all user codes result in the same success rate.

VIII. Engineering Rules

In this section we provide a number of essential engineering rules when deploying a multiple access channel with a user code based algorithm. The loss tolerance ϵ and the number of slots N, related to the maximum delay, typically both stem from the application under consideration. Given ϵ and N, we determine the maximum load ρ that the random access channel can carry without violating the loss tolerance, for various population sizes T when $R = 1$ to 5.

When $R = 1$, we will assign a single slot to the first N users, meaning if $T \leq N$ all packets are successful, while the remaining $T - N$ users select a single slot at random. For $R = 2$, the set of user codes $\mathcal{S}_{N,2}$ corresponds to all the 2-element subsets of $\{0, \ldots, N-1\}$. These length 2 user codes are CBIBDs, for all N, where all the base blocks are full and given by $(0,1), (0,2), \ldots, (0,(N-1)/2)$ when N is odd. For N even, we have $N/2 - 1$ full base blocks $(0,1), \ldots, (0, N/2-1)$ and the short block $(0, N/2)$. For $T \leq N(N-1)/2$, we can therefore order the list of user codes as explained in Section IV and apply the closed formulas as presented in Section V.

Figure 7 depicts the maximum supportable load ρ for an application with a loss tolerance of 1% and a maximum delay of $N = 61$ slots. Clearly, as the population size T increases, this load decreases. We also observe a small drop in the $R = 5, 4$ and 3 curve at $T = |\mathcal{S}_{N,R}| = 183, 305$ and 610, respectively, as larger populations imply that the remaining $T - |\mathcal{S}_{N,R}|$ users make use of a random transmit strategy. Thus, for T to infinity these curves should converge to the random selection strategy (as the percentage

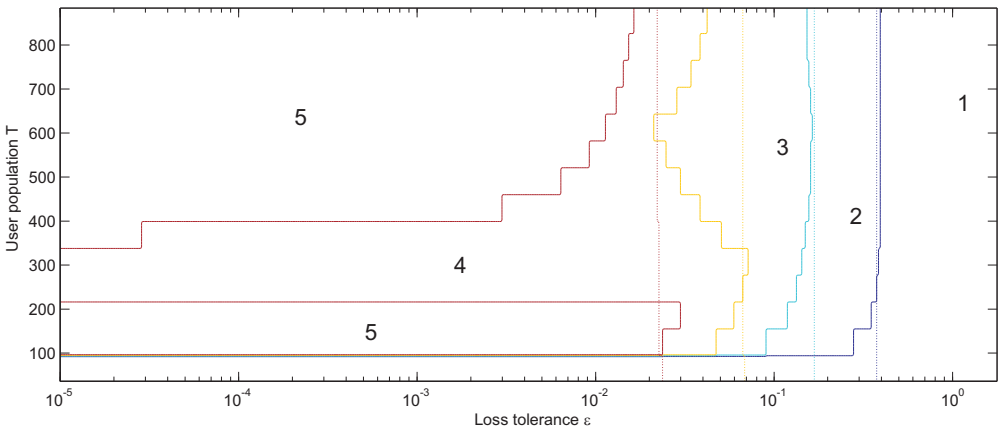

Fig. 8. Optimal number of transmission attempts R as a function of the population size T and the loss tolerance ϵ for both the code based and random algorithm for $N = 61$

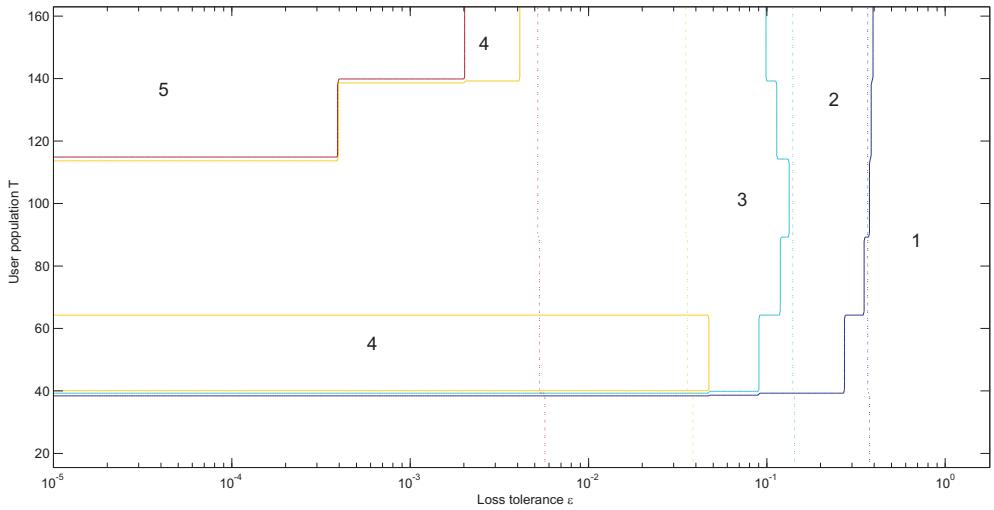

Fig. 9. Optimal number of transmission attempts R as a function of the population size T and the loss tolerance ϵ for both the code based and random algorithm for $N = 25$

of users with a code decreases to zero). For $T \leq N$, we obviously find that a single transmission is best as all packets are successful (as every user has his own slot). However, as soon as T only marginally exceeds N, its performance deteriorates quickly. The $R = 2$ system, with its 1830 user codes, performs better, but is still well below the other R values. The optimal number of transmission attempts R for this specific setup turns out to be either $R = 5$ or $R = 4$ depending on the population size T.

In order to get some general understanding of the optimal choice for R as a function of the population size T and the loss tolerance ϵ, we have included Figure 8. In this figure we have partitioned the (ϵ, T) plane for $\epsilon \in [10^{-5}, 1]$ and $T = 0, 61, \ldots, 854$ into different areas. The number of the area indicates which value of R supports the highest load without violating the loss tolerance when the code based algorithm

is used with $R = 1$ to 5. The four nearly vertical lines in the plot make the same partitioning, but for the random selection strategy (i.e., without using codes).

A first observation is that as the loss tolerance increases, fewer transmissions R perform better for both the random and code based algorithm. For the random selection, the population size has hardly any impact on the optimal choice of R, that is, the four lines partitioning the plane are nearly vertical. For the code based system the population size has a much stronger impact and as R increases the lines become less and less vertical (the $R = 5$ line even dissappears temporarily from the figure). Each of these lines follows the same pattern, that becomes more pronounced as R grows. Typically, for T small, it is not far from its corresponding random line. In this case, all the users have a code. When the population size exceeds the number of available codes, which occurs as $T = 3660/R(R-1)$, the line moves away from the random line. This comes as no surprise as part of the user population start to transmit randomly for $T > 3660/R(R-1)$, causing a drop in the achievable maximum througput (see Figure 7). As the population size T continues growing, it slowly converges back to the random line (which is what we expect as T to infinity causes both schemes to behave identical).

Figure 9 depicts the same results for $N = 25$ slots. The behavior is completely analogue to the previous figure (except that the $R = 5$ line only enters the figure for T large as we only have 25 codes). We also observe that a smaller N value implies that fewer transmission attempts tend to perform better for a given loss tolerance ϵ, as all the lines are shifted to the left when going from $N = 61$ to $N = 25$.

We also remark that $R = 4$ and 5 with $N = 25$ are both among the few exceptions for which no CBIBD exists (see Section II). However, for $R = 4$ and 5 we can construct a set of user codes using a difference system on $\mathbb{Z}_5 \oplus \mathbb{Z}_5$ [11] and the two dimensional affine plane over $GF(5)$, respectively. Both these codes allow a partitioning as discussed in Section IV and therefore, all closed formulas are applicable to these codes as well.

APPENDIX A
BASE BLOCKS FOR CBIBDS WITH $R = 3, 4$ AND 5

In this section we list the base blocks needed to generate the user codes of the CBIBDs with $N < 85$ and $R = 3, 4$ and 5. Most of these entries were copied from [16]. If present, the short block is emphasized.

Base blocks for CBIBDs with $R = 3, 4, 5$

$R = 3$											
7	0 1 3										
13	0 1 4	0 2 7									
15	0 1 4	0 2 9	*0 5 10*								
19	0 1 4	0 2 9	0 5 11								
21	0 1 3	0 4 12	0 5 11	*0 7 14*							
25	0 1 3	0 4 11	0 5 13	0 6 15							
27	0 1 3	0 4 11	0 5 15	0 6 14	*0 9 18*						
31	0 1 12	0 2 24	0 3 8	0 4 17	0 6 16						
33	0 1 3	0 4 10	0 5 18	0 7 19	0 8 17	*0 11 22*					
37	0 1 3	0 4 26	0 5 14	0 6 25	0 7 17	0 8 21					
39	0 1 3	0 4 18	0 5 27	0 6 16	0 7 15	0 9 20	*0 13 26*				
43	0 1 3	0 4 9	0 6 28	0 7 23	0 8 33	0 11 30	0 12 26				
45	0 1 3	0 4 10	0 5 28	0 7 34	0 8 32	0 9 29	0 12 26	*0 15 30*			
49	0 1 3	0 4 9	0 6 17	0 7 23	0 8 30	0 10 31	0 12 36	0 14 34			
51	0 1 3	0 4 9	0 6 25	0 7 35	0 8 22	0 10 21	0 12 27	0 13 31	*0 17 34*		
55	0 1 3	0 4 9	0 6 16	0 7 32	0 8 29	0 11 42	0 12 27	0 14 36	0 17 37		
57	0 1 3	0 4 9	0 6 13	0 8 26	0 10 33	0 11 32	0 12 40	0 14 41	0 15 35	*0 19 38*	
61	0 1 3	0 4 9	0 6 13	0 8 25	0 10 33	0 11 30	0 12 32	0 14 40	0 15 37	0 16 34	
63	0 1 3	0 4 9	0 6 13	0 8 25	0 10 41	0 11 44	0 12 36	0 14 37	0 15 43	0 16 34	*0 21 42*
67	0 1 3	0 4 9	0 6 13	0 8 23	0 10 38	0 11 33	0 12 42	0 14 32	0 16 43	0 17 36	0 20 46
69	0 1 3	0 4 9	0 6 13	0 8 24	0 10 38	0 11 47	0 12 32	0 14 40	0 15 50	0 17 42	0 18 39
	0 23 46										
73	0 1 3	0 4 10	0 5 35	0 7 32	0 8 24	0 9 55	0 11 53	0 12 52	0 13 39	0 14 29	0 17 54
	0 22 45										
75	0 1 67	0 2 47	0 3 41	0 4 69	0 5 68	0 11 55	0 13 61	0 15 33	0 16 52	0 17 43	0 19 40
	0 22 51	*0 25 50*									
79	0 1 29	0 2 19	0 3 14	0 4 42	0 5 13	0 6 22	0 7 52	0 9 55	0 10 53	0 12 59	0 15 54
	0 18 48	0 21 56									
81	0 1 39	0 2 58	0 3 34	0 4 21	0 5 67	0 6 15	0 7 36	0 8 59	0 10 63	0 11 37	0 12 61
	0 13 48	0 16 40	*0 27 54*								
$R = 4$											
13	0 1 3 9										
37	0 1 3 24	0 4 9 15	0 7 17 25								
40	0 1 4 13	0 2 7 24	0 6 14 25	*0 10 20 30*							
49	0 1 3 8	0 4 18 29	0 6 21 33	0 9 19 32							
52	0 1 3 7	0 5 19 35	0 8 20 31	0 9 24 34	*0 13 26 39*						
61	0 1 3 8	0 4 13 31	0 6 25 41	0 10 24 39	0 11 23 44						
64	0 1 3 7	0 5 18 47	0 8 33 44	0 9 19 43	0 12 26 49	*0 16 32 48*					
73	0 1 3 7	0 5 13 37	0 9 26 55	0 10 22 43	0 11 25 45	0 15 31 50					
76	0 1 7 22	0 2 11 45	0 3 59 71	0 4 32 50	0 10 37 51	0 13 36 60	*0 19 38 57*				
85	0 2 41 42	0 17 32 38	0 18 27 37	0 13 29 36	0 11 31 35	0 12 26 34	0 5 30 33				
$R = 5$											
21	0 1 4 14 16										
41	0 1 4 11 29	0 2 8 17 22									
61	0 1 3 13 34	0 4 9 23 45	0 6 17 24 32								
65	0 1 3 31 45	0 4 10 19 57	0 5 16 41 48	*0 13 26 39 52*							
81	0 1 3 7 33	0 5 20 28 39	0 9 21 52 65	0 10 24 46 64							

APPENDIX B

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 W 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 $W-1$ users equals

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

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

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

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

REFERENCES

[1] J. Massey, "The capacity of the collision channel without feedback," in *Abstracts of Papers, IEEE Int. Symp. Info. Th.*, Les Arcs, France, June, 1982, p. 101.

[2] J. Massey and P. Mathys, "The collision channel without feedback," *IEEE Trans. Inf. Theory*, vol. IT-31, pp. 192–204, March 1985.

[3] L. Bassalygo and M. Pinsker, "Restricted asynchronous multiple access," *Probl. Peredachi Inform.*, vol. 19, pp. 92–96, Oct. 1983.

[4] M. Tsatsanis, R. Zhang, and S. Banerjee, "Network-assisted diversity for random access wireless networks," *IEEE Trans. Signal Process.*, vol. 48, no. 3, pp. 702–711, Mar 2000.

[5] R. Zhang, N. Sidiropoulos, and M. Tsatsanis, "Collision resolution in packet radio networks using rotational invariance techniques," *IEEE Trans. Commun.*, vol. 50, no. 1, pp. 146–155, Jan 2002.

[6] B. Ozgul and H. Delic, "Wireless access with blind collision-multiplicity detection and retransmission diversity for quasi-static channels," *IEEE Trans. Commun.*, vol. 54, no. 5, pp. 858–867, May 2006.

[7] "Digital Video Broadcasting (DVB): Interaction Channel for Satellite Distributed Systems, European Telecommunication Standardization Institute (ETSI) EN 301 790 V1.4.1 (2005-9)."

[8] G. Choudhury and S. Rappaport, "Diversity aloha–a random access scheme for satellite communications," *IEEE Trans. Commun.*, vol. 31, pp. 450–457, Mar 1983.

[9] S.-L. Su and V. Li, "Comments on diversity ALOHA," *IEEE Trans. Commun.*, vol. 32, pp. 1143–1145, Oct 1984.

[10] E. Casini, R. D. Gaudenzi, and O. del Rio Herrero, "Contention resolution diversity slotted ALOHA (CRDSA): An enhanced random access scheme for satellite access packet networks," *IEEE Trans. Wireless Commun.*, vol. 6, pp. 1408–1419, April 2007.

[11] I. Anderson, *Combinatorial Designs: construction methods*. New York: John Wiley & sons, 1990.

[12] R. M. Wilson, "An existence theorem for pairwise balanced designs, iii: proof of the existence conjecture," *Journal of Combinatorial Theory, Series A*, vol. 18, no. 2, pp. 71–79, 1975.

[13] H. Hanani, "The existence and construction of balanced incomplete block designs," *Annals of Mathematical Statistics*, vol. 32, pp. 361–386, 1961.

[14] R. Peltesohn, "Eine losung der beiden hefferschen differenzenproblem," *Compositio Mathematica*, vol. 6, pp. 251–257, 1938.

[15] K. Chen and R. Wei, "A few more cyclic Steiner 2-designs," *The electronic journal of combinatorics*, vol. 13, no. #R10, 2006.

[16] C. J. Colbourn and J. H. Dinitz, *Handbook of Combinatorial Designs*. Chapman & Hall/CRC, 2006.

PLACE PHOTO HERE

Gino T. Peeters received his M.Sc. degree in Computer Science from the University of Antwerp (Belgium) in July 2005. In September 2005, he joined the "Performance Analysis of Telecommunication Systems" research group, at the Mathematics and Computer Science Department of the University of Antwerp. His main research interests include the performance analysis of telecommunication systems, more specifically medium access control problems and scheduling issues in satellite communication networks.

PLACE PHOTO HERE

Raf Bocklandt received his M.Sc. degree in mathematics in July 1999 and physics in July 2000 from the University of Ghent (Belgium). He obtained a PhD in science at the University of Antwerp (Belgium) in April 2002. From October 1999 until September 2003 he was assistant at the University of Antwerp. This position was interrupted by a 9 month Postdoctoral position at the university of La Sapienza in Rome (Nov 2002-July 2003). Since October 2003 he is a postdoctoral fellow at the FWO-Flanders.

His main research interests are noncommutative geometry, invariant theory, representation theory of groups and algebras and their applications to both physics and computer science. He has published papers in a variety of international mathematical journals (e.g., Math. Zeitschrift, Journal of Algebra, Linear Algebra and its Applications, Journal of Pure and Applied Algebra, etc.)

PLACE PHOTO HERE

Benny Van Houdt (M'04) received his M.Sc. degree in mathematics and computer science, and a PhD in science from the University of Antwerp (Belgium) in July 1997, and May 2001, respectively. From August 1997 until September 2001 he held an Assistant position at the University of Antwerp. Starting from October 2001 onwards he has been a postdoctoral fellow of the FWO-Flanders. In 2007, he became a professor at the Mathematics and Computer Science Department of the University of Antwerp, where he is a leading member of the PATS research group.

His main research interest goes to the performance evaluation and stochastic modelling of wired and wireless communication networks and random access systems in particular. Other areas of interest include manufacturing, operating systems, tool development, etc. He has published various papers, containing both theoretical and practical contributions, in a variety of international journals (e.g., IEEE JSAC, Performance Evaluation, Journal of Applied Probability, Stochastic Models, Queueing Systems, etc.) and in conference proceedings (e.g., ACM Sigmetrics, Networking, Globecom, Opticomm, ITC, etc.).