Analysis of Access Selection Algorithms for Multi-access Networks with Elastic Traffic*

Igor Cananéa[1], **Dênio Mariz**[1], **Jeísa Oliveira**[1]
Djamel Sadok[1], **Gábor Fodor**[2]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Av. Prof. Luis Freire, S/N – Recife – PE – Brazil

[2]Ericsson Research
SE-164 80 Stockholm, Sweden

{icc,denio,jeisa,jamel}@gprt.ufpe.br, gabor.fodor@ericsson.com

Abstract. *It is expected that future wireless systems will consist of several distinct radio access technologies, forming a multiaccess system offering voice and multimedia services. Previous works have shown that the combined capacity region of such systems depends on the service allocation policy that assigns user sessions to the subsystems. Current policies typically operate off-line, implying that the service mix is known a priori. In this paper we consider the on-line problem where sessions arrive one after the other and no assumptions on the service mix can be made. We study the performance of four on-line bin-packing algorithms by simulation and show that the algorithm termed LessVoice provided the best performance in terms of blocking probabilities and throughput.*

1. Introduction

We expect that future wireless systems will consist of several distinct radio access technologies, such as WCDMA/HSDPA, GSM/EDGE/GPRS, WLAN and others, forming a multiaccess system offering advanced multimedia services. Users can take advantage of such systems by using terminals equipped with multiple interfaces in such a way that any terminal can connect to the available subsystems. Thus, these grouped subsystems together establish multiaccess, multiservice capacity regions [4], [5], [6].

On top of multiaccess multiservice systems, Always Best Connected (ABC) Networks can be seen as a concept that allows users to not only be always connected but to do so while meeting some quality of service (QoS), cost or other criteria [9], [7]. In order to realize the ABC concept, a common radio resource management (CRRM) component must be employed that considers the available subsystems as a whole, rather than treating the accesses separately. The improvement of the performance of multiaccess systems was evaluated by simulation in [13]. The results were promising since they showed that when common operation of the subsystems is considered, the "trunking gain" increases when more systems are integrated into the multiaccess system. Although both conversational and packet data services were evaluated, they were considered separately; that is, only a single-service multiaccess scenario was considered.

*This work was supported by the Research and Development Centre, Ericsson Telecomunicações S.A., Brazil.

Recent studies showed that if the capacity region of individual subsystems are different, the combined capacity region for the multiservice systems varies according to the strategy used to assign users to subsystems [4], [5]. These papers show the existence of allocation strategies that maximize the total number of users that can be accommodated by the multiaccess system under some QoS constraints. The off-line allocation strategies in these papers assume that either a set of sessions can be allocated at the same time or in-progress sessions can be re-allocated.

Koo *et al.* [10] studied the combined capacity of multiaccess systems considering multiservice systems in a dynamic environment and provided upper and lower bounds for the gains, expressed in terms of their Erlang capacity. They also obtained an "assignment gain" when user services are taken into consideration in the assignment process. Considering specific settings for GSM and WCDMA-like subsystems, they presented gains ranging from 15% to 60% depending upon how users are assigned to subsystems and also upon the input traffic mix. Along another line, it was shown in [1] that for a scenario with a single radio access technology and multiple services including voice and elastic data traffic, the blocking probability can be made arbitrarily small by sufficiently reducing the bit rate of elastic applications, under certain conditions for their arrival rate.

It is clear that well thought access selection techniques play a crucial role in these dynamic environments in order to maximize the network capacity region and balance the applications over the subsystems, while satisfying the applications requirements. As recognized in [4], online algorithms have the advantage over off-line algorithms that sessions are allocated one by one and in-progress sessions do not need to be reallocated.

In order to deal with the problem of allocating user services onto a set of resources offered by several cooperating networks, we draw a parallel with the *bin packing problem* [8], [2]. This approach is similar to the one in [14] where the authors consider "FirstFit" as the base access selection algorithm to a pool of different access subsystems. The main focus of that paper was on reducing power consumption of user terminals while respecting access preferences, which must be defined *a priori* by the users for each application class. In that work, the authors considered a single-service (data flows) scenario and elastic applications were not taken into consideration.

The main contribution of this paper is the analysis of on-line algorithms for access selection in multi-access, multi-service systems with elastic traffic and the mapping to the binpacking problem. In this paper we examine the impact of these algorithms in terms of blocking probabilities and session throughputs. We take into consideration real time and elastic user sessions that arrive dynamically and leave the system after some residency time. We model the problem of access selection as the classical online variable-size bin packing problem, where applications are the objects to be packed and subsystems are the bins. Then, we adapt approximation algorithms that solve the bin packing problem and additionally propose a resource sharing algorithm. We show that in addition to the gains obtained from the presence of elastic applications, additional ones can be reached depending upon the access selection algorithm used.

The remainder of this paper is organized as follows. Section 2. gives some insights into the bin packing problem, the algorithms used for access selection and their mapping onto access selection in ABC networks. The scenario settings and results for the

performance analysis of access selection algorithms are shown in Section 3., with results for the scenario with non-elastic applications being discussed in Section 3.1., and the case for elastic applications in Section 3.2.. The impact of load of data applications and data elasticity in the blocking probability of voice calls and number of applications served by the network is discussed in Section 3.2.1. and the behaviour of the algorithms dealing with a worst-case input traffic mix is discussed in Section 3.3.. Section 4. presents the concluding remarks and discusses some proposals for future works in the area.

2. Access Selection and the Bin Packing Problem

2.1. Modeling Subsystems and Applications

The classical bin packing problem is a well studied optimization problem [2, 12, 8, 11]: given n objects with sizes $a_1, ..., a_n \in (0, 1]$, find a packing in unit-sized bins that minimizes the number of bins used. In the off-line version of this problem, it is possible to consider all the objects and choose the order of assignment. In the online version however, each object must be assigned in turn, without knowledge of the next objects. That is, given $n - 1$ already packed objects with sizes $a_1, ...a_{n-1} \in (0, 1]$, the new object n with size $a_n \in (0, 1]$ must be packed in such a manner that the number of used bins is minimized [11]. It is worth mentioning that the problem of finding an optimal packing is known to be NP-Hard [8]. We say that an online bin packing problem is *bounded space* if the number of available bins at any one time is predefined [2]. In the *variable-size* version of the online bin packing problem the bins can have different capacities [12], and the goal now becomes minimizing the sum of the capacities of the used bins.

In this work we consider a multiaccess, multiservice system (henceforth mentioned network) N consisting of s cells (henceforth mentioned subsystems). Just like in [4], each subsystem $s \in N$ is associated with an access technology T supporting a set K of bearer services (or application classes). The subsystems are assumed to have the same coverage area, and the terminal capabilities are assumed to be such that any multimode terminal can connect to any subsystem. A number of application instances $A = \{a_1, a_2, ...\}$, each one associated with an application class $k \in K$, arrive to be allocated in N. We denote $Cap(s) > 0$ as the capacity of the subsystem s, measured in bits per second (*b/s*), which is determined by the access technology associated with s (e.g. GSM/EDGE, WCDMA, ...). Hence, it is possible to have $Cap(i) \neq Cap(j), i \neq j$.

Applications arrive one after the other and there is no *a priori* knowledge on the next arriving applications. We denote $Class(a)$ as the application class associated with the application instance $a \in A$ and $Size(a)$ as the bandwidth requirement of *a*, measured in *b/s*. It is important to note that for a given application instance a, $Size(a)$ is determined by each subsystem, in such a way that $Size(a, s)$ is used to denote the bandwidth resource that *a* would take from s if s was selected to hold a. Assuming $A(s)$ as the set of application instances already allocated to subsystem s, we define $Free(s) = Cap(s) - \sum_{a \in A(s)} Size(a, s)$ as the available resources in s. Thus, an application a can be allocated to a selected subsystem s if $Size(a, s) \leq Free(s)$, being otherwise rejected (blocked). If successfully allocated, application a consumes $Size(a, s)$ resource units from $Cap(s)$ under its residency time, freeing the resources upon leaving.

Access selection is formulated as the problem of finding the best way of allocating applications in subsystems while minimizing blocking probability and maximizing

system capacity. Given the dynamic nature of this problem, it is clear that online algorithms are the most appropriate to deal with it. Thus, we map the problem of access selection onto the bounded space variable-size online bin packing problem where objects are applications arriving and bins are subsystems where applications should be packed.

Unlike in classical bin packing where the object size is known before packing is performed, in our problem the size of a given application can not be determined *a priori*, since it depends on the subsystem that will hold it. Hence, without loss of generality, we adapt existing algorithms to consider the actual size of the objects as the one it would have if packed to a specific target bin. That is, the algorithms always consider $Size(a, s)$ to know the amount of space object a would take from bin s. A second adaptation made refers to working in a dynamic environment, that is, applications arrive, hold part of network resources for a while and frees them upon leaving the network. Therefore, any access selection decision must consider the current network state to make a decision.

2.2. Algorithms for Access Selection

We consider three well-known approximation algorithms for the online bin packing problem: FirstFit, BestFit and WorstFit [2]. In addition, we devise two others, named LessVoice and Random. Each algorithm makes a decision based on the current state of the network and the application that needs to be allocated. Next, these algorithms are described.

- *FirstFit (FF):* a subsystem s is randomly selected with equal probability among N. Application instance a is allocated to s if $Size(a, s) \leq Free(s)$ holds. Otherwise, the next subsystem is selected in a round robin fashion until a new subsystem s in which a fits is found. If no subsystem is found, a is rejected.
- *BestFit (BF)*: a subsystem s is selected if there is enough space available for application a and if it has less available resources s compared to the others.
- *WorstFit (WF):* a subsystem s is selected if there is enough space available for application a and if it has more available resources compared to the others.
- *LessVoice (LV):* We present an algorithm that allocates application a into a subsystem s where its expected resource cost $Size(a, s)$ relative to $Size(voice, s)$ for the application class in question is the smallest. This idea was proposed in [4] (see Section 3.). *LessVoice* computes $r(a, s) = Size(a, s)/Size(voice, s)$ for each $s \in N$ and selects the subsystem s with smallest $r(a, s)$. If there is more than one subsystem with the same minimum $r(s)$, the subsystem with the smallest $Size(voice, s)$ among these is selected. If even though a tie remains, the subsystem having more free resources is chosen among the tied ones.
- *Random (RN):* a subsystem s is randomly selected with equal probability among N. Application a is allocated to s if $Size(a, s) \leq Free(s)$ and rejected otherwise. This algorithm is used as a worst case reference.

Regarding the time complexity of algorithms, we have: $\mathcal{O}(nm)$ for *FirstFit*, *BestFit*, *WorstFit* and *LessVoice*; and $O(n \log m)$ for *Random*, assuming n as the number of applications submitted for allocation and m as the number of available subsystems.

3. Performance Analysis

We consider two dynamic scenarios in which applications of two classes, voice call and data applications, arrive to be allocated in the network. For both scenarios, we assume the

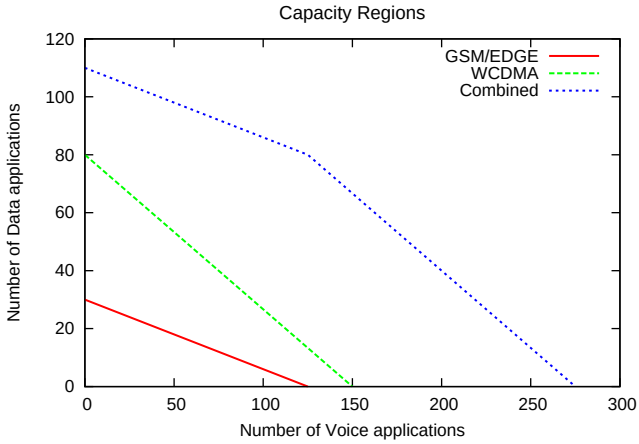


Figure 1. Capacity regions for GSM, WCDMA and combined GSM+WCDMA.

presence of two different subsystems, namely, WCDMA and GSM/EDGE. Without loss of generality, both subsystems are assumed to have linear capacity regions and a single cell of each. Both application classes are assumed to arrive in a common coverage region for both subsystems, in such a manner that any subsystems may be chosen.

Capacity regions for voice and data bearer services for a multiaccess system comprising GSM/EDGE and WCDMA access technologies are assumed as in [4] (see also detailed discussion in [6, Section 9.1, Table 5]). The QoS requirements include a bit error rate (BER) that yields acceptable voice quality for the voice bearers and a maximum perceived throughput of $150kb/s$ for the data bearers, assuming 10MHz worth of spectrum being available to both subsystems. We define $Q(s,k)$ as the maximum number of single-service applications k that a subsystem s can hold and, based on the characteristics assumed for GSM and WCDMA, we obtain: $Q(GSM, voice) = 125$, $Q(GSM, data) = 30$, $Q(WCDMA, voice) = 150$ and $Q(WCDMA, data) = 80$ [4]. Since GSM/EDGE can bare 125 voice applications or 30 data applications, there is a 1:4.17 ratio between voice and data applications classes (it can hold 4.17 voice applications for every data application). WCDMA, on the other hand, can handle 150 voice applications or 80 data applications, giving a 1:1.875 ratio.

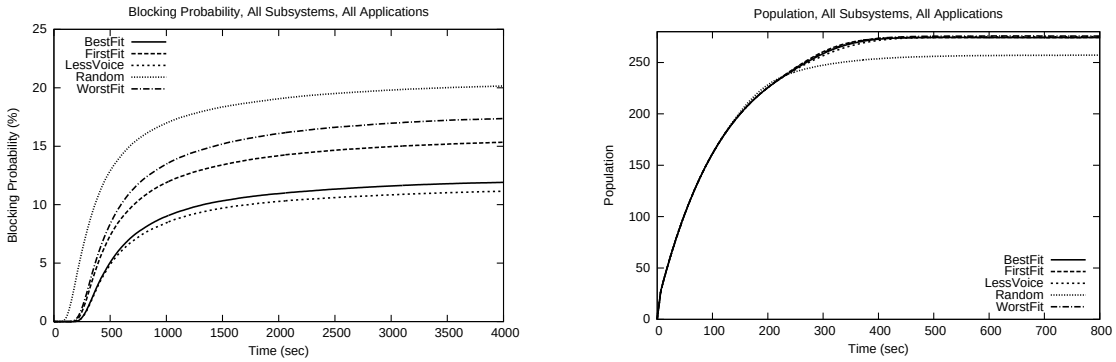
We define the combined capacity region as the set of expected number of applications of each class that can be accommodated by all subsystems under common operation, while maintaining acceptable quality [6]. Thus, as discussed in [4], we have $Q(Combined, voice) = 275$ and $Q(Combined, data) = 110$. Figure 1 shows the capacity regions, measured in number of applications, for GSM/EDGE, WCDMA and for them both combined. The curves depicted impose limits on all the possible combinations of voice and data applications.

We assume that applications of class k arrive following a Poisson process with $\lambda(k) = 1/460ms$, for $k = \{voice, data\}$. If the application is successfully allocated, it remains in the system for an exponentially distributed holding time with mean $\mu(k)$, where $\mu(voice) = 120s$ and $\mu(data)$ is defined by the time it takes to transmit 120kB, that is, $\mu(data) = 6.4s$. These parameters result in reasonable blocking probabilities and are applicable to compare access selection algorithms.

The results were obtained by simulation, using a discrete event simulator developed in C++. All curves shown in the graphics are an average of 45,000 replications and the adopted confidence level for the mean was 95%. Each replication represents a simulation of 800 seconds of operation of the network, the necessary time to reach the steady state of the network for all adopted metrics, except for blocking probability; in this case, we simulated a longer operation period of 4000 seconds, since its convergence was slower. The samples for the chosen metrics were collected once every second and each algorithm received exactly the same amount of applications to be allocated, in the same order and at the same time instant. The following subsections discuss the results.

3.1. Access Selection for Voice and Non-Elastic Data Applications

In this section we consider a multiaccess network offering two different services in a dynamic environment. Subsystems and applications are configured as described in Section 3., in which both voice and data applications bandwidth do not vary once allocated in the network. We assume $Size(data) = 150kb/s$ and $Size(voice) = \varphi kb/s$, where φ is a constant determined by the subsystem holding voice applications. According to [6, 4], we are using $\varphi = 36kb/s$ for GSM and $\varphi = 80kb/s$ for WCDMA.



(a) **(b)**

Figure 2. (a) Blocking probability obtained by each algorithm in a scenario with voice and non-elastic applications; (b) The network population obtained by each algorithm in the scenario with voice and non-elastic applications.

Figure 2(a) shows the blocking probability obtained by each access selection algorithm. Under the configured offered load, the algorithms *LessVoice*, *BestFit*, *FirstFit, WorstFit* and *Random* (in this order) obtained the best results. Figure 2(b) shows the instantaneous network population, measured as the total number of applications served by the network at a specific point in time. Except for *Random*, all the algorithms performed fairly equally, although towards the end *LessVoice* and *BestFit* outperformed the others, allocating approximately 255 applications. *FirstFit* and *WorstFit* allocate approximately 252 applications whereas *Random* allocated only 242 applications.

We introduce the concept of "utilization", which measures, in percentile, how much resources from a subsystem is occupied at a given instant. Then, we define the "load balance of the network" (balance, for short) as the standard deviation of the utilization of the subsystems, in such a way that it compares utilization of the subsystems during the access selection process. For example, considering two subsystems, if the utilization for one subsystem is 100% and the utilization for the other one is 0%, the balance

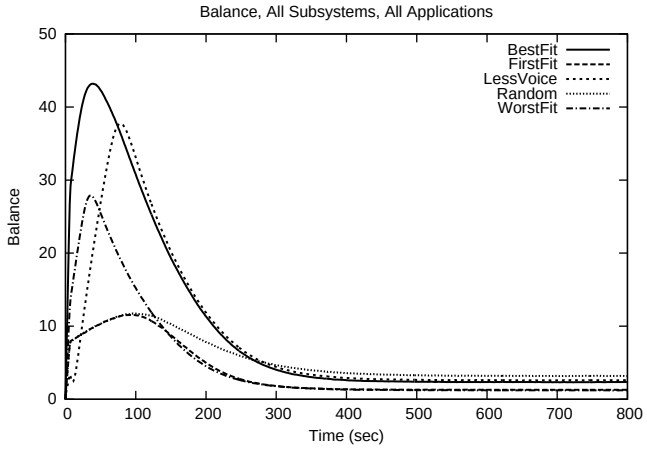


Figure 3. The balance of the network, as obtained by each algorithm in a scenario with voice and non-elastic applications.

reaches its maximum value of 50. If the utilizations are equal, the balance will be zero and when these are close, the balance will have a low value. In other words, lower values indicate better balancing among subsystems and, in general, this happens when the access selection algorithm used has no or little preference for a specific subsystem.

Figure 3 shows the balance obtained by the algorithms for the analyzed network and applications configuration. One can note that the balance is different for the algorithms at the beginning of simulations when the network show low utilization. As the utilization grows for the subsystems, all algorithms tend to stabilize. After approximately 400 seconds, they maintain their values until the end of the simulation. *Random*, *LessVoice*, *BestFit*, *FirstFit* and *WorstFit* algorithms (in this order) obtain the best balance for the network under this scenario, with values ranging from 3.17 for *Random* to 1.22 for *WorstFit*. These low values indicate that the subsystems were used (nearly) equally.

Although considering the population all algorithms performed fairly equal, the blocking probabilities were different and actually determined the best algorithm, in this case *LessVoice*. This was somewhat expected, as *LessVoice* measures the resource consumption of each application class, giving a better estimation of the best subsystem.

3.2. Access Selection for Voice and Elastic Data Applications

In this section we discuss the dynamic scenario when elastic applications are present. We maintain the same parameters for subsystems and applications, except that in this scenario, applications can be slowed down (that is, its throughput can be reduced) to their minimum operational requirements when needed. It is expected that the subsystems and consequently the network populations will increase and the blocking probability will decrease as discussed in [1, 10]. In this scenario, we analyze the relationship between the network population gain and reduction of blocking probability, and the amount by which applications need to be slowed down in order to serve more applications.

It is now assumed that an application a has attributes $Min(a)$ and $Max(a)$, representing its minimum and maximum bandwidth requirement, respectively. The actual bandwidth that must be assigned by the network to a, defined as $Size(a)$, can be any value respecting $Min(a) \leq Size(a) \leq Max(a)$. When application a arrives, its alloca-

Algorithm 1 Allocation_Policy(a, N)

1. // Goal: Choose from N a subsystem to allocate application a using a specific access selection algorithm. If there is no space for a, applies a bandwidth sharing policy to redistribute resources
2. $C = \{s \in N \mid Max(a) \leq Free(s)\}$
3. **if** (C == Ø) {
4. // Shrinking is necessary.
5. $N' = \oslash$
6. **for each** $s \in N \mid Max(a) \leq MaxFree(s)$ **do** {
7. Create an empty subsystem s'
8. $Cap(s') = MaxFree(s)$
9. $N' = N' \bigcup s'$
10. }
11. s' = SELECTION(a, N')
12. let s be the correspondent in N for s' in N'
13. BWSHAREPOLICY(a, s)
14. } **else** {
15. // No need to shrink.
16. s = SELECTION(a, C)
17. Allocate application a in subsystem s
18. }

tion will be controlled by an "allocation policy", which allocates application a according to its maximum required bandwidth $Max(a)$ only if the network has that resource available, without changing the actual bandwidth of any $A(s)$. Otherwise, it must somehow "shrink" applications already allocated and also a, in order to allocate it. In such case, a new $Size(a_i)$ must be reassigned to each $a_i \in A(s) \cup \{a\}$ in the chosen subsystem s.

Since there are many different ways to find values for $Size(a_i)$, we adopted the term "Bandwidth Sharing Policy" to describe a particular mechanism used to accomplish the task by reassigning adequate values for their actual bandwidth share. In summary, the algorithm that implements the Allocation Policy must decide when applications must be slowed down (shrunk) and use a specific Bandwidth Sharing Policy (among a set of existing ones) to decide how they must be shrunk and how the resources from the network must be shared among them. We observe that only Elastic applications can be slowed down while voice calls always maintain their required bandwidth.

In this section we assume that $Size(data)$ can vary between $Max(data) = 150kb/s$ and $Min(data) = 37.5kb/s$. In order for a shrinking algorithm to be effective, the Bandwidth Sharing Policy should be work conserving. This means that applications should be shrunk as little as necessary to accommodate new applications in the system and be able to complete their task as early as possible. This also implies that, when any application leaves the network, the freed bandwidth must be redistributed (shared) among the others currently using that subsystem. Algorithm 1 describes how this approach is implemented.

In line 6 of Algorithm 1, $MaxFree(s)$ denotes the available space on subsystem s if all applications inside s were shrunk to their minimum required bandwidth. The idea

Algorithm 2 Brotherhood(a**,** s **)**

1. // Goal: Allocates the application a in subsystem s after shrink applications
2. $X = A(s) \bigcup a$
3. $MinReqRes(s) = \sum_{x \in X} Min(x)$
4. $MaxFree(s) = Cap(s) - MinReqRes(s)$
5. **if(** $MaxFree(s) \geq 0$ **)** {
6. $MaxContrib(s) = \sum_{x \in X} (Max(x) - Min(x))$
7. **for each** application $x \in X$ **do** {
8. $Share(x) = Max(x) - Min(x)$
9. $ShareIdx = \frac{Share(x)}{MaxContrib(s)}$
10. $Size(x) = Min(x) + MaxFree(s) * ShareIdx$
11. $Size(x) = Min\{Max(x),\ Size(x)\}$
12. }
13. Insert application a in subsystem s
14. } **else** {
15. Reject application a
16. }

behind lines 5-10 is to create a "virtual" network N' based on the original N, in which the free space for its subsystems reflects the free space that would be have in the original subsystems if applications were shrunk, that is, $MaxFree(s)$. Thus, the access selection algorithms do not need to be modified to work with elastic applications. In line 11, SELECTION(.) represents the access selection algorithm being used, which makes its decision based on the virtual network N'. In line 12, this decision is mapped back to the original network. In line 13, BWSHAREPOLICY(.) represents the algorithm that implements the actual bandwidth sharing policy, whose function is shrink applications already in subsystem s and also to insert the new application a in s. Both BWSHAREPOLICY(.) and SELECTION(.) can be any defined algorithm. In this work we evaluate the algorithms discussed in section 2.2. as access selection algorithms.

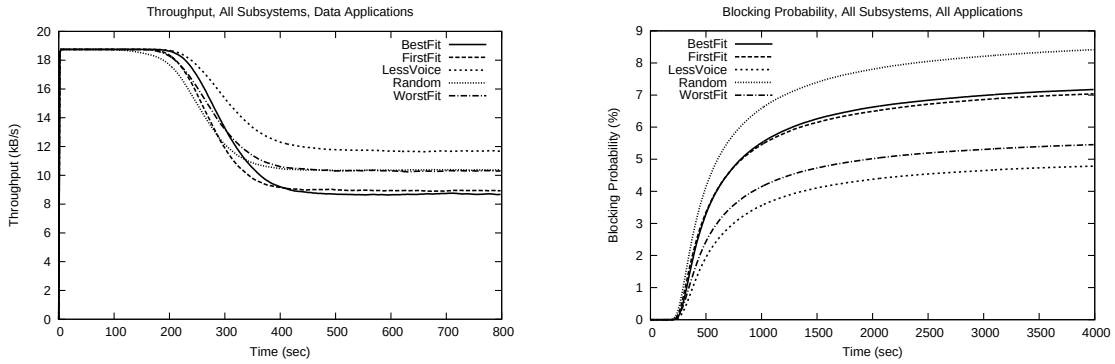
For the bandwidth sharing policy, we adopt the "Brotherhood" algorithm, described in Algorithm 2. Its name comes from the idea of sharing the available bandwidth in an equal manner, but proportional to the amount of resources that each application is about to share with other applications. The basic idea is that any excess bandwidth that would be needed by inserting a new application with its maximum requirements must be subtracted from all applications currently in the subsystem and also from a. The sum of the subtracted bandwidths is the required bandwidth and the amount subtracted from each application is proportional to the amount each application can contribute. In other words, it is assumed that $Share(a) = Max(a) - Min(a)$ is the amount of resources that an application a is willing to share. Then, the application will always receive from the network the minimum it requires plus an amount that is proportional to the value $Share(a)$. The update method used when any application leaves is based on the same mechanism.

Table 1 shows the results for all algorithms for the configured scenario with elastic data applications. In Table 1, *Population* refers to the average instantaneous population (higher is better) and *Balance* refers to the average balance of the network (les is better). *Blocking* refers to the average blocking probability (in percentile) considering both

Metric	Algorithm				
	BF	WF	FF	LV	RN
Population	270.00	271.97	270.29	272.00	262.22
Balance	0.72	0.65	0.36	1.49	0.67
Blocking (%)	7.18	5.46	7.04	4.79	8.41
Slow down (%)	53.63	44.89	52.37	37.49	44.49
Throughput (kB/s)	8.69	10.33	8.93	11.72	10.41

Table 1. Average results obtained by each algorithm in the scenario with voice and elastic applications after 4000 seconds

voice calls and data applications. *Slowdown* refers to the amount of bandwidth reduced by each algorithm from the maximum required by each data application, measured in percentile. The slowdown has a direct impact on the values shown for *Throughput* of elastic applications and, as shown in Table 1, the lower the slowdown, the higher the throughput.



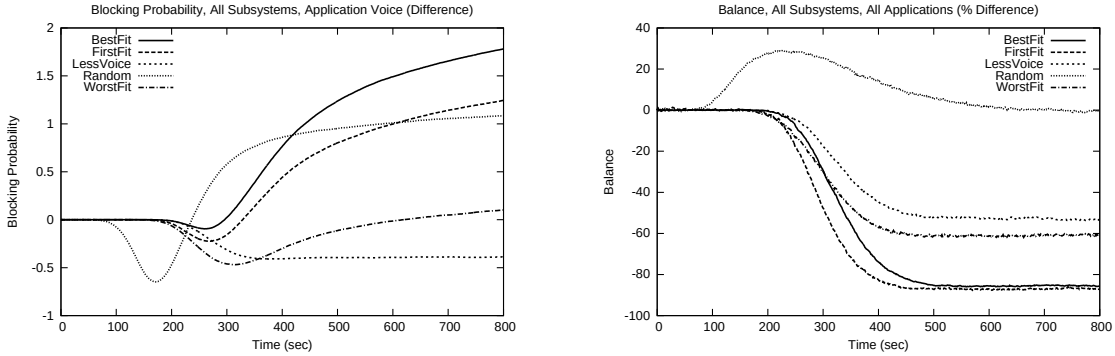
(a) **(b)**

Figure 4. (a) The throughput variation forced by each algorithm over the elastic applications; (b) Blocking probability obtained by each algorithm in a scenario with voice and elastic applications.

Figure 4(a) shows the evolution of the throughput of data applications considering the time. In the beginning of the network operation data applications are admitted in the network and receives the maximum required bandwidth, since there are resouces available. As time goes on, new applications and voice calls are admitted, the network becomes saturated and data applications are slowed down in order to accommodate the new sessions. It can be seen that algorithms *BestFit* and *FirstFit* achieved similar performance in terms of throughput, and the same can be observed for algorithms *Random* and *WorstFit*. Algorithm *LessVoice* forced less slow down over the applications and achieved the highest throughput among all the algorithms.

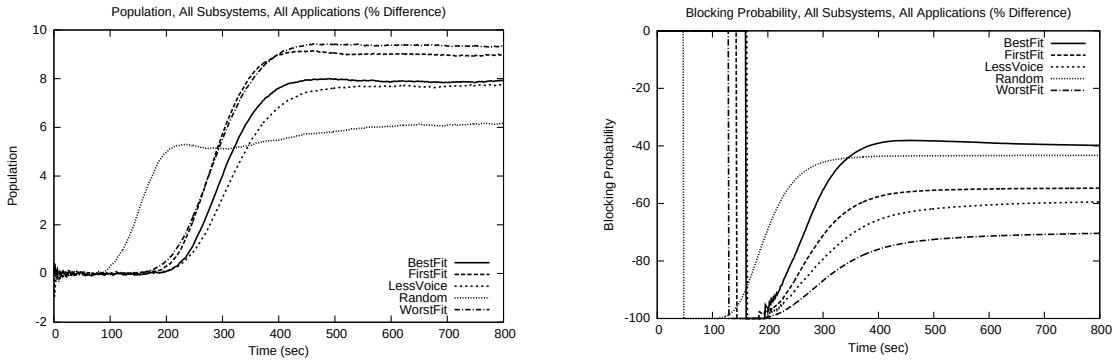
Figure 4(b) shows the blocking probability obtained by each access selection algorithm. Under the configured offered load, the algorithms *LessVoice*, *WorstFit*, *FirstFit*, *BestFit* and *Random* (in this order) obtained the best results. Comparing these results with those depicted in figure 2(a) (for non-elastic applications) it is clear that *WorstFit* outperformed both *BestFit* and *FirstFit* when elastic applications are considered. In general, the slow down forced over the elastic applications explains the blocking probabilities decreas-

ing for all algorithms. Despite that, when only voice calls are considered, the blocking probability actually increased for every algorithm except for *LessVoice*, as can be seen in Figure 5(a). It is worth mentioning that *Random* obtained higher throughput than *FirstFit* and *BestFit* and similar one to *WorstFit*, but this fact must not lead us to think on *Random* as a better algorithm, because it obtained actually the higher blocking probability, as can be seen in Table 1 and figure 4(b). Hence, the higher throughput achieved by *Random* was obtained by rejecting a higher number of aplications, that is, admitting a lesser number of applications and distributing higher amount of resources among them.



(a) **(b)**

Figure 5. (a) Difference $B - A$ for blocking probability of voice calls obtained by each algorithm, where B is the scenario with elastic applications and A is the scenario with non-elastic applications; (b) Relative difference $100(B - A)/A$ for the balance obtained by each algorithm, where B is the scenario with elastic applications and A is the scenario with non-elastic applications.



(a) **(b)**

Figure 6. (a) Relative difference $100(B-A)/A$ for the network population obtained by each algorithm (in percentiles), where B is the scenario with elastic applications and A is the scenario with non-elastic applications; (b) Relative difference $100(B - A)/A$ from blocking probability where B is the scenario with elastic applications and A is the scenario with non-elastic applications.

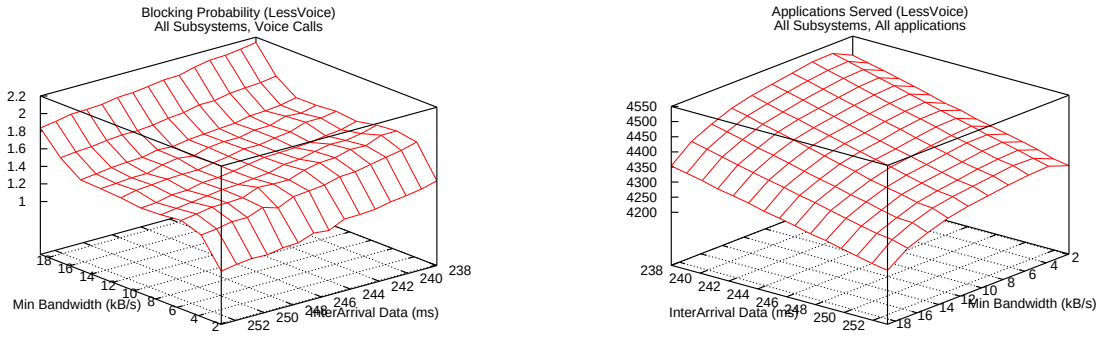
Compared to the scenario with non-elastic applications described in section 3.1., all algorithms performed better due to application slowdown. As can be seen in figures 6(a), 6(b) and 5(b), population size, blocking probability and balance were improved: population increase ranged from approximately 6% to 9%, blocking probabilities decrease

ranged between 40% and 75% and with the exception of the *Random* algorithm, balance decrease ranged between 45% and 85%.

The bandwidth sharing policy "adjusted" the allocation scheme used by each algorithm, yielding better performance for all of them and the better the allocation scheme, the less the algorithm needed to resort to sharing. It is clear from these results that the presence of elastic applications definitely increases the network capacity, in spite of elastic applications remaining allocated for a longer period, as observed in [1]; but from the results we obtained, the access selection algorithm used can bring additional benefits.

The small difference between the algorithms is interesting from a network point of view, because it wouldn't matter which algorithm was chosen in order to achieve a high population size and a low blocking probability, but from the user point of view this is problematic, since applications can end up using more or less bandwidth, depending on the chosen algorithm. As QoS is taken into account in the bandwidth sharing policy, the chosen algorithm (or algorithms) becomes an issue for best-effort elastic applications and in this context, the *LessVoice* algorithm is the best choice.

3.2.1. The impact of load and data elasticity on voice blocking and on the number of applications served



(a) **(b)**

Figure 7. (a) Impact of data load and data elasticity in the blocking probability of voice calls; (b) Impact of data load and data elasticity in the total number of applications served by the network

Since *LessVoice* has achieved the best results in the previous experiments, it is worth analysing its performance under different conditions. The previous results were obtained using arbitrarily picked values for voice and data arrival rate and minimum bandwidth size for elastic applications. In this section we vary these parameters to study the behavior of the *LessVoice* algorithm, in terms of voice call blocking probabilities and the total number of applications served by the network.

For this scenario, we maintain $\lambda(voice) = 1/532ms$ and observe the blocking probability obtained for voice calls when two parameters are compared at the same time. The first parameter is the minimum bandwidth required by data applications, which varies from $18.75kB/s$ to $1.875kB/s$ (that is, from $150kb/s$ to $15kb/s$), while the maximum

bandwidth required is kept in $18.75kB/s$. The second parameter is $\lambda(data)$, which varies from $1/252ms$ to $1/238ms$ in 15 steps. These values result in blocking probabilites for voice calls between 1.8% and 2.2% when no bandwidth sharing is employed.

As shown in the Figure 7(a), when the data applications arrive faster, voice calls have a greater blocking probability. This result is coherent since a low data arriving interval generates more data applications to compete with voice calls for the subsystems resources. Similar behaviour is observed as the minimum bandwidth requirement assumes greater values, meaning that when data applications are less elastic, they force an increasing in the blocking probability of voice calls (althought not shown here, it also occurs an increasing in the blocking probability of data applications as well). That is, the lower the minimum bandwidth requirement is, the smaller the resources the data applications consume, since the bandwidth sharing policy forces a slow down over the data application when needed.

The interesting about this comparison is to observe the relationship among data load, data elasticity and blocking probability of voice calls: linear variation in the data load have linear impact over blocking probability of voice calls. However, linear variation on data elasticity causes a non-linear variation in the probability of voice calls.

Figure 7(b) shows the number of applications served by the network (including voice calls and data applications), considering the same parameters variation discussed for Figure 7(a). One can observe that the number of applications served is inversely proportional to the data inter-arrival time and to minimum bandwidth requirement. In other words, the slower the data applications arrive and the smaller the required minimum bandwidth, the greater the number of applications served by the network. This is expected, since when data applications can be slowed down, more new applications can be accommodated in the subsystems.

However, it can be observed from Figure 7(b) that the number of applications served decreased when the minimum bandwidth requirement decreased from about $4kB/s$ to $2kB/s$ and, despite this sounds counterintuitive, this is in fact correct. The explanation is that this suggests the existence of a limit from which reducing the minimum bandwidth requirement of data applications does not lead to more applications being served, since data applications will spend excessive time inside the network.

3.3. Worst-case Analysis for specific input mixes

In the experiments conducted so far, applications arrived assuming equal input mix and according to a randomly interlaced sequence. Now, we change the way applications arrive in order to analyze the algorithms' behavior under a worst-case arrival pattern. We consider the maximum single-service applications for the network, that is, $Q(Combined, voice) = 275$ and $Q(Combined, data) = 110$. Then we submit $Q(Combined, k_1)$ applications of class k_1, followed by $\alpha Q(Combined, k_2)$ applications of class k_2, where $\alpha \in \{0.1, 0.2, ..., 1\}$, in such a way that the offered load to the network grows with α. Then, we use $k_1 = voice$, $k_2 = data$ in a first experiment and use $k_1 = data$, $k_2 = voice$ in a second one and observe the blocking probabilities obtained by each algorithm. Both voice and data have exponentially distributed holding time with mean $\mu(data) = \mu(voice) = 120s$.

Figure 8(a) shows the blocking probability obtained by the algorithms when all voice applications arrive before data ones. Algorithm *Random* obtained the worst perfor-

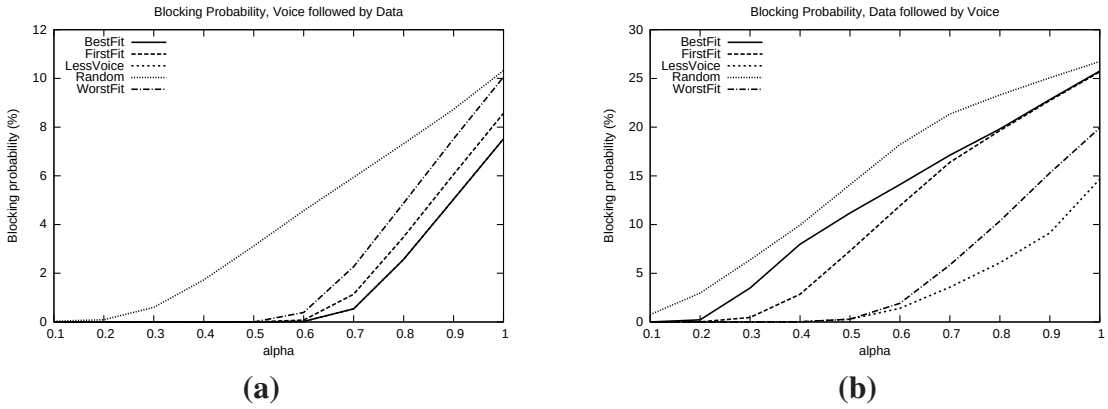


Figure 8. (a) Blocking probability obtained when 275 voice calls arrive before a fraction α of 110 data applications; (b) Blocking probability obtained when 110 data applications arrive before a fraction α of 275 voice calls.

mance, while *LessVoice* and *BestFit* shared equal best performances. In fact, when voice applications arrive, they are all allocated to GSM by both algorithms. *BestFit* allocates voice to GSM because it has less capacity and will leave less free space left after allocating the arriving voice call. *LessVoice* does the same because $Size(voice, GSM) < Size(voice, WCDMA)$, as discussed before. Next, all data applications are allocated to WCDMA by both algorithms, which explains the obtained results.

Figure 8(b) shows the blocking probability obtained by the access selection algorithms when all data applications arrives before voice. The algorithms *LessVoice*, *WorstFit*, *FirstFit* and *Random* obtained lower blocking rates, in this order. Both *WorstFit* and *LessVoice* performed similarly with almost no blocking until $\alpha = 0.5$, when *LessVoice* becomes better. This can be explained by the fact that when data applications arrives first, they are allocated to WCDMA by both algorithms: *WorstFit* selects it because there will be more free space left after allocating the latest arriving data application, while *LessVoice* selects it because the computed r (see section 2.2.) is smaller. When the amount of free space left by both subsystems becomes roughly the same, *WorstFit* starts selecting them in an alternate fashion, although there is still space left in WCDMA. *LessVoice* only starts selecting GSM when there is no more space left in WCDMA. This analysis reforced the assumption that algorithms that measure resource consumption yield better near-optimal packing with lower blocking probabilities, as the results from section 3.1. already inferred.

Now, we turn our attention to consider the same worst case arrival scenarios, but taking into account the presence of voice calls and elastic applications. When elastic applications were considered, it was observed that the blocking probability droped to zero for all algorithms, except *Random*. The fact that *Random* algorithm had not reduced the blocking probability like the others also shows that applying some intelligence to the access selection algorithm is also an important issue in order to obtain additional gains in multiaccess, multiservice networks in the presence of elastic applications.

4. Conclusion and Future Research

This paper presents a performance analysis of different access selection algorithms for ABC networks, which were modeled as a bounded space, variable-size online bin packing

problem. A dynamic environment was considered with the presence of both voice calls and data applications, in two scenarios: non-elastic and elastic data applications. Further, an algorithm was proposed to accomplish the task of sharing network resources among elastic applications in a work conserving way.

It becomes clear from the results that the presence of elastic applications increases the network capacity, even if they remain in the network for a longer period of time, which is in line with previous analytical results [1]. However, for the studied scenarios, our simulations indicated that there is a limit from which reducing the minimum bandwidth requirement of data applications does not lead to more applications being served by the network, since data applications will spend excessive time inside the network. An investigation towards finding this limit analytically could be an interesting research activity.

We also verified that, considering a network composed by subsystems with different capacities in a dynamic environment, the access selection algorithm used makes difference in terms of reducing blocking probability. Considering the scenarios evaluated, we observed that:

- Algorithm *LessVoice* obtained best performance in terms of blocking probability in the presence of both non-elastic and elastic applications.
- In the presence of voice calls and non-elastic applications, *LessVoice* obtained 0.075 of blocking probability. The relative increase from the others when compared to *LessVoice* were +7%, +40%, +58% and +105% for algorithms *BestFit*, *FirstFit*, *WorstFit*, and *Random*, respectively.
- In the presence of voice calls and elastic applications, *LessVoice* obtained 0.030 of blocking probability. The relative increase from the others when compared to *LessVoice* were +7%, +22.5%, +22.6% and +75% for *WorstFit*, *BestFit*, *FirstFit* and *Random*, respectively.
- The load balance of the network obtained for all algorithms were similar in the presence of both non-elastic and elastic applications.

As expected, all algorithms obtained less blocking probability in the presence of elastic traffic. When compared to the scenario with non-elastic applications, the observed reductions were -70%, -59%, -55%, -43% and -40% for algorithms *WorstFit*, *LessVoice*, *FirstFit*, *Random* and *BestFit*, respectively. Additionally, the blocking probability of all algorithms becomes closer, since they all benefit from the slow down of elastic applications and the bandwidth sharing mechanism. Considering only voice applications, it was noticed that the blocking probability increased for algorithms *BestFit*, *FirstFit* and *Random*, maintained almost the same for *WorstFit* and decreased only for *LessVoice*. In our studies this is shown to be dependent on four factors: a) the relative load for voice and data applications; b) the bandwidth requirements for voice and data applications; c) the maximum slow down rate allowed for data; and d) the access selection algorithm used.

We consider this work as an initial step and see interesting works ahead. In section 3.2. we adopted a bandwidth sharing policy that distributes the network resources among elastic applications considering their minimum required bandwidth and, if possible, distributes a "bonus" proportional to the distance between their minimum and maximum required bandwidth. In [3] the author proposes the "resource sharing with priority" policy, which establishes a sequence in which given classes of applications start being slowed

down before others. Therefore, comparing a set of different bandwidth sharing policies could be an interesting investigation.

References

[1] Eitan Altman. Capacity of multi-service cellular networks with transmission-rate control: a queueing analysis. In *MobiCom'02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 205–214, New York, NY, USA, 2002. ACM Press.

[2] J. Csirik and D. S. Johnson. Bounded space on-line bin packing: Best is better than first. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1991.

[3] Gábor Fodor. Performance analysis of the uplink of a cdma cell supporting elastic services. In *Boutaba et al. (eds), IFIP Networking 2005*, pages 205–216. Springer, 2005.

[4] Gábor Fodor, Anders Furuskar, and Johan Lundsjo. On access selection techniques in always best connected networks. In *ITC Specialist Seminar on Performance Evaluation of Wireless and Mobile Systems*, Aug 2004.

[5] Anders Furuskar and J Zander. Multiservice allocation for multiaccess wireless systems. *IEEE Transactions on Wireless Communications*, 4:174–184, Jan 2005.

[6] Anders Furuskär. *Radio Resource Sharing and Bearer Service Allocation for Multi-Bearer Service, Multi-Access Wireless Networks*. PhD thesis, Royal Institute of Technology (KTH), Radio Communication Systems, Dept of S3, Stockolm, Sweden, Apr 2003.

[7] A. Eriksson G. Fodor and A. Tuoriniemi. Providing qos in always best connected networks. *IEEE Communications Magazine*, 41(7):154–163, June/July 2003.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman, 1979.

[9] Eva Gustafsson and Annika Jonsson. Always best connected. *IEEE Wireless Communications*, 10(1):49–55, Feb 2003.

[10] I Koo, Anders Furuskar, J Zander, and Kiseon Kim. Erlang capacity of multiaccess systems with service-based access selection. *IEEE Communications Letters*, 8(4):662–664, Nov 2004.

[11] Steven S Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, Sep 2002.

[12] Steven S. Seiden, Rob van Stee, and Leah Epstein. New bounds for variable-sized online bin packing. *SIAM J. Comput.*, 32(2):455–469, 2003.

[13] Antti Tölli, Petteri Hakalin, and Harri Holma. Performance evaluation of common radio resource management (crrm). In *Proc. Int. Conf. Communications*, pages 3429–3433, Apr 2002.

[14] Bo Xing and Nalini Venkatasubramanian. Multi-constraint dynamic access selection in always best connected networks. In *Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous 2005)*, pages 56–64, Sep 2005.