Gomez-Miguelez *et al. EURASIP Journal on Wireless Communications and Networking* 2013, **2013**:59
http://jwcn.eurasipjournals.com/content/2013/1/59

EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

# Deployment and management of SDR cloud computing resources: problem definition and fundamental limits

Ismael Gomez-Miguelez*, Vuk Marojevic and Antoni Gelonch

**Abstract**

Software-defined radio (SDR) describes radio transceivers implemented in software that executes on general-purpose hardware. SDR combined with cloud computing technology will reshape the wireless access infrastructure, enabling computing resource sharing and centralized digital-signal processing (DSP). SDR clouds have different constraints than general-purpose grids or clouds: real-time response to user session requests and real-time execution of the corresponding DSP chains. This article addresses the SDR cloud computing resource management problem. We show that the maximum traffic load that a single resource allocator (RA) can handle is limited. It is a function of the RA complexity and the call setup delay and user blocking probability constraints. We derive the RA capacity analytically and provide numerical examples. The analysis demonstrates the fundamental tradeoffs between short call setup delays (few processors) and low blocking probability (many processors). The simulation results demonstrate the feasibility of a distributed resource management and the necessity of adapting the processor assignment to RAs according to the given traffic load distribution. These results provide new insights and guidelines for designing data centers and distributed resource management methods for SDR clouds.

## 1 Introduction

Wireless communications technology continuously improves and already facilitates the provisioning of a wide variety of advanced communications services at competitive prices. Whereas current systems provide data rates of a few mega-bits per second (Mbps), 4G systems will offer up to 100 Mbps per user. A few seconds may be necessary today before a connection is established between the user equipment and the network. Long term evolution (LTE) and LTE-Advanced (LTE-A) promise connection establishment times of less than 50 and 10 ms, respectively, [1,2].

Base stations are the wireless access points of cellular communications systems. They comprise antennas and analog and digital signal processing resources for implementing radio transmitters and receivers. The network operator deploys base station resources, that is, wireless transceivers, as a function of the expected peak load. The goal is guaranteeing a certain quality figure, for example, the probability of granting a user service request. Providing resources for the worst case scenario however leads to long idle times and resource inefficiencies because of the sporadic use of wireless communications services [3]. Deploying fewer resources would increase the mean resource utilization while increasing the user rejection probability. Base stations may be shared between radio operators, but temporarily unused resources can still hardly be reassigned for other purposes. Radio operators thus purchase, maintain, and update considerably more resources than needed for most of the time (Figure 1). We therefore suggest a more efficient and scalable infrastructure, the *SDR cloud* [4].

A software-defined radio (SDR) cloud comprises a set of distributed antenna sites that connect to one or several data centers through low-latency and high-bandwidth communication links. The antenna sites process the radio frequency (RF) signals and convert signals from analog to digital and vice versa. The digital data is processed entirely in the data center, employing SDR and cloud computing technology.

*Correspondence: ismael.gomez@tsc.upc.edu
Department of Signal Theory and Communications, Universitat Politecnica de Catalunya, Barcelona, Spain

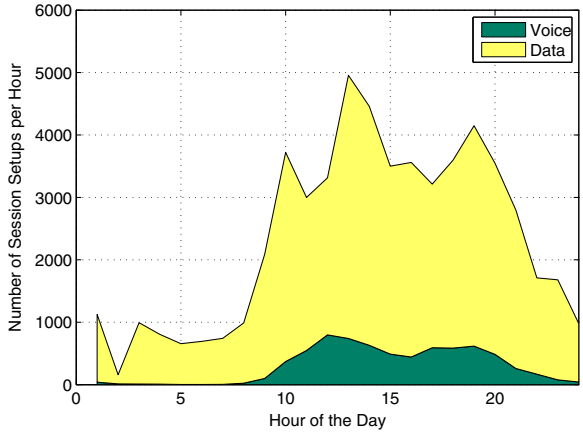


**Figure 1 Single sector traffic.** Typical weekday traffic pattern within an antenna sector in a city center.

SDR describes wireless transceivers that implement a significant part of the physical layer signal processing (DSP) in software that executes on general-purpose hardware [5]. SDR applications or waveforms define the transceiver functionality of future radio equipment. This facilitates dynamic reconfigurations or radio transceivers, changing their transmission modes through changes in the software.

Cloud computing provides IT services to clients without reference to the infrastructure that hosts the services [6]. The cloud is a generic platform for different business types, from small-scale to very-large scale. The upfront cost is minimal as the infrastructure is provided by the cloud operator, who rents resources to cloud clients. The elasticity of clouds permits business grows without long-term planning and resource preallocations. A pay-per-use business model on top of a virtualized computing resource pool enables resource sharing and on-demand resource provisioning. Computing resources (hardware and software) can then be dynamically allocated and efficiently used, ensuring faster amortization (CAPEX) and better scalability as well as savings in power consumption, security, maintenance and software licensing, among others (OPEX).

The SDR cloud provides essentially the same benefits as a general purpose cloud. It inherits the resource-as-a-service and pay-per-use business concepts: computing power (infrastructure as a service—IaaS), system software (platform as a service—PaaS), and applications (software as a service—SaaS) will be provided on demand and without knowledge of the physical location and types of CPUs, discs, software repositories, and so forth.

A single data center is shared between several radio operators and thousands of end users. (Some 100,000 user sessions may be active at the same time in a city of one million or more inhabitants.) Virtualization is employed for ensuring secure and fair resource sharing, where one radio operator—the SDR cloud *client*—is not aware of others using the same physical machines. Different business models or agreements are possible. A minimum set of resources may be guaranteed to each radio operator, for instance. The remaining or unused resources can then be shared—fairly or competitively—as a function of the market, environment, or policy, among others. This requires a flexible, though efficient computing resource management framework as a basis for the SDR cloud business. Such framework, in other words, plays an essential role for the deployment and operation of SDR clouds. It, particularly, needs to ensure real-time resource allocation and execution in dynamic environments with different resource and service constraints. This is the topic of this article.

This article elaborates a relation between the wireless communications system requirements or constraints and the SDR cloud computing resource management before deriving optimal solutions for the high-level resource provisioning. Each service request requires loading the corresponding transceiver waveform. Real-time resource provisioning and *hard* real-time execution needs to be ensured for seamless service provisioning. The SDR cloud resource allocator (RA) will therefore determine the mapping of waveforms to the available computing resources on demand and under stringent timing and resource constraints. We show that the maximum traffic load that a single RA can handle is limited. It is a function of the complexity of the resource allocation algorithm, the call setup delay, and the user rejection or blocking probability. The radio access technology specifies the maximum call setup delay, whereas the radio operator determines a blocking probability target. We introduce a general execution time model for characterizing the complexity of different resource allocation algorithms and derive expressions for the average call setup delay and maximum traffic load. The results show that SDR cloud data centers can be efficiently managed in a distributed way. They provide guidelines for designing data centers and distributed resource management methods for SDR clouds.

The rest of the article is organized as follows: After providing some background on computing resource management methods and algorithms (Section 2), we identify the problem (Section 3) and elaborate a RA complexity model (Section 4). In the central part of the article, we define and solve an optimization problem for assigning computing resources to an RA as a function of the environmental parameters (Section 5). We finally apply our solution for managing the resources associated to a single radio cell (Section 6) and multiple cells (Sections 7 and 8) under different wireless communications traffic characteristic.

## 2 Background

Massively parallel computing architectures will dominate the high-performance computing landscape. A platform with a large number of parallel processors is more suitable for executing many applications than a single powerful processor [7]. The high and heterogeneous computing demands of SDR applications, in particular, are executed more efficiently on a multiprocessing execution environment [8,9]. Empirical studies have shown that scheduling hard real-time tasks on many-core processors is challenging [10,11]. Sophisticated resource allocation algorithms are consequently necessary for managing the real-time computing demands and the limited computing resources.

Distributed computing has a long research record. The multiprocessor mapping and scheduling problem, in particular, has been vastly investigated in the heterogeneous computing context [12-14]. Heterogeneous computing refers to a coordinated use of distributed and heterogeneous computing resources [15]. It is similar to grid computing [16] or metacomputing [17].

It is well known that the computing resource allocation problem is NP-complete, in general [18]. Heuristic approaches were therefore proposed, presenting a polynomial relation between the problem size and the computing complexity. Grid or cloud computing RAs dispatch computing jobs or independent task for their distributed execution. Grid computing workloads exhibit little intra-job parallelism, the average job completion time is several hours, and typical job inter-arrival times are in the order of seconds or minutes [19]. Many grid or cloud workloads are data-intensive [20].

Grids and clouds are accessed via the internet, which is relatively slow and has unpredictable delays. They were originally built for providing very high computing power for scientific or popular applications with no stringent real-time constraints. Rather than ensuring real-time allocation and execution, grid or cloud RAs therefore follow other objectives. Doulamis et al. [21], for example, discuss the fair sharing of CPU rates and allocate resources to users as a function of resource availabilities, user demands, and socio-economic values. Lui et al. [22] focus on the joint resource allocation of computing and network resources in federated computing and network systems. They present various resource allocation schemes that can provide performance and reliability guarantees for modern distributed computing applications. Entezari-Maleki and Movaghar [23] develop a probabilistic task scheduling method for minimizing the mean response time of grid jobs.

The SDR cloud concept has been recently introduced [4] and merges three fundamental technologies: centralized baseband processing, automatic computing resource allocation and virtualization. Related study addresses centralized baseband processing [24,25] and offline, that is, design-time resource allocation [26]. We focus on automatic computing resource allocation, enabling runtime resource management and seamless real-time execution. Each wireless communications service request needs to be served in real time, providing sufficient computing resources for the continuous real-time data processing. Two general approaches exist for scheduling real-time tasks on multiprocessor platforms. Tasks can be statically assigned prior to execution or migrate between processors during execution. The former can be achieved through partitioned scheduling, where an application is partitioned among the processing elements (mapping) before being locally scheduled. The latter approach is typically associated with global or dynamic scheduling. The contention for the global scheduling queue and non-negligible migration overheads among processing elements can result in significant scheduling overheads in practice [10]. The migration cost limits the number of cores that a global scheduler can manage [10,11]. Non-preemptive static partitioned scheduling, on the other hand, is pertinent to high performance many-core and multiprocessor platforms. It facilitates implementation and introduces low run-time resource overheads [27].

A constant execution period and practically deterministic and regular execution patterns characterize SDR applications. The real-time constraints of the DSP processing chains can then be given as minimum throughput and maximum latency constraints and static schedulers can be employed [8]. The mapping and scheduling can thus be calculated only once for each waveform as part of the session establishment process. The SDR cloud resource management performance is then limited by the RA's execution time per invocation (user session request) and the session arrival rate. The derivation of this limit is the objective of this article.

## 3 Problem formulation

Wireless subscribers access communications services anywhere, anytime, and under different circumstances. Measurements have shown that the average user establishes seven or eight voice sessions per day of 90 s in the mean [28]. Data users realize a larger number of shorter sessions. The number of concurrent sessions in a large city may range between 10,000 and 120,000 as a function of place and time.

The SDR cloud RA needs to be able to handle the spatial and temporal variety in the traffic load. A single data center ideally executes all waveforms and centrally manages all session requests. The corresponding RA then needs to be able to dispatch thousands of requests per second.

Modern wireless communications standards, however, impose restrictions on the maximum session establishment time $t_s^{\max}$. The call setup delay $t_s$ is the transition

time from a dormant (camping or idle [1]) state to the transmission or reception state. Each session establishment here consists of allocating sufficient computing resources to the corresponding transceiver waveform. The shorter the call setup time the better the *always connected* illusion. LTE-A therefore establishes 10 ms as the target call setup delay. Wireless operators moreover define a maximum blocking probability target $p_b^{\max}$, which should be satisfied in the mean. The blocking probability $p_b$ denotes the probability of a user session request being rejected due to insufficient computing resources. Wireless communications systems need to be accordingly dimensioned.

The session establishment time and blocking constraints determine the RA capacity in terms of manageable users. The number of users that can be concurrently served is directly proportional to the available processing resources. More processors ensure a lower $p_b$, whereas fewer processors a shorter $t_s$. The objective of this article is analyzing the relation between the RA capacity and the call setup time and blocking probability. We identify fundamental SDR cloud management limits and indicate possible SDR cloud data center design and management solutions. Table 1 summarizes the parameters that are used throughout the paper.

## 4 RA complexity model

The algorithmic complexity of any RA is a function of the number of tasks $m$ and the number of processing cores or nodes $n$. A polynomial expression can be used for modeling the complexity of practical RAs, such as

$$t_{\mathrm{RA}}(n,m) = Fn^{\alpha}m^{\beta}. \tag{1}$$

Parameters $\alpha$ and $\beta$ specify the complexity order of a RA. The same expression also serves as a general execution time model of an RA implementation. Parameters $F$, $\alpha$, and $\beta$ can be found by measuring the RA execution time for different $n$ and $m$ and then performing model fitting. Although other models may be more accurate for certain RA algorithms, (1) is simple and general.

Without loss of generality, we suggest a simple and well-known algorithm for providing numerical examples for the analysis performed in this article. The $g$- or greedy-mapping [8] is a baseline mapping algorithm. It maps one process after another, choosing the processor that leads to the minimum mapping cost. Cost metrics are therefore computed based on a suitable cost function. The cost function we suggest manages the limited processing and interprocessor bandwidth resources and, accordingly, distributing the processing load while minimizing the data flows between processors [8].

The algorithm is implemented in C and available as open source code [29]. Measuring the execution time of our implementation as a function of $n$ and $m$ and performing non-linear least-squares model fitting leads to $F = 3.98 \cdot 10^{-9}$, $\alpha = 2.94$ and $\beta = 1.04$. We can thus approximate the execution time model of the $g$-mapping algorithms as

$$t_{\mathrm{RA}}(n,m) \approx 4mn^3[\,ns\,]. \tag{2}$$

The $g$-mapping execution time thus increases linearly with the number of waveform modules $m$ and with the number of processors $n$ cubed. Figure 2 plots the

**Table 1 Description of parameters**

| Parameter | Description |
|---|---|
| $n$ | Number of nodes or processors |
| $m$ | Number of waveform modules or tasks |
| $t_s$ | Call setup delay |
| $t_s^{\max}$ | Call setup delay constraint |
| $p_b$ | Blocking probability |
| $p_b^{\max}$ | Blocking probability constraint |
| $t_{\mathrm{RA}}$ | Resource allocator's (RA's) execution time model |
| $F$ | Scaling factor of RA model |
| $\alpha$ | Nodes' exponent ($n^{\alpha}$) of RA model |
| $\beta$ | Modules' exponent ($m^{\beta}$) of RA model |
| $\theta$ | Cost function's weight |
| $\rho$ | Traffic load in Erlangs |
| $\rho^{\max}$ | Maximum traffic load a single RA can manage |
| $\lambda$ | Average session initiation requests per second |
| $1/\mu$ | Average session duration in seconds |
| $\Phi(n)$ | Number of users that can be served with $n$ processors for a given waveform model |

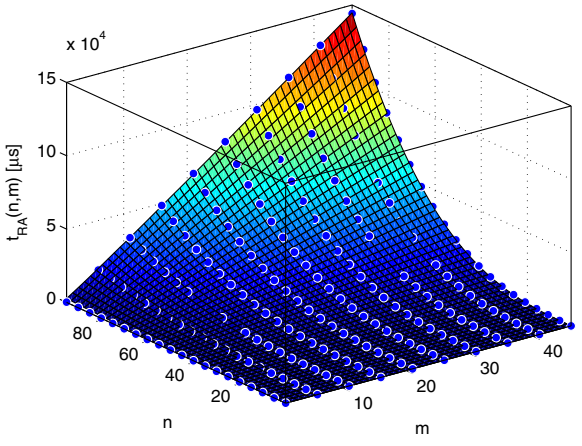


**Figure 2 Execution time model.** Measured and modeled execution times of the $g$-mapping RA.

execution time measurements together with the least-squares model.

## 5 Resource provisioning

Throughout this section, we will use the previously derived execution time model (2). The analysis is also valid for other RA complexity models provided that the complexity increases with the number of processors.

### 5.1 Optimization problem

We analyze the relation between the call setup delay, the blocking probability, and the RA capacity. To this aim, we derive the optimal number of resources for processing user signals as a function of the environmental conditions and constraints. The wireless communications traffic model is a stochastic birth-death process. The time between consecutive session establishments follows a Poisson distribution with a mean of $1/\lambda$. That is, $\lambda$ corresponds to the average number of new user session requests per second. The session duration follows an exponential distribution, where $1/\mu$ corresponds to the average session duration in seconds. The traffic load is then $\rho = \lambda/\mu$ Erlangs.

We assume that a single RA needs to handle $\rho$ Erlangs of traffic. The objective is then determining the optimal number of processors $n$ that satisfies the system constraints $t_s^{\max}$ and $p_b^{\max}$. This value is obtained as the solution to an optimization problem maximizing the following objective function:

$$f(n) = (1-\theta)\, U(n) - \theta \frac{n}{\rho},\ \rho > 0, \tag{3}$$

$U(n)$ is the average number of users that can be served with $n$ processors. Function $f(n)$ weights off the benefit (average number of served users) and the cost (allocated resource per Erlang). Parameter $\theta$ weights the importance of one term with respect to the other. Equation (3) allows minimizing the number of allocated resources $n$ ($\theta = 1$) or maximizing the average number of served users $U(n)$ ($\theta = 0$). Applying Little's law, we can express $U(n)$ as

$$U(n) = (1 - p_b(n))\, \rho. \tag{4}$$

The optimization problem can then be formulated as follows:

$$\begin{aligned} \max_{n} \quad & f(n) \\ \text{s.t} \quad & p_b(n) \le p_b^{\max} \\ & t_s(n) \le t_s^{\max} \\ & n \in \mathbb{N}. \end{aligned} \tag{5}$$

Before solving this problem, we first need to model the call setup delay $t_s(n)$ and blocking probability $p_b(n)$ constraints.

### 5.2 Constraints

The session establishment process can be modeled as a double-queuing process: New users enter an infinite queue whose service time is the execution time of the RA, that is, $t_{\mathrm{RA}}(n,m)$. They leave this *allocation queue* and enter a second multi-server queue of size $c$. The service time of the *active sessions queue* is exponentially distributed with an average of $1/\mu$, which corresponds to the average session duration.

This model can be represented by a two-dimensional state transition diagram, where state probability $p_{i,j}$ indicates the probability that there are $i$ users waiting for the allocation queue while $j$ users have active sessions. The model can be simplified if we consider that the mapping time is much shorter than the average session duration, that is, $t_{\mathrm{RA}}(n,m) \ll 1/\mu$. This allows separating the two queues. Following Kendall's representation, we model the allocation queue as an infinite length $M/D/1$ queue and the active sessions queue as a blocking and finite-size $M/M/c/c$ queue with no wait sates. For simplifying the mathematical analysis, here we consider waveforms of $m = 10$ tasks and analyze $t_{\mathrm{RA}}$ as a function of $n$.

#### 5.2.1 Call setup delay

User session requests are random and independent from one another. The random session requests lead to random session establishment times. The call setup delay constraint (5) will thus be satisfied on average despite the deterministic mapping time $t_{\mathrm{RA}}(n)$. Applying the Poisson arrivals see time averages (PASTA [30]) property, we know that $t_s(n)$ follows a Poisson distribution. According to the $M/D/1$ model, the average call setup delay then becomes

$$t_s(n) = \begin{cases} \frac{2-\lambda t_{\mathrm{RA}}(n)}{2(1-\lambda t_{\mathrm{RA}}(n))/t_{\mathrm{RA}}(n)} & \text{if } \lambda t_{\mathrm{RA}}(n) \le 1 \\ \infty & \text{if } \lambda t_{\mathrm{RA}}(n) > 1. \end{cases} \tag{6}$$

This function is monotonically increasing with $n$ (Figure 3) for $\lambda t_{\mathrm{RA}}(n) \le 1$. The system becomes unstable and the average waiting time infinite beyond that point. Figure 3 shows that the call setup delay limits the maximum number of processors that can be managed. For $t_s(n) = 100$ ms and $\lambda = 10$ user arrivals per second, for example, up to 150 processors can be managed with the $g$-mapping RA, but less than 100 processors for $\lambda = 100$. Figure 3, moreover, shows that a low $t_s(n)$ significantly limits the RA capacity.

#### 5.2.2 Blocking probability

The blocking probability of the active sessions queue ($M/M/c/c$ queue) is the probability that $c$ users are

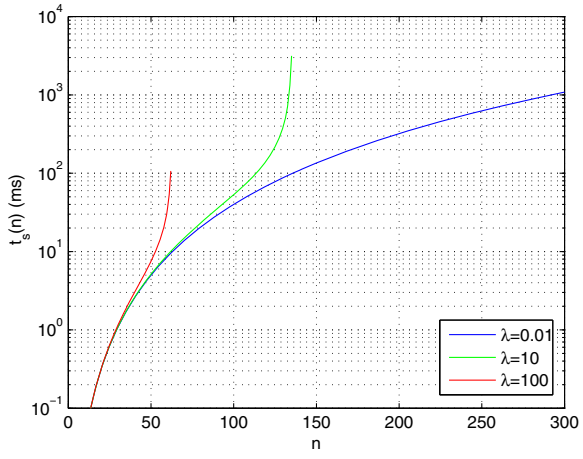


**Figure 3 Average waiting time.** Average waiting time $t_s(n)$ as a function of $n$ and $\lambda$ for $t_{RA}(n, m = 10)$.

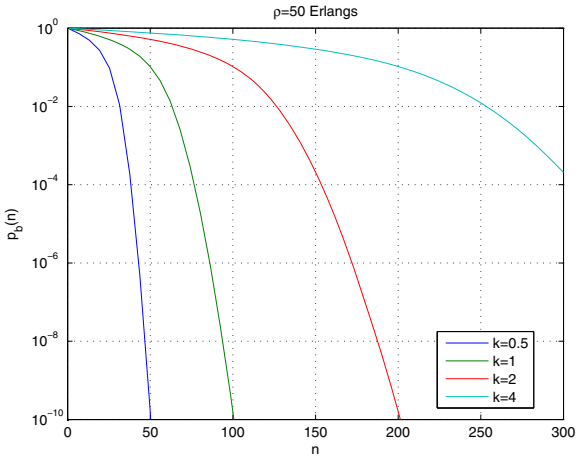


**Figure 4 Blocking probability.** Blocking probability $p_b(\rho, n/k)$ as a function of $n$ for different $k$.

occupying all available resources. When this happens, a new user is rejected due to insufficient computing capacity. Parameter $c$ therefore represents the maximum number of waveforms that can be loaded to $n$ processors. This number is difficult to characterize since depending on many factors, including the computing capacity of each processor, the interprocessor communication network, the waveforms' computing characteristics, and the performance of the RA algorithm.

For an analytical treatment the capacity of the queue needs to be abstracted. We propose defining $c = \Phi(n)$, which defines the maximum number of users that $n$ processors can accept. Without loss of generality, we assume the linear model $\Phi(n) = n/k$. Parameter $k$ is a real positive value and indicates the percentage of a single processor that is needed for executing a waveform. For $k > 1$, more than one processor is required for processing a single-user digital transceiver. For $k = 1.8$, for instance, one waveform requires 180 % of the processing resources of a single processor for real-time execution. Note that $U(n)$, which provides the average number of users that can be loaded to $n$ processors, depends on the traffic load and blocking probability, whereas $\Phi(n)$ essentially depends on the processor capacity, waveform characteristics, and RA algorithm efficiency. The blocking probability of the $M/M/c/c$ queue with $c = \Phi(n)$ is then

$$p_b(\rho, \Phi(n)) = \mathrm{B}(\rho, \Phi(n)), \tag{7}$$

where $B(\rho, c)$ is the Erlang-B function [30] for $\rho$ Erlangs and $c$ servers. Figure 4 indicates the evolution of the blocking probability as a function of $n$ for different $k$.

### 5.3 Solution

The objective function (3) is strictly concave because the blocking probability (7) is strictly convex [31]. This ensures that the optimization problem has a unique solution. Figure 5 plots the objective function $f(n)$ for different weights $\theta$.

The solution is trivial for $\theta = 0$ or $\theta = 1$, because the objective functions and the constraints are monotonic with $n$ over the entire range of processors. More precisely, $p_b(n)$ decreases and $t_s(n)$ increases with $n$. We therefore define $n_{\min}$ as the minimum number of processors that satisfies the blocking probability constraint $p_b^{\max}$ and $n_{\max}$ as the maximum number of processors that meets the call setup delay limit $t_s^{\max}$. That is, $n_{\min}$ and $n_{\max}$ limit the number of processors to a range that provides the desired quality. They satisfy

$$p_b(n_{\min}) \leq p_b^{\max},\ p_b(n_{\min} - 1) > p_b^{\max} \tag{8}$$

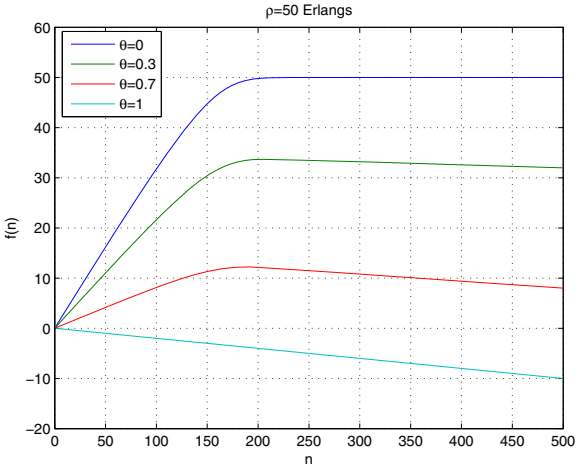


**Figure 5 Objective function.** Objective function $f(n)$ for $\rho = 50$ for different $\theta$.

and

$$t_s(n_{\max}) \leq t_s^{\max},\ t_s(n_{\max}+1) > t_s^{\max}. \tag{9}$$

We can say that $n = n_{\min}$ processors minimize the number of resources, whereas $n = n_{\max}$ processors minimize the blocking probability, or maximize the average number of concurrently served users, while still satisfying the call setup delay constraint. If, zhowever, $n_{\min}$ exceeds $n_{\max}$, the two constraints cannot both be satisfied and the problem becomes unsolvable. The solutions that minimize the blocking probability ($\theta = 0$) and the allocated processors ($\theta = 1$) then become

$$n^*_{\theta=0} = \{n_{\max} \mid n_{\min} \leq n_{\max}\} \tag{10}$$

$$n^*_{\theta=1} = \{n_{\min} \mid n_{\min} \leq n_{\max}\}. \tag{11}$$

We need to solve the problem numerically for arbitrary $\theta$. The first option is using numerical optimization. Integer optimization problems are very complex to solve, though. We therefore relax the integer nature of the optimization variable $n$ and use a convex solver for finding a non-integer solution. We then evaluate the objective function for the two closest integers, choosing the maximum that satisfies the constraints.

The Erlang's $\mathrm{B}(\rho, c)$ function is defined for natural $c$. The Erlang's extended B-formula is a continuous representation of the Erlang-B function based on the incomplete Gamma function. Computing this function numerically however requires numerical integration. We rather propose using the recursive method

$$\mathrm{B}(\rho, i) = \frac{\rho \mathrm{B}(\rho, i-1)}{\rho \mathrm{B}(\rho, i-1) - 1}, \tag{12}$$

where $i$ is a real positive number. If we are able to obtain $\mathrm{B}(\rho, z)$ for a real number $z < 1$, then we can compute $\mathrm{B}(\rho, i)$ for any $i$. Various approximations for $\mathrm{B}(\rho, z)$ have been published based on parabolic interpolations. We used the expression of [32] for the numerical examples that follow.

### 5.4 Numerical examples

More than one processor is typically needed for executing a modern waveform consisting of 10 or more tasks [8]. The numerical examples therefore consider $\Phi(n) = n/2$ allocatable users, $m = 10$ waveform tasks, and $1/\mu = 40$ s average data session duration. We use the interior-point numerical algorithm for solving problem (5) and obtaining a non-integer solution (Figure 6).

Assigning $n^*$ processors to the RA maximizes the system efficiency $f(n)$. The optimal number of processors $n^*$ is a function of $\theta$. The curve corresponding to $\theta = 0$ represents the solution that minimizes the blocking probability ($n^* = n_{\max}$) while meeting the call setup delay constraint. The curve corresponding to $\theta = 1$, at the other extreme, indicates the solution that minimize the use of processing resources ($n^* = n_{\min}$) while satisfying the blocking probability constraint. The intersection of the two curves provides the maximum system capacity $\rho^{\max}$. Parameter $n_{\min}$ becomes larger than $n_{\max}$ beyond that point and the problem has no solution.

The system capacity is almost 50 Erlangs for a call setup delay constraint of 50 ms, which corresponds to the LTE standard specification (Figure 6a,b). LTE-A indicates call setup times of 10 ms, reducing the RA capacity to some 25 Erlangs in this case (Figure 6c,d). The capacity can be improved by using more powerful processors. Assuming $\Phi(n) = n/1.5$, for example, leads to $\rho^{\max} = 35$ Erlangs for the LTE-A case (Figure 6e,f).

## 6 RA capacity

The previous section has indicated that the RA capacity $\rho^{\max}$ is finite. Here we analytically derive this limit. The manageable number of processors is obtained from the tolerable execution time. The blocking probability then determines the RA capacity.

### 6.1 RA execution time limit

The tolerable RA execution time $t_{\mathrm{RA}}^{\max}$ is a function of the call setup delay constraint and the user arrival rate. It is obtained assuming that the average call setup delay (6) is equal to the call setup delay constraint $t_s^{\max}$:

$$t_{\mathrm{RA}}^{\max} = \lambda^{-1} + t_s^{\max} - \sqrt{\lambda^{-2} + (t_s^{\max})^2} \tag{13}$$

Equation (13) can be simplified when $t_s^{\max}$ is either considerably smaller or considerably larger than the user inter-arrival time:

$$t_{\mathrm{RA}}^{\max} \approx \begin{cases} \lambda^{-1} & \text{if } t_s^{\max} \gg \lambda^{-1} \\ t_s^{\max} & \text{if } t_s^{\max} \ll \lambda^{-1}. \end{cases} \tag{14}$$

When $t_s^{\max} \gg \lambda^{-1}$, the capacity is limited by the stability of the $M/D/1$ mapping queue (see (6)). The call setup delay is then dominated by the time the user needs to wait before being be served rather than the RA execution time itself. On the other hand, when $t_s^{\max} \ll \lambda^{-1}$ the capacity is limited by the call setup delay constraint. This is the case with modern communications standards, such as LTE and LTE-A, where the call setup delay is dominated by the RA execution time.

The general expression of $t_{\mathrm{RA}}^{\max}$ is a function of $\lambda$ (13). Therefore, $n_{\max}$ is also a function of $\lambda$.

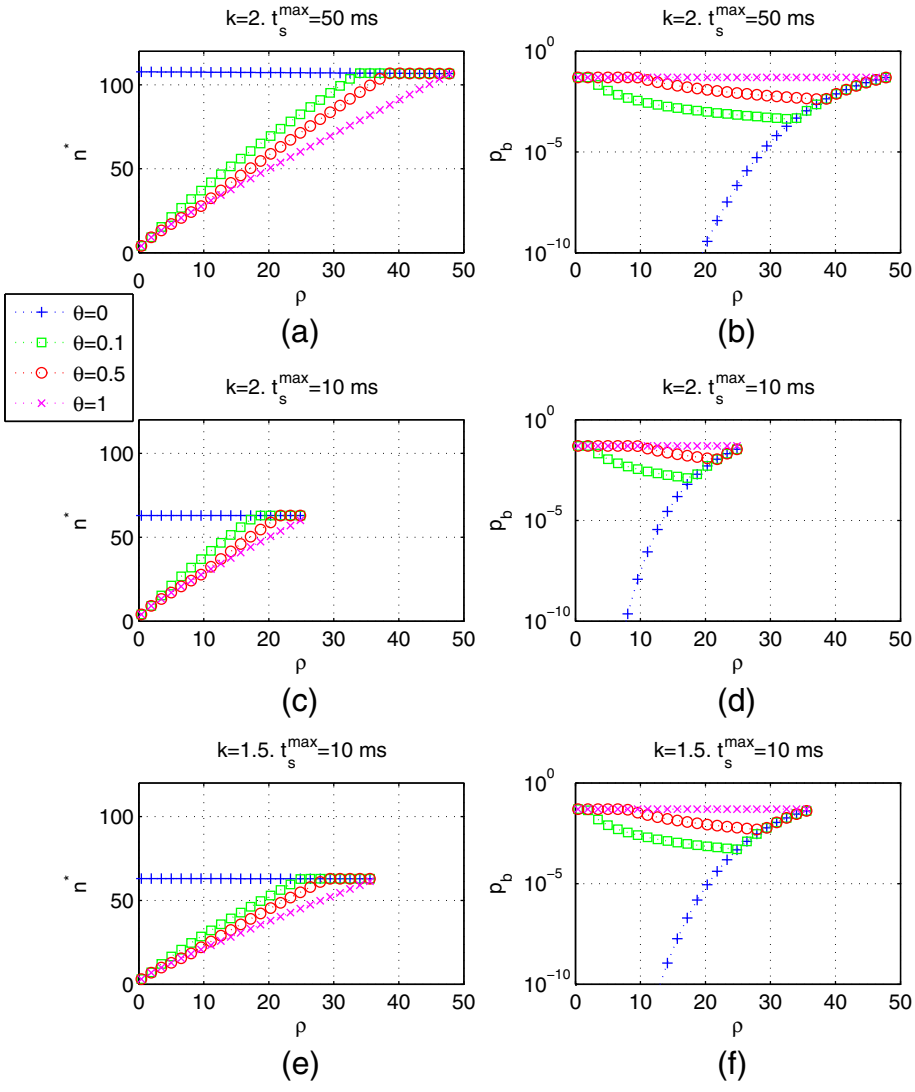


**Figure 6 Optimal processor allocation.** Optimal number of processors **(a, c, e)** and corresponding blocking probability **(b, d, f)** for different $t_s^{\max}$ and $k$.

### 6.2 Processor limit

The maximum number of processors that a RA can manage is a function of $t_{\mathrm{RA}}^{\max}$ and follows from inverting Equation (1):

$$n_{\max} = \left\lfloor \sqrt[\alpha]{\frac{t_{\mathrm{RA}}^{\max}}{Fm^{\beta}}} \right\rfloor. \tag{15}$$

The expression $\lfloor \cdot \rfloor$ indicates rounding off to the closest integer value.

### 6.3 Traffic limit

Considering the blocking probability constraint, the maximum traffic load $\rho_{\max}$ that the RA can manage is then the solution to

$$\begin{aligned} \mathrm{B}\left(\rho_{\max}, c\right) &= p_b^{\max} \\ \Phi\left[n_{\max}(\rho_{\max}\mu)\right] &= c. \end{aligned} \tag{16}$$

The capacity in Erlangs is thus a function of the average user session duration $\mu$. The expression can be simplified when $t_s^{\max} \ll \lambda^{-1}$. The maximum number of processors $n_{\max}$ then depends only on the maximum call setup delay constraint and the capacity becomes independent of $\mu$. The maximum traffic load $\rho_{\max}$ that the RA can manage is then the solution to

$$\mathrm{B}\left(\rho_{\max}, \Phi\left(n_{\max}\right)\right) = p_b^{\max}, \text{ when } t_s^{\max} \ll \lambda^{-1}. \tag{17}$$

Figure 7 plots the capacity of the $g$-mapping RA, assuming that $k = 1.5$ and $k = 2$ processors, are needed for

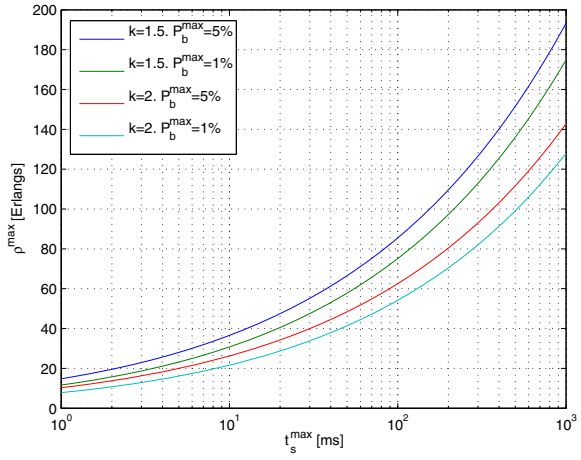


**Figure 7 Resource allocator capacity.** Capacity of the *g*-mapping RA as a function of $t_s^{\max}$ and *k*.

digitally processing the signals of a single user. The figure assumes the approximation $t_s^{\max} \ll \lambda^{-1}$ and, thus, is the solution to (17). It shows that a single RA can handle up to 200 Erlangs, assuming legacy cellular communications standards, which are characterized by loose call setup delay constraints. However, the capacity drops way below 80 Erlangs for the emerging LTE and LTE-A standards, which establish maximum session establishment delays of 50 and 10 ms.

## 7 Distributed resource management

SDR clouds will provide wireless communications services to very wide service areas and will, consequently, need to manage huge traffic loads. Incoming user session requests will then need to be assigned to different RAs. Each RA will absorb only a portion of the total traffic demand, managing part of the data center resources. The assignment of processors to RAs should adapt to the traffic load distribution while satisfying the constraints of (5).

Here we assume a reduced SDR cloud model, where a data center of $N = 100$ processors serves two radio cells with a total traffic load of 40 Erlangs. The maximum call setup delay is 10 ms and the target blocking probability 5%. The capacity limit of a single RA is 35 Erlangs for $\Phi(n) = n/1.5$. We thus need at least two RAs, one per radio cell. The problem then consists of splitting the $N$ processors between the two RAs in such a way that all constraints are satisfied. Problem (5) thus extends to

$$\begin{aligned} \max_{\{n_1,n_2\}} \quad & f(n_1)+f(n_2) \\ \text{s.t} \quad & p_b(n_i) \le p_b^{\max},\ i=1,2 \\ & t_s(n_i) \le t_s^{\max},\ i=1,2 \\ & n_1+n_2 \le N \\ & n_1,n_2 \in \mathbb{N} \end{aligned} \tag{18}$$

Parameters $n_1$ and $n_2$ represent the number of processors allocated to RA1 and RA2. Figure 8 shows the optimal solution for RA1. The plot of $n_2^*$ is symmetrical to $\beta = 0.5$. The traffic of each cell is $\rho_1 = \epsilon\rho$ and $\rho_2 = (1-\epsilon)\rho$, where $\rho = 40$ Erlangs and $0 \le \epsilon \le 1$.

For $\theta = 0$ all processors will be employed for maximizing the sum of $U(n)$ (see (3)). The processors are distributed between the RAs depending on the slope of the Erlang-B function. For $0.1 \le \epsilon \le 0.3$ and $0.7 \le \epsilon \le 0.9$ more processors are assigned to the cell with higher traffic load. This is different for $0.3 \le \epsilon \le 0.7$, because assigning more processors to the cell with lower service demand decreases the overall blocking probability. For $\epsilon \le 0.1$ or $\epsilon \ge 0.9$, the traffic of one or the other cell exceeds the corresponding RA capacity and the problem becomes unfeasible. The deployment of additional RA are necessary for such traffic distributions.

When $\theta = 1$, the number of processors is directly proportional to the traffic load, because the slope of the objective function is constant with $n$. For $0 < \theta < 1$, the resources are allocated as a function of the performance increment in relation to the amount of allocated resources. The number of allocated processors linearly increases with $\epsilon$ and $(1-\epsilon)$, respectively, until reaching the maximum number of processors $n_{\max}$ that still meets the session establishment delay constraint.

## 8 Simulation results

We simulate a non-homogeneous traffic demand, where the user session initiation and termination is modeled as a Poisson arrival and departure process. The user arrival rate is 4 times the departure rate, simulating an unstable situation for better analyzing the performance of the

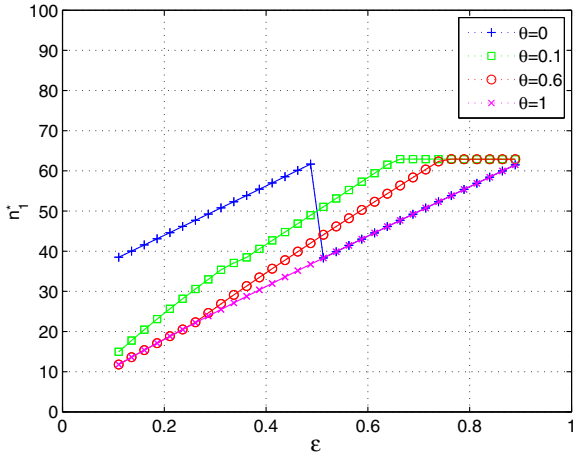


**Figure 8 Optimal processor distribution for two RA.** Optimal distribution of processors to two *g*-mapping RAs as a function of $\epsilon$ for different $\theta$ ($t_s^{\max} = 10$ ms $p_b^{\max} = 5\%$, and $\Phi(n) = n/1.5$).

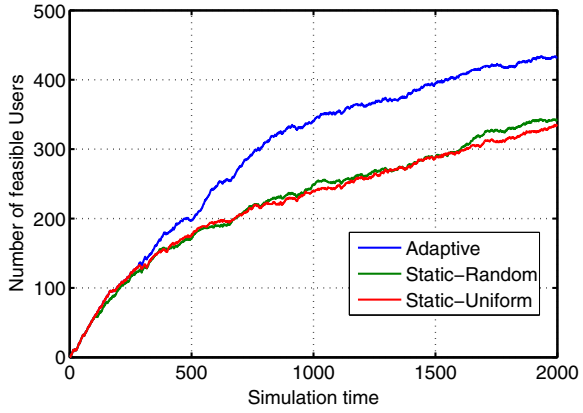


**Figure 9 Total number of served users.** Accepted user sessions over simulation time for 256 processors, 16 RAs and a heterogeneous traffic distribution.

different strategies. The *adaptive strategy* solves Equation (18) with 16 RAs instead of 2. The *static strategy* does not track the traffic load distribution, but rather assigns 16 of the 256 processors to each RA. The second variant of the static strategy randomly distributes the 256 processes among the RAs.

Each processor has a capacity of 12 giga-operations per second (GOPS). The waveforms offer 64, 128, 384, and 1024 kbps data rate services, which are solicited with a probability of 0.5, 0.2, 0.2, and 0.1, respectively. The four waveform models are those from [4], requiring between 50 % and 75 % of a processor's capacity ($k < 1$). The users follow a two-dimensional Gaussian distribution, centered and with a variance of 0.25 relative to the service area.

The blocking probability constraint is dropped in order to enable a fair evaluation between the three strategies. The optimal strategy then maximizes the number of served users ($\theta = 0$). The session initialization constraint is 50 ms and the average session duration 40 s.

Figure 9 shows the accumulated number of accepted user session requests over time. The adaptive strategy assigns processors to RAs according to the traffic demand and optimization parameters, resulting in 2–33 processors assigned to each RA. It accepts considerably more users than both variants of the static strategy. This indicates the performance improvement of adapting the processor assignment to the actual user distribution.

## 9 Conclusions

SDR clouds provide an alternative concept for designing and managing wireless communications infrastructure. Higher resource efficiencies are achievable by merging the digital signal processing resources of today's base stations into data centers and employing cloud computing technology. The limited resources need to be properly managed, though. This article has addressed the SDR cloud computing resource management problem. Defining the concept of a RA that manages a subset of computing resources facilitates separating the signal processing algorithms design from the infrastructure and enables using resources on a pay-per-use basis.

Based on the call setup delay and the blocking probability constraints, we have defined the RA processor allocation problem as a constrained convex optimization problem. The feasibility region provides the maximum traffic capacity that a single RA can manage. The results have shown that modern cellular communications standards, such as LTE and LTE-Advanced, considerably limit the RA capacity. Assuming that two processors or more are required to process a transceiver processing chain in real time, less than 50 Erlangs of traffic can be handled by a single RA employing a greedy mapping algorithm. A distributed resource management is therefore necessary.

The data center processors need to be distributed among several RAs subject to the call setup delay and blocking probability constraints. The simulation results moreover indicate that the number of accepted users is severely degraded if the processors distribution is not adapted to the traffic distribution. Our solution is optimal, but does not scale well with the problem size. More precisely, the complexity of problem (18) grows exponentially with the number of RAs. Future research will therefore develop sub-optimal solutions for dynamically recalculating the assignment of processors to RAs and so adapt to varying traffic loads in very large-scale computing systems. Different traffic patterns and empirical data sets will also be considered.

**Competing interests**
The authors declare that they have no competing interests.

**Acknowledgements**
This study was supported by Spanish Government (MICINN) and FEDER under project TEC2011-29126-C03-02.

Received: 10 June 2012 Accepted: 4 February 2013
Published: 4 March 2013

**References**

1. 3rd Generation Partnership Project: 3GPP specification: 36.913 Rel-9; Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN). Tech. rep., 3rd Generation Partnership Project (2009)
2. 3GPP specification 36.913 Rel-10; Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced). Tech. rep., 3rd Generation Partnership Project (2011)
3. IF Akyildiz, WY Lee, MC Vuran, S Mohanty, NeXt generation/dynamic spectrum access/cognitive radio wireless networks: a survey. Comput. Netw. **50**(13), 2127–2159 (2006). http://dx.doi.org/10.1016/j.comnet.2006.05.001
4. I Gomez Miguelez, V Marojevic, A Gelonch Bosch, Resource management for software-defined radio clouds. IEEE Micro. **32**, 44–53 (2012)
5. J Mitola, The software radio architecture. IEEE Commun. Mag. **33**(5), 26–38 (1995)

6. R Buyya, CS Yeo, Venugopal S, J Broberg, I Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gen. Comput. Syst. **25**(6), 599–616 (2009). http://dx.doi.org/10.1016/j.future.2008.12.001
7. M Hill, M Marty, Amdahl's law in the multicore era. Computer. **41**(7), 33–38 (2008)
8. V Marojevic, XR Balleste, A Gelonch, A computing resource management framework for software-defined radios. IEEE Trans. Comput. **57**, 1399–1412 (2008)
9. C van Berkel, in *Design, Automation Test in Europe Conference Exhibition, DATE*. Multi-core for mobile phones (Nice, France, 2009), pp. 1260–1265
10. A Bastoni, BB Brandenburg, JH Anderson, in *Proceedings of the 2010 31st IEEE Real-Time Systems Symposium, RTSS'10*. An empirical comparison of global, partitioned, and clustered multiprocessor EDF schedulers (IEEE Computer Society, Washington, DC, USA, 2010), pp. 14–24. http://dx.doi.org/10.1109/RTSS.2010.23
11. BB Brandenburg, JM Calandrino, JH Anderson, in *Proceedings of the 2008 Real-Time Systems Symposium, RTSS'08*. On the scalability of real-time scheduling algorithms on multicore platforms: a case study (IEEE Computer Society, Washington, DC, USA, 2008), pp. 157–169. http://dx.doi.org/10.1109/RTSS.2008.23
12. K Ramamritham, J Stankovic, W Zhao, Distributed scheduling of tasks with deadlines and resource requirements. IEEE Trans. Comput. **38**(8), 1110–1123 (1989)
13. YK Kwok, I Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surv. **31**(4), 406–471 (1999). http://doi.acm.org/10.1145/344588.344618
14. YC Lee, Zomaya A, A novel state transition method for metaheuristic-based scheduling in heterogeneous computing systems. IEEE Trans. Parallel Distrib. Syst. **19**(9), 1215–1223 (2008). http://dx.doi.org/10.1109/TPDS.2007.70815
15. A Khokhar, V Prasanna, M Shaaban, CL Wang, Heterogeneous computing: challenges and opportunities. Computer. **26**(6), 18–27 (1993)
16. D Thain, M Livny, in *The Grid: Blueprint for a New Computing Infrastructure*, ed. by I Foster, C Kesselman. Building reliable clients and servers (Morgan, Kaufmann, 2003). ch. 19
17. L Smarr, CE Catlett. Metacomputing. Commun. ACM. **35**(6), 44–52 (1992). http://doi.acm.org/10.1145/129888.129890
18. SH Bokhari, On the mapping problem. IEEE Trans. Comput. **30**(3), 207–214 (1981). http://dx.doi.org/10.1109/TC.1981.1675756
19. A Iosup, D Epema, Grid computing workloads. IEEE Internet Comput. **15**(2), 19–26 (2011)
20. S Sakr, A Liu, D Batista, M Alomari, A survey of large scale data management approaches in cloud environments. IEEE Commun. Surv. Tutor. **13**(3), 311–336 (2011)
21. N Doulamis, A Doulamis, E Varvarigos, T Varvarigou, Fair scheduling algorithms in grids. IEEE Trans. Parallel Distrib. Syst. **18**(11), 1630–1648 (2007)
22. X Liu, C Qiao, D Yu, T Jiang, Application-specific resource provisioning for wide-area distributed computing. IEEE Netw. **24**(4), 25–34 (2010)
23. R Entezari-Maleki, A Movaghar, A probabilistic task scheduling method for grid environments. Future Gen. Comput. Syst. **28**(3), 513–524 (2012). http://dx.doi.org/10.1016/j.future.2011.09.005
24. MW Jonathan Segel, LightRadio: White paper 1. Tech. rep., Alcatel-Lucent. http://www.alcatel-lucent.com/lightradio/
25. Liquid Radio: Let traffic waves flow most efficiently. Tech. rep., Nokia Siemens Networks 2007–2013 http://www.nokiasiemensnetworks.com/portfolio/liquidnet/liquidradio
26. Z Zhu, P Gupta, Q Wang, S Kalyanaraman, Y Lin, H Franke, S Sarangi, in *Proceedings of the 8th ACM International Conference on Computing Frontiers*. Virtual base station pool: towards a wireless network cloud for radio access networks (ACM, New York, 2011), pp. 34:1–34:10. http://doi.acm.org/10.1145/2016604.2016646
27. GD Micheli, *Synthesis and Optimization of Digital Circuits*, 1st edn. (McGraw-Hill Higher Education, New York, 1994)
28. J Guo, F Liu, Z Zhu, in *International Conference on Wireless Communications, Networking and Mobile Computing, WiCom*. Estimate the call duration distribution parameters in GSM system based on K-L divergence method, Shanghai, China, 2007), pp. 2988–2991
29. Flexible Wireless Communications Systems and Networks (FlexNets) Web Site. http://flexnets.upc.edu
30. D Gross, C Harris, *Fundamentals of queueing theory*, 3rd Ed., Wiley edition. (A Wiley-Interscience publication, New York [u.a.], 1998)
31. A Jagers, EAD van, On the continued Erlang loss function. Oper. Res. Lett. **5**, 43–46 (1986). http://doc.utwente.nl/69738/
32. Y Rapp, Planning of Junction Network in a Multi-Exchange Area. Ericsson Tech. Rep. **4**, 77–130 (1964)

doi:10.1186/1687-1499-2013-59
**Cite this article as:** Gomez-Miguelez *et al.*: **Deployment and management of SDR cloud computing resources: problem definition and fundamental limits.** *EURASIP Journal on Wireless Communications and Networking* 2013 **2013**:59.