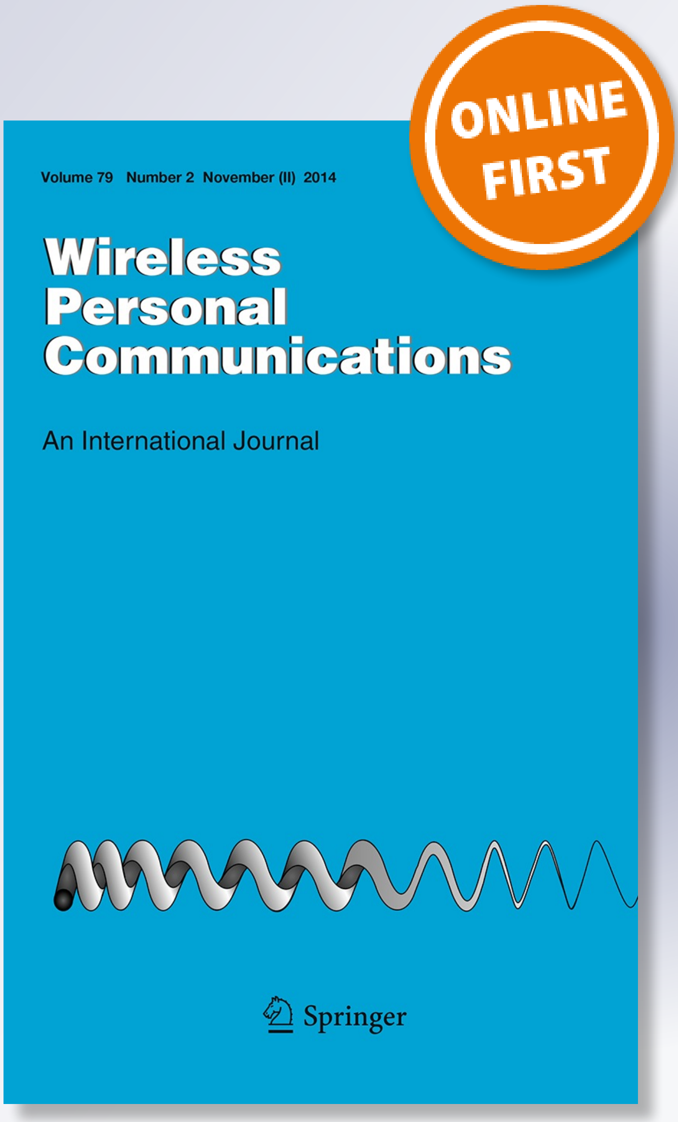
Context Monitoring Optimization in Autonomic Networks

Panagis Magdalinos, Alexandros Kaloxylos & Nancy Alonistioti

Wireless Personal Communications
An International Journal

ISSN 0929-6212

Wireless Pers Commun
DOI 10.1007/s11277-014-2128-x


ONLINE FIRST
Volume 79 Number 2 November (II) 2014
Wireless Personal Communications
An International Journal
Springer

Wireless Pers Commun
DOI 10.1007/s11277-014-2128-x

Context Monitoring Optimization in Autonomic Networks

Panagis Magdalinos · Alexandros Kaloxylos · Nancy Alonistioti

Abstract The recent advances in network management systems suggest the adoption of autonomic mechanisms in order to minimize the need for human intervention while handling complex heterogeneous networks. Data acquisition performed by monitoring processes is an essential part of autonomic mechanisms. The rate of sampling is a crucial factor since it is related to (1) the successful/unsuccessful detection of events, (2) the processing power needed to perform the sampling and (3) the energy that a node consumes during such actions. In order to address these issues we designed a simple and efficient mechanism that dynamically adapts the sampling rate of the context monitoring procedure. The merits of the mechanism are quantified by means of an analytical model as well as through extensive simulations that validated the theoretic outcomes. Finally, we experimentally assessed the effectiveness and efficiency of our approach through two real-world experiments. Overall results showcase that our mechanism achieves high detection rates while in parallel minimizes significantly the number of monitoring loops thus, emerges as a viable approach for context monitoring optimization in autonomic networks.

Keywords Context awareness · Monitoring Optimization · Autonomic networking

P. Magdalinos (✉)
Intracom Telecom S.A., 19th km Markopoulo Ave., 19002 Paiania, Athens, Greece
e-mail: magdpa@intracom-telecom.com

A. Kaloxylos
Riesstraße 25, 80992 München, Germany
e-mail: alexandros.kaloxylos@huawei.com

N. Alonistioti
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece
e-mail: nancy@di.uoa.gr

Published online: 25 October 2014

1 Introduction

Nowadays, mobile telecommunication networks are becoming more complex since they need to cater for an increased number of end devices and support a number of heterogeneous access networks. For example, an LTE mobile network [1] may have to support both 3GPP (e.g., GSM, GPRS, UMTS, HSPA and LTE networks) and non-3GPP access networks like IEEE 802.11. The management of these networks is becoming a tedious task and automated solutions are required.

During the past decade considerable effort has been placed to address these issues, so the notion of "autonomic network management" was created. A number of papers [2–5] surveys the proposed solutions and architectures [6] that support autonomic solutions. 3GPP has specified a number of autonomic operations, called Self-Organizing Networks (SON) operations, for LTE systems. These procedures are categorized as self-configuration, self-optimization and self-healing automated operations [7,8]. These research efforts have also inspired the design of new architectural approaches, like [9], where an architecture is defined for the self-organization of collaborating heterogeneous networks that are able to dynamically reconfigure their purpose and operation.

All the above mentioned architectures and mechanisms are based on the so called Monitor-Analyze-Plan-Execute (MAPE) loop. A SON operation starts from the monitor state where it observes network parameters and acquires network related measurements. During the analyze state, the collected information is analyzed while during the plan (also called decide) state the decision for the execution, or not, of an autonomic operation is taken. The execute state is the last state before starting over the loop where the healing, configuration or optimization actions are enforced.

Early enough, it was recognized that the process of collecting information while being in the monitoring state can play a crucial role for the performance of the autonomic systems [10]. The main issue is that a coarse monitoring process will lose events that have to be detected, while oversampling will place serious demands on resources like the CPU or the energy consumption of a node. An example of MAPE cycle is an IEEE 802.11 access point (AP) that needs to optimally monitor its channel for increased interference levels so as to execute a channel reselection mechanism. Another example is a network node that needs to check for any signs of congestion in one of its links, so as to execute a load balancing mechanism. Monitoring also plays an important role to energy consumption minimization e.g. mechanisms which are able to shutdown APs under specific coverage and capacity constraints that have to be monitored in an appropriate rate.

All the above examples give an indication that the monitoring process which is present in all autonomic mechanisms has to be carefully designed to accurately estimate the monitoring idle time and reduce the required CPU cycles as well as energy consumption. In principle, all the aforementioned functionality enables systems to become context aware. According to [11], by the term context we refer to "any information that can be used to characterize the situation of an entity". Specifically for the monitoring processing, in [12] it is recognized that "it should be continuously adapted to an ever changing network infrastructure following thus, self-configuration concepts".

In this paper we present an algorithm that adapts the monitoring process based on the presence or lack of events. The algorithm is very generic and can be used by every autonomic operation since its adaptation rate is essentially based on the density of the occurring events. By the term event we refer to any contextual data that has to be monitored.

For our analysis we assume that events occur in batches. The reasoning for this is the following. If an event (e.g., congestion on a network link) takes place it will remain unresolved until either the monitoring process detects it and an autonomic operation resolves it, or the conditions that created the event are altered for other reasons (e.g., a heavily transmitting node moves to another AP). Based on this observation we develop an event-driven algorithmic solution that adapts the monitoring rate based on the detection of specific events. The main idea is to reduce the monitoring rate to a minimum if no events are detected. If an event is detected, then the algorithm increases the monitoring rate to a maximum. For the reduction of the monitoring rate we evaluate different flavors of the algorithm based on an analytical model. Simply put, we wish to identify as many events as possible (preferably all) while at the same time each node executes as less monitoring loops as possible by minimizing the number of sensing attempts. Depending on the autonomic procedure, the minimization of the monitoring loops is closely related to the CPU load and/or the consumed energy.

As we will describe in the following sections, the algorithm achieves significant successful detection rates while at the same time minimizes the number of the monitoring actions that have to be executed. The theoretical results are validated in extended experiments. These experiments have been conducted for a number of autonomic operations that were developed in the context of the EU funded project "Cooperative and Self growing Energy aware Networks" CONSERN [13].

The rest of the paper is structured as follows. In Sect. 2 we discuss work that is related to our algorithm. In Sect. 3 we provide a detailed description of all flavors of the algorithm, the performance metrics and an analytical model that helps the theoretic evaluation of its performance. Section 4 presents detailed results derived from extensive simulation efforts as well as the implementation of the algorithm on the CONSERN architecture. Finally, Sect. 5 concludes the paper and sketches future research directions.

2 Related Work

The monitoring process is of paramount importance in autonomic networking. However, to the best of our knowledge, limited work has been undertaken focusing on sampling rate optimization. To a certain extent, research efforts that can be considered related for this area are those dealing with energy optimization schemes, since they introduce the notion of sleep or idle time.

Recent studies showed that significant energy savings can be achieved by turning radios off when not in use. For example, several scheduling-based approaches have been proposed where nodes turn on their radio interface only in scheduled slots. The active slots of different nodes can be synchronous or asynchronous. Examples of the former case include medium-access control protocols used by nodes that periodically go into a sleep mode so as to reduce energy consumption by listening to an idle channel. Examples of the asynchronous case (e.g., [14]) include wakeup mechanisms in ad hoc networks that do not require global clock synchronization. In [15], a carrier sense media access protocol for wireless sensor networks employs an adaptive preamble sampling scheme to reduce duty cycle and minimize idle listening. In [16], energy savings are achieved by powering down nodes, during idle times identified through dynamic scheduling.

Additionally, several adaptive sleep schemes dynamically adjust the schedules based on traffic activities such as in [17] where a contention-based medium-access control protocol for wireless sensor networks is described, reducing the amount of energy wasted on idle listening. An extensible on-demand power management framework is proposed for ad hoc networks that adapts to traffic load in [18]. In this case, nodes maintain soft-state timers that determine transitions between the different power management modes. In [19], an efficient sleep scheduling based on application timing management scheme is developed, that aggressively exploits the timing semantics (traffic patterns in time) of wireless sensor network applications. In parallel, an adaptive scheme is apparent in the notion of Discontinuous Reception (DRX) used in LTE—[1]. DRX introduces long and short cycles during which if no packet is to be transmitted to/from a node, then it can enter into a sleep period to preserve its battery.

The aforementioned solutions are defining sleep/idle periods when some activity (e.g., transmission of data) can be skipped. However, it is obvious that such solutions are not adequate for optimizing the sampling rate in a monitoring process since they are designed to operate based on the traffic patterns in a network. Note that in principle the monitoring process is very generic, since it can be applied for several autonomic functions (e.g., scanning the radio interface to detect the presence of a network, checking for a possible congestion on a network interface, detect the presence of interfering devices) that have very different characteristics from data traffic.

Another work in the broader area is presented in [20], where the authors discuss scanning schemes in WLAN networks. For this case, several researchers target either to reduce the required channel scanning time (e.g., stop the scanning process if a criterion is met) or minimize the QoS degradation during the scanning period. The work in [20] suggests the introduction of a policy-based scheme to address these issues.

Specifically designed for the monitoring process, the authors in [10] present an appropriate network management probe to collect information from network elements using appropriate SNMP traps. However, the sampling rate is not discussed since it is well known that defining the SNMP polls per event depends on the number of variable bindings in a protocol data unit and the response time for an interface from the packet forwarding engine.

A work most directly related to ours is presented in [21]. The authors build on the contemporary fact that smart-phones turn-on the Wi-Fi on demand in order to minimize energy consumption. However, in this way, Wi-Fi access opportunities are minimized. This is especially important for the use of applications running on the background like social media services and gaming. It can also be a crucial issue for smart-phones that can perform vertical handovers from a cellular network to Wi-Fi. As the authors suggest, it is important to "determine an optimal Wi-Fi sensing frequency for energy efficient WiFi discovery". Thus, they describe algorithms that take advantage of context information like the mobility of the user (using the accelerometers of the smart-phone), the AP density and the received signal strength that is collected opportunistically to determine the sensing frequency. This proposal suggests significant gains from minimizing energy consumption while presenting reduced false triggering of scanning actions. Note however, that in order to achieve these results, appropriate information needs to be collected and processed to determine the scanning frequency.

Another interesting work is presented in [22] where authors attempt to minimize the battery consumption of smart-phones that are using their sensors (e.g., accelerometers) to monitor the health status of a patient. As in [21] they use additional information that is inferred by specific signals (e.g., an accelerometer indicates that the user is static) and reduce the sampling intervals from 10 to 1 Hz. However this is a tail cut solution and the different sampling rates are statically pre-decided. Finally, the authors in [23] discuss in detail different uploading strategies to handle continuous sensing application in an energy efficient manner. Their improvements are based on using additional information (e.g., location) to predict the behavior of a user or by performing an online stream analysis.

Based on the previous discussion, it is clear that although the sampling rate in a monitoring process is important and can result in significant gains in the operation of autonomic networks, few efforts have been made up to now on how to optimize it. Additionally, the few related solutions are tightly integrated with specific problems (e.g. [21–23]) and require the combination of information as well as network related data (e.g., AP density) to find an optimum value. In the following sections we present a new algorithm that can provide significant gains by simply monitoring events that are defined for different autonomic operations. These gains are calculated based on a theoretical model and more importantly are validated in extensive simulated and real-world experiments.

3 Context Monitoring Optimization Algorithm

In accordance with the observations above, we design a simple and efficient monitoring algorithm, particularly suitable for autonomic networks. The algorithm attempts to kill two birds with one stone; on one hand, in the absence of events minimize the number of unnecessary monitoring actions, while on the other hand detect—ideally all—events when they appear.

Our motivation stems from the application of a large number of network management actions in autonomic environments. For example, consider the case where an IEEE 802.11 Access Point (AP) is monitoring the traffic load on its wireless and wired interfaces. In case the traffic surpasses a predefined threshold, the AP can execute a load balancing mechanism. It is neither effective nor efficient to monitor for such a threshold violation periodically (e.g. every second). However, if the threshold is violated and the load balancing procedure is applied, the AP should be able to keep track of all subsequent events in order to verify that the problem has been successfully resolved. Similar arguments can be applied for situations like interference identification [24], coverage and capacity optimization [25], etc.

Towards this end, our Context Monitoring Optimization algorithm (CMO), attempts to minimize the number of monitoring activities while keeping the number of undetected events at a minimum. We assume that events follow a batch process; if a particular event is monitored at a given time then it will be present until the problem that created it has been resolved.

From a high level, methodological point of view, upon initiation, the algorithm searches for events scanning at the highest possible frequency (e.g. one scan per designated time). In the absence of events, its scanning rate is progressively reduced until a predefined, low threshold is reached. When an event is identified the algorithm is re-calibrated to the highest possible scanning frequency and keeps measuring at this frequency until no event is monitored. The same process is repeated until all events have been identified.

Algorithm 1 provides the description of the aforementioned idea. The algorithm receives four input parameters; t_m which signifies the duration (in milliseconds) of the monitoring procedure itself,[1] G_{init} the designated time (e.g., one measurement per G_{init} seconds), G_{final} the lowest scanning threshold (e.g. one measurement per G_{final} seconds with $G_{init} < G_{final}$) and k the number of progressive reductions in the algorithm's scanning frequency before reaching the lowest scanning threshold.

[1] The algorithm is generic in nature, thus the same procedure can be used for monitoring events which require minimum processing time (e.g. movement) or a significant amount (interference identification in IEEE 802.11 networks).

Algorithm 1: The adaptable scanning algorithm

```
Input  $t_m, G_{init}, G_{final}, k$
1:     $sleep\_time \leftarrow G_{init}$
2:     $step \leftarrow (G_{final} - G_{init})/k$
3:     $i \leftarrow 1$
4:     while true
5:         Scan
6:         if event
7:             $sleep\_time \leftarrow G_{init}$, $i \leftarrow 1$
8:         else if $sleep\_time < G_{final}$
9:             $sleep\_time \leftarrow G_{init} + i * step$,
               $i$++
10:        $sleep(sleep\_time - t_m)$
```

For example consider again the case of load monitoring on a particular interface. In the simplest implementation, the context monitoring procedure should first perform a file read (e.g. under Linux operating system it should parse file /proc/net/dev), find the interface under consideration, sum the transmitted and received bytes and forward the result to the decision making entity of the device. Although simple, this process requires a certain amount of time which we denote as t_m. Thus, if CMO is calibrated to take 1 measurement every G_{init} seconds, it should initiate with a sleep interval of $G_{init} - t_m$ seconds. Every time the interface is monitored and no change is identified, the sleep time is increased (thus the monitoring frequency is reduced) by a constant factor, which is defined as $\frac{G_{final}-G_{init}}{k}$, where k is the minimum number of monitoring loops the algorithm should perform before reaching its lowest scanning frequency.

The generic nature of the algorithm enables its configuration and adaptation in various ways, thus enabling the derivation of various flavors. Depending on the application environment the procedure can be configured to change the scanning rate either in a faster or slower pace. Four major directions are envisaged:

- *Go-Back-1 with a single step* The algorithm is deployed in its generic form, as presented in Algorithm 1.
- *Go-Back-1 with an N-step* The algorithm remains in the same monitoring rate for N consecutive loops before transcending to a lower monitoring frequency.
- *Go-Back-Half with a single step* A more aggressive flavor, in which case the monitoring frequency is reduced by a factor of $2^i * step$.
- *Go-Back-Half with an N-step* Similar in nature to Go-Back-1, N-step; the algorithm remains in the same monitoring frequency for N consecutive scans before transcending into a lower frequency.

CMO is of course adaptable, however a key question rises; given a specific environment, how can we identify the best configuration in terms of G_{init}, G_{final} and k? We will attempt to answer this question through the next paragraphs.

Initially, we define P the probability of detecting an event when monitoring with a specific frequency. For the purpose of estimating P we can consider a time-slotted model where each event and each sensing operation take place in one time-slot. P can be expressed as a function

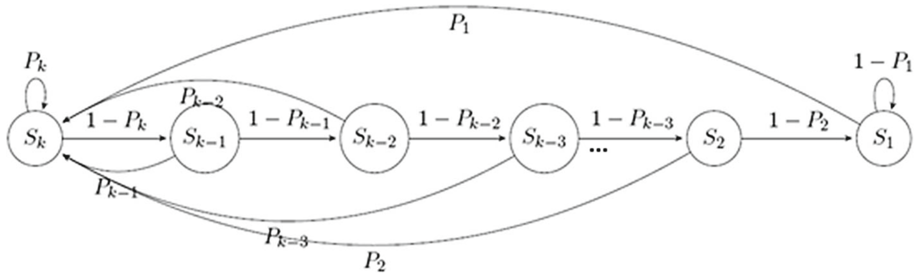


Fig. 1 Markov Model for the Context Monitoring Optimization algorithm

of the overall density of events (d) and the scanning frequency $1/G_i$ at a given time t_i (at time t_i we measure once every G_i time slots). Obviously, when the scanning frequency $1/G_i$ is large, the probability P of detecting an event grows. Thus, if we take one measurement every G_i time-slots the probability of detecting an event is $P = d/G_i$.

Of course the best results will be provided when $P = d$, thus $G_i = 1$ (we measure once every time-slot, thus we manage to identify all events); however in the same time, we want to save monitoring loops (which are eventually translated to CPU cycles). Therefore, we would like to find a way to theoretically combine the values of P and G_i that will enable us to calibrate the algorithm in such a way so as to identify a large number of events and in parallel minimize the number of required monitoring actions.

3.1 Algorithm Modeling

In order to perform the aforementioned optimization task, we model CMO as a Markov chain, each state of which essentially adjusts the scanning frequency of the monitoring procedure. We assume that a scanning/sensing/monitoring attempt is successful if it identifies an event, and unsuccessful otherwise. Each state of the Markov chain is characterized by a monitoring frequency. If the algorithm detects the occurrence of an event, it transitions to state S_k in which it operates at the highest possible monitoring frequency. If the node does not detect an event, it transitions to a 'lower' state (towards state S_1), in which sensing is sparser. Transition probabilities for any state S_i are P_i and $1 - P_i$ where the former denotes the probability of transition from S_i to S_k and the latter from S_i to S_{i-1}. This model is depicted in Fig. 1.

The input parameters of CMO (G_{init}, G_{final}, k) are mapped as follows:

- G_{init} Initial monitoring interval of state S_k (i.e. duration of initial timeslot),
- G_{final} Final monitoring interval of state S_1 (i.e. duration of final timeslot),
- k Number of states (k)

This chain is described by the following transition matrix P:

$$\begin{array}{llllllll}
1 - P_1 & 0 & 0 & \cdots & 0 & 0 & 0 & P_1 \\
1 - P_2 & 0 & 0 & \cdots & 0 & 0 & 0 & P_2 \\
0 & 1 - P_3 & 0 & \cdots & 0 & 0 & 0 & P_3 \\
\cdots. & \cdots. & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & P_{k-3} \\
0 & 0 & 0 & \cdots & 1 - P_{k-2} & 0 & 0 & P_{k-2} \\
0 & 0 & 0 & \cdots & 0 & 1 - P_{k-1} & 0 & P_{k-1} \\
0 & 0 & 0 & \cdots & 0 & 0 & 1 - P_k & P_k
\end{array}$$

The steady state probability vector $\vec{\pi}$ can be computed by solving the following system of equations:

$$\vec{\pi} * P = \vec{\pi}$$

Solving this system leads us to the following set of equations:

i. $\pi_1 (1 - P_1) + \pi_2 (1 - P_2) = \pi_1$
ii. $\pi_3 (1 - P_3) = \pi_2$
iii. $\pi_4 (1 - P_4) = \pi_3$
iv. $\pi_5 (1 - P_5) = \pi_4$
$\dots$
v. $\pi_{\kappa-2}(1 - P_{k-2}) = \pi_{k-3}$
vi. $\pi_{\kappa-1}(1 - P_{k-1}) = \pi_{k-2}$
vii. $\pi_\kappa(1 - P_k) = \pi_{k-1}$
viii. $\pi_1 P_1 + \pi_2 P_2 + \cdots + \pi_{k-1} P_{k-1} + \pi_k P_k = \pi_k$

Additionally, we know that:

ix. $\pi_1 + \pi_2 + \cdots + \pi_k = 1$

Using equation i. we derive:

x. $\pi_2 = \pi_1 P_1/(1 - P_2)$

In parallel, equations ii.- vii. lead us to:

xi. $\pi_i = \pi_{i-1}/\left(1 - P_i\right)$, where $i = 3 \dots k$

Notice that by combing x. and xi. we can express all π_i, $i = 2\dots k$ as a function of π_1. Thus we extract equation xii. :

xii. $\pi_i = \pi_1 P_1 \Big/ \prod_{j=2}^{i}\left(1 - P_j\right), \quad i = 2 \dots k$

Finally by using equation (viii) and solving for π_1 we find formula (xiii.):

xiii. $\pi_1 = \left(1 + \sum_{i=2}^{k} P_1 / \prod_{j=2}^{i}\left(1 - P_j\right)\right)^{-1}, \quad i = 1$

Equations xii. and xiii. are extremely useful as we will see in the following. The next step is to calculate the P_i values as a function of our input parameters. Given the analysis at the end of the previous section (Sect. 3), we already know the relation between P_i and G_i (i.e. $P_i = d/G_i$) therefore it suffices to determine G_i in order to derive the corresponding π_i values. Towards this end, we analyze the four different flavors of CMO.

3.1.1 Go-Back-1 (GB1) Algorithm

In the GB1 algorithm, if CMO detects an event it transitions to the 'maximum' state S_k, and if not it transitions from its current state S_i to state S_{i-1}. The two different configurations are formalized as follows:

- $N = 1$: CMO needs only one unsuccessful attempt in order to transition to a lower state.

This is in essence a direct implementation of the model above. We set $G_k^{GB1} = 1$ slot,[2] so that in state S_k no event is missed since CMO is constantly scanning. We also set $G_{k-1}^{GB1} = 2$,

[2] A slot can have a duration of x milliseconds depending on the context of implementation. In order to simplify the presentation, we purposely omit time duration, however the concept of the time-slotted model inherently incorporates it.

so that in state S_{k-1} the algorithm performs one scan every two time slots. Similarly, we set $G_{k-2}^{GB1} = 3$, so that in S_{k-2} the algorithm performs one scan every three time slots. In general, we set $G_i^{GB1} = k - i + 1, i = 1...k$.

- $N = c$: CMO waits for c consecutive unsuccessful attempts before transitioning to a lower state.

Since the model of Markov chain does not have memory, we cannot count the number of unsuccessful attempts within one state. To overcome this limitation, we augment the Markov chain by duplicating each state c times and transitioning through these intermediate states before eventually jumping to the 'lower' state. The resulting Markov chain has ck states, with scanning frequencies G_i^{GB1n} set as follows: $[G_{ck-ic}^{GB1n} \dots G_{ck-(i+1)c+1}^{GB1n}] = G_{k-i}^{GB1}, i = 0 \dots k - 1$.

3.1.2 *Go-Back-Half (GBH) Algorithm*

In GBH, when CMO in state S_i does not detect an event within a specified number of trials, it transitions to state $\mathrm{S}_{\mathrm{i}/2}$. If it does detect an event, it transitions—as in all versions of the algorithm—to state S_k. Two different flavors are investigated:

- $N = 1$: CMO needs only one unsuccessful attempt in order to transition to a lower state.

This algorithm is similar to GB1, where all states between S_i and $S_{i/2}$ are never reached and therefore removed. Consequently, the corresponding rows and columns are removed from the transition matrix P, resulting in a Markov chain with $f = \log_2 k + 1$ states instead of k. The scanning rates are then given by $G_{f-i}^{GBH} = G_{k/2^i}^{GB1}$ for $i = 0...f - 1$

- $N = c$: CMO waits for c consecutive unsuccessful attempts before transitioning to a lower state.

Similarly to the second flavor of GB1, we duplicate each state c times. The resulting Markov chain has *cf* states, with monitoring rates G_i^{GBHn} set as follows: $[G_{cf-ic}^{GBHn} \dots G_{cf-(i+1)c+1}^{GBHn}] = G_{k-i}^{GBH}$, $i = 0...f - 1$, $f = \log_2 k + 1$.

3.2 Theoretic Evaluation

Following our initial discussion, we introduce two metrics in order to evaluate our approach:

- *Successful Detection Percentage (SDP)* the percentage of successfully detected events defined as n/n_0 where n is the number of detected events and n_0 the total number of events,and,
- *Events Computation Percentage (ECP)* the percentage of the monitoring loops performed in order to identify events defined as c/c_0, where c is the number of computation loops performed for the detection of n events and c_0 the number of loops required for the computation of all events.

These metrics attain their highest values (i.e. 1) when calculated in a scenario where CMO continuously senses for events at the highest possible frequency. In order to select optimal points, we introduce the Success Evaluation Product (SEP), which is given by the following equation:

$$\mathrm{SEP} = \mathrm{SDP} \times (1 - \mathrm{ECP})$$

The optimum point is the one that maximizes SEP. Notice that by adding weights in any of the two factors of the product, optimum points that favor the corresponding factor can be generated.

Given the analysis of the previous paragraphs we can estimate the steady-state SEP for any Markov chain. When in state S_i the node retrieves 1 measurement every G_i slots, thus, it performs $1/G_i$ iterations instead of 1. The steady state probability of this state is π_i. Thus (recall that d is the density of events),

$$ECP_{k,d} = \sum_{i=1}^{k} \pi_i/G_i$$

Following a similar argument we can also derive $SDP_{k,d}$. The probability of detecting an event in each measurement is d, while the measurement rate is $1/G_i$. Therefore,

$$SDP_{k,d} = \sum_{i=1}^{k} d\pi_i/G_i$$

By combining the two relations we get:

$$SEP_{k,d} = \sum_{i=1}^{k} d\pi_i/G_i \left(1 - \sum_{i=1}^{k} \pi_i/G_i\right)$$

Using this formula we calculate the optimal number of states in the Markov chain for event densities in the range 0–0.95 with a step of 0.05. For each specific value of event density, both *SDP* and *ECP* decrease when the number of states in the Markov chain increases, since in 'lower' states (towards S_1) the monitoring cycle is lower (lower monitoring frequency) hence exhibiting a low number of iterations translates to higher probability of missing events. Consequently, the optimal *SEP* is determined as a trade-off between *SDP* and *1- ECP*. Moreover, for $d \rightarrow 1$ the Markov process always stays in state S_k (irrespective of the number of states k), and therefore $ECP \rightarrow 1$ and $SEP \rightarrow 0$.

For each density value we calculate the resulting *SEP* for all chains for $k = 2 \ldots 100$ states, and report the number of states (k) that maximizes it. If the same *SEP* value is achieved by more than one chain then we select the longest one (maximum k), as it provides the highest potential for saving iterations. The results are illustrated in the following four Fig. (Fig. 2a–d). Note that in practice, the system does not behave so deterministically, since for example the density of events fluctuates. Therefore, the graphs should be considered as approximations of the actual system behavior. Also, we have not evaluated the GB1 and GBH flavors with $N = c$ since the behavior will be the same (recall that $N = c$ calibration is reduced to an $N = 1$ case with a larger chain).

GB1 Algorithm: Simulation Results Figure 2a depicts the optimal number of states as a function of event density for the GB1 algorithm, and Fig. 2c depicts the resulting *SEP*. For low event densities the probability of reaching low frequency states in the Markov chain is high. Consequently, the limiting factor of *SEP* is *SDP*, and the optimal number of states is small and growing as the density of events grows. For high event densities the probability of reaching low frequency states is low. In this case, the limiting factor of *SEP* is 1—*ECP*, and the optimal number of states is large—taking as much advantage as possible of every (infrequent) case of a long 'quiet' period. We already know that when $d \rightarrow 1$ we have $SEP \rightarrow 0$ (i.e. CMO will constantly monitor due to event-full context, thus $ECP \rightarrow 1$ and therefore $SEP \rightarrow 0$). Therefore it is not surprising that at some point *SEP* starts to decline.

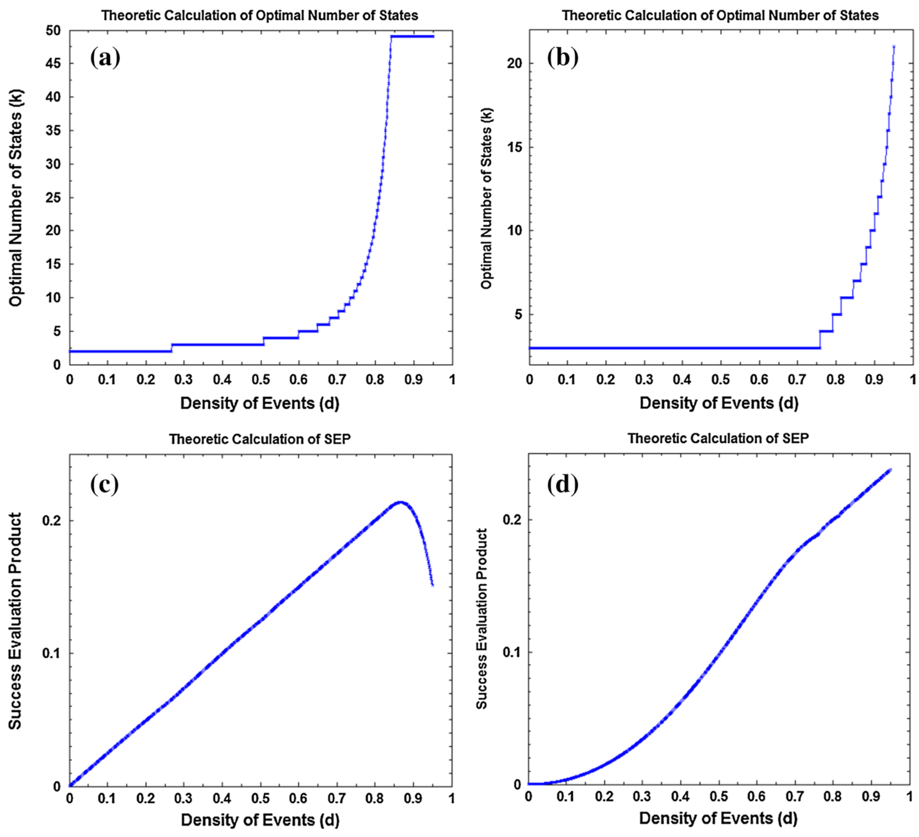


Fig. 2 Theoretic computation of optimal k according to density (d) for **a** GB1, **b** GBH. Theoretic computation of SEP according to density (d) for **c** GB1, **d** GBH

Moreover, this implies that beyond this point the Markov chain almost never reaches the low frequency states, therefore there is no point in adding more states. The latter explains the flattening of the optimal number of states in Fig. 2a.

A careful study of Fig. 2a, c reveals that for event densities ranging between 30 and 70 % a chain with 5–10 states suffices for attaining the best possible performance. For higher event densities (i.e. 70–80 %) there is a sharp increase in the number of states required, reaching as many as 50 for a density level of 85 %. From 85 to 95 % the optimal number of states remains 50 and *SEP* starts to decline. For low event densities (e.g. <20 %), a chain of 2–3 states emerges as a valid compromise between events losses and computation resources waste (we waste some resources and measure once every two or three slots in order to capture as many as possible of our rare events)

GBH Algorithm: Simulation Results Similar behavior is exhibited for the GBH algorithm (Fig. 2b, d). Contrary to GB1 however, GBH can achieve higher energy gains at high event densities since the transition to a lower state occurs with a much larger/aggressive step (of $k/2$). When there is an opportunity to save energy at e.g. state 50 (continuous measurements), the system transits to state 25 (1 measurement every 25 time-slots), as opposed to state 49 in GB1.

A careful study of the corresponding figures reveals that for event densities ranging between 10 and 75 % a chain with 3-4 states suffices for attaining the best possible performance. For higher event densities (i.e. 75–90 %) there is a sharp increase in the number of states required, reaching as many as 10 for a density level of 90 %. From 90 to 95 % the optimal number of states steadily increases, reaching 20. GBH appears to be the model which manages to carefully balance events losses and energy consumption through a small number of states.

4 Experimental Analysis

In order to assess the validity of our results we performed two sets of experiments. The first set focused on the evaluation of the algorithm by means of extensive simulations and is presented in Sect. 4.1. With the second set, we attempted to assess the viability of the algorithm in a large real world context; towards this end we designed and executed two real world experiments which we report in Sect. 4.2.

4.1 Simulation Results

In order to assess the actual behaviour of the algorithm we generated a large dataset against which all variations of CMO were tested. We used a compound Poisson process with rate λ_1 in order to identify events arrival times. The duration of each event follows a log-normal distribution with $\mu = 0$ and $\sigma = 1$. Each event is followed by an idle time period which is distributed exponentially with rate λ_2. Both λ_1 and λ_2 ranged from 0.05 to 2 with a step of 0.05. Simply stated, the vector resulting from the Poisson process is augmented by substituting all 0 elements with vectors of exponentially distributed lengths. For example, for $\lambda_1 = 1$ and $\lambda_2 = 1$ we get the following vectors:

$$Y = [0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 2\ \ 0\ \ 0\ \ 1\ \ 0\ \ 3\ \ 0\ \ 2\ \ 1\ \ 2\ \ 0\ \ 0\ \ 2\ \ 4\ \ 0]$$
$$Z = [19\ \ 11\ \ 5\ \ 17\ \ 4\ \ 15\ \ 1\ \ 14\ \ 3\ \ 17\ \ 13\ \ 24\ \ 6\ \ 4\ \ 7\ \ 9\ \ 5\ \ 5\ \ 4\ \ 5]$$

The resulting vector resembles an On/Off process and is constructed by concatenating Y and Z according to the following procedure:

- Substitute the ith element of Y with $Y[i]$ ones
- Add $Z[i]$ zeros after end of the batch

The final vector X simulates our experimental validation environment; if $X[i] = 0$, then there is no event, thus CMO can transcend to a lower frequency state; otherwise it identifies an event and ascends to the state with the highest monitoring frequency. Subsequent 1s or 0s signify event-full or event-less periods of time.

For each set of λ_1 and λ_2 values (totaling a set of 1,600 different combinations) 20 distinct inputs were generated. These inputs were evaluated against the four configurations of CMO for different values of k and the mean values of *SEP*, *SDP* and *ECP* are reported. In our experiments we evaluate for $k = 2...50$. Taking into account the latter, each distinct configuration of CMO, for each k, was assessed against 32000 different input cases.

Since in our data generation procedure we controlled only λ_1 and λ_2, the length (L) of the input, the density of events (d) as well as the mean duration (T) of the event-full periods were defined as a function of these parameters. Vectors Y and Z which formulated the input vector X were of length 2000 in all cases. A number of graphs depicting these characteristics of the dataset are presented in Fig. 3a–d.

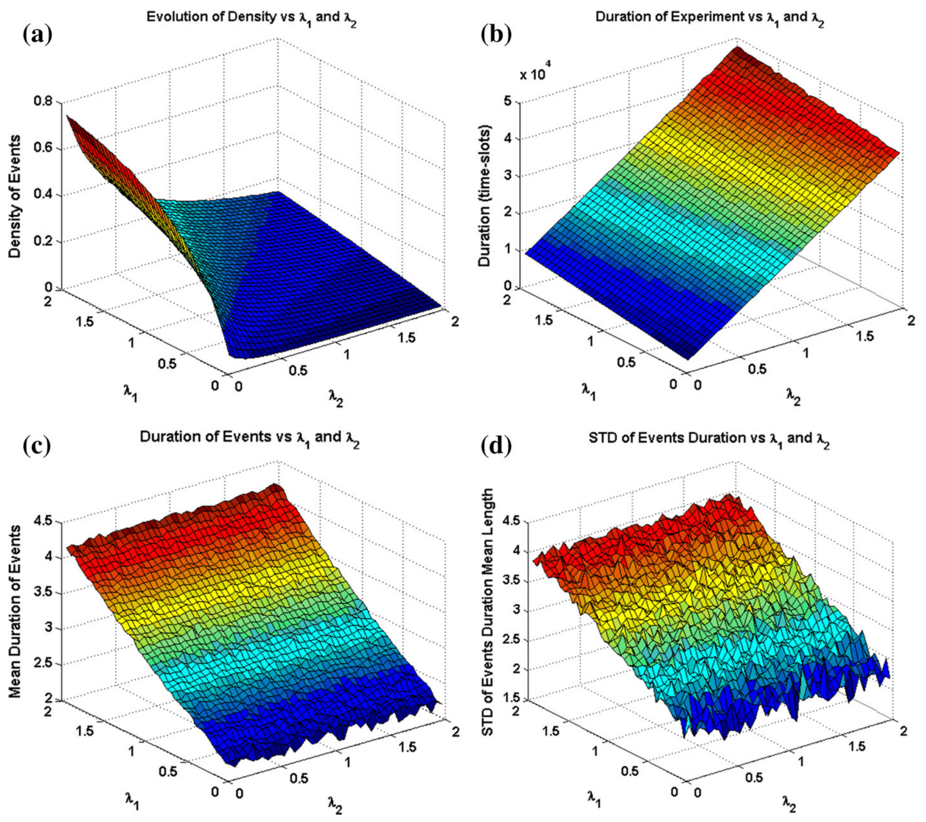


Fig. 3 Evolution of **a** density (d) of events, **b** mean length of input vectors, **c** mean length of events, **d** standard deviation of the length of events according to λ_1 and λ_2

Statistical Properties of the Dataset The duration of experiments (essentially the length of the input vectors) evolves linearly with respect to the rates λ_1, λ_2, as depicted in Fig. 3b. However the density of events (Fig. 3a) falls sharply as soon as the idle period increases (higher λ_2 values). The latter is translated to more zero-cells—eventless period—between two consecutive batches of 1s—eventful periods-. Finally, Fig. 3c, d present the mean duration of the eventful experimentation periods (e.g. how long the events last). We observe that λ_1 drastically affects their duration; it increases almost linearly with respect to λ_1 exhibiting minimum deviation.

In order to ease presentation, for each combination of λ_1 and λ_2 we present the best *SEP* and corresponding *ECP*, *SDP* values (i.e. the highest mean *SEP* attained for one of the given λ values). Finally, we provide the best k (the length k of the chain that produced the best *SEP*) for every combination.

Results for CMO, GB1 Configuration Overall results for GB1 appear in Fig. 4a–d. Figure 4a shows the *SDP* evolution with respect to λ values. As it is clearly depicted, the event detection capability of the algorithm is constantly above 50 %, while for medium and high density cases (e.g. $\lambda_1 > 1$) *SDP* is constantly above 65 %. The latter is achieved most of the time with half the loops of the always-on configuration; *ECP* is constantly less than 50 %—as depicted in Fig. 4c and reaches 70 % only when $\lambda_1 \rightarrow 2$.

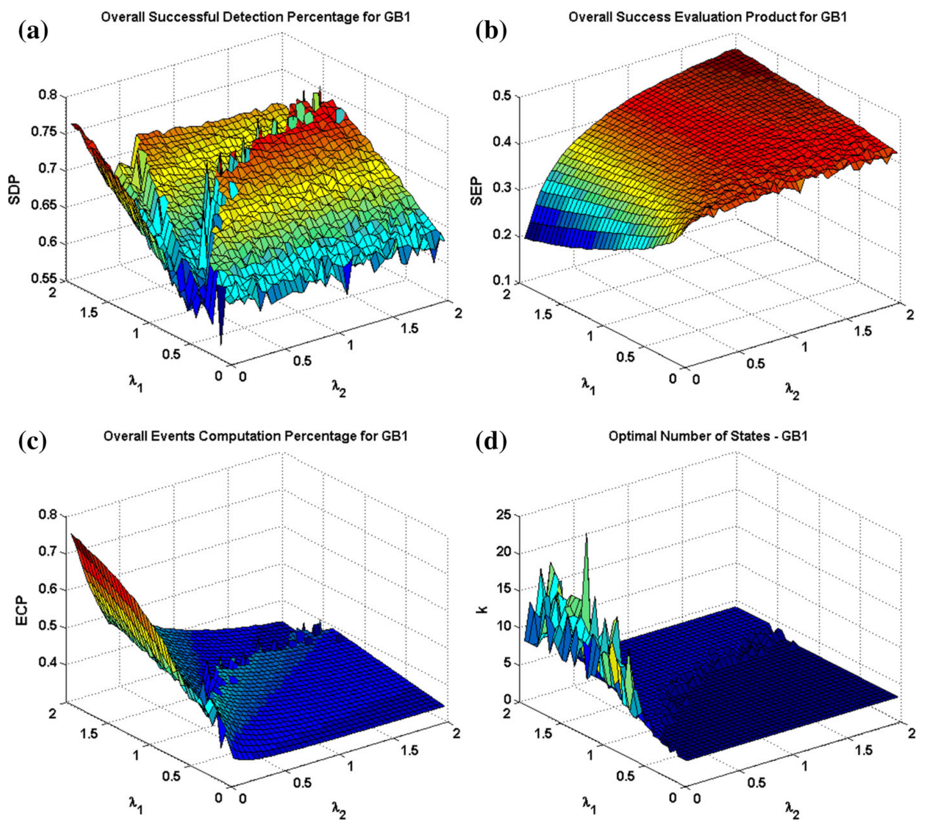


Fig. 4 Evolution of **a** SDP, **b** SEP, **c** ECP with respect to λ_1 and λ_2—GB1. **d** Optimal k with respect to λ_1 and λ_2—GB1

In general, we observe that GB1 provides results of high quality; for example it requires only 40 % of the energy of the always-on case in order to detect 75 % of events when events appear with a density of 35 % ($\lambda_1 = 1,\ \lambda_2 = 0.5$). In parallel, a closer look reveals that if the density of events is smaller than 50 % ($\lambda_2 > 0.5$) then GB1 necessitates less than 50 % of the energy of the always-on case in order to detect more than the 70 % of events. The latter can be achieved with a Markov Chain of 3-5 states (Fig. 4d) which is in accordance with the theoretic results of Fig. 2.

Results for CMO, GB1 Configuration with N = c Similar results have been obtained when we tested GB1 setting N to 10 (Fig. 5a–d). Following the description of Sect. 3.1.1, the algorithm retained the same scanning rate for 10 consecutive scans before transiting to a lower frequency state. This configuration gives better results in terms of *SDP* (Fig. 5b), however at a significantly higher cost, since more scans are required to take place (Fig. 5c). Therefore, in terms of SEP (Fig. 5a), this configuration is generally less competent compared to GB1 with $N = 1$. However, for sparse environments (a density lower than 20 %, i.e. $\lambda_2 \rightarrow 2$) it produces results of comparable quality with GB1 $N = 1$ giving an SDP of 85 using 60 % of the energy with a chain of 10–15 states (Fig. 5d).

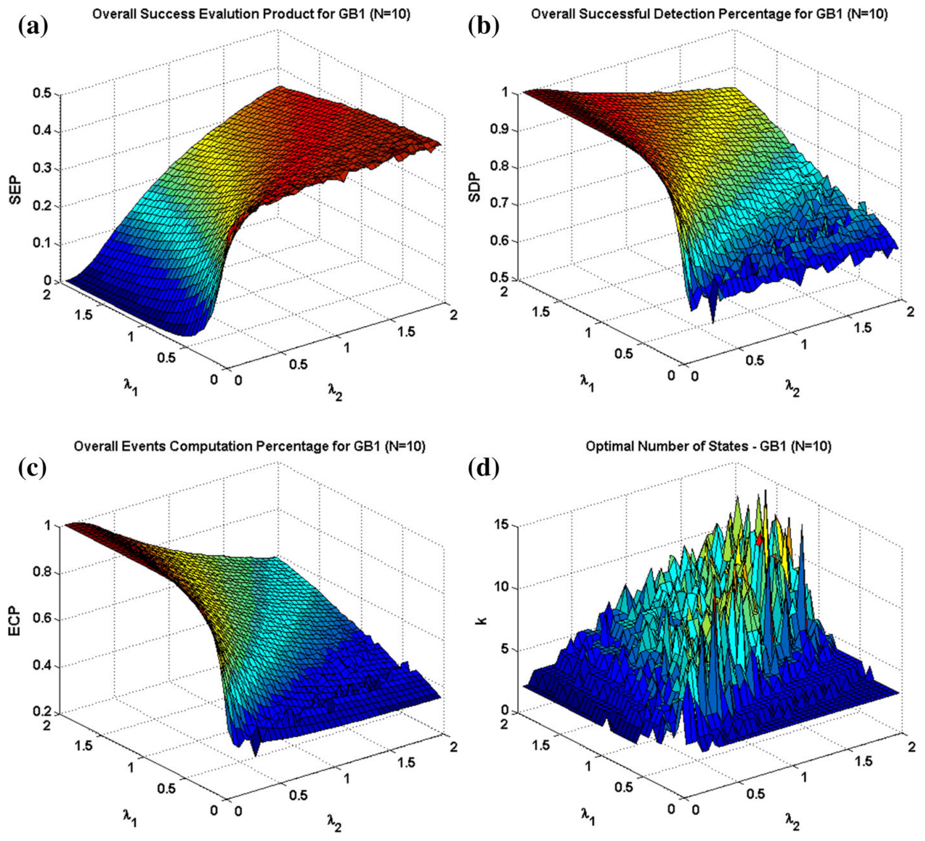


Fig. 5 Evolution of **a** SEP, **b** SDP, **c** ECP with respect to λ_1 and λ_2—GB1, N = 10. **d** Optimal number of states according to λ_1 and λ_2—GB1, N = 10

Results for CMO, GBH Configuration The evaluation of the GBH model gave promising results (Fig. 6a–d). In fact, in the case when $N = 1$ it managed to constantly monitor 50–75 % (Fig. 6b) of events using less than 50 % of the required energy (Fig. 6c). The latter took place with a chain of 3-9 states (Fig. 6d). Only when events are extremely dense ($\lambda_1 \rightarrow 2,\ \lambda_2 \rightarrow 0$) the algorithm performed a lot of monitoring actions without identifying events.

Overall behavior can be justified by considering the fact that upon detecting an eventless period, it decreases its scanning frequency by a factor of $1/2^i$, thus losing any events that follow. Therefore GBH is ideal for environments where events exhibit low or medium densities due to the fact that its aggressive nature facilitates energy savings.

Results for CMO, GBH Configuration with $N = c$ The $N = 10$ configuration in GBH required high amounts of energy only when the density of events was extremely high (i.e. more than 80 % with $\lambda_1 \rightarrow 2,\ \lambda_2 \rightarrow 0$). The latter is depicted in Fig. 7c. In general, GBH with $N = 10$ proved quite robust and managed to constantly detect more than 50 % of events (Fig. 7b) using most of the times half the computations required by the always-on case (Fig. 7c). The latter is achieved with a relatively high number of states (almost 20, Fig. 7d).

The combined view of these observations verifies the intuition that in the absence of events it is preferable to implement an aggressive scanning rate reduction strategy (GBH) than a

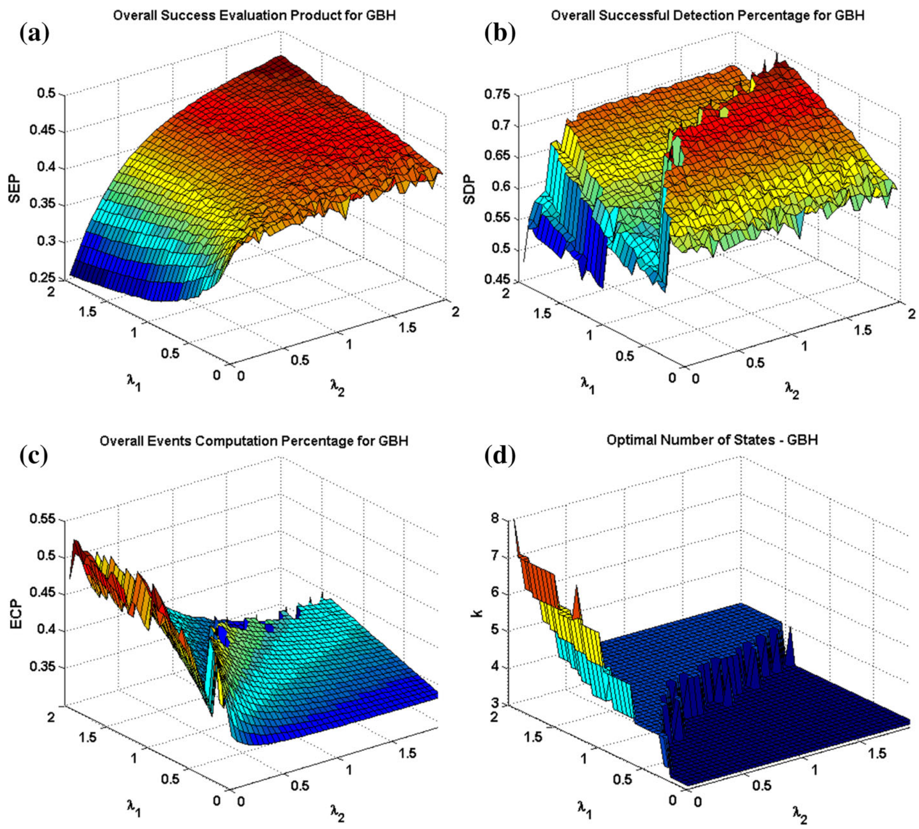


Fig. 6 **a** Evolution of **a** SEP, **b** SDP, **c** ECP with respect to λ_1 and λ_2—GBH. **d** Optimal number of states according to λ_1 and λ_2—GBH

progressive one (GB1). However, depending on the context of application, GB1 can become a viable alternative; for example, if we focus primarily on the detection of events, then GB1 with $N = 10$ gives the best results with a higher cost in terms of computations.

4.2 Real World Experiments

The generic nature of CMO enables its seamless integration in a variety of environments. In the context of evaluating and stress-testing our algorithm in real world conditions, we integrated it in the CONSERN [13] proof-of-concept platform. The algorithm was been integrated in the monitoring module which provided contextual information to all other software entities. The CONSERN monitoring module is a software entity which constantly monitors the operational environment and provides reports regarding the occurrence of specific (network related) events. CMO adjusted the scanning rate of the threads that managed the wireless interfaces based on the occurrence of events in the area. Before delving into the details of the experiments, we detail the cases that signified events in our experiments:

- Change in the required bandwidth more than X% on an interface of the host device
- Change in the packet error rate more than X% on an interface of the host device
- Change in the number of attached clients on the host device

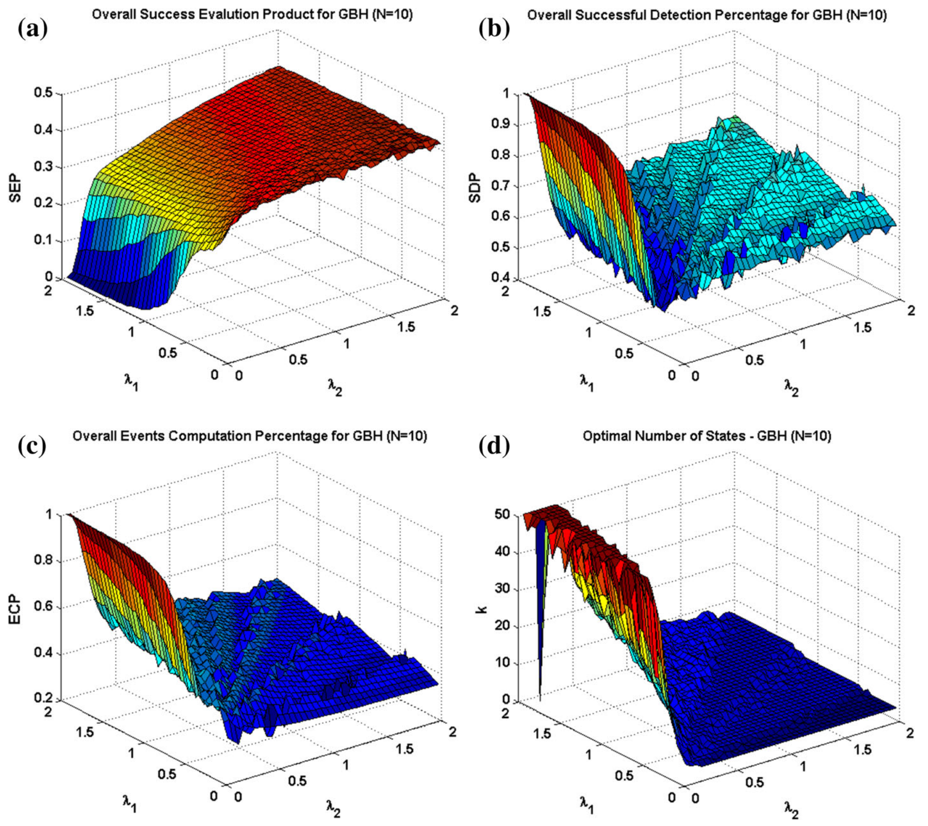


Fig. 7 **a** Evolution of **a** SEP, **b** SDP, **c** ECP with respect to λ_1 and λ_2—GBH, N = 10. **d** Optimal number of states according to λ_1 and λ_2—GBH, N=10

- Change in the TxPower of an interface of the host device
- Change in the monitored noise more than X% on an interface of the host device
- Change in the received signal quality more than X% an interface of the host device
- Change in the number of monitored neighboring APs (e.g. deployment of new devices, change in the channels etc) from the host device

Following this configuration, two real world experiments were executed. The first (small scale) took place in an office environment and is presented in Sect. 4.2.1 while the second one (larger scale) took place in a whole building and is described in Sect. 4.2.2 (Fig. 8).

4.2.1 Experiment 1: A Day in the Office

The first real world experiment took place in our office facilities[3] where 15 researchers work on a daily basis. Four Soekris Access Points (APs) were deployed in our offices following the topology presented in Fig. 9. All devices implemented the CONSERN proof-of-concept environment and therefore had CMO calibrating their monitoring threads. Soekris are compact devices (Fig. 8, [26]) (500 Mhz AMD Geode LX, 512 Mbyte DDR-SDRAM, USB 2.0 interface, one (1) Serial port, eight (8) 10/100 Mbit Ethernet ports and an ATHEROS MINIPCI

[3] Self-evolving cognitive and autonomic networking Laboratory: http://scan.di.uoa.gr.

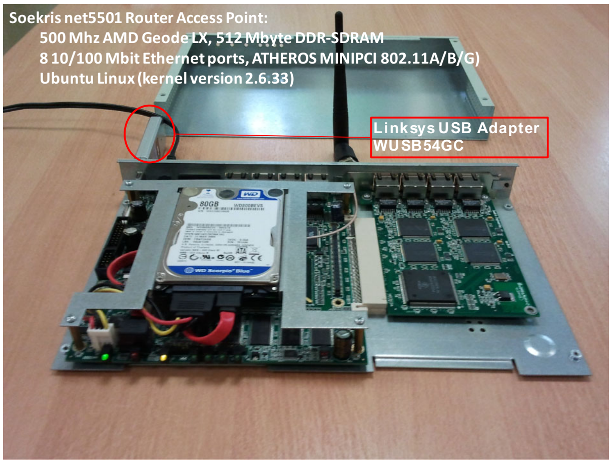

Soekris net5501 Router Access Point:
500 Mhz AMD Geode LX, 512 Mbyte DDR-SDRAM
8 10/100 Mbit Ethernet ports, ATHEROS MINIPCI 802.11A/B/G)
Ubuntu Linux (kernel version 2.6.33)
Linksys USB Adapter WUSB54GC

Fig. 8 Soekris net5501 used in the experiments

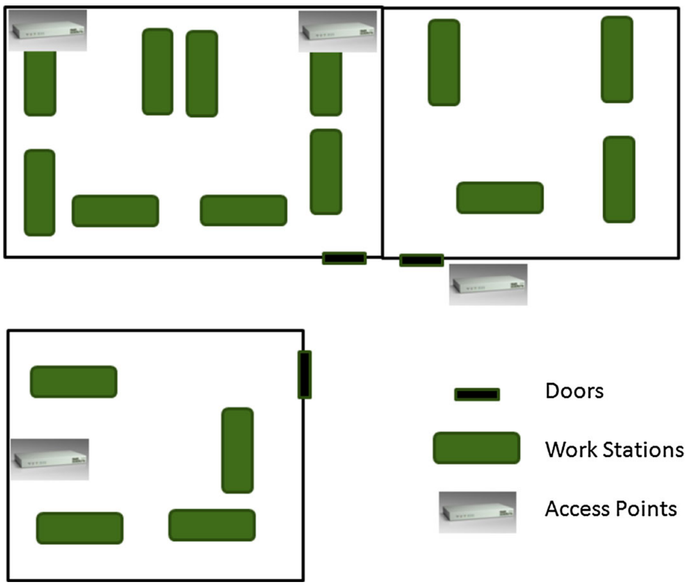


Fig. 9 Physical topology. Offices layout and AP placement

802.11A/B/G [27]) usually employed for networking purposes. They were accompanied with Linux (kernel version 2.6.33); fully programmable for the needs of the prototyping activity. All software has been developed with the use of Java (J2SE 1.7 [28]). Each AP was also equipped with an 802.11 Linksys USB adapter [29].

All APs were equipped with two antennas; one was deployed as an access point, using hostapd daemon [30], while the second one was used to scan the wireless environment. Each AP deployed its own network and routed the information to the internet through NAT. APs were connected through the backbone network and were controlled by a standalone machine which aggregated. The network layout used in the experiments is depicted in Fig. 10. Lab

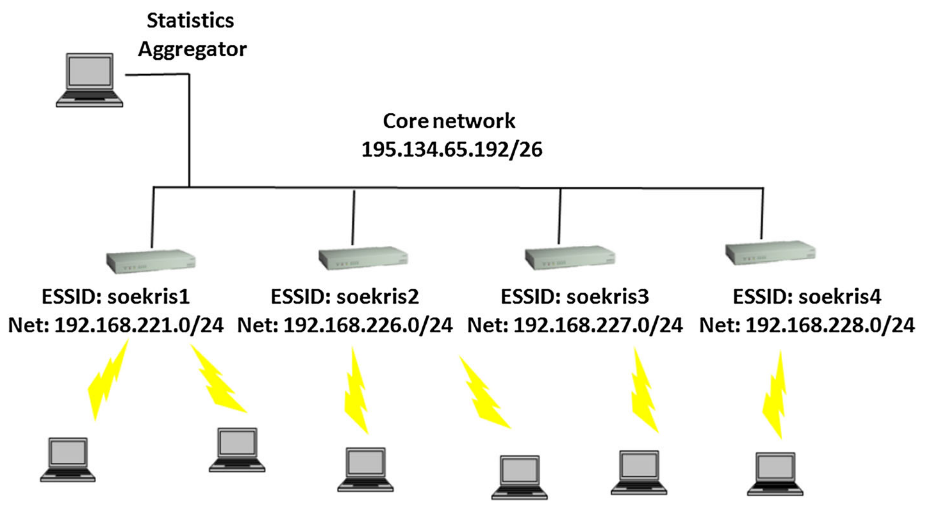

Statistics Aggregator
Core network 195.134.65.192/26
ESSID: soekris1 Net: 192.168.221.0/24
ESSID: soekris2 Net: 192.168.226.0/24
ESSID: soekris3 Net: 192.168.227.0/24
ESSID: soekris4 Net: 192.168.228.0/24

Fig. 10 Network Topology

Table 1 ECP and SDP values retrieved after the 12-h experiment from each AP

	AP 1 (%)	AP 2 (%)	AP 3 (%)	AP 4 (%)
ECP	47	36	36	37
SDP	91	96	98	97

members used these access points for 12 consecutive hours (from 11:00 CET until 23:00 CET on May 28th 2012) in order to access the internet and perform all normal, working-day, activities. Overall traffic throughout the day ranged from 1 to 10 Mbps while APs were configured to serve clients at 5.5 Mbps.

As stated in the beginning of this section, all threads adjusted their scanning rate based on CMO. Due to the fact that the experimentation involved people working, CMO was calibrated in order to maximize the percentage of successful events detection rather than minimize the number of scanning loops. Thus, CMO was configured with GB1 model and $N = 10$ states, since this configuration provides high detection of events capability. We assess the quality of the method using *ECP* and *SDP* as before. In order to perform the comparison we had a second monitoring process, which operated with a fixed sleep interval and run in parallel. Finally, we set X to 10 %.

The obtained results appear in Table 1. As one can observe, CMO achieves a significant reduction in terms of performed monitoring loops (constantly achieving a reduction of more than 50 %) while in parallel manages to retain a high level of events detection (constantly more than 90 %). The best results were obtained on Access Point 3, where the monitoring module performed 64 % less monitoring loops than the always-on case and lost only 2 % of the events that took place. The worst results were obtained by Access Point 1, where a 53 % reduction in the number of loops resulted in a 9 % loss of events. The latter successfully validated in practice the applicability and viability of CMO and paved the way for the large scale expansion trial presented afterwards.

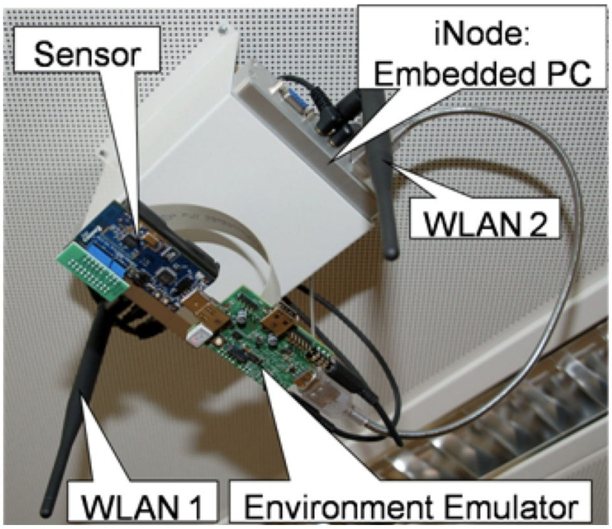


Fig. 11 iNode mounted to the ceiling

4.2.2 Experiment 2: Large Scale Expansion Trial

A second set of experiments was executed in order to stress-test CMO as well as assess its scalability. We used the same implementation and followed the same configuration as before, however this time we used the w.iLab.t testbed [31]. The w.iLab.t testbed is deployed in an office building of 18x90m and is spread out over three floors. It consists of more than two hundred (200) test nodes that are installed at fixed locations at the iMinds [32] office premises in Gent, in meeting rooms, classrooms, offices and corridors. Every w.iLab.t node is generic and is equipped with one or more sensor nodes, an intermediate node with two (2) WiFi 802.11 radios and the environment emulator. Additional radio devices, e.g. Bluetooth, can be added easily through USB interfaces. A photo of a w.iLab.t node is shown in Fig. 11.

The nodes (called iNodes) are Alix 3C3 devices running Linux. These are mini PCs equipped with Ethernet, USB, serial, VGA, audio and two IEEE 802.11 a/b/g interfaces. All the iNodes are connected to the management backbone using Power-over-Ethernet switches, making it possible to remotely switch power on and off to each iNode individually without physical interaction. Each iNode can be configured individually to participate in any specific experiment, and it is possible to adjust the kernel, the driver, to add click router code or to add java-based applications. Finally, the Environment Emulator is located in between the iNode and the sensor node.

Table 2 depicts the results derived from the experimentation of CMO on w.iLab.t testbed. In all experiments, the devices exhibit high SDP values (90 % or higher) with very small variance (0.03). In parallel, and when compared to the non-adjustable monitoring process (i.e. sleep time is fixed), CMO achieves a reduction of more than 50 % in the number of loops (in 11 out of 12 cases, the reduction percentage was reaching 60 %). Only when the environment became extremely dense (115 nodes)—thus events were extremely dense—ECP values surpassed the 50 % threshold and approach 70 %; still however the APs performed 30 % less monitoring loops and managed to identify almost 99 % of events. It is worth pointing

Table 2 CMO experimental assessment on w.iLab.t testbed

Experiment	Number of nodes	Mean ECP	Variance	Standard deviation	Mean SDP	Variance	Standard deviation
1	24	0.414	0.270	0.520	0.932	0.002	0.043
2	24	0.420	0.288	0.537	0.954	0.003	0.054
3	26	0.418	0.275	0.525	0.936	0.006	0.079
4	27	0.420	0.292	0.540	0.959	0.002	0.045
5	45	0.439	0.245	0.495	0.916	0.017	0.130
6	47	0.422	0.297	0.545	0.966	0.001	0.039
7	50	0.429	0.274	0.523	0.942	0.011	0.103
8	50	0.424	0.276	0.526	0.940	0.010	0.098
9	61	0.427	0.291	0.540	0.965	0.002	0.043
10	61	0.421	0.296	0.544	0.955	0.010	0.101
11	68	0.368	0.310	0.557	0.898	0.029	0.170
12	115	0.696	0.085	0.292	0.986	0.001	0.028

out the fact that in all cases, the behavior of the algorithm is in accordance with the theoretic results.

5 Conclusions

In this paper we presented an algorithm that attempts to minimize the monitoring sampling rate while keeping a low percentage of undetectable events. For this we have provided an analytical model, the results of which have been validated in a number of autonomic functions which were implemented and tested in a real environment. The derived results prove the validity and viability of our proposal. The added value of our work lays in its generic nature which renders it as potential candidate for application in several autonomic operations that require monitoring actions.

Our future work will focus on using CMO and its flavors in several mechanisms currently designed to operate in cellular networks (e.g., monitoring of information to perform vertical handovers, sampling for interference identification schemes, clustering of vehicles that need to exchange emergency information). Finally, we plan to develop a learning scheme so as to take into consideration past information and autonomously calibrate the input parameters the algorithm without requiring any human intervention.

Acknowledgments This work has been performed in the framework of the ICT-257542 CONSERN project funded by the European Union.

References

1. Stefania, S., Issam, T., & Matthew, B. (2009). *LTE The UMTS long term evolution. From theory to practice*. Wiley, ISBN: 978-0-470-69716-0.
2. Dobson, S., Denazis, S., Fernández, A., et al. (2006). A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems*, *1*(2), 223–259.

3. Agoulmine, N. (Ed.). (2011). *Autonomic network management principles: From concepts to applications*. New York: Elsevier/Academic Press.
4. Fortuna, C., & Mohorcic, M. (2009). Trends in the development of communication networks: Cognitive networks. *Computer Networks*, *53*(9), 1354–1376.
5. Samaan, N., & Karmouch, A. (2009). Towards autonomic network management: An analysis of current and future research directions. *IEEE Communications Surveys and Tutorials*, *11*(3), 22–36.
6. Tsagkaris, K., et al. (2013). A survey of autonomic networking architectures: Towards a Unified Management Framework. *International Journal of Network Management*, *23*(6), 402–423.
7. 3GPP. *Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements*. 3rd Generation Partnership Project (3GPP), TS 32.500, Dec. (2011).
8. 4G Americas. Self-Optimizing Networks in 3GPP release 11: The benefits of SON in LTE, October 2013. http://www.4gamericas.org/documents/Self-Optimizing%20Networks-Benefits%20of%20SON%20in%20LTE_10.7.13.pdf.
9. Bochow, B., Emmelmann, M., Makris, A., Kaloxylos, A., & Koudouridis, G. (2012). *The self-growing concept as a design principle of cognitive self-organization*. IEEE Globecom.
10. Chaparadza, R., Coskun, H., & Schieferdecker, I. (2005). Addressing some challenges in autonomic monitoring in self-managing networks. In *13th internation conference on networks*.
11. Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, *5*(1), 4–7.
12. Makris, P., Skoutas, D., & Skianis, C. (2013). A survey on contexto-aware mobile and wireless networking: On networking and computing environments' integration. *IEEE Communications Surveys and Tutorials*, *15*(1), 362–386.
13. FP7-ICT CONSERN. https://www.ict-consern.eu/.
14. Zheng, R., Hou, J. C., & Sha, L. (2003). Asynchronous wakeup for ad hoc networks. In *Proceedings of the 4th ACM international symposium on mobile ad hoc networking and computing* (pp. 35–45). New York: ACM Press.
15. Polastre, J., Hill, J., & Culler, D. (2004). Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on embedded networked sensor system (SenSys)*.
16. Hohlt, B., Doherty, L., & Brewer, E. (2004). Flexible power scheduling for sensor networks. In *Proceedings of the 3rd international symposium on information processing in sensor networks (IPSN)*.
17. Van Dam, T., & Langendoen, K. (2003). An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the international conference on embedded networked sensor systems (SenSys)*.
18. Zheng, R., & Kravets, R. (2003). On-demand power management for ad hoc networks. In *Proceedings of the annual joint conference of the IEEE computer and communications societies (InfoCom)*.
19. Chipara, O., Lu, C. & Roman, G.-C. (2005). Efficient power management based on application timing semantics for wireless sensor networks. In *Proceedings of the 25th IEEE international conference on distributed computing systems, 2005 (ICDCS 2005)*, pp. 361–370, June 10, 2005.
20. Yoo, S., & Golmie, N. (2010). Policy-based scanning with QoS support for seamless handovers in wireless networks. *Wireless Communications and Mobile Computing*, *10*, 405–425. Wiley InterScience.
21. Kim, K. H., Min, A. W., Gupta, D., Mohapatra, P., & Singh, J. P. (2011). Improving energy efficiency of Wi-Fi sensing on smartphones. In *INFOCOM, 2011 Proceedings IEEE* (pp. 2930–2938). IEEE.
22. Alshurafa, N., et al. (2014). Battery optimization in smartphones for remote health monitoring systems to enhance user adherence. In *Proceedings of the 7th international conference on PErvasive Technologies Related to Assistive Environments*.
23. Musolesi, M., et al. (2010). *Supporting energy-efficient uploading strategies for continuous sensing applications on mobile phones. Pervasive computing* (pp. 355–372). Berlin: Springer.
24. Magdalinos, P., Kousaridas, A., Spapis, P., Katsikas, G., & Alonistioti, N. (2011). Enhancing a fuzzy logic inference engine through machine learning for a self-managed network. *The Mobile Networks and Applications (MONET) Journal: Special Issue on Mobile Networks and Management*, *16*(4), 475–489.
25. Mihailovic, Andrej, Kousaridas, Apostolos, Jaron, Alexandre, Pangalos, Paul, Alonistioti, Nancy, & Hamid Aghvami, A. (2013). Self-management for access points coverage optimization and mobility agents configuration in future access networks. *Wireless Personal Communications*, *72*(1), 343–374.
26. Soekris Engineering Inc. http://www.soekris.com.
27. Atheros MINI PCI on Wistron Neweb 9 card. http://www.wneweb.com/.
28. The Java Programming Language. http://www.java.com.
29. Linksys USB WLAN Adapters. http://homesupport.cisco.com/en-us/wireless/lbc/wusb54gc.
30. Hostap. http://hostap.epitest.fi/hostapd/.
31. The w.iLab.t testbed. http://www.crew-project.eu/portal/wilabdoc.
32. iMinds. http://www.iminds.be/en.

Panagis Magdalinos received a B.Sc. degree in Informatics and Telecommunications from the University of the Athens and an M.Sc. in Information Systems from Athens University of Economics and Business. In 2010 he acquired his Ph.D. diploma entitled "Linear and Non Linear Dimensionality Reduction for Distributed Knowledge Discovery" from the Department of Informatics of the Athens University of Economics and Business. Since 2004 he has participated in a number of European research projects, namely E^2R, E^2R II, E3, SelfNET, CONSERN, SmartAgrifood, METIS and LiveCity. He is currently working as a data scientist in Intracom Telecom S.A. His research interests focus on supervised and unsupervised knowledge extraction from distributed data collections (e.g., Learning and Mining in Distributed Environments, Parallel Data Mining).

Alexandros Kaloxylos received the B.Sc. degree in Computer Science from the University of Crete, Greece, in 1993, the M.Phil. degree in Computing and Electrical Engineering from the Heriot-Watt University, Scotland, in 1994 and the Ph.D. degree in Informatics and Telecommunications from the University of Athens in 1999. From 1990 to 1993 he was a staff member of the Computer Centre of the University of Crete, and a researcher in the Foundation of Research and Technology Hellas (FORTH). From 1994 to 1995 he was a research associate at the University of Wales. From 1995 until 2014 he was a senior researcher at the Communications Network Laboratory of the University of Athens. In 2002 he joined the faculty of the University of Peloponnese, where he is presently an Assistant Professor in the Department of Informatics and Telecommunications. Since June 2014 he is working as a Principal Researcher in Huawei's European Research Center in Munich. He has participated in numerous projects realized in the context of EU programs as well as National Initiatives. He has published over 100 papers in international journals, conferences and book chapters. He is a senior member of IEEE, a member of the editorial board of the IEEE Communication's Society Surveys and Tutorials Electronic Journal and TPC member in numerous international conferences.

Nancy Alonistioti has a B.Sc. degree and a Ph.D. degree in Informatics and Telecommunications (Department of Informatics and Telecommunications, University of Athens). She has working experience as senior researcher and project manager in the Dept. of Informatics and Telecommunications at University of Athens. She has participated in several national and European projects, (MOBIVAS, ANWIRE, E^{2}R, LIAISON, E^3, SELFNET, SACRA, CONSERN, UNIVERSELF etc) and has experience as Project and Technical manager of the IST-MOBIVAS, IST-ANWIRE, ICT-SELFNET, ICT-CONSERN projects, which had a focus on reconfigurable mobile systems, cognitive mobile networks and future internet. She is co-editor and author in "Software defined radio, Architectures, Systems and Functions", published by John Wiley in May 2003. She has served as lecturer in University of Piraeus and she has recently joined the faculty of Dept. Informatics and Telecommunications of University of Athens. She is TPC member in many conferences in the area of mobile communications and mobile applications for systems and networks beyond 3G. She has over 100 publications in the area of mobile communications, reconfigurable, cognitive and autonomic systems and networks and Future Internet.