A Context-Aware Architecture Supporting Service Availability in Mobile Cloud Computing

Gabriel Guerrero-Contreras, José Luis Garrido, Sara Balderas-Díaz, and Carlos Rodríguez-Domínguez

Abstract—Mobile systems are gaining more and more importance, and new promising paradigms like Mobile Cloud Computing are emerging. Mobile Cloud Computing provides an infrastructure where data storage and processing could happen outside the mobile node. Specifically, there is a major interest in the use of the services obtained by taking advantage of the distributed resource pooling provided by nearby mobile nodes in a transparent way. This kind of systems is useful in application domains such as emergencies, education and tourism. However, these systems are commonly based on dynamic network topologies, in which disconnections and network partitions can occur frequently, and thus the availability of the services is usually compromised. Techniques and methods from Autonomic Computing can be applied to Mobile Cloud Computing to build dependable service models taking into account changes in the context. In this work, a context-aware software architecture is proposed to support the availability of the services deployed in mobile and dynamic network environments. The proposal is based on a service replication scheme together with a self-configuration approach for the activation/hibernation of the replicas of the service depending on relevant context information from the mobile system. To that end, an election algorithm has been designed and implemented.

Index Terms—Context-awareness, software architecture, mobile cloud computing, Service Oriented Architecture (SOA), self-configuration, election algorithm, service availability.

1 INTRODUCTION

In recent years, there has been a considerable increase in the use of mobile applications and systems [1]. The number of systems that are exclusively based on mobile technologies is increasing, and new computation paradigms (e.g., Cloud Computing) provide advances in these systems. Cloud Computing helps to address one of the main weaknesses of mobile systems: the lack of computational resources. The Cloud allows the delegation of some processing and storage tasks that must otherwise be carried out in nodes with limited resources. This opens up new possibilities for mobile systems [2], [3], and as a result the concept of Mobile Cloud Computing has emerged, that is, “an infrastructure where data storage and processing could happen outside the mobile device”, which can operate under three main schemes [4]:

1) The traditional scheme, where the mobile node delegates part or all of its operation to the Cloud, through an Internet connection. This is performed under a client-server paradigm, where the mobile node acts as a regular client of the Cloud. Other additional approaches take into consideration the degree of computational offloading on the Cloud [5].

2) A Local Mobile Cloud or Spontaneous Mobile Cloud [5]. In this scheme a set of mobile nodes make up a local Cloud between them through short-range connections. The storage or processing is carried out through a distributed process between the nodes that make up the local network. This schema is of special interest in domains where a stable Internet connection cannot be established, and thus it is not possible to connect with an external Cloud.

3) Finally, a hybrid scheme, where the mobile system is supported by a Local Mobile Cloud, which is made up of the nodes of the mobile system, and, at the same time, it is supported by an external Cloud under a traditional scheme.

Currently, the first scheme is widely spread, as nowadays the majority of the current applications are offered through a SaaS (“Software as a Service”) service model [6] (e.g., Google Apps: Calendar, Gmail, Drive, etc.). The Cloud technologies and SaaS model are closely related to SOA (“Service Oriented Architecture”) [7]. While SOA is a model for the design and development of software systems, Cloud and SaaS are models for their software implementation and delivery, respectively [8]. Thus, it is especially interesting to complement SOA and SaaS models to develop different services at different levels of abstraction [9], which can be accessed from mobile devices connected through the Internet.

The second scheme, the Local Mobile Cloud, is required in domains where mobile applications also need to be supported by data storage and processing services provided by the mobile platform itself in a transparent and flexible way, such as rescue teams [10], security forces [11], tourism [12], et cetera. This is of special interest in case of disasters in remote areas (e.g., natural disasters, accidents, terrorist attacks, etc.), where usually the common network infrastructures are not available and the governmental agencies need to apply action protocols to support victim search, rescue, and identification, managing effectively and efficiently the action plan (situation, intervention and risks), resources
(human, material and technological) and the cost of all involved parts [13].

Nevertheless, the Local Mobile Cloud approach is usually supported by a Mobile Ad-hoc NETwork (MANET), which poses new challenges [14] (e.g., battery constraints, limited bandwidth, dynamic topology, transmission errors, or routing overhead, etc.) that must be conveniently addressed in order to guarantee the dependability of the Mobile Cloud. The concept of dependability involves the quality attributes of availability, reliability, safety, integrity, maintainability [15], of which availability is particularly vulnerable to the dynamicity of the network topology that characterizes the mobile networks. This dynamicity is produced by the mobility of the nodes that make up the network. Further, the nodes may be switched off or may be disconnected (temporarily or permanently). Since these networks are typically multi-hop, this usually implies link failures, route changes or even network partitions, which could have a deep impact on the availability of the services deployed in the network.

The availability of a service is directly proportional to its number of active replicas. However, an intensive replication will require intensive energy consumption. Therefore, it is also necessary to provide an energy-aware solution that balances the performance of the system with efficient energy consumption. For this reason, software architects must make suitable design decisions to address availability in such a way that decisions about service deployment and replication are taken at run-time. As a result, self-adaptive architectures have been gaining importance in the research community [16]. A self-adaptive architecture has been complemented with self-* features (i.e., self-healing, self-configuration, self-optimization and self-protection [17]), and it has the capability of reducing the consequences of context changes in the quality attributes of the system.

This paper introduces a context-aware software architecture, which aims to provide an adaptable and energy-efficient solution to support the availability of services in mobile clouds. The proposal is based on service replication techniques together with a self-configuration approach for the activation/hibernation of the replicas of the service depending on relevant context information from the mobile system. To that end, an election algorithm has been designed and implemented. This proposal is based on previous work [18], which allows us to now address in detail all the structural and behavioural design issues, as well as the context information model, for the complete definition of the architecture. Additionally, its implementation and a simulation on the Network Simulator 3 (ns-3) have been performed, which allows us to show the feasibility of the architecture. The architecture proposed is designed to support medium or small size networks. It follows a copy-primary scheme [19] in physical partitions, which is inefficient in large networks, where the active replica could easily result in a bottleneck. Clustering techniques can be applied to provide scalability to system, through the creation of node groups, i.e., a set of nearby nodes that working together, sharing the same service replicas.

The rest of this paper is organized as follows. Section 2 presents a study of the main aspects that have been addressed in the literature in order to improve the availability of services in dynamic environments. Section 3 introduces the structural design of a context-aware software architecture to support the availability of services in mobile clouds, while Section 4 introduces the election algorithm for the run-time activation/hibernation of the service replicas. In Section 5 some example scenarios are presented in order to show how the proposed architecture addresses different situations that can affect to the availability of services in mobile clouds. Section 6 presents the evaluation of the architecture on the ns-3 network simulator and its results. Finally, Section 7 provides a brief discussion and Section 8 summarises conclusions and outlines plans for future research.

2 RELATED WORK

Several approaches to address the availability of the services in mobile environments can be found in the literature. Generally, they are based on adaptive schemes, following the reference model for autonomic control loops, MAPE-K (Monitor, Analyse, Plan, Execute, and Knowledge) [20]. In the case of this kind of systems, two steps in the adaptation loop provide a major impact on their efficiency: when to replicate and where to deploy a service.

There are different events that can trigger the creation, migration or deletion of replicas: the battery of the host node is running out, or the node switches off, the demand for the service increases, along with others. In addition to these events, it is worth mentioning the prediction of a network partition. A network partition can affect the availability of a service and the consistency of the shared information [21]. The fact of predicting a partition allows the necessary actions to be taken to ensure the consistency of the shared information and the availability of the service, while the connection is still available. Chandrakala et al. [22] propose a prediction algorithm based on the position of the nodes, their travel speed and range. When a partition is predicted, the node in which the service replica will be deployed is chosen by the distance from the source node (i.e., the node that has a copy of the service and from where the replica will be created and sent), its battery and storage capacity. In [12], the TORA routing protocol is used in combination with an estimation of the residual link lifetime of wireless links. When a node predicts a partition, this node will host a replica of the service, regardless of its characteristics.

While in other applications domains the partition prediction algorithms are necessary to anticipate a partition and replicate the service, ensuring its availability in both partitions, in systems supporting work teams there are other options. In this kind of systems [10], [23], the set of services is well-known at deploy time, thus approaches based on hibernation [24], [25], in which the replicas of the services can be deployed in each node, are applicable. On the one hand, these approaches provide the following advantages: (1) better response times; (2) the service does not have to be migrated at real time, and hence, the bandwidth and energy use is more efficient; and (3) the system is less dependent on the available technology as the requirements on bandwidth

are less stringent. On the other hand, the system will be less flexible as new services cannot be introduced in the network at run-time.

In large scale systems [14] scalability is a main objective for architects. Node cluster methods are applied in order to turn a distributed network into a set of interconnected local clusters that can be handled individually, like a centralized network. In this way the management of the network is simplified, achieving scalability under a “divide and rule” approach. Psaiar et al. [24] create node clusters on the basis of the distance between mobile nodes. However, in [26], [27] it is shown that the speed of nodes is a better measure to create clusters of mobile nodes. In [27], service replicas are created when too many requests are made to a service from an external group. That system aims to achieve the property of localized scalability [28] by reducing the communications between distant entities, in order to safeguard the bandwidth and the energy of the system. In both systems [24], [27], the node that will host the new replica is chosen by considering its computational capabilities (battery, CPU, memory, etc.), without taking into consideration either its current workload or the network topology. Consequently, the resources of the host node can be quickly depleted.

In [29], it has been identified a set of high-level Mobile Cloud Computing oriented primitives, and the Networked Autonomic Machine (NAM) framework has been enhanced to include them and a formal characterization of its semantics. That has led to the formalization of a framework to support the design of Mobile Cloud Computing systems.

Finally, Hamdy et al. [25] propose a replication protocol based on the interest of nodes in the use of the service. When an application needs to access a service and there is not a replica of that service in its neighbourhood, a replica of the service will be created and deployed in the same node in which the application is hosted.

Generally, these ad-hoc solutions have been developed for specific scenarios, and they are based on an implicit, and often restricted, context model. The definition and use of an explicit context model can facilitate a wider adoption and applicability of the proposal. The possibility of extending the model according to particular requirements for each system would provide a more efficient solution in terms of energy consumption to several scenarios. In respect to address the dependability in Cloud computing, recent works [30], [31], are mainly focused on the traditional Cloud scheme (as mentioned in the previous section), in which the issues related with battery constraints, limited bandwidth, dynamic topology, or transmission errors have less impact on the system.

3 Design of a Context-Aware Architecture

In this section, the structural design of the context-aware architecture will be introduced. The architecture is made up of four main elements (Figure 3.1): Communication Middleware, Monitoring Subsystem, Context Manager Service, and Replica Manager Service. These elements will be replicated in each node of the network, and its union and coordination will provide an energy-efficient solution to satisfy the availability of a service in a Mobile Cloud.

The Communication Middleware allows communication between the different entities of the system, addressing the heterogeneity problem of mobile environments. The Monitoring Subsystem is made up of the different monitoring services. These services are responsible for monitoring relevant context information of each node and its environment. The information gathered by those monitoring services is processed and stored by the Context Manager Service, in order to provide pertinent information to the Replica Manager Service. The Replica Manager is responsible for coordinating and achieving an efficient configuration for the activation/hibernation of the replicas of the service according to the current state of the mobile network.

3.1 Communication Middleware

The Communication Middleware [32] allows communication between the different entities of the system under two different communication paradigms: (1) a Publish-Subscribe paradigm, where for example, a Monitoring Service notifies the Replica Manager Service about events of its interest, through a push-based communication model. For instance, an event to report the battery level every time that the battery level decreases by 5%, or an event when the active replica is not reachable; and (2) under a Request-Response paradigm, when, for example, the Replica Manager requests the Context Manager about information to evaluate the adequacy of the node to host a service replica and to obtain a score that reflects it. The combination of both paradigms is usually known as SOA 2.0 [33] (a.k.a. advanced SOA), in which services are not just passive entities, but are also able to receive and generate events proactively.

3.2 Monitoring Subsystem

The Monitoring Subsystem encompasses a set of monitoring services, which sense the context information in relation to the node in order to detect potential events that could affect the availability of a service. In the proposed architecture, both the node capabilities (e.g., remaining battery, storage, memory, etc.) and the network topology are monitored. However, to monitor the context is a costly operation in terms of energy and bandwidth. Nevertheless, the response time of a self-adaptive system depends on the precision of the monitoring action. The information about the node capabilities is local information that usually can be obtained easily, while a dynamic network topology requires a continuous monitoring because of constant changes that it suffers. Further, the cost increases exponentially with each hop in the network.

In this work it is proposed to use the information provided by the routing protocol to estimate the network
topology. However, according to the OSI (Open Systems Interconnection) model the routing protocols are designated as network layer responsibility, it has been proved that a more flexible design is required for managing the dynamics of mobile environments, where the network status is obtained from a cross-layer interaction [34].

The routing protocol builds and maintains in each node a routing table with information about the reachable nodes, and for each of them the gateway and the number of hops. Through the direct connections of a node, the system can approximate its position within the network topology. For instance, a node with four direct connections will be in a more centric position within the node group, and therefore will be a more stable node than other ones with only one direct connection. However, it does not provide the exact position of a node, but it can provide enough information to successfully approximate the network topology. Note that this approach requires proactive routing protocols [35], which periodically update routing tables according the network topology changes.

### 3.3 Context Manager Service

The Context Manager Service is responsible for processing and storing the information received from the Monitor Services. This information will be used by the Replica Manager Service in order to adjust the configuration of the activation/hibernation of the replicas according to the changes produced in the context.

Anind K. Dey defined the context as “[... ] any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves” [36]. Accordingly, one of the main issues in the design of a context-aware system is to identify the relevant information that can help to adjust dynamically the functionality of the system.

The diagram depicted in Figure 3.2 shows the context model managed by the Context Manager Service. Although all the nodes share the same context model, the information stored is mainly local to the node and only the score of the nodes is shared between them, in order to reduce bandwidth consumption. The importance of the different context information that may influence the activation/hibernation of services will depend on the particular application domain. Accordingly, in addition to common context information (such as network topology or node battery), the proposed context model considers the possibility of adapting the context model to the specific requirements of the node that a service will require. This would allow the system to distribute the workload among the nodes of the network, providing the appropriate deployment of the services according to their requirements. In the proposed data model, the requirements of a service are specified in an XML file (see [23] for more detailed information). This file will contain for each computational requirement of the service its optimal, normal and critical values. For instance, if a service will need storage capabilities its requirement file will contain the following sentence:

```xml
<resource name="Storage"
      resourceId="#CR04"
      optimumValue="4GB"
      normalValue="2GB"
      criticalValue="1GB"></resource>
```

This sentence indicates that the service requires almost 1GB of storage to operate correctly. Moreover the number of clients (i.e., applications and other services), dependencies between services (i.e., regarding service composition) and number of replicas of the service, are also relevant context information to provide an efficient service deployment solution.

- **Network topology.** Usually monitoring and obtaining a precise representation of the topology of a multi-hop mobile network is a costly task, due to the constant number of changes and their propagation between the different nodes of the network. Thus, the routing protocols for dynamic network topologies usually try to make an easy topology representation by considering only the best path between two nodes [35]. However, it is enough to obtain the number of direct links of a node, which is a measure that is simple to obtain and powerful enough to provide an efficient configuration of activation/hibernation.
of the replicas among the network.

3.4 Replica Manager Service

The Replica Manager Service performs the election algorithm to achieve an efficient configuration for the activation/hibernation of the replicas of the service. It follows a component-based design (Figure 3.3), which allows reuse that components and even the dynamic adaptation of its behaviour through the exchange of such components at runtime. It is made up of three main components:

- **Evaluator (Evaluation component).** This component contains the evaluation function that will be used by the Coordinator component to evaluate the suitability of the node to host a specific service. The evaluation is based on the information provided by the Context Manager Service through the IContextInformation interface (using Publish-Subscribe paradigm). The weight of each context feature will depend on the application domain and the specific situation. For instance, if the battery of a system is running low, then the evaluation function can dynamically change to assign more priority to this context feature. Therefore, different Evaluator components could be defined to address different situations that could happen in the same application domain.

- **Triggers (Control component).** This component contains a set of rules that describes the trigger activation policies of the Replica Manager Service. The trigger policies implemented take as input the context events provided by the monitoring services, through the Event Listener component and the IEventListener interface (using the Request-Response paradigm). The activation of any of these rules activates the Coordinator component, and thus the election process too. For example, these rules could specify the following actions and activation policies: replicate a service when the resources of the current host node are running out (e.g., the battery power is below 20%), when the topology of the network changes (e.g., a partition is predicted or the active replica is not already reachable) or when new clients are present; migrate a replica when the distribution of the clients changes within the network or a better candidate node to host the replica is discovered; or hibernate a replica when the number of clients is reduced. This component allows the system to adapt to the particular requirements of a specific application domain.

- **Coordinator (Manager Component).** When a change in the context is detected that could affect the quality attributes of the service, the Coordinator component will be responsible for coming to an agreement with the rest of the coordinators deployed in the system. The objective of this coordination is to know if there is a better activation/hibernation of the replicas of the service and, if so, to establish what replica will be activated. Once the election is performed, the activation or hibernation of a replica is performed through the IServiceActivation interface. It should be emphasized that the election process can be initiated by any Coordinator of the set. This is because, in order to save bandwidth, each of them has local information that is not accessible to the rest, for instance, the remaining battery of the node.

4 Election Algorithm

This section describes the election algorithm that is performed by the different replicas of the Replica Manager Service to achieve an efficient configuration for the activation/hibernation of the replicas of the service according to the current status of the mobile network. There are three possible states for each node: “Local Mode”, “Client Mode” or “Server Mode”. A node is in the “Local Mode” state when there is no reachable node in its neighbourhood. Moreover, this state is used as the initial state when a context change triggers the election process to modify the current configuration for the activation/hibernation of the replicas of the service (Pseudocode 1). The availability of the service and the response time of the system against changes of the context will be directly proportional to the time required for the transition between the “Local Mode” state and the new state (“Client Mode” or “Server Mode”). The election algorithm between nodes is based on the exchange of the following messages:

- **SCORE.** Through this message a node communicates its score. The score is expressed through a float number, in the range 0 and 1, where 1 represents the best score attainable. This score is obtained through the evaluation function implemented in the “Evaluator” component at the Replica Manager Service (Section 3.4).

- **SERVER_REQUEST.** This message is used when the requesting node wants to establish a client-server connection with the receiver node, where the requesting node will act as client and the receiver node as server. The receiver node must respond to this request, depending on its condition to act as server. If there is another node that can act as server, from the point of view of the receiver node, it will reject the request. Else, it will accept it.

- **SERVER_REJECTION.** This message is used to respond to a SERVER_REQUEST message. Through this
message, the receiver node of a SERVER_REQUEST rejects the request to act as server for the sender node. In the case where the receiver node has another node established as server, it will attach in the SERVER_REJECTION message information about this server (its ID and its score), in order to complete the information about the network of the issuer node.

- SERVER_ACCEPTANCE. This message is used to respond to a SERVER_REQUEST message. Through this message, the receiver node of a SERVER_REQUEST accepts the request to act as server for the issuer node. Therefore, when the sender node of a SERVER_REQUEST message receives the SERVER_ACCEPTANCE, it will establish the client-server connection and it will start to use the service provided.

**Pseudocode 1** On activation of a trigger policy

1. \( state := \text{LOCAL\_MODE}; \)
2. if the neighborhood is not empty then
3. proceed to election;

The election algorithm proceeds according to the Pseudocode 2, which is represented visually in the state machine diagram of Figure 4.1.

**Pseudocode 2** The election algorithm to self-configure the activation/hibernation of the replicas

1. \( \text{myscore} := \text{calculateScore();} \)
2. broadcast \( \text{myscore} \) to all know nodes;
3. wait until (number of scores received = \( N \) or the timer expires);
4. \( \text{candidateNode} := \text{calculateBestNode();} \)
5. if \( \text{candidateNode} = \text{myID} \) then
6. \( state := \text{SERVER\_MODE}; \)
7. else
8. send request to \( \text{candidateNode}; \)
9. wait until (the \( \text{candidateNode} \) response or the timer expires);
10. if the timer is expired then
11. go to line 8;
12. else if SERVER_ACCEPTANCE received then
13. \( \text{serverID} := \text{candidateNode}; \)
14. \( state := \text{CLIENT\_MODE}; \)
15. else
16. if SERVER_REJECTION has additional information then
17. \( \text{candidateNode} := \text{calculateBestNode();} \)
18. else \( \Rightarrow \) In the case that the SERVER_REJECTION has not additional information
19. wait \( T; \)
20. go to line 8;

When a node starts the election, the first step is to calculate its score and broadcast it to its group. Then, the node goes into a passive mode where it waits to receive the scores of its neighbours. When the node receives all the scores of its neighbours, or the timer expires, the node calculates what is the most suitable node (within its group) to act as server. The use of the timer is because of message loss. In an ideal network, where there is no message loss, this timer would be meaningless. However, in this kind of environment the reception of a message cannot be guaranteed, thus if a node does not receive a score from its neighbour node within a specific time period, it can assume that the message has been lost.

In the calculation of the best node, the node will select the node of the group with the best score, this node could be itself. The score of the nodes is replicated information, this is, all the nodes within the group manage the same list of scores. Hence, all the nodes will take the same decision about what node will act as server, if there is no message lost. In the case of the best node will be the node itself, it will go into the “Server Mode” directly, assuming that the rest of the nodes will take the same decision. This approach avoids the server having to communicate its selection to the rest of the nodes, improving the use of the bandwidth. In the case where another node of the group is the best, the node will send a message requesting service to that node. At this point, one of the following situations can happen:

- The requested node is in the “Server Mode” state.
  Therefore, the requesting node will receive an affirmative answer (i.e., the SERVER_ACCEPTANCE message) and it will go to “Client Mode”.
- The requested node is in the “Client Mode” state.
  The requesting node has not been sent the request to the best node of the group. This can be caused by two reasons: (1) the requesting node does not already know all the nodes of the network, (2) some of the score messages have been lost. Thus, the requesting node has taken a wrong decision because it does not have all the information about the group.

![Fig. 4.1. The transition process between “Local Mode” state and the “Client Mode” or “Server Mode” states.](image)
In this case, the requested node will respond with a `SERVER_REJECTION` message. Additionally, this message will include information about the server of the requested node and its score, in order to complete the information of the requesting node. Therefore, the requesting node will calculate again the best node, including this new information, and it will make the request to the best node correctly.

- The requested node is still in transition. At this point, the requested node will return a `SERVER_REJECTION` message, without additional information. The requesting node will repeat the request after the `SERVER_REJECTION`, to the same node if it has not received new scores. Usually, the transition to the requested node has been already completed, and it will accept or reject the request, according to its state (“Client Mode” or “Server Mode”).
- The request message or the acceptance/rejection message is lost. In this case, once that the waiting timer has expired, the request will be made again.

Finally, in the case of the “Server Mode” state, when the node receives a request for service (`SERVER_REQUEST`), it accepts the request through the message `SERVER_ACCEPTANCE` (Pseudocode 3).

## 5 Examples

The Local Mobile Cloud can be of special interest in scenarios where people use local collaborative tools. Concretely, in case of disaster scenarios in remote areas, the possibility to use a support system that allows managing effectively and efficiently the action plan (situation, intervention and risks) and resources (human, material and technological) is a priority [13]. In this section three different scenarios are provided in order to illustrate the operation and utility of the proposed architecture: scenario 5.1 shows the behaviour of the proposed architecture in the case of network partitions and combinations; scenario 5.2 illustrates the behaviour of the system in case of message loss; and scenario 5.3 reveals how the system configure the activation of the services according to its computational requirements.

### 5.1 Scenario 1

Figure 5.1 depicts a hypothetical scenario where three members of a work team are investigating two related scenes. In this scenario two kinds of nodes can be found: (1) a laptop deployed statically in a vehicle (Node 1); and (2) three mobile nodes, one for each team member (Nodes 2, 3 and 4). Also, different services can be found, for instance an Image Repository service. This service allows that each user takes pictures of evidence with his mobile device; different users can photograph the same evidence; or the same user can photograph pieces of evidence that belong to different scenes. Also, the users can add annotations to these pictures, which can be linked to other pictures. Therefore, the Image Repository service must keep an ordered and consistent set of this information, and at the same time it must provide high availability, to allow accessing and sharing pictures of the evidence found.

In this scenario, the proposed context-aware architecture can be useful to satisfy the availability of the Image Repository service. The situation will proceed as follows:

A) Initially, all the users are near scene 1. All of them have a connection with the laptop located in the vehicle. Therefore, the active service replica is the one deployed in the laptop (Node 1 in Figure 5.1), as it has better computational features.

B) The user of Node 4, currently in the “Client Mode” state, will move to the area where scene 2 is located. This area is out of the laptop range. The Network Monitor Service detects this situation by way of a disconnection prediction method or because the active replica is not already reachable. Thus, it emits the proper event that is received by the Replica Manager Service, through the Context Manager Service. The Replica Manager Service has this situation reflected in its trigger policies and it starts the election process. As the only replica available is itself, it will transition to “Local Mode”. The deployment of Nodes 1, 2 and 3 does not change, since, although the network topology has changed, the server is still available and no new nodes that could be new host candidates have appeared.

C) At this point, the network is partitioned into two groups: (1) Nodes 1, 2 and 3, where Node 1 hosts the active replica; and (2) Node 4, that provides service to itself.

D) Later, the users of Nodes 2 and 3 move to the area of scene 2.

E) Nodes 2 and 3 lose the connection with Node 1, and establish a connection with Node 4. Thus, all the Nodes transition into the “Local Mode” state. Nodes 2, 3 and 4 exchange their scores, for instance, the following ones can be assumed: Node 2, 0.6; Node 3, 0.67; and Node 4, 0.75. Once the nodes have the score list, the nodes select the best ranked node. Node 4 will transition directly to the “Server Mode” state, as it is the best ranked. Nodes 2 and 3 will send a `SERVER_REQUEST` message to Node 4, which will respond with a `SERVER_ACCEPTANCE` message to each of them. When Nodes 2 and 3 receive the `SERVER_ACCEPTANCE` message they transition to the “Client Mode” state.

F) Finally, Nodes 2, 3 and 4 make up a group, where Node 1 hosts the active replica; and Node 1, is in the “Local Mode” state, since there are no nodes in its area.

This scenario highlights how the Image Repository service can be provided to the users under a SaaS model supported by the distributed resource pooling provided by the set of mobile nodes, and how the proposal helps to satisfy the availability of the storage service against network partitions in a transparent way for the user.

### 5.2 Scenario 2

The scenario depicted in Figure 5.2 illustrates the behaviour of the system in case of message loss. This scenario is made up of five nodes. Their scores are the following: Node 1, 0.87; Node 2, 0.6; Node 3, 0.71; Node 4, 0.5; Node 5, 0.51.
Fig. 5.1. A hypothetical scenario within a work team case study.

A) Initially, the Nodes are in the “Local Mode” state, at the beginning of the election process.
B) All the nodes evaluate themselves and broadcast their scores to their neighbours.
C) However, the message with the score of Node 1 destined for Node 5 is lost.
D) When Nodes 1, 2, 3 and 4 receive the score of their neighbours they select the best node. Node 1 is the best ranked and thus it transits into the “Server Mode” state directly. Nodes 2, 3, and 4 send a SERVER_REQUEST to Node 1, which will respond with a SERVER_ACCEPTANCE message to each of them. When Nodes 2, 3 and 4 receive the SERVER_ACCEPTANCE message they transit into the “Client Mode” state. However, Node 5 has not received all the scores, and when the timer expires, it selects Node 3 as the best node and sends it a SERVER_REQUEST.
E) Node 3 receives this request, and as it is in the “Client Mode” state, because there is another node in the group with better capabilities, it will reject the request. In the SERVER_REJECTION message Node 3 will add information about its server, Node 1. This information is the node id (id: 1) and its score (0.87).
F) Node 5 will receive the SERVER_REJECTION message and the additional information. With this information the Node can complete its knowledge of the group, and now it can select the best node of the group. Therefore, Node 5 will send the SERVER_REQUEST message to Node 1, which will respond with a SERVER_ACCEPTANCE message. Finally, Node 1 will be the server of the group and the rest of the Nodes are its clients.

In this way it is highlighted how the proposal avoid that transmissions errors, which could be frequent in mobile environments, affect to the proper operation of the system and therefore to the availability of the service.

5.3 Scenario 3
The scenario depicted in Figure 5.3 illustrates how the system configures the activation of the services according to its computational requirements. In this example there are two services: Service A and Service B. Both specify their requirements through an XML file as follows [23]:

```xml
<service name="ServiceA" serviceID="0001">
  <requirements>
    <computingResources>
```

Fig. 5.2. A scenario where the architecture addresses the loss of a score message.
Fig. 5.3. A scenario where there are two different services with different computational requirements and the architecture configures the activation of their replicas according to those requirements and the computational features provided by the nodes.

\[
\text{Ev}(dC, rB) = W_1 \frac{dC}{N} + W_2 \times rB, \forall \sum_{i=1}^{n} W_i = 1 \quad (1)
\]

Where \(dC\) is the number of the direct connections of the node, which is obtained from the information provided by the routing table of the node, normalized with \(N\), the number of the nodes of the group, in order to obtain a float value between 0 and 1. The variable \(rB\) is the remaining battery of the node represented as a float value between 0 and 1. The \(W_1\) and \(W_2\) constants are the weights for each context feature.

From this default evaluation function, the specific of each service is obtained as an interval function, where the default evaluation function is multiplied by a weight (\(H_i\) and \(P_i\)) depending on the range of the computational requirements that the candidate node provides.

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Node 2</th>
<th>Node 3</th>
<th>Node 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery</td>
<td>71.50%</td>
<td>63.3%</td>
<td>31.2%</td>
</tr>
<tr>
<td>Direct Connections</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>CPU</td>
<td>1.4 GHz</td>
<td>1.2 GHz</td>
<td>1.2 GHz</td>
</tr>
<tr>
<td>Storage</td>
<td>1 Gb</td>
<td>2 Gb</td>
<td>4 Gb</td>
</tr>
<tr>
<td>(1) (Ev(dC, rB))</td>
<td>0.59</td>
<td>0.55</td>
<td>0.42</td>
</tr>
<tr>
<td>(2) (Ev_A(dC, rB, C))</td>
<td>0.59</td>
<td>0.50</td>
<td>0.38</td>
</tr>
<tr>
<td>(3) (Ev_B(dC, rB, Stg))</td>
<td>0.29</td>
<td>0.50</td>
<td>0.42</td>
</tr>
</tbody>
</table>
C) Node 1 and Node 4 will transit respectively to be the hosts for Service A and Service B. It should be noted that Node 3 in other conditions would be a better candidate for Service B, because of its storage capability. However, in this case the battery level of Node 3 is running low, and this causes the selection of Node 4 as host.

D) Finally, Node 1 hosts Service A, with nodes 2, 3, and 4 as clients. Node 4 hosts Service B, with nodes 1, 2, and 3 as clients. As this scenario has shown, define the computational requirements of the services helps to adapt the behaviour of the system and provide a more adequate configuration for the activation/hibernation of the replicas to the computational features of the nodes.

As this scenario has shown, define the computational requirements of the services helps to adapt the behaviour of the system and provide a more adequate configuration for the activation/hibernation of the replicas to the computational features of the nodes.

6 Simulation and Evaluation

The proposed architecture has been simulated and evaluated using the ns-3 network simulator. The simulated scenario aims to approximate the situation of a work team, as presented in the scenario 1 (Section 5.1), where different users are moving around of a wide scenario, sharing information through a mobile cloud service. It consists of 8 mobile nodes with a random walk mobility model. The speed of the nodes varies between 0.5-2 m/s, and they have pauses with a duration that varies between 60-300 seconds. The nodes have an IEEE 802.11 wireless connection, with a range of 250 m and a bandwidth of 1 Mbps. The mobility area is 1000 m², and they are introduced in a random initial position. The time of the simulated execution is one hour. Each of the nodes has a Replica Manager Service, and a replica of the service.

As described above, the architecture makes use of the information provided by the routing protocol to approximate the network topology. In this implementation, the OLSR (“Optimized Link State Routing”) routing protocol has been used [37]. Moreover, the different replicas of the Replica Manager Service communicate with each other by using the UDP protocol. In order to know the efficiency of the proposed solution, an approximation of the battery consumption of the service has been implemented. Therefore, in this implementation a node will consume battery by the wireless communication [38] and an additional penalty if it hosts a replica of the service active. The initial battery of the nodes is set randomly with a charge between 1800 and 2300 mAh, to provide a heterogeneous scenario with regard to the resources of the mobile network and to know the behaviour of the proposal in this situation.

Three main measures have been taken to evaluate the proposed architecture: the average availability of the service, the average battery consumed and the average coordination messages sent and received. Additionally, in order to eliminate the influence of any possible random factor, the simulation has been performed 1000 times with 1000 different initial seeds. An overview of the results obtained is presented in Table 2.

As Table 2 shows, the proposed architecture provides on average a service availability of the 95.93% of the execution time, i.e., the service is available on average 3453.48 seconds in one hour of execution. The battery consumption on average is 895.43 mAh, and this highlights that the standard deviation of the final battery of the nodes is lower than the initial. This measure shows that the battery levels of the nodes are more similar at the end of the simulation than they were initially. In other words, the proposed architecture distributes the execution of the service among the nodes of the network, making a balanced consumption of the available resources of the nodes.

Furthermore, in order to compare the benefits of this architecture, a simulation of a static deployment of the service has been carried out. In this simulation, the scenario has the same configuration parameters (mobility, battery, space, etc.). However, a number of prefixed mobile nodes will act as static nodes, i.e., there is no self-configuration approach to the activation/hibernation of the service replicas. The results of the different simulations are presented in Table 3. The simulations have been performed with 1, 2, 3, 4 and 5 static servers of 8 total nodes in the network, as the availability of a service must improve as increases the number of replicas.

As shown in Table 3 and the chart of Figure 6.1, the availability of the service increments with each addition of a new static server. Although, the maximum availability provided by a static deployment, 78.97% with 4 servers nodes (i.e., 50% of the nodes of the network in this case are servers), is far off the 95.93% (Table 2) provided by the context-aware architecture. Additionally, the execution of the service with a static deployment depletes the battery of the host nodes (Figure 6.2), i.e., with a static deployment the host nodes always deplete their battery, while in the self-configuration approach no node deplete its battery. Note that the battery consumption is an approximation to use as a guide to measure the efficiency of the proposal. In a real scenario the battery consumption depends on of
TABLE 2
Overview of the results obtained from the evaluation in the ns-3 network simulator of the proposed context-aware architecture. Each simulation measures the availability of the service in an execution of the architecture during one hour.

<table>
<thead>
<tr>
<th>Service Availability (%)</th>
<th>Initial Battery (mAh)</th>
<th>Final Battery (mAh)</th>
<th>Battery Consumed (mAh)</th>
<th>Coordination Messages (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>±σ</td>
<td>Average</td>
<td>±σ</td>
</tr>
<tr>
<td>Average</td>
<td>95.93</td>
<td>1901.12</td>
<td>223.37</td>
<td>1005.69</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>0.89</td>
<td>83.43</td>
<td>96.67</td>
<td>-</td>
</tr>
<tr>
<td>Max</td>
<td>91.44</td>
<td>2143.62</td>
<td>-</td>
<td>1323.09</td>
</tr>
<tr>
<td>Min</td>
<td>97.09</td>
<td>1606.75</td>
<td>-</td>
<td>673.10</td>
</tr>
</tbody>
</table>

FIG. 6.2. Comparison of the battery consumption of the proposed architecture in the case of a static deployment of the replicas of the service.

different variables (service requests, service failures, service synchronization, etc.).

As shown Figure 6.3 the efficiency provided by the proposed architecture (Service Availability / Battery Consumed) is considerably higher than that provided by the statics schemes. This can be understood as the context-aware architecture configuring the system to activate the minimum number of replicas and providing the maximum availability according to the current state of the network (i.e., network partitions).

Fig. 6.3. Comparison of the efficiency (Service Availability / Battery Consumed) provided by the proposed architecture in the case of a static deployment of the replicas of the service.

7 DISCUSSION

The current proposal is designed to support collaborative activities, as presented in Section 5: medium or small size work teams, in a delimited area, where the members usually move on foot, and in certain moments they stop while they are performing some common tasks. The architecture follows a copy-primary scheme [19] in physical partitions. However, in large scale networks this replication scheme is inefficient, since in large groups the active replica could easily result in a bottleneck, and therefore, a physical partition would need more than one active service replica. Creating logical partitions provides scalability to system and it facilitates the management of a large scale network. This can be done by applying clustering techniques, which will be transparent and independent for the proposal, as the election algorithm is applicable both to physical network partitions and to manage logical partitions. Moreover, in highly dynamic networks, constant changes of the network can make difficult to detect the connection and disconnection of the nodes, and therefore to establish a stable client-server connection. In such case, if it not feasible to create stable node groups, owing to a high dynamism, providing complex services will be not possible and it would need other architectural approaches, for example peer-to-peer.
The architecture follows a hibernation-based deployment approach. For instance, in collaborative systems providing support for tasks in work groups, such as firefighters or rescue and emergency teams, the set of the services is well-known and this makes possible to deploy the replicas of the service before execution. On the one hand, this approach reduces the flexibility of the system, as it is not possible to deploy a new service or introduce new devices at run-time. On the other hand, it reduces the requirements for bandwidth and can improve the response time of the configuration process. This is particularly interesting, since the MANETs currently have important technology restrictions, as for example a reduced bandwidth. However, these are technological issues that could be resolved in the future, and thus, a hybrid approach would be a better solution in terms of pulling together the flexibility of a dynamic replication approach and the response time of a hibernation approach. In this way, the well-known set of devices could be deployed before run-time and also it would be possible to add and deploy new services at run-time. This is a technical question, since the current implementation of the election algorithm is transferable between both approaches, and the code mobility techniques [5], [39] required to provide the replication and deployment of services at run-time can be implemented in a transparent way for the current election process.

Moreover, the self-configuration approach for the activation/hibernation of the replicas of the service has been addressed from a distributed approach, which provides robustness to the system against node disconnections or failures, as no node is indispensable. Additionally, the provision of an explicit context model, and the tools to adapt it to the particular requirements of the particular services, helps to adapt the behaviour of the system, and thus to provide a more effective solution in terms of efficiency in several scenarios. In this respect, the component-design of the Replica Manager Service allows the adaptation of its behaviour even at run-time through the exchange of some of its components.

Regarding the monitoring of the network topology, different approaches can be considered. Nevertheless, the majority of them are based on the position of the nodes and their travel speeds. This requires the use of the GPS system and the constant exchange of information. Consequently, such approaches have a heavy impact on the energy of the node. The proposed architecture uses the information provided by the routing protocol to estimate the network topology as an alternative to GPS-based architectures. However, this approach requires the routing tables to be periodically updated according the network topology changes, i.e., it is required a proactive routing protocol [35]. The simulation has highlighted the feasibility of this approach and its benefits.

8 Conclusions

In this paper, a context-aware software architecture has been presented. The proposed architecture provides a reusable and adaptable base to support the availability of services in mobile clouds. Complementing SOA with service replication techniques together with a self-configuration approach for the activation/hibernation of those replicas, allows us to address the availability problem successfully. This has been achieved through the design and implementation of an election algorithm. By addressing the dynamicity of the mobile system we can take advantage of its flexibility without having a negative impact on the availability of the deployed services, and therefore on the dependability of the system. This is of special interest in case of disasters in remote areas where usually the common network infrastructures are not available, and it is necessary to support the collaboration between the different work teams through data storage and processing services in the mobile clouds. With regard to the implementation and simulation of the context-aware architecture, it has demonstrated the feasibility of the proposal and the benefits of its implementation when addressing the availability of services in the mobile clouds.

The next step towards improvement of the proposal includes the extension of the architecture with clustering techniques in order to improve the scalability, and with code mobility, in order to provide a dynamic replication and deployment of services at run-time, and thus obtain a more flexible solution. Moreover, the current proposal has not addressed the synchronization of replicas of the service [40]. To this regard, it is working in a synchronization service [21], which is expected to be included in the current architecture to provide a more complete solution. Finally, a more comprehensive and in-depth study of the behaviour of the proposal in the network simulator will be carried out. In this way, a study with different routing protocols, mobility models and travel speeds, communication technologies, and hybrid networks (i.e., including static and mobile nodes), will help to learn about the proposal behaviour and the need for other possible improvements.

Acknowledgments

This research work is funded by the Spanish Ministry of Economy and Competitiveness with European Regional Development Funds (FEDER) through the project ref. TIN2012-38600, the Innovation Office from the Regional Andalusia Government through the project ref. P10-TIC-6600, and the scholarship FPU program ref. FPU13/05520 granted by the Spanish Ministry of Education, Culture and Sports.

References

[42] M. Amoretti, M. Laghi, F. Tassoni, and F. Zanichelli, “Service migration within the cloud: Code mobility in SP2A,” in High Performance Computing and Simulation (HPCS), 2010 International Conference on, June 2010, pp. 196–202. The University of Granada. He obtained his PhD in the Software Engineering Department at the University of Granada (Spain). He is a Ph.D. student in Information and Communication Technologies. His Ph.D. is funded by the scholarship program FPU granted by the Spanish Ministry of Education, Culture and Sports. His research interest is focused on context-aware and adaptive systems for mobile and ubiquitous environments. His research interest is focused on context-aware and adaptive systems for mobile and ubiquitous environments.
Sara Balderas-Díaz is Computer Science Engineer by University of Granada. She completed a M.Sc. in Software Development at the same university. Currently, she is a Ph.D. student in Information and Communication Technologies. Her research interest is focused on context-aware and adaptive systems, especially applied to the ehealth domain.

Carlos Rodríguez-Domínguez obtained his Ph.D. in Information and Communication Technologies at the University of Granada (Spain). He is funded by an Excellence Fellowship granted by the Regional Government of Andalusia, and he is a co-founder and Chief Technology Officer at Everyware Technologies, Spin-off company of the University of Granada. His research work is focused on software development for mobile and ubiquitous systems, with a special interest in the middleware development field.