Mobile Services: Context-Aware Service Migration in Ad Hoc Networks

Oriana Riva\(^1\), Tamer Nadeem\(^2\), Cristian Borcea\(^3\), and Liviu Iftode\(^4\)

\(^1\) Helsinki Institute for Information Technology, Finland
\(^2\) Department of Computer Science, University of Maryland, USA
\(^3\) Department of Computer Science, New Jersey Institute of Technology, USA
\(^4\) Department of Computer Science, Rutgers University, USA

oriana.riva@hiit.fi, nadeem@cs.umd.edu, borcea@cs.njit.edu, iftode@cs.rutgers.edu

Abstract

Short-range wireless technology is on its way of becoming ubiquitous, and it will soon be possible to program real world ad hoc networks. Deploying services in such networks, however, is constrained by the lack of proper service models and system support. This paper presents mobile services, a context-aware service model that addresses the highly dynamic configurations and rapidly changing contexts encountered in ad hoc networks. Unlike a statically identified service that is always located on the same node, a mobile service can migrate to different nodes in the network to accomplish its task. The service migration occurs transparently to the client application, which is presented with a unique virtual service end-point.

We designed and implemented a middleware for developing mobile services. We built a proof-of-concept mobile service that dynamically computes the average vehicular speed in a region ahead of a given car by using only car-to-car short-range wireless communication. The experimental results obtained over a mobile ad hoc network of PDAs show the effectiveness of our approach in the presence of frequent disconnections. We also present simulation results that demonstrate the benefits of mobile services in large scale networks compared to statically centralized approaches.

1. Introduction

The critical mass of wireless devices that allows us to program real world ad hoc networks will soon be reached. Such networks are formed either spontaneously (e.g., vehicles on the road) or deployed for specific tasks (e.g., monitoring a certain region in a city during an emergency situation). Ad hoc networks can be used to provide a large spectrum of services. For instance, a mobile ad hoc network of vehicles can provide traffic information in a region five miles ahead of a given car on a highway, while an ad hoc network of intelligent video cameras can transmit images from the proximity of a disaster area. There are two main benefits of developing services over ad hoc networks. First, these services can provide users on the move with information that they could not acquire otherwise, in the absence of an infrastructure. Second, turning ad hoc networks into service providers have the benefit of acquiring real-time information from nodes located in the immediate proximity of geographical regions, objects, or activities of interest. Deploying services in such networks, however, is constrained by the lack of proper service models and system support.

Traditional service models assume the service located on a statically identified node and the client communicating with such a node for the entire duration of the interaction with the service. For instance, web services in the Internet execute on nodes statically identified by IP addresses and are connection-oriented. Such models can hardly support the deployment of services over highly dynamic ad hoc networks. The ad hoc network scenario consists of nodes supporting services that dynamically join and leave the network, thus continuously modifying the set of available services in the region of interaction. Additionally, due to limited resource availability and unpredictable loss of network connectivity, a node may stop providing a service currently in use, and alternatively new services may appear.

Beside the dynamic availability of the nodes offering services, a service provider may stop guaranteeing the required quality of service as result of a change in the service context or client context. For example, a service monitoring a certain region might run on a node that moves farther away from that region. In this case, the change in the service context leads to a decrease in the quality of service or even to the impossibility of providing the service. Another example involves an object tracking service; the service provides the required quality as long as it runs on a node close to the object, but as the object moves, the quality of service decreases. Changes in the client context can also render the service inadequate. In
such situations, the traditional solution for client-service interactions is to interrupt the current interaction and let the client find another service that can serve the request. Such an approach is highly inefficient. A novel solution could be to make services context-aware such that they can adapt to context changes by migrating their execution to nodes where they can accomplish their task better.

This paper presents a novel model of service interaction in ad hoc networks based on the concept of mobile services. Unlike a statically identified service entity which is always located on the same node, a mobile service is capable of migrating to different nodes in the network in order to effectively accomplish its function and guarantee the desired quality of service. Although a service end-point can migrate, it constantly appears to the client application as a unique virtual end-point: the service migration is transparent to clients. Furthermore, the service migration is context-aware since it is triggered by a change in the context of either the client application or the service. The service is designed such that it is able to learn about context changes and migrate to other nodes in the networks as soon as the current hosting node becomes unsuitable. Additionally, services carry any relevant state information during migrations. Therefore, the interaction between a user and a service can continue uninterrupted, except for small delays generated by migrations.

We designed a middleware that supports the development and execution of mobile services. This middleware provides context-based message passing primitives, mobile service control, and context management. The system support for this middleware includes content-based routing and execution migration. We implemented the mobile service middleware on top of Smart Messages, a distributed computing platform for ad hoc networks based on execution migration. Furthermore, we built a proof-of-concept mobile service that dynamically computes the average vehicular speed in a region ahead of a given car using only car-to-car short-range wireless communication. Experimental results for this mobile service executed over mobile ad hoc networks of PDAs show the feasibility and effectiveness of our approach. We also simulated the same service to analyze its behavior in large scale networks and compare it to a traditional centralized and static implementation.

The rest of this paper is organized as follows. In Section 2, we introduce the mobile service model built on the concept of context-aware service migration. Section 3 presents the mobile service framework that supports the deployment of mobile services in ad hoc networks. Section 4 describes the design and implementation of our mobile service prototype. Experimental results over ad hoc networks of PDAs and simulations results for larger scale networks are presented in Section 5. The related work is discussed in Section 6. The paper concludes in Section 7.

2. Mobile Services over Ad Hoc Networks

Providing services over ad hoc networks is especially beneficial for applications that need to monitor the physical world. Such services must be capable of acquiring, in real-time, information from nodes located in the immediate proximity of the monitored geographical region, object, or activity. These nodes dynamically create ad hoc networks that may be exploited to support the required exchange of information. Unfortunately, traditional models for service interaction can hardly suit such highly dynamic ad hoc environment. Indeed, the common assumption behind these paradigms sees the service located on a statically identified node and the client communicating with such a node for the entire duration of the interaction with the service.

Essentially, the deployment of services over ad hoc networks is constrained by the lack of a stable network configuration. End-to-end connections are volatile, especially when the end-points are several hops away. Frequent disconnections occur either accidently due to loss of wireless connectivity of users, service providers, and intermediate nodes or voluntarily to save battery or reduce connection costs. The overall result is that both the nodes providing services and the intermediate nodes between a client and a service dynamically join and leave the network in an unpredictable manner. Additionally, the node capabilities (hardware and software) vary widely across the nodes in an ad hoc network, and they change over time.

To cope with such uncertainty, there is a need to design more agile services (and interaction models between clients and services) that are able to adapt to changes in the network or, broadly speaking, to changes in both the client context and service context. Context includes parameters such as location, speed, time, device capabilities, or network topology. In order to accomplish a context-aware interaction between clients and services, we envisage three elements to be combined.

The service has to constantly monitor its context to validate its ability to provide the desired service quality. Unlike traditional networking where QoS refers strictly to network parameters such as throughput or delay, services deployed over ad hoc networks have to consider the whole execution context when evaluating whether they can guarantee a certain quality (i.e., this is an application-level service quality). For instance, if the client wishes to receive information about events taking place in a certain geographical region (e.g., intrusion detection), the service needs to continuously verify that the hosting node is still located in the region of interest. If not, the traditional solution is to interrupt the interac-

---

1 The term mobile service has been historically used to refer to services provided to mobile users using wireless networking. In our case, the services themselves are mobile as they move their execution from node to node.
tion and let the client find another service that can provide the desired service quality. A solution that solves this problem is to allow the current service to migrate to other nodes where it can provide the desired service quality.

The client application, at its turn, must provide a dynamic representation of the user’s context, and the service must be able to retrieve and process those changes in real-time in order to provide highly adaptive delivery of services. Specifically, user context changes must trigger the adaptation of the requirements of the client application and the new parameters have to be communicated to the service to allow it to adapt.

The last required element is a mechanism that supports a context-aware interaction between the two parties. Indeed, context changes not only trigger the adaptation of request parameters and updating of service parameters, but they may cause the migration of the service to other physical nodes. The challenge, however, is how to guarantee a continuous and semantically correct interaction between the client and the service.

To address these requirements, we propose a novel model of service interaction in ad hoc networks based on the concept of mobile service. Unlike a statically identified service entity which is always located on the same node, a mobile service is capable of migrating to different nodes in the network in order to effectively accomplish its function. This migration occurs transparently to the client and, except for a certain delay, no service interruption is perceived by the client. With this model, the clients do not need to discover new services when the old ones are not suitable any longer. Although a mobile service is physically located on different nodes over time, it presents a unique virtual end-point to the client.

In this model, there is a one-to-one mapping between mobile services and client applications. Once a service that has the potential resources to serve the request is discovered, this service will instantiate a mobile service that migrates through the network to provide the client with the desired answers (note that the initial service just spawns mobile services, but it does not send responses to clients). The service migration is context-aware since it is triggered by a change in the context of either the service or the client application. In our model, the nodes hosting the client application can be mobile, but we assume that the client application does not change the node of execution. Example of applications that benefit from mobile services include monitoring certain geographical regions (i.e., the service always execute on a node located in the desired region despite node mobility) and object tracking (i.e., the service follows the motion path of the tracked object moving from node to node).

Figure 1 illustrates an example of interaction between a client and a monitoring mobile service that periodically reports observations to the client. The service changes the node of execution by migrating from node $n_2$ to $n_3$, and subsequently from $n_3$ to $n_4$. During the entire period, the client application is not aware of the service migration on different nodes (i.e., it sees a virtual service end-point) and the service provisioning is not affected by any interruption. Although not shown in the figure, we assume that any message between two nodes travels across multiple intermediate nodes.

Figure 2 details the same example of interaction between a client and a mobile service. The client application periodically monitors the user context parameters and when context changes, updates are triggered and sent to the service side. For example, it may be the case of a user that needs to monitor a certain geographical region and requests to receive more accurate observations as he gets closer to region. In the figure, we see that after the service receives the update, it migrates to Node3 (e.g., this node is closer to the region of interest).
Similarly, the model assumes that the service constantly monitors its context parameters and checks whether it can still provide the desired service quality. For instance, the second migration depicted in the figure is triggered by a change in the service context. This can be as simple as “low battery” or “current node is too far away from the monitored region”. The example ends with a new update sent by the client to the current location of the service (i.e., service migration is transparent for the client).

Note that certain client context changes may be inferred without the need for the client end-point to explicitly send an update message when the context changes. Indeed, for certain types of applications, it may be feasible that the service predicts client context changes based on history and autonomously triggers adaptations.

### 3. Mobile Services Framework

This section describes the overall framework that supports the deployment of mobile services in ad hoc networks. The framework also supports the specification of client applications that interact with mobile services. Since nodes in ad hoc networks can act as both clients and service providers, we designed a common software architecture that has to be offered by all nodes willing to cooperate. In the following, we present such software architecture and the implementation we carried out by using the Smart Messages platform [5, 12].

#### 3.1. Mobile Services Software Architecture

The mobile services framework is designed according to the software architecture shown in Figure 3. The architecture consists of a middleware layer and a system layer. The middleware layer provides the functionality needed to develop and execute mobile services and client applications. This middleware is common for all the nodes and shields the programmers from the hardware and operating system heterogeneity encountered in ad hoc networks. The system layer provides the support needed by the middleware with basic functionality that may be implemented differently on each node.

The middleware includes mechanisms for monitoring and updating context data, a mobile service controller, and a communication manager. The **Context Manager** is in charge of supporting the provisioning of context data and dynamically updating their values. The updating is proactive when it occurs by periodically polling the **Context Monitor** (in case the context information is local to the node) or by querying the **Context Predictor** (in case the context information is remote). The updating is reactive when triggered by the reception of an external update message containing the new context parameters.

The **Mobile Service Controller** continuously verifies whether the mobile service is still capable of correctly processing the client query and guaranteeing the promised service quality. If it deems the node hosting the service unsuitable, it triggers a service migration by sending a request to the **Migration Manager** at the system layer. Additionally, the **Mobile Service Controller** is in charge of performing control management and user authentication. Finally, the **Communication Manager** manages the communication between clients and services (i.e., it is responsible for the exchange of request, response, and update messages).

The system layer provides mechanisms to address most common issues in context provisioning, support for execution migration, content-based routing, and neighbor discovery in wireless environments. Among the system components, the **Migration Manager** plays the fundamental role of supporting the service migration. Its primary task is to perform the marshalling/un-marshalling operations during service migrations. When it is called by the **Mobile Service Controller**, it extracts the execution state of the service, and then marshals it along with the code and data. The reason behind the need to transfer the code is that we cannot assume that all the nodes hold the code for any service implementation. Once the marshalling is done, the **Migration Manager** invokes the **Content-Based Routing** module to send the marshalled service to a suitable node. The **Migration Manager** at destination un-marshals the service and resumes its execution.

The multi-hop communication between nodes takes place through the **Content-Based Routing** module. This module is not only invoked during migrations, but also during regular data exchanges between clients and services. Service discovery and routing are integrated in a single module that performs content-based routing. This decision was determined by the necessity to achieve a more flexible naming scheme in ad hoc networks. Fixed bindings between names and node addresses assumed in the Internet are not suitable for our model. After a fixed binding has been established during the
name resolution phase, an application is required to contact
the same node each time it needs to access the mobile ser-
vice. This naming is too rigid for our model because we need
to name the service end-point, not the node hosting the ser-
vice. Therefore, we decided to employ content-based naming
for both the service and client end-points.

The Neighbor Discovery component maintains and peri-
odically updates a list of the one-hop neighbor nodes. For in-
stance, this information is required by geographical routing
(which is used by the routing module to locate nodes in cer-
tain geographical regions). Finally, the Content Provider is in
charge of obtaining context data by either directly employing
sensors on the device or by means of external infrastruc-
tures.

3.2. Programming mobile services and clients

Content-Based Message Passing API. For the mobile
service model to become successful, we need to provide a
simple and intuitive way of programming the corresponding
client applications and services. We believe that program-
ners are well versed with the message passing distributed
programming model. Therefore, the mobile services middle-
ware provides an API that allows client applications to send
requests and receive responses, while the services can re-
ceive requests and send responses. The only difference be-
tween these primitives and their message passing counter-
parts is that the sources and destinations are named by con-
tent as described in Section 3.1. The entire process of service
migration is transparent to applications and services them-
selves.

Client/Service Registration. Both client applica-
tions and services have to register with the middleware.

Client applications and services have to register with the
middleware. The main goal of this operation is to clearly identify enti-
ties and relative context parameters to be monitored by the
Context Manager.

Specifying Validation Policies. On the client side, the ap-
plication programmer has to define a validation policy in or-
der to enable the middleware to validate the received results
and pass to the application only policy-respondent results.
Specifically, the client application interface has to provide a
validation function to be called by the middleware layer
each time a response for this application is received. When
this function is evaluated, the Context Manager is queried
in order to update the context parameters of the application.
If the validation is successful, the result is delivered. Other-
wise, the middleware sends an updated client request to the
mobile service. Optionally, at registration time the client may
instruct the middleware to send periodic updates.

On the service side, the service has to define an analogue
validation function to allow the Mobile Service Controller to
evaluate the provided service quality according to the cur-
rent service context and possible request updates received
from the client. This function is evaluated before sending a
new response (i.e., it is possible that the newly computed re-
sponse does not satisfy the service quality for the current
context). If the service quality is no longer satisfactory, the
Mobile Service Controller initiates a migration with the Mi-
gration Manager at the system layer.

To simplify programmers’ task and enforce them to de-
fine validation policies, we defined abstract clients and mo-
bile services that can be extended in order to implement ap-
lications and services. They perform the registration with
the middleware and declare abstract functions for validation.
These functions must be defined by the clients and services,
respectively, on the basis of their semantic information and
the type of service to be supported.

3.3. Implementation using Smart Messages

We implemented the mobile services middleware on top
of Smart Messages (SMs) [5, 12] as depicted in Figure 4.
An SM is a user-defined application whose execution is dis-
tributed over a series of nodes using execution migration.
The nodes on which SMs execute, called nodes of inter-
est, are named by properties and discovered dynamically
using application controlled routing. To move between two
nodes of interest, an SM calls explicitly for execution mi-
gration. An SM consists of code bricks, data bricks (mobile
data explicitly identified in the program), and execution con-
trol state (e.g., instruction pointers, operand stack pointers).

Code bricks are transferred with SMs only if the code is not
cached at the destination. An SM can use its code and data
bricks to create new, possibly smaller SMs during execution.
In this way, an application can eventually generate multiple
SMs although it started as a single SM.

The nodes cooperate to support the SM execution by pro-
viding a virtual machine (VM) for execution over hetero-
genous platforms, a shared memory addressable by names
(tag space) for inter-SM communication and synchroniza-

Figure 4. Software Architecture Implementa-
tion
Neighbor Discovery

and using 802.11 cards for wireless communication.

Virtual Machine (KVM). The tag space and SM operations are available to applications through a Java API. Our prototype implementation was tested on HP iPAQs running Linux.

Corresponding to their functionality, there are two types of tags: application tags for “persistent” memory across SM executions (i.e., they can store application-specific data for a limited period of time), and I/O tags for interaction with the host’s operating system and I/O. The collection of all tags available at a node forms the tag space. Each tag has a name that is employed in content-based naming. In our implementation, the application tags are used for data exchange and synchronization between clients and mobile services. Some I/O tags are used for accessing context information (e.g., location, speed); thus, they implement the interface of the Context Provider defined by our software architecture. Furthermore, the SM platform has an I/O tag that implements the Neighbor Discovery module. The tags are used to uniquely name both end-points of the communication, as well as to define the destination node of a service migration.

SMs use a high level migration function to migrate to nodes of interest named by tags. The routing is executed at each node toward a node of interest (i.e., SMs are self-routing applications). The implementation of routing uses information stored by SMs in the tag space and a system-provided primitive for one hop migration. This primitive captures the current execution control state and migrates it to the next hop along with the code and data bricks. A very important feature of the self-routing mechanism is the ability of SMs to use multiple routing algorithms during their lifetime and change these algorithms dynamically. The current implementation of mobile services provides two basic SM routing algorithms for the Content-Based Routing module defined in our architecture: geographical routing, and region-bound content-based routing. The geographical routing is similar to GPSR [14]. At each node, the algorithm migrates the SM to the neighbor closest to the location of interest. The content-based on-demand routing (similar to AODV [17]) is used to discover a node of interest within a given geographical region (reached using the geographical routing). Due to its limited geographical scope, flooding does not represent a major problem for the scalability of this algorithm.

We implemented an SM prototype by modifying Sun’s K Virtual Machine (KVM). The tag space and SM operations are available to applications through a Java API. Our prototype implementation was tested on HP iPAQs running Linux and using 802.11 cards for wireless communication.

3.4. Limitations

Although it covers a large spectrum of potential services for ad hoc networks (especially those for monitoring the physical world), the current design of our model does not support certain types of services. The interaction between mobile services and applications is not entirely reliable as messages can be lost or delivered out of order. The reason is that we targeted applications that send one request and receive multiple responses (either in a time-driven or in an event-driven manner), which are capable to deal with such situations. We plan to extend the mobile services model to provide reliability and eventually support stateful interactions, which assume reliable delivery of every message.

Other issues have to be addressed to achieve fully deployment of services in ad hoc networks. Authentication, encryption, and access control are security mechanisms that can be hardly introduced in ad hoc networks mostly due to the absence of a main trusted server. A possible solution could be the definition of a security and privacy profile at each node, which defines the conditions for allowing or denying service migration. Social acceptance, privacy, and trust will also be significant problems faced by services deployed in ad hoc networks. Many people will not allow the access and sharing of private context information unless their identity is hidden (i.e., anonymity of the users). However, we expect that as services will start to appear in such networks, people will have incentives to cooperate and will be less concerned about privacy and trust.

Our current migration implementation drives a service to a node that provides all the tags of interest, but it does not check the values of all those tags. If some tag values make the service quality unacceptable, the middleware re-starts the migration process. This procedure can be ameliorated by using the content-based routing capable of selecting only those nodes that provide all the necessary tag names along with suitable tag values to guarantee the quality of result.

4. Case Study: Speedy

To demonstrate the appropriateness of mobile services, we have designed and developed Speedy, a mobile service prototype. Speedy provides drivers with information about vehicular speed in a driver-defined region on a road. For instance, a driver may want to know the average speed of the vehicular traffic in a region located between 15 and 20 miles ahead from her position. We assume that all cars are equipped with embedded computers that run our middleware for mobile services. The cars communicate using short-range wireless networking (e.g., IEEE 802.11), and they have GPS receivers that report their current location. An ad hoc network of cars performs the distributed computing necessary
to calculate the average vehicular speed in the desired region.

At the beginning of the interaction, the client application provides the service with the user location, its current speed, and the coordinates of the region of interest. The service periodically re-calculates the region of interest according to the location and speed of the user with the aim to ideally predict the driver’s progress on the highway. Nevertheless, at any time the middleware at the client node can decide to re-calibrate the service estimation by sending an update of the request parameters.

An example of two instants of interaction between the client application and an instantiation of Speedy is depicted in Figure 5. Every car has an associated context that includes its current location and speed. At time $t$, the node $n1$ hosts the user $u$, which has an associated context $uc1$. The client application accesses the service $s$, located on node $n3$ and associated with context $sc1$. Since our implementation of the middleware for mobile services is based on Smart Messages, Speedy migrates to node $n4$ and $n6$ to carry out the computation (i.e., showed with dotted arrows in the figure). Note that these migrations are just byproducts of our implementation, not the context-aware migration specific to mobile services. Assuming that three speed measurements are enough to calculate a reliable average, the service execution completes on $n6$, and the response is sent to $n1$.

After $\Delta t$, the position and potentially the speed of nodes have changed. Therefore, the space of interaction is subject to a new computing environment. The new region of interest is either predicted by the service based on the request parameters and the user context history or computed based on an update received from the client application that respect the new user context, i.e., $sc2$ in the figure). At this time, the service realizes that the node on which is running (i.e., $n6$) is out of the desired region. Speedy uses the context-aware migration specific to mobile services to reach a node in the new region of interest (depicted with a continuous arrow in the figure). Once arrived at node $n4$, it will migrate to nodes $n8$ and $n7$ to compute a new average speed. These nodes globally constitute the new service instance with context $sc2$ associated.

### 5. Evaluation

This section presents experimental and simulation results for Speedy, our prototype mobile service. We ran experiments in a testbed modelling a mobile ad-hoc network. The goal of the testbed study was to prove the feasibility and effectiveness of our model in mobile ad hoc networks. Given the rapid changes in the configuration and user location, it is rather relevant to evaluate how the application can adapt to such changes and recover from disconnections.

We also simulated the prototype service to analyze its performance in larger scale networks and compare it to a traditional centralized approach. We simulated two different versions of the Speedy application: Speedy-Smart and Speedy-Base. Speedy-Smart implements our model of context-aware migration. Speedy-Base represents the baseline centralized approach.

In Speedy-Base, instead of migrating the service from node to node based on the user requirements, a few mobile nodes host the service in the network. The client sends the request to one of those nodes and receives results only from that node. In the Speedy-Base mechanism, a service node processes the client request, calculates the average vehicular speed by directly querying nodes in the region of interest, sends back the results to the client, automatically updates the query parameters, and finally initiates a new query cycle. In the next subsection, we will describe our prototype implementation of Speedy-Smart mechanism only as a proof for the concept of service migration. Later in the simulation results subsection, we compare the performance of Speedy-Smart and Speedy-Base.

#### 5.1. Experimental Results

The ad hoc wireless network topology consists of 11 HP iPAQs (the initial topology is shown Figure 6). The iPAQs run Linux and use Orinoco’s 802.11b PC cards for wireless communication. Since it has proved very difficult to run the experiments with the iPAQs moving at an adequate speed, we have emulated the mobility by instructing each node to periodically update its position and speed with data from a local file. Specifically, each file contains the location coordinates (i.e., latitude and longitude) and the speed of the node. Those values are read every 5s and the speed is a uniform variable between 5-10m/s. Almost all responses in our ex-
Experiments were jointly calculated by a maximum of 2 nodes at 2-hop distance from the user. The service code was initially injected only on one service node. We ran the same experiment 20 times, and each replication includes 100 responses (i.e., correct, wrong or missing response).

In order to reduce the overhead due to the exchange of notifications of context changes, the service predicts the new user context parameters based on the past ones and also includes them in the response to the user. At the reception of such response, if the parameters are incorrect based on the current context, the answer is discarded and an update is sent to the service. Otherwise, the answer is delivered to the client application and no update is sent. This approach is feasible for Speedy since speed and direction of moving vehicles can be easily estimated on a highway. Furthermore, the user application sends an update if no answer is received within a certain timeout. In the experiments, the timeout is fixed to 7.5s.

The main metrics employed in the evaluation are:

- **inter-response time** measures the elapsed time between consecutive correct answers. It includes the time to update (either proactively or reactively) the user parameters;
- **service discovery time** measures the elapsed time to receive the first correct response;
- **service quality** is the percentage of correct answers out of the total number of received answers;
- **update rate** is the average number of updates the client application sends to get one response.

The experimental results are summarized in Table 1. We report average value and the 90% confidence interval for every metric. The reason why the inter-response time is rather high is that it includes the time to propagate and process the updates sent from the user side to re-calibrate the query parameters. However, those values are more than acceptable for this type of application. Our goal was to study how the model of mobile service allows the user to constantly receive correct observations in spite of disconnections and mobility. As shown in Figure 7, the service migration closely follows the user movement. The graph compares the movement of the user to the service migration. The service location is estimated as the average position of all vehicles participating in the speed calculation. If the service position is out of the range of interest, the answer is labelled as wrong. When disconnections occur, the user application sends update messages. One update is necessary to get almost three correct responses, thus reducing the communication overhead due to context changes. Commonly, the service quality is high as few wrong inter-responses are received.

### 5.2. Simulation Results

The Speedy-Smart and Speedy-Base services have been simulated by means of ns-2 simulator [3], enhanced with the CMU-wireless extensions [2]. Speedy-Smart implements the mobile service. Speedy-Base represents the baseline centralized approach. Different scenarios were considered to test the algorithms. Hereafter, we use the terms “vehicle” and “node”, and the terms “answer” and “result” interchangeably.
5.2.1. Vehicular Traffic Generator

Modelling vehicular traffic is a research topic that is receiving large attention for its potential capabilities to support the analysis of many intelligent vehicular services. For example, CORSIM [1] is a microscopic traffic simulator developed by The Federal Highway Administration in the United States. Unfortunately, none of the traffic modelling tools are freely available to public. Therefore, we have developed our own traffic generator tool based on seidest, a generator tool for random-way point mobility model in ns-2. The scenario generator accepts as parameters the simulation time, road length in meters, number of lanes per road, average speed of the vehicles in meters/sec, average gap distance between vehicles on same lane, number of service nodes on the road, and the number of users on the road. It uses a simplified traffic model as follows:

- **Entries and Exits:** The entries and exits are evenly distributed along the road each 1000 meters. Vehicles may enter the road at each entry except the last one and leave at any subsequent exit. Vehicles enter the road at the front-end entry with a probability of 0.7, and at side entries with a probability of 0.3.
- **Speed Changes:** To model speed changes at each node, the road between the entry point and exit point of a certain node is divided into regions of 50 meters, and a constant speed of \( \text{average speed} \pm (0.25 \times \text{average speed} \times \text{rand}(k)) \) is used for each region, where rand() returns a uniformly distributed random from the range \([0, 1]\).
- **Changing Lanes:** Vehicles can change their lanes independently on other vehicles. The probability of staying on the same lane is 0.6 whereas the probability of changing to the right or left lane is 0.2.
- **Vehicle Density:** The density of vehicles is an important factor because it determines the number of neighboring nodes in the transmission range of a vehicle. The traffic generator initially puts

\[
\text{active nodes, evenly distributed, on the road. Once a vehicle leaves the road at one of the exits, it is deactivated, and a new node is added (activated) to the road randomly. As soon as a node is deactivated, it will no longer affect our metric calculations introduced in the next section.}
\]

5.2.2. Metrics

To evaluate the performance of our applications we employ the **inter-response time** and the **service quality** parameters previously defined, and additionally, we consider the following metrics:

- **Correct response generation time** is the average time needed by a correct response to be computed and transmitted. It measures the elapsed time between the time at which a query process is started (either by the latest node in the Speedy-Smart or by the service node in the Speedy-Base) and time at which it is delivered to the client side. It differs from the inter-response time as it does not include the time needed for the update process or the time wasted due to lost responses. Basically, it represents the ideal case in which no responses are lost and no updates are required.
- **Accuracy** measures how close the estimated average speed (calculated by employing a subset of the available vehicles in the area of interest) is to the actual average speed in the region of interest (calculated by considering all present vehicles in the same area).
- **Utilization rates:** the node utilization rate measures the average percentage of the number of nodes participating in the answer’s computation out of the total number of nodes that received the corresponding query packet. At each periodic result calculation, a query process packet is circulated in the network. Such a packet may go through the same node several times. However, a node cannot participate in calculating the average speed more than once. Therefore, already visited nodes simply forward the packets. Each node within the query area maintains for each neighbor a counter of the number of times it forwarded to that neighbor the query process packet. A node forwards the packet to the node with the lowest counter. A node stops processing the query and forwards the current result back once the counters for all its neighbors reach a certain threshold (5 in our simulations). Similarly, the packet utilization rate measures the average percentage of the number of exchanged query process packets that were actually employed in calculating a result out of the total number of exchanged query process packets. A packet is considered to have participated in the result computation when it is received for the first time by a node in the area of interest.
- **Response packets overhead** is the average number of packets (i.e., query process, update, and result packets) that need to be exchanged for every correct result received by the user.

5.2.3. Results

Service nodes and clients are selected uniformly among the vehicles. Once a user vehicle becomes active, a client request is generated after 25 seconds warm up period. Once a service node is selected by a client, the service node initiates a query process phase. This query process phase is periodically re-initiated either by the last node in Speedy-Smart mechanism or by the service node in the Speedy-Base mechanism. We
set the period to 5 seconds. Furthermore, if a client does not receive any result after three time periods, it sends again the service request.

As wireless media we used 802.11b with a data transmission rate of 11Mb and a transmission range of 250 meters \(^2\). The received signal strength threshold used in maintaining information about neighbors is set to be corresponding to distances below 200m in order to accommodate with the fast dynamics of the networks and to maintain consistency in neighbors information.

**Effects of number of clients.** We first studied how Speedy-Smart and Speedy-Base mechanisms scale with an increasing number of clients. The scenarios consists of a highway of length 25km with three lanes. The average speed of vehicles is 30m/s with a gap of 150m \(^3\). We fixed the number of service nodes to 150 nodes randomly selected among all vehicles.

Figure 8 plots the average time needed to generate new correct responses and the inter-response time experienced by Speedy-Smart and Speedy-Base. In Speedy-Smart the time needed to compute and transmit new correct responses is almost constant, while in Base it increases as the number of clients grows. The explanation resides in the capability of the Speedy-Smart service to follow the user motion while the Base service is statically fixed on a node. In the Base mechanism, as the number of clients grows, the average distances in hops between the client node and the service node as well as between the service node and the region of interest increase, thus negatively affecting the service computing time. In the Smart mechanism those distances remain practically the same since the service is migrating according to the user movement. If the query and result packets need to traverse a longer path, the chance for them to get lost because of hitting a geographic routing hole or because of collisions increases and negatively affects the performance. Consequently, the inter-response times for the two mechanisms differ accordingly.

Service quality and accuracy metrics present similar values in both mechanisms as shown in Figure 9. On the other hand, the analysis of the utilization rates for this scenario confirms the previous observations (see Figure 10). The Speedy-Smart service provides in general higher utilization rates due to the longer paths packets need to traverse in Speedy-Base. Nevertheless, the Speedy-Smart service significantly reduces the overhead due to the number of packets transmitted per received correct response. As expected, in the Base mechanism, such overhead is especially high in correspondence of a high number of clients.

---

\(^2\) In practice, we found out that the wireless transmission range is less than 250 meters. However, using external antennas, we can restore this transmission range.

\(^3\) The total number of vehicles in such scenario configuration is 800 vehicles with only 500 vehicles active at any time.
Effects of vehicles speed. In this scenarios we investigated the effects of the average vehicles speed on the performance of both mechanisms. We fixed the number of clients to 150 and varied the average speed from 10m/s up to 30m/s. The values of the rest of the parameters are as given in the previous subsection. The time the service needs to compute and deliver a correct response is 2-3 times higher in Speedy-Base (bars in Figure 11). Furthermore, similar to what we observed by varying the number of clients, this time is rather constant in Speedy-Smart simulations, while it decreases with the increase in the average speed in Speedy-Base. As a consequence, Speedy-Smart has a lower inter-response time than the Speed-Base as shown by the lines in Figure 11. Similar to what observed by varying the number of clients, Speedy-Smart and Speedy Base demonstrate similar service quality and accuracy, which do not get affected with the change in vehicles speed. Finally, the bar graph in Figure 12 depicts the node and packet utilization rates of both mechanisms. The variation of the vehicular speed has no effect on them. As expected, Speedy-Smart provides higher node and packet utilization as well as a significantly lower response packets overhead (lines in Figure 12).

Effects of vehicle density. Next, we studied the effects of variations in the vehicles density. We fixed the number of users to 150 users, the average speed to 20m/s, and changed the average gap between each consecutive car from 100m up to 200m. The values of the rest of the parameters are as given previously. As shown in Figure 13, in both mechanisms, variations in the vehicles distance have no effect on the time the service needs to generate correct responses. However, the inter-response times observed by the client (which include times to process updates and timeouts due to lost responses) increase as the vehicle gap grows since the chance of having holes in the geographic routing increases. As expected, the latency and inter-response for Speedy-Smart is significantly lower than for Speedy-Base. Both mechanisms present similar node utilization rates as shown by the bar graph of Figure 14, while Speedy-Smart demonstrates 2 times higher packet utilization rates. Finally Speedy-Smart performs better also in terms of response packet overhead.

Effects of error model. Finally, we studied the effect of the error model in estimating the average speed. We fixed the number of users to 150 users, the average speed to 30m/s, the average gap to 150m, and we varied the percentage of result error from 5% to 25%. Figure 15 shows that neither the correct response generation time nor the inter-response time change with the error percentage for the Speedy-Base while the latency and the inter-response decrease with the increase in error percentage for Speedy-Smart. The reason is that Speedy-Base already suffers from high response generation time and inter-response time; hence, it is insensitive to the decreasing in the number of updates due to increas-
Figure 14. Node utilization rate (bars), packet utilization rate (bars), and response packets overhead (lines) versus vehicles distance

Figure 15. Correct response generation time (bars) and inter-response time (lines) versus allowable error rate

Figure 16. Node utilization rate (bars), packet utilization rate (bars), and response packets overhead (lines) versus allowable error rate

6. Related Work

In the near future, mobile users will expect to be provided with continuous access to personalized adaptive services [21]. Services become personalized when they are tailored to the context and able to adapt to context changes, changing situation. The adoption of context awareness in service provisioning usually aims at making services automatically aware of the current situation in which the user is embedded. For example in [15], the authors demonstrate how service matching based on static service description provides minimal service discovery and filtering, leaving the rest of the work to user’s manual selection. Additionally, context-awareness is important when dealing with different network QoS offerings and diverse connectivity in order to transparently and dynamically choose the most appropriate network service at any given time. In [9], the authors propose an intelligent software agent that transparently and constantly selects the most appropriate network service based on user’s profile. Unlike these approaches, we propose a service model that exploits the context information by migrating the services on different nodes in a mobile ad hoc network in order to provide the required QoS. In our model, not only the user application monitors and reacts to context changes, but the service itself is context-aware and adaptive.

The support of context-awareness in nomadic computing has been investigated in many projects. Gaia [18] introduces the notion of Active Spaces as computing environments for context-aware applications. Users move from one Active Space to another, seamlessly extracting from one space and integrating into a new one. GAIA addresses the needs of context-aware applications in small networked en-
environments where the available resources in the space can be centrally managed by a “context file system”, but do not apply to large scale context-aware applications in completely ad hoc environments that necessitate a decentralized solution.

The CORTEX project [4] has examined fundamental issues related to the support of context-aware applications and the development of a middleware in pervasive ad hoc environments. They propose a programming model for supporting the construction of context-aware applications. Applications are constructed from large numbers of software components which accept input via a variety of sensors and autonomously react by acting upon the environment using a variety of actuators. These components, namely the Sentient Objects, must have a certain level of intelligence to act autonomously based upon the gathered information, and be able to cooperate with each other using a range of different networking technologies [22]. To demonstrate the appropriateness of their approach, they used Sentient Objects to model sentient vehicles that are context-aware autonomous cars cooperating by using mobile ad hoc networks [20].

The application of context-awareness to mobile ad hoc networks has been recently investigated also in [6]. The authors propose the EgoSpaces model and demonstrate how context-awareness can be employed to abstract the data available in an ad hoc network into a data structure. EgoSpaces consists of logically mobile agents that operate over mobile nodes. The agents can specify which data have to be included in their operating context by means of declarative specifications containing properties of the data items, of the agents that own the data, of the hosts on which those agents are running, and of the attributes of the ad hoc network. The coordination model see agents interacting with a dynamically changing environment through a set of views, which are custom-defined projections of the set of data objects present in the surrounding ad hoc network.

Mobile services are similar with these approaches in the sense that they are oriented to support context-awareness in nomadic computing. Additionally, some of these approaches specifically address cooperation over mobile ad hoc networks. We differ from them in the problem we solve. Mobile services target directly the deployment of context-aware, adaptive services in ad hoc networks. The uniqueness of our solutions consists of the context-aware service migration model and the middleware that supports it.

In this context, we should mention [11], which presents a nomadic and resource aware web service framework that provides capabilities to dynamically migrate a web service to different nodes according to the available resources at nodes. We share with this work the concept of relocation to other available hosts in the ad hoc network due to new context requirements on the original host. However, this work addresses services running in relatively stable IP-based networks and is mostly similar to traditional process migration for load balancing. Our approach specifically focuses on deploying context-aware services in mobile ad hoc networks. Nevertheless, our migration strategies include a wider range of context parameters (both user’s and service’s parameters).

Few attempts have been done so far to deploy services over ad hoc networks. An original contribution has been presented in [8]. They propose a mobile context-aware narrative that allows the dynamic presentation of scenes and sequences based on the interaction of the user with the story itself. Indeed, the behavior and actions of the user influence the narrative and makes the story interactive. At the core of the system there is an ad hoc network, consisting of fixed multimedia nodes embedded in the environment and mobile nodes associated with users.

The Smart Messages (SM) platform, used to implement the mobile services framework, shares the idea of execution migration with mobile agents [13, 10], and active networks [7, 16, 19]. Mobile agents name nodes by fixed addresses and commonly know the network configuration a priori, while SMs name nodes by content and discover the network configuration dynamically. In contrast to mobile agents, SMs are responsible for their own routing at each node in the path between two nodes of interest. This feature allows SMs to adapt quickly to changes that may occur both in the network topology and the availability of resources at nodes.

Although the SM computing platform shares some of the design goals and leverages work done in active networks (AN), it differs from AN in several key features. AN target improved performance for end-to-end data transfer in relatively stable networks, while the SM platform helps the development of outdoor ubiquitous applications. Unlike AN, the SM platform defines a computing model whereby several SMs can cooperate, exchange data, and synchronize with each other through the tag space. In terms of migration, AN do not transfer the execution state from node to node whereas the SM model does. The migration of the execution state for SMs trades off overhead for flexibility to react “on-the-spot” to adverse network conditions.

7. Conclusions

The design and deployment of services over ad hoc networks motivate the need for agile and context-aware service interaction models able to cope with the highly dynamism of such networks. To address this need, we designed and implemented the mobile service model. The main feature of our model is that services are capable of adapting to context changes that often occur in ad hoc networks by migrating, during execution, to more suitable nodes available in the network. Such context-aware migration allows the service to
efficiently accomplish its task and to constantly respond to the user context profile. Initial experiences with our model have shown promising results. We applied the model to a prototype mobile service, and the evaluation consisted of experiments over a mobile ad hoc network of PDAs and simulations for larger scale networks. The experimental results demonstrated the viability of our model in highly dynamic ad hoc networks such as cars moving on a highway. The simulation results showed that our mobile service performs better than a traditional implementation based on a centralized approach in terms of response times, efficiency, and communication overhead. Furthermore, the mobile service approach proved to be scalable in the studied scenarios.

References