Supporting Real-Time Applications in Mobile Mesh Networks

Károly Farkas
Computer Engineering and Networks Laboratory, ETH Zurich
Gloriastrasse 35
CH-8092 Zurich, Switzerland
farkas@tik.ee.ethz.ch

Bernhard Plattner
Computer Engineering and Networks Laboratory, ETH Zurich
Gloriastrasse 35
CH-8092 Zurich, Switzerland
plattner@tik.ee.ethz.ch

ABSTRACT
Real-time applications in mobile mesh (often called self-organized or ad hoc) networks are a major technical challenge. On one hand, mobile devices and wireless connections between them are becoming ubiquitous, and real-time applications such as games or Voice-over-IP would be very attractive to their users. On the other hand, supporting or provisioning real-time services in such an environment is quite difficult since the service supporting procedures must cope with the high level of device heterogeneity, degree of mobility, and take limited device resources into account.

In this paper, we identify and discuss the different phases of service provisioning, than we introduce SIRAMON, a generic, decentralized service provisioning framework for self-organized mobile networks. SIRAMON integrates the required functions to deal with the whole life-cycle of services. SIRAMON offers sufficient capabilities to specify, deploy, instantiate and manage not only trivial but also complex services like real-time mobile group applications.

Keywords
Service Provisioning, Mobile Mesh Networks

1. INTRODUCTION
Mobile mesh networks have been receiving much attention recently due to their immense field of application. A mesh network is built of a collection of diverse nodes (users). These nodes form a multi-hop network communicating spontaneously without relying on any pre-existing infrastructure or central administration. In order to make a mesh network functional, the nodes must organize themselves. They not only provide terminal but also relaying functionality for distant nodes. In this mobile ad hoc environment, efficient service provisioning is difficult to realize and requires flexible and distributed mechanisms. While the different applications and the great number of devices make them interesting, the lack of central infrastructure, the high level of heterogeneity, the degree of mobility and the resource constraints of devices make it hard to provide ad hoc services.

So far, applications of Mobile Ad hoc NETworks (MANETs) [1] have been envisioned mainly in the field of emergency and military situations. However, MANETs offer many more possibilities. Real-time applications, such as multiplayer games\(^2\), group-work, multimedia entertainment, Voice-over-IP or the so-called 'edutainment', are the most attractive candidates to be used over mobile networks [2].

While new applications in MANETs will provide powerful environments for real-time group services, the complexity of service deployment and management in mobile mesh networks calls for the support of a service provisioning framework. Traditional techniques for service provisioning used in communication and data networks (Jini [3], UPnP [4], SDP [5], Chameleon [6], etc.) are not well suited for MANETs. First, they focus on a subset of service provisioning functionality (e.g., resource discovery) only. And second, they are often based on the client-server model using a central, fixed infrastructure which is not available in a mobile mesh network.

The lack of a solution motivated us to develop a new service provisioning framework for self-organized environments, called SIRAMON (Service provisioning FRAMework for self-Organized Networks) [7]. In this paper, we identify and discuss the necessary functions to provide mobile ad hoc services, and introduce the architecture and some mechanisms of SIRAMON. SIRAMON is based on a modular and distributed design. Its components can be replaced according to specific demands which makes it possible to support different type of services. SIRAMON consists of the following modules: (i) Service Specification; (ii) Service Indication; (iii) Service Deployment; (iv) Service Management; (v) Environment Observer.

---

1We are using the terms mesh, self-organized and ad hoc in the same meaning in this paper

2As part of the co-operation between SUN and NOKIA, the latest game console of NOKIA, called N-Gage, is now furnished with a networking middleware and mobile Java technologies providing a Java-based, cross-platform approach to mobile multiplayer gaming
Upon establishment of the ad hoc network, the initiator device advertises the availability of a new application/service on the network. When each of the connected devices is informed they can decide whether to join the game or not. Thereafter, the service will be automatically deployed on the selected devices (and in the network) and the game can be started. Further passengers are allowed to join (if the game allows for late arrivals) and leave the ongoing group game at any time. Even the initiator must be able to leave without interrupting the game session for the remaining players.

2.2 Phases of Service Provisioning

Analyzing this sample application scenario we can identify the following phases of service provisioning.

In the first phase, the service to be provisioned has to be specified and named. We call this phase Service Description. The service specification must be able to describe the role of the node in the service, the functions of the service elements and the connections among them. For example, in our game service the specification has to describe the participating device’s role (e.g., active player or auxiliary node which just aids the service to run), the device-level local view (the game is composed from moving pictures with music, sounds and voice requiring real-time interaction from the player, etc.), and the network-wide global view (the device has to communicate real time with all other player devices, etc.) of the game.

When the ad hoc network is established, the initiator device (the host) has to advertise the game service such that new participants can join. On the other hand, the devices of the other passengers must be able to lookup the service before deploying and using it. This phase is called Service Indication. Mobile networks have special constraints which must be addressed to develop appropriate service discovery mechanisms. Usually, MANETs cannot provide a permanent, central directory where the announced services can be registered and from where the available services can be read out. Moreover, the service hosting role can change dynamically in a mobile ad hoc group service. For example, in our scenario the initiator device hosted the game service but when it quits the game some other device has to take this role over or this role has to be implemented in a distributed manner.

When the devices in the ad hoc network are informed of the game service the next step is the Service Deployment phase when the game software has to be requested, downloaded, installed, configured and activated. Every device has to accomplish these procedures separately but in a synchronized manner to get a consistent state by when the game starts. This phase introduces some interesting problems. In general, heterogeneous devices form the ad hoc networks (e.g., laptops, PDAs, mobile phones, etc.) representing several different platforms. This requires the support of all the ported versions of the service software to these platforms from the service hosting device or the service has to be implemented in a platform independent way. Moreover, the lack of central infrastructure and the mobile behavior of the devices make it impossible to discover network resources beforehand and to use them permanently during the service session. For example, it can happen that a route to a re-

2. SERVICE PROVISIONING IN MOBILE MESH NETWORKS

Service provisioning in mobile mesh networks is faced special difficulties due to the constraints of MANETs—such as lack of central infrastructure, high level of device heterogeneity, degree of mobility, limited device and network resources. In this section, we survey these difficulties and identify the different service provisioning phases using a sample mobile ad hoc application scenario.

2.1 Sample Application Scenario

We use an imaginable scenario for a real-time multiplayer game in a mobile mesh network as our sample application scenario (cf. Figure 1).

A tourist, travelling by train, wants to spend his travel time playing a multiplayer game. To do so, he has a mobile device (e.g., a laptop) with a game software installed on it. Some other passengers on the train are supposed to possess also mobile devices that are able to communicate among each other and, at least, a couple of them are willing to play. While only some devices are capable to actively participate in the game, all may act as a relay to forward data. Since they have come close together, a mobile ad hoc network has been setup spontaneously.

![Figure 1: Distributed multiplayer game in a mobile mesh network](image-url)
mote player device has to contain auxiliary nodes to relay data. But mobile devices can move almost arbitrarily so the selection of these auxiliary nodes cannot be done very in advance, furthermore it is doubtful that these nodes will be permanently available during the whole game session.

After the game started the service maintenance, reconfiguration and termination functions come to the front. We call this phase Service Management. In the course of service maintenance the dynamic adaptation of the service to the resource variations must be assured. The players may experience frequently degraded performance of the service due to dynamic changes in network conditions (e.g., sudden drop in bandwidth when one of the player devices moves away from the others hereby reducing its communication capabilities) or other resources such as local computational cycles and memory space. Service reconfiguration takes place fundamentally in two forms, locally and globally. We are talking about the former if a service user device modifies the configuration of its local service instance (e.g., selecting a better resolution for the pictures, etc.). In the latter, the network-wide global view of the service session changes, for example, when a new player joins the game session. Local reconfiguration is mostly transparent for the other users while global reconfiguration is always perceptible for every participant. Service provisioning must support global reconfiguration. Service termination can be considered as a special reconfiguration if a service device stops running the service instance, such as the game initiator when it quits the game. This incurs also the release of the reserved resources.

Dynamic adaptation to the resource variations and reconfiguration of the service session demand the continuous monitoring of the resources and the service context. This comprises the gathering and transformation of resource and context information into an appropriate form which can be used as input in the aforementioned functions (e.g., in service maintenance or reconfiguration).

Figure 2 depicts the logical/chronological sequence of the functions and phases discussed above.

3. SIRAMON, OUR GENERIC SERVICE PROVISIONING FRAMEWORK

Above, we discussed briefly the characteristics of service provisioning in self-organized networks using a sample multiuser application scenario. In this section, we introduce our new, generic service provisioning framework, called SIRAMON, for self-organized networks which integrates the discussed functions.

3.1 Design Goals

Our design goals were the following:

(i) **Target application context is in self-organized networks:** The framework must be designed for use in self-organized networks.

(ii) **Decentralized operation mode:** The framework must work without relying on central infrastructure.

(iii) **Comprehensive solution:** The framework must support the service session during its whole life-cycle, i.e., from creation to termination.

(iv) **Transparency:** The framework must conceal the details of service provisioning functions from the applications providing a transparent interface to them.

(v) **Robustness:** The framework must aid the service session to dynamically adapt to mobility and resource variations. Moreover, it must cope with the heterogeneity of devices.

(vi) **Flexibility:** The framework must be generic and give support to a large variety of applications. Specifically, the framework must be flexible enough to support complex ad hoc group applications.

3.2 Assumptions

We assume that all the nodes wishing to communicate with other nodes within the mobile mesh network are willing to participate fully in the communication protocols. Moreover, all nodes are mobile, connected via wireless links and are capable of relaying packets on behalf of others in this network.

Nodes within the ad hoc network may appear and disappear at any time without notice. They may even move continuously, but we assume that the speed with which they move is moderate with respect to the packet transmission latency and wireless transmission range.

Each node has a unique identifier (address) by which it will be known in the mesh network.

Moreover, all nodes are furnished with SIRAMON in advance and the required software of the service (including ported versions for different device platforms) can be found...
on at least one networking node (very often this will be the initial service host).

### 3.3 SIRAMON

Our service provisioning framework is based on a decentralized and modular design. Every device runs an instance of SIRAMON which handles the control and synchronization among the devices, as well. We integrated the functions of the different service provisioning phases, as we discussed above, into separate modules. This makes the framework generic giving the possibility to replace the functions of a module with others which may fit better with the actual environment or the application's needs. Moreover, we don’t want to reinvent the wheel, thus with the modular design we can easily integrate procedures which have been already developed for given functions (such as service advertisement/lookup or service deployment).

![Figure 3: Ad hoc device model with SIRAMON](image)

In our device model (see Figure 3), we introduced a management middleware layer where SIRAMON integrating the service provisioning functions is located. This layer provides an interface between the device’s resources and the applications. The Device Resource Manager is responsible for controlling these resources and mapping them to service elements. Note that in this paper, we do not cover this latter module. In the following, we describe the different modules of SIRAMON.

#### 3.3.1 Service Specification

It defines the service model used in the given service. SIRAMON is not bounded to any specific service model. The only requirement is that the service model must be able to describe the role of the node in the service, the functions of the service elements and the connections among them. In case of different types of services, it can be reasonable to use different service models if the other model suits better for describing the given service. For example, a simple, location based service (e.g., a network printer) can use an attribute-value based compound service model. In this model, the service is described by using a hierarchical attribute-value pair tree structure which reflects the location of the resource (i.e., the printer). On the other hand, a complex, distributed group service (such as an online group game) can be more appropriately described by using a component-based service model in which the service is composed by simpler, reusable service components in a well defined way. However, the service model should be carefully selected for the given service because it constitutes the basis of service provisioning and it is used in the course of the whole provisioning activity.

In case of our prototype services (such as the discussed multiplayer game service), we are using a component-based service model with XML (eXtensible Markup Language) [8] as the service description language. In this model, services are designated as leaves of a hierarchical service identifier tree. The hierarchy of the tree can be adapted to the respective application(s). Figure 4 depicts a sample service tree. In this tree, we distinguish service categories and specific services. A service category represents a class of specific services but does not mean any real service itself. While a specific service is a real service which can be accessible by the users. In Figure 4, ellipses represent service categories while specific services are denoted by boxes.

![Figure 4: Sample hierarchical service identifier tree](image)

Moreover, we are using Uniform Resource Identifiers (URIs) [9] to label the services. A URI is built of the scheme’s name (e.g., siramon) and the hierarchical part of the service identifier. The slash ‘/’ character is used to separate the hierarchical components, thus the definition of our URI’s syntax is the following:

```
absolute_URI = scheme ‘://’ hier_part
hier_part = ‘//’ authority ‘/’ path_segments ‘/’
local_name
```

After this, a game service (e.g., DeathMatch) can be addressed by the following URI:
The service description of a specific service, what we are using, can contain general information about the service; details about connections (ports) to enable modular service decomposition; the additional sub-services required together with the connections among them; information about the service code together with the resource demands for code execution; the service’s available sessions; as well as other service specific things. Figure 5 depicts the structure of the XML service descriptor of a specific service. In this figure, XML elements are graphically represented as ellipses and their attributes as boxes. Dashed ellipses indicate optional elements. Double ellipses depict one or more, or if dashed, zero or more elements. And dashed boxes indicate optional attributes.

Note that here we disregard the detailed discussion of all the elements of this service descriptor due to the lack of space and its reduced relevance to understanding the main idea of the framework.

3.3.2 Service Indication
This module covers service advertisement and service lookup procedures. The URI based service identifiers are used in these procedures, as well. With these URIs not only exact but also partial matching of the identifiers are possible making service discovery more powerful, universal and easier. For example, if an application looks for multiplayer games (‘.../Games/Multiplayer’) using partial matching, it will receive replies with service descriptions that match exactly the requested URI, as well as replies that match for more general services (e.g. ‘.../Games’).

As we discussed earlier, due to the lack of central infrastructure, for service advertisement and lookup the traditional central directory model cannot be applied in ad hoc environments. A solution could be to replicate this directory on every device connected to the mesh network. This introduces synchronization and propagation overhead to maintain a consistent database on all the devices but makes service lookup very fast and efficient. Another solution is to establish a so-called virtual backbone in the ad hoc network selecting some devices which store a copy of the service directory. The virtual backbone must be reachable by every other device (i.e., every device has to be able to reach at least one backbone node within one hop). This approach slows down the service lookup and is less efficient compared to the previous one, but introduces less administrative overhead—however, it requires an extra procedure to select and maintain the set of virtual backbone nodes. In case of mobile ad hoc group applications, the latter solution seems to be a better choice, since the service devices constitute by themselves an overlay network which can be considered as part of the virtual backbone.

3.3.3 Service Deployment
Deploying a service requires requesting/downloading software according to the specification; discovering/gathering resources; mapping of this specification to resources; configuring the resources and installing/configuring the down-loaded softwares; activating the service in a synchronized manner along with the other service participants; and handling the control on to the management module. The Service Deployment module includes these functions.

Traditionally, in infrastructure-based networks service deployment consists of two levels: the network level and the node level. Briefly, on the network level the participants of the service should be explored and the required network resources should be discovered and selected, since on the node level the appropriate service components should be deployed on the selected resources. However, in ad hoc networks due to the lack of central infrastructure and the mobile behavior of the devices these two levels cannot be clearly distinguished. It can easily happen that a previously selected relay node moves away even during the deployment phase and another has to be selected on the network level of the deployment activity. Thus, the network and node level service deployment functions have to synchronize continuously each other during the whole deployment phase.

The Service Deployment module has to tightly co-operate with the Environment Observer for getting information about the available resources and with the Device Resource Manager for mapping the resources to the service components.

3.3.4 Service Management
This module integrates the service maintenance, reconfiguration and termination functions.

Service maintenance is responsible for the dynamic adaptation of the service to the resource variations in order to optimize user’s perceived service quality. Using the terminology in [10] we distinguish three ways to adapt to changes in resources. These adaptations should be performed only after the effect has been observed to persist beyond a threshold amount of time, since reacting to transient resource changes results in unjustified overhead and instability of the system.

(i) Service-intelligent adaptation: The application is powerful enough to do its own adaptation to resource changes. For instance, the audio subsystem of a teleconference service can combine multiple streams encoded for different bit rates into a single clip. Thus, during runtime, the audio stream dynamically adapts to changes in bandwidths. In this case, the management module should directly take advantage of the application adaptation mechanism.

(ii) Service-specific adaptation: The application provides mechanisms for dynamic adjustment, but does not do so automatically. For instance, for bandwidth adaptation, there are different instances of the same codec intended for different bit rates. To select the appropriate one the application needs to be notified about the resource changes. In this case, the management module, consulting with the Environment Observer for resource changes, is responsible to provide feedback to the application to enable dynamic adaptation.

(iii) Service-independent adaptation: If the management module doesn’t have information about the underlying
Figure 5: Structure of the XML service descriptor of a specific service
implementation of the application, it treats the application as a black box and tries to find other means to adapt to the changes. For example, to adapt to high packet loss rate, forward error correction can be used.

Service reconfiguration, as we discussed above, has two fundamental forms, local and global reconfiguration. Local reconfiguration is mostly transparent for the other users and it takes place when the context of the device changes or the service user intends to modify the running service session. In global reconfiguration, the network-wide global view of the service session changes (e.g., a new user joins) and global reconfiguration is always perceptible for every participant and requires global synchronization. The Service Management module gives support for global reconfiguration while local reconfiguration is in charge of the application.

Service termination can be considered as a special form of reconfiguration when a device stops running the service instance. All the other service participants have to be informed about the termination and the reserved resources have to be released.

The Service Management module also has to tightly cooperate with the Environment Observer and the Device Resource Manager.

3.3.5 Environment Observer
This module is responsible for monitoring the device resources and the service context. Resource and context information have to be gathered and transformed into an appropriate form, which can serve as input for the deployment and management modules.

4. RELATED WORK
Several proposals have been developed to standardize the functions of service advertisement and service/resource lookup (e.g., Sun’s Jini [3], Microsoft’s Universal Plug and Play (UPnP) [4], Bluetooth’s Service Discovery Protocol (SDP) [5], Service Location Protocol (SLP) [11] of IETF, IBM’s Salutation [12]). However, these proposals mainly based on the use of a central directory. Moreover, almost all of them include the client-server paradigm and mainly focus on resources/services provided by devices (e.g., printers, fax-machines) which restricts their applicability in ad hoc networks.

In contrast, some recently appeared new proposals directly target infrastructure-less networks. The Konark system [13] is a middleware designed specifically for service discovery and delivery of device independent services in ad hoc networks. Konark is based on a peer-to-peer model with each participating device having the capabilities to host its local services, deliver these services using a resident micro-HTTP server, query the network for available services offered by others, and use the services it discovered in the network. Another proposal is the distributed service discovery architecture [14] which relies on a virtual backbone for locating and registering available services within a dynamic network topology. It consists of two independent mechanisms: the formation of a virtual backbone which includes the nodes acting as service brokers; and the distribution of service registrations, requests and replies which establishes sub-trees rooted at service requesting devices and registers servers for efficient dissemination of the service discovery messages. However, none of these proposals provide the full functionality of service provisioning.

Several research projects have recently focused on service specification, dynamic service creation and deployment in the context of active networks. Chameleon [6, 15] is a platform independent, node level service deployment framework which facilitates the installation of a service on a single node. It features a component-based service model in which a service is specified using a node-independent, XML based service description. Active pipes [16, 17] serve as an abstraction to describe the network level requirements of service components which are to be deployed on a path between two end-systems. Hierarchical service deployment [18] introduces an automated approach to network level service deployment that scales well to large heterogeneous programmable networks. However, these research works target only infrastructure-based networks and don’t offer comprehensive service provisioning functionality.

Context information management has been receiving much attention due to the increasing demand of dynamic adaptation of service sessions to context changes. Recent research projects (e.g., agent based context usage [19], CONTEXT [20], general context management framework [21]) propose solutions for collecting and managing user context information to make its usage possible in the service. However, neither these proposals offer full service provisioning functionality.

And finally, provisioning of Web related services is a broad area which is in extensive research and standardization phase at the moment. The Web Services architecture [22] is the most widely used and referenced current work on distributed service provisioning. Technologies and standards of this architecture (such as SOAP - Simple Object Access Protocol, WSDL - Web Services Description Language, UDDI - Universal Description Discovery and Integration, etc.) gives a generic way to provide and use Web-based services. However, this approach requires central infrastructure with permanent server nodes and often powerful devices with sufficient computation and communication capabilities which is not the case in mobile networks.

Our service provisioning framework targets ad hoc environments. By distributed design it doesn’t rely on central network infrastructure. It is comprehensive and integrates all the required functionality of service provisioning.

5. CONCLUSIONS
In this paper, we identified the different phases of service provisioning in ad hoc networks and discussed the related difficulties. Based on this overview, we introduced SIRAMON, a new, generic service provisioning framework for self-organized networks. SIRAMON, with its decentralized, modular design suits well for ad hoc environments and is flexible enough to cope even with complex services, such as real-time mobile group applications.

Currently, we are working on the exact definition of the in-
terfaces between the individual modules to alleviate the replaceability of them. Moreover, focusing on management issues, we plan to develop algorithms which support dynamic service maintenance and reconfiguration. To investigate the scalability and other performance behaviors of SIRAMON we have been implementing it in a network simulator. As proof of concept, we also started to build up an ad hoc testbed network from mobile devices with the prototype implementation of SIRAMON and a simple example real-time multiplayer game application. And finally, based on the SIRAMON framework we are about to setup an EU international research project with the topic of Multiplayer Game Support in Mobile Ad Hoc Networks.

6. ACKNOWLEDGMENTS
This work was encouraged by NTT DoCoMo Euro-Labs. We would like to thank DoCoMo for the beneficial and interesting discussions in this new area.

7. REFERENCES