Framework for Dynamic Resource-Constrained Service Composition for Mobile Ad Hoc Networks

Gerald Kaefer[1], Reiner Schmid[1], Guenter Prochart[2], and Reinhold Weiss[2]

[1] Corporate Technology - Software and Engineering
Siemens AG
Munich, Germany
{gerald.kaefer, reiner.schmid}@siemens.com

[2] Institute for Technical Informatics
University of Technology
Graz, Austria
{prochart, rweiss}@iti.tu-graz.ac.at

This work is part of the PalCom project. http://www.ist-palcom.org/

Abstract. In this paper we present a framework for dynamic resource-constrained composition in mobile ad hoc networks (MANETs) in order to manage the permanently changing environment by reason of mobility and resource variability. Moreover, dynamic composition alone is not sufficient because devices could typically be very resource-constrained and the resource situation may change rapidly, e.g. battery power may fade away, or due to memory limits a service may not currently run on a device, though it is available in principle. The described framework provides two novel aspects: First, the framework provides automatic execution of dynamic compositions for end-to-end functional descriptions even if functional nodes contained in that description are not directly compatible. Second, the framework approaches resource optimization based on simple rules, but with powerful means, such as deployment provisioning and migration of services and components within an ad hoc network.

1 Introduction

Service composition refers to the method of constructing composite services with the help of small and simple executable services or components. In mobile ad hoc networks (MANETs) the availability of services running on different devices changes dynamically, because devices joining and leaving the network due to movement and resource variability. Hence, static composition is not sufficient. In such environment even dynamic composition alone is not sufficient, because devices could typically be very resource-constrained and the resource situation may change rapidly, e.g. battery power may fade away, or due to memory limits a service may not currently run on a device, though it is available in principle. Therefore, resource requirements have to be considered during a dynamic composition process.

We work toward a framework for dynamic resource-constrained service composition. Task assignments are characterized with Functional Task Descriptions (FTD), which provide basic information about necessary functionality for the composite service. Our approach uses end-to-end descriptions of services and functions; all in-between functionality is provided by compatible services. The framework approaches resource optimization based on simple rules, but with powerful means, such as deployment provisioning and migration of services and components within an ad hoc network. Hence, the optimization strategy for resource-constrained service composition is rule based rather than depending on path-finding and planning methods.

2 Architectural Overview

The nature of the dynamic composition process is a service composition procedure involving one or more devices, and therefore requiring a distributed architecture. Fig.1 outlines the basic services, which have to be present as core services at every device (or at least at selected broker devices). These basic services have to be installed at least at one initial device; any other device can use a bootstrap service. This service sends out a framework deploy request which can be answered by the initial device, where all basic services reside. The services can then be deployed to all other devices.

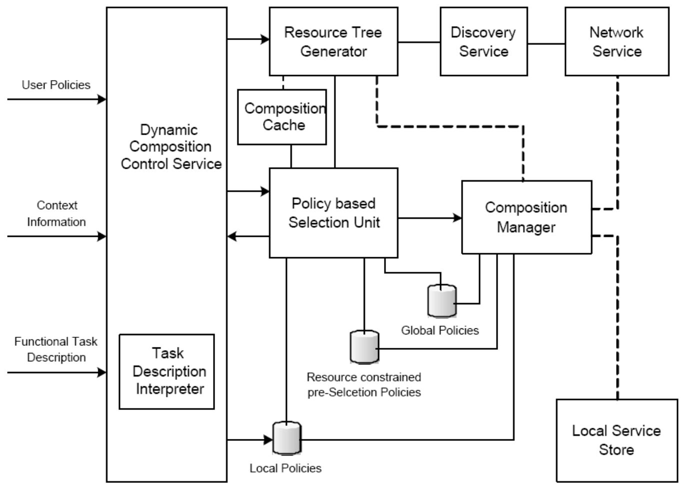

Fig. 1. Resource-constrained dynamic composition framework architecture

The framework consists of the following core services:

Network Service: provides the connectivity to other devices and hides the specific characteristics of the underlying communication hardware.

Discovery Service: receives look-up queries for services and resources at the provider side and uses the network service. The look-up scope depends on query attributes and local context of the device.

Local Service Store: is a collection of services locally available. This information is used for internal as well as external discovery.

Task Description Interpreter: analyzes end-to-end functionality descriptions, which are specified with a script-based language. Such a description is referred to as Functional Task Description (FTD).

Composition Cache: stores valid compositions for assignments outlined in different functional task descriptions.

Policy-based Selection Unit: is responsible to process decisions based on local, global, and resource-constrained pre-selection policies. The latter are already used during resource tree generation in order to reduce the tree complexity.

Dynamic Composition Control Service: provides the outbound interface (takes FTDs, user policies, and context information) and supervises the overall composition process.

Resource Tree Generator: builds an information tree about resources and services which are locally and externally available. All elements of the tree have compatible interfaces to their neighbors. Thus, the dynamic grown resource tree is the basis for finding the required end-to-end functionality.

3 Resource Description Model

In this section we characterize resource properties and dependencies, because they are a basic requirement of dynamic composition. In order to do this a metadata approach is used. For the focus of our work, we constitute QoS parameters, device resources, deployment, and functional resource requirements. Fig. 2 shows the resource classes. In general, resources are segregated into two categories:

- **First Order Resources (FOR):** Low level device-oriented resources such as power, CPU capacity, bandwidth, memory, etc.
- **Second Order Resources (SOR): Virtual** artifacts that can be manipulated meaningfully as resources, e.g., software components, services, etc.

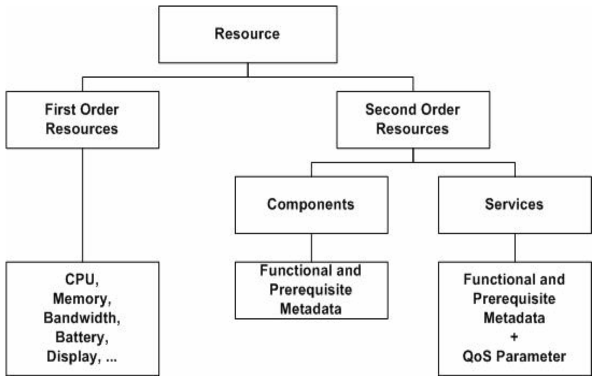
Resource
First Order Resources
Second Order Resources
Components
Services
CPU, Memory, Bandwidth, Battery, Display, ...
Functional and Prerequisite Metadata
Functional and Prerequisite Metadata + QoS Parameter

Fig. 2. Resource classes used in the resource description model

Resources are characterized with descriptors. Table 1 shows the structure of a descriptor for second-order resources. Basically, we differentiate two types of metadata. The first type is called functional metadata, whereas the second type is called prerequisite metadata.

- **Functional Metadata** describes exported interfaces, types, and contracts for interface handling, or other information as for example quality of service or charging issues.
- **Prerequisite Metadata** describes essential information for component deployment and distribution.

Table 1. Structure of a resource descriptor

Attribute	Attribute Type	Requirement
GUID	General	mandatory
Component or service description	Functional Metadata	mandatory
QoS parameter	Functional Metadata	optional for components; mandatory for services
Deployment FOR	Prerequisite Metadata	mandatory
Deployment SOR	Prerequisite Metadata	mandatory
Functional FOR	Prerequisite Metadata	mandatory
Functional SOR	Prerequisite Metadata	optional

Resource descriptors consist of a basic set of attributes. This set is not exhaustive and may be extended. Table 1 shows the structure of a resource descriptor and Table 2 shows a basic set of attributes used to describe QoS parameters of a service. In most instances we are consistent with parameters outlined in [1][2][3][4]. The basic set of attributes in Table 3 points out the required device capabilities for component deployment, and Table 4 presents the required resource needs in order to execute components and services at different service levels (Minimal Quality, Low Quality, Medium Quality, High Quality, and Best Quality).

Table 2. Basic set of QoS parameters

Attribute	Explanation
Response Time	The response time measures the expected delay between the time when the request is sent and the time when results are received.
Service Cost	The price that a service requestor has to pay for invoking an operation of a service.
Reliability	The reliability of a service is the probability that a request is correctly responded within the maximum expected time frame.
Availability	The availability of a service is the probability that the service is accessible. In mobile ad hoc networks this can be described in correlation with the movement probability of the node providing a service.
Robustness/ Flexibility	Degree to which a service can function correctly in the presence of invalid, incomplete or conflicting inputs.
Exception handling	Since it is not possible for the service designer to specify all the possible outcomes and alternatives the treatments of exceptions have to be declared.
Transaction	Transaction support is used for maintaining data consistency. From the perspective of a requestor, whether a service provides an undo procedure to rollback the service execution in a certain period without any charges is an important factor for the selection of a service.

Table 3. Basic set of attributes for FORs – Deployment [5]

Attribute	Explanation
CPU Class (Machine Class)	Uses a CPU descriptor (SH3, ARM, MIPS, x86), a benchmark value, or a clock frequency to indicate the necessary computing capability in order to execute a component or service in a reasonable way.
Memory (Primary Storage)	Required amount of memory to load and execute a component or service.
Secondary Storage	Required amount of secondary storage in order to perform tasks that need additional memory capacity.

Table 4. Basic set of attributes for FORs – Functional [5]

Attribute	Explanation
Memory (Primary Storage)	Required amount of memory to execute a component or service at a particular service level.
Secondary Storage	Required amount of secondary storage at a particular service level.
Bandwidth	Required communication bandwidth at a particular service level (e.g. for streaming services).
Energy Consumption	Benchmark value to indicate the energy consumption of a component or service at a particular service level. May be some classified indicator: high, medium, low.

4 Dynamic Resource –Constrained Service Composition

The workflow of the framework can be regarded as the dynamic resource-constrained composition process and is described in this section. The initiator of a dynamic composition process is always a device –capable of ad hoc networking– triggered by a human user, another device, or itself. A Functional Task Description (FTD) containing the required end-to-end functionality is the input for the dynamic composition process which comprises the following steps:

- functionality-based service lookup
- resource tree generation
- policy-based selection
- resource based optimization.

In the first step (functionality-based service lookup), the end-to-end functionality description in form of a FTD is passed to the discovery service. Found services are checked for compatibility. If compatible begin-node and end-node services have been found the composition process could be stopped, but could also be continued for optimization purposes. In the second step (resource tree generation), services, directly compatible to already found services, are recursively looked for. So for every discovered service all identified compatible services are valid results. In a following recursive step again all services which are compatible to the found services are valid resource tree results (leafs of the tree).

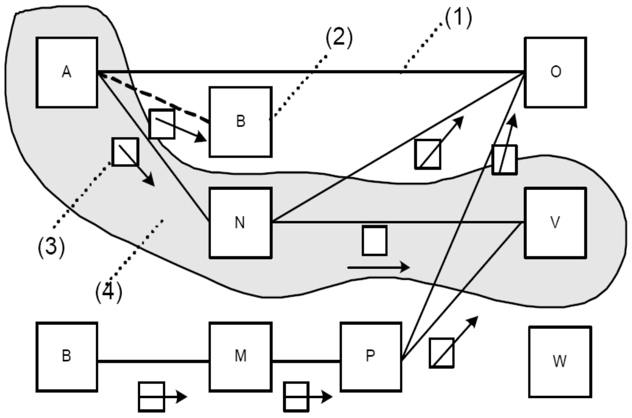

Fig. 3. Process of resource constrained composition (A and B denotes begin-nodes; O, V, and W denotes end-nodes). Valid compositions are end-to-end paths within the resource tree

Within the found services the resource tree generator always looks for services matching the required end-node functionality. The found services may not be directly compatible to the begin-node services. If the required end-node functionality is found within the resource tree a valid composition is found. Of course more than one composition could be identified during such a composition process. Fig. 3 illustrates this composition process with the following cases:

(1) Functional lookup provides direct compatible services, which would allow direct composition
(2) Dead end of resource tree, because of incompatible interfaces or resource constraints
(3) Lookup request for tree generation – including the requested end-functionality, policies, and an accumulated list of involved services.
(4) Shows the composition selected from the discovered solutions according to the given local and global selection policies.

In the third step (policy-based selection), policies are used to select a composition from the previously found ones. Within the framework it is possible to define rules of precedence for the following three types of policies used in the selection process:

- global policies - are mostly provided by network participants and should be used for general resource conflict management. Typical global policies are: all perfect (largest display, best connection, largest memory, …), most energy efficient, as cheap as possible (not all service will be free of charge), only trusted devices, best availability.
- local policies - are used locally at device level. They are predefined as default policies for a service by the developer of the service. The goal is to control the performance and security aspects of the device.
- resource-constrained pre-selection policies - used to provide physical resource requirements checks during the resource tree generation phase in order to reduce complexity and resource effort of the resource tree.

In the last step (resource-based optimization) all compositions that fulfill the required assignment specified in the functional task description are found. The appropriate requirement is to select the best solution based on resource needs.

5 Use Case Example

A user wants to show a video stored on his or her local device. So the task is *Show Video*. The extreme policies for this task, which the user could choose from, could be:

- as energy efficient as possible
- as large display as possible
- as cheap as possible

The FTD *Show Video* lists the necessary functional components in order to solve the task. In this special use case that means the user has to define at least *Video Source* and *Display*:

```
Link <Video Source> with <Display>;
```

Optionally, it is also possible to bring in anticipatory knowledge about structure of a requested composition. Relations can to be defined in the FTD. This may be done in the following manner for instance:

```
Link <Video Source> with <Display>;
Optional By
Link <Video Source> with <Streamer>
with <Display>;
Optional By
Link <Video Source> with <Streamer>
with <Format Converter> with <Display>;
```

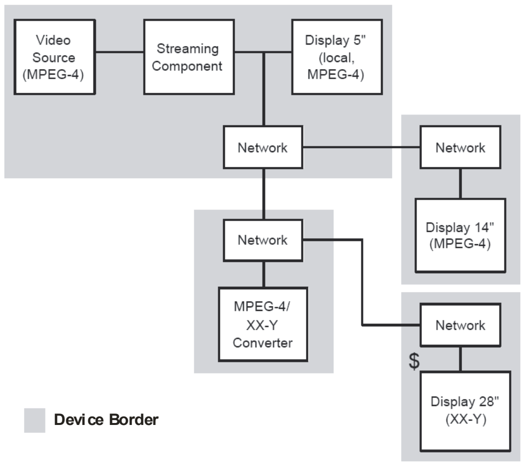

Fig. 4. Dynamic Composition and Resource management use case - showing functional composition and selection of the *Show Video* scenario

Fig. 4 shows the above described use case situation after the successful functional lookup. The resource management is responsible for connecting them by finding the missing functionality in-between. A developer or user can also define rules for an indirect connection if a direct connection is not possible. The resource management takes care of the efficient deployment of the requested functional components.
All resource constraints, e.g. memory, bandwidth, latency have already been checked during the pre-selection phase and are met. As shown, there are three possibilities of composition:

- show the video on the local display
- show the video on the 14” networked display
- show the video on the 28” networked display with costs

Thus, there are three possible composition policies. In the case, the default policy is *as large display as possible*, the 28” networked display with costs would be chosen. When the user realized the chosen composition he or she can accept it or decide for another policy. If the user would not like to pay for the service then he or she can add the policy *as cheap as possible*. This would lead to the composition with the 14” networked display. Furthermore, if the user adds another policy, e.g. *only own devices*, and then the video would be shown on the local display (of course, assuming that the two other displays do not belong to the user).

6 Summary and Future Work

We have illustrated our approach for implementing a framework for dynamic resource-constrained service composition in ad hoc network environments. The novel aspect of the presented framework is the automatic discovery and connection of glue logic for the requested end-to-end functionality. Devices can typically be very resource limited and the resource situation may change rapidly, hence our framework considers resource constraints. In order to keep the framework's resource consumption during the composition process low, the resource constraints of the required services are considered in an early composition process stage.

Prototyping is done with C# and MS Visual Studio 2005 for the target platform .NET CLR 2.0 and .NET CF 2.0, respectively. UPnP is used as internal underlying service discovery technology in order to find services, described as technical UPnP devices. Message and service descriptions contain data in XML-format providing the information required for dynamic composition.

Most of the described core services are implemented and used for internal performance evaluation. The performance evaluation of the dynamic resource-tree generation in real-world environments is of special interest. Therefore, resource descriptions of real-world components and Functional Task Descriptions in real ubiquitous environments are used for evaluation purposes.

These evaluations results will be the basis for future improvements of the resource optimization algorithms, i.e. service distribution, load sharing, and the resource variability estimation process.

7 Related Work

There is a lot of completed work and ongoing research in the field of service composition for wireless ad hoc networks.

Chakraborty et al. [6] have described distributed service composition protocols for mobile environments. These protocols are decentralized and utilize the service topology to compose services. Each composite request is independently assigned a composition manager. The selection of a composition manager is based on a device-specific potential value and takes into account services present in the device, computation and energy resources, and service topology of the surrounding vicinity. We agree on their broker selection protocol in order to find a composition manager device that handles the integration and execution of the composite requests.

Basu et al. [7] have illustrated an approach for modeling service composition using hierarchical task graphs. They proposed distributed algorithms for instantiating hierarchical task graphs and for handling disruptions in services due to mobility of devices.

We agree on their hierarchical approach to query for services, which start at a top level and then queries the neighbor devices.

Maltz [8] has demonstrated a scheme of path-state and flow-state mechanisms that can be used to explicitly manage resources in an ad hoc network. The work attempts to provide a way to control the consumption of resources in the network, such as the battery power or the carrying capacity of the nodes. We agree on the problem formalization, the work concerns resource constraints.

Mokhtar et al. [9] have shown ad hoc composition of user tasks in pervasive computing environments. A solution is achieved in two steps. The first step performs a semantic matching of interfaces that leads to the selection of the set of services that may be useful during the integration. The second step performs a conversation matching starting from the set of previously selected services, thus obtaining a conversation composition that behaves as the task's conversation. We agree on operation matching.

Varshavsky et al. [10] have presented a new cross-layer architecture that integrates service discovery and service selection functionality with existing routing protocols, thus allowing nodes to learn about available servers and routes to them simultaneously. We agree on the approach for reselection and rediscovery policies.

Ni [11] has sketched an ontology-enabled service oriented architecture for general issues in pervasive computing. Based on the semantic description of web services it is argued that planning could be applied in service composition but need to be customized. Differences between planning and service composition are analyzed and an approach to realize ad hoc service composition is presented. We agree on the parameter matching between services.

We partially agree with the outlined work. Our work differs from related work in the description of services and software components, especially by adding a description of hardware resources, and an optimization strategy for resource-constrained service composition which is rule based rather than depending on path-finding and planning methods.

References

1. Yu, T., Lin, K.: A Broker-Based Framework for QoS-Aware Web Service Composition. Proc. IEEE International Conf. on e-Technology, e-Commerce and e-Service (2005) 22-29
2. Liu, Y., Ngu, A.H.H., Zeng, L.: QoS Computation and Policing in Dynamic Web Service Selection. Proc. 13th International World Wide Web Conf. (Alternate Track Papers & Posters) (2004) 66-73
3. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality Driven Web Services Composition. Proc. 12th International Conf. on World Wide Web. ACM Press (2003) 411-421
4. Ran, S.: A Model for Web Services Discovery With QoS. SIGecom Exch. ACM Press (2003) 1-10
5. Kaefer, G., Haid, J., Voit, K., Weiss, R.: Architectural Software Power Estimation Support for Power Aware Remote Processing. Proc. International Conf. on Parallel and Distributed Computing Systems PDCS (2002)

6. Chakraborty, D., Joshi, A., Finin, T., Yesha, Y.: Service Composition for Mobile Environments. Journal on Mobile Networking and Applications, Special Issue on Mobile Services (2005) 435-451
7. Basu, P., Ke, W., Little, T.D.: Scalable Service Composition in Mobile Ad hoc Networks using Hierarchical Task Graphs. Proc. 1st Annual Mediterranean Ad Hoc Networking Workshop (2002)
8. Maltz, D.A.: Resource Management in Multi-hop Ad Hoc Networks. Technical Report, School of Computer Science, Carnegie Mellon University (1999)
9. Mokhtar, S.B., Georgantas, N., Issarny, V.: Ad Hoc Composition of User Tasks in Pervasive Computing Environments. Proc. 4th International Workshop on Software Composition (2005) 31-46
10. Varshavsky, A., Reid, B., de Lara, E.: A Cross-Layer Approach to Service Discovery and Selection in MANETs. Proc. 2nd International Conference on Mobile Ad-Hoc and Sensor Systems (2005)
11. Ni, Q.: Service Composition in Ontology enabled Service Oriented Architecture for Pervasive Computing. In Workshop on Ubiquitous Computing and e-Research, Imperial College London, UK (2005)