OPEN Project

STREP Project FP7-ICT-2007-1 N.216552

Title of Document:	Detailed network architecture
Author(s):	A. Nickelsen, R. L. Olsen, M. Martin, E. Kovacs, G. Ghiani, H. Klus, S. Marzorati, A. Grasselli, M. Piunti
Affiliation(s):	AAU, NEC, CNR, ClU, Vod
Date of Document:	Feb. 2009
OPEN Document:	D3.1
Distribution:	Public
Keyword List:	Service migration, network architecture, OPEN Platform
Version:	V1.0

OPEN Partners:

CNR-ISTI (Italy)
Aalborg University (Denmark)
Arcadia Design (Italy)
NEC (United Kingdom)
SAP AG (Germany)
Vodafone Omnitel NV (Italy)
Clausthal University (Germany)

Title: Detailed network architecture	**Id Number:** D3.1

Abstract

Deliverable D3.1 presents the work carried out in Work Package 3 during the first year of the OPEN project, and elaborates on the OPEN platform and the architecture for supporting service migration. The deliverable presents and the developed architecture, distinguishing between core and support functions. This description is followed by a set of deployment scenarios where we further detail the interaction patterns, under the assumption of a centralized solution, i.e. using a dedicated migration server. Finally, we also present an initial analysis of the security requirements and sketch potential solutions to address them, The conclusion section wraps up the documents and provides an outlook on the next steps.

Title: Detailed network architecture	**Id Number:** D3.1

1 EXECUTIVE SUMMARY 3

2 INTRODUCTION 4

3 CORE FUNCTIONS FOR MIGRATION 8

3.1 MIGRATION ORCHESTRATION FOR TASK CONTINUITY 8
3.1.1 Candidate systems 8
3.2 CONNECTIVITY SUPPORT FOR MIGRATION 12
3.2.1 Candidate systems 14
3.3 CONTEXT MANAGEMENT 16
3.3.1 Candidate systems 18

4 SUPPORT FUNCTIONS FOR MIGRATION 24

4.1 TRIGGER MANAGEMENT 24
4.1.1 Functionality 24
4.1.2 Interactions 25
4.1.3 Requirements/Scenarios 25
4.1.4 Candidate systems 25
4.2 POLICY/PROFILE MANAGEMENT 25
4.2.1 Functionality 25
4.2.2 Interactions 26
4.2.3 Requirements/Scenarios 26
4.2.4 Candidate systems 26
4.3 SESSION MANAGEMENT 27
4.3.1 Functionality 27
4.3.2 Interactions 27
4.3.3 Requirements/Scenarios 27
4.3.4 Candidate systems 27
4.4 PERFORMANCE MONITORING 27
4.4.1 Functionality 27
4.4.2 Interactions 28
4.4.3 Requirements/Scenarios 28
4.4.4 Candidate systems 28
4.5 CLOCK/FLOW SYNCHRONIZATION 29
4.5.1 Functionality 29
4.5.2 Interactions 29
4.5.3 Requirements/Scenarios 29
4.5.4 Candidate systems 29
4.6 DEVICE DISCOVERY 30
4.6.1 Functionality 30
4.6.2 Discovery model 30
4.6.3 Interactions 32
4.6.4 Requirements/Scenarios 32
4.6.5 Candidate systems 32
4.7 SERVICE ENABLERS 35
4.7.1 Presence server 35
4.7.2 Location 36
4.7.3 Access Layer 37
4.7.4 Device Provisioning 37
4.7.5 Streaming 37
4.8 SUMMARY 37

5 DEPLOYMENT SCENARIOS 38

5.1 POTENTIAL ARCHITECTURES AND CHALLENGES 38
5.1.1 General Infrastructure 38
5.1.2 Network Operator / Cellular networks 39
5.1.3 Enterprise Networks 40
5.1.4 Home Networks 41
5.1.5 Ad-hoc/peer-to-peer networks 41
5.2 SUMMARY AND DEVELOPMENT PLAN 42

6 DETAILED INTERACTIONS IN THE OPEN SCENARIOS 43

6.1 HIGH LEVEL SCENARIO WITH INFRASTRUCTURE SUPPORT 43
6.1.1 SIP/IMS based architecture 44
6.1.2 Internet based architecture 47
6.2 DETAILED OPEN PLATFORM INTERACTIONS AT COMPONENT LEVEL 49
6.2.1 Interfaces between OPEN platform components 49
6.2.2 Interaction and component activity – example scenario 50
6.2.3 Migrating from many devices to many devices 52
6.3 DETAILED SOLUTIONS FOR SELECTED COMPONENTS 53
6.3.1 Solution proposal for Device Discovery 54
6.3.2 Context Management Framework 55
6.3.3 Mobility support in central migration server 60

7 SECURITY ASPECTS OF MIGRATION SUPPORT 65

7.1 OPEN SECURITY REQUIREMENTS AND SELECTION CRITERIA 65
7.2 OVERVIEW AND ANALYSIS OF EXISTING SECURITY SOLUTIONS 65
7.2.1 Secure connectivity 65
7.2.2 Security and privacy control for Context Management Framework 66
7.2.3 Trust establishment 67
7.3 SECURITY SOLUTIONS IN OPEN 67

8 CONCLUSIONS AND NEXT STEPS 69

9 REFERENCES 71

A GLOSSARY 73

1 Executive summary

This deliverable presents the work of Work Package 3 which has been carried out during the first year of the OPEN project. Work Package 3 is focused on developing the supporting platform for service migration as envisioned in the project.

The deliverable starts by introducing the platform, followed by a dedicated chapter on the core functions and describing them in further detail. The core functions entail *migration orchestration*, *connectivity support* and *context management,* which are core to the envisioned service migration. Following this, we describe the supporting functions, e.g. *performance monitoring*, *clock/flow synchronisation* etc. which are needed in order to obtain an efficient service migration and for supporting the application and core components in the migration process.

From this we investigate different deployment scenarios in order to place functionalities in the network, clarify assumptions, and explore possible interaction models. Based on these investigations we choose to focus the remaining work on scenarios which includes a migration server, i.e. a centralized support service that enables the migration process. We elaborate this with message chart diagrams, first with respect to external actors, and then with a focus on internal OPEN components. Scenarios without a migration server, e.g. ad-hoc scenarios are left for the second year in OPEN.

Finally, although not the core topic of the project and WP3, we take a look at the security requirements of the OPEN platform, and take a brief look at what possible means we can meet these requirements to obtain a secure service migration.

We conclude the deliverable after the security section, and provide an outlook for the coming period, which will focus on scenarios where no infrastructure is available, i.e. scenarios where we cannot assume a migration server being available. Further enhancements and clarification of the final interfaces specifications between components is also to be expected in the coming time period.

Work Package 3 work is supported in shape of several demos, developed by different partners, illustrating different aspects of the OPEN platform as described in this deliverable. The descriptions of those demos are documented in Deliverable D3.2, whereas the reader is encouraged also to read this deliverable in conjunction with Deliverable D3.1. Future aspects on this part are to reach a fully integrated platform which is capable of running all the demo applications as envisioned in different scenarios.

2 Introduction

An important aspect of ubiquitous environments is to provide users with the possibility to freely move about and continue the interaction with the available applications through a variety of interactive devices (including cell phones, PDAs, desktop computers, digital television sets, and intelligent watches). With such freedom envisioned, one big potential source of frustration is that people have to start their session over again from the beginning at each interaction device change.

The purpose of the OPEN project is to enable migration of services and applications, so that users may easily shift interaction devices and focus on the service itself. In [4] this focal point was summarized as

Migration = Device Change + Adaptation + Continuity.

In [3] a set of scenarios describing the migration process as seen from a user's point of view was described. Two major themes were creating the basis of this document, namely gaming and business. From these, requirements to the system were derived, which have been further analysed during the project.

In Work Package 3 the focus is on the supporting platform that enables applications and services to migrate easily. By careful study of the requirements described in [3] and [4] a set of functional blocks has been proposed. A short summary of the investigations and rationale for the components are as follows:

- *Migration orchestration*: We will need a component to manage the migration process, both at inter component level and also between platform and application/services.
- *Context management*: We plan to trigger the migration based on the user's given situation or context, hence we need a context management solution to gather, distribute and provide access to context information for other components.
- *Mobility support*: When the user moves around, there should be minimal hassle of adjusting and setting the system to the correct network etc. hence mobility support is needed.
- *Trigger management*: We will need a component to evaluate, based on the given context, when to trigger a migration process.
- *Session management*: We will need a component to extract, store and inject session information during migration in order to support application state persistence, which is needed to feature for the continued use of the application after migration.
- *Policy management*: Since many decisions are based upon policies, we need to have a management framework to handle policies.

- *Performance monitoring*: In order to make the right decisions about when and where to migrate, different performance metrics of the computational environment can be monitored. These can be host specific such as processing load, network specific such as end-to-end delay or bandwidth and service (session) specific in terms of experienced throughput as a quality of the service.
- *Clock/flow synchronization*: Some applications and services may require data streams to be synchronized, in particular in gaming scenario synchronization is important.
- *Device discovery*: Prior to any activity, it is important that relevant devices (for a potential migration) have discovered each other. This is the purpose of this component.
- *Security*: The application or service may contain sensitive and personal data within its internal states potential for migration, which should not be migrated to the wrong devices; hence security mechanisms are surely needed.

All the above functions are graphically shown in Figure 1, which includes components for the entire project.

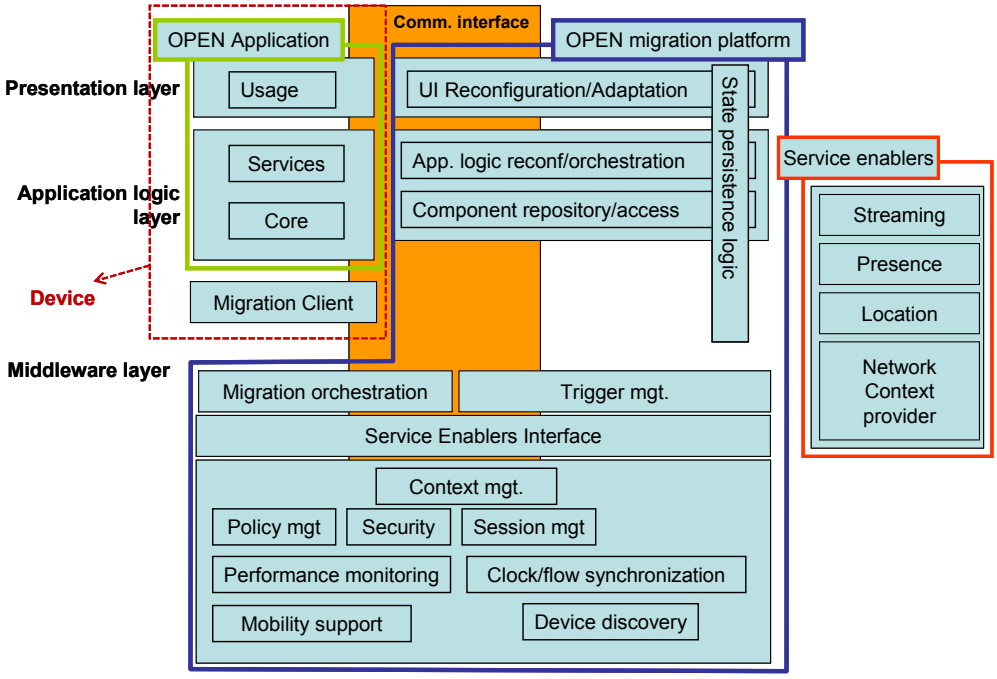

Figure 1: Illustration of components for the OPEN platform

Given the previous explanations, one can see that WP3 will focus on the lower part of the the OPEN migration platform. The upper part is mainly related to WP2 while there is some interface between functions developed in other work

packages; this is in the system manifested between these upper and lower component groups.

To recap the involvement of WP4, dealing with integration and inter communication of the system functions: it is related directly to the common communication interface marked as a background vertical block.

In the following part of the deliverable, we will focus on describing details of the different components; detailed functionality descriptions, how components interact and their respective interfaces. The purpose is to provide the understanding of the system necessary to carry out first demo concept of the system which is also Deliverable 3.2, [11]. Further work and refinements in the coming part of the project will be based on this document.

The deliverable is structured as follows: First, in Chapter 3, we introduce the core functionality required for service migration. Core functionality, includes migration orchestration to ensure a correct migration, connectivity support during the migration process and finally also context management to support a context sensitive migration process, e.g. to trigger the service migration at the right time and place.

Chapter 4 introduces the various support functions in the OPEN platform which are not necessarily directly used in the migration process, but are supporting the process in some way. Component requirements, functionalities and interactions are described in the chapter.

In Chapter 5 we provide deployment scenarios for the components and functionality shown in Figure 1, and detailed in Chapter 4 and 5. Different types of network scenarios are described, including the description of aspects to the different deployment scenarios. Finally, we focus ourselves in the year one, to a scenario which relies on a migration server (centralised) which is the focus of the following chapter.

In Chapter 6, based on the decision from previous to focus on scenarios including a migration server, we investigate and detail the component interaction in three levels; 1) an overall manner which addresses interaction with external service components (external to OPEN), 2) internal interfaces and high level interaction between OPEN components and finally 3) detailed interaction and activity diagrams of the scenario detailing the interaction of components at the message level.

In Chapter 7 we address the security part, which mainly focuses on security analysis of the components. It should be noticed that as the main focus of the project is not on security, this part will be limited to requirement analysis and how we can utilise existing off-the-shelf solutions in the migration to meet the most important requirements.

In Chapter 8 we conclude and provide an outlook of the coming project period, as well as which challenges we will address. This outlook will be focused as mentioned on the ad hoc scenario solution.

3 Core functions for migration

In this chapter an overview of the solutions is presented under development for migration and technologies that will support service migration in OPEN. The chapter focuses here on the core functions related to service migration, which are under development in WP2 for user interface migration and in WP4 (T4.1) for application logic reconfiguration.

3.1 Migration orchestration for task continuity

Migration orchestration controls the migration process – its operations start when a migration trigger is received, and finishes when the application has successfully migrated to the new device(s). As an orchestration component, this function will interact with every other module in the platform, and ensure that all their operations are carried out successfully and in the right order.

An example of migration orchestration is when migrating a user interface: then various migration services need to be accessed: the reverse engineering for building the logical description; the adaptation service; the state-mapper, which maps the state of the source user interface in the target one; the user interface generator for the target device; and the target device itself for uploading the migrated user interface.

Migration orchestration interacts with the context management function, the trigger management function, policy management and synchronization for its management functionalities. It then orchestrates the functionality of the rest of the modules in the platform, including network, user interface and application logic reconfiguration.

The migration orchestration function is relevant in all of the migration scenarios.

3.1.1 Candidate systems

3.1.1.1 Generic orchestration mechanisms

There is a lot of literature that deals with the orchestration of a sequence of events, or the concatenation of services. At the bottom of the problem, however, lies the fact that orchestration systems are central to the environment they organize, and as such, are highly specific.

3.1.1.2 Mechanisms for logic reconfiguration

In [12] some orchestration engines are analyzed as potential candidates for application logic reconfiguration, these can be also considered as solutions for OPEN migration orchestration.

3.1.1.3 UI and state migration: Scorcia

Sorcia, presented in [13] is a solution for enabling the migration of interactive applications among various devices while preserving the state and adapting the user interface to the capabilities of the target device is under development in OPEN WP2 (see [13]) and WP4 (see [12]). This is allowed by a dynamic user interface generation for different platforms (digital TV, mobile device, desktop) using transformations that are able to generate UI in different implementation languages (XHTML, XHTML+Javascript, Java for Digital TV, …). Logical descriptions of existing interactive Web applications are automatically built and are used, by means of suitable transformation, to dynamically generate user interfaces that are adapted to various types of target devices and implementation languages, including non-Web languages, with the state updated to the point at which the interaction was left off in the previous device.

The architecture behavior relies on several software modules that perform a number of activities. For example, the context management module *proactivly* performs the monitoring of the current context and, at each change, evaluates the new scenario. If better conditions are recognized, the module triggers this module alert the user the possibility for migration: this is the case of a better device (meaning that the new device has more suitable interaction resources, or more processing power and/or memory, etc.) which has become available for migration or of another one that has been freed by the previous users. The purpose of *transformational* modules is, instead, to transform the user interface description for a given platform into a new description suitable for a platform with different interaction resources. Such transformation is obtained following a number of rules indicating the most suitable transformation for each type of user interface element or structure.

While users interact with the applications available in the intelligent environment, the main migration features have to be supported; such features are: task continuity, state preservation and adaptation to interaction resources of the target device. For this purpose, the migration architecture supports a number of reverse and forward transformations for adapting existing desktop Web applications to various interaction platforms and supporting task continuity.

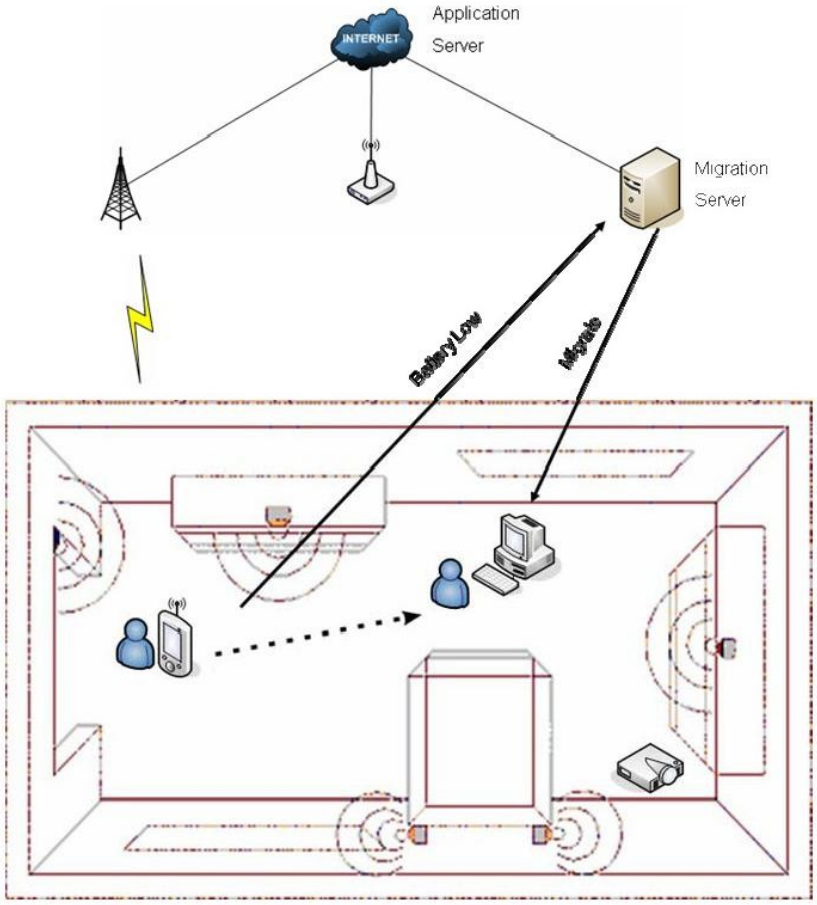

Figure 2: Example of a migration environment

The basic assumption is that there exists a large amount of easily accessible content for desktop Web applications, which can be processed and transformed to support migratory interfaces, even across non-Web implementation languages. Figure 2 shows an overview of an environment empowered with migratory interfaces. The client devices run a module, called *migration* client that subscribes to the migration service and provides information regarding the device characteristics. The devices access Web applications through the *migration server*, which includes proxy functionalities. Migration can be triggered either by the user (through the migration client) or can be automatically triggered by the smart environment when some specific event (such as very low battery or connectivity) is detected. A mixed initiative solution is also possible, in which the environment suggests possible migrations based on the devices available and the user decides whether or not to accept them. In the case of automatic identification of the migration target, there is a module in the migration infrastructure that is able to analyze the characteristics of the available devices and to identify the most suitable one according to a number of rules.

The migration platform has been designed using a service-oriented architecture and implemented using Web services. This means that the main functionalities have been encapsulated in the following modules that can communicate with the external world through XML-based interfaces:

- Device Discovery, which is the module in charge of discovering the devices currently available for migration)
- Trigger Manager, which decides when the migration has to be activated
- Reverse Engineering, which builds the logical user interface descriptions from the desktop Web implementations;
- Semantic Redesign, which transforms the logical description of the source user interface into the logical description for the target device;
- State Mapper, which associates the state of the source user interface to the logical description for the target device;
- User Interface Generator, which generates the user interface implementation for the target device.
- Migration Device Module, which runs on each device notifying its presence/availability for migration and provides information on the state of the user interface running on each device;
- Migration Manager, which orchestrates the general behavior of the system

When the user accesses the application through an interaction platform other than the desktop, the server transforms its user interface by building the corresponding logical description and using it as a starting point for creating the implementation adapted to the accessing device. In addition to interface adaptation, the environment supports task continuity. To this aim, when a request for migration to another device is triggered, the environment detects the state of the user interface, which depends on the user input that has been provided in the source device before activating the migration (e.g., elements selected, data entered) and identifies the last element accessed in the source device. A logical version of the interface for the target device is then generated, and the state detected in the source device version is associated with the target device version so that the user inputs (selections performed, data entered, etc) are not lost. Lastly, the user interface implementation for the target device is generated and activated remotely at the point corresponding to the last basic task performed in the initial device. In the process of creating an interface version suitable for a platform different from the desktop, Open uses a semantic redesign process. This part of the migration environment automatically transforms the logical description of the desktop version into the logical description for the new platform. Therefore, the goal of this transformation is to provide a description of the user interface suitable for the new platform. This means that intelligent rules are used for adapting the description of the user interface to the new platform taking into account its

capabilities (e.g.: using constructs that are suitable for the new platform) but ensuring at the same time that the original goal is maintained.

This solution allows the environment to exploit the semantic information contained in the logical description and obtain more meaningful results than transformations based only on the analysis of the specific user interface implementation languages. In this case the semantic information is related to the basic tasks that the user interface elements are expected to support.

3.1.1.4 Summary

While no single system can satisfy the specific requirements of the Open platform (or any other functionality specific platform, for that matter), there are valuable lessons to be learned from the orchestration strategies presented in these sections.

Web based applications in the Open platform will borrow from the Scorcia system, while WP4 is likely to take its roots on the systems put forward in section 3.1.1.2. The rest of the platform will be adapted to the needs of the components, bearing in mind the available orchestration patterns.

3.2 Connectivity support for migration

Mobility support is required in the middleware to handle connectivity when the user is mobile so that correspondent nodes do not need to be aware of and handle mobility. Correspondent nodes are application node(s) in remote networks. As these could be non-OPEN-aware application peers, such as regular web- and application servers which are relevant in web-service scenarios and applications, it is not feasible to expect that they can handle mobility themselves. Thus to support users moving around, e.g. from one network to another or from a fixed infrastructure network to an ad-hoc network, the functionality must be embedded in the OPEN platform and it must provide transparent mobility support to non-OPEN service providers.

Several types of mobility support are considered [2]:

- **Personal mobility** is the ability to reach a mobile user through devices currently available to the user. As this is more focused on the user, and less on the device and the services used by the user, personal mobility is not in the scope of OPEN.
- **Terminal mobility** is the ability for a mobile device moving between networks to be reachable by a correspondent node. Terminal mobility considers the entire device, and not only one application (or even just one application flow). Therefore, the mechanisms supporting terminal mobility may only be applicable for OPEN when the migration application is the only application on the device, i.e. there is a one-to-one mapping between device and application. If migration of one among several applications

from a device was migrated using terminal mobility, the remaining applications on the device would have their environment changed after migration.
An example of terminal mobility is handling IP mobility, i.e. where a device moves between LANs, the mobility support handles mapping the new IPs to the old IP (in mIP this is done using home agent entities) and masks the change from other nodes. If a device holds more applications, and only one migrates, but the IP of the device is masked into a new one, the remaining applications will loose connectivity after migration.

- **Session mobility** is the ability to maintain and continue user sessions while moving between terminals. Session mobility is very relevant to OPEN and requires additional support from protocols above the network level, e.g., the session mobility support functions in OPEN will have to employ higher level support. The handling of the application state and state persistence at the application layer of the OPEN platform will require interaction with the application level components of the OPEN platform from the system support level in order to get information about application state information.
- **Service mobility** the ability to change device and still have access to the same services. Service mobility combined with session mobility is called *service migration* as migration of services requires session continuity.

With a migration server in the architecture, the session- and service mobility should be handled at the server. Placing the migration server in the data-stream allows for complete knowledge of and control over the potential mobility of the involved devices.

The primary objective of the mobility support function is to make mobility transparent for other functions. These functions can be either within the OPEN platform or in the application.

To achieve this in the OPEN platform, the mobility support needs to interact with the functions handling application changes during migration. These functions are the migration orchestration (for control and information) and context management (for information).

To achieve mobility support towards the application, the mobility support needs to act as a façade toward correspondent nodes, hiding any mobility during migration and still allowing for continuity. In the infrastructure scenario, with a dedicated server supporting migration, much of this mobility support can occur within the server. On the network level, mobility support can be attained using simple NAT for redirection of traffic where the server will know sender and destination of each flow of data. On higher levels SIP proxy functionality or adaptors to the middleware (e.g. CORBA or OSGi) can provide redirection of flows to support service mobility.

Every scenario in D1.1 [3] contains one or more of the above mentioned mobility types and therefore this function is relevant to all scenarios.

3.2.1 Candidate systems

Overall connectivity support in a mobile environment means support for continuous and consistent addressing and message delivery. Both can be challenged by mobile users, as the sending or receiving devices may change networking context while in session.

Two types of support exist to handle such mobility in dynamic networks:

- Network-level: NAT, mobileIP, SCTP
- Application-level: SIP

These solutions are described in the following section.

3.2.1.1 NAT redirection / masquerading

Routers used to establish end-to-end connections can also provide a very basic form of mobility support. By utilizing different kinds of *network address translation* (NAT) techniques, a router in the data stream can redirect traffic from one destination to another destination. The typical use of NAT is for masquerading, to hide devices connected to a LAN behind one external IP. The NAT router then handles translation in packets to enable transparent shielding.

Redirection using NAT can be used either purely on network addresses simply mapping IP-addresses to each other; it can be used to map sockets on the external IP address to sockets on the internal network or as described above to mask entire networks. In terms of mobility the most relevant features are to use redirection on sockets and when applications (or parts thereof) migrate, redirection of traffic flows would be enforced in the NAT table of the NAT device. Other peers would then not be required to provide any additional mobility support.

However, general to NAT rules are that they are set statically on the NAT device meaning that each time a device moves, the rules have to be updated.

A current trend towards supporting more dynamic reconfiguration of NAT is through using UPnP. Then the NAT device acts as a UPnP control point, to which client UPnP devices can connect and issue changes to the NAT tables.

NAT is a solution to both terminal and flow mobility. The mapping of addresses can be used to redirect both all traffic to a mobile node or particular flows (bound to sockets).

3.2.1.2 MobileIP

MIP enables a mobile device to maintain its IP address and transport layer connections while its point of attachment to the network changes.

MIP does this by using *home agents* (HA) that are static entities in the home network of a mobile node (MN). When the MN operates in its home network, regular routing mechanisms are used to deliver messages. If the MN moves to a new network, it obtains a new IP address. The MN communicates its new address to the HA which then handles redirection of traffic to and from other nodes. This way peers connected to the MN (also called correspondent nodes (CN)) do not need to implement MIP. If they do, the rerouting step using the HA can be omitted by *route optimization* where a direct connection between CN and MN is established to begin with.

Although, MIP can be classified as a solution to terminal mobility, extensions such as filters for MIP [20] can also support service mobility. Through redirection of particular flows instead of all traffic from an MN in one network to the MN in another network, services can be accesses by and from the MN when in different network, and thus mobility of the services between networks is supported.

3.2.1.3 SCTP

At the transport level, extended versions of the *stream control transport protocol* (SCTP) can provide mobility support suitable for OPEN.

Basic SCTP enables control of multiple streams of data to destinations. The destinations are defined as IP address-port pairs (transport addresses) and possible multiple destinations are exchanged between peers during connection initialization. SCTP features transport of data to all multiple destinations simultaneously (called *associations*). One of these associations is called the *primary path* and during transmission, this primary path can be relocated if a device moves.

Through an extension called mSCTP [1] using an auto-reconfiguration protocol called *dynamic address reconfiguration* (ADDIP), the possible destinations for the *primary path* need not be fixed. When a device moves, it receives a new IP address and through automatic reconfiguration this new address can be used as a new destination.

SCTP is a solution to many of the described mobility problems, adding also support for personal mobility through the use of the multiple destinations as multi-homing and thus being able to address a user at several terminals.

3.2.1.4 SIP

The *session initiation protocol* (SIP) is a signalling protocol that enables control of multiple data streams. SIP does not handle data transport, which is typically handled by other protocols such as UDP or RTP.

SIP offers a number of logical entities, namely user agents, redirect servers, proxy servers and registrars. Through these entities SIP offers management of different kinds of mobility of user agents and data streams before and during a session. Management of user agents prior to a session is called *pre-session mobility* and is handled purely by a SIP server using static redirection. *Mid-session mobility* occurs when user agents move during a session and is handled by message exchange between user agents only, which makes mobility fast but requires support of all peers.

SIP signalling from user agents must be performed by the application itself. SIP can be used as a solution to all the described mobility types.

3.2.1.5 Summary

In summary the mobility aspects of service migration can be supported on several layers of the communication protocol stack. Some technologies require the application to be aware of potential mobile entities and provide means for handling the mobility such as in SIP, while other technologies make mobility completely transparent to the applications such as SCTP, MobileIP and NAT.

All technologies have different requirements in terms of additional logical entities, which in turn may pose requirements to the migration platform architecture. This must be considered when the specific technologies are evaluated as a part of the migration platform solution.

3.3 Context management

At each device, there is a set of information elements useful to the migration process, which could for example be the current processing capabilities at the node, position of the device, achievable data rates or other user oriented parameters as user activity or the orientation of the user, see Req. 137 in [3]. This information is not static, but changes over time, hence requires real time access if it should be used actively to adapt to situations.

A context management system generally handles information distributed in the network and provides easy access for services, application and networking components to needed information. Thus it is responsible for collecting, storing, processing and delivering relevant information from different sources of context to functions in need information, which can be either directly measurable or inferred/processed information. Context providers can cover anything from environment changes (the user is standing in front of the device) to very device specific information, such as the remaining battery life. In the OPEN context it is used for many purposes as envisioned and required in [3]. For example, it is used in the trigger management function that is capable of issuing a migration trigger based on a context change (e.g. once the battery capacity is below a certain

threshold). In addition, application logic reconfiguration, UI migration and mobility functional modules also require use of context and thus access to it.

The challenge faced by the context management system is twofold:

1. *To find out where to obtain specific information at any time*: Mobility of the user may render it impossible to get data from a specific node, making the desired information element unavailable, or the other way around, make it possible to get new types of information, or even one with better characteristics (e.g. with better accuracy).
2. *To ensure that the information is up-to-date, reliable and trustworthy*: Information elements are often dynamic, and changes over time. Hence, it is important to realise that the communication delay which will always be present when accessing remote information, may result in data becoming outdated. Furthermore, suspect entities may provide false information which could trigger migration to malicious devices, prevent migration or other types of failures in the migration process.

Many of the components in OPEN are envisioned to utilise context information to enhance their behaviour according to the given circumstances, i.e. context, hence would need to interact with the Context Management system. Examples from the OPEN communication platform include the *device discovery*, *performance monitoring* (by checking device status) the *migration manager* and *Trigger management* for triggering service migration at the right time and place. At one hand, those client applications would require information in two ways (and more generally also other client application):

- **Reactively**: Request for information on demand
- **Proactively**: Expect a notification from the Context Manager under certain circumstances, which may be either
    - *Periodical*: the application client will expect a notification at certain time intervals
    - *Event based*: the application client will expect a notification only when a certain event has occurred (like a threshold violation of the value of the context or similar)

At the other hand, a Context Manager would also need to interact with the sources of the information. This is trickier since sources provide information in all different data formats using different protocols or access methodologies, but gives the benefit that the above client applications do not need to worry about this issue.

From [3] there has been set up some requirements to the Context Management system. In the following the requirements that have a direct implication on the technical requirements to the Context Manager are described. These OPEN requirements are high-level, user-oriented requirements. The following list

contains a summary of technical requirements for the context management list derived from the selected OPEN requirements. They are specific technical requirements for the context system of OPEN and listed and related to each other in the following list. Some of the requirements are originally picked from [3] (indicated in parenthesis) while others are derived from the original ones.

- Context system shall provide information about different entities
    - Context system shall provide information about devices
        - Context system should support the process of filtering the target devices list for suitable devices
        - Context system shall provide information about the privacy level of devices (especially displays)
        - Context system shall provide information about devices in vicinity (20), hereunder their location
    - Context system shall provide information about users
        - Context system should provide access to location, direction and user preferences (137)
    - Context system should support service selection based on location and direction (22)
    - Context system shall provide traffic information of roads from a road information provider (112)
- Context system shall provide context information from various sources (136)
- Context system shall provide good accuracy on context information (89)
- Context system shall provide an API to applications (59), hereunder a continuous, asynchronous monitoring API
- Context system shall operate in a pervasive computing environment
- Context system shall handle all communication means necessary to access context information

In the following we elaborate some candidate subsystems which may offer OPEN context management a good starting point for further development and shaping into service migration scenarios.

3.3.1 Candidate systems

3.3.1.1 Universal Plug and Play

Universal Plug and Play is a set of protocols promoted by the UPnP forum, [14] and widely supported by the Windows operating system. Its objective is to allow multiple devices to discover each other and be able to exchange services and functionalities, and for such purpose, it enjoys relative success in the home networking environments.

UPnP often operates using multicast UDP packets, which provides for good discovery support on a local network, but does not extend beyond it. Several authentication protocols are supported, but none of them provides a lightweight standardized solution. On the other hand, device description is reasonably flexible, as shown in [6], where OWL-S is used to semantically describe the required inputs and outputs of a service.

Overall, while UPnP enjoys a solid installed base, but it ultimately lacks the query flexibility required for a Context Management Framework.

3.3.1.2 The Context Toolkit

The Context Toolkit presented by Dey in [7] provide abstractions and support for the management of Context Information. According to [8], it also offers middleware for building and executing context-aware applications. The Context Toolkit builds on top of a simple, distributed infrastructure that uses peer-to-peer communications. Each object in the context toolkit, widgets, discoverers and applications, are based on a BaseObject. This BaseObject encapsulates the distributed communication abilities. A widget is responsible for the binding between a specific piece of context and an application. All context widgets register at a centralized discoverer. The application can find the right widget by contacting that discoverer. An application can then subscribe to the context offered by that widget via the BaseObject. When the sensor delivers the context, the widget passes it on to the application if it matches the subscription.

3.3.1.3 PACE

As described in [8], Henricksens' work in the PACE (Pervasive Autonomic Context-aware Environments) project resulted in the PACE infrastructure. [9] This infrastructure has a Context Manager that consists of a distributed set of context repositories. These repositories provide their clients, which can be context producers or consumers, with five interfaces: query (pull), update, transaction (multiple queries), subscription (push), and metadata (discovery of the available data). A preference manager is also provided; which is a repository for user preference information coupled with the application states and context information. On top of the preference and the context manager is the programming toolkit, which implements a simple conceptual model for formulating and carrying out context based choices. This toolkit makes the process of discovering and communicating with the management systems transparent to the applications.

3.3.1.4 PERSONA Project

The PERSONA framework for Context-Awareness builds on the PERSONA middleware. Said middleware is designed as a solution for facilitating interoperability within distributed systems, in which each node is able to talk to other nodes directly, abstracting the connectivity layer and providing common functionalities to all the applications that use the middleware. It is built using Java and the OSGi framework to provide functionalities for easy deployment of the components of choice in each node.

The middleware is composed by three different layers, from the bottom: the abstract connection layer (ACL), handling p2p interconnectivity, over UPnP, R-OSGi or Bluetooth, The Sodapop[1] Layer (SPL) introducing the concept of buses (either push or pull) and peers and serialization, and the PERSONA-specific layer (PSL), which implements the input, output, context, and service buses .

The implementations supporting Context-Awareness are part of the PSL, and define the context bus and the context specific anthologies used both on the context bus and by the rest of the system. A high level diagram of the architecture is shown on Figure 3.

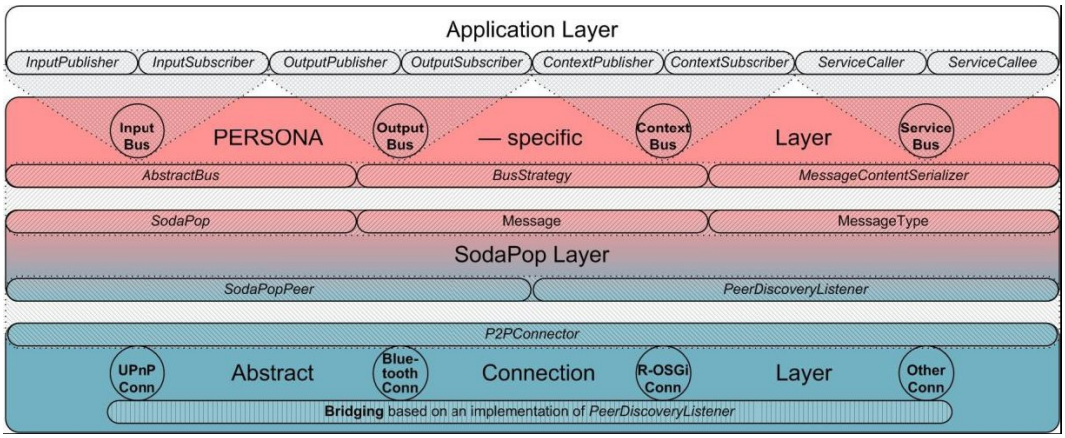

Figure 3: The three different layers of the PERSONA middleware

3.3.1.5 MAGNET Beyond (Secure Context Management Framework)

[1] SodaPop: a software infrastructure supporting self-organization in intelligent environments; M. Hellenschmidt, T. Kirste - Fraunhofer Inst. for Comput. Graphics, Darmstadt;

The framework from MAGNET Beyond is called the Secure Context Management Framework (SCMF) and is presented in [15], [18], [19] and [16].

In MAGNET Beyond the 'Secure context management framework' (SCMF) [16] has been developed for the very purpose of management and provisioning of context information in a secure manner. The basis assumption of the framework is that all nodes involved have a so-called Context Agent running, These are capable, in a distributed manner, of collecting, gathering and distributing context information to each other. The framework we elaborate on in the following section.

Figure 4 shows the internal functional blocks of a Context Agent. So the main functionality provided by the framework is divided into different functional blocks. The first and one of the most fundamental is the Data Source Abstraction Manager (DSAM). This block is responsible for collecting node local context information, such as from built-in sensors on the node, current system status (current memory usage or processing capabilities) or other specific information, which effectively creates a data source abstraction layer which abstracts raw data sources and their individual data representations, into a common data model and representation. The key point with this block is that it allows any arbitrary context information into the framework through specifically written software components, called *retrievers*. The retrievers are creating the interface between the framework and any arbitrary data source that may be of interest.

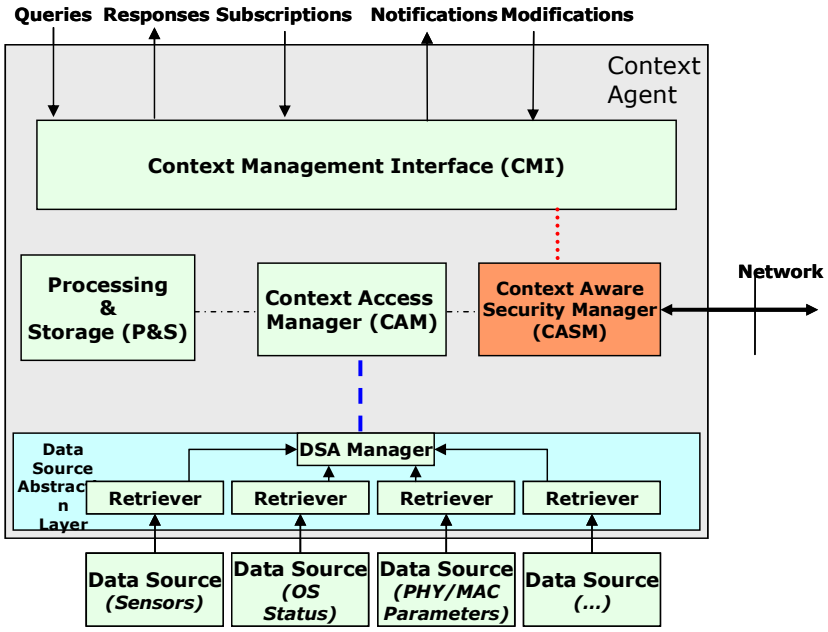

Figure 4: Internal structure of Context Agent (see e.g. [16], [19]). Each device in a network is assumed to have a Context Agent running.

The retrievers take the raw input, meaning the input as provided by e.g. a sensor, the OS, file etc., and map it into the specifically used data format specified in [15].

This allows later on very generic searches on a highly diverse type of information, i.e. exactly what is required for context management.

Similar to the DSAM, the Processing and Storage Unit (P&S module) allows for further processing of context information in a similar way. The major difference is that the information provided by the P&S module, is not directly measured information, but will typically be information inferred, derived or estimated by some algorithm which uses measured data from the DSAM (either locally or externally to the node). This module works in the same way, hence allows for easy add on of new types of context information. Furthermore, this module provides a storage space of e.g. user profiles, and allows easy access to user profiles in a distributed environment similar to other context information.

At the very heart of things, the Context Access Manager (CAM) is responsible of creating an index of the whereabouts of different context information elements in the network. This means, by starting from creating a list of locally available context information (both retrieved, processed and stored information) this is then registered at the CMN which then later on can respond to queries on the location of given context information.

To ensure network communication and the security related to accessing context information, the Context Aware Security Manager is ensuring access control to the information. Context information is potentially very personal to the user, and should only be distributed to trusted entities. Thus, this module would either not allow access to requested context information, or it may provide obfuscated information, e.g. providing the location of the user but with much less accuracy than is actually available, or instead of a GPS position, providing the name of the city that the user is currently in. In this block, all communication functionality such as serialization of internal data objects, registration messages etc., are carried out as well.

Finally, the Context Management Interface (CMI) is a module which accepts and provides notifications to the client application in a dedicated XML based query language. This language is called Context Access Language (CALA) and allows for flexible types of queries, see [15]. This CALA language is based on an information model specified within the MAGNET Beyond framework, hence terminology and definitions will need to be based on this too. In particular the main entity from which everything is based is called a *Context Entity*. This ensures another abstraction level of context description, which effectively defines the interface towards client applications of the framework.

Additionally to this, a context enabler based on IP-based Multimedia Subsystem (IMS) called ICE has been developed in the IST-Spice project which provides a layer atop the Magnet CMF that enables cross-domain communication using the SIP protocol and the IMS infrastructure.

3.3.1.6 Summary

Assessing the requirements to the Context management framework for OPEN, and mapping those to the existing solutions, we find that the SCMF developed in the MAGNET Beyond projects holds the best potential for being the basis for a context management framework in OPEN.

In particular, one of the strong points with the SCMF is the extensibility of the framework, i.e. if a new source is becoming available, a developer in OPEN would simply write a retriever or processing component that maps the information from the new source into the SCMF data format, and this information becomes now automatically available to all interconnected Context Agents. These facts matches quite well our requirements in OPEN, and considering the fact that the implementation is available to the consortium through partners previously in the MAGNET Beyond project, makes the framework a highly interesting candidate to continue working with.

The downside of the framework so far, is that this has to be manually setup up and reconfigured via xml files, and the whole life-cycle of components is not really covered, hence requires a lot of user interaction at an undesirable level. Therefore it would be useful to investigate the possibilities for extending the SCMF to work in such frameworks, which we elaborate in the following. Furthermore, the security part of the SCMF from MAGNET Beyond cannot directly be applied in OPEN context particular because it is based upon the assumption of working in a secure and trusted network environment, whereas in OPEN these assumptions does not hold.

4 Support functions for migration

The functions required to support service migration are located in a common middleware layer. In the following chapter, we focus on the functionalities which are supporting the migration process in some way. The context of each function is derived from the overall architecture presented below. The architectural diagram lets the user assess the layer placement and cross interaction of the multiple components.

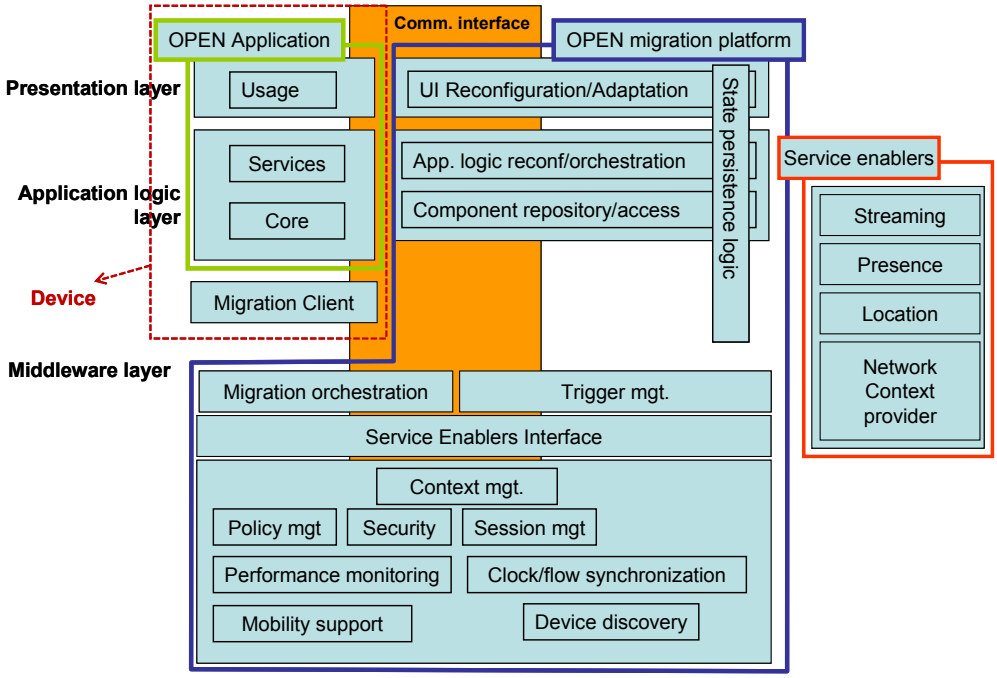

Figure 5: OPEN software architecture

The functions shown in lower part of Figure 5 are described in the following sections.

4.1 Trigger management

4.1.1 Functionality

The Trigger Management module is responsible for prompting applications to start a migration process. In a sense, this module behaves as a classifier that chooses when the application state should be changed.

Such a classifier takes its decisions based on the application semantically described characteristics, the situation of the user, and the situation of the devices involved in the migration. In most circumstances, said information can be derived

from Context sensors, using the Context Management Framework, and from the knowledge derived from the lower network layers.

In regard to the algorithm used to trigger the migration based on the situational information, a modular architecture is proposed, which would allow for the instalment of multiple reasoners of arbitrary complexity, ranging from simply rule sets to more complex classifiers.

4.1.2 Interactions

This module primarily interacts with the migration control function or higher layer reconfiguration functions, such as application logic reconfiguration. The interaction is in effect bidirectional, in that while trigger management can alert these modules when a migration is needed, the peer modules may also program their triggers into this component; for instance for the Application Logic Reconfiguration may add required triggers during runtime based on the rules defined for reconfiguration.

One example for a reconfiguration rule (informally) could be: "connect required services of component A with nearest services". Out of such a rule the reconfiguration manager will register new triggers at the trigger management which could informally look like: "trigger me, if new device available within 20m". Thus, the definition of the concept “nearest” will be the responsibility of the reconfiguration manager. Further method calls to the context manager will be used if needed to query further information about the current situation. So, for the first version of the prototype, basic triggers should be adequate in order to realize reconfiguration functionality.

4.1.3 Requirements/Scenarios

This component references to Requirement no 89, 13, 82, 86 and 37 in [3], and relates to all migration scenarios.

4.1.4 Candidate systems

For simple context triggers, a context management system candidate could provide these. For more complex triggers, systems for collecting relevant information and inferring context or migration triggers could context reasoning systems based on ontologies or frameworks of Bayesian Networks.

4.2 Policy/profile management

4.2.1 Functionality

The policy management function is responsible for deciding if migration is allowed or forbidden, e.g., because it violates restrictions defined by users, service providers or a 3rd party. The function is based on contextual calculations,

functionality requirements, application context and user profile settings (e.g. security and privacy permissions). A migration policy can contain information relevant to allowing or denying migration of an application or parts of it.

The management part of the function concerns the ability to provide storage and access to the different policies used during migration. This could be a user profile or an application migration policy. Note, that these policies affect the migratory aspects of the platform and not the internal configurations/policies specific to the applications. These are expected to be handled internally in the applications supporting migration.

In the centralized architecture, the policies are effectively stored on the migration server, and updated appropriately when the user makes updates. An update could be to allow for a new user group to migrate onto the user's devices.

4.2.2 Interactions

As a part of generating a migration trigger the policy management function is consulted to accept or deny migration. If migration is allowed, policies can specify further restrictions like the allowed destination devices and services to migrate. This way policy management affects both trigger management and migration orchestration.

4.2.3 Requirements/Scenarios

In requirements 49, 52, 66, 98 and 142, scenarios are referred to as demanding policy support and management. For instance, if the user does not want the migration to execute automatically in a certain situation, this may be specified in a policy. This way unintended migration will occur, and the conditions governing this restriction will be controlled by the user.

As a result, some kind of user interface is required for this function to enable the user to make updates to the policies. This should preferably be placed as close to the user as possible, as such updates may be decided upon rather fast and thus access to this functionality (for using them during migration) should not be a complex or cumbersome effort.

4.2.4 Candidate systems

A simple database located in the central migration server can provide the storage and access (in the infrastructure scenario) for the trigger management and migration orchestration functions. XACML is a mark up language for describing policies which can be used. In the same time the SCMF offers storage capability which may be used, but requires then the information to be specified accordingly, which means some adaptation between XACML and SCMF descriptions may be needed.

4.3 Session management

4.3.1 Functionality

Ongoing networking, security or application sessions must be respected when performing migration to provide continuous services.

Examples of such sessions are

- application session: logged in on website, a state full server (http session and corresponding timeout, web service), service/device registration (e.g. from UPnP)
- security session: established security association (authentication, authorization, credential/key exchange)
- network session: NAT connections, state full firewalls, TCP connections, multicast group assignment

The session management function helps ensure that such sessions can continue during and after migration.

The timing in this session notion is controlled by the migration orchestration function and the internals needed to provide sessions and make them secure is handled be the session management.

4.3.2 Interactions

This function primarily is triggered by migration orchestration function and then interfaces to the application to extract the state of the application in order to transfer it during migration.

4.3.3 Requirements/Scenarios

The component refers to Requirement no 118, 119, 126, 43, 77, 45, 54 and 64, and is required for all scenarios.

4.3.4 Candidate systems

This can be handled by using existing session signalling frameworks such as SIP.

4.4 Performance monitoring

4.4.1 Functionality

Performance can be related to the overall migration and to the application before and after the migration, so these are the two axes to monitor as illustrated in Figure 6.

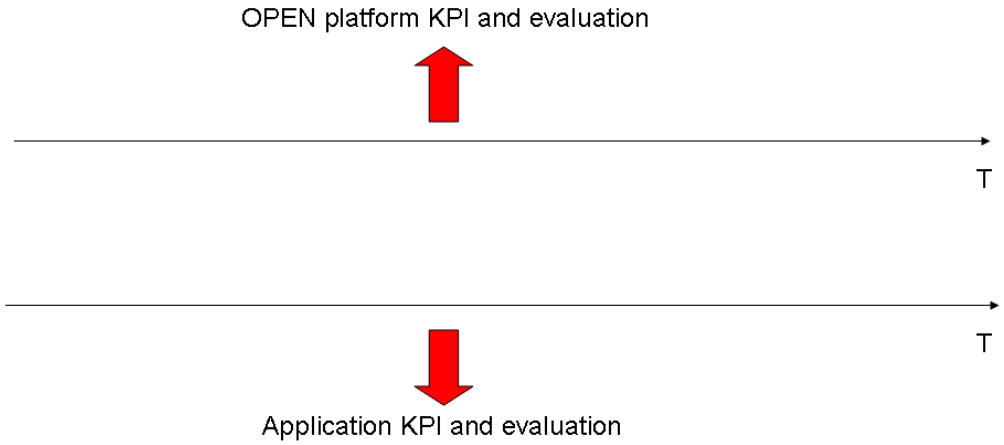

Figure 6: Axes to monitor during performance monitoring

Timing is the key element to realize performance monitoring; it can be evaluated along the usual sequence of events before, during and after a migration, in order to underline the contributions of all the other functional elements.

In order to reach a more complete approach, timing has to be coupled with other indicators, such as failure percentages and application performance indicators.

Figure 7 presents an example of what could be inserted in the log for performance monitoring, thus separating the different impacts on it.

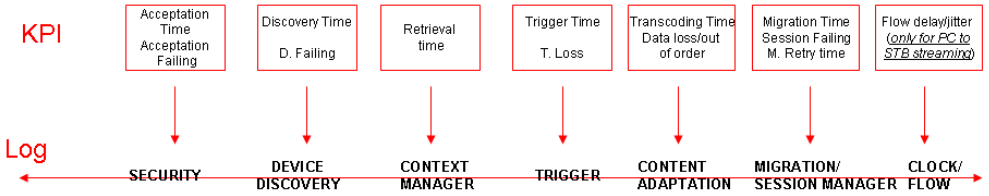

Figure 7: Examples of performance indicators to monitor during migration

4.4.2 Interactions

As it can be seen from Figure 7 the Performance Monitor needs to ensure logs from several components are present; device discovery, context manager, service migration trigger, content adaptation, migration orchestration and clock flow.

4.4.3 Requirements/Scenarios

The functionality is not directly mentioned in the scenarios, but is required for later system performance evaluation.

4.4.4 Candidate systems

As it is mentioned, performance metrics are based on information from different components. There are several standard log mechanisms existing depending on which implementation language is being used, for instance *log4j*. For now, we do

not fix ourselves to a specific system to use. The format of the specific logging and the methods to process the log data are described in D6.4 [28].

4.5 Clock/flow synchronization

4.5.1 Functionality

Different streams, existing or new, need to be synchronized in order to provide continuous service when migrating/distributing an application. An example is enhancing an audio call with video capabilities. When establishing the video stream, this should be synchronized to the audio stream already running.

To apply such synchronization some sort of clock synchronization must be in effect. Also, if application state is dependent on time, synchronized time between involved devices must be present in the migration process.

4.5.2 Interactions

The synchronization of clocks does not by itself interact with other components, but is seen as a system to maintain a general assumption of the other function, namely that the time on all nodes can be considered synchronized.

Flow synchronization will interact with the migration orchestration function in the sense that migration orchestration decides when to start, stop and pause components in order for the migration to succeed. When data is flowing between these components, the flow synchronization component will need to know the state of the component, i.e. when to buffer data and when to forward and synchronize flows.

Also if flows are part of a session in an application and this application is distributed from one device to several devices the session management component will notify the flow synchronization component to handle this new situation. Also if a session is governed by requirement to flow-specific parameters (such as bound upper delay between audio and video), the flow synchronization component will need to know these parameters in the first place, and also be notified of potential changes during migration, again especially if the session is either distributed or aggregated between devices.

4.5.3 Requirements/Scenarios

The component refers to Requirement no 145 and 58, and is limited to scenarios which include multiple data streams.

4.5.4 Candidate systems

Network Time Protocol (NTP) can enable timely synchronization between nodes within the OPEN platform. They can use either the migration server as main time

server or an external one. NTP supports different levels of servers and both internal and external server can be used simultaneously – if available.

4.6 Device discovery

4.6.1 Functionality

The main aspect in migrating services and applications is enriching user-experience. This is done by utilizing additional resources and/or devices available. To use these additional entities, their presence needs to be discovered and their capabilities established. This is handled by the device discovery function.

Discovery can be performed in several manners; locally (short-range communication) or centrally (using a commonly accessible registry). The objectives for discovery are also multiple:

- device discovery (network presence)
- service discovery (what services are provided by a device)
- resource discovery (which resources, e.g. battery lifetime, processing power, storage capabilities, etc. are offered by a device).

Regarding the communications with other modules, Device discovery can interact with the Context Management module in the following way: whenever there is a change of device, the Device discovery module provides such information to the Context Management module, which is in charge of updating the information about the current context.

4.6.2 Discovery model

In the OPEN project the purpose of the device discovery function is identifying the devices that are available to be involved in the migration process and the device attributes that can be relevant for the migration.

We do not consider device discovery aimed to form a network, such as, for example, an ad hoc network. Rather we assume that a network is already in place to connect the devices, when these try to discover candidate devices for the migration.

A list of device attributes can be the following:

- **Execution environment**, specifying which type of executable or interpreted code is supported. Examples are: J2ME, J2SE, Microsoft Windows XP, Linux, etc.
- **Supported Connectivity**: i.e. GSM; GPRS, UMTS, WiFi, Bluetooth, NFC, ZigBee, etc.
- **Device name**, a unique device identifier

- **Battery,** specifying the battery level, fully charged battery and plugged-in devices could be treated in the same way
- **Device location:** the granularity should be in terms of meters in order to be able to determine a "surrounding devices area" where migration could make sense
- **Privacy**, specifying whether the device is private or public
- **Interaction resources**, in terms of input and output resources (keyboard, mouse, remote control, voice, etc.) and related capabilities.

Device discovery con be performed in two ways:

- **Distributed**: a sort of peer to peer device discovery is performed from the device to the surroundings devices. The discovered device has to be able to confirm its availability to receive and execute the migrated application from the migration server.

  This approach appears to best fit with the purpose of deploying an ad hoc network than the OPEN purpose to select a migration device. The candidate device for the migration must however be reachable by a centralized component, the migration manager, which sends the migrating application. Moreover, this approach implies a potentially heavier traffic and a more complex migration node within the device.
- **Centralized**: every device provides its characteristics to a centralized registry. When there is the need to discover a new device, the device in use asks the registry for a list of devices to which the application can migrate. The migration manager selects the best candidate device for the migration and verifies if the it is still available for a migration.
  This approach can reduce the network traffic and simplify the migration node within the device.

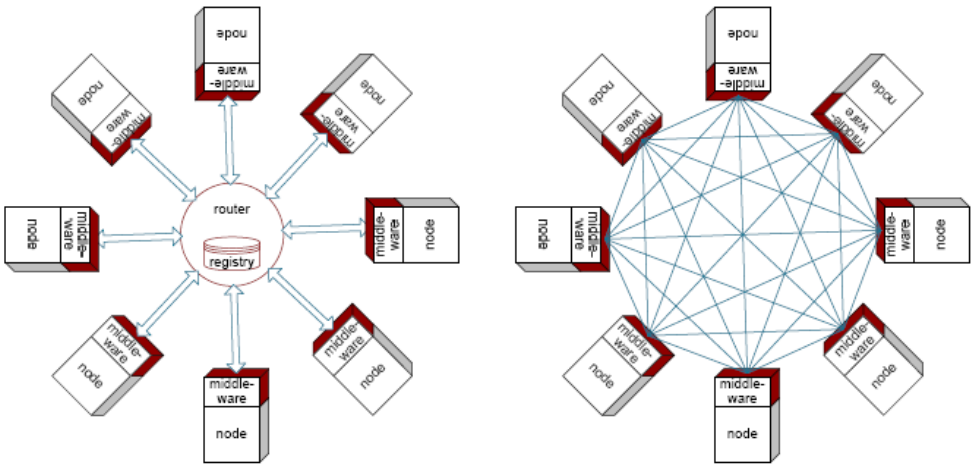

Figure 8: Device discovery approaches: centralized (left), peer to peer (right)

4.6.3 Interactions

The device discovery function can be a self-contained function interacting directly with trigger management and migration control, or it can be a provider for the context management function and discoveries will be an output of context management instead.

4.6.4 Requirements/Scenarios

The component relates to scenarios where changes in device number, service availability or resources are included, and refers to Requirement no 20 and 33 in [3].

4.6.5 Candidate systems

Given the goals of OPEN, device discovery overlaps somewhat with service discovery and device management, hence we have enlarged our analysis to a wider set of protocols. Moreover the candidate protocols must cover the whole set of devices considered in OPEN scenarios, comprising: mobile phone, PDA, PC, Set-top-box, game console, smart display.

Hereafter we summarize their main characteristics in order to evaluate them, the list is the following:

- **Technical Report 069**
- **Bluetooth**
- **OMA Device Management**
- **OSGi**

4.6.5.1 Technical Report 069

TR-069 (short for Technical Report 069) is a Broadband Forum technical specification entitled CPE (Customer Premises Equipment) WAN Management Protocol (CWMP). It defines an application layer protocol for remote management of end-user devices such as modems, routers, gateways, Set-top boxes (STB) and VoIP-phones.

As a bidirectional SOAP/HTTP based protocol it provides the communication between CPE and Auto Configuration Servers (ACS). It includes both a safe auto configuration and the control of other CPE management functions within an integrated framework.

Using TR-069 the terminals can contact the Auto Configuration Servers (ACS) and establish their configuration automatically. Similarly, other service functions can be provided. We propose that through these functions the basic OPEN

components, the Node Manager for instance, are downloaded, installed and executed in the CPE.

TR-069 is the current standard for activation of terminals in the DSL (Digital Subscriber Line) broadband market. Other fora, such as Home Gateway Initiative (HGI) and Digital Video Broadcast (DVB), are endorsing CWMP as the protocol for remote management of home network devices (e.g. the HGI gateway) and terminals (e.g. the DVB IPTV STB).

4.6.5.2 Bluetooth

It is a wireless protocol utilizing short-range communications technology facilitating data transmission over short distances from fixed and mobile devices, creating wireless personal area networks (PANs). Bluetooth is defined as a layered protocol architecture consisting of core protocols, cable replacement protocols, telephony control protocols, and adopted protocols.

Among the mandatory protocols SDP (Service Discovery Protocol) is used to allow devices to mutually discover what services each device supports, and what parameters to use to connect to them.

Since this is a short range communication technology, not enabling the connection with central OPEN components involved for migration, we would not consider Bluetooth for device discovery purposes.

Bluetooth can be instead used for device connectivity during application execution as for example in the gaming scenario where a mobile device acts as a remote control and uses Bluetooth to connect with a set-top-box for sending commands.

4.6.5.3 OMA Device Management

It is a device management protocol specified by the Open Mobile Alliance (OMA) Device Management Working Group and the Data Synchronization (DS) Working Group.

OMA Device Management specification is designed for management of mobile devices in order to support the following typical uses:

- Provisioning – Configuration of the device (including first time use), enabling and disabling features
- Configuration of Device – Allow changes to settings and parameters of the device

- Software Upgrades – Provide for new software and/or bug fixes to be loaded on the device, including applications and system software.
- Fault Management – Report errors from the device, query about status of device

Although OMA DM specifies a way to install applications and system software, its implementations do not support this functions, so we propose to assume that basic OPEN components, the OPEN Node Manager, are already installed and running in the device.

4.6.5.4 OSGi

The **OSGi Alliance** (formerly known as the Open Services Gateway initiative) is an open standards organization that has specified a Java-based service platform, which is remotely manageable. The core of OSGi specifications is a framework that defines an application life cycle management model, a service registry, an execution environment and modules.

The Framework implementations provide APIs enabling applications or components (coming in the form of bundles for deployment) to be remotely installed, started, stopped, updated and uninstalled without requiring a reboot; The service registry allows bundles to detect the addition of new services, or the removal of services, and adapt accordingly.

OSGi specifications are organized in layers covering very well the OPEN purposes and scenarios:

- L1: Modules, which realize the concept of modules (bundles) and controls their dependencies.
- L2: Life Circle Management, handling the life cycle of a bundle without the need to restart the VM.
- L3: Service Registry, providing the cooperation model for the bundles.

(For completeness we mention layer L0, Execution Environment, which specifies, the Java environment: J2SE, CDC, CLDC, MIDP, etc.).

4.6.5.5 Other discovery protocols

Other discovery protocols that we have considered not in scope for OPEN are the following ones:

- **Service Location Protocol (SLP, srvloc)** is a service discovery protocol that allows computers and other devices to find services in a local area network without prior configuration. It is frequently used by LAN-enabled printers.

- **Simple Service Discovery Protocol** (**SSDP**) is an expired IETF Internet draft. SSDP is the basis of the discovery protocol of UPnP (Universal plug-and-play).
- **Universal Plug and Play** (**UPnP**) The UPnP architecture offers pervasive P2P network connectivity of PCs, intelligent appliances, and wireless devices. The UPnP Device Architecture specification provides the protocols for a P2P network. It is a set of computer protocols which allow devices to connect seamlessly and to simplify the implementation of networks for simplified installation of computer components.
- **Digital Living Network Alliance (DLNA)** is an international, cross-industry collaboration of consumer electronics, computing industry and mobile device companies. DLNA is aimed to enable sharing of digital content such as photos, music, and videos, through consumer electronics (CE), personal computers (PCs), and mobile devices in and beyond the home.

4.7 Service enablers

This section covers those functionalities which, while being necessary for migration, do not fit into the core or supporting functions. Rather, these are assumed to be services and interfaces provided by the surrounding execution environments. Note that some of these might accessed directly by the Migration Platform, while others might be wrapped by some of its modules, such as Location being offered through the Context Management module.

Based on the D1.1 scenarios, the service enablers involved in the migration are:

- Presence,
- Location,
- Access layer,
- Device provisioning
- Streaming

Others service enablers that can be involved to provide a given service, SMS, chat and similar services, for example, are not actively involved or affected by the migration.

The respective roles of the involved service enablers are summarized here.

4.7.1 Presence server

It provides presence information related to the user. For the scenarios considered the significant presence states are:

- The user is watching an IP based Television (IPTV)
- The user is playing (gaming scenario)

- The user is offline
- The user is chatting

The presence server notifies the presence status as context information in accordance with the context management interfaces and notification rules. The presence information is also used by the gaming, streaming and chat server for the respective services. Detailed presence information are not managed by the presence server but by the server actually involved in the delivered service, this means that:

- the specific IPTV channel watched by the user is known by the IPTV (streaming server)
- the detailed game parameters, (e.g. as x,y coordinates, laps to finish) are known by the gaming server
- information, such as the chat room, is known by the chat server.

4.7.2 Location

As the migration could in certain cases need a certain amount of time, for example when a new client has to be installed and launched in the destination device, it could be useful to start the migration preparation before it is requested by the user or when it is mandated by the context (when the battery goes under an attention level but it is still above an alarm level). A location information provider could be used to provide this kind of information, notifying the context manager when the user enters certain areas where potentially a migration can occur. The granularity of such localization could be the following:

- the user is at home,
- the user is in the office
- the user is close to a certain Point Of Interest (POI)

The notification from the location enabler to the context manager follows context management rules and interfaces. It has to be evaluated if the location information is managed through a centralized location server, interacting with the cellular network, or if we prefer to use a distributed architecture where the mobile devices are responsible to provide their location. A centralized solution can be potentially tricky to integrate, but it can concentrate in one element the information regarding user position for both mobile and fixed situations; moreover it avoids the need to deploy and manage localization agents on the devices. A distributed approach is simpler to integrate and provides potentially better localization but mandates the configuration on the device of the location parameter to identify the home area. The architecture selected has to take in consideration also scenarios where it is significant to know that the user is close to a certain POI that cannot be a priori know, such as the pub for the gaming scenario.

4.7.3 Access Layer

The Access Layer can be used to provide context information, in terms of:

- Application Port Number (APN) and hence type of data session, GPRS or UMTS
- device user agent which can be used to determine the phone model and their characteristics.

Access layer cannot provide information regarding a voice session.

4.7.4 Device Provisioning

This enabler is used to send a configuration SMS to a mobile device. The configurations available are limited to a set of the several apn's used by the phone for data connectivity such as the web apn, the MMS apn and so on. Besides this kind of low level device provisioning there is a need for an OPEN platform component responsible to deliver, install, activate and manage a new application on a mobile involved as a migration destination, when this device is lacking the application. The application provisioning mentioned before should be server side initiated without the user being involved, or involved only to provide confirmation for security reasons. So far this kind of delivery is part of widget dashboard designs from some vendors such as the Nokia and Opera browsers. We are not aware of available prototypes.

4.7.5 Streaming

In case the destination device has limited CPU power and is not able to perform the effective 3D rendering involved in the gaming-IPTV game, a streaming server is used to stream the live content to the destination device, e.g., the STB. The content stream is generated by the streaming server according to the 3D model and game parameters provided by the gaming server.

4.8 Summary

Many functions are needed to support migration. Different functions come into play in different phases of the migration and some are specific to a scenario. Chapter 3 and 4 captured both core functionalities but also supporting functionalities required for an effective and reliable service migration platform. It is the purpose that this platform will support the application logic reconfigurability and other such functionality developed in WP2 and WP4 to enable the end goal, service migration of different kinds as envisioned in [4].

In the next chapter we take a look at deployment scenarios of the described function, and how components may be distributed in different network scenarios.

5 Deployment scenarios

The scenarios presented in [3] and [4] present very different settings in terms of involved devices and network architecture. In this chapter, we describe possible network architectures in terms of involved entities and also assumptions regarding the providers of the architecture. Following the architecture descriptions, we argue to focus first on scenarios where changes to the infrastructure can done to support migration and then move afterwards to more restricted scenarios regarding support.

5.1 Potential architectures and challenges

In this section we discuss various models for the network architecture based on different usage scenarios. The aim of the section is to analyse how different existing technologies can be applied for the migration support, for which the analysis will include discussion on involved network elements, their relationships, security, and configuration of the system.

5.1.1 General Infrastructure

The general infrastructure case assumes a networking setup in which application servers are provided to clients. As this is a general case, there cannot be any assumptions on special network setups and network support for the system. This case therefore is the most general case for Internet-based services. Cases for Enterprise scenarios, Cellular Operators, or Special Networks (like Home Networks) can rely on better infrastructure support for migration. Figure 9 depicts the setup.

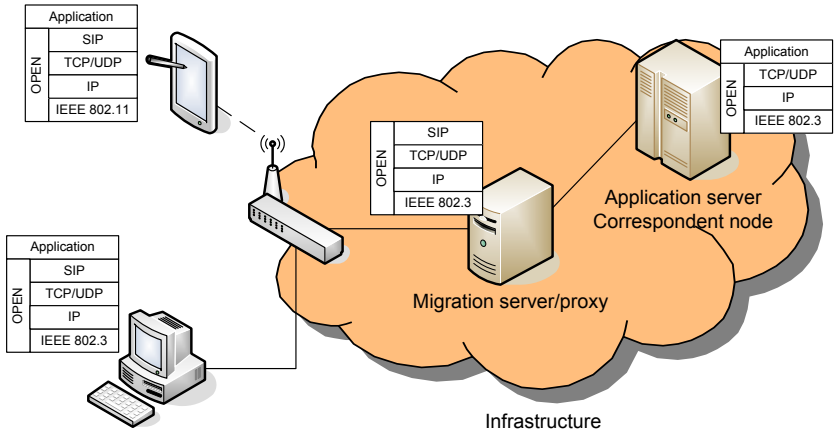

Figure 9: A scenario utilizing a proxy in the network architecture for migration

Description of Elements:

Client	**Clients execute the applications. Clients and applications executed on the clients can be Migration Platform-aware or not aware.**
Migration Server/ Proxy	**A dedicated network element that supports the migration between clients.**
Application Server	**Any type of application server that provides services to client machines.**

Relationship between Elements

As a consequence of the general infrastructure assumption, client machines need to be configured by hand for using the migration server/proxy when accessing the applications servers. Migration server/proxies can be provided independent of the applications server as long as the client-proxy trust relationship allows to use client credentials (e.g. login data) so that the migration server can sniff the needed security information. When the migration server and the application server is in one administrative domain, they can share the security relationship.

Security

Security is provided by the general means of the used communication system. This is usually a one-to-one relationship between the client and the application server.

Variant of this scenario: Infrastructure gateways

This variant assumes that the migration proxy is contained in dedicated network elements, e.g. in routers (using for example embedded operating systems like openWRT [10] as execution platform) or in dedicated entities, e.g. a proxy that intercepts service sessions.

5.1.2 Network Operator / Cellular networks

This scenario is characterized by a network operator that is involved in the service delivery. Security is provided between the client and the service provider. A distributed security and identity management system can be used for multiple application servers. Furthermore, traditionally in this case, operators like to be able to configure the end-users terminal for the given network. As a consequence, the general Infrastructure model is missing some important properties for the network operator, which may affect the configuration of the system.

Terminal and service-specific configuration can be done in various ways. For example, in recent years, it has become popular to send Over-the-Air configuration information to terminals, e.g. for e-mail access, for network configuration, or for WAP access. A drawback of this solution is that users need

to actively request updated information and network operators cannot easily change the configuration of their networks. Especially in roaming cases, this causes also a lot of management overhead to serve the visiting nodes

Next Generation Networks (NGN) is the initiative of moving existing telecommunication networks to IP based technologies. NGN rely on a SIP-based signalling infrastructure, the IP Multimedia Subsystem (IMS). The main characteristic of IMS is that terminals dynamically find their first SIP proxy (the P-SCSF) using standardized IP means like DNS lookups or DHCP. From there on, the network operators can handle network and services access on a session-by-session base.

For OPEN, this has the advantage that the needed migration server/proxy can be dynamically and network-controlled assigned based on the current network structure and services need. Furthermore, SIP enables dynamically negotiate and re-negotiate session parameters. This has the advantage that migrations can be controlled by standardized SIP mechanisms. SIP/IMS has a strong security model that seamlessly integrates with the security mechanisms of today's cellular networks.

An SIP/IMS-based migration platform would re-use the existing infrastructure like P/I/S-CSCF and HSS servers. Furthermore, this would enable also to migrate SIP-based session (e.g. IM, Voice, Presence, etc).

Compared to the General Infrastructure Mode, the SIP-based approach has also drawbacks. SIP-based communication is inherently asynchronous and event-driven. For application programmers this programming model is harder to master. Furthermore, changing all existing systems – especial Web-based applications - to a SIP-based model is not practical.

Nevertheless, for important services like voice, SIP-based session management is in wide use. The VCC standard (Voice Call Continuity) defines how voice session can migrate between, e.g. the PS (packet switched) and CS (circuit switched) domain.

Variants of the Model

SIP-based device and service configuration: A variant of this model is to use a SIP-based approach for initial device and service configuration. This means that operators can use the SIP-mechanism for dynamically changing configurations (like the used Migration server/proxy). In the end-terminal, the respective SIP signals will be mapped to configuration changes of, e.g. the browser.

OMA-based Device Management: The OMA DM standard defines ways to dynamically configure a device.

5.1.3 Enterprise Networks

Enterprise networks are dominated by Network-OS, e.g. based on Windows servers or on Unix (NFS/ONC) servers. Security is provided by a centralized user

management system, sometimes based on directory-servers like Active Directory/ LDAP

In this scenario, configuration and management of a migration platform can be done with the existing mechanisms, e.g. a centralized LDAP directory or network-OS specific mechanisms like Policy Management in Windows.

Increasingly, enterprises are also using SIP servers for their telecommunication services. An alternative for the enterprise scenario is to base the migration platform on the existing SIP infrastructure. Again, the problem remains that a majority of existing applications rely on the Web platform or on proprietary mechanisms.

5.1.4 Home Networks

Today's, home networks have no commonly agreed infrastructure. Nevertheless, DLNA/UPnP is an emerging standard that allow dynamic collaboration between home devices based on device/service discovery and a set of standardized interaction protocols.

Security: In home networks, security and identity management is usually not used.

Discovery: DLNA/UPnP can be used for dynamical discovery of a migration server/proxy at home.

5.1.5 Ad-hoc/peer-to-peer networks

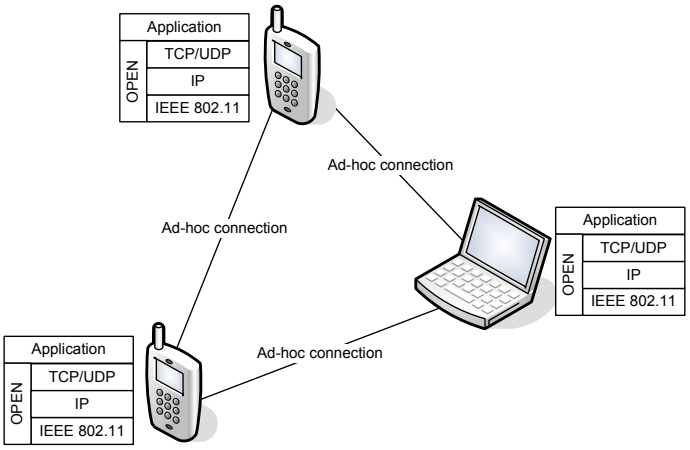

Figure 10: An ad-hoc network illustrating capabilities of involved nodes

Contrary to the previously described cases, ad-hoc networks do not assume migration support from the network – i.e. migration servers or proxies. Therefore, all functionality required to perform migration must be located on the involved

devices alone. Also, in this network the provider of the application is one of the involved devices, and not a central dedicated server.

Typical devices involved in ad-hoc networks are smaller, highly mobile devices such as smart phones, PDAs or laptops. More static devices that connect to mobile devices can also be involved such as a TV, a set-top-box or a gaming console.

In Figure 10 an example of an ad-hoc network is illustrated. A typical limitation is that entities communicate directly (often with wireless technologies) with each other and do not use intermediate communication entities such as access points or routers. In terms of scalability such networks therefore rarely become huge as the communication technologies for such direct communication do not support a high amount of nodes.

To create ad-hoc networks either Bluetooth (IEEE 802.15) or the WLAN (IEEE 802.11) family can be used as communication technologies. However, both technologies require manual interaction in order to setup the ad-hoc network. For instance when using WLAN, the common frequency must be preset on all devices.

5.2 Summary and development plan

The plan for WP3 is to start out by investigating the challenges of performing migration in an infrastructure-supported scenario, where entities to aid migration are present and usable to place middleware functions on. As a solution for infrastructure migration is developed, the challenges in performing migration in distributed and ad-hoc networks are investigated. These are for instance where to place middleware functions that in the first place reside on infrastructure entities.

Having access to infrastructure support in terms of centralized servers and (re-) configuration options make it simpler to perform migration compared to fully distributed scenarios and ad-hoc scenarios. Therefore, to develop migration functionality for the latter cases at all, we need to investigate the challenges in providing migration in the simplest environment possible to not also have to address many challenges presented by the non-infrastructure scenarios. Once migration functionality is stable, the additional – more strict – assumptions from the distributed networks can be handled within the project.

In the following chapter, the identified middleware functions that are required to perform migration are described based on this plan. This means that for each function it is discussed how it will behave in an infrastructure mode and also if it depends on a migration server or additional configuration support.

6 Detailed interactions in the Open scenarios

As mentioned in Chapter 5, there are two basic scenarios that needs to be considered; 1) with some centralized support in terms of a migration server, and 2) one without such a central support entity (the ad-hoc scenario). As decided within the project planning, the first year we focus on the centralized solution, so as to keep the focus on the core migration part, and then eventually remove the assumption on having infrastructure to support the migration, namely the ad-hoc scenario.

6.1 High level scenario with infrastructure support

In this section we focus on describing the external settings for scenarios, describing which entities will interact with the OPEN platform. The architecture scenario is depicted in Figure 11, and contains:

- A mobile phone sending through a short range network, e.g. Bluetooth, the commands for interacting with the UI which is rendered by the set-top box (STB) on the TV set;
- A broadband connection is used by the STB to exchange data related to the gaming session;
- A second connection is used to send to the STB a video stream representing the 3D rendering of the game.
- The game server provides to the streaming server the data flow to generate the 3D rendering.

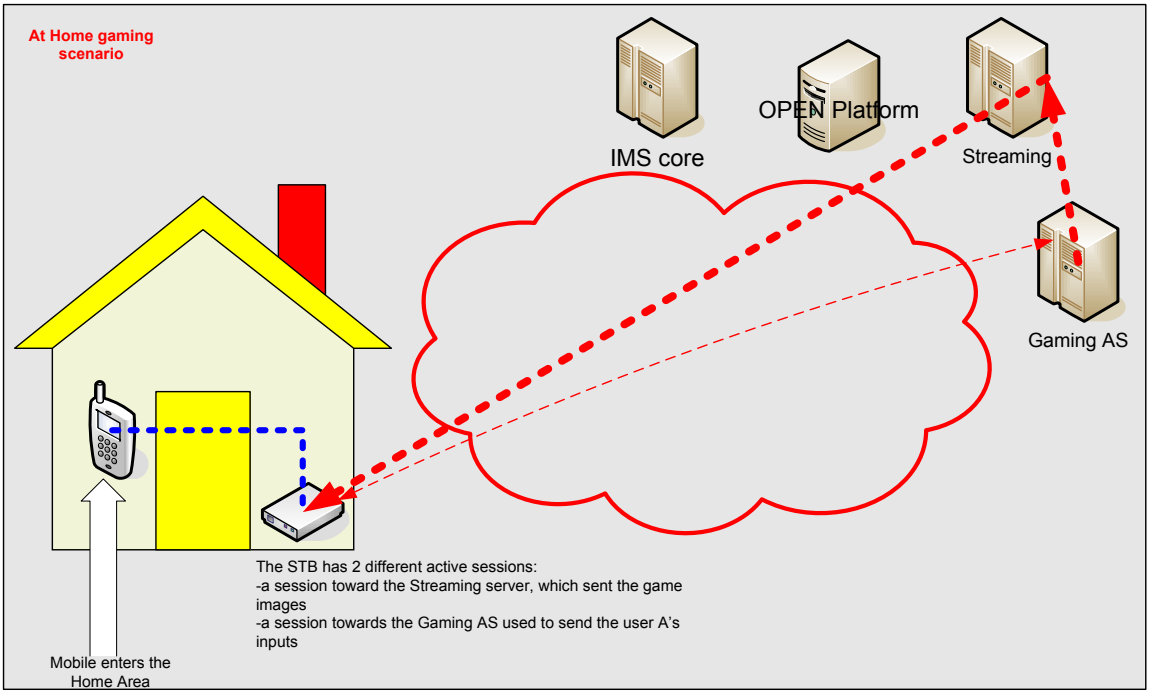

Figure 11: Infrastructure supported home gaming migration scenario

The following analysis starts from the migration described in the IPTV Gaming Scenario between the mobile phone and the STB.

Two different architecture hypotheses have been made:

1. SIP/IMS based architecture
2. Internet based architecture

and two different migration triggers:

- the mobile phone, entering the user's home, detects the STB and triggers the migration
- the Location enabler is notified that the user has entered the cell corresponding to his/her home and triggers the migration

These scenarios are further investigated in the next section.

6.1.1 SIP/IMS based architecture

In the first scenario, shown in Figure 12, the trigger of the migration process is initiated by the mobile phone.

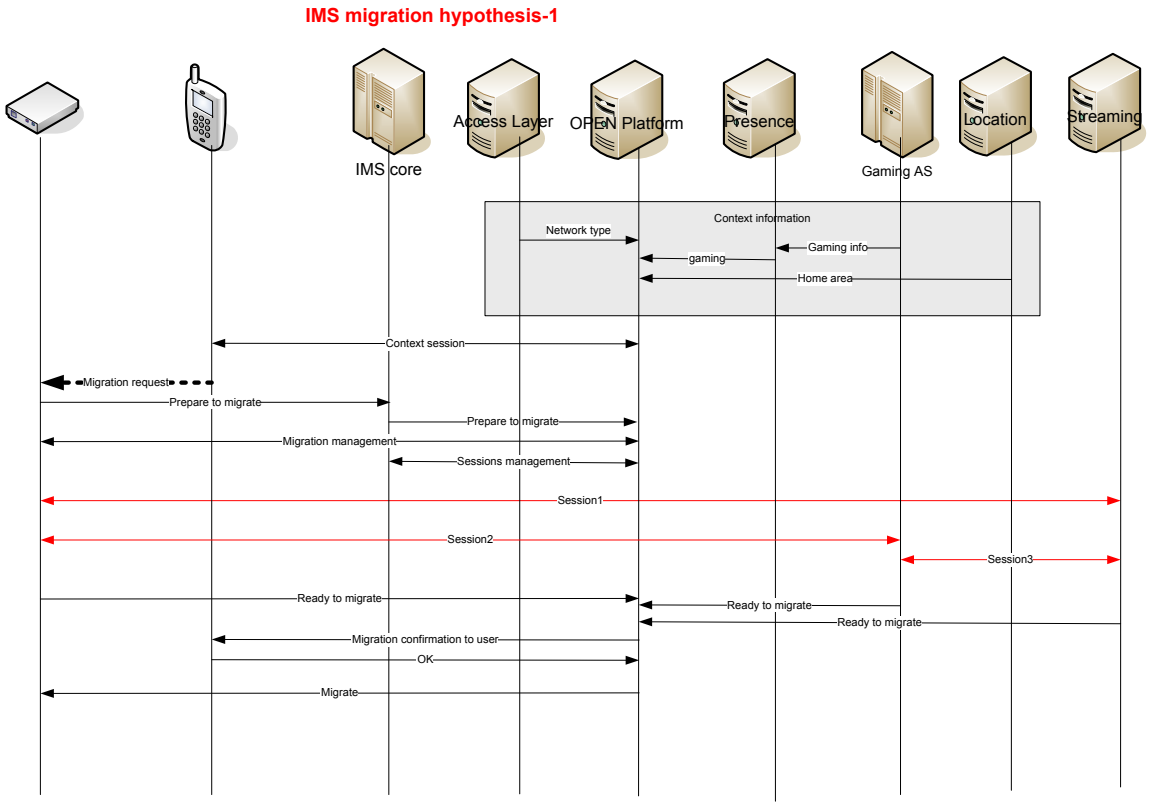

Figure 12: IMS architecture and a mobile triggered migration.

The steps shown in Figure 12 are as follows

1. In the gray box the context information exchange between the different context providers and the OPEN platform is described. Between the

mobile and the OPEN platform a context session is opened, and is used for context information exchange.

2. When the mobile enters the home, it triggers the migration to the STB.
3. The STB communicates to the OPEN platform the migration request.
4. The STB establish an IMS session with the OPEN platform in order to exchange the information required during the migration process.
5. The OPEN platform uses the IMS in order to set up the 3 new different sessions required for the migration.
    a. Session 1, between STB and Streaming server, used for sending the game images to the STB
    b. Session 2, between the STB and the Gaming AS: the STB sends to the Gaming AS the user's inputs
    c. Session 3, between the Gaming AS and the Streaming server: the Gaming AS sends to the Streaming server the game information necessary for the game rendering
6. When the sessions are established, involved modules notify the OPEN platform of the migration readiness
7. The OPEN platform sends a request for migration confirmation to the mobile phone
8. The user confirms the migration, the mobile phone notifies the OPEN platform
9. The OPEN platform confirms the migration to the STB, which now takes over control of the game.

In the second scenario, it is not the mobile phone, but the OPEN platform which initiates the trigger. The scenario is shown in Figure 13, and is described by the following steps.

1. In the gray box the context information exchange between the different context providers and the OPEN platform is described. Between the mobile and the OPEN platform a context session is opened, and is used for context information exchange.
2. When the mobile enters the home, the Location module sends to the OPEN platform the information that the user is in his/her home area. The OPEN platform sends a migration request to the STB.
3. An IMS session is established between the STB and the OPEN platform in order to exchange the information required during the migration process.
4. The following steps are the same as described for the previous scenario (points 5-9) in Figure 12.

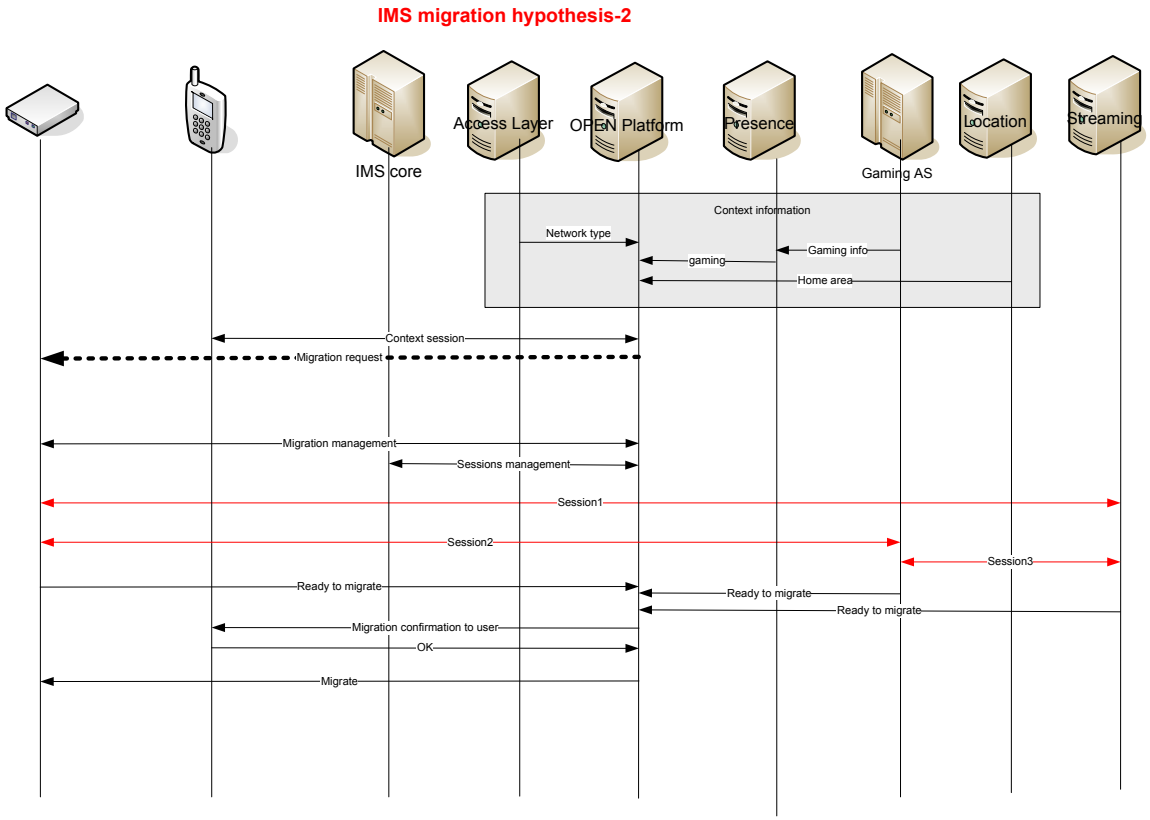

Figure 13: SIP/IMS architecture, network triggering

So in fact the two scenarios shown do not differ significantly despite the fact that the triggering of the migration happens from different locations.

6.1.2 Internet based architecture

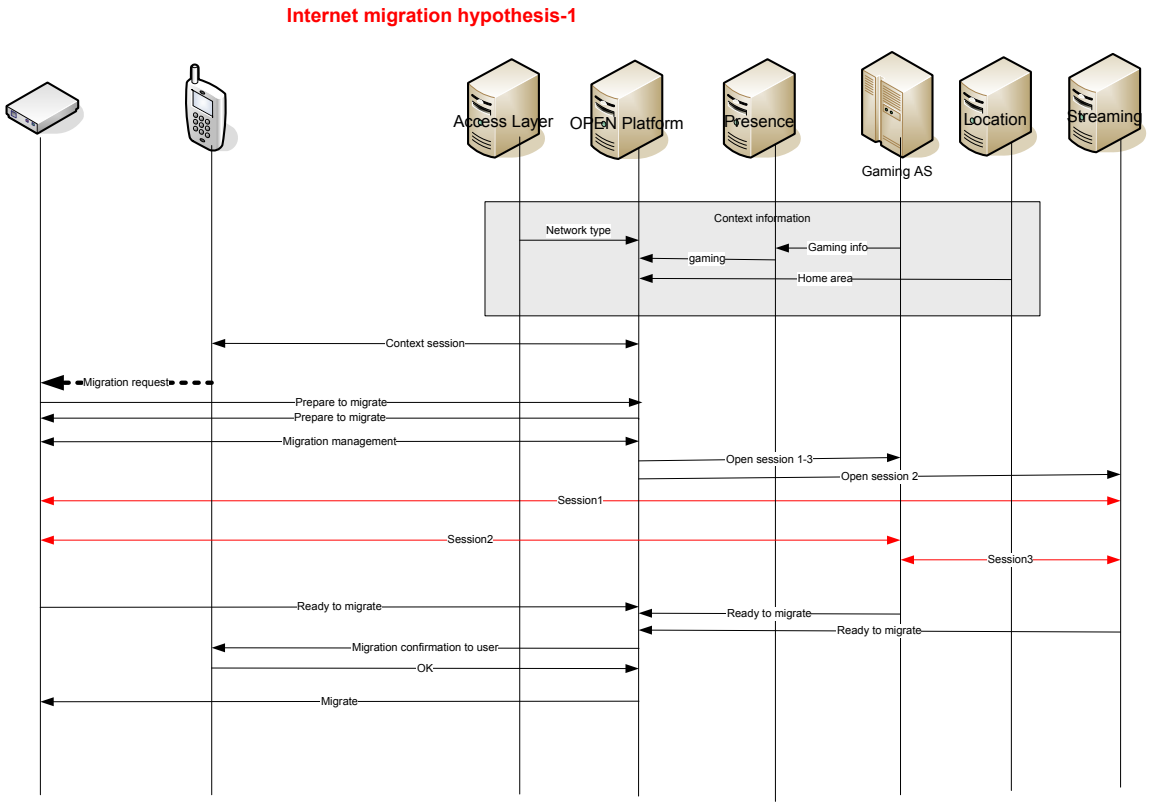

Figure 14: internet architecture, mobile triggering

In Figure 14 we consider an Internet based architecture with a mobile triggered migration. The steps are as follows:

1. The first three steps are the same described in the IMS migration hypotesis-1 scenario, describing the mobile triggered migration.
2. The STB establish a session with the OPEN platform, in order to exchange the information required during the migration process.
3. The OPEN platform sends different messages to the Gaming AS and the Streaming server in order to set up the 3 different sessions required for the migration. In this scenario, the OPEN Platform manages the session establishment, without the IMS support.
4. The following steps are the same as described for the IMS migration hypotesis-1 scenario (points 6-9) in Figure 12.

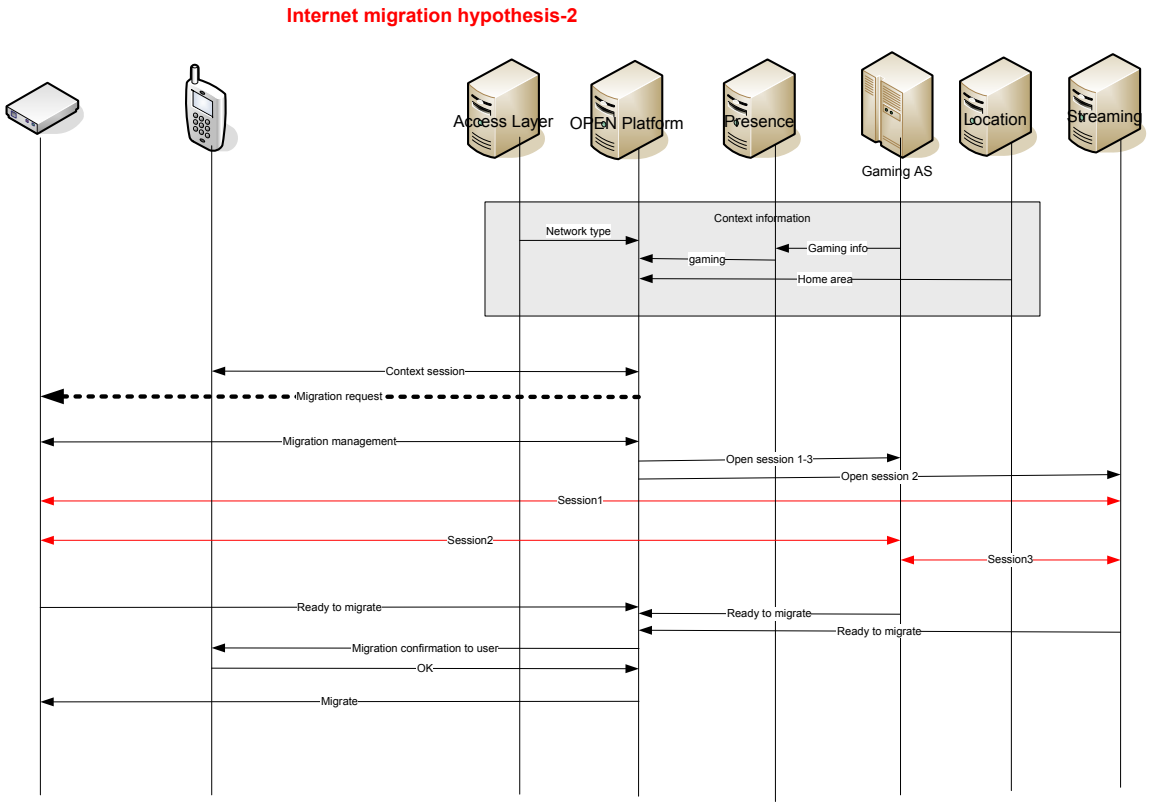

Figure 15: Internet architecture, network triggering

In Figure 15, we consider an Internet based architecture, with network initiated migration triggering mechanism.

1. The first two steps are the same described in the IMS migration hypotesis-2 scenario, describing the network triggered migration.
2. The STB establish a session with the OPEN platform, in order to exchange the information required during the migration process.
3. The OPEN platform sends different messages to the Gaming AS and the Streaming server in order to set up the 3 different sessions required for the migration. In this scenario, the OPEN Platform manages the session establishment, without the IMS support.
4. The following steps are the same as described for the IMS migration hypotesis-2 scenario (points 6-9) in Figure 13.

The IMS migration hypothesis differs from the Internet migration hypothesis for the use of the IMS features for the session management.

6.2 Detailed OPEN platform interactions at component level

In this section we now focus on how the OPEN platform and its internal components interact in order to support the scenarios as described in Section 6.1.

6.2.1 Interfaces between OPEN platform components

Figure 16 provides an outline of interactions between the major components involved in the migration process. Starting from the top, we have the application clients, which interact with the application server.

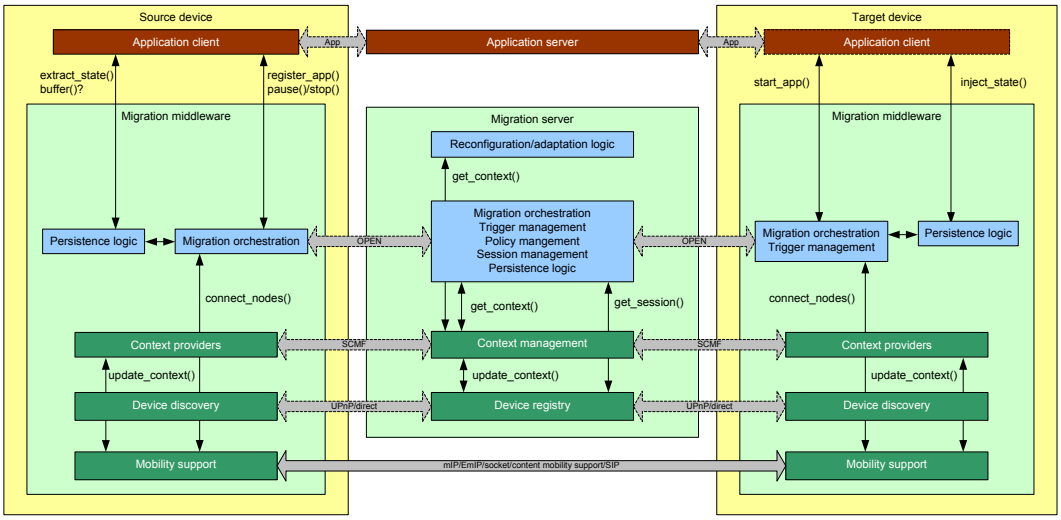

Figure 16: Description of interfaces and interactions within the infrastructure using a migration server, illustrating deployment of the different components

The application clients also interact with the persistence logic in order to transfer the internal application state of the source device to the target device. It is the migration middleware **Persistence Logic** that manages this application state transfer under the control of the **Migration Orchestration** via an OPEN specific protocol.

At the heart of everything, the migration server contains many of the core functionalities like migration orchestration, trigger, policy and session management to ensure that data flow and control flow between the source and target device is happening correctly.

In the background, the context management node is operating independently, gathering and distributing context information, so that all context information is easily accessible for all of the above software entities on the involved devices, i.e. source, target and migration server can easily access context information from each other via the context management interfaces provided. This interface is called the SCMF after the Secure Context Management Framework, which, as explained, has been adopted by the OPEN project.

Next to this component, we have the device discovery and registry system, which is ensuring that the migration server is aware of which devices are available for migration and what their capabilities are. Parts of this information are also considered context, and may be fed into the context management framework as needed, but the discovery process itself requires a separate component as shown in Figure 16.

Finally, we also need to consider mobility support in order to ensure that sessions are not disrupted when the user is mobile.

6.2.2 Interaction and component activity – example scenario

In Figure 17 an activity diagram of the given scenario is shown, which illustrates not only what information is being exchanged between entities, but also which components are in fact active at which time.

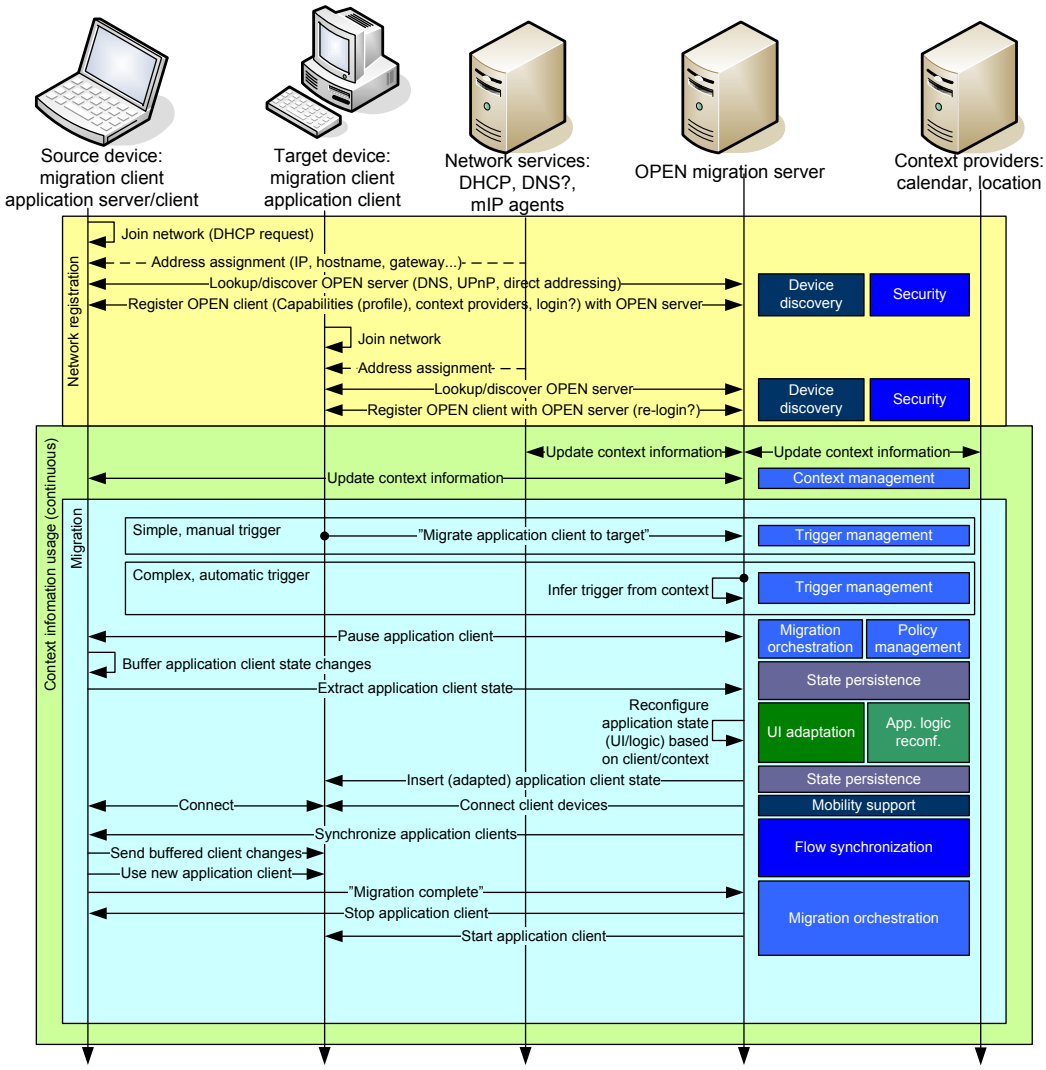

Figure 17: Detailed sequence diagram illustrating when different functions are active during migration in the infrastructure-supported scenario

Although this is just the realisation of one scenario, it should provide more detailed insight into how the components are working and interacting with each other. At first network registration is required. The involved nodes need to be assigned network addresses, locate the migration server and communicate their capabilities to it. The **Device Discovery** is the main active component in this part of the scenario, and involves obviously security measures to ensure that the devices discovered can also be trusted.

Subsequently, after the discovery process, the **Context Management** can establish connections and exchange information about what context is available on which node. Following this exchange of information, the context manager can be seen as a network service running in the background providing easy access to the context information.

Finally, at some point in time, the migration process will happen. It is up to the **Trigger Management** to decide when this happens, it can be done either manually by one of the devices, or it can happen by an automatic trigger based on context information and thus the given situation. Once it has been decided to trigger a migration process, the **Migration Orchestration** component will take over the process management, so for example to first pause the application, and then activate the Persistence Logic for obtaining and moving the application state. Within this phase, there might be some UI and application reconfiguration that happens, which may also have an impact on the application state being transferred, hence the components **UI** and **Application reconfiguration** may interact with the Persistence Logic component at this time. These components are however treated outside of WP3, and a clear interface here is needed. However, whether the application state has been manipulated or not, it will eventually be transferred to the target device, after which the Migration Orchestration can start the application in the transferred state, i.e. the application continues from where the application stopped at the source device.

There may be more steps involved if synchronization between the application state is required, e.g. if there are buffered data like sound and graphics associated to the game, which also need to be transferred and handled synchronously to the game engine itself. Then the **Mobility Manager** may establish connections between the source and target device dynamically and let the **Synchronization** module ensure buffers are the same, so that the game starts exactly at the same point with relation to game time, game state, sound and graphic

As seen it also becomes clear that the migration server is a very central point in the process. It will take many decisions and have access to a lot of information, and the challenge for the next part of the OPEN project is now to focus on how these scenarios would look, if there were no central migration server, i.e. for ad hoc scenarios, and what that would do to the components and their respective information needs. The Context Management system could be a potential important component in this part, since it independently can transfer relevant information between devices, making parts of the information exchange required by each component easier.

6.2.3 Migrating from many devices to many devices

In some of the application scenarios specified in WP5, complex migration situations arise. Examples of these are when applications from multiple devices are migrated onto one device, or the opposite, when an application previously run on one device is distributed onto multiple devices. An extreme situation following the two examples is when applications run on a set of multiple devices and then are distributed to a different (not necessarily disjoint) set of multiple devices.

Generally we call these types of migration *aggregation* when migration from multiple devices to one and *distribution* in the opposite direction. Migrating from many-to-many devices can be seen as an extreme case of distribution.

The network architecture and the connectivity support may be challenged by migration of special types of these applications.

Migration (regular, aggregation or distribution) can be handled completely within the OPEN platform if applications are self-contained and running on the devices before the migration without external communication.

This can be illustrated by using the WP5 example of the simulation applications migrated to the smart wall (Figure 3-4 in D5.1 [22]). In this example, two self-contained applications are aggregated from multiple devices onto a single device, and the applications are either merged into one or represented as two on the single display. In any case, the migration of the applications can already be handled by the current functionality and the complexity in supporting this type of migration lies in the migration orchestration and application reconfiguration functions.

If the application is distributed between multiple devices, i.e. going from the smart wall to multiple devices, the task is similar when the applications do not communicate.

The challenge is complex if the applications communicate with other devices – either with devices in range or remote servers, especially if the applications have non-OPEN-aware peers such as remote 3rd party service providers. In this case, the OPEN framework must support the complex mobility of the applications as well as the coordination and synchronization of the data-flows going to and from the applications.

However, as such applications do not exist in the scope of OPEN we will not develop specific support for many-to-many migration of communicative applications.

6.3 Detailed solutions for selected components

Obviously, there are many different components envisioned in the OPEN platform, but due to resource limitations we will need to focus on a set of the components at first. The selection criteria of the relevant components have been based upon

- Investigation of the core components required by a infrastructure solution
- Those directly supporting the demos as described in Deliverable D3.2, [11]
- Those showing the most prominent concept of service migration

In the following we describe solutions for two aspects, namely Device Discovery and Context Management.

6.3.1 Solution proposal for Device Discovery

As a result of the previous analysis we propose to use OSGi to develop some basic device discovery functions to cover OPEN purposes.

The preferred discovery model is based on a shared registry.

We define the OPEN environment as the set of devices where an OPEN Node Manager is in execution and has provided the device characteristics to the OPEN registry, this means that:

- The basic connectivity is established between the device and the OPEN platform
- An OPEN Node Manager has been successfully set up in the device; with set up we mean that an OPEN Node Manager has been downloaded, installed and executed in the device. The set up method changes according to the device type:
  - for a CPE we propose TR069,
  - for a mobile device we propose to assume the usage of the Device Management Solution
  - for a PC we propose to assume that the user installs the OPEN Node Manager

An OPEN Node Manager in every device should be able to perform a set of functions here described and illustrated in Figure 18:

- **Device Characteristics Notification**: the OPEN Node Manager in every device should be able to provide to the OPEN platform its characteristics such as: **Execution environment**, **Supported Connectivity, Device name, Device location, Privacy**, **Interaction resources**. The notification should be performed when the device connects to the OPEN environment and updated when the device characteristics change or according to certain update policies (to verify if there are needs for this updating policy). When the device enters the OPEN Environment the OPEN registry will provide a unique device identifier. We have to emphasize that this function could be part of the more general context notification process where the node manager of every device provides context information unrelated to device status
- **Surrounding Device List**: the OPEN Node Manager in every device should be able to ask the OPEN platform for the list of surrounding devices and their respective characteristics. This function can be invoked when the user wants to migrate the application, or can be automatically triggered depending on the context, for example when the device battery level reaches a low level.

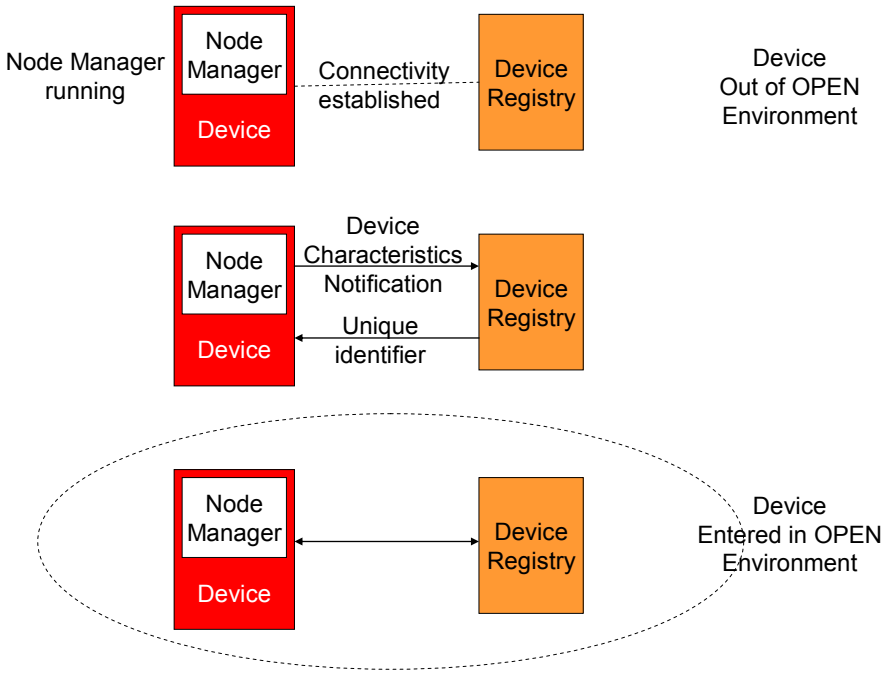

Figure 18: Illustration of device discovery concept

As a device entered in the OPEN environment could become unavailable for various reasons, there is a need for the OPEN registry to verify the status of the devices that previously were in the OPEN environment. With this function, which we call **Device Availability,** the OPEN registry requests a given device to update its characteristics using the **Device Characteristics Notification** function.

6.3.2 Context Management Framework

6.3.2.1 Basic entity definitions

The basis of the Context Management in OPEN is, as mentioned earlier, the Secure Context Management Framework from MAGNET Beyond, [17]. This means that some modifications would be required as the network assumptions are different than in MAGNET. Specifically the SCMF in MAGNET was operating in a secure, trusted environment with nodes organised in groups or clusters of nodes. In OPEN the focus is different, since most devices are locally present, and the whole overlay concept as required by the large network in MAGNET Beyond and its Personal Network concept, is not needed for OPEN. However, the scenarios chosen in OPEN would make use of existing terminology. Those we intend to use and how they relate to the OPEN platform are as follows:

- **Context Agent (CA)**: A software component that is assumed on all OPEN enabled nodes, implementing the functionality to collect, process and

distribute context information for OPEN components and applications/services.

- **Context Management Node (CMN):** A specific CA which is responsible of knowing what context information is available and where in the network different context information can be found.

As for now, OPEN is focusing on a centralised solution based on a service migration server, the CMN is naturally chosen as being on the Service Migration server, since this anyway has a central role in the whole migration process. Figure 19 shows the settings of a laptop and mobile phone, which will migrate an application/service session via a migration server.

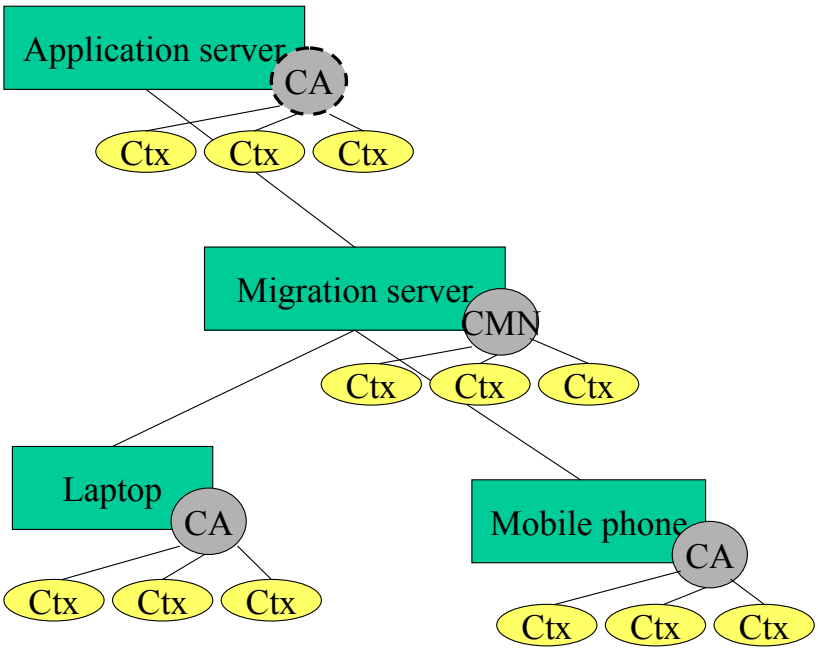

Figure 19: Service migration of an application/service session between a laptop and mobile phone, via a migration server, and how CA's and the CMN are distributed on the involved devices, collecting locally available context information (Ctx).

As shown in Figure 19, the assumption is that there is at least a CA on every node involved. Depending on whether the application server is OPEN enabled or not, there is also a CA on it. The CMN is always located on the migration server, since this is anyway the central point in the whole migration process.

6.3.2.2 The Context Management Framework in infrastructure supported scenario

Figure 20 shows the general overview of the interactions between the Context Agents in the scenario with the infrastructure solution. In the following a description of the scenario is given, with focus on the activities of the Context Management Framework.

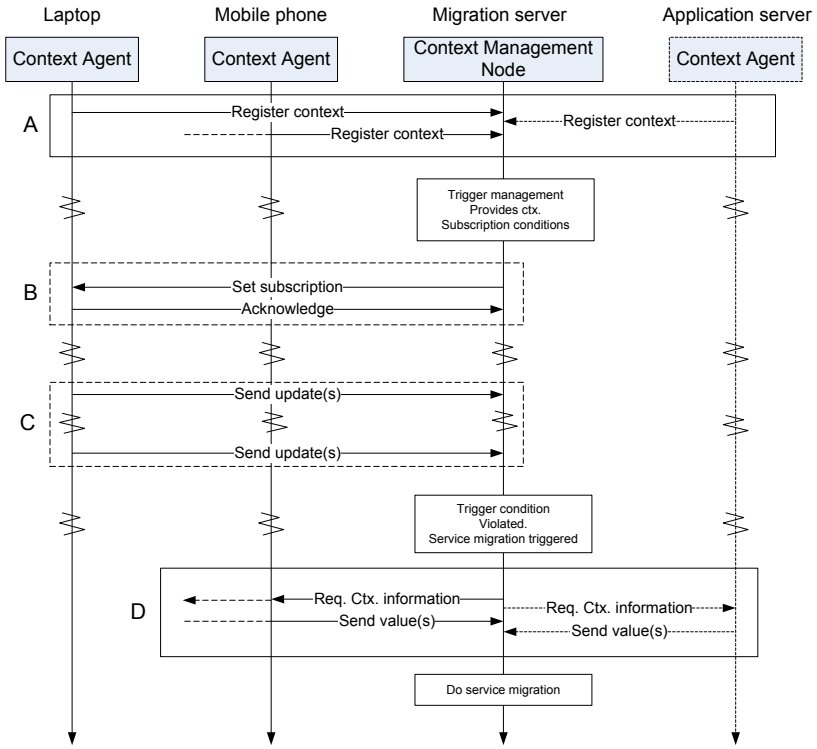

Figure 20: General message chart diagram of Context Agents interacting in infrastructure - supported service migration solution. The application server may or may not provide context information, depending on whether it is OPEN enabled or not.

As seen, the first step (Step A) is to have all the Context Agents on the different devices register their local node information to the Context Management Node. This is simply done by providing the Context Manager with context locators and unique identifiers by which the CMN can distinguish similar context elements from different devices.

Following this, the Trigger Manager is responsible for determining the type of context information to be used and the conditions for the Triggering mechanism. This information is passed on to the Context Management Node as a subscription request to the source node for the relevant context information, as shown in Step B. There may be more than one element, or in fact, the information subscribed to, may require information from other nearby nodes. It is up to the Context Management Node to find the best source of information (which for simplicity is shown in Figure 20 as the laptop which also happens to be the source device).

Once the right information source has been located, a subscription with the desired subscription conditions is sent to the device as a part of Step B. An acknowledgement with subscription ID is send back. The ID is used to differentiate subscriptions from each other, if the CMN has multiple ongoing subscriptions. Updates are sent either regularly (periodically with a specified time interval) or whenever an event occurs in Step C. An event or the time period of updates is a part of the subscription conditions. When the CMN on the migration

server receives the updates, it simply forwards them to the Trigger Manager, which, in turn, decides whether to trigger a service migration or not.

When the service migration is being triggered, the Migration server will have to locate the best device to migrate to, which depends highly on context. But before the Context Manager can request context information, it relies on the Device/Service discovery to discover potential candidates. The additional step is to ask those devices discovered in the device/service discovery process about their individual context information. Thus the migration server (Migration Manager) will request context information related to potential candidate targets of a service migration (done in Step D).

Whether an application server is OPEN enabled or not, impacts the accessibility of context information from that server. If it is OPEN enabled, the assumption of the presence of a Context Agent on the server is valid, and therefore the CMN can directly access context related to the application server to be used in the service session and/or the service migration process as well. If it is not OPEN enabled, then in order to get that information, potentially, retrievers or processing components (see Figure 4) may be written explicitly to gain access to the relevant information at the application server. Because retrievers operate on a native interface and do not depend on the Context Managers own internal interface, this may be possible. Furthermore, inferred context may be needed, since some state information may not be directly accessible via a retriever, but may require inference or estimation techniques to be provisioned. This is, of course, a sub-optimal solution, but provides effective means to provide context information, even if the source (here the application server) is not a part of the Context Management Framework.

6.3.2.3 Scoping of context information requests

An important aspect that allows such a system to work efficiently, is to provide the capability of scoping of the requests, e.g. it is not a good idea to blindly provide information of battery status, if it is measured on a device that is not a potential candidate for migration. Thus, it must be possible to directly address a request to a specific node in the network.

Query scoping is a mechanism that allows the application to redirect or limit queries on context information to specific nodes or groups of nodes. For MAGNET Beyond, the scopes of interest were defined by; *local node*, *cluster*, *PN* or *PN-Federation*. In OPEN the requirements are different, firstly because the PN and PN-Federation concept is not used. Instead it would be necessary to redirect queries to specific nodes or entities, so that in the example of Figure 19, the migration server may request *battery status of mobile phone*. It is, of course, a problem that the mobile phone (target device in this case) is not known a priori. Thus the request should happen to those devices that have been found first by Device Discovery for later selection of the appropriate device. Once Device

Discovery has discovered potential migration targets, the Service Migration Manager, may request the Context Management System about battery status of the potential candidates, and thereafter select the device with most battery power left (or considering also other information in this selection process).

Hence, it is a clear requirement for the Context Management Framework to be able to direct context queries directly to specified devices with given ID's such as IP address or similar.

6.3.2.4 Bootstrapping the framework

When starting up a Context Agent, it will register the accessible context information on its node to the CMN, from which other Context Agents, may retrieve information of the network location of specific context information. In the first year goal, the CMN is statically chosen as being on the Migration Server, which makes the bootstrapping of the Context Management system fairly easy. Static configuration of the Context Agents to inform them that the CMN is on the service migration server is enough. Since this information is required by all other OPEN components, it is safe to assume it is also available for all of the Context Agents.

For later scenarios with ad-hoc networks, there is no such central entity, and therefore the Context Management system must decide independently where to store the context registry information. This will be addressed in the next step of OPEN, but previously solutions from MAGNET Beyond were based on an election mechanism, that allowed nodes in the network to compete to be the CMN, and allowed for dynamically triggering such a contest when nodes were arriving, leaving or not responding for a long time (failures), see e.g. [19].

6.3.2.5 Further adaptation of the SCMF into OPEN context

As a conclusive outcome of the MAGNET Beyond it was found that some service frameworks would benefit the SCMF in deployment scenarios. In the following we investigate, based on these aspects, different frameworks, and how the SCMF fits into those.

6.3.2.5.1 OSGi framework

OSGi would for the SCMF be beneficial in particular to those components which are interchangeable, i.e. retrievers and processing components. OSGi would allow runtime installation of needed/required components, and uninstallation of no longer needed components. Beside this, OSGi would also allow for better control on the life cycle of the components, strengthening the framework.

Because of the way OSGi works and defines component and service interaction, there will be required some changes to the current implementation from MAGNET Beyond in order to make it run in an OSGi environment. Those obviously foreseen are

- Internally, the classes or one super class would need to implement a *start* and *stop* methods as required by the OSGi framework
- Service event listeners would potentially be needed to be implemented in order to receive and handle service messages from the environment.

Besides an implementation effort, turning the SCMF into OSGi framework, will obviously make it dependent to this framework.

6.3.2.5.2 Corba

Since Corba is considered being used in OPEN, it is also of potential interest to see if the SCMF fits to this framework. However, since the interaction with the SCMF happens through XML-RPC and the implementation is done in Java, the interaction with Corba based components is not envisioned to be a huge problem.

6.3.2.5.3 SCMF and Web Services

Finally, as web services is an important part of OPEN, the SCMF should also be considered in relation to web services. This we explain in the following:

- Usually, all access to the SCMF is done locally and the SCMF handles distribution aspects. Therefore, there is no other API needed.

For reasons of uniform service composition, the SCMF API could be realized as an Web Service. Currently the SCMF uses XML-RPC for providing an API to applications. The reason is that XML-RPC has been available for small devices like Windows Mobile. Furthermore, there is a Java object that hides the XML-RPC.

6.3.3 Mobility support in central migration server

As described in section 3.2, mobility must be supported by the service migration platform. It is the responsibility of the 'mobility support' function to provide this support. In the first year version of the migration platform in OPEN, a central server is placed in the data path of communicating applications. This means that whenever an application contacts its peer, this contact is established through the central server – here called a migration server.

In order to define mobility, something must be moving. In OPEN, mobility means that components of the application, providing the service to the user, moves during the service session. In general the moving components can be located both server-side and client-size in the communicating application, but in OPEN we only consider client-side components in the first year version. Thus, the mobility type to be supported in the platform is 'service mobility', as the client-side

interaction components of the applications move between devices and thereby represent moving the service.

As the components move between peers, the connectivity to the corresponding peer(s) must be maintained – which is basically the requirement to the mobility support function. If the following we describe a set of scenarios illustrating service mobility and discuss possible mobility support solutions in the scenarios.

The scenarios are based on examples of video streaming migration from one client to another client. The stream is served by a corresponding server outside the domain of OPEN and the stream is displayed on a client part of the application on the client nodes, within the OPEN domain. To have peers in the OPEN domain basically means that some part of the OPEN functionality can be assumed installed on the peers.

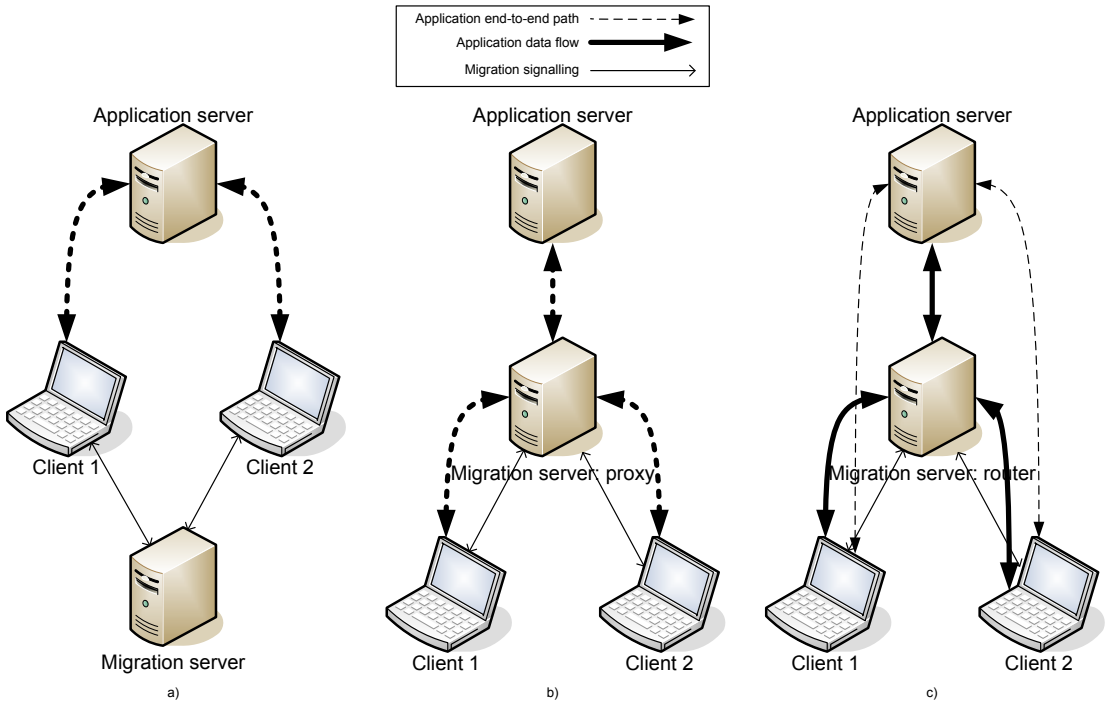

Figure 21: Scenarios with different use of the migration server: a) Migration is outside the application data path (i.e. mobility support > L4), b) Migration server is in the transport path (L4 mobility support), c) Migration server is in the network path, so end-to-end connection between client and server exists (L3 mobility support)

In Figure 21, three different scenarios are illustrated. These are described in the following.

a) Direct client-server connection: In this scenario, the clients have direct connectivity to the corresponding peers of the application. The migration server is used to orchestrate migration of application components. As it is not placed in the data path of the application, support of mobility with regards to the application

traffic cannot be handled by the server, meaning that the mobility support functionality must be placed on the client part of the OPEN platform.

As this is not within the scope of the first year version, this scenario is only described to illustrate service mobility support when implemented on the service layer, i.e. directly on the peers involved in the applications. A solution for this scenario will be described in the second year version, which is described in D3.4 [21].

b) Migration server in transport path: In this scenario, the migration server is placed in-between the clients and the server. It is placed in such a way that all application traffic is tunnelled through the migration server. From the client-side the migration server takes the role of a proxy, meaning that to establish a connection to the application server, the client requests the migration server to take part in the establishing process.

A detailed message flow of the process of establishing a connection is illustrated in Figure 22. The migration trigger and the migration itself are illustrated as the handover where the second client device, after registering with the migration server, receives the application traffic instead of the first client device. As the migration server is placed within the data-path, it has complete control over the connections established between the clients and the server, and can thus decide who to receive the traffic.

The most challenging task is, when using TCP, to have the second client device establish a new TCP connection to the migration server on which to receive the video stream. This is handled by the application client, as this is the component that needs to receive the data. However, for the migration server to be able to redirect the traffic to this new socket, the port number needs to be known on which the TCP connection is established. This is the task of the mobility support function on the client device.

When the new socket is opened, redirection of traffic can be handled in the migration server by use of regular proxy mechanisms such as address redirection, network-address-translation or port-mapping.

c) Migration server in network path: In this scenario, the migration server is too placed in-between the clients and the server, although here the clients establish end-to-end connections with the application server. This means that the migration server is transparent on the transport layer and only visible on the network path. This is a more complex scenario than the previous, as traffic must correctly be rerouted from the first to the second client device by the migration server without direct control. The task is similar, but the means are limited to layer 3.

Here, mobile IP with filtering mechanisms [20] should be applied. The filter-bindings extension to mobile IP lets an agent forward traffic determined by filter that can be applied on address or even socket level. The migration server would the take the role of a *home agent*, redirecting traffic to the new client device.

A detailed message flow of the process is illustrated in Figure 23. Here, the problem of establishing the new TCP connection remains unsolved, as the migration server is not a part of the end-to-end communication. This problem is to be investigated in the design of the solution, if the migration server is designed as a network layer solution. Further analysis of the problems and solutions will be performed and used to document the final migration support solution in D3.4 [21].

Assumptions of the scenarios and the solution:

- Possible to place the migration server in the data path.
- Complete control of the network in the OPEN domain. No NAT or firewalls prevents client-to-migration-server communication. This requires the migration server to be located either in home- or corporate-network types.

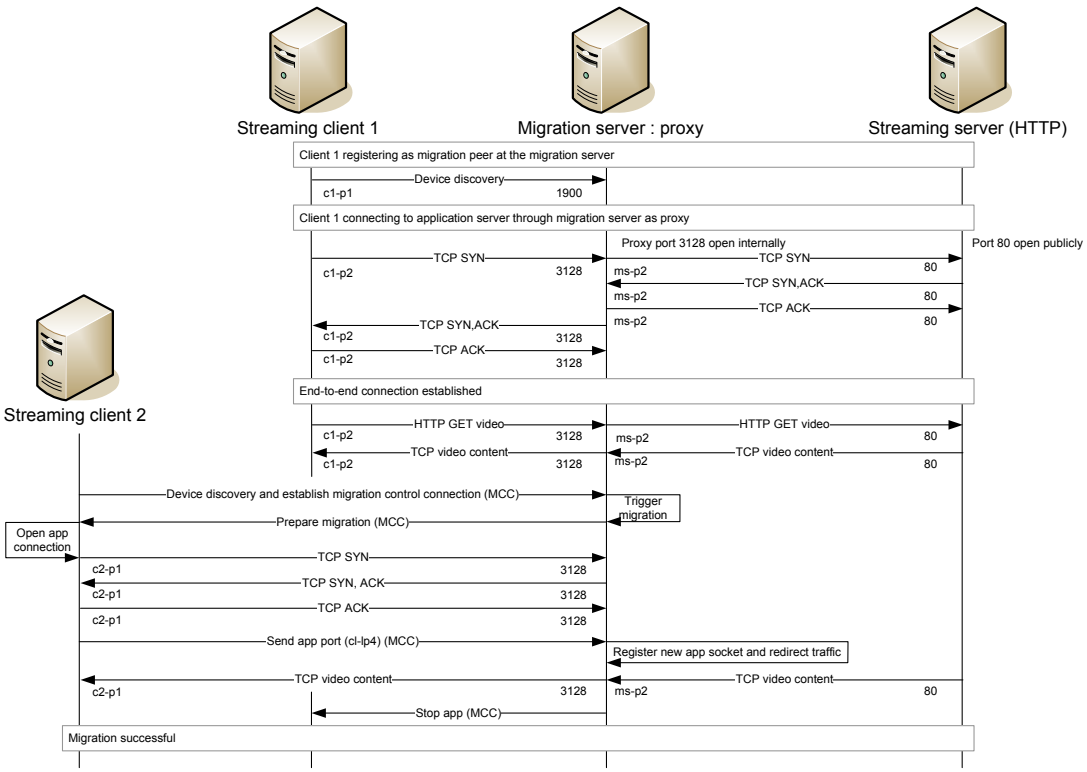

Figure 22: Mobility support in migration proxy in a stream-hand-over scenario – the migration proxy is directly in the transport-path

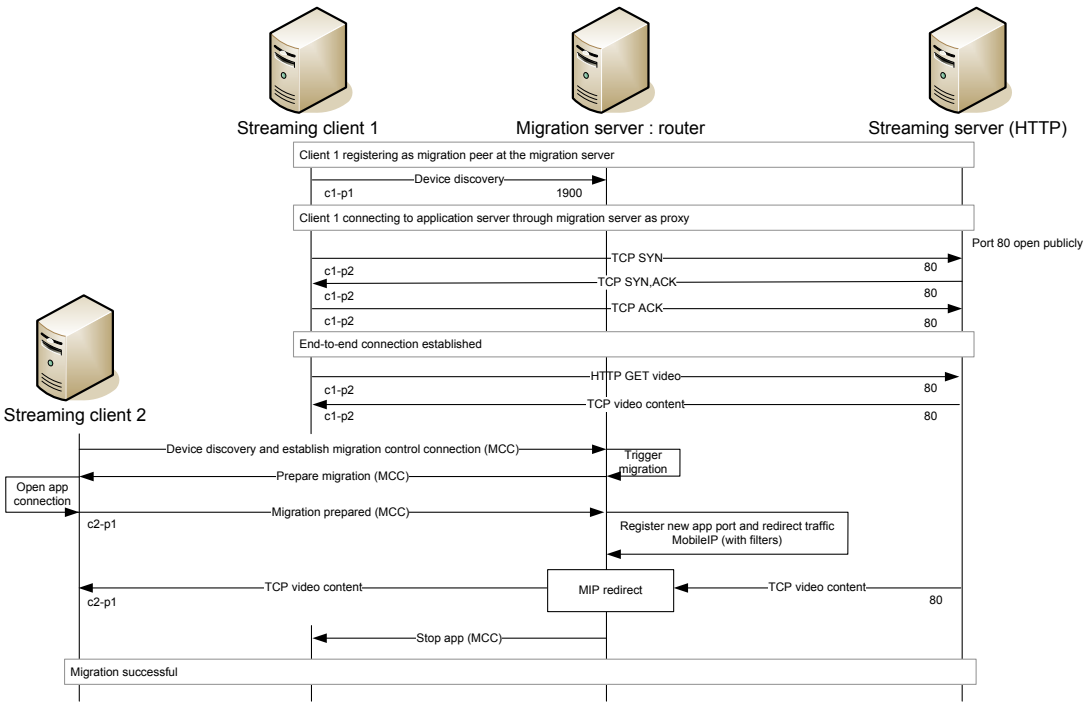

Figure 23: Mobility support in migration server in stream-hand-over scenario – the migration server is in the network-path and client-server communication is end-to-end.

As shown, solutions to mobility support exist on different layers of a communication model. In the first year version of the OPEN platform, we aim at providing basic versions of all of the functionality needed to perform migration. There we choose to use the L4 mobility support solution, and assume that we have the migration server in the transport path of any communicating applications. The migration server will then be involved in the communication in the form of a proxy, i.e. instead of connecting directly to the application server a client connects through the migration proxy. This way the migration server has full control over the traffic and has thus the best basis for supporting mobility when migrating.

Placing the migration server in the network paths is an interesting solution for future version, especially as several of the mechanisms of mobile IP used to handle mobility can be applied in the more distributed scenarios of the second year version of the OPEN platform, which then also encompasses scenario a).

7 Security aspects of migration support

Now we turn our attention to the security part of the service migration. We focus here on simple analysis and requirements, and provide a brief overlook of the possibilities of existing solutions that may fit into the solution described. It should be noticed that OPEN is not focused on security as per se, hence we do not aim to develop new technologies, but focus on off-the-shelf solutions for the functionalities requiring security considerations.

7.1 OPEN security requirements and selection criteria

Most of the security requirements in D1.1 [3] relate to privacy and itspreservation. These are, for instance, requirements 15, 23, 38 and 141. As the platform developed in OPEN is intended to be an open platform to support many different users and many different applications, a lot of the interaction going on during migration will use information from many sources. This means that also information private to one user will be used for migration. Storing and accessing this information in a secure and 'private' way is a key challenge to the OPEN platform. This means the establishment of trust between nodes are key important to the OPEN security.

Furthermore, since services needs to be controlled remotely, access control is also required in some form, as to decide which entity may have access to starting and stopping services and applications. Access control is also a requirement as to control which context information is being accessed, and as it will be clear in the subsequent section, access control to context information poses a special set of challenges.

7.2 Overview and analysis of existing security solutions

7.2.1 Secure connectivity

Several of the technologies described as state-of-the-art for connectivity support in OPEN (see section 3.2) implement their own security model dealing with attacks and faults that may threaten the security of the particular technology.

For instance, the binding messages of mobile IP (that enable redirection of messages) must be authenticated and protected against replay attacks. This is handled through registration to the home agent using IPsec [5] to create a security association. This way keys can be exchanged between peers to allow for encrypted communication.

In general, IPsec can handle many security issues transparently to the higher layers as it operates independently on the networking layer using its own security mechanisms. It has support for authentication and encryption of packets.

In general, SCTP is also able to rely on lower layer security mechanisms such as IPsec, however, it does provide some mechanisms itself. When establishing individual flows inside an SCTP connection, a four-way-handshake is used between client and server for authentication.

SIP relies on secure transportation and has no built-in security mechanisms. Work is currently ongoing to find solutions for this.

7.2.2 Security and privacy control for Context Management Framework

The SCMF, as used in the MAGNET Beyond Project, was based upon the assumption that it was working in a trusted network environment. This assumption was correct, since the networking concept in this project, Personal Networks, was exactly providing such an environment, meaning that the interactions between Context Agents would always happen between trusted entities. Thus, only privacy and access control was needed to meet the security and privacy requirements.

In OPEN, however, it is not the case that devices necessarily trust each other, hence prior to the exchange of context information these must ensure they trust each other. This potential lack of trust puts high requirements on what information is sent to which devices, hence access control in necessary. For context information the requirements on access control are not necessarily the same requirements as for other types of information or services. Normally, access control is about either allowing or not allowing access to information or services, but for Context Management this is different as we elaborate next.

To utilise context sensitive services, the user has to reveal context at some time, e.g. for location based services, the location of the user has to be revealed at some time in the service usage process, as otherwise it is not a location based service. One aspect of access control for context management is obfuscation of context information; so for example, the user will reveal his location to less trusted entities but with less accuracy, or ambient temperature but using terms such as *warm* or *cold* instead of 10 or 29 degree Celcius.

This type of access control, which includes obfuscation of data does not have any off-the-shelf type of solution, as it will involve data manipulation on the internal data structure of the SCMF in the first place, but needs to be included in the existing SCMF, if at some point in time it should be deployed. It should be noted here, that security and privacy control for context management frameworks is not a simple matter of inventing a clever technology, but relates closely to legislation of storing and distributing personal data. Such legislation differs from country to country, and may pose several types of additional challenges that may well require the focus of dedicated projects.

7.2.3 Trust establishment

As one of the important security aspects is the establishment of trust between the involved entities, since service migration potentially includes exchange of personal sensitive information (application data, context information, …) the user will need to trust that such information is not transmitted to malicious devices. Within the scenarios which rely on a centralized server, i.e. the migration server, trust may be established on certificate exchanges between involved entities prior to the migration process. This, however, would require additional functionality and infrastructure enabling the platform to distribute such certificates which, for example, can be either web-of-trust (such as done in OpenPGP, [23]) or public key infrastructure based (PKI) e.g. SCVP [24][24]. Trust establishment can therefore be determined and easily verified by a third party trusted entity. This, obviously, ties the solution even more to an infrastructure based solution.

For ad-hoc scenarios, things are different, since there is not necessarily any central entity to be trusted. This is an area of research, whereas some aspect covering the issue of trust establishment in ad-hoc scenarios are based upon online observations of neighbouring devices, [26], [27], others on imprinting techniques as has been done in e.g. [25] which is a procedure similar to the Bluetooth pairing. From a time performance point of view, the latter option appears to be a good solution, since once the imprinting procedure has happened keys are readily available to be used on both devices, which distribution is based on the user's trust in the device, which is then used to ensure secure communication and service migration. It requires, however, that the user has had a priori knowledge of the devices involved, and the user has actively been doing the imprinting procedure. In that perspective, the observation based approaches, offers simpler solution to the user and requires not necessarily an a priori knowledge of devices.

7.3 Security solutions in OPEN

Several security issues exist in OPEN and this section identifies which solutions are applied to address these issues. As security is not the primary focus of the project, the solutions described are standard basic security solutions. When the functional details of the architecture in different scenarios are known the specific security solutions can be designed into the platform and implemented for testing in WP6. Below is a list of security risks and appropriate solutions:

- **Authenticity:** During migration a high degree of trust is required as personal information may flow between differently owned devices due to applications being deployed on several devices. To ensure that no device is able to spoof its identity and thereby be able to obtain personal information, the migration server is used as an authentication server. When a device registers its OPEN participation on the migration server, the client is requested to authenticate towards the OPEN platform. This can be either

using passwords or by pre-authenticated private/public key infrastructure (PKI).

- **Eavesdropping:** As we have wireless media in the network domains of OPEN the risk of eavesdropping exist. Especially because trust relations are to be established on-the-fly when device enter and leave networks confidential information is passed back and forth using such insecure media. To avoid this encryption using IPSec/IKEv2 is applied. IPSec is able to use different encryption algorithms to encrypt message passed between communicating peer using a security association. To establish this association Internet Key Exchange v2 (IKEv2) is used. The benefit of using v2 is that it is able to decouple the key used to establish the security association and the keys used to encrypt the actual data sent. This strengthens the solution as long-term keys can be deployed on the devices, which makes the establishment phase less complex. As these keys are difficult to update in case they are disclosed, there should be a low risk of learning these keys from eavesdropping. Thus decoupling the keys used for the data, the long term keys are only used in a very short time period during the migration phases.
- **Authorization:** Migration authorization will be based primarily on user acceptance. This can be manual or automatic based on previous choices or user profiles.
- **Integrity protection:** Using shared keys integrity protection is encompassed in IPSec.
- **Access control:** As mentioned access to context information, but also to services is required to be controlled. The simplest approach for the context management is to implement an access list (ACL), which based upon requesting ID's (which contains unique ID's from devices) checks whether access can be granted or not.

8 Conclusions and next steps

The deliverable presents the outcome and achievements for Work Package 3 which, in collaboration with the other work packages has developed relevant scenarios [3], derived requirements [4] and from those, extracted the most relevant and extrapolated those into a set of functional components which now entails the OPEN platform. This work we briefly introduced in Chapter 1.

The components developed reflect different needs and requirements to the overall service migration process, whereas some are core to the process, and some are supporting the process, which is reflected in the deliverable structure in terms of Chapter 3 and 4. In those chapters we elaborate on the functionalities of each component, including interaction with other components and how the component can be realised based on existing technologies. A set of relevant components is included in the demos that are described in Deliverable D3.2 [11].

In Chapter 5 we put our focus on the deployment of the described functions and components in different network scenarios. The main outcome of the chapter, is two-fold. First, an outcome is the light cast on the possible scenarios and the challenges and aspects that are found within the different deployment scenarios. Second, focus of the work package was identified and properly selected on which challenges to address at what time within the project time. For the first year, it was thus decided to keep it simple, which meant that we aimed for an infrastructure-based solution including a central migration server providing the core and some supporting functionalities in the OPEN platform. This delimits the project for problems related to e.g. locating information and functions, hence enables the project to focus on the core part of the migration process itself. In the second year, we will then turn our attention to the ad-hoc scenarios, for which there are no longer assumptions on the whereabouts of e.g. certain information and functions.

In Chapter 6 we used the fact of having a migration server available to the migration process, and described possible interactions with first external service components and architectures, then secondly internally between OPEN components and described their respectively interfaces. This is further elaborated in detailed message chart diagrams for given scenarios based on the work found in D1.2, [4]. Again, it should be mentioned that part of this work has been demonstrated in Deliverable D3.2 as to prove different aspects, and in particular context triggered service migration, device discovery and the support of application reconfigurability.

Finally, in Chapter 7 we address the security issues related to OPEN and service migration. Since this project is not focusing on security as such, the value of Chapter 7 is mainly related to security analysis and possible solution that may fit into the overall vision of service migration in OPEN.

In summary, at the point of writing OPEN Work Package 3 has now achieved to describe the OPEN platform as we envision it to be, although for a final specification the feedback from implementation and integration process still is needed. Further, we have elaborated the functionality in different deployment scenarios and message diagrams. To support the descriptions some implementation and integration has also taken part in shape of three demos shoving different aspects of the OPEN platform (see Deliverable D3.2 [11]).

In the following year of the OPEN project Work Package 3 will put focus on 1) finalisation/refinement of component function and interfaces based on feedback from research and implementation, 2) investigate the case of ad-hoc scenarios, i.e. when we cannot assume a migration server being available for the migration process and 3) continue integration of the platform to the overall OPEN system. The latter is also closely related to an integrated demonstrator of the platform working across the project, where collaboration with in particular Work Package 4 will be intensified. Finally, Work Package 3 will also put focus on system performance evaluation, which will be done in close collaboration with Work Package 6.

9 References

[1] SJ Koh, MJ Chang, M Lee, mSCTP for soft handover in transport layer, Communications Letters, IEEE, 2004

[2] H Schulzrinne, E Wedlund, Application-layer mobility using SIP, ACM SIGMOBILE Mobile Computing and Communications Review, 2000

[3] Requirements for OPEN service platform, EU FP7 ICT project OPEN, deliverable D1.1, May 2008.

[4] Initial OPEN Service Platform architectural framework, EU FP7 ICT project OPEN, deliverable D1.2, September 2008.

[5] J Arkko, V Devarapalli, F Dupont, Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents, RFC3776, June, 2004

[6] Claudia Villalonga, A Proxy based solution for the Discovery of Self Promoting Services, Diploma thesis, March 2006

[7] Salber, D., *Dey*, A.K., and Abowd, G.D. The *Context Toolkit*:. Aiding the development of *context*-enabled applications. Proceedings of CHI'99, 1999

[8] Jasper Aarts, Transparent Context Framework Interopoerability, Diploma Thesis at University of Twente and Telematika Instituut

[9] Henricksen, K., et al., Middleware for distributed context-aware systems. Lecture Notes in Computer Science, 2005. 3760(International Symposium on Distributed Objects and Applications (DOA)): p. 846-863.

[10] Open Wireless RouTer, OpenWRT, http://openwrt.org/

[11] Anders Nickelsen, Rasmus Olsen (AAU), Holger Klus (ClU), Giuseppe Ghiani (CNR), EU FP7 ICT project OPEN, Deliverable D3.2, February 2009

[12] OPEN Deliverable 4.1

[13] Fabio Paternò, Carmen Santoro, Antonio Scorcia, Architecture for migratory user interfaces, OPEN Deliverable 2.2, February 2009

[14] Online: http://www.upnp.org/

[15] IST-027396 My Personal Adaptive Global NET and Beyond, MAGNET Beyond, Deliverable D2.3.1, "Specification of PN networking and security components**",** December, 2007.

[16] IST-027396 My Personal Adaptive Global NET and Beyond, MAGNET Beyond, Deliverable D2.3.2, " PN secure networking frameworks, solutions and performance", June 2008

[17] Online: http://www.ist-magnet.org/

[18] M. Bauer, R.L.Olsen, M. Jacobsson, L. Sanchez, J. Lanza, M. Imine, N. Prasad, "Context Management Framework for MAGNET Beyond", 15th IST

Mobile & Wireless Summit Communications Summit, Myconos, Greece, June 2006

[19] L. Sanchez, J. Lanza, L. Muñoz, "Cluster Head Selection and Maintenance over Heterogeneous Mobile Wireless Personal Area Networks. An experimental approach", 9th International Symposium on Wireless Personal Multimedia Communications - San Diego, September 2006

[20] Kuladinithi, K. and Fikouras, NA and Könsgen, A. and Timm-Giel, A. and Görg, C., Enhanced Terminal Mobility through the use of Filters for Mobile IP, Proceedings of the Summit on Mobile and Wireless Communications (IST Summit), 2003

[21] Final communication and context management solution for migratory services, EU FP7 ICT project OPEN, deliverable D3.4, September 2008.

[22] Initial application requirements, EU FP7 ICT project OPEN, deliverable D5.1, September 2008.

[23] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer, OpenPGP Message Format, RFC4880, November 2007

[24] T. Freeman, R. Housley, A. Malpani, D. Cooper, W. Polk, RFC5055, Server-Based Certificate Validation Protocol (SCVP), December 2007

[25] Shahab Mirzadeh, Hossam Afifi, Jordi Jaen Pallares, Jasper Goseling, Jan Stoter, Final PN key management solution and Cryptographic techniques, IST-027396 project MAGNET Beyond Deliverable 4.2.2, June 2008

[26] Prakash Veeraraghavan, Vikram Limaye, Trust in Mobile Ad hoc Networks, Proceedings of the 2007 IEEE International conference on Telecommunication and Malaysia International Conference on Communications, 14-17 May, 2007, Penang, Malaysia

[27] Sameer Pai, Tanya Roosta, Stephen Wicker, Shankar Sastry, Using Social Network Theory Towards Development of Wireless Ad hoc Network trust, 21st International Conference on Advanced Information Networking and Application Workshops (AINAW'07), Vol. 1 (2007), pp. 443-450

[28] Testing and validation methodology, EU FP7 ICT project OPEN, deliverable D6.4, January 2009.

A Glossary

ACS	Auto Configuration Servers	NAT	Network Address Translator
ACL	Abstracted Connection Layer	NGN	Next Generation Networks
API	Application Programming Interface	NFC	Near Field Communication
APN	Application Port Number	NTP	Network Time Protocol
CA	Context Agent	OMA	Open Mobile Alliance
CALA	Context Access LAnguage	ONC NFS	Open Network Computing Network File System
CAM	Context Access manager	OSGi	Open Services Gateway initiative
CASF	Context-Awareness Supporting Framework	OWL	Web Ontology Language
CB	Context Bus	PACE	Pervasive Autonomic Context-aware Environments
CE	Consumer Electronics	PAN	Personal Area Network
CMI	Context Management Interface	PC	Personal Computer
CMN	Context Manager Node	PDA	Personal Digital Assistant
CN	Corresponding Node	POI	Point Of Interest
CORBA	Common Object Request Broker Architecture	PS	Packet Switched
CPU	Central Processing Unit	P&S	Processing and Storage
CPE	Customer Premises Equipment	PSL	PERSONA Specific Layer
CS	Circuit Switched	RTP	Real Time Protocol
CSCF	Call Session Control Function	RPC	Remote Procedure Call
CWMP	WAN Management Protocol	SB	Service Bus
DSAM	Data Source Abstraction Manager	SCMF	Secure Context Management Framework
DHCP	Dynamic Host Configuration Protocol	SCTP	Stream Control Transport Protocol
DLNA	Digital Living Network Alliance	SCVP	Server-based Certificate Validation Protocol
DNS	Domain Name System	SDP	Service Discovery Protocol
DS	Data Synchronization	SIP	Session Initiation Protocol
DSL	Digital Subscriber Line	SMS	Short Message Service
DVB	Digital Video Broadcast	SLP	Service Location Protocol
GPRS	General Packet Radio Service	SOAP	Simple Object Access Protocol
GSM	Global System for Mobile communications	SPL	SodaPop Layer

HA	Home Agent	SSDP	Simple Service Discovery Protocol
HGI	Home Gateway Initiative	STB	Set-Top Box for gaming application
HSS	Home Subscriber Server	TCP	Transmission Control Protocol
HTTP	Hyper Text Transfer Protocol	UI	User Interface
IKE	Internet Key Exchange	UMTS	Universal Mobile Telecommunications System
IMS	IP-based Multimedia Subsystem	UDP	User Datagram Protocol
IP	Internet Protocol	UPnP	Universal Plug and Play
IPTV	IP based Television	VCC	Voice Call Continuity
KPI	Key Performance Indicator	VoIP	Voice over IP
LAN	Local Area Network	WAN	Wide Area Network
LDAP	Lightweight Directory Access Protocol	WLAN	Wireless Local Area Network
MMS	Multimedia Messaging Service	XHTML	Extensible Hypertext Markup Language
MN	Mobile Node	XML	Extensible Markup Language
MIP	Mobile IP		