Context-aware Service Allocation in Personal Area Networks

JOHANNES JANSSON

KTH Microelectronics
and Information Technology

Master of Science Thesis
Stockholm, Sweden 2004

IMIT/LCN 2004-10

Context-aware Service Allocation in Personal Area Networks

JOHANNES JANSSON

Examiner
Prof. Gerald Q. Maguire
(IMIT/KTH)

Master of Science Thesis
Stockholm, Sweden 2004

2004.11.24

Abstract

Portable devices such as cellular phones, PDAs, and laptops, are getting more and more powerful and are popular with users. One way to make these devices even more useful is to interconnect them in a Personal Area Network (PAN), and via this PAN a single dual interface device can enable all local devices to access services in other networks. To be able to reach the devices in the PAN from outside in a simple manner, some location independent form of addressing may be used. One attractive approach is to use Session Initiation Protocol (SIP). For a user anywhere in the Internet, SIP enables them to reach a user within this PAN by an address that does not depend on where the user is located. To make life more convenient for the callee, the selection of device for an incoming multimedia invitation can be made automatically, by utilizing context information when selecting the device.

Consequently, a method is needed for allocating services to the devices that are most suitable for the service and for the moment. The current context in the PAN will affect this decision. This context information includes current status and capabilities of devices, the user, and their surroundings. The environment of a PAN is dynamic and thus this context information may change. This knowledge has to be reflected in the service allocation. This thesis will investigate and propose a method for how this service allocation is best performed.

To be able to investigate this problem and develop a suitable method, knowledge in several areas is needed. First of all an understanding is needed of the concept of “context” and how it can be used when making decisions about service allocation. Furthermore, a method for collecting and managing context information has to be used.

Sammanfattning

Portabla enheter såsom mobiltelefoner, PDA:er och bärbara datorer blir alltmer kraftfulla och vinner popularitet hos användarna. Ett sätt att få ut än mer av dessa enheter är att koppla ihop dem i ett Personal Area Network (PAN) och genom detta PAN låta alla lokala enheter nyttja tjänster i andra nätverk. För att nå enheterna i PAN:et från utsidan på ett enkelt sätt kan någon form av platsoberoende adressering användas. Ett attraktivt alternativ för detta ändamål är Session Initiation Protocol (SIP). Med hjälp av SIP kan en användare på Internet nå en användare i ett PAN genom en adress som är platsoberoende. För att ytterligare underlätta för användaren i PAN:et kan valet av enhet vid en inkommande multimediainbjudning ske automatiskt genom användande av kontextinformation.

Det behövs således en metod för att kunna allokera tjänster till de enheter som är mest passande för tillfället. PAN:ets aktuella kontext påverkar detta val. Denna kontextinformation berör aktuell status och funktionalitet hos enheter, användare och den övriga omgivningen. Miljön i ett PAN är av dynamisk karaktär där kontextinformationen ständigt förändras. Hänsyn till detta måste tas när tjänsteallokeringen sker. Denna rapport kommer att utreda och föreslå en metod för hur denna tjänsteallokering på bästa sätt kan genomföras.

Att undersöka detta problem och komma fram till en lämplig metod kräver insikt i en rad områden. Dels måste en förståelse kring begreppet ”kontext” och hur den kan användas som grund för beslutsfattning vid tjänsteallokering utvecklas. Vidare måste en metod för insamling av information och hantering av densamma tas fram.

Table of Contents

1. Introduction

1.1 Overview

Portable devices are gaining popularity among consumers as the devices are getting smaller, less expensive, and more user friendly; but at the same time many of these devices are more powerful than desktop computers a decade ago. The rapid evolution of cellular phones, PDAs, and laptops creates opportunities that haven't existed before due to technical limitations. Today there is no longer a problem, when it comes to processing power, to stream high quality video or play a multiplayer 3D-game on a device that easily fits into a pocket. Often a user has and actually uses several of these devices at the same time. To make these devices even more useful to the user they should interact and exchange information, not only internally, but also with devices attached to other networks.

By locally interconnecting the devices a Personal Area Network (PAN) can be formed. This network allows the devices to exchange information between each other and possibly also with external networks. Wireless communication is possible through the use of short-range radio, infrared, or ultrasound. Today Bluetooth™ [1] and Wireless LAN (WLAN) are the most important technologies for locally connecting these devices. Even though there are major differences in their design, these two wireless communication technologies can also be used side by side in a PAN. An important factor making this possible is the Bluetooth profiles, especially the PAN profile that can carry IP traffic. Furthermore, additional services can be accessed through a WLAN or PAN access point (AP) that is connected to an IP infrastructure as illustrated in
Figure 1.

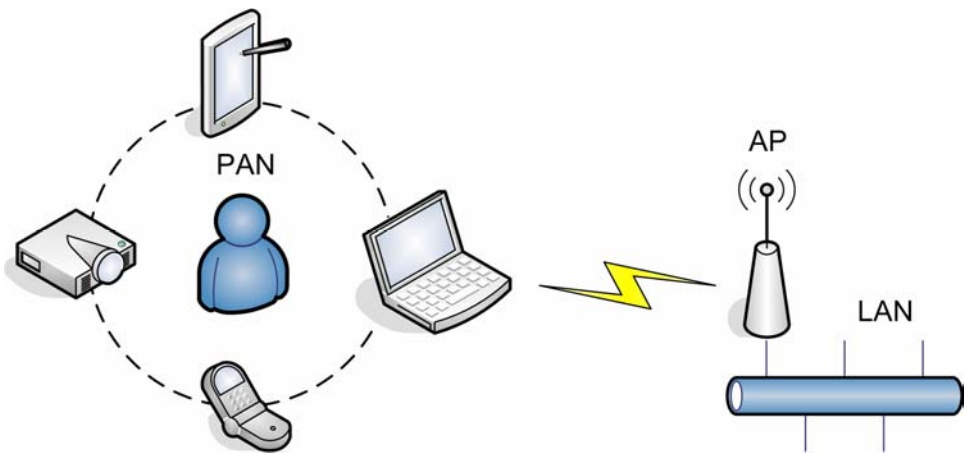

Figure 1. A PAN connected to a LAN through an access point.

The use of IP as the addressing protocol for the devices in the PAN has its advantages and disadvantages. IP is widely deployed, especially in fixed networks, which simplifies the communication between the PAN and existing networks. The problem with IP is the fact that it was originally designed for fixed networks and therefore addresses are topology dependent. IP assumes that the user resides at the same IP address during a communication session, and preferably between sessions as well. Consequently, to make IP feasible in environments such as a PAN, where the user and devices may frequently move and change, IP needs additional functionality, i.e. Mobile IP [2].

Even if the problem of addressing devices can be solved at the network layer there are still difficulties with finding and contacting specific users. If for instance the IP address is used for finding a user this address has to be fixed, but in several cases, for instance when using dynamic addresses this is not possible. Consequently, some additional technique is needed. This technique should support the ability to contact a user independent of the IP address of the user's (current) device. Using the Session Initiation Protocol (SIP) [3] to find and contact a user is one of several possible methods.

SIP offers much of the functionality needed in a dynamic environment where users come and go and their IP addresse changes over time. Users can be anywhere on the Internet and still be reachable through their SIP address. One or more devices can share the same SIP URI and the user can be invited to and participate in several sessions at once. SIP is independent of the underlying transport protocol and the type of session that is to be established. In this thesis SIP will be used in several different ways. For the rest of this thesis we assume that an incoming request is a SIP request and that SIP can be used for internal communication when the allocation of the service to the SIP enabled devices takes place.

1.2 Problem statement

In this thesis we will look at the problem of forwarding multimedia invitations or allocating incoming service requests to the best suited devices in a PAN. The context of the entities in the PAN will be used when deciding if a given device is suitable or not. In some cases there will only be a single obvious choice, while in other cases several devices may be suitable and thus some selection process is needed in order to select the *best* alternative.

In order to make these decisions, knowledge about the current context of the entities is needed. The environment of the PAN is assumed to be dynamic and as a result the context will change. The information used when making the allocations has to reflect this property, and the context information should be sufficiently up to date to make a suitable allocation.

When new information is available there must be a way to utilize it. Existing information will be combined with the information in the incoming request to decide where to allocate the service. Consequently, up to date context information and some decision-making logic have to be available in order to select, for the service, the best suited device(s).

As mentioned above, appropriate context information is needed by the allocation process at the right time. The process determines which information is collected and when it is collected. The question of *when* the information is collected depends on the performance requirements for the service allocation and the delays associated with the information collection and decision process. If the latency when communicating with the information source is high and the performance requirements of the allocation process are strict it may not be possible to perform the collection of information *when* the service request arrives.

If the collection of information is made in *advance*, possible restrictions on the amount of traffic generated must be taken into consideration. If communication in the network is expensive, the frequency of the information collection has to be low. However, this may conflict with the requirement for up to date information. If the information is not up to date, the allocation may be suboptimal, wrong, or in the worst case fail altogether.

The question of which information should be collected is fairly interesting as well. One method is to collect as much information as possible about the entities that *may* be involved in the allocation. By doing this, there will never be a lack of information, but the cost may be excessive. Another approach is to only collect relevant information. Although this second alternative may appear much more attractive, the problem of deciding what information is *relevant* has to be solved. This is especially hard to determine before you know what the service request is.

When a service request arrives it has to be analyzed to determine its requirements. This information is to be combined with the context information by some decision mechanism. A very detailed request may implicitly or even explicitly specify a certain device, while a less detailed request might not specify a specific device. If more than one device is capable of serving the request, several factors should be used when performing the allocation. These factors may include status of devices and users, device capabilities, and user preferences.

The issues above require the design of a service allocation system capable of collecting, extracting, and analyzing context information and taking appropriate actions. It is the goal of this thesis to design, implement, and evaluate such a system.

2. Background

To be able to fully understand the problem addressed in this thesis, some background information may be needed. In this section we will introduce basic definitions and some of the technologies essential for understanding the rest of this document.

2.1 Context information

First of all we need to define the term context. Many definitions exist and you may get a new definition from each person you ask. In this thesis context is the status and capabilities of an entity and its surroundings. The context of a user will not necessarily be the same as the context for the user's cellular phone. However, the context of the user's cellular phone will affect the context of the user.

If context information can be used when allocating services to devices in a PAN many new possibilities arise. In the best case, the user not longer has to bother with static settings that control how and when a certain device should be used for communication. This can be decided by some node attached to the network, based on context information.

2.1.1 Context awareness

Context awareness is based on using context information in order to make decisions. This translates into taking different actions depending on the context of a device or a user. Depending on the characteristics of the context information that is used, the decisions may change frequently or infrequently. This is especially true if the context information is detailed and fluctuating.

There are many different kinds of context information. Context information might describe the ambient light level or the type of display of a certain device. This information may at first look completely uncorrelated with what the user would like, but the combination can actually be useful. To illustrate this we will consider a simple example.

Imagine a user equipped with two devices where one device has a Thin Film Transistor (TFT) display and the other has a Super Twisted Nematic (STN) display. The information on the STN display is unfortunately not clearly visible in bright sun light. Consequently it is better to receive visual information, such as a video stream, on the device with the TFT display if the context information about the ambient light level indicates that the user is outside and the sun is shining where the users' STN display is.

2.1.2 Categorization of context information

Some of the context information describes the device and its capabilities. Examples are bandwidth of the network connection, communication cost, available memory and available storage space. Other context information may describe more abstract information, such as patterns of user behaviour. Below are more examples of context information given and categorized by their relation. This division was proposed by Chen and Kotz [4].

Device context	network connectivity, available bandwidth, available memory, processing capabilities, battery level, display, communication cost, load, running processes, nearby resources
User context	user name, user ID, user profiles, heart rate, respiration rate, muscle activity, emotional state, cognitive load, nearby users
Physical context	ambient light level, noise level, temperature, humidity, air quality, location, orientation, speed, acceleration
Time context	date, time, time zone, time source(s)

2.1.3 Collecting context information

The entities used for collecting the data that is later transformed into context information are called sensors. A sensor may be a physical device or implemented in software. For measuring temperature some kind of physical thermometer is required, such as a thermistor, equipped with an interface readable by a computer and driver software. On the other hand a sensor monitoring available user memory space in a device can be completely implemented in software. Some sensors reside in a device and monitor the device or the surrounding environment. The memory sensor mentioned above is a good example of a sensor found inside a device. Other sensors are placed outside the device, for example a thermometer placed on the wall in an office where the device is.

The data delivered by the sensors needs to be interpreted, managed, and expressed in some way to be useful for applications. Furthermore, the context information may need to be stored somewhere and there must be some mechanism for exchanging context information between devices. The information relevant to a device could be stored locally or remotely. If the former approach is used, popular information will be duplicated. On the other hand, if the latter is used more communication is required. These issues, amongst others are being investigated in the ACAS project [5] (see section 3.5).

2.2 Personal Area Networks

A Personal Area Network (PAN) is a set of devices near the user that are interconnected in some way. These devices may all be used to provide the user with some service. Examples of such devices are PCs, PDAs, printers, storage devices, cellular phones, and a variety of consumer electronics equipment. The difference between a PAN and a Local Area Network (LAN) is that the former is centred around one person, while the latter generally serves multiple users as it covers a larger area.

A PAN may be a convenient way to synchronize and share data between personal devices. However, the idea of a PAN becomes much more attractive when the PAN is no longer isolated. Consequently, to make the PAN more useful it is often connected to an external network. Via this connection it is possible to interact with devices and users outside the PAN. Furthermore, the PAN may change depending on the environment. If a device nearby the user is allowed to join the PAN, preferably in a dynamic fashion, the user can take advantage of services provided by devices that are not always in direct possession of the user. For example, if the user enters a room with a projector, the user might now have the possibility to receive a video call utilizing this device. Accordingly, the PAN may logically span over devices that resides inside a LAN when a LAN gateway is within the PAN.

The PAN concept isn't that useful if the devices need cables for communication. Consequently, some wireless technique to connect the devices is needed. Several options exist. Today, the two most attractive are Wireless LAN (WLAN) and Bluetooth. A third, Zigbee [6], is just staring to appear.

In this thesis, the actual technique for transferring data or forming networks is **not** very interesting. Instead, the focus is on what higher level services the communication technologies can support. Even if we do not care about how the data is transported, it is crucial to understand that it will indirectly affect the service allocation. For instance, it is not possible to stream high quality video to a device that can only receive data at a maximum rate of 60Kb/s. We will consider our underlying networks to be IP over Bluetooth™ or WLAN.

2.2.1 Bluetooth

Bluetooth™ allows devices within close proximity to join an ad hoc wireless network. The strength of Bluetooth is the simplicity of forming an ad hoc network. Two types of networks are possible in the Bluetooth architecture: Piconets and Scatternets.

A **Piconet** is formed by a master that periodically emits requests. The slave will answer with its identification number. The Piconet consists of up to eight devices. To allow more devices in the Bluetooth network either the Piconets can be interconnected and form a **Scatternet** or devices have to share the Piconet by switching between an active state and some other state. In theory, up to 10 Piconets can overlap without excessive interference and moreover different Scatternets can be linked together forming a linear chain.

Some comments need to be made about the available bandwidth. Bluetooth devices can work in both circuit switched mode and packet switched mode. If the latter is used, asynchronous communication is possible where the maximum bandwidth is 712Kb/s in one direction and 57.6Kb/s in the opposite direction [1]. A more balanced alternative is 433.9Kb/s in both directions. However, these rates are theoretical upper bounds and are unlikely to be achieved in practise.

2.2.2 Wireless Local Area Network

A Wireless Local Area Network (WLAN) is another way to connect and transfer data between the devices. Basically, a WLAN is a LAN using radio waves in free space. A WLAN can be implemented in an infrastructure-based or an ad hoc network fashion. The former requires an access point and resembles a traditional LAN. The high bandwidth of a WLAN makes it a good complement to Bluetooth. By supporting data rates up to 54Mb/s in IEEE 802.11g [7] a WLAN enabled device has the ability to support services with high demands on bandwidth. However, higher bandwidth and large coverage area come at the price of greater power consumption.

2.3 Service allocation and Service Discovery

In this thesis service allocation is the act of forwarding an incoming request to the most suitable device. “Most suitable” refers to the device that has the best chance to successfully serve this specific request in a manner which will satisfy the user or process. A request is normally addressed to a specific user, but may be delivered to one or more devices, accessible to the addressed user. Note: Services could also target specific devices, but we will view this as a subset of requests to the users.

The most important criteria for how successful a service allocation can be is up to date context information. Of course the decision logic needs to be capable of making proper decisions, but even if the service allocation system knows how to analyze and combine information, the decisions may not be accurate if this information is obsolete.

Service allocations may vary in many different ways. Some allocation decisions are very simple, while others can be fairly complex. The type and amount of information needed by the allocation algorithm and the complexity of the algorithm itself will affect the delays associated with a service allocation. To give the reader a better understanding of a service allocation process and what problems may arise when doing one, we will consider three example service allocation algorithms. These service allocation examples will later on be used to describe algorithms and demands on the information collecting process.

Service allocation A is an allocation algorithm which allocates the incoming service to the device that has not a certain process running. The reason is that if there is a process running on the device that is untrusted the device should not be allocated the service. Consequently, information about the running processes on all devices is needed and this information has to be processed for every available device in the allocation algorithm. Furthermore, the available bandwidth will also affect the choice of device.

Service allocation B allocates an incoming video request to the device that can best display video and has the most battery capacity left. This allocation depends on information both from the devices in the PAN and the incoming request. The information that will be used is listed below.

Ambient light level – Indicating the current light level (the bigger value, the brighter the ambient light).
Display size – The size of the display (the bigger, the better).
Display type – The type of display. Some display types are easier to watch than other depending on the ambient light level.
AvailableBandwidth – The current available bandwidth. More bandwidth allows higher bit rate CODECs.
Capabilities – The capabilities of the device (indicates if it is capable of receiving a video stream).

The information needed from the incoming request is the media descriptor (see section 2.5.1) that tells us about the requirements of the service.

Service allocation C may allocate several devices in the PAN. This example assumes that the Context Aware Service Allocator (CASA) receives a service request that has more than one desire when it comes to media capabilities (both audio and video capabilities are requested). More concretely this means that there is more than one media descriptor in the incoming request. The service will be allocated to the device or devices that can best play the audio and video streams. Screen size and the number of speakers will be used to decide which devices to use.

It is necessary to be aware of the available services in the network to be able to contact them and ask for information. In this thesis we will assume that the services in the network already are discovered. How this was done is not of interest here, it was addressed in the earlier report of Cecile Ayrault [8].

2.4 Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) is an application layer protocol that is capable of establishing and managing multimedia sessions. The session may have two or more participants, also called endpoints or User Agents (UAs). The UAs may come and go during a session, just as the media used in the session may change. SIP is designed to be generic and is independent of the underlying transport protocol and type of the session. One of the strengths of SIP is that the user only needs one address or more importantly, the rest of the world only needs to associate one Uniform Resource Identifier (URI) with a specific user. This single URI will be mapped to the user’s current location.

2.4.1 Basic concepts

A brief description of SIP can be further distilled into five basic concepts, namely: user location, user availability, user capabilities, session setup, and session management. User location deals with the problem of finding the end system that is to be invited to participate in a session. User availability ensures that the user at that end system is actually interested in participating in a session. User capabilities are the media and media parameters that could be used in the session. Session setup consists of session parameter establishment at the involved parties and the invitation to start a session. After the session is initiated it can be modified and eventually terminated. These later activities are included in session management.

A SIP architecture consists of a number of entities. The architecture includes four servers: the Location server, the Proxy server, the Redirect server, and the Registrar server. The Location server keeps information about the current location of an end system and is used by the Redirect and Proxy servers. Note that the Location server does **not** speak SIP. The Proxy server is responsible for routing and delivery. The Redirect server is used for finding a user. It takes the URI in a SIP request and maps it to zero or more UA addresses. These addresses, if any, are returned to the client. The Registrar server manages UA registrations. A UA needs to register at least one address before it can be used. Thus the Registrar server is consulted to locate registered users. Note that the servers are logical roles that can be played a single device. A common design [8] is to combine the Location, Redirect, and Registrar servers into a single server, simply called a Redirect server. The UA acts both as a server (UAS) and a client (UAC), depending on whether it is receiving or sending a request (respectively).

2.4.2 Typical example

To get an idea of how a session is established in SIP we will start with a simple example, shown in Figure 2. Assume that a user, Alice, is interested in talking to another user, Bob, by using her soft phone. Fortunately, Bob has a SIP capable device as well. To call Bob, Alice uses his SIP identity. This has the form of an *address-of-record* (AOR) URI that is unique to this user. The AOR can be used as the public address of a user and resembles an email address with a user name and a host name. Actually, the host name often simply is a domain name, it is then up to the DNS system to provide the correct host address(es) for the SIP server (that may map the AOR to another URI where the user might be available). Bob's AOR is sip:bob@bobobert.com. This address is all Alice has to know to contact him. Her own address is sip:alice@alicenter.com.

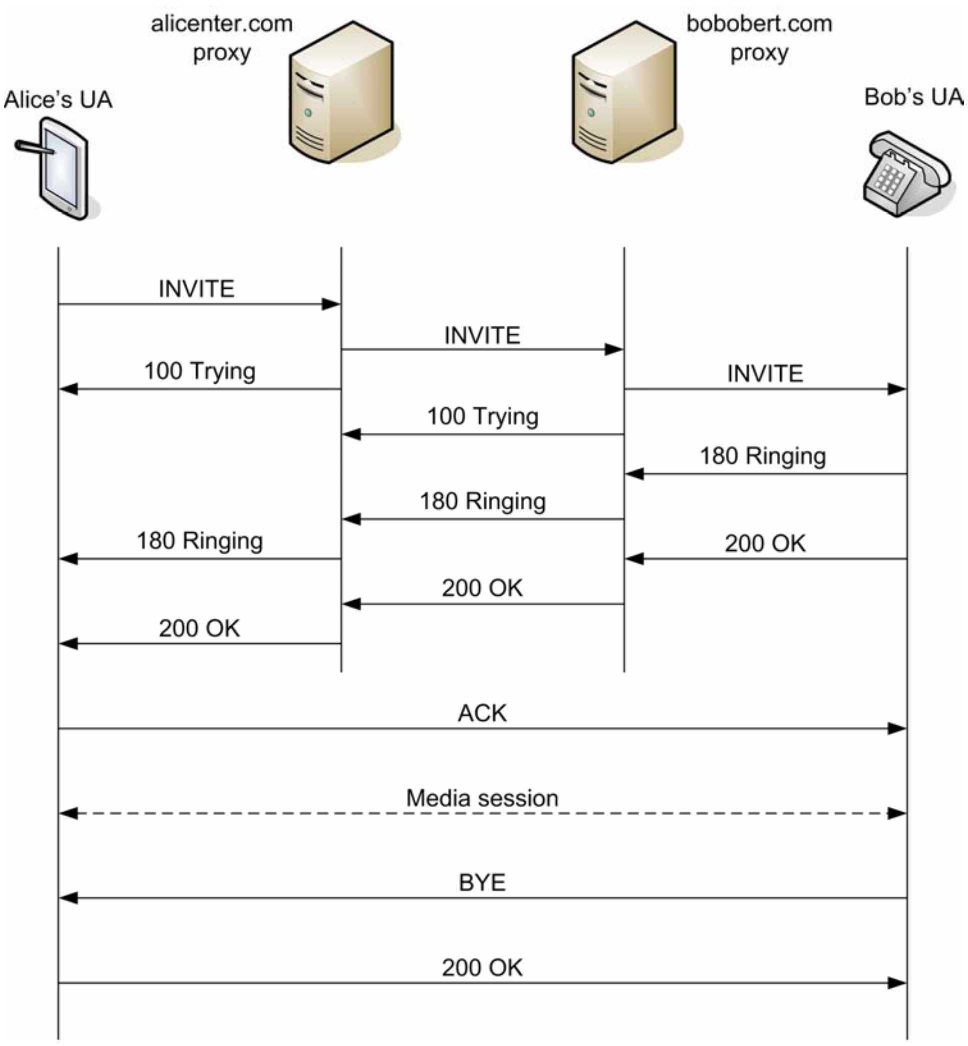

Figure 2. Typical example of SIP message exchange.

The only entity able to create an original request is the UAC. Accordingly, it is Alice's UAC that initiates a session by sending an INVITE message to the end system it wishes to communicate with. In the scenario considered in Figure 2, the INVITE message from Alice will be sent to an outgoing proxy server in the same domain, namely alicenter.com. This proxy will forward the message to the next hop, which in this case is the proxy server in domain bobobert.com. A 100 Trying message will be sent to Alice to inform her that the INVITE message was successfully received by the network and no retransmission is required. Other messages will be forwarded without any acknowledgement. The bobobert.com proxy server will forward the INVITE to Bob's UA. Upon receiving the INVITE message Bob's phone will start ringing and a 180 Ringing message will be returned to Alice. When the user at Bob's phone (hopefully Bob himself) answers the phone a 200 OK response message is sent to Alice. After both the 180 Ringing and 200 OK messages are received an ACK message is returned by Alice. This acknowledgement completes the three-way handshake. Media streams can now be established between Alice and Bob. When any of the participants, say Bob, wants to end the session and hangs up, the phone will send a BYE message to Alice's phone which responds with a 200 OK. After this message the media streams for this call are torn down.

After this simple example we are ready to look at what happens behind the curtains, i.e. what the messages look like and what other entities have to be involved.

2.4.3 SIP Message structure

To get an idea of how a message in SIP is constructed we will look at the INVITE message from the example above (see Figure 3). Here only the most common fields are discussed.

```
INVITE sip:bob@bobobert.com SIP/2.0
Via: SIP/2.0/UDP
pc123.alicenter.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@bobobert.com>
From: Alice <sip:alice@alicenter.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc123.alicenter.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc123.alicenter.com>
Content-Type: application/sdp
Content-Length: 142

/* SDP content */
```

Figure 3. An INVITE message (the SDP content is not shown).

The first line specifies the message type ("INVITE" in this example). The "INVITE" is followed by the address of the addressee and the version of SIP that the caller supports.

`Via` – specifies the transport protocol, the client's host name or network address (and optionally port number), and a branch value that uniquely identifies the transaction and is used by the proxies for loop detection. The history of the message's path through the network will be available in this field, it is added to by each proxy.

`Max Forwards` – inserted by the UAC to limit the number of hops a request can traverse on the way to the destination. The recommended default is 70.

`To` – contains the address of the recipient of the request. In the example above "Bob" is a display name (more convenient for the users) and thereafter follows the URI. Both SIP and SIPS URIs may be used. The latter is further explained in section 2.4.6.

`From` – indicates the initiator of the request with a display name and a URI. A tag is included to uniquely identify the dialog.

`Call-ID` – a globally unique identifier for this call. The identifier is generally based upon a random number and the host name or IP address of the UA.

`CSeq` – CSeq or Command Sequence is a sequence number incremented for each new request within a dialog. The request method name is also included. This header field serves to order transactions within a dialog and can be used for differentiating requests and retransmissions. When an ACK message or a CANCEL message is sent as a response to an INVITE message, the CSeq number is the same as in the INVITE message.

`Contact` – contains a SIP or SIPS URI that points directly to the sender (Alice in the example). The address is composed of a user name at a fully qualified domain name (FQDN). An IP address may be used if the end system does not have a registered domain name.

`Content-type` – the media type of the message body. In this example Session Description Protocol (SDP) [10] is used. See section 2.5 for more information.

`Content-length` – the size in bytes of the message body.

2.4.4 SIP Request messages

SIP requests can only be created by the UAC. The request is sent to a server that takes appropriate actions depending on the content and type of the request. All the request messages except the ACK message are followed by a response.

INVITE – As described in section 2.4.2 the INVITE message is the sent by the UAC to initiate a session. The message contains information about the caller, the upcoming session and of course the address of the addressee. A session is established after a three-way handshake is completed, where the INVITE message is the first message.

ACK – The ACK message is sent by the UAC to confirm that the response to the INVITE request, namely the 200 OK, was received.

BYE – The BYE message is sent by a client to terminate an ongoing session.

CANCEL – Cancel a previous request. If a session is being cancelled before it is established this is the message that should be sent. Please note that the BYE message shouldn't be used in this case. If a final response for the request already has been received, the CANCEL message will have no effect. CANCEL requests can be made by both proxies and UACs.

REGISTER – The message used for registering a user. The address in the "To" header will be associated with the user. See section 2.4.7 for details about the registration process.

OPTIONS – The OPTIONS message can be used for discovering information about another UA or a proxy (see section 2.4.9 for details).

2.4.5 SIP response messages

The response message sent by a server indicates success, failure, or provides the client with more information that may make the request successful. Many different response messages exist in SIP and they are grouped into six different types. Each response is identified with a three digit number where the first digit is the response type.

1xx: Provisional – the request was received by the server, but further processing is required. This message should be sent if the request is expected to take longer than 200ms to process, e.g., 100 Trying.

2xx: Success – the action was successfully received and accepted, e.g., 200 OK

3xx: Redirection – gives information about the callee´s new location or, if the call wasn’t successful even though the address was correct, possible alternative services, e.g., 302 Moved Temporarily

4xx: Request Failure – the client has sent a request that contains bad syntax or cannot be fulfilled at this server and the client should modify the request or try another server, e.g., 404 Not found

5xx: Server Failure – the request was valid, but the server was unable to process it successfully, e.g., 501 Not implemented

6xx: Global Failure – the contacted server has enough information to claim that the request cannot be fulfilled anywhere in the system, e.g., 600 Busy everywhere

2.4.6 SIP URI

As briefly discussed in section 2.4.2 the SIP URI identifies the communication resource. This address is sufficient to initiate a session with the resource. The general form of a SIP URI is given below.

```
sip:user:password@host:port;uri-parameters?headers
```

The token “sip:” specifies that the URI is a regular SIP URI. There are two types of URIs in SIP, namely SIP and SIPS where the latter provides additional security. When a SIPS resource is contacted, the path from the initiator to the target domain should be secured. In practice this means that either Transport Layer Security (TLS) [11] or Secure Sockets Layer (SSL) [12] is used for protecting the data.

“`user:`” the identifier of the particular resource at the host.

“`password:`” the user’s password. However, this field is not recommended to be used due to the fact that it is sent in clear text.

“`host:`” the IP-address or domain name where the resource is located.

“`port:`” this field may optionally specify a port number where the request should be sent. If not specified, then the default port will be used (5060 for SIP and 5061 for SIPS).

“`uri-parameters:`” an arbitrary number of parameters to the request, e.g., `transport=tcp`

“`headers`” hold the values that should be present in the header of the request constructed from the URI, e.g., `subject=meeting`

2.4.7 Register

To participate in a session the UA needs to register its contact URI at the Registrar server. Normally, this is the first thing that is done when a SIP enabled device, e.g. a SIP phone, comes online. For this purpose the Registrar server is contacted. Registration is initiated by the UAC by sending a REGISTER request, either directly to the Registrar server or via a Proxy server. If the registration is accepted the server will store a binding between the URI and one or more contact addresses. The Registrar server will respond with a 200 OK message if the registration was successful. The same REGISTER request message is used when the user wants to update an address, get the addresses stored on the server, or delete an address.

The registration will expire after a certain time specified by the local policy. However, the client can suggest an expire interval in the REGISTER request. Because a binding is deleted when the registration expires it has to be renewed. The client learns the current expire interval in the 200 OK message from the registrar server. Now it is up to the client to renew the bindings before they expire, if the client wishes to be able to receive incoming invite requests.

Before the REGISTER message can be sent the Registrar server has to be located. Several alternatives exist, such as manual configuration, DHCP, or DNS SRV Resource Records.

2.4.8 Redirect

In the example in section 2.4.2 the proxy forwarded the message to the next proxy or UA. Consequently the proxy server needs to know how to reach the next hop. However, this is not always the case. If the proxy hasn’t contacted the next hop before and thereby learned the address, a Redirect server may be contacted.

To get the address of the next hop the proxy forwards the INVITE message to the Redirect server. This server will look up the destination in its table of registered users. If the callee can’t be found, then the server consults a dialling plan to determine where the INVITE should be sent. As mentioned earlier, the redirect server works closely together with the Location and Registrar servers. The Registrar server stores registration information at the Location server. The latter serves the Redirect server by providing information about where a user currently is located.

The Redirect server returns routing information about how to reach the next hop to the proxy in a 302 Moved temporarily message. This message will be acknowledged by the proxy. The original INVITE message is then modified according to the routing information and sent to the next hop in the network.

Figure 4 shows how the messages flow in the system when SIP phone A is trying to contact SIP phone B with an INVITE message. Here we use a SIP address to address a specific phone rather than a user.

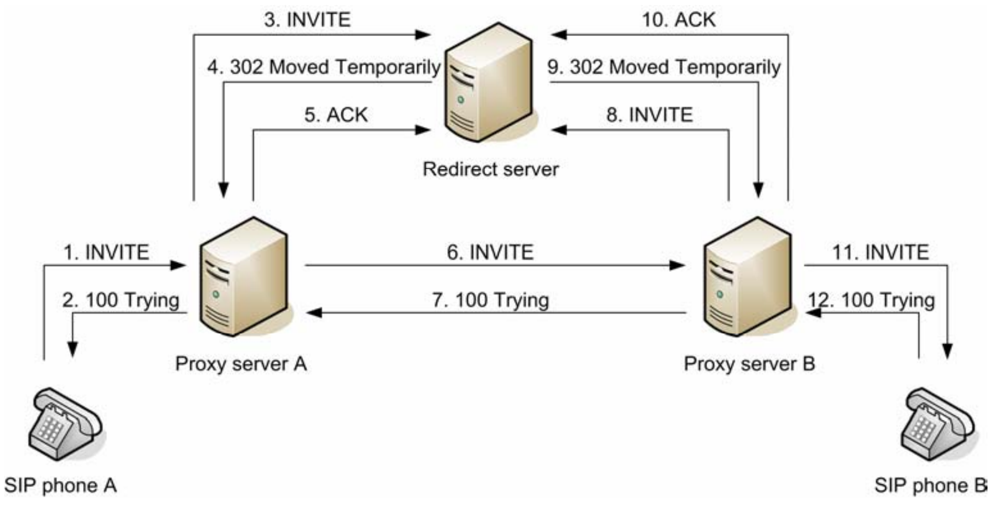

Figure 4. An INVITE message is sent from SIP phone A to B.

2.4.9 Determination of capabilities

There exists one mechanism in SIP that allows a client to query a UA or a proxy server for its capabilities. This special request is called OPTIONS and makes it possible to get information about supported methods, content types, extensions, CODECs, etc. without sending an INVITE message to the server.

When the server receives an OPTIONS request message it should return a 200 OK message containing the information queried by the request. *Allow*, *Accept*, *Accept-Encoding*, *Accept-Language*, and *Supported* header fields should be present in the response message.

Allow lists the methods that are supported by the server. Note that this field should not be included if the requested server is a proxy since a proxy is method agnostic. *Accept* specifies the supported formats of the message body. *Accept-Encoding* and *Accept-Language* gives information about supported encodings and languages. *Contact* header fields may be present in the response to list alternative addresses for reaching the user.

Furthermore, a body may be attached to the response. The type of the body is specified in the OPTIONS request message. The default body type is "application/sdp" and if this type is present the server should include a body with a listing of media capabilities.

An example of a 200 OK message sent in response to an OPTION request is shown below.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP
pc123.alicenter.com;branch=z9hG4bKhjhs8ass877
;received=192.0.2.4
To: <sip:bob@bobobert.com>;tag=93810874
From: Alice <sip:alice@alicenter.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:bob@bobobert.com>
Contact: <mailto:bob@bobobert.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Accept: application/sdp
Accept-Encoding: gzip
Accept-Language: en
Supported: foo
Content-Type: application/sdp
Content-Length: 274

/* SDP */
```

Figure 5. Example response to an OPTION request (the SDP body is not shown).

2.5 SDP

The Session Description Protocol (SDP) [10] describes multimedia sessions and is mainly used for session announcement and session invitation. It is a simple protocol and accordingly *negotiation* of session content or media encodings are not supported, rather it is an offer-accept model. However, it is possible to carry other protocols inside SDP. This feature is for example used when distributing media stream encryption keys in a secure manner, as used in MIKEY [13].

SDP is intended to be general purpose and can be used in many different applications. The most common one, and also the one we find the most relevant, is when it is used together with SIP to establish a media session. Here the SDP information is carried inside the SIP INVITE message and specifies for instance which CODECs should be used and between which IP addresses and ports the communication should take place. The other party will respond with acknowledgments of the descriptions that it accepts.

2.5.1 Protocol format

An advantage of SDP is that it is very simple, both to understand and to implement. The information carried by the protocol is coded in UTF-8 text format. Different letters are used as labels to identify the fields. The defined protocol structure is as follows. Optional items are marked with a '*'.

```
    v=  (protocol version)
    o=  (owner/creator and session identifier)
    s=  (session name)
    i=* (session information)
    u=* (URI of description)
    e=* (email address)
    p=* (phone number)
    c=* (connection information)
    b=* (bandwidth information)
    z=* (time zone adjustments)
    k=* (encryption key)
    a=* (zero or more session attribute lines)
```

Time description

```
    t=  (time the session is active)
    r=* (zero or more repeat times)
```

Media description

```
    m=  (media name and transport address)
    i=* (media title)
    c=* (connection information)
    b=* (bandwidth information)
    k=* (encryption key)
    a=* (zero or more media attribute lines)
```

The SDP parser expects that the labels appear exactly in the order specified above. It is possible to have several SDP session descriptions in the same packet. The ‘v=’ field marks the start of a new session description.

2.5.2 SDP example

An example of a SDP description is given in Figure 6.

```
    v=0
    o=- 4711 4711 IN IP4 130.237.251.216
    s=Example Session
    c=IN IP4 130.237.15.216
    t=0 0
    m=audio 1061 RTP/AVP 0
    a=rtpmap:0 PCMU/8000/1
    m=video 1062 RTP/AVP 31
    a=rtpmap:31 H261/90000
```

Figure 6. A typical SDP body.

The following is an explanation of the less intuitive lines in the example.

‘`v=`’: The version number is always set to zero.

‘`o=`’: The owner/creator and session identifier. The syntax is o = <username> <session ID> <version> <net type> <address type> <address>. In this example the session ID and the version are the same. The username in the example is “-“, which is a placeholder for the real name.

‘`t=`’: The time the session is active. The syntax is t= <start time> <stop time>. For unbounded time values 0 can be used, as in the example.

‘`m=`’: The media name and transport address. The syntax is m= <media> <port>/<number of ports> <transport> <fmt list>. When the transport is RTP/AVP the <fmt list> is a list of integers specifying the CODEC that can be used. The list can be found in [14].

‘`a=`’: Session attribute. The syntax for the rtpmap is a= rtpmap: <payload type> <encode name>/<clock rate>[<encode parameters>]. In the example we use PCMU (G.711 µLaw) with a sample rate of 8 KHz. Several other attributes may be used [10].

3. Related work

The problem of allocating services utilizing context awareness is similar to other problems, such as policy management, access control, and request distribution. In this section we will look at related work that may be of interest when considering context aware service allocation.

3.1 Service Policy Management

In his thesis ”Service Policy Management for User-Centric Services in Heterogeneous Mobile Networks” [15], Konstantinos Avgeropoulos looked into the issue of performing policy based management of a network with several UAs. The UAs form the user’s personal service network which could be for instance a PAN. An entity called a SIP Service Manager (SSM) was proposed for monitoring the service network and issuing polices. The SSM resides between the service network and the SIP backbone. Apart from the policy related components it combines proxy, UA, registrar, and redirect functionality.

Avgeropoulos´ work is relevant to this thesis for several reasons. The problem of doing service policy management is related to service allocation. First, information in the request will be used together with information about the entities when issuing policies. The architecture for this process may be of interest for this thesis. Furthermore, Avgeropoulos introduces mechanisms for collecting information about the devices in the service network and also his use of SIP INVITE request multiplexing that may be interesting. However, his information collecting mechanism is based on polling which has drawbacks due to its communication overhead and this leads to scalability problems.

3.2 Content-based request distribution

Web-servers are often arranged in clusters on the Internet to distribute load and to increase service reliability. To distribute the incoming requests a so called Web switch is used. Consequently, all requests are sent to the same IP address. A Web switch can be implemented as a layer-4 or layer-7 switch. A layer-4 switch can only use the information in the transport layer for dispatching a request to a server. Things get more interesting when we consider a layer-7 switch. Here the content of the request can be used when dispatching the request. By using information about the servers together with the information acquired by analysing the request, the switch is able to perform content-based request distribution. The advantages when used in such web-server clusters is increased hit-rates in memory caches, increased storage scalability, and the ability to have specialized servers for certain types of requests, such as video or audio.

When designing content-aware request distribution in general the performance of the web switches is the main issue. The request rate is often high and if too much processing is needed for each request, then the switch will be a bottleneck. Therefore the decision logic has to be simple and not depend on too much data. In this thesis, the entity that will allocate the service to the proper device will most likely not experience a request rate as high as in the server cluster.

3.2.1 Locality-Aware Request Distribution (LARD)

LARD [16] is a specific policy for content-based request distribution. The main goal is to achieve high locality in the main memory of the servers. This means that when a specific object is requested the same server *should* handle that request each time. An object (and request) consequently are associated with a specific server. Of course some load balancing needs to occur, but that is not of interest in this thesis as it is likely that the best available device should be used as long as it can, then the next best, and so on.

Even though LARD and other content-aware request distribution systems primary consider performance, the basic mechanism enables efficient service allocation. The incoming request has to be analyzed, and the content of the request will affect where the request will be forwarded. Information about the current state of the servers has to be available when request distribution is performed. This is the basis also of the problem considered in this thesis. While the reliability of the servers in a cluster can be considered high, the same assumption cannot be made about the devices in a PAN. Furthermore, the front-end system and the servers are physically connected, usually with high bandwidth connections. On the contrary, the communication in a PAN can be expensive and in such cases should be kept to a minimum.

3.3 Context-aware access control

Another area where context information may be helpful is access control. Using context information can be a powerful tool when making access control decisions. In [17], a context-aware access control model for Web services is presented. By looking for example at the system resources, current time, and location of the user the authorization mechanism now has the ability to dynamically grant and adapt permissions of a user based on their current context.

Furthermore, the paper proposes a tuple language to express context information, and algorithms to evaluate service access requests. Access patterns may be used when making access control decisions. If something differs from the normal pattern of a specific user, fewer privileges are given to that user.

The ideas introduced in the context-aware access control system may also be useful in this thesis, especially when it comes to expressing context information internally and designing algorithms to evaluate the incoming requests. However, context awareness in the access control system concerns the context of the entity that **makes** the request, while in this thesis we are primarily interested in the context of the entity that **receives** the request.

3.4 Heterogeneous wireless network management

Power consumption is often a critical issue in portable devices running on batteries. In [18] a policy for selecting the network interface that consumes less power, but still serves the needs of the application is presented. Two network interfaces were used, one Bluetooth interface and one WLAN 802.11b [7]. The Bluetooth interface provides low power consumption, but only moderate bandwidth, while the WLAN 802.11b interface offers high bandwidth, but consumes more power. If only a small amount of bandwidth is needed by an application (for instance when sending a text based e-mail) the Bluetooth interface can be used. For applications with higher bandwidth demands, such as when streaming video, the WLAN 802.11b interface is needed. The mechanism for selecting interface monitors the variations of application data consumption rate and dynamically changes interface if needed.

Interesting for this thesis is to see both how bandwidth affects a service and how the need of an application (type of service) affects the choice of a device. The two types of interfaces considered in the paper are frequently used in PANs, which makes it highly relevant for this thesis.

3.5 Allocating Web-Services

When allocating Web-Services performance is an important issue. High volume Web-Services are often replicated over several servers to spread the load over different machines and to provide redundancy. An example of a Web-Service that spans over multiple machines and where performance is central is the search-engine Google^{TM} [21]. A Web-Service can be allocated randomly or by looking at the status of the Web-Service and/or the hosting nodes. A node hosting a Web-Service replica may host copies of other Web-Services as well; which of course may impact performance. In [20], different schemes are proposed for allocating a Web-service. The most interesting schemes are Least Utilized and Least Response Time. Here, the status of the Web-Service has to be monitored and taken into consideration when allocating new service requests. The information describing the status of the Web-Service and the hosting nodes are called Performance Metadata. This status information is stored in a distributed registry that is consulted when a Web-Service allocation is about to be performed. Each node and Web-Service will update its utilization or response time in the registry either periodically of after a request is processed. The more often this information is reported, the more accurate allocation is possible. However, the reporting process implies additional processing overhead at the node.

The problem of doing successful allocations in a Web-Service has similarities with doing service allocations in a PAN. The more sophisticated allocation schemes, namely Least Utilized and Least Response Time, shares the problem of keeping remote information up to date. When allocating Web-Services a suboptimal allocation will result in longer service times. In our system a suboptimal or faulty allocation may result in a session not being established.

3.6 Adaptive and Context-Aware Services (ACAS) project

The ACAS project [5] considers the difficult issues of sensing, processing, and distributing context information. It has been shown that there is a need for both a context description language and architecture to manage the context information. There is a gap between the raw information given by the sensors and the information needed to make suitable conclusions about a certain condition. This gap is called a context gap. Filling the context gap makes the context information more useful. Combining different information may create new information that is much more useful than the original information. This process is called *context refinement*.

The ACAS project proposes a language for exchanging and processing context information called *Context Description Language* [5]. This is a XML [19] based language for describing context where the context entity is the central focus.

3.6.1 Architecture

The central concept in the ACAS architecture is the Context Management Entity (CME). This entity consists of a context server (the server part of the CME), context manager (client part of the CME), and a context repository used by the context manager with service and sharing policy management (see Figure 7)

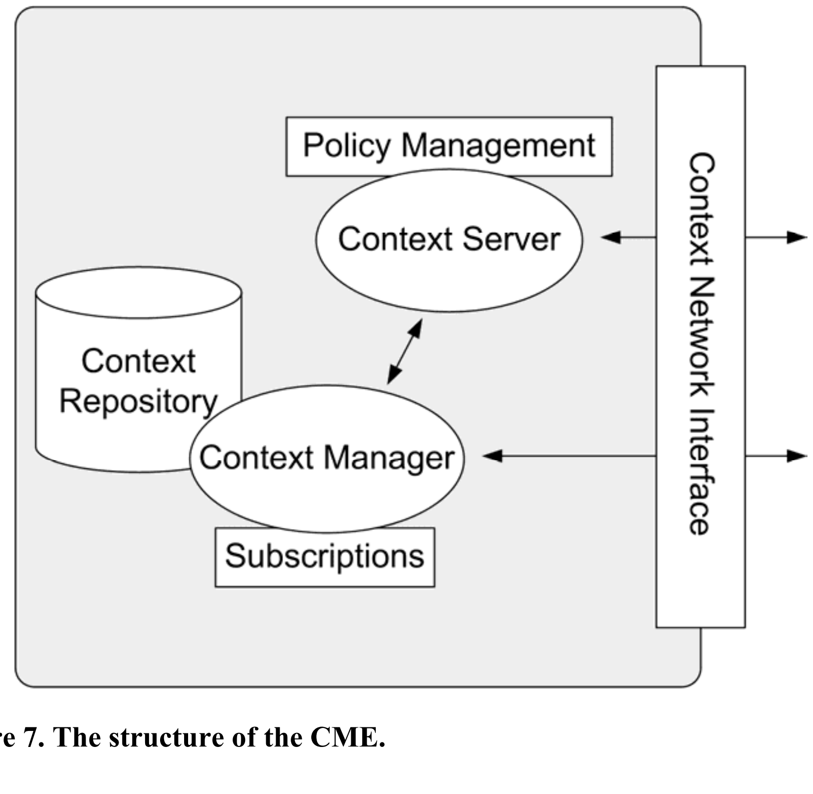

Figure 7. The structure of the CME.

The CME enabled entities form a context network. They know how to talk to and reach each other. The CME is employed on every device that wants to connect to the network and will provide the local applications with context information. On devices with very limited resources a simplified manager can be used to contact a CME on another device and thereby use the more powerful device as a server for context information. The applications will subscribe with the context manager in order to receive context information.

The context network can be static or dynamic. To provide a basic network or backbone for the context information the CMEs should have a static binding with at least one other CME. A connection with this CME is established on startup. The bindings may be manually or automatically configured (DHCP is one option). For example, the devices in possession of a certain user could have static bindings with each other and with CMEs representing the current location. By using dynamic bindings between CMEs the context network can be somewhat optimized and direct connections can be established between CMEs that wish to communicate. This will reduce the path length and optimize dataflow.

When an application wishes to connect to the context network it will search for a local CME (locally, on a well known port) and request a connection. The CME can either accept the connection and serve the application, or refer it to another CME. If no local CME is found, a list of default CMEs to connect to should be consulted. Context-aware applications can use the context server interface to access managed context information. The context server may base its decision on whether or not the request should be served based on (local) service policies. If the request is accepted it will be passed on to the context manager for processing and management.

4. Design

In this section we will look at the design of the context-aware service allocation system and discuss various issues that arise when an allocation is made. Furthermore, a number of techniques for making the system adaptive are presented.

4.1 Motivation

The main goal of the context-aware service allocation system proposed in this thesis is to make it more convenient for a user in a PAN to be reached and for other users to reach the user in the PAN. Ease of use is a very important aspect of the system design. The users of the system should not be bothered with manual settings. Preferably, they are not to notice the system at all. It should work silently in the background, always making accurate service allocations.

Our system, called "Context Aware Service Allocator" (CASA), will ease communication between users in two ways. First, it will make the devices in the user's PAN look like one device by enabling access to all of them by a single URL. This address is all that an initiator of a session has to know to reach the user in the PAN. Furthermore, this address will also be used for outgoing calls so that the internal structure and addressing never has to be revealed outside of the local system. An exterior initiator of a session can be seen as an indirect user of the CASA and has no knowledge about the PAN and the devices within. This user will never know about the CASA, provided that the delays associated with the session establishment are low enough not to be evident.

The second way the system serves the user is choosing the device that is best suited for handling the incoming request for the moment. This service allocation process will directly affect the user in the PAN, but also the exterior user. Both will of course suffer from a bad service allocation where a suboptimal device or no device at all is allocated to the incoming service.

The service allocation process requires context information about the PAN. The CASA will use a context network to gather this context information. This context network can have many different forms. In this work we will assume that this network has the same structure as the context management network introduced in the ACAS project [5]. However, CASA will also work with other types of context networks and as we will see later, a single stand-alone server may be enough for a smaller network with modest requirements on the context information.

Because the type and performance of the context network can vary, the service allocation system has to be flexible and tolerant of high delays. As discussed later in section 4.5, the number of devices in the PAN and the delays associated with getting information from the context network will affect the total service allocation delay. This total delay has to be hidden from the users if it is too high to be tolerable.

A delay longer than 1-2 seconds is considered to be annoying by most people, especially if there is no feedback during the waiting [22], [23]. Consequently, the goal is to allocate the service faster than this.

The allocation will be based on context information from the PAN, user settings, and information contained in the incoming request (e.g., the media descriptor located in the SDP body). Consequently, there are high demands on the accuracy of the context information. The decisions made by the CASA always have to be based on up to date information. A goal of the system design is to provide the allocating algorithm with as current information as possible.

4.2 Basic design

CASA will be partly based on the SIP Service Manager (SSM) designed by Konstantin Avgeropoulos [15]. SSM uses many of the same basic components needed for processing SIP requests and responses.

The functionality of CASA and SSM are related and the two systems should be able to be used side by side. The design of CASA will make it possible to merge the two systems completely and have both policy management and context-aware service allocation functionality in one system. To make the description of the design easy to follow for the reader, the two systems will be separated throughout this document. However, to make clear which parts can be shared by the two systems we will start by looking at those components.

As mentioned earlier, CASA will make all devices in the PAN look like one device to users outside the PAN. This is achieved by making them all reachable through one single SIP address. Consequently, the devices in the PAN have to register with CASA which will act as a proxy server.

In order for the devices in the PAN to be able to register at the CASA a Registrar server is needed. This local registrar will work closely with the location service, which will keep the mappings between the network address and the SIP address stored in a hash table. Observe that the SIP address registered at the CASA will *only* be used **internally**. The location service will later be used when a service allocation occurs and the network address is needed to route the SIP message to the right device. This is due to CASA acting as a proxy.

Each device will be managed by a Service Controller (SC) (see Figure 9). This SC is responsible for communication with the device and consequently an incoming message will be passed on to one or more SCs by CASA. The service allocation process determines which service controllers will be selected if the message is not part of an existing transaction. If it is, the message should be sent to the service controller that is participating in the transaction.

All communication between the PAN and the outside nodes will go through CASA (see Figure 8) which works as a switch for the SIP messages. When a register request is received from a device in the PAN, it is switched to the registrar. When an incoming message is received, the message will be switched to the proper service controller. An outgoing request (other than a register) from one of the devices inside the PAN will be switched to the responsible SC.

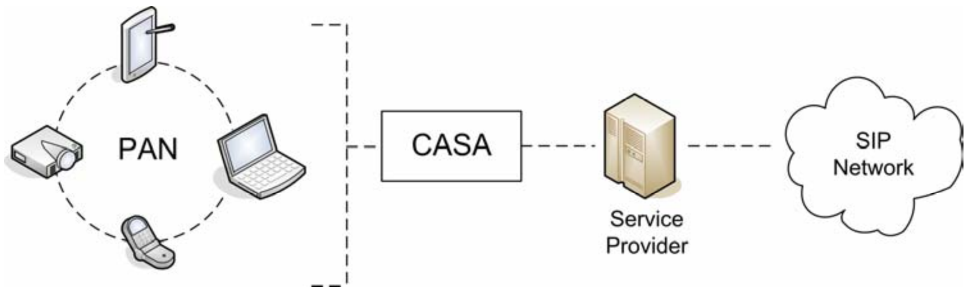

Figure 8. Placement of the CASA.

Now that the management of the devices inside the PAN has been covered it is time to look at how CASA is reachable from outside the PAN. By including a registration client in the design the CASA is able to register the Address Of Record (AOR) of the PAN at a service provider proxy. The AOR is the address that will be used for reaching the user and a device inside the PAN. Once this address is registered, all the service requests destined for the PAN will end up at the CASA.

If all the devices in the PAN together are represented by a single address, the internal or local addresses need to be hidden when communicating with a user on the outside. Consequently, the address of each message sent from a device in the PAN needs to be changed in order to make the message look like it came from the official AOR (i.e., the CASA). This is similar to a Network Address Translation (NAT) or IP proxy.

4.3 CASA specific design

The components described above are necessary for registering the devices inside the PAN, registering the official AOR at the service provider proxy, and handling incoming and outgoing messages. Given this base functionality we are now ready to discuss the components needed for service allocation. Two main components will be added to the design: (1) the Device Allocating unit and (2) the Information Collector unit. Additionally, some enhancements will be made to the existing components. For an overview of the design, see Figure 9.

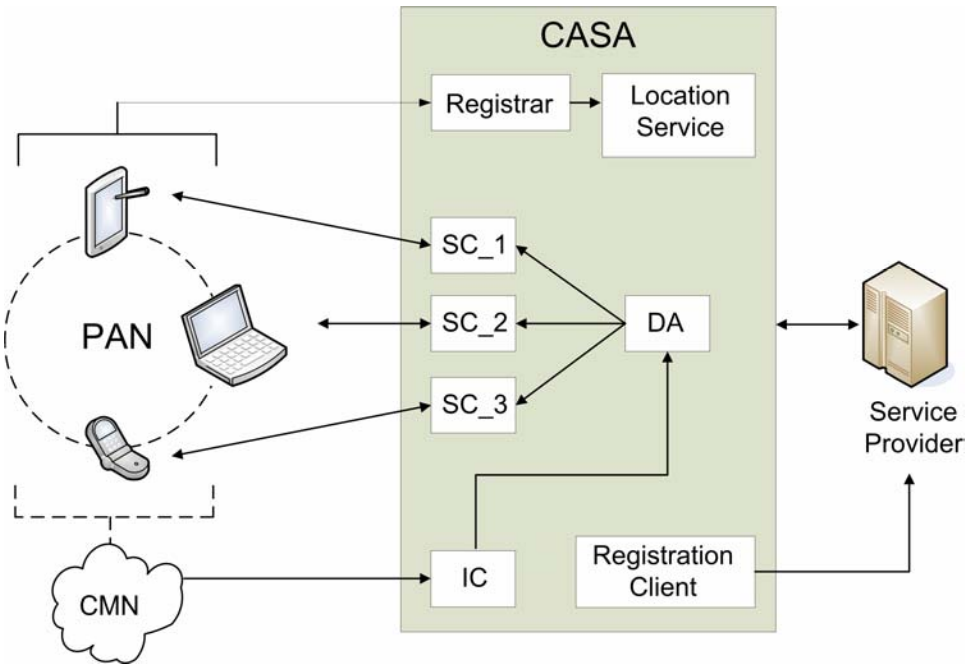

Figure 9. Overview of CASA.

As shown in Figure 9, the Registration client is responsible for registering the AOR of the PAN at the service provider. Furthermore, a Registrar and a Location Service is available at the CASA in order to manage the registrations of the devices in the PAN. Three devices are registered in the example and every device is associated with a service controller that manages the communication with the device. The Device Allocating unit (DA) and the Information Collector unit (IC) are the core components of the CASA and are described in detail below. For more information about the Context Management Network (CMN), see section 4.3.5.

4.3.1 The Device Allocating unit

First we examine the core of the CASA: the Device Allocating unit. The DA will make the actual service allocation decisions based on context information from the PAN and information provided in the incoming service request. These two sets of information are crucial for service allocation. A third important part affecting the outcome of device allocation is the allocation algorithm. The allocation algorithm determines how the DA uses the context information when making a decision. The allocation algorithm is the heart of the decision logic and may look very different depending on the context of the CASA (see section 4.3.3 for more information about the allocation algorithm).

The DA will always perform its work in the foreground, which implies that the delays associated with the service allocation decision process cannot be hidden from the user. The reason for this is that the service allocation decision has to be made (immediately) after the CASA receives the incoming service request. Even if the allocation algorithm is not using information in the request (which is possible) the service allocation decision should be taken as close in time to the actual allocation of the service as possible. If instead the service allocation decision was made in advance, it could, and in most cases would, be based on potentially obsolete information.

4.3.2 The Information Collector unit

The other main component is the Information Collector unit (IC). The duty of the IC is to collect context information from the entities in the PAN. This information is later used by the DA when performing a service allocation. The IC will get the requested information from a context network or more specifically a server in a context network, which provides context information about the entities in the PAN.

The context server is called CME (see section 4.3.5) and is assumed to be multithreaded. Likewise, each information request from the CASA is handled by a new thread within the IC in order to avoid waiting for earlier requests to finish. As discussed in section 4.5, the delays associated with the information collecting process may be visible to the users in some cases. If the information is collected in conjunction with the service allocation decision the delays associated with collecting the needed context information will be exposed to the external session initiator.

4.3.3 The allocation algorithm

The allocation algorithm can be seen as a standalone element and should be easy to alter or replace to satisfy the needs of the user in the PAN and CASA. In some scenarios the allocation algorithm may be simple and only need very limited amounts of information, i.e. one or two pieces of information from the devices and none from the incoming request may be enough. Other algorithms may require lots of information from all entities in the PAN and perhaps all the information contained in the incoming request. Different algorithms will generate very diverse loads on the DA and the hosting machine.

While the allocation algorithms may differ a lot, there are a few sections that can be identified in most algorithms. Often the allocation algorithms consist of loops iterating over the information. In each of these iterations different values will be inspected and compared according to the rules specified in the algorithm. The rules are often of the form IF {} THEN {} ELSE {}. These simple rules are used for comparing values and often to find extreme values, that is the smallest or biggest value of a certain type. After all the information has been inspected some additional processing may be required.

When an allocation algorithm is designed the priority of the properties and status of the devices in the PAN has to be decided. The algorithm designer has to decide for instance if available battery power is more important than available bandwidth. We will start by looking at service allocation A.

The allocation algorithm in service allocation A depends on two pieces of information, namely RunningProcesses which is a list of the running processes at the device and the BatteryStatus. This algorithm consists of one main loop that iterates through the devices and searches for a certain process in the process list.

In the same loop the battery status variable is read. The device that does not have this specific process running, but has the most battery power left will be allocated the service.

The allocation algorithm in service allocation B is more complex. Here context information such as ambient light level, display size and type, battery status, and device capabilities are used in the allocation algorithm together with information from the media descriptor in the incoming request. This allocation algorithm is designed with picture quality as the most important criterion and can be described as follows:

```
1.      Extract the media descriptor from the incoming request
2.      For each device do:
2.a.    {Check if device is capable of receiving video
2.b.    IF {AmbientLightLevel > threshold T} THEN {compare
        DisplayType}
2.c.    Compare DisplaySize
2.d.    Compare AvailableBandwidth}
3.      Allocate service
```

In the worst case, these five steps will be performed for every device in the PAN. The most important property is that the device is capable of receiving video. Consequently, this is checked first. The second thing to check is the ambient light level. The threshold T has to be decided by the algorithm designer by looking at the meaning of the value AmbientLightLevel. Suppose it can take on 5 different values where 1=completely dark and 5=bright sun light. If the threshold T=3, and the AmbientLightLevel is bigger, the DisplayType will matter (see section 2.1.1 for an explanation why) and should thus be included in the comparisons. Consequently, if there are only two devices that have a display that performs well in bright light among the devices that are capable of receiving video, only those two will be left for next step where DisplaySize is compared. The reason is that in this algorithm we do not care about the device (or its display size) if the ambient light is so strong that its display is useless. After this step AvailableBandwidth is compared. If more than one device is left after these eliminations they are equally good according to the algorithm and one of them is picked at random.

The allocation algorithm has to know how to compare values. For example, knowledge about different display types is needed. It must know in advance that a TFT display is better than a STN display when ambient light level is taken into consideration. This information has to be available to the DA and the allocation algorithm. In fact, it has to be built into the algorithm.

The algorithm in service allocation C consists of one loop. The loop iterates through the devices to find the biggest screen size and the largest number of attached speakers by looking at the DisplaySize and NumberOfSpeakers variables. When the loop has finished, the device (if the same device was chosen in both of the cases) or devices (if two different devices were chosen) will be allocated the service.

While these example algorithms are pretty simple, an allocation algorithm can be very complex. If advanced AI is used together with rich context information some really ingenious allocations are possible. However, the design of such algorithms is out of the scope of this work.

4.3.4 The outcome of the allocation algorithm

It is important to understand the difference between a *satisfying* and an *accurate* allocation. An allocation is considered satisfying if the user is pleased with the result. This means that the outcome of the allocation algorithm has to match the user's wishes. If not, the user probably won't be satisfied and will stop using the automatic service allocation feature. The best way of archiving satisfying allocations is thus to have customized allocation algorithms.

An accurate allocation, as discussed in 4.5.1, has to do with the allocation algorithm only and not the user. An algorithm can be accurate but still unsatisfying. If the information used by the allocation algorithm is up to date, the allocation will be accurate. On-demand information fetching or call-backs will always lead to accurate allocations. However, the situation is different when prefetching information. To get an idea of how accurate an allocation can be, we have to look at how up to date the information is.

If the information from a certain entity is updated (because of a change) at the context server with an interval (or a mean interval) T_{update} and this information is fetched with an interval T_{fetch}, the probability of having up to date information is $P_{uptodate} = 1 - (T_{fetch} / 2T_{update})$ when $T_{fetch} < T_{update}$ and $P_{uptodate} = (T_{update} / 2T_{fetch})$ when $T_{fetch} >= T_{update}$. The probability of performing an accurate allocation depends on $P_{uptodate}$ for the information needed in the algorithm.

Consider the algorithm of service allocation A and assume that prefetched information is used from three entities a, b, and c. The probability that the result of the allocation algorithm is accurate ($P_{accurate}$) is the product of all $P_{uptodate}$ for the information (provided that the information is updated independently at the server). Thus, in our example $P_{accurate} = P_{uptodate(a)} * P_{uptodate(b)} * P_{uptodate(c)}$.

The conclusions that can be drawn is that (1) the shorter prefetch interval, the higher probability of having up to date information and (2) the more information needed by the allocation algorithm, the greater risk that the allocation will be inaccurate when using prefetched information. Observe that it is not a given that the allocation will be inaccurate just because the information is not up to date, but there is a risk.

For example, consider a simple allocation algorithm that compares one value from each device and allocates the service to the device that has the largest value. If the devices in the PAN have similar values a missed update can cause the allocation to be inaccurate. However, if the values are far away from each other and only small changes occur a missed update of one of the values will not affect the outcome of the allocation.

4.3.5 Communicating with the context management network

Our system will work very closely with an entity in the context network called the CME, briefly introduced in section 3.5.1 and further described in [5]. The

CME is an important component of the context network serving applications, in our case the DA will utilize this context information in its decisions. It is possible to have a CME running on every device in a PAN (Figure 10), but this is not a requirement. Consequently, a CME may or may not reside on the same machine as the CASA. If the CME is external (Figure 11), it has to be reached through the network.

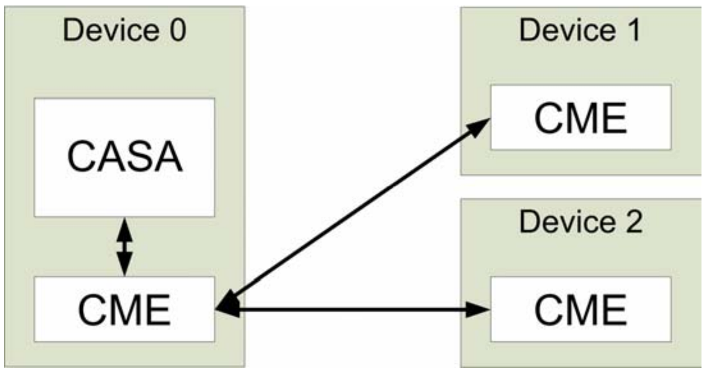

Figure 10. Local CME.

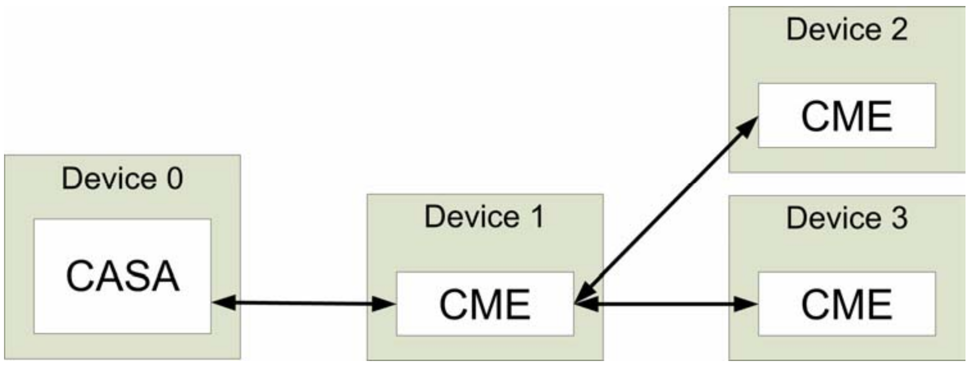

Figure 11. Remote CME.

The advantage of having a CME available locally is the reduced communication costs for the traffic to and from the CME. All traffic can go over the loopback interface and the communication delays will be very low. In this scenario, no traffic will be generated in the network directly by the CASA. If the CME is not locally available on the same machine as the CASA it is accessed through the network. This inevitably implies greater communication delays and more traffic in the network.

Ideally, from the point of view of the CASA, the CME always has all the information about all the entities in the PAN. Furthermore, this information is always up to date. At the time of writing this thesis no implementation of a CME exists and we do not know how near the ideal case a CME may perform.

Even though it is not the goal of this thesis to investigate the performance of a CME, we have to be prepared for some delays at the CME while the request for context information is being processed. The CME may be busy, or it may not have the information ready when receiving the request. Our system has to be tolerant to these possible additional delays. Furthermore we assume that the CME may or may not support call-backs, which means that the server automatically sends information to the CASA when it has changed.

4.4 Information fetching approaches

To be able to make proper decisions in the DA, up to date information is needed. If the CME doesn't support call-backs two different approaches are suggested. Either this information is collected just **when** it is needed, that is when an incoming request arrives. This approach is called *on-demand* information fetching. The other way is to periodically collect context information and store it locally. The information is thus *pre-fetched*. If *call-backs* are supported, the CASA will always get recently changed information and consequently it doesn't have to be fetched. These three approaches have advantages and disadvantages which we now will examine in the following sections.

4.4.1 On-demand information fetching

On-demand information fetching is the most simple and straight forward approach. When an incoming request arrives at the CASA the IC will be consulted to get context information. The IC connects to the context management network and queries the network for the information. The CME in the context management network will respond with the requested information. This information is then used by the DA to decide which devices are the most suitable for handling the request.

The most obvious advantage of on-demand information fetching is that the context information used by the DA will always be up to date. However, the fact that the query to the context management network has to be made in conjunction with the incoming request causes problem. The root of the problem is that communication with the context management network (that is between the IC and the CME), will take some time. Thus the DA will have to wait for the IC to return the information before a service allocation decision can be made. This delay associated with fetching the context information can not be hidden from the user if on-demand information fetching is used. The obvious reason is that the context information has to be fetched before the incoming request is forwarded.

Exactly how time consuming the communication with the CME is depends on several factors. First, as with all data communication the transmission and propagation delay will limit how fast data can be transferred. In an environment like a PAN the transmission delay is more important than the propagation delay because of the short distances. Additionally, the link type may affect the communication delays. In a PAN the communication will often take place over a wireless link which introduces slightly higher delays in comparison to a physically wired LAN due to lower bandwidth and longer contention time to access the media.

With a distance of 10 m between the sender and the receiver the propagation delay is ~10^{-7} s. If the link speed is 11 Mbps (802.11b) the transmission delay will be ~10^{-3} s when transferring 1KB. Thus, the propagation delay in this example is only ~$1/10^4$ of the total time.

In addition to the delays associated with sending and receiving data on the network the CME will spend some time processing the request. As discussed in section 4.3.3 this time may vary and consequently the system has to be prepared for further delays. See section 5.2.3 for the total delays when fetching context information from the context server.

In summary, the delays that can be expected when performing a service allocation are (1) the system to CME communication delay T_{Comm} ($T_{Comm} = T_{Comm\_request} + T_{Comm\_response}$), (2) CME request processing delay T_{CME}, and of course (3) the time it takes to execute the allocation algorithm on the system T_{Alloc} . Figure 12 shows a simplified scenario with an allocation only requiring one information fetch. The total delay (T_{tot}), which is the time between receiving the incoming request and the actual device allocation, depends on the number of devices in the PAN and the processing power of the machine hosting the CASA. The larger number of devices, the more information needs to be fetched and processed.

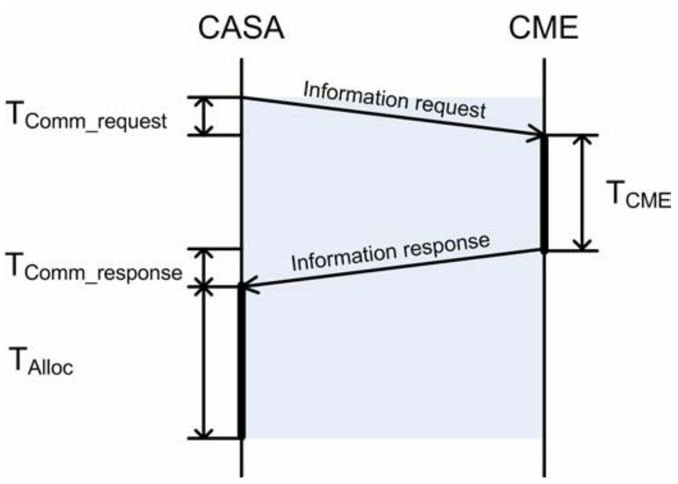

Figure 12. On-demand information fetching delays when performing an allocation. Visible delay shaded in grey.

4.4.2 Prefetching information

To hide the delays occurring when the context information is fetched the information can instead be pre-fetched. The IC will now periodically contact the context information network and fetch context information that will be stored locally in the system. The delay associated with fetching context information from the context management network is now no longer visible to the user. The only remaining delay that can not be hidden is the time it takes to execute the allocation algorithm. Figure 13 shows a simplified scenario with an allocation only requiring one information fetch.

The polling interval has to be carefully chosen. If the polling is carried out often the information stored in the system will be more up to date, but the amount of interaction needed with the CME will grow. This is an important trade-off when polling for information.

Pre-fetching information is an effective way to hide delays associated with fetching the information needed by the DA. The only delay observable for the user is the time spent processing the information when executing the allocation algorithm.

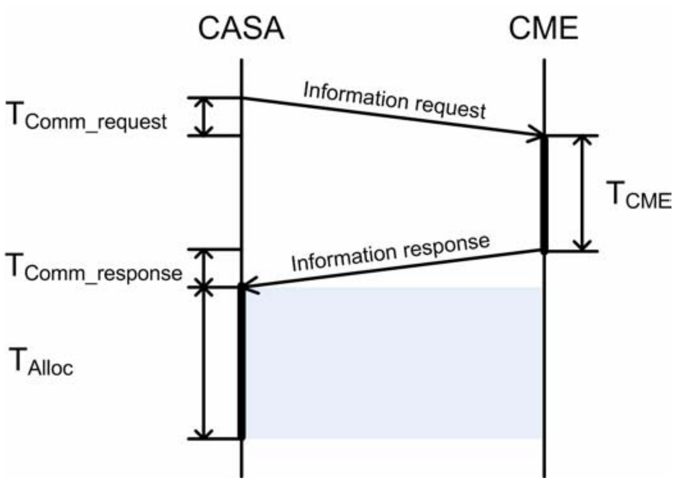

Figure 13. Information prefetch delays when performing an allocation. Visible delay shaded in grey.

4.4.3 Call-backs

If the CME supports call-backs, the problem with high delays when fetching the information is eliminated. The CASA will be fed with fresh information continuously in the background. When the information is needed by the DA it will already be there, fully updated. The only problem with call-backs is the lack of control of the network utilization. Information is sent over the network when it has changed and not necessarily when it is needed (compared to on-demand) which means that information that never is used will be sent over the network (as in prefetch mode). The network utilization depends on the information source and not the incoming request rate (on-demand) or a predefined interval (prefetch).

4.5 Deciding when to fetch information

Different situations put different demands on the fetching process of context information. In some cases the PAN may consist of just a few entities providing only a small amount of context information. In others the PAN may include lots of entities (theoretically, hundreds of entities are possible), each capable of delivering very rich context information. Likewise, in some settings the context information network may be very responsive and deliver information with very low delays, while in other cases each request for information may take a considerable amount of time. Furthermore, the process of updating information generates network traffic, which in some networks should be kept to a minimum due to high communication costs and power consumption.

In section 4.4 three approaches for collecting context information were presented. By looking at the basic demands on a service allocator we will now try to find the best approach for collecting context information.

4.5.1 High demands on accurate allocations

An allocation is considered accurate if the device(s) that were the most suitable for handling the request, according to the allocation algorithm, at the moment when the service request arrived at the CASA, were allocated the service. The most important factor for an allocation to be accurate is thus to have up to date information for the allocation process. If old information is used by the allocation process, the device(s) that **were** the most suitable when that information was collected will be allocated the service. Provided that the information has changed in the meanwhile, an inaccurate or suboptimal allocation will be performed.

If the information used by the allocation process is collected upon receiving the incoming service request an accurate allocation will be preformed, according to the definition above. If information is automatically updated at the CASA the allocation will be accurate as well. Consequently, on-demand information fetching or call-backs are preferred in this case. If delay is **not** an issue, the former should be used due to its low network utilization.

4.5.2 High demands on low allocation delays

As mentioned in section 4.4.1, there are three delays that will affect the total time T_{tot} that is needed for a service allocation when using on-demand information fetching. Two delays are associated with the information collecting process (T_{Comm} and T_{CME}) and the third with the allocation process (T_{Alloc}). The delay occurring when fetching all the information needed by the DA is called T_{IC}. When collecting information on-demand the time T_{IC} will be spent after the CASA receives the service request and consequently $T_{tot} = T_{IC} + T_{Alloc}$. On the other hand, when using call-backs or prefetching information $T_{tot} = T_{Alloc}$.

T_{IC} is the time needed for collecting **all** the information needed. To get an indication of how large this delay can be, we have to look at the number of entities that are subject for information fetching (n), T_{Comm}, and T_{CME}. If information is fetched only from one single entity (n=1) $T_{IC} = T_{Comm} + T_{CME}$. When information is needed from several entities T_{IC} will depend on the amount of parallelism of the CME and IC.

If the CME is capable of processing the requests in parallel T_{IC} will be reduced. Its important to note that the size and cause of the delay $T_{CME(i)}$ affects T_{IC}. If for example the main part of $T_{CME(i)}$ is because of lack of resources in the CME, every thread on the CME may experience this delay. In this case it will not help that the CME is multithreaded, the delays for every request will be additive (although not necessarily linearly). On the other hand, if the main part of $T_{CME(i)}$ is spent by the CME waiting for the queried entity to answer a request, the next request from the IC can be served in the meantime. Consequently, the largest processing time for a request will have big impact on T_{IC} and $T_{IC} \geq \max(T_{CME(i)})$ + (communication delays).

Suppose that T_{IC} is t ms for a certain PAN. This means that an allocation will take more than t ms in total ($T_{tot} = t + T_{Alloc}$, where $T_{Alloc} > 0$). However, if a maximum of t milliseconds are acceptable by the user, then on-demand information fetching can not be used. The system should instead prefetch information or use call-backs if supported.

T_{Alloc} is the time needed for the DA to execute the allocation algorithm. It depends on the complexity of the allocation algorithm, the number of devices, and the performance of the machine hosting the system. This delay is perhaps short in comparison with T_{IC}, but is of course machine dependent. There are extreme cases where T_{Alloc} can be considerable longer. If for example the hosting machine has insufficient amounts of free memory the information about the devices (which normally is kept in the main memory), will be swapped out on disk and trashing may occur upon accessing the information.

4.5.3 Minimizing the amount of traffic

In some environments network traffic is expensive and should be kept to a minimum. If information is collected frequently from the devices, the amount of traffic can be considerable. This applies especially to a PAN with many devices. Prefetching information periodically at short intervals is therefore not recommended if traffic costs or power consumption are an issue. Neither are call-backs because there is no control over the update rate. Instead on-demand information fetching should be used if possible.

If the prefetch interval is p seconds and the communication needs per prefetch is c bytes, c/p bytes will be sent every second. A simple example will show the amount of traffic that can be expected in a small PAN with 5 entities. Assume a combined request/response size of 10000 bytes (message sizes for the request and response messages including the context information in the response for the 5 entities). Now totally 1000 bytes will be sent every second if the prefetch interval is 10 seconds. This means that 82.4 Mbytes will be shuffled between the system and the context network every day and occupy frame space in the network. Even though this traffic is only a small part of the total bandwidth (~0.2% for Bluetooth in 433.9Kbit/s mode and ~0.005% for 802.11b in 11Mbit/s mode) extra background traffic may be undesirable in some settings. This background traffic is comparable to the 802.11b network beacons which are sent every 100ms (typically). With a beacon frame size of 60 bytes a total of 50 Mbytes is sent every day, just as beacons.

4.6 Adapting to actualities

The main goal of the system is to make it easier for the user in the PAN by automatically allocating the incoming service to the best suited device(s). The user may affect the choice of device(s) by changing preferences, but the average user is assumed to be uninterested in reconfiguring the system when conditions in the PAN change. In this section a number of techniques are presented for making changes in the PAN more transparent. These changes may depend either on the way the user acts or the entities in the PAN. Each of these features can easily be deactivated if the user wishes to have more control over the system.

4.6.1 Adapt to the request rate

As discussed in section 4.5 the best way to hide delays associated with fetching the context information is to prefetch the information or use call-backs (if supported). However, prefetching information normally implies high network utilization which in some cases is undesired while call-backs can give uncontrollable network utilization if the information values are fluctuating widely. If both low delay service allocation **and** low network utilization is important, the system can be used in a mode where the information fetching method is dynamically adjusted to the utilization of the system.

The system will start in on-demand mode for minimized network utilization. When the system starts receiving a certain number of service requests per unit of time the system will switch to prefetch or call-back mode if supported in order to minimize the delay of each service allocation. If the request rate later drops, the system will go back to on-demand mode.

Of course the requests prior to the switch to prefetch or call-back mode will experience the delay associated with the on-demand information fetching. However, a system receiving service requests more frequently than the predefined interval will soon reduce these delays, while at the same time reducing network traffic during time periods with low activity, such as the middle of the night.

4.6.2 Adapt to context network performance

The performance of the context network may vary over time and depend on the (user's or device's) location. Even if the CME contacted in one room was very responsive, the CME in the next room may be a bad performer. As discussed in section 4.5, a slow context network can be hidden from the user by prefetching the needed information or using call-backs. However, it would be very inconvenient for the user to be required to actively choose the method. Consequently, a mechanism is needed that estimates T_{IC} and selects the method that best suites the current situation. The main goal with this technique is reducing T_{tot} while at the same time trying to minimize the network utilization when call-backs are not supported.

The system will start in prefetch mode and the first round of information fetching will be measured to estimate T_{IC}. If T_{IC} exceeds the threshold value predefined by the user, on-demand information fetching is out of the question. Consequently, the CASA has to continue to work in prefetch mode. To be able to adapt to new conditions in the PAN each information retrieval will be measured. If considerable changes to T_{IC} are discovered (in this case if T_{IC} suddenly turns out to be less than the threshold value) the system will switch information fetching mode.

The advantage with this technique is that the user does not have to bother with tricky settings when the conditions in the PAN changes. Actually, very little extra communication and processing are needed. The disadvantage is that if the system is in on-demand mode when switching to another CME (for example, changing rooms) the first incoming service request may experience delays if this new CME is slow.

5. Analysis

To prove that a service allocation system for PANs is feasible we implemented the ideas presented in section 4. By implementing the system it is possible to identify possible problems with the design, but it also gave us the ability to perform tests that can give valuable input to other work in this area.

5.1 Implementation

In section 4 we proposed a system for allocating services called CASA, and also expressed a need for a context server to serve CASA with context information. We presented three different ways to get and update the context information at the CASA (see section 4.4). Furthermore, two techniques were introduced in section 4.6 in order to make changes in the environment of the PAN more transparent. All of the above were implemented in a prototype that shows that the components and techniques are feasible.

To demonstrate how a service allocation may happen the service allocation algorithms in section 4.3.3 were implemented as well. These allocations were also used when testing the system (see section 5.2).

5.1.1 Implementation tools

The prototype was implemented in Java using the Java 2 Standard Edition SDK 1.4.2_04 [25]. Java is a programming language well suited for prototyping due to its simplicity and debugging capabilities. Furthermore, Java is platform independent which makes it easy to perform tests on different platforms. The SIP signaling was facilitated by using an Application Programming Interface (API) called Java APIs for Integrated Network Framework, i.e., JAIN SIP, developed by the U.S. National Institute of Standards and Technology (NIST) [26].

To generate service requests and test the allocations, three different types of soft phones were used, namely Linphone (ver. 0.12.2) [27], Kphone (ver. 4.0.3) [28], and X-Lite (ver. 2.0) [29]. Linphone is designed for the GNOME desktop environment and consequently runs on Linux. Kphone is a Linux softphone as well, but developed for the K Desktop Environment (KDE). X-lite, on the other hand, is a softphone for Windows (including Windows CE) and MacOS.

5.1.2 Implementation details

The implementation shares several components with the SSM presented in [15]. Policy decisions and service allocations are closely related and it should be possible to use both mechanisms in conjunction with each other. The implementation allows the devices permitted by the policy decision to be used in the service allocation. Consequently, only devices allowed by the policies may be allocated the service.

To serve the CASA with context information a context server supporting all the information delivery methods, namely on-demand information fetching, prefetching, and call-backs is needed. This context server will act as a CME with server capabilities in a context network. The server should be external to CASA and able to run on a different machine. Furthermore, the server should be multithreaded so each request can be handled by its own thread.

The server is contacted by sending a request specifying which information is wanted. In our prototype a static IP address is used to reach the server. Upon receiving the request, the server reads the requested information and returns it in a response message. The context information about the PAN is stored in objects on the server and the request/response messages are sent as serialized objects. To reduce the number of requests and make the communication more effective, it is possible to ask the context server for more than one information element and entity per request. This means that all information needed in the allocation algorithm can be requested and returned in two messages (but not necessarily only two packets).

If call-backs are used the CASA will constantly listen for incoming information messages on port 8081 (currently unassigned according to Internet Assigned Numbers Authority (IANA) [30]). The context server will send information to this port whenever it is updated. When an information message arrives (similar to an on-demand or prefetch response message) the information stored in the CASA will be updated.

All requests for context information are made through the IC which acts as a client. Each request is handled by a new thread. The IC is implemented supporting a generic interface. This design makes it easy to replace the current context server and the communication protocol with other solutions when available. The TCP protocol is used for data transport due to its reliability and flow control.

The Device Allocator (DA) is responsible for choosing the devices that will handle the incoming service request. This is where the allocation algorithm is implemented and also where the context information is processed. The allocation algorithm is intended to be easy to replace to satisfy the needs of the user in the PAN. Consequently, the DA is implemented using an interface as well.

Note that establishment of the RTP streams when more than one device is allocated the incoming service and the media needs to be forked is excluded from the implementation. We will only make sure that the right devices are allocated (invited to the session). Media forking can be handled in different ways [31]. One is to deliver multiple streams to the caller from different addresses, another is to let the RTP streams go through the gateway (i.e. CASA). Read more about this in section 6.2.

For more details of the implementation, the reader can consult the Javadoc documentation provided with the code. Here each class and its associated methods are described in detail.

5.2 Test

Given an implementation of a system for automatic service allocations it is time to prove that it actually works and that it performs as expected. There are several questions that need to be answered. Are the right devices allocated the service? How accurate is the allocation? Is the total delay for allocating a service acceptable? How do different delays affect the total delay? How much traffic can be expected on the context network with a certain information collection method? All these questions are answered in the following sections.

As discussed in section 4.3.3 the example allocation algorithms were designed to show different properties and aspects of service allocations. These allocation algorithms will be used throughout all the tests.

5.2.1 The correctness of a service allocation

A fundamental requirement of the system is that the correct devices are allocated in a service allocation. The devices that are allocated the service are chosen by the allocation algorithm. Here we do not care if the allocation is **accurate**, only if the devices from the outcome of the allocation algorithm are allocated, i.e. receives an INVITE message with a correct body. Details about the example allocation algorithms can be found in section 4.3.3.

In this test we will look at a PAN with 5 devices that are capable of providing some kind of service.

Service allocation A allocates the incoming service to the device that does not have a certain process p running, but has the most battery power left. A list of the current running process and the battery status is needed from every device. Throughout all the tests the size of the process list was 1.8KB.

When the service request arrived at CASA the devices in the PAN had the status shown in Table 1:

Device	p in list (boolean)	Battery status
0	false	70
1	false	93
2	true	99
3	false	24
4	false	11

Table 1. The status of the devices in the PAN before service allocation A.

Result: Device 1 was allocated the service which satisfies the allocation algorithm.

Service allocation B needs information about the ambient light level, device capabilities, display type, display size, and available bandwidth in order to select the device that can best play an incoming video stream. All information but the ambient light level is collected from the devices that may be allocated the service. The ambient light level value is fetched separately and comes from a device (device 5) that cannot handle a service request. Furthermore, the media descriptor of the incoming request needs to be read.

When the service request arrived at CASA the devices in the PAN had the status presented in Table 2. The media descriptor in the incoming request held the text string "video".

Device	Capabilities	Display type	Display size (pixels)	Available bandwidth	Ambient light level
0	video	TFT	20480	9967	-
1	video	STN	20480	711	-
2	video	TFT	40960	901	-
3	video	TFT	40960	9609	-
4	video	TFT	20480	808	-
5	-	-	-	-	4

Table 2. The status of the devices in the PAN before service allocation B.

Result: Device 3 was allocated the service which is in accordance with the allocation algorithm.

Service allocation C allocates the incoming service request to the device or devices that have the biggest screen size and the largest number of speakers.

When the service request arrived at CASA the devices in the PAN had the status shown in Table 3.

Device	Display size (pixels)	Number of speakers
0	20480	2
1	20480	8
2	40960	6
3	40960	2
4	20480	4

Table 3. The status of the devices in the PAN before service allocation C.

Result: Device 1 was allocated the audio stream and device 3 the video stream which satisfies the allocation algorithm.

5.2.2 The accuracy of a service allocation

As discussed in section 4.3.4 the accuracy of a service allocation depends on how up to date the context information is. If on-demand information fetching is used, the context information will always be up to date when needed because it is fetched after the service request arrived. Consequently, the allocation will always be accurate if on-demand information fetching is used. Likewise, the information will also be up to date when it is needed by the allocation when call-backs are used. In fact, the context information will **always** be up to date at the CASA when call-backs are used.

The situation is different when the context information is prefetched. Now the context information stored at the CASA and used by the allocation algorithm can be outdated. Consequently, a service allocation based on prefetched context information will be more or less accurate. The prefetch interval and the update interval of the information at the context server will give us the probability that the information used by the CASA is up to date. If the context information is up to date when it is used it means that same information can be found on the context server.

How inaccurate can a service allocation be, but yet be acceptable is up to the user. However, the lower the accuracy the greater the risk that the wrong device(s) is allocated the service. If a faulty allocation is totally unacceptable, prefetch should not be used at all. In this thesis, we accept an allocation outcome only if the probability that it is accurate is greater than 50%.

To test how often the allocation algorithm is using outdated information we have to simulate changes in the context information at the server as well the incoming service requests. One way to study the consistency of the information at the server and the information used by the allocation algorithm is to call a special method on the server when the information is used by the CASA. This call will initiate a dump of the information to a buffer. If the information used in the allocation is dumped as well, these buffers can later be compared for consistency. The server and the CASA should run on the same machine when performing this test to avoid communication delays that otherwise will affect the dump method call. The call can now be a direct method call on the server object.

We will test the consistency by having a fixed context information update interval (this can be seen as a mean update interval) and vary the prefetch interval to see how this affects the accuracy of the allocation by changing the prefetch interval. Each information update at the server indicates a change.

The context information is updated at the server with a mean interval of 10 seconds (uniformly distributed update times). Every time an incoming request arrives at the CASA the information is written to a log file, both at the CASA and the context server. The arrival rate of the requests was 1/T where T was uniformly distributed and mean(T)=10*T_{fetch}. Note that the updates from the devices are not necessarily always uniformly distributed in time. This depends on the sensors and the behavior of the measured quantity.

In the first test information is only fetched from one device in order to see how the prefetching interval affects the amount of information that is up to date (i.e. matching the server's information) at the CASA when a request arrives. The test was run for 200 incoming requests. The results are presented in Table 4.

T_{fetch} (seconds)	Theoretical $P_{uptodate}$	Matching information
1	0.95	0.94
2.5	0.875	0.87
5	0.75	0.74
10	0.5	0.52
20	0.25	0.27
40	0.125	0.12

Table 4. Theoretical values and test results for different prefetch intervals. Information about one single device was fetched.

According to the reasoning in section 4.3.4 the accuracy of an allocation should drop when information from more devices are needed. To test this statement we will look how the accuracy of an allocation changes with the number of devices involved in the information prefetching process. A prefetch interval of 5 seconds was chosen for this test.

Number of devices	Theoretical $P_{accurate}$	Matching information
1	0.75	0.74
2	0.56	0.53
3	0.42	0.41

Table 5. Theoretical values and test results for different numbers of devices with a prefetch interval of 5 seconds and a mean update interval of 10 seconds.

As Table 5 shows, the probability that the service allocation is accurate is already less than 50% when 3 devices are involved in the allocation. This means that even for a small PAN the prefetch rate has to be high to ensure that up to date information is used by the allocation algorithm.

It is important to note that *every* change in information is only significant for the outcome of the allocation algorithm in a worst case scenario. Depending on the algorithm and the values used (e.g., how close the different values are in relation to each other or if thresholds are used) this is not always the case and a missed update can still lead to a accurate allocation. However, one should be aware that there is a risk that the allocation will be inaccurate, and that this risk is bigger if the context information is fetched with a long prefetch interval.

5.2.3 The delay of a service allocation

When a service allocation is performed different delays will slow down the allocation process. Some of the delays are visible to the user (the caller) while others are possible to hide by performing the tasks that generate the delays in the background. As discussed in section 4.4, the delay associated with executing the allocation algorithm, T_{Alloc} will always be visible to the user. The delays occurring when fetching information (T_{IC}) will on the other hand only be visible when on-demand information fetching is used. In this series of tests we will examine the total delay for an allocation, both when fetching information in the background (prefetch or call-backs) and in the foreground (on-demand).

When the context information is fetched in the background the delay related to the service allocation process depends only on the time it takes to run the allocation algorithm. T_{Alloc} is of course highly dependent on the allocation algorithm, that is how it is implemented (for example how lists are searched) and how much context information it uses. The available processing power of the machine hosting the CASA is also important. To get an idea of the size of T_{Alloc} we will look at the example algorithms A, B, and C. These tests were performed on a 2.4 GHz AMD Athlon64 (running in 32-bit mode) with 1024MB of RAM. The number of devices and the context information are the same as in section 5.2.1. To measure time with higher precision than what is possible with the System.currentTimeMillis() a timing library written by Vladimir Roubtsov [32] using Java Native Interface (JNI) was used.

Although the delays are machine dependent the following numbers will give an indication of what performance that can be expected of the proposed system and which delays that may lead to problem.

Figures 14, 15 and 16 shows the delays of the allocation algorithms for different PAN sizes.

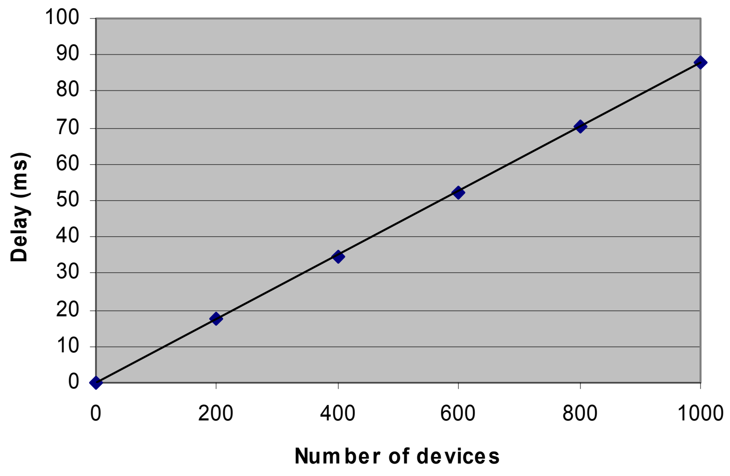

Figure 14. The delay associated with running the allocation algorithm A for different PAN sizes. The line is a linear curve fit to the measured values.

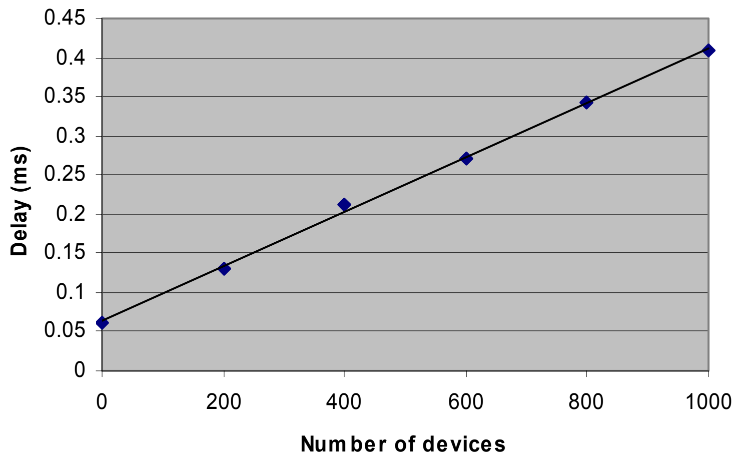

Figure 15. The delay associated with running the allocation algorithm B for different PAN sizes.

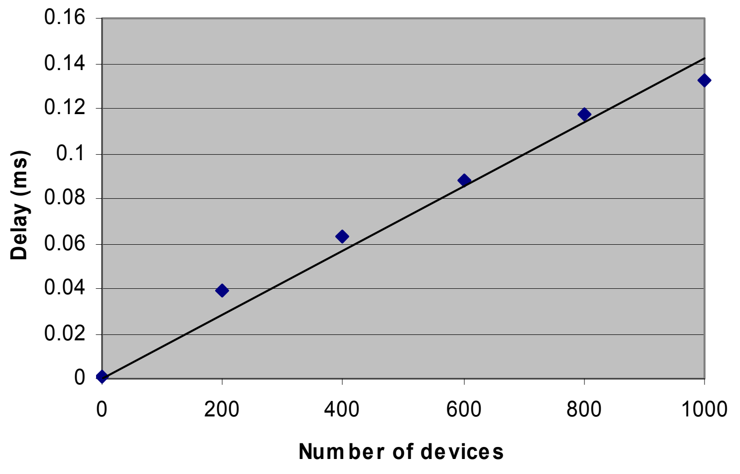

Figure 16. The delay associated with running the allocation algorithm C for different PAN sizes.

As the Figures 14-16 show, service allocation A is the most time consuming algorithm. The reason is that the algorithm searches for a process in the process list for each device. Service allocation B utilizes information from the incoming service request and consequently this information has to be extracted and analyzed. This procedure takes 61µs, and needs to be done every time the allocation algorithm is executed. All algorithms were fed with context information that would cause the worst case for each algorithm. This means that even the algorithms that eliminate devices had to iterate through all devices in every loop. All of the example algorithms scale linear but this may not be the case for future algorithm used with the CASA.

For 5 devices in the PAN the allocation algorithm delays were:

Service allocation A: 419µs
Service allocation B: 74µs
Service allocation C: 1 µs

The SIP message handling, i.e. parsing the incoming message and composing the outgoing message took 4.898ms.

When background information fetching is used the allocation algorithm delay and the SIP message handling will be the only delay visible to the user. None of these delays will be a problem when doing a service allocation as they together are much lower than the critical boundary of 1000ms. Even when the most complex service allocation algorithm (A) is used it will only take about 5.5ms before the INVITE message is ready to leave the CASA on the test machine for the example PAN with 5 devices.

Furthermore, when the context information is fetched in the background it is possible to pre-compute the parts of the allocation algorithm that does not depend on information in the service request. By doing this, the visible part of T_{Alloc} will be reduced. However, the pre-computation has to be performed every time new context information arrives at the CASA, which may be undesirable because of higher CPU utilization and power consumption.

When the context information is fetched on demand or in the foreground neither the communication delays nor the allocation algorithm delay can be hidden. These delays will always be visible to the service request initiator. Figure 17 shows the delays occurring when fetching information for a service allocation with different sizes of the PAN. In the following two tests, the context server resides on the same machine as the CASA.

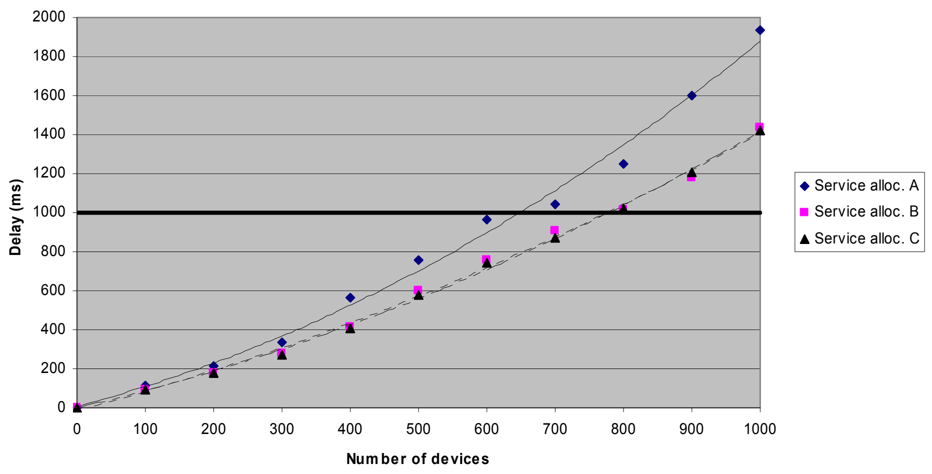

Figure 17. Delays associated with on-demand context information fetching.

As Figure 17 shows, up to nearly 800 devices can be included in the PAN before the information fetching delay gets too high in the test environment when service allocation B or C are used. If service allocation A is used the delay is slightly higher so the limit is ~650 devices. The reason is that the allocation algorithm requires more information which means larger response messages from the context server that takes a longer time to transfer.

In the example PAN with five devices that are able to serve a request, on-demand context information fetching is no problem. As Figure 18 shows, none of the service allocation examples take more than 25 ms. The reason why service allocation B takes the longest time is that this allocation algorithm requires information from one more device, the ambient light sensor. This sensor is located on a separate device in the example PAN.

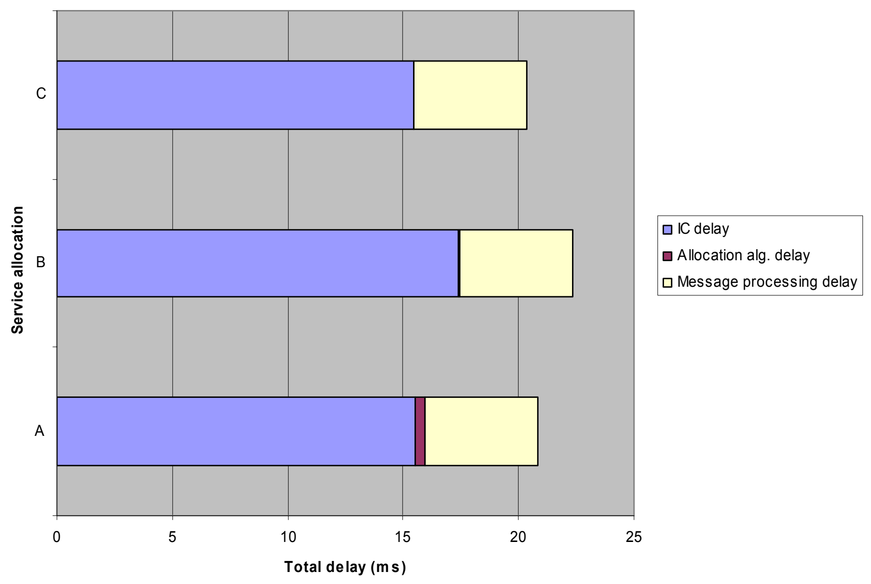

Figure 18. Total delays for the example service allocations in the example PAN (N=5) when using foreground information fetching.

5.2.4 The generated amount of traffic

The amount of traffic that is generated by the communication between the CASA and the context server during a certain time period depends on several factors. The way that the context information is acquired is of course an important factor. On-demand information fetching requests all information needed in the allocation and consequently the generated traffic will be the sum of all information elements. How much traffic that is generated for a certain time period is dependent on the incoming service request rate.

The incoming service request rate may look very different for different PANs. Many factors come into play. If the CASA for example is used by a mobile employee at the support department of a company the request rate can easily reach one request every fifth minute during the office hours. On the other hand, if the CASA is used by a recluse, the request rate can be a few incoming requests per year. Because the request rate depends on who is using the system and for what purposes it is difficult to predict what a typical request rate is. It is more important that the design of the CASA supports different usage scenarios.

Prefetching generates the same amount of traffic that on-demand fetching generates for one request **every** fetching round (for the same allocation algorithm and the same number of entities in the PAN). Consequently, the total amount of traffic for a certain time period is highly dependent of the prefetch interval.

When call-backs are used the server will send information only when it is updated. This means that to predict the generated amount of traffic, the behavior of the entity reporting to the server has to be analyzed (of course the size of the information elements has to be known as well).

Under these tests we will assume that the number of devices in the PAN subject for information fetching remains constant. It should be noted that service allocation B needs information about the ambient light level from a light sensor that we assume is located on a device that not is able to handle any incoming service. Consequently, if the PAN consists of N devices that can handle an incoming service, N+1 fetches is needed from the context server because information is fetched per entity.

As mentioned above, the amount of traffic generated when using prefetch depends on the number of devices subject to fetching, the amount of information needed from each device, and the prefetch interval. Figure 19 shows the amount of traffic that will be generated per day for the prefetch intervals discussed in section 5.2.2 for a PAN with N=1, while Figure 20 shows the daily traffic amounts in the example PAN with 5 devices.

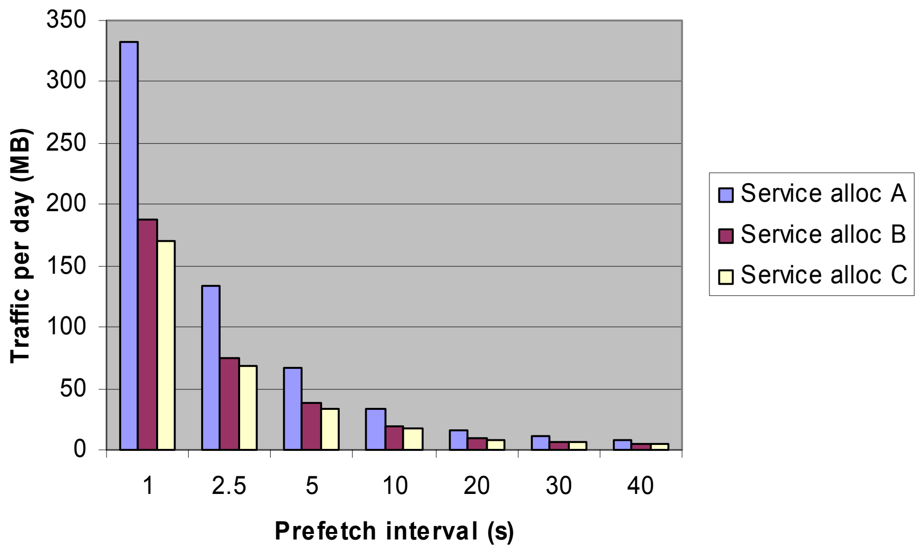

Figure 19. The generated amount of traffic with N=1 devices in the PAN when the context information is prefetched.

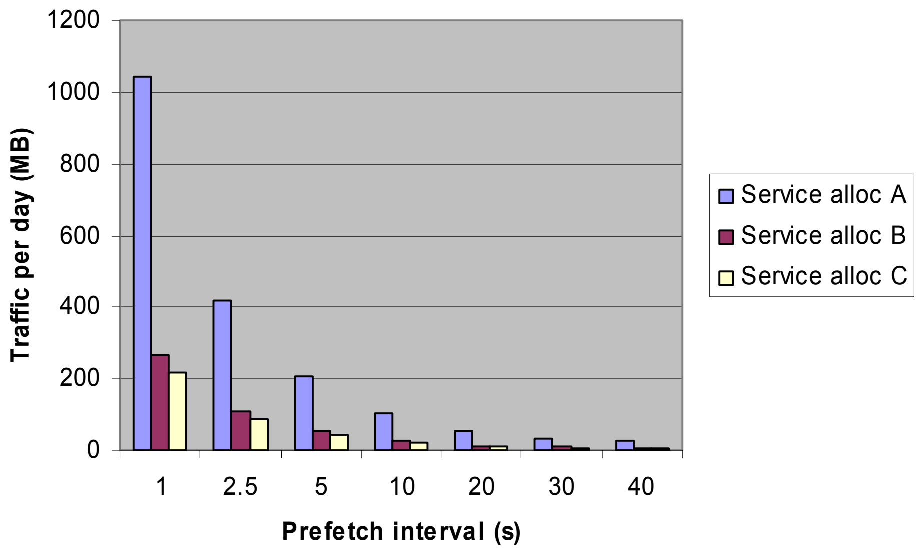

Figure 20. The generated amount of traffic with N=5 devices in the PAN when the context information is prefetched.

As Figure 20 shows, a short prefetch interval generates lots of traffic. In the example PAN consisting of 5 devices more than 1 GB is sent to and from the context server every day if the prefetch interval is 1 second and a service allocation that depends on bulky data (e.g., allocation A) is used. If the communication is expensive, some other fetching technique should be considered. According to [33], the energy consumed per every bit transmitted in a 802.11b (11 Mb/s) network is 327 nJ (peak). This means that up to approximately 350J is consumed when fetching context information every day (service allocation A, prefetch interval = 1 s).

The amount of traffic generated when using call-backs depends on the number of devices subject to fetching, the amount of information needed from each device, and the update interval at the context server. The last factor depends on the characteristics of the information, the sensors delivering the context information and the context server [5][24]. The update intervals in Figure 21 and 22 are the mean update intervals for each entity.

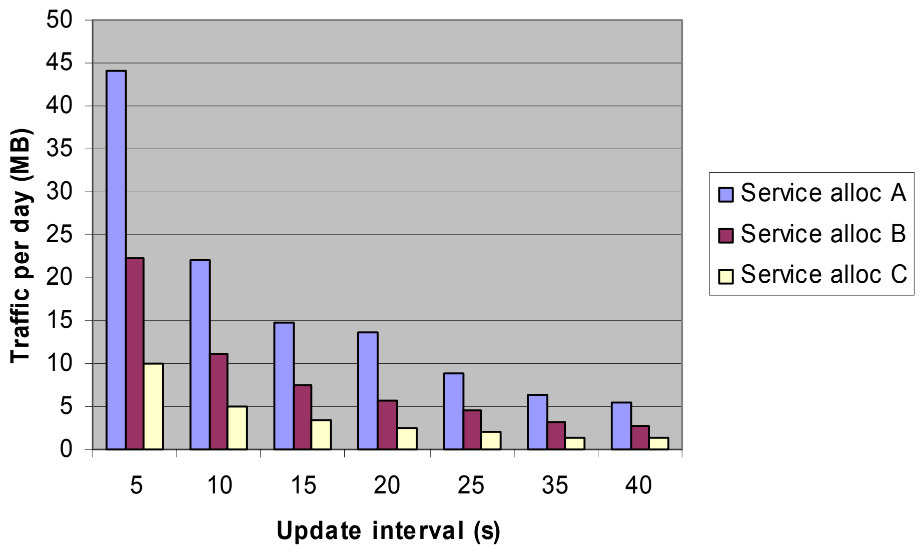

Figure 21. The generated amount of traffic with N=1 devices in the PAN when using call-backs.

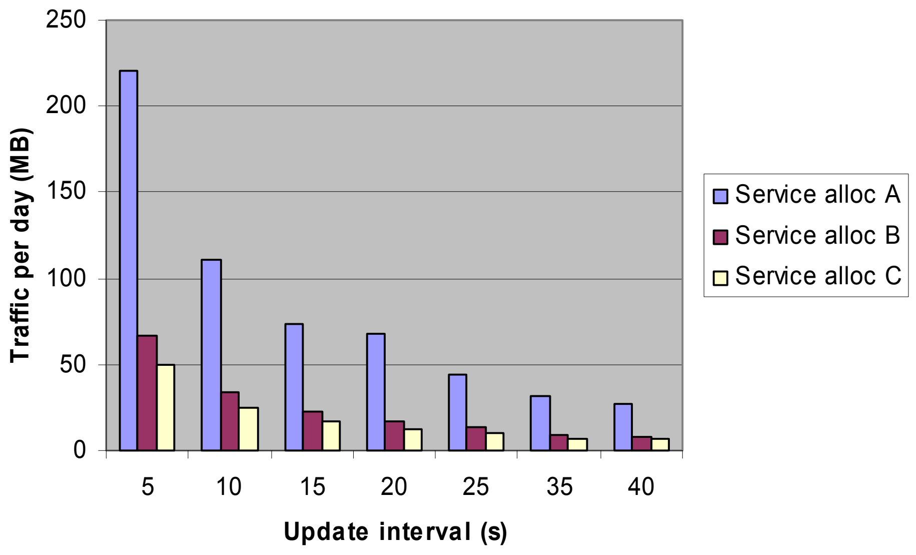

Figure 22. The generated amount of traffic with N=5 devices in the PAN when using call-backs.

The generated traffic amounts when the CASA uses on-demand information fetching are similar to prefetching with the difference that now the incoming request rate determines how often data is fetched and not a predefined prefetch rate. Figure 23 shows the amount of traffic generated for a single incoming service request for a certain PAN size.

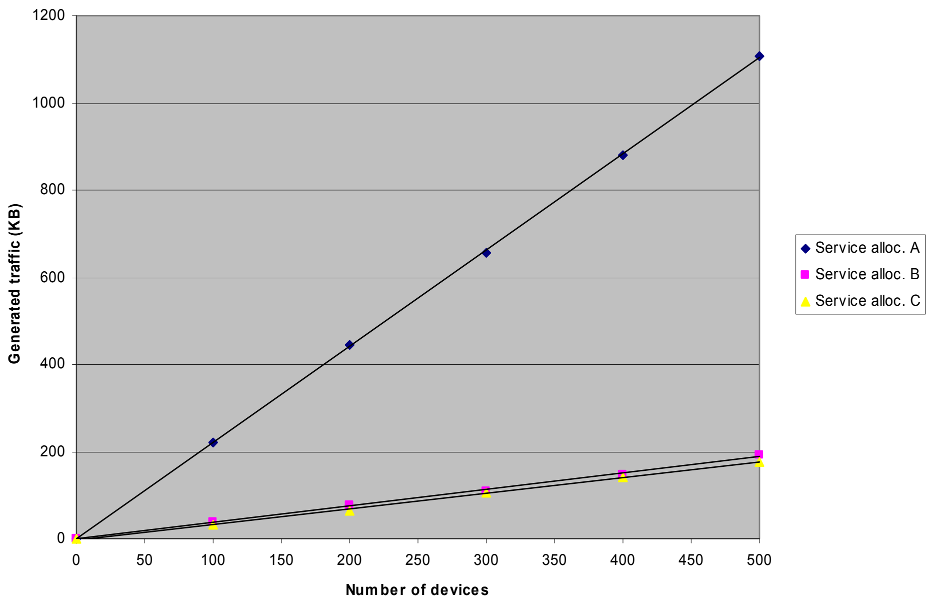

Figure 23. Traffic generated in the context network after receiving a service request.

As mentioned above, the rate of the context information fetching is an important factor of how much traffic that will be generated. When the information is prefetched a fixed prefetch rate will tell how often context information is fetched. If call-backs are used, it is instead the update rate of the information elements that determines how often new context information will arrive at the CASA. In the last case, when information is fetched on-demand, it is the incoming service request rate that determines how often information is fetched. Consequently, if the request rate is higher than either the prefetch rate (in prefetch mode) or the mean update rate (in call-back mode), more traffic will be generated compared to if the system was in background mode. As we will see in the next section, large amounts of traffic can be saved if the fetching mode is adjusted to the service request rate.

5.2.5 Test the ability to adapt to the request rate

By adapting the information fetching method to the request rate (see section 4.6.1) the delays are lowered when the system is heavily used, but still the network traffic is kept to a minimum when the system is rarely used. In practice this means that the CASA will work in prefetch or call-back mode when the incoming request rate is high and in on-demand mode when it is low. This test shows that the CASA changes fetching mode depending on the request rate and how this affects the traffic amounts.

The threshold value determining when the system should use background context information fetching is a balance between allocation delay and network utilization. A user concerned about high allocation delays should set a high threshold value, meaning that the CASA will switch to background mode as soon as another incoming request arrives at the system within that interval. On the other hand, if low network utilization is more important the threshold value should be small.

Consider a PAN that is used at work exclusively by an employee at the sales department. If for example the incoming service request rate is 2 per hour during office time in and the threshold value is 1 per hour the CASA will work in background mode after receiving the second service request. At the end of the day, when no service request arrives in 1 hour, the CASA will go back to on-demand mode and avoid network traffic during the night. In this example, call-back was used as the background information fetching technique with a mean update interval of 10 seconds.

To visualize this example and test that the CASA really switches mode, incoming service requests were generated with a mean arrival rate of 2 per hour and the arrivals occurring according to a Poisson process. The example PAN consists of 5 devices and service allocation A was used to select the device. Figure 24 shows when the service requests arrived at the CASA and which information fetching mode that was used.

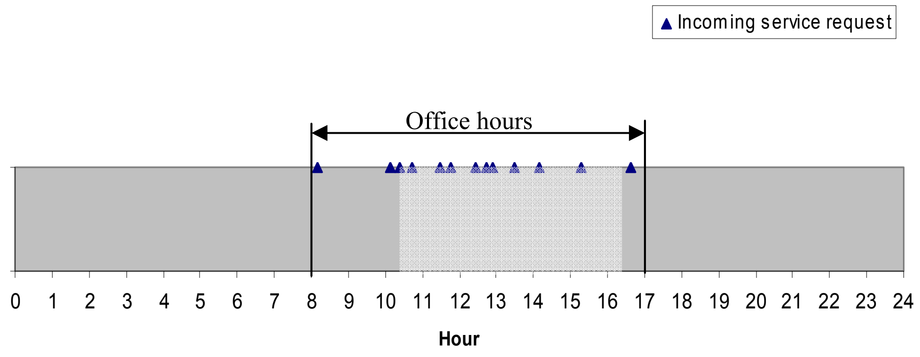

Figure 24. The incoming service requests and the resulting information fetching mode. Background information fetching mode in light gray.

During this day 13 incoming requests arrived at the CASA. The third request (at t=10.39) made the system switch from on-demand mode to call-back mode. The subsequent requests arrived within one hour after each other and consequently the system stayed in call-back mode until one hour after request 12. This means that the system was in call-back mode for 4.91 hours. During this day a total of 22.67 MB was sent between the CASA and the context information server. If the CASA should have been in call-back mode all the time, 110.51 MB would instead have been sent over the network. On the other hand, if the delay had been low enough to always fetch context information on-demand, only 157 KB would have been sent over the network. This shows that the cost of having low allocation delays often is paid for with increased network utilization in many networks. Note that the cost in accuracy of the allocation when going from 110.51 MB to 22.67 MB is zero because neither on-demand information fetching nor call-backs uses other than up to date context information.

5.2.6 Test the ability to adapt to context network performance

The aim with the technique to adapt to context network performance (proposed in section 4.6.2) is to have control over network utilization and allocation delays. If the information fetching delay is low enough (lower than the predefined threshold value) on-demand information fetching will be used to minimize the amounts of traffic. On the other hand, if the delay is higher than acceptable when fetching information on-demand, the system will instead prefetch information. In this test it is uninteresting **what** causes the information fetching delay, only that there is a delay. We will instead show that the CASA switches between modes depending on the fetching delay and that it will have impact on the total allocation delay.

To show the transition between the context information fetching modes a context server with adjustable delay for each request will be used. By altering the processing delay we can simulate different amounts of load on the context server. There are many possible reasons for why the fetching delay may change. For example the context server or some of the devices it communicates with may be overloaded. Figure 25 illustrates the behavior of the CASA when the fetching delays changes and the threshold value is set to 1000 ms.

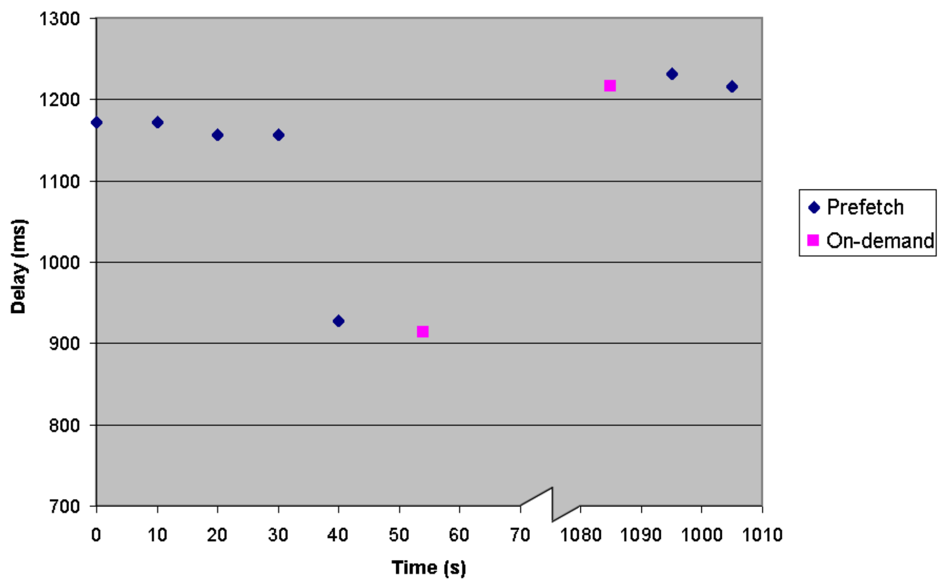

Figure 25. Adaptation to context network performance. The dots indicate context information fetches.

The CASA starts in prefetch mode with a prefetch interval of 10 seconds. The first four fetches take longer than the allowed maximum of 1000 ms. Therefore, the system will remain in prefetch mode. Information fetch number 5, occurring at time t=40, takes less than 1000 ms and the CASA will now switch to on-demand information fetching. At t=54 a service request arrives at the system and information will be fetched to serve the allocation algorithm with up to date context information. The delay associated with this fetch is also lower than the threshold. The next service request arrives at t=1085 and this time the information fetch takes longer than allowed, thus the CASA switches to prefetch mode.

The amount of traffic generated by the system will be reduced when the system is in on-demand mode (t=40 to t=1085). Consider the example PAN with 5 devices. If the CASA is configured with service allocation A generating 12073 bytes per fetch, a total of 1.24 MB will be saved during this period of time. For service allocation B and C the numbers are 0.31 MB (3098 bytes per fetch) and 0.26 MB (2532 bytes per fetch) respectively.

6. Conclusions and future work

The proposed service allocation system is able to use context information, gathered at a context server, to allocate the devices that can best serve an incoming service request. This concluding chapter presents findings and open issues that may be addressed in future work.

6.1 Conclusions

In this thesis we have looked at three different context information fetching methods, namely on-demand information fetching, prefetching, and call-backs. We have seen that the context information fetching method affects the accuracy of the allocation, the utilization of the context management network, and the delays associated with allocating a service request in different ways.

On-demand information fetching is the most straight-forward approach for serving the allocation algorithm with context information. The CASA acts as a client and requests information from the server when needed, that is only when receiving an incoming service request. Because the context information is requested immediately after receiving the incoming service request, the information returned by the context server is up to date and the allocation will always be accurate. However, there are two disadvantages with on-demand information fetching. Firstly, because the context information is fetched in the foreground, the delays associated with the fetching process will be visible to the service request initiator. If the context server for some reason is overloaded, perhaps due to a large number requests, this will affect the total allocation time. Furthermore, there is no control over how much information will be sent in the context network when on-demand information fetching is used because the incoming service request rate is often unknown.

One way to control the context network utilization is to prefetch the context information. Now information will be fetched in the background with a predefined interval. As long as the information subject to fetching isn't fluctuating that much in size, the traffic amounts generated by the system can be calculated in advance, thus leading to a higher average rate, but a lower peak value. Furthermore, because the context information is fetched in the background and no communication is needed with the context server upon receiving the service requests, the time spent waiting for the context information is no longer visible to the user. The problem with prefetching the context information is that the accuracy of the service allocation can be low. Even if the update rate of the context information is low and the prefetch rate is high there is no guarantee that the information used in the allocation algorithm is up to date. Consequently, if the accuracy of a service allocation is important, the context information should **not** be prefetched.

Instead of letting the system request context information, either upon receiving a service request or on a regular basis (both are pull methods), the context server can send or push context information to the system when it is updated. This approach of call-backs has several advantages. The most important is that the context information will always be up to date at the system. Moreover, the context server will send information to the system only when it is updated, which means that information that is already synchronized between the context server and the system won't occupy bandwidth in the network. However, it is important to understand that the update rate of the context information directly affects how often the context server will contact the system with new information. If the information is bulky, the amount of traffic generated can be considerable. At the same time, this technique can generate less traffic than on-demand information fetching if the information update rate is lower than the incoming service request rate. If communication costs are an issue, the update rates of the context information should be well understood or otherwise the information is better fetched on-demand.

6.2 Future work

While we show that multiple devices can be allocated one incoming service by forking and modifying the SIP INVITE message, the actual media streams still need to be established between the service request initiator and the allocated devices. When only one device is allocated the service, the media streams are established directly between the initiator and the selected device. But if several devices where allocated some mechanism to fork or replicate the media is needed. There are at least two ways of doing this [31]. Either the streams are established directly between the initiator and the allocated devices, or the CASA acts as a media proxy and splits/merges and distributes the media stream in the PAN. The first approach means less work for the CASA, but requires support for multiple stream endpoints at the initiator and more external traffic to/from the PAN. If instead the CASA acts as a proxy, the set of allocated devices will be hidden from the initiator and only one media stream needs to be established between the initiator and the CASA. However, this approach requires more logic and processing at the CASA when splitting and merging the media streams, but reduces external traffic.

There is currently work in progress in the context network area [5]. CMEs and protocols for distributing context information are under development. When these parts are finished the CASA could use this protocol and get the context information from a context network built upon the CMEs. The CASA is designed to be easy to use with other protocols and context information infrastructures.

To reduce traffic in the context network when using call-backs a mechanism that lets the CASA specify when new information should be sent could be added to the context server. This would make it possible to tell the server that a certain value only needs to be sent to the system if it exceeds a threshold value (that is sent to the server with the subscription). Such a mechanism is further described in [24].

Another way to avoid unnecessary communication with the context server is to add logic to the CASA that predicts changes in the context information. With this functionality it would be possible to use cached information instead of always asking the context server when information is needed. This would of course be most efficient with context information that changes at a relative fixed rate.

The CASA would benefit from a service discovery method that could identify available services and include them into the set of services provided in the PAN. Such protocols exist [8], but are yet to be integrated with the CASA.

Lastly, security aspects have been left out in this work. Neither the context server nor the CASA includes any mechanism that guarantees that the context information really comes from the requested entity. Consequently, context information can easily be modified or eavesdropped between the endpoints. Authorization between the context server and the CASA combined with integrity protection and encryption would be desirable.

7. References

[1] Bluetooth consortium, "Bluetooth core specification v.1.2", https://www.bluetooth.org/foundry/adopters/document/Bluetooth_Core_Specification_v1.2, November 2003.

[2] C. Perkins, "IP Mobility Support for IPv4", Internet RFC 3344, August 2002.

[3] J. Rosenberg, et al., "SIP: Session Initiation Protocol", Internet RFC 3261, June 2002.

[4] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research", Department of Computer Science, Dartmouth College, 2000

[5] C-G. Jansson, et al., "Context Data Distribution - Concepts and Approaches in the ACAS Project", Wireless@KTH, Royal Institute of Technology, May 2004

[6] ZigBee Alliance, URL: http://www.zigbee.org/, accessed November 9, 2004.

[7] Institute of Electrical and Electronics Engineers, Inc. "IEEE 802 standard", http://www.ieee.org/portal/index.jsp?pageID=corp_level1&path=about/802std&file=index.xml&xsl=generic.xsl, June 2004

[8] C. Ayrault, "Service discovery for Personal Area Networks". Master's thesis, Royal Institute of Technology (Stockholm), http://vvv.it.kth.se/docs/Reports/DEGREE-PROJECT-REPORTS/040826-Cecile_Ayrault-with-covers.pdf, August 2004.

[9] L. Dang, et al., "Practical VoIP: Using VOCAL", O'Reilly, 2002, ISBN 0-596-00078-2.

[10] M. Handley, and V. Jacobson, "SDP: Session Description Protocol", Internet RFC 2327, April 1998.

[11] T. Dierks, and C. Allen, Internet RFC 2246, "The TLS Protocol Version 1.0", January 1999.

[12] Netscape Communication Corp, "The SSL protocol", http://developer.netscape.com/docs/manuals/security/sslin/contents.htm#1041986, September 1998

[13] J. Arrko, et al., "MIKEY: Multimedia Internet KEYing", Internet RFC 3830, August 2004.

[14] H. Schulzrinne, et al., "RTP Profile for Audio and Video Conferences with Minimal Control", Internet RFC 1890, January 1996.

[15] K. Avgeropoulos, "Service Policy Management for User Service Policy Management for User-Centric Centric Services in Heterogeneous Mobile Networks". Master's thesis, Royal Institute of Technology (Stockholm), http://vvv.it.kth.se/docs/Reports/DEGREE-PROJECT-REPORTS/040401-Konstantinos_Avgeropoulos-with-cover.pdf, March 2004.

[16] Vivek S. Pai, et al., "Locality-Aware Request Distribution in Cluster-based Network Servers", In Proceedings of the Eighth International Conference on

Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII), San Jose, California, October 1998.

[17] R. Bhatti, et al., "A Trust-based Context-Aware Access Control Model for Web-Services", Publication in 3rd International Conference on Web Services (ICWS), San Diego, July 2004.

[18] W. Qadeer, et al., “Heterogeneous wireless network management”, http://akebono.stanford.edu/users/tajana/papers/PACS03.pdf, PACS 2003.

[19] T. Bray et. al., “Extensible Markup Language (XML) 1.0 (Second Edition)”, http://www.w3.org/TR/2000/REC-xml-20001006, October 2000.

[20] E. Alwagait and S. Ghandeharizadeh, "A Comparison of Alternative Web Service Scheduling and Allocation Policies", IEEE International Conference on Services Computing (SCC), Shanghai, China, September 2004.

[21] Google, URL: http://www.google.com, accessed September 20, 2004.

[22] J. Nielsen, “Usability engineering”, Morgan Kaufmann Pub, 1994, ISBN 0-12-518406-9.

[23] B. Schneiderman, "Response Time and Display Rate in Human Performance with Computers", Computing Surveys 16, no. 3, 1984, p. 265-285.

[24] A. Wennlund, “Context-aware Wearable Device for Reconfigurable Application Networks”, Master's thesis, Royal Institute of Technology (Stockholm), http://vvv.it.kth.se/docs/Reports/DEGREE-PROJECT-REPORTS/030430-Andreas_Wennlund.pdf, April 2003.

[25] Java Technology, Sun Microsystems, Inc. URL: http://java.sun.com, accessed September 20, 2004.

[26] Projet IP telephony / VOIP, URL: http://snad.ncsl.nist.gov/proj/iptel/, accessed September 20, 2004.

[27] Linphone.org, URL: http://www.linphone.org/, accessed September 20, 2004.

[28] WIRLAB - Network Research Center, URL: http://www.wirlab.net/kphone/, accessed September 20, 2004.

[29] Xten.com, URL: http://www.xten.com/, accessed September 20, 2004.

[30] Internet Assigned Numbers Authority (IANA) Port Numbers, URL: http://www.iana.org/assignments/port-numbers, accessed November 9, 2004.

[31] M. Shankar, “SIP Forked Media”, IETF Internet Draft, draft-shankar-sip-forked-media-00.txt, Februari 2001, expired.

[32] V. Roubtsov, High precision timer library, URL:http://www.javaworld.com/javaworld/javaqa/2003-01/01-qa-0110-timing.html, accessed October 5, 2004.

[33] R. Min, "Energy and Quality Scalable Wireless Communication", Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, (MA), http://www.mit.edu/~rmin/research/min-phd-thesis.pdf, June 2003.

Appendices

Appendix A – Abbreviations

ACAS	Adaptive and Context-Aware Services
AOR	Address-Of-Record
AP	Access Point
API	Application Programming Interface
AVP	Audio-Video Protocol
CASA	Context Aware Service Allocator
CDXP	Context Data eXchange Protocol
CME	Context Management Entity
CMN	Context Management Network
CODEC	Coder Decoder
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
IETF	Internet Engineering Task Force
IP	Internet Protocol
JAIN	Java APIs for Integrated Network Framework
KDE	K Desktop Environment
LAN	Local Area Network
LARD	Locality-Aware Request Distribution
MIKEY	Multimedia Internet KEYing
NIST	National Institute of Standards and Technology
PAN	Personal Area Network
PCM	Pulse Code Modulation
PDA	Personal Digital Assistant
RFC	Request For Comments
RTP	Real-Time Protocol
SDP	Session Description Protocol
SIP	Session Invitation Protocol
SRV	DNS Resource Record for locating services
SSL	Secure Sockets Layer
SSM	SIP Service Manager
STN	Super Twisted Nematic
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TFT	Thin Film Transistor
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
URI	Uniform Resource Identifier
UTF-8	Unicode Transformation Format-8
WLAN	Wireless LAN
XML	eXtensible Markup Language

Appendix B – Table of figures

Appendix C – Table of tables

IMIT/LCN 2004-10