Diameter service creation investigation and HSS evaluation

PETTER LINDGREN

KTH Information and
Communication Technology

Master of Science Thesis
Stockholm, Sweden 2010

TRITA-ICT-EX-2010:29

Diameter service creation investigation and HSS evaluation

Final 2010-02-23

PETTER LINDGREN

Master's Thesis
Supervisor: Stefan Östergaard, Kajsa Goffrich
Examiner: Prof. Gerald Q. Maguire Jr.

Abstract

With a prospect of creating a new generation of cellular networks using IP based communication, service providers, operators and developers have created the IP Multimedia Subsystem (IMS). Which with a more open architecture than older cellular networks will make it more like the internet in terms of service creation.

This thesis creates an example service that uses subscriber data in the Home Subscriber Server (HSS) to deliver a service to the network's users. The service is created to extend the functionality of the IMS test environment at Attentec AB to assist them in their efforts of learning and experimenting with the IMS core network. In this thesis the FoHSS used in the test environment is evaluated in order to measure its performance.

This thesis gives examples of the implementation overhead of creating services in the IMS network and shows some of the functionality that is available to IMS services compared to service creation on the internet. Load tests of the FoHSS are made and shows that it uses polling to maintain subscription updates and that it has a negative impact on performance under heavy load. This thesis suggests that it is good for IMS test environments but not commercial use.

Keywords: Diameter, HSS, IMS, service creation

Sammanfattning

Med ambitionen att skapa nästa generations mobilnätverk med IP-baserad kommunikation, har tjänsteleverantörer, operatörer och utvecklare skapat IMS. IMS som med en mer öppen arkitektur än dess föregångare försöker närma sig internet när det gäller skapande av nya tjänster. Det ska vara enklare och ta mindre tid än i ett äldre mobilnät.

Detta examensarbete skapar en exempeltjänst som använder användaruppgifter i IMS användardatabas för att leverera en tjänst till nätets användare. Tjänsten har skapats för att utöka funktionerna i den IMS testmiljö som Attentec AB har satt upp för att underlätta för sina anställda att lära sig mer om med IMS-kärnan genom att experimentera med den. I denna avhandling utvärderas även den HSS som finns i testmiljön.

Examensarbetet ger exempel på det extraarbete som behöver göras för att skapa tjänster i ett IMS-nätverk och visar på en del av den funktionalitet som finns tillgänglig för tjänster i IMS jämfört med skapande av nya tjänster på Internet. De lasttester som utfördes på användardatabasen visar att den använder pollning för att upptäcka ändringar i användardata och skicka ut uppdateringar till prenumeranter samt att dessa har en negativ inverkan på prestandan under tung belastning. Denna avhandling visar att den passar bra i IMS testmiljöer men för inte kommersiellt bruk.

Contents

List of Figures

List of Tables

Listings

List of Acronyms and Abbreviations

3GPP	Third Generation Partnership Project
AAA	Authentication, Authorization, and Accounting
AB	Aktiebolag
Abbr.	Abbreviation
ACA	Accounting-Answer
ACR	Accounting-Request
ADSL	Asymmetric Digital Subscriber Line
AS	Application Server
ASR	Abort-Session-Request
AVP	Attribute value pair
B2BUA	Back to back user agent
BGCF	Border Gateway Control Function
CAMEL	Customized Applications for Mobile network Enhanced Logic
CEA	Capabilities-Exchange-Answer
CER	Capabilities-Exchange-Request
CSCF	Call/Session Control Function
Cx	Cx interface in IMS (connects S-CSCF and I-CSCF with HSS)
DNS	Domain Name System
DPA	Disconnect-Peer-Answer
DPR	Disconnect-Peer-Request
DSAI	Dynamic Service Activation Information
DWA	Diameter-Watchdog-Answer
DWR	Diameter-Watchdog-Request
Dx	Dx interface (connects S-CSCF and I-CSCF with SLF)
FOKUS	Fraunhofer Institute for Open Communication Systems
FQDN	Fully Qualified Domain Name
GSM	Global System for Mobile communication
HSS	Home Subscriber Server
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I-CSCF	Internet Assigned Numbers Authority
IANA	Interrogating-CSCF
ID	Identifier
IETF	Internet Engineering Task Force

IFC	Initial filter criteria
IM-SSF	IMS Application Layer Gateway
IMS	IP Multimedia-Service Switching Function
IMS-ALG	IP Multimedia Subsystem
ISIM	IP multimedia Services Identity Module
LIA	Location-Information-Answer
LIR	Location-Information-Request
MAA	Multimedia-Auth-Answer
MAR	Multimedia-Auth-Request
MB	Megabyte
MGCF	Media Gateway Control Function
MGW	Media Gateway
MRF	Media Resource Function
MRFC	Media Resource Function Controller
MRFP	Media Resource Function Processor
MSISDN	Mobile Subscriber ISDN
NAI	Network Address Identifier
NAPTR	Name Authority Pointer
NASREQ	Diameter Network Access Server Application
NAT	Network Address Translation
OR	Logic operation or.
OSA	Open Services Access
OSA-SCS	Open Service Access - Service Capabilities Server
P-CSCF	Personal Computer
PC	Proxy-CSCF
PDF	Policy Decision Function
PNA	Push-Notification-Answer
PNR	Push-Notification-Request
PPA	Profile-Push-Answer
PPR	Profile-Push-Request
PSI	Public Service Identity
PSTN	Public Switched Telephone Network
PUA	Profile-Update-Answer
PUR	Profile-Update-Request
QoS	Quality of Service
RAA	Re-Auth-Answer
RADIUS	Remote Authentication Dial-In User Services
RAR	Re-Auth-Request
RTA	Registration-Termination-Answer
RTCP	Real-Time Control Protocol
RTP	Real-Time Protocol
RTR	Registration-Termination-Request
S-CSCF	Serving-CSCF
SAA	Server-Assignment-Answer

SAR	Server-Assignment-Request
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SGW	Signalling Gateway
Sh	Sh interface (connects AS and HSS)
SIP	Session Initiation Protocol
SL	Subscriber Locator
SLF	Subscription Locator Function
SMTP	Simple Mail Transfer Protocol
SNA	Subscribe-Notifications-Answer
SNMP	Simple Network Management Protocol
SNR	Subscribe-Notifications-Request
STA	Session-Termination-Answer
STR	Session-Termination-Request
SVN	Subversion
TACACS.	Terminal Access Controller Access-Control System
TEL	Telephone
TLS	Transport Layer Security
TRGW	Translation Gateway
TS	Technical Specification
UA	User Agent
UAA	User-Authorization-Answer
UAR	User-Authorization-Request
UDA	User-Data-Answer
UDR	User-Data-Request
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
WAP	Wireless Application Protocol
WLAN	Wireless Local Area Network

Chapter 1

Introduction

This chapter contains the background, purpose, goals, and scope of this thesis.

1.1 Background

As the Internet extends to include cellular networks, users have begun to expect more Internet and Internet like services via their mobile phones. Deploying new services within older cellular networks was both difficult and costly; in contrast to the Internet where creation of new services is both cheap and easy. The Third Generation Partnership Project (3GPP) introduced the IP Multimedia Subsystem (IMS) to combine the cellular communication networks with Internet technology in order to facilitate the ability of developers to introduce new services.

In order to learn more about IMS, and to find out what kinds of new services are possible to create, many companies set up IMS test environments of their own. These setups enable them to experiment with, and learn more about, IMS.

1.2 Purpose

The purpose of this project is to give users of the IMS environment at Attentec AB an example service that uses the interface between the Application Server (AS) and the Home Subscriber Server (HSS). An example service was created to explore the concept of and show a proof by existence of a service that uses subscriber data in the HSS to grant access to a service, based on a subscriber's privileges. This project also tries to estimate the maximum load of the HSS used in this test environment.

1.3 Goals

The goals of this thesis project were to investigate IMS in general and take a closer look at the Sh interface and to see what new possibilities are available for new services. More concretely this was to be shown by creating a proof-of-concept

service that uses information about subscribers stored in the HSS. A final goal was to evaluate the performance of the HSS using load tests.

1.4 Thesis Outline

Chapters and their contents are briefly described below.

1: Introduction contains the background, purpose, goals, scope, and outline of this thesis.

2: IP Multimedia Subsystem describes the framework upon which the rest of the thesis is based.

3: Diameter: Base Protocol covers the most important aspects of the Diameter protocol.

4: Diameter SIP Application examines the Diameter SIP application and describes how it is used in IMS.

5: An IMS Service investigates creation of an example service using IMS and Diameter.

6: Load tests tries to determine how the HSS in the test environment scales with load.

7: Conclusions contains the results of the service creation and load tests.

Chapter 2

IP Multimedia Subsystem

This chapter describes the framework upon which the rest of the thesis is based.

2.1 What is IMS?

The IP Multimedia Subsystem (IMS) uses Internet technology to create a walled garden of IP based multimedia services (i.e., the operator controls what services are offered to their subscribers). IMS was developed by the Third Generation Partnership Project (3GPP) and the Internet Engineering Task Force (IETF), mainly using existing open protocols or new open standards. 3GPP is a group of telecommunications organizations collaborating to create a third generation mobile phone system and to control the evolution of GSM. IETF is an open standards organization that develops Internet standards.

2.2 History

3GPP was formed in 1998 in order to create the next generation GSM network. 3GPP worked together with the IETF to create a set of open standards for this new network. IETF contributed the base technology and protocol specifications while 3GPP created specifications for the framework and defined the architecture of IMS. When IMS was initially designed, both IETF and Internet Assigned Numbers Authority (IANA) believed that the current Internet Protocol version 4 (IPv4) would be replaced with the new version 6 (IPv6) by the time IMS was to be deployed. However, the lifetime of IPv4 has been prolonged by subnetting and Network Address Translation (NAT) technologies. Therefore, although many IMS specifications use IPv6, IPv4 implementations of IMS are available today [23].

2.3 Requirements and design goals

When designing a new architecture for wide area mobile communication there are many requirements that were desirable, one example is interworking with older

(existing) systems and another was a desire to allow both vendors and operators to introduce technology in an evolutionary manner. These requirements make it more difficult to design a simple system. Both 3GPP and IETF have a lot of knowledge from earlier systems and protocols, which helped when developing IMS. Comparing the difficulty of creating and deploying new services in GSM networks versus the Internet, the advantages of using open standards and protocols becomes quite clear. The success of open protocols like Hyper Text Transport Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP) are because of their open standards and simplicity [23].

While the failure of WAP and other efforts was partially due to the closed nature of the networks and traditional telecommunications thinking, rather than the more modern approach of the Internet. Another cause for failure was the desire by vendors to make their customers (the telecommunication operators) happy - which many interpreted to mean that the network operator should retain control of what services were offered.

IMS supports many functions and features from existing 2G networks, such as roaming to different service providers (both within and between different countries), interworking with existing networks for instance the Public Switched Telephone Network (PSTN) and the Internet.

In IMS the focus is shifted to standardizing service *capabilities* instead of services. With the use of open protocols and IMS taking care of all signaling, third-party providers can focus on creating the main service functionality instead of worrying if their implementation will work in networks belonging to other operators. This is a big step forward in the design of wide area cellular networks and this change was made in the hope that it would make it easier for developers to design, implement, and deploy new services [23].

Another argument for operators to start migrating to IMS is that IP networks are easier to manage than older circuit switched networks. In existing networks the average revenue per user is decreasing, but by being able to deliver new kinds of services along with the basic access offer, the operator's profits can increase again; IMS provides new opportunities to do this [26].

2.4 IMS Architecture

IMS uses layers to separate different functional areas from each other, in an effort to make the architecture less complex. The first of these three layers is the transport layer which handles routing and transportation of data. Because IMS is based on IP this lower layer is access-independent. This access-independence means IMS can be accessed by all IP access media, such as WLAN, ADSL, and GPRS.

The second layer is the control layer, which is situated on top of the transport layer. In the control layer the IMS Core and gateway functions to other systems are found. The control layer handles session control, authorization, authentication, and accounting. It is also called the signaling plane.

The third, and topmost, layer is the application layer where the application servers providing services to end users reside. It is important to note once again, that IMS standardizes service capabilities instead of services. In comparison, in GSM services are either standardized or have a proprietary implementation, but even if they are standardized there are no guarantee that the service will work when roaming to another network [23]. An overview of the IMS architecture is shown in Figure 2.1.

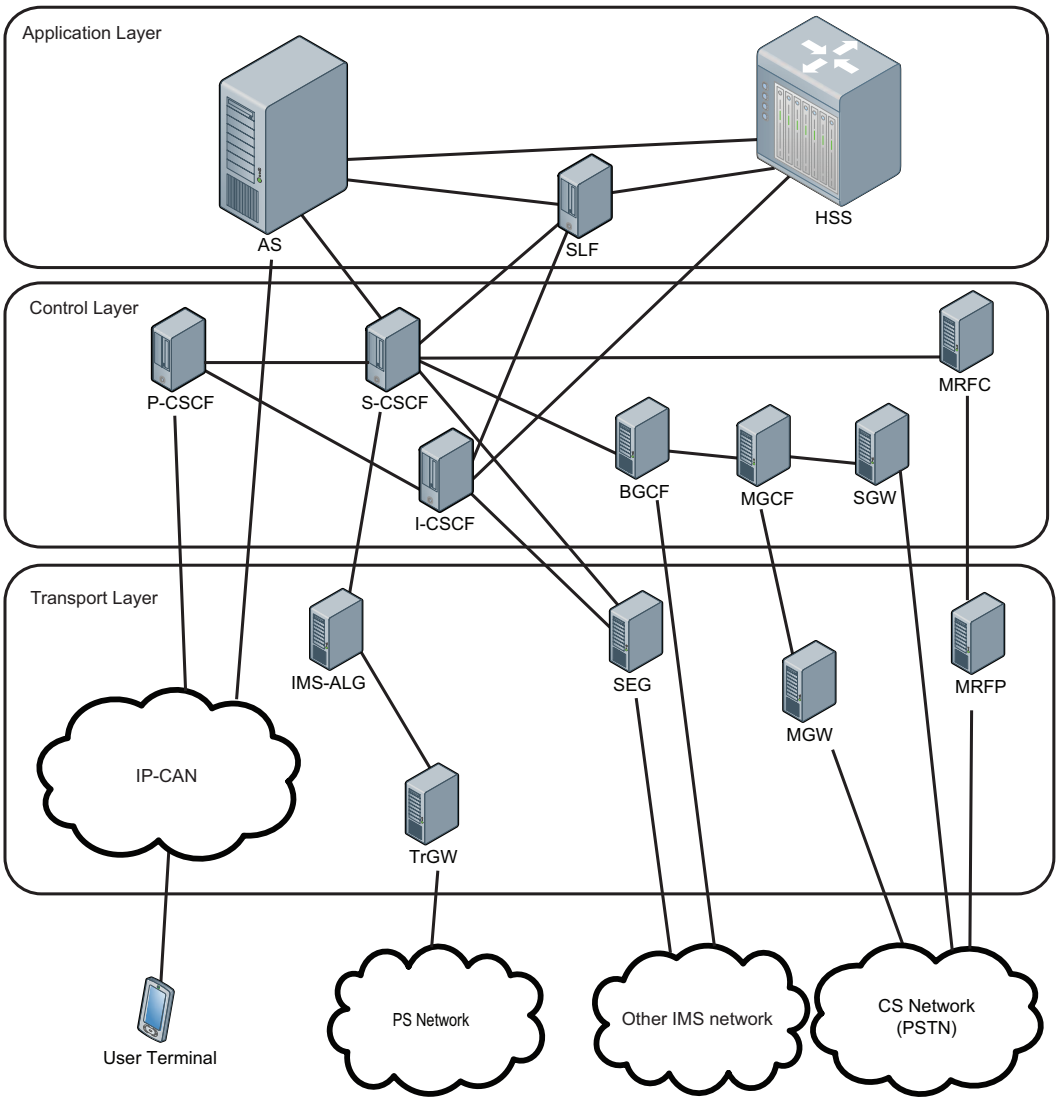

Figure 2.1. IMS Architecture

The core of IMS consists of the Call/Session Control Functions (CSCFs) and the Home Subscriber Server (HSS). Together these elements handle routing, session

handling, and AAA [6].

There are many functions in IMS that handle different interfaces or gateways to other systems, such as gateways to/from the circuit switched networks (PSTN or GSM) and IPv4 transition gateways. Each of these gateways uses specific protocols to communicate with the world outside IMS. Because this thesis is focused on an IMS service utilizing the HSS, these gateways and interworking elements are outside the scope of this thesis and will only be briefly mentioned.

The IMS architecture does not define nodes - but rather defines functions. These functions need not be implemented in separate nodes, rather the mapping of functions to nodes is a design choice during the implementation and configuration of an IMS network. The actual execution of a function can be spread over a number of nodes in order to balance the load. Conversely, in smaller networks many functions can be found on one single node (see for example [38]).

The important parts of IMS are:

- The Home Subscriber Server (HSS) where the master copy of the subscriber database is stored.
- SIP server(s) with Call/Session Control Functions (Proxy, Interrogating, and Serving - as explained in the next section).
- Application Servers (ASs) providing services.
- Media Resource Function (MRF), which consists of a MRFC and a MRFP, that provides media manipulation, such as transcoding.
- PSTN gateway consisting of MGCF, MGW, and SGW. These allow IMS terminals to make and receive calls to and from the PSTN.
- Border Gateway Control Function (BGCF) that enables sessions, initiated from IMS terminals, to find the correct addresses in a circuit-switched (i.e. PSTN) that is reached via another IMS network.
- IMS Application Layer Gateway (IMS-ALG) and Transition Gateway (TRGW) provides translation to networks with other IP versions.

2.5 Call/Session Control Functions

The Call/Session Control Functions (CSCFs) are the core of the IMS network and provide a Session Initiation Protocol (SIP) signaling layer. The CSCFs are SIP servers that process and route all SIP messages in IMS. CSCFs can classified into one of three categories: proxy, interrogator, or serving. Each of these categories is described briefly below.

2.5.1 Proxy-CSCF

The Proxy Call/Session Control Function (P-CSCF) is the first point of contact for the user terminal. A P-CSCF is assigned to the terminal after its initial registration in the network, and all subsequent SIP signaling passes through this P-CSCF. Upon contact the P-CSCF authenticates the IMS terminal and establishes an Internet Protocol Security (IPsec) security association with it. Other nodes in the IMS network know that the P-CSCF will have done this, thus they can rely on the IMS terminal's identity having been established. The P-CSCF inspects the correctness of messages from the terminal to ensure their correctness, it also modifies these messages to preserve privacy of the terminal. The P-CSCF compresses and decompresses all SIP messages sent to and from the IMS terminal; this is done with Sig-Comp ([39], [27]). P-CSCF may include a Policy Decision Function (PDF) to manage QoS on the media plane. It can also generate charging records. One P-CSCF can serve a number of IMS terminals [23].

2.5.2 Interrogating-CSCF

The Interrogating Call/Session Control Function (I-CSCF) is the entry point for traffic into a network and is located on the edge of the core network. The address of a network I-CSCF can be resolved through a DNS query (e.g. this might occur when a SIP server tries to forward a SIP message). When a SIP message is received by the I-CSCF it queries the HSS for the message's recipients and their corresponding S-CSCF(s), then forwards the message to the appropriate S-CSCF(s). The connection to the HSS is called the Cx interface and it enables the I-CSCF to access subscriber specific data. If the HSS is distributed over a number of nodes, then the I-CSCF first queries an Subscription Location Function (SLF) (over the Dx interface) to find the appropriate HSS. Both the Cx and Dx interfaces use the Diameter protocol (described further in § 4.5).

2.5.3 Serving-CSCF

All SIP messages sent in the IMS traverse the Serving Call/Session Control Function (S-CSCF). The S-CSCF provides routing services and decides whether a session should be routed to an Application Server (AS) to be provided with a specific service or not. The S-CSCF plays a crucial part in handling SIP registrations; note that registration binds the respective user's SIP URI to an IMS terminal (e.g. specifically the IP address of the terminal is added to the list of bindings associated with this user's SIP URI). Note that a given SIP URI can be associated with multiple IMS terminals. The S-CSCF inspects all SIP signaling messages, thus the S-CSCF can enforce the network's policies based on both the specific user's permissions and the operator's policies. The S-CSCF connects to the HSS over the Cx interface. This connection is used to download authentication vectors (in order to authenticate users), user profiles, and service profiles. The service profiles can contain triggers

that the S-CSCF matches against incoming SIP messages in order to trigger events. (Details of event processing are found in § 2.9.)

2.6 Home Subscriber Server

The Home Subscriber Server (HSS) is a database that contains the data needed for user authentication and authorization, along with other subscriber related data. Thus the HSS stores user profiles, bindings between the subscriber's URI and their current location information, and the associated S-CSCF of every subscriber. As noted earlier, the HSS can be distributed, in this case a Subscription Locator Function (SLF) is used to route requests to the appropriate HSS. Communication with the HSS is done over the secure Cx and Sh Diameter interfaces, while communication with the SLF is done over the Dx interface. The Sh interface allows an AS to query the HSS. The Sh interface can also be used between multiple AS applications. For further details of the Sh interface see section § 4.6.

In addition to the user profile with the private user identity, the HSS also stores service profiles with initial filter criteria. These filters include trigger points which specify which AS is to act on this trigger (more about identities in § 2.8 and more filters and triggers in § 2.9).

2.7 Application server

The application server hosts and provides services to the network. In IMS there are three types of application servers. Two of them (the OSA-SCS and the IM-SSF, see below) are there to provide services available in older networks to the IMS. In this thesis focus is put on in the third type, the IMS native AS (Subsequently referred to in this thesis simply as an Application Server (AS).).

Open Service Access - Service Capabilities Server (OSA-SCS)
: A gateway functionality to an application server in the OSA framework. This is used to be able to offer services from the OSA AS in the IMS network.

IP Multimedia-Service Switching Function (IM-SSF)
: This AS provides a gateway to networks implementing Customized Applications for Mobile network Enhanced Logic (CAMEL) services, which are used extensively to provide services in GSM networks.

Application Server (AS)
: A native IMS SIP Application Server hosts IMS based services to users. The IMS architecture allows both network operator and third party service providers to offer services to users.

2.7.1 Operational modes

An application server can act as an end user, caller, or callee, depending on what services the AS is to deliver. These roles are realized by different operational modes, specifically: SIP Proxy, SIP UA, SIP Redirect, or SIP Back-to-Back UA mode. Each of these modes will be briefly described below.

2.7.1.1 SIP UA

The AS can act as a User Agent (UA) for sessions either originated in the subscriber's home network or in another network. An example would be an IMS terminal checking for messages on a voice mail server - in this case the AS acts as a UA server to receive SIP messages from the IMS terminal. The AS can also act as a UA caller (client), for example to provide a wakeup call service. Figure 2.2 shows the AS acting in UA mode.

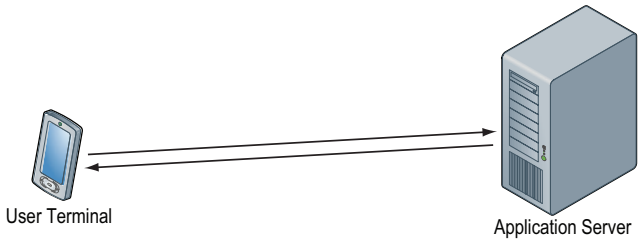

Figure 2.2. The Application Server is acting as an user agent.

2.7.1.2 SIP Proxy

In this mode the terminal or the S-CSCF includes the AS in the SIP signaling path in order to offer a service. For example, the service could change the destination addresses or other header information in the SIP messages according to a user profile or filter criteria. The AS can modify the first SIP message to include itself in the path (in order to get all the following messages in the session - becoming a stateful proxy), or only modify the first message SIP message and be left out of remaining session messages. Examples of services with an AS working in this mode include offering a call-forwarding service. Figure 2.3 shows the AS acting as a proxy for the entire session while in Figure 2.4, the AS handles only the initial message and is left out of the rest of the session.

2.7.1.3 SIP Redirect

As a redirect server, the AS can respond to incoming SIP messages with a message that informs the caller that the recipient is available at another SIP address. A usage example could be a call-forwarding service (were in this case the subscriber is notified about a alternate destination address). In Figure 2.5 the Application

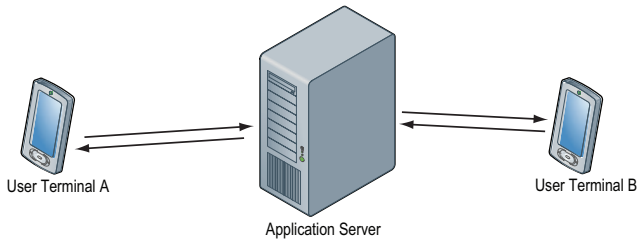

Figure 2.3. The Application Server acts as a proxy and stays in the signaling path.

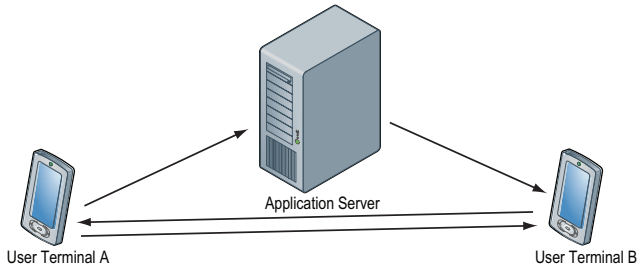

Figure 2.4. The AS handles the first message but then leaves the signaling path.

Server receives an incoming request and responds with another SIP address where the callee can be reached.

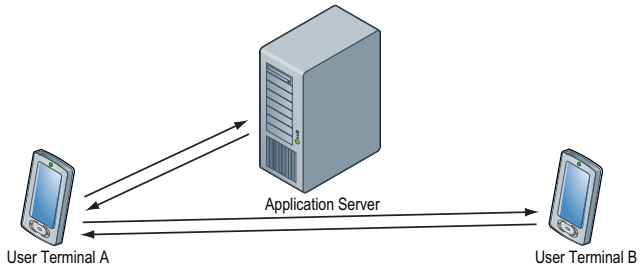

Figure 2.5. The Application Server acting as a redirect agent.

2.7.1.4 SIP Back-to-Back UA (B2BUA)

When acting as a B2BUA, the AS works much like in SIP Proxy mode, but with two differences. It has server logic that can make decisions based on the incoming message or other criteria and it is able to change header information in the SIP messages. This enables the B2BUA to conceal the caller's or callee', user information from each other. Examples could be an anonymisation service or a prepaid call service which wants to check the subscriber's credit before establishing the SIP session. In Figure 2.6 the AS is acting in B2BUA mode.

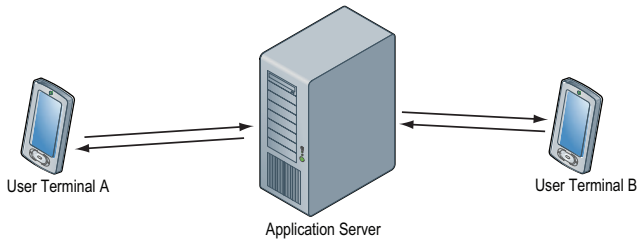

Figure 2.6. An AS acting in Back-to-Back User Agent mode.

2.8 Identities

Each subscriber has one private identity and one or more public identities. Additionally, an AS can have a Public Service Identity (PSI) without having a private identity. Each of these types of identities are discussed in the subsections below.

2.8.0.5 Private Identity

A private identity is used for registration, authorization, administration, and accounting purposes. This is the identification of the subscription, not the user, and has the form of a globally unique Network Access Identifier (NAI) RFC 2486 [10], `subscriber@operator.com`. The NAI is stored in the IP multimedia Services Identity Module (ISIM) in the IMS terminal and in the HSS database. It can not be modified by the subscriber. This private identifier is not used for routing of SIP messages (See for example 3GPP TS 23.228 version 2 [5].).

2.8.0.6 Public Identity

The public identity is used for SIP message routing in IMS. It can be either a SIP URI or a TEL URI.

SIP URI might look like `sip:firstname.lastname@operator.com`.

TEL URI might look like `tel:+46-8-123456` - representing a phone number in IMS. TEL URIs are used to dial PSTN users from IMS, it is also the only type of URI reachable from the PSTN (where it is only possible to dial digits).

2.8.0.7 Public Service Identity

The Public Service Identity (PSI) can be issued to Application Servers (AS) enabling them to receive SIP messages (for example, when a AS is to act as a UA). PSIs have the same form as a Public Identity, but are not tied to private identities as they can be created on the fly for different entities that want to be reachable from IMS.

2.9 Filters and triggers

Filter criteria are a part of the users, service profile stored in the HSS. Filters enables the S-CSCF to trigger the forwarding of SIP messages to an AS. In 3GPP's, specifications 3GPP TS 23.218 [3] there are two types of filter criteria: initial and subsequent filter criteria. The Initial Filter Criteria (IFC) are evaluated by the S-CSCF upon receiving the first SIP message in a SIP dialog (or a standalone request like `MESSAGE`). Based on the result of the evaluation of the trigger, the S-CSCF could decide to route the SIP message to an AS to provide a specific service to the session [23]. Figure 2.7 shows the relation between service profiles and initial filter criteria.

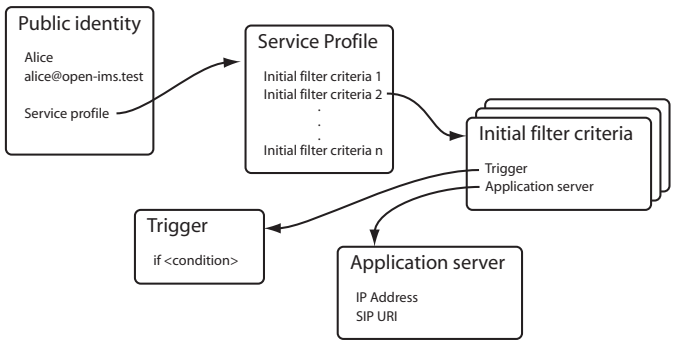

Figure 2.7. Subscription profiles

2.9.1 Service profile

All subscribers are identified by one or more public identities. These identities can be attached to a service profile to provide a specific service. The service profile contains one or more IFCs that forward matching traffic to an AS. Each IFC contains a trigger and specifies the AS that would be involved in the session. The trigger can be matched against different fields of a SIP message, e.g Request-URI, SIP method, and SIP header. Listing 2.1 shows an example of a IMS service subscription for subscriber Alice. The IFC is a boolean condition built up by a number of fields, `ConditionTypeCNF` that can be conjunctive or disjunctive normal form, `Group` that indicates if the set is boolean AND or OR depending on the `ConditionTypeCNF`, and `condition negated` which negates the entire condition. In the example all INVITE and PUBLISH requests for subscriber `alice@open-ims.test` will be sent to the specified application server.

```
<IMSSubscription>
  <PrivateID>alice@open-ims.test</PrivateID>
  <ServiceProfile>
    <PublicIdentity>
      <Identity>sip:alice@open-ims.test</Identity>
      <Extension>
        <IdentityType>0</IdentityType>
```

```
        <Extension>
          <DisplayName>Alice Surname</DisplayName>
        </Extension>
      </Extension>
    </PublicIdentity>
    <InitialFilterCriteria>
      <Priority>1</Priority>
      <TriggerPoint>
        <ConditionTypeCNF>1</ConditionTypeCNF>
        <SPT>
          <ConditionNegated>0</ConditionNegated>
          <Group>0</Group>
          <Method>INVITE</Method>
          <Extension></Extension>
        </SPT>
        <SPT>
          <ConditionNegated>0</ConditionNegated>
          <Group>0</Group>
          <Method>PUBLISH</Method>
          <Extension></Extension>
        </SPT>
        <SPT>
          <ConditionNegated>0</ConditionNegated>
          <Group>1</Group>
          <RequestURI>sip:alice@open-ims.test</RequestURI>
          <Extension></Extension>
        </SPT>
      </TriggerPoint>
      <ApplicationServer>
        <ServerName>sip:as.open-ims.test:5050</ServerName>
        <DefaultHandling>0</DefaultHandling>
      </ApplicationServer>
      <ProfilePartIndicator>0</ProfilePartIndicator>
    </InitialFilterCriteria>
  </ServiceProfile>
</IMSSubscription>
```

Listing 2.1. A service subscription for user Alice Surname.

2.10 Protocols used in IMS

During the standardization of IMS, 3GPP decided that instead of creating new specialized protocols they would try to use existing protocols [23]. By using widespread robust protocols both time and money that would have to be spent on standardization and development were reduced. Consequently IMS is built upon well known protocols such as TCP, SCTP, RTP, RTCP, SIP, Diameter, etc. An introduction to the protocols most relevant to this thesis are given in the following subsections.

2.10.1 Session Initiation Protocol (SIP)

SIP, described in (RFC 3261 [41]), was chosen as the signaling protocol in IMS. SIP borrows properties from HTTP and SNMP and is, like HTTP, text-based and easy to extend and debug. It is a client-server protocol with a request/response schema [23].

SIP is, as the name implies, a protocol for negotiating and creating sessions. It is used to set up, modify, and tear down media sessions between two or more participants. In IMS, SIP is used together with the Session Description Protocol

(SDP). SDP delivers a description of what kind of session is to be set up and what kinds of media streams the session wishes to initiate. During the session SIP can be used to modify aspects of the session, such as the types and numbers of streams, number of participants, destination IP addresses and ports, and other characteristics.

SIP has an event notification framework with both subscriptions and triggers. These features make it possible to subscribe to an event and later be notified when the event occurs. This notification ability is defined in RCF 3265 [40].

2.10.2 Diameter

The Diameter protocol is used in conjunction with Authentication, Authorization, and Accounting (AAA) in IMS. Diameter is described in detail in chapter 3.

2.10.3 RTP and RTCP

The Real-time Transport Protocol (RTP) is a protocol for sending real-time data, such as audio and video. It was developed by the IETF and initially specified in RFC 1889 [24] later this was replaced by RFC 3550 [42]). RTP uses the User Datagram Protocol (UDP) for transport. UDP is used because packets may be lost and there is not sufficient time for retransmission and there is no need for retransmission. Additionally, packets may arrive out of order and applications need not wait for the missing data to arrive if they need to play out the contents of the packets that have arrived on or before they need to be playout. Packet-loss and delays are monitored and information about them is sent over Realtime Transport Control Protocol (RTCP). RTCP-data can be used by the sender to tune its sending data-rate and by the receiver to compute the size of the playout buffer and for other adaptations in order to provide the best playback experience [42].

Chapter 3

Diameter: Base Protocol

Diameter is the protocol responsible for Authorization, Authentication, and Accounting (AAA) in IMS. This chapter will cover the most important aspects of the Diameter protocol.

3.1 History

Diameter is a successor of an older AAA protocol called Remote Authentication Dial In User Service (RADIUS). RADIUS was designed to make it easier for larger networks to provide centralized access management. Diameter is not based on RADIUS, but represents a transition with some backward compatibility. The name Diameter is a hint to Diameter being a successor to RADIUS, being a better and newer AAA protocol. The name itself is a pun because in geometry the diameter of a circle is twice its radius.

3.2 Overview

The Diameter Base protocol, specified in RFC 3588 [13], is called a base protocol because it does not go into details of specific services. The base protocol only consider clients handling authentication and authorization of users on behalf of a server, it does not go into the details of the messages. The specification defines the structure of messages and their attributes, and defines a basic set of messages. In addition, the base protocol defines the concept of applications. Applications are based on the Diameter Base and extends its functionality with additional protocols, procedures, and services. Application examples include the Diameter Mobile IPv4 Application (RFC 4004 [12]) and Diameter Network Access Server Application (RFC 4005 [14]). All applications must support the base protocol functionality, and use the routing and message structures provided by the base protocol.

Diameter does not provide end-to-end security, this must be provided by another protocol such as TLS or IPsec.

All Diameter traffic could be protected end-to-end by using TLS on the transport layer or IPsec on the network layer, but doing so would make it impossible to forward and route messages using information other than what TLS and IPsec provide.

Instead all connections between Diameter Peers (hop-by-hop basis or direct connections) are protected with IPsec or TLS, and only specific data elements in the messages are encrypted to make routing and forwarding possible. This scheme does not only solve the immediate security issues, but also enables Diameter traffic to be routed and handled in a roaming scenario while passing through foreign networks carrying encrypted data [34].

3.3 Diameter Message

The messages sent by Diameter contains a header, describing the type of message, and a number of Attribute Value Pairs (AVPs) carrying different data depending on the message type. Details of the message header and AVPs are given in the sections below. The over all message is shown in Figure 3.1.

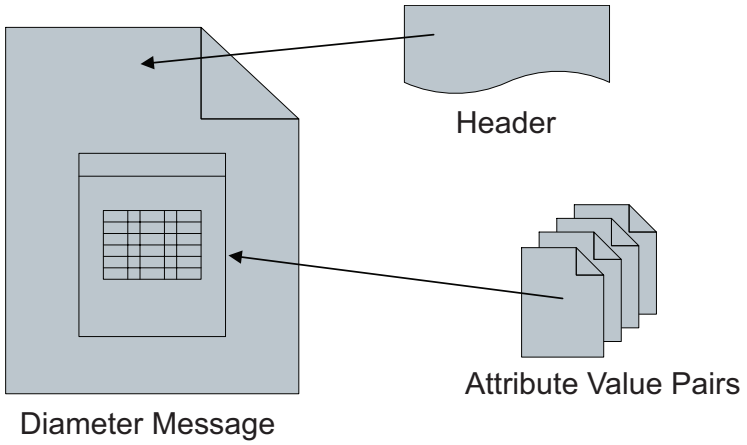

Figure 3.1. A Diameter message with header and Attribute Value Pairs

In Diameter there are only one type of message, thus messages are primarily differentiated by the command code and the request flag within the message. Based upon the command code and request flag it is possible to determine the purpose of the message and whether it is an answer or a request. The Diameter message header is shown in Figure 3.2. The header contains the Diameter version, the total message length (including the header's length), four command flags, and the command code. The command flags carry information indicating if the message is a request or an answer, if it is proxiable or not, if it is an error message, and if it is a possible retransmission. The command code identifies a single Diameter command. Command codes between 0 and 255 are reserved for backward compatibility with RADIUS. Table 3.1 lists all command codes in the Diameter Base protocol [13].

The Diameter header is binary encoded and is sent in network byte order.

3.3.1 Diameter Header

Version	Message length
Command Flags	Command Code
Application-ID	
Hop-by-Hop Identifier	
End-to-End Identifier	
AVPs...	

Figure 3.2. Layout of the Diameter header

The header also contain an Application identifier (Application-ID), an end-to-end identifier, and a hop-by-hop identifier. The Application identifier specifies which Diameter application the message belongs to, e.g. application id 6 corresponds to the Diameter SIP Application [21]. Application IDs are assigned by IANA [29]. The end-to-end and hop-by-hop identifiers are used to match requests to answers (or errors) and to detect duplicate messages. The remainder of the message carries a number of Attribute Value Pairs (AVPs).

Table 3.1. Command Codes defined in the Diameter Base protocol.

Command Name	Abbr.	Code
Abort-Session-Request	ASR	274
Abort-Session-Answer	ASA	274
Accounting-Request	ACR	271
Accounting-Answer	ACA	271
Capabilities-Exchange-Request	CER	257
Capabilities-Exchange-Answer	CEA	257
Device-Watchdog-Request	DWR	280
Device-Watchdog-Answer	DWA	280
Disconnect-Peer-Request	DPR	282
Disconnect-Peer-Answer	DPA	282
Re-Auth-Request	RAR	258
Re-Auth-Answer	RAA	258
Session-Termination-Request	STR	275
Session-Termination-Answer	STA	275

3.3.2 Attribute Value Pairs

The AVPs can be used for different purposes. Some AVPs are used during message routing, such as the `Record-Route` AVP that is both read and modified during

routing. Other AVPs are only of interest to the originating host and the destination host. Some AVPs are encrypted to hide information from nodes other than the originator and the destination host. All AVPs are identified by a Vendor-ID and an AVP-code. Each Diameter application vendor is assigned an integer representing their company and assigned a range of AVPs. If the Vendor-ID is not set, then the AVP-code belongs to the Diameter Base protocol. The three flags indicate if the AVP is vendor specific, encrypted, or if it is mandatory to this Diameter command. The structure of an AVP is shown in Table 3.3. AVPs can carry different types of data, it is up to the vendor to define the details of their AVPs. Diameter has a number of different data types pre-defined, these include strings, integers, octetstrings, and grouped. Grouped is an AVP whose data are a sequence of AVPs. For example, if the Vendor-ID is set to `10415` and the AVP-code is set to `700` it means that 3GPP is the vendor and that AVP-code `700` represents the `User-Identity` AVP that is of the AVP type grouped [7].

<table>
<tr><td colspan="2">AVP-code</td></tr>
<tr><td>Flags</td><td>AVP-Length</td></tr>
<tr><td colspan="2">Vendor-ID (optional)</td></tr>
<tr><td colspan="2">Data...</td></tr>
</table>

Figure 3.3. Structure of an AVP.

3.4 Roles and Peers

There are a number of roles in the Diameter infrastructure that are involved in message processing. These are the roles specified in the base protocol [13]:

Diameter Node
A host process that implements the Diameter Base protocol.

Diameter Peer
A Diameter node that has a direct connection with another Diameter node.

Diameter Client
A device that performs access control. The end user is never the Diameter Client because he/she is not participating in the Diameter signaling.

Diameter Server
A device that handles AAA requests for a realm. It must support the Base protocol and all Diameter applications that it is responsible for.

Diameter Agent
Devices that provides proxying, relaying, redirecting and translation services to Diameter messages. More on different agents in section § 3.8.

Diameter peers can be either configured manually or discovered dynamically. All Diameter peers must support manual configuration of peers, while dynamic discovery is up to the implementation. IETF suggests two types of dynamic discovery: Service Location Protocol (SLP) (described in RFC 2608 [25]) and the DNS NAPTR (RFC 2915 [36] and RFC 3403 [35]).

A connection between two peers is called a connection. Sessions are end-to-end connections between a client device and a server, see Figure 3.4. The Diameter nodes keep track of outstanding requests by storing the message's unique end-to-end and hop-by-hop identifier.

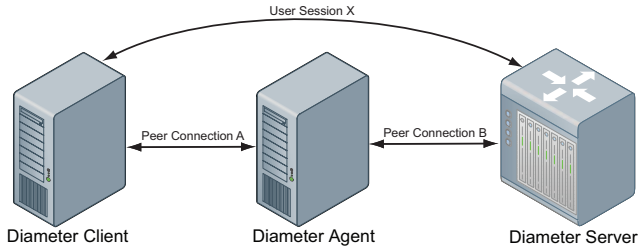

Figure 3.4. Dimeter connections and sessions.

Note that *any* Diameter node can initiate a request (and thus becomes a client in a server-client relationship).

Each Diameter node has a list of known peers, stored in a peer table, but only maintain connections with a few of these peers to save resources. The peer table contains information about the peer, identity, capabilities (see § 3.5), a notation indicating whether the peer was statically configured or dynamically discovered, and other state information. Typically, a peer maintains two sessions. A primary and a secondary session depending on the specific implementation and the Diameter network's configuration. Agents and nodes that handle routing also maintain a realm-routing table, used to forward messages on a hop-by-hop basis. For every session that a peer has, it stores the end-to-end identifiers used by messages it has originated, until it receives a corresponding answer. For every connected peer a list of hop-by-hop identifiers are kept until an answer message has been received.

Open peer connections need to be kept alive, this is done by sending so called `Device-Watchdog` messages. If a peer fails to answer a periodic keep alive message called a `Device-Watchdog` request (DWR) with a matching `Device-Watchdog` answer (DWA), then the peer is considered dead and the session is closed. If this happens to the primary or secondary Diameter peer, then a connection to another peer is established to replace the disconnected peer [34].

3.5 Capabilities exchange

When two Diameter peers first establish a session over TCP or SCTP they inform each other of their capabilities using the `Capability-Exchange` request (CER)

and `Capability-Exchange` answer (CEA) messages. The request indicates all the applications and capabilities the sender supports. When received, the receiving peer removes the applications and capabilities it does not support and returns the remaining ones in an answer message. The peers' capabilities are remembered for the entire session and are used to avoid sending AVPs that the peer does not support (and would discard). If the peers do not share any capabilities or do not share common security mechanism, then the session is not established [13].

3.6 The Peer state machine

Every Diameter peer has to implement the state machine described in RFC 3588 [13]. While the implementation may vary, the behavior and interaction with other Diameter peers has to follow the Diameter specifications.

3.7 Message processing and routing

Diameter message routing is mainly based on the `Destination-Host` AVP and `Destination-Realm` AVP. These AVPs are used when the request is not sent directly to the receiving peer (i.e., the CER or a DWR). When a message is destined for a specific server in a specific realm, then both AVPs are used. When routing to an unspecified realm server, a server responsible for the realm, within a specific realm, then the `Destination-Host` AVP can be omitted. The `Destination-Realm` AVP must also be present in all proxiable messages in order to make it routable. When a subscriber requests authorization to use a network resource he sends a request to a Diameter client. To do so, the client sends an authorization request (defined in a Diameter application e.g., NASREQ see RFC 4005 [14]) to the Diameter server. The request (and all subsequent messages for that user) contains the `Session-Id` AVP which enables servers and clients to correlate the Diameter message with a specific user. Cases when a specific server, specified by the `Destination-Host` AVP, is used include a peer who is receiving an authentication procedure involving multiple messages (more than one request and answer), two peers using a pre-established session key, or when a server initiated the message (for example when a user session is to be terminated for a specific subscriber) [34].

3.8 Diameter Agents

Diameter agents are mainly used for routing purposes. Diameter agents help Diameter clients forward Diameter messages to destination realms or hosts unknown to the user of the network service. They can also route Diameter messages to hosts that provide the services requested by the client. Diameter agents can also be used for load balancing or to multiplex a number of Diameter messages into fewer messages in order to send them to a destination server. All Diameter agents must implement the Diameter base protocol and maintain transaction state in order to

handle retransmissions. Four types of agents exist: relay agent, proxy agent, redirect agent, and translation agent. While they are quite similar, they preform different tasks [34].

Relay Agent

The relay agent routes Diameter requests and answers based on routing information stored in its own realm routing table. The realm routing table contains entries for the supported realms and known peers. The relay can change routing information AVPs in the Diameter messages that it routes, but cannot change any non-routing related AVPs. The relay agent inserts its own identifier in the `Record-Route` AVP (if this AVP exists in a Diameter message, or adds a `Record-Route` AVP with its own identifier - a FQDN). The `Record-Route` AVP is a list of all Diameter nodes that the message has passed, and is used for loop detection. If a node's identifier already is in the `Record-Route` list when a message is received, then the message has been routed in a loop - therefore the message is dropped and an error message is sent back to the sender. The relay agent advertises a Relay Application Identifier in order to enable other Diameter peers to find the relay agent. A relay agent does not maintain session states, but must maintain transaction states in order to be able to handle retransmissions.

Proxy Agent

Proxy agents are much like relays, but are able to edit all AVPs in passing Diameter messages. Therefore, a Diameter Proxy can enforce network policies and hide (i.e., remove) certain AVPs that contain network specific information. To better enforce policies, the Diameter proxy agent maintains peer session states on the side facing the network's perimeter. Because of the need to modify AVPs, a proxy agent can not allow end-to-end security by access devices.

Redirect Agent

Redirect agents are the least complex of the four types of Diameter agents. In contrast to the Proxy agent, the Redirect agent answers the request with an answer message indicating an error and one or more redirect addresses (of which the sender chooses one). This type of Diameter agent is stateless and does not modify any data in a Diameter message.

Translation Agent

A translation agent provides translation between Diameter and another AAA protocol, such as: RADIUS or TACACS. A translation agent is needed to interface with older RADIUS devices.

3.9 Sending and receiving: Requests and Answers

When receiving a Diameter message the mandatory `Origin-Host` AVP and `Origin-Realm` AVP identifies the sender of the message. If the message is to be sent via proxies or other Diameter agents, then at least one of the following AVPs must be present: `Accounting-Application` AVP, `Authentication-Application` AVP, or `Vendor-Specific-Application` AVP. For routing purposes a `Destination-Host` AVP and/or a `Destination-Realm` AVP has to be included (the destination realm can also be found in the sending user's realm part of the NAI). The NAI, identifies the subscriber and which network domain or realm the subscriber belongs to. For example the identifier `subscriber1@operatorA.com` tells us that `subscriber1` belongs to `operatorA.com`'s network realm.

The Diameter message is sent direct to the receiver if the destination is listed in the peer table. If the destination is not listed in the peer table, then the Diameter message is sent to a Diameter agent to be routed. The Diameter agent checks the `Destination-Host` AVP and/or `Destination-Realm` AVP and makes a routing decision based on its routing table. If the agent can not find a matching routing entry, then a Diameter error message is returned.

A locally unique end-to-end identifier is created to keep track of outstanding requests. For each hop a hop-by-hop identifier is used for the same purpose. Every node adds its node identifier to the messages, `Record-Route` AVP when it passes (as noted earlier makes it possible to detect routing loops).

Messages can be sent directly to the receiver or routed via agents. Routing is done by looking at the `Destination-Realm` and `Destination-Host` AVPs which specify which realm or which server the message is destined for. For example, when a user wants to authenticate themselves, they send a message to the closest Diameter agent (whose IP address is learned by making a DNS query). This agent checks which realm the authentication request should be forwarded to. The agent determines which realm the user belongs to based upon the realm part of the users provided NAI. The agent forwards the Diameter authentication request message to that realm.

Every request has a corresponding answer (or error message). An answer message always has the same command code and end-to-end identifier (and the same hop-by-hop identifier for each hop) as the request. The answer must also include the Result-Code AVP that indicates if the request succeeded or failed in some way [13].

3.10 Error handling

Two types of errors can occur in Diameter. The first type is protocol errors. These are transmission errors and routing errors which often are handled on a hop per hop basis. When a Diameter node encounters an error, it returns an answer message with the Error-bit set (see Figure 3.5). Answer messages with the error bit set are

error messages and the `Result-Code` AVP specifies the error. When the answer is sent back it passes all the previous nodes (according to the Diameter routing rules; see section § 3.7). Any one of the nodes in the traversed path can decide to handle the error message and reroute or retransmit the message (see Figure 3.6).

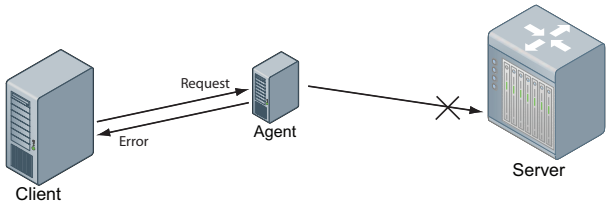

Figure 3.5. Diameter error handling.

The second error type is application errors, which **never** uses the error flag in the Diameter header. Application errors are based on errors in the AVPs or in the application logic. In either case, an answer message with the `Result-Code` AVP set to an appropriate error message is sent back to the sender. Some errors insert their own AVPs with error information into the answer message. For example an application error could occur because the recipient received an unsupported AVP or an AVP contains an invalid value. When one of these errors occurs another AVP called `Failed-AVP`, containing the erroneous AVP is inserted into the answer message. An error may also occur because a mandatory AVP is missing in the request.

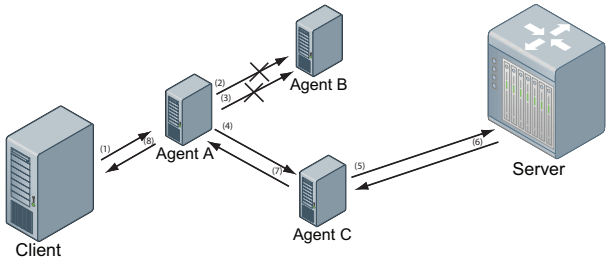

Figure 3.6. Diameter protocol error handling.

(1) A client sends a request to the nearest agent for routing. (2) The agent A tries to send and resend (3) to agent B. (4) Agent A reroutes and sends the request to agent C. (5) Agent C sends the request to the destination server. (6)(7)(8) The answer message is passed back to the client.

3.10.1 Error result codes

The `Result-Code` AVP is present in all answer messages and tells the sender the result of the request. If the result type is any other than success, then the `Error-Reporting-Host` AVP is included to identify the origin of the error message (if it is not the same as the Origin-Host AVP). The result code contains a number

Table 3.2. Diameter Error codes

Category	**Error code range**	**Description**
Informal	1000-1999	Errors in this category tells the sender that the request did not succeed and that it will require additional actions. An example is an authentication mechanism that requires more than one message to complete.
Success	2000-2999	Reports that the request succeeded.
Protocol errors	3000-3999	Protocol errors are mostly transmission and routing errors and can be solved by proxies on a per hop basis.
Transient failures	4000-4999	This category of errors are for requests that did not succeed when they was received, but it may succeed later (if a new request is done). E.g a wrong password was entered.
Permanent failures	5000-5999	These errors inform the peer that the request failed and it should not be tried again. E.g, missing or incorrect AVPs sent in a request.

that specifies what caused the error. There are five categories of result codes (and ranges) see Table 3.2 [13].

3.11 Authentication, Authorization, and Accounting

Authentication, Authorization, and Accounting (AAA) concern access control, policy enforcement, and auditing for computing & communication systems. The AAA subsystem is responsible for verifying users, validating identities, checking access rights, and if a user has the right to access a given resource - then granting permissions to use the resources (Note that resources can be devices, connectivity, services, etc.). Accounting information is often collected when resources are used - in order to later generate a bill. In some cases this billing is done in real-time, thus usage of a resource might be terminated when an account limit is reached. The Diameter base protocol supports two types of services for Diameter applications. First is authentication and/or authorization and optionally accounting. The second is accounting only. Diameter applications must define their own AVPs for authentication and authorization or reuse other AVPs from other applications (as none exist in the Diameter base protocol).

Authentication and authorization are closely coupled in Diameter. An authentication (shortened as "Auth") request message is used for authentication or authorization, or both. The exact usage depends on the `Auth-Request-Type` AVP in the message. The result of an authentication request is quite simple, it can only succeed or fail. Authorization on the other hand grants one-time access or give access to a resource for a period of time. The Diameter base protocol has support for user terminations, server abortions, and re-authorization to extend the authorization time. Authorization for single access events can be handled by stateless Diameter servers; while stateful servers are required for maintaining authorization sessions over a period of time.

Accounting can be divided into two categories. First, one-time events (that start and end at the same time) and second, sessions are events with a duration (having a start and an end time). The first event type only requires a single `Accounting` request (ACR) while the second requires messages to communicate starting and ending times. Additional messages for status updates during the session are also supported [13][34].

3.12 IPsec and TLS

All Diameter implementations must support IPsec (defined in RFC 4301 [32] and RFC 4309 [28]). All Diameter servers must support Transport Layer Security (TLS) (RFC 4346 [16]), although this is optional for Diameter clients. IPsec and TLS are sufficient for networks without untrusted third party agents [13]. However, if untrusted parties exist, the additional end-to-end security is needed. For more information see RFC 3588 [13].

Chapter 4

Diameter SIP Application

In this chapter the Diameter SIP application (RFC 4740 [21]) will be examined. Specifically descriptions for how the Diameter SIP Application is used in IMS for communication between different nodes and functions will be given.

4.1 General Architecture

The main purpose of the Diameter SIP application is to expand the Diameter base protocol to meet the AAA requirements described in the specifications for SIP (RFC 3261 [41]) and 3GPP (3GPP TS 29.229 [2]). The specifications include authentication, authorization, and ways to handle resource access management. These requirements are not met by the Diameter base protocol nor any of the other existing Diameter applications, therefore the Diameter SIP application was designed.

The SIP application supports SIP user agents using HTTP-digest as an authentication mechanism (see RFC 2617 [20]) as stated in the SIP specification (RFC 3261 [41]). Accounting in IMS is not handled by the Diameter protocol, but by an IMS specific counterpart. However, accounting information is sent to these IMS-accounting servers using Diameter. Accounting is not examined further in this thesis.

In the 3GPP specifications, a single (logical) Diameter server (the HSS in the case of IMS) can be distributed over several nodes for increased scalability, hence a Diameter Subscriber Locator (SL) (called a SLF in IMS) redirects Diameter requests to the correct Diameter server. The SL is in fact a Diameter redirect agent (as described in section § 3.8) that knows where a specific subscriber's information data is stored and replies with this address in a redirection reply.

4.2 New Command Codes

Diameter SIP Applications utilize a set of new Command Codes (shown in Table 4.1). These command codes must be supported by all implementations of SIP

Diameter Applications. These command codes add functionality that IMS requires and that the Diameter base protocol does not offer.

Table 4.1. Diameter SIP Application Command Codes

Command Name	Abbr.	Code
User-Authorization-Request	UAR	283
User-Authorization-Answer	UAA	283
Server-Assignment-Request	SAR	284
Server-Assignment-Answer	SAA	284
Location-Info-Request	LIR	285
Location-Info-Answer	LIA	285
Multimedia-Auth-Request	MAR	286
Multimedia-Auth-Answer	MAA	286
Registration-Termination-Request	RTR	287
Registration-Termination-Answer	RTA	287
Push-Profile-Request	PPR	288
Push-Profile-Answer	PPA	288

4.3 New Attribute Value Pairs

New AVPs were created for use in the Diameter SIP application. See Table 4.2 for a summary of the new AVPs. The Diameter SIP Application specifications also define several new values for existing AVPs. These new values are used in the `Result-Code` AVP, `Transient-Failure` AVP, and `Permanent-Failure AVP` to better describe SIP related errors and results.

Table 4.2. A selection of Diameter SIP Application Attribute Value Pairs

Attribute name	AVP-code	Datatype
SIP-Server-Capabilities	372	Grouped
SIP-Mandatory-Capability	373	Unsigned32
SIP-Optional-Capability	374	Unsigned32
SIP-Server-Assignment-Type	375	Enum
SIP-Auth-Data-Item	376	Grouped
SIP-Reason-Code	384	Enum
SIP-Reason-Info	385	UTF8StringS
SIP-Visited-Network-Id	386	UTF8StringS
SIP-Supported-User-Data-Type	388	UTF8StringS
SIP-User-Data	389	Grouped
SIP-User-Data-Already-Available	392	Enum
SIP-Method	393	UTF8StringS

4.4 Security

In IMS, both the network signaling core and user traffic use IP and other open protocols, hence more security threats are expected compared to when the signaling network was physically separate from the user traffic or when the signaling protocols were not widely known. IP-based protocols are protected by IPsec (RFC 2401 [31]). IPsec provides data integrity, data origin authentication, protection against replay attacks, confidentiality when needed, and limited protection against traffic flow analysis when confidentiality is used. The control plane is divided into security domains, usually with the same borders as the operator's domains. Security domain gateways are located on the domain's borders to secure the IP based protocols. For more information about security see 3GPP TS 33.210 [1].

4.5 Cx and Dx Interface

The Cx interface is the Diameter connection between the I-CSCF, the S-CSCF, and the HSS. Communication across this interface is used to authorize subscribers, to update the subscriber's current location information, and to store other data about the subscriber in the HSS. The Dx interface is the interface between the I-CSCF, the S-CSCF, and the SLF. Both interfaces are specified in 3GPP TS 29.229 [2]. The Diameter commands codes were shown earlier in Table 4.1.

4.6 Sh Interface

The Sh interface (defined in 3GPP TS 29.328 [4]) exists between the AS and the HSS. A connection across this interface gives the AS access to subscriber data stored in the HSS. In addition to the subscriber data the AS can store its own information about subscribers in the HSS (see section § 2.6 for more about the HSS).

There are four new command codes needed for this interface, these extend the Diameter Base protocol. The new command codes are `User-Data` request, which is used to download data from the HSS to an AS. The `Profile-Update` request is used to update data in the HSS. The AS can also send a `Subscription-Notification` request to receive notifications from the HSS when the data is changed. When changes are made to the data, the HSS sends a `Push-Notification` request to the AS containing the updated data [4].

Communication via the Sh interface is handled by these four new command code requests and their corresponding answers. These new command codes and answer codes are presented in Table 4.3.

According to 3GPP TS 29.328 [4] the `Result-Code` AVP is used for all errors defined in the Diameter Base protocol, whilst the grouped `Experimental-Result` AVP is used for all Sh interface errors with `Vendor-Id` set to 10415, indicating 3GPP. The grouped `Experimental-Result` AVP contains an `Experimental-Result-Code` AVP that holds the error code.

Table 4.3. IMS Sh interface specific command codes

Command Name	Abbr.	Code
User-Data-Request	UDR	306
User-Data-Answer	UDA	306
Profile-Update-Request	PUR	307
Profile-Update-Answer	PUA	307
Subscribe-Notifications-Request	SNR	308
Subscribe-Notifications-Answer	SNA	308
Push-Notification-Request	PNR	309
Push-Notification-Answer	PNA	309

4.6.1 User Data over the Sh-interface

The Sh interface and its commands enable the AS to both download and update subscriber data stored in the HSS. Additionally, the AS can subscribe to receive updates when specific subscriber data is updated. The updated data is sent along with the notification of the update. Associated with the stored data is a sequence number that is incremented every time the data is changed. If more than one AS is providing the same service, then all ASs will be able to subscribe to updates making the source of the data transparent to the subscriber. Listing 4.1 shows the content of a `User-Data` AVP in a typical `User-Data` answer or an `Push-Profile` request.

```
<Sh-Data>
  <RepositoryData>
    <ServiceIndication>test</ServiceIndication>
    <SequenceNumber>127</SequenceNumber>
    <ServiceData>Some data to store in the HSS</ServiceData>
  </RepositoryData>
</Sh-Data>
```

Listing 4.1. Contents of User-Data AVP stored in RepositoryData

In Listing 4.2 the contents of the `User-Data` AVP in a `Push-Profile` request sent by the HSS to the AS is shown. The AS has previously subscribed to the subscriber's state of the subscriber with the private identity `sip:alice@open-ims.test`. When this subscriber registered with the IMS Core this changed her IMS user state, hence the HSS pushed this change of status to the AS that subscribed for this notification. The AS has access to a number of different data items associated with each subscriber in the HSS (see Table 4.4).

```
<Sh-Data>
  <Sh-IMS-Data>
    <IMSUserState>1</IMSUserState>
  </Sh-IMS-Data>
</Sh-Data>
```

Listing 4.2. Contents of User-Data AVP in Push-Profile Request

Table 4.4. Data available via the Sh interface

Data	Code	Operation
Repository data	0	D,U,S
IMS Public identity	10	U,S
IMS User state	11	U,S
S-CSCF Name	12	D,S
Initial filter criteria	13	D,S
Location Information	14	D
User state	15	D
Charging Information	16	D,S
MSISDN	17	D
PSI Activation	18	D,U,S
DSAI	19	D,U,S
Aliases Repository Data	20	D,U,S

D = download, U = upload, S = subscribe.

Chapter 5

An IMS Service

In this chapter an example service is created using IMS and Diameter.

5.1 Creating a service

The proposed new service will give users an interactive service. This example service does not demand any real service behind it, for example it could simply return the current time, but for fun the six degrees of Kevin Bacon game was chosen. The main focus of the service is the Diameter interface that connects the application server with the user database. This interface will be tested by storing service permissions in the HSS, on a separate node from where the service is located. The service will apply user permissions to the search from the central user database. These permissions will narrow the searches a user can do by providing one of the two actors in the game. Users provide a name of an actor or actress to the service and as a response get a chain of movies and actors that "connect" them.

5.2 Architecture

To implement this service several IMS nodes/functions have to be present. First of all a platform for the service itself must be available, i.e., an application server. This platform needs to be able to both send and receive text messages from IMS users in order to reach the "Oracle of Bacon" web page, and to query the networks central database (i.e., the HSS).

5.3 Available components

The IMS system used for experiments used OpenIMSCore [15] from The Fraunhofer Institute for Open Communication Systems (FOKUS) [17]. Open-IMSCore consists of the central parts of IMS namely the P-CSCF, I-CSCF, S-SCSF, and the HSS. The application server was implemented using Sun's application server GlassFish [43] and a variety of IMS client applications. The configuration of the testbed is

shown in Figure 5.1. The service executes on the application server. The IMS core uses SIP, which support text messages and routes and delivers SIP messages. The HSS store subscriber service permissions and subscriber service profiles (i.e., configurations for each service).

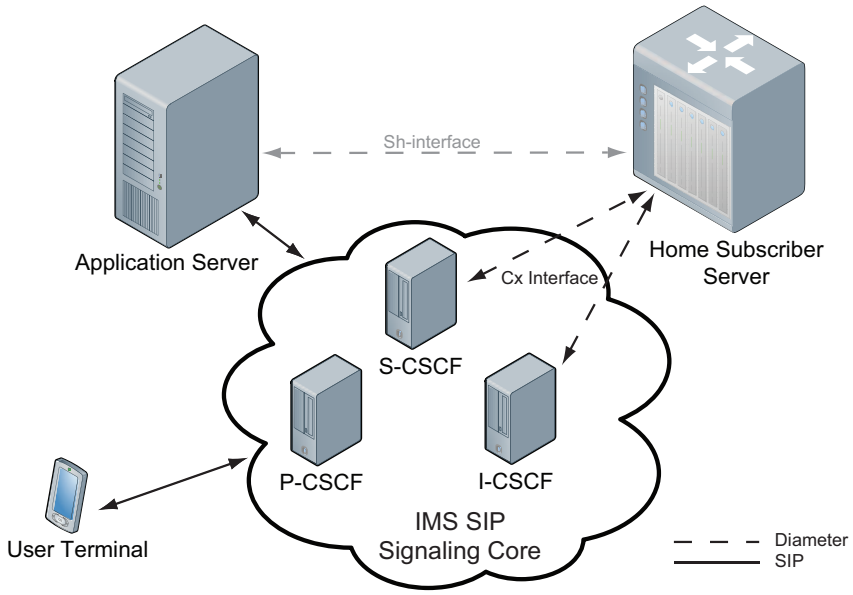

Figure 5.1. Experimental testbed configuration.

5.4 Missing components

In Figure 5.1, the Diameter connection (via the Sh interface) between the application server and the HSS was not implemented on the AS. Thus a Diameter stack implementation was necessary to enable the AS able to communicate with the HSS. The FoHSS [18] from FOKUS is written in Java and implements FOKUS's own Diameter implementation called JavaDiameterPeer [30]. Thus this Diameter stack implementation was also used in the AS, thus ensuring that the two stacks would be interoperable.

5.4.1 JavaDiameterPeer

JavaDiameterPeer [30] is as a Java implementation of the Diameter stack developed by the Fraunhofer Institute for Open Communication Systems (FOKUS). The entire source (with development branches) is available via SVN along with JavaDocumentation. Support is given in mailinglists and discussion forums which are answered by the developers and other users. At this time only peer connections are supported, authentication, authorization, and accounting are not yet implemented, but they are not needed for the Cx/Dx/Sh interfaces of IMS.

Realm routing is not yet supported either, this means that all messages must be sent to the Fully Qualified Domain Name (FQDN) of the destination host. The lack of realm routing functionality makes it unsuitable for a commercial IMS, but will work in a lab environment with one single realm and one HSS.

The implementation creates a Diameter peer using a configuration file which specifies realm, application ID, and the peer's FQDN. The configuration file also specifies the port where the node will accept incoming requests, a peer list, and other settings such as vendor and application ID pairs. JavaDiameterPeer is used in the open source HSS that FOKUS includes in the OpenIMSCore project. JavaDiameterPeer has a sister project called CDiameterPeer which is a Diameter stack implemented in C.

5.4.2 Application Server - Glassfish

GlassFish (version 2) [43] is an open source application server which implements Java Enterprise Edition 5 from Sun Microsystems, Inc [44]. The GlassFish application server supports servlets to dynamically add functionality created using the Java platform. GlassFish has a subproject called SailFin. Project SailFin is based on contributions from Ericsson and Sun Microsystems, which adds SIPservlet functionality to GlassFish.

5.4.2.1 Servlets and SIPServlets

Servlets are plugins to web servers that handle requests, process them, and return a response. Servlets were invented by Sun Microsystems in 1997 [45] to make it possible to add dynamic content to web pages on Sun's Java Web Server. SIPServlets are servlets that can handle SIP requests and responses. Servlets (and SIPServlets) can be deployed on Java web servers in order to create dynamic services and content.

5.4.2.2 Life cycle of a SIPServlet

The servlet is started when the web server is started or the first time the servlet is invoked. When started the `init()` method is called and when the application is stopped, the `destroy()` method is called. The `init()`and `destroy()` methods are only invoked once and are used (respectively) to initialize and clean up resources used by the servlet.

Each time a SIP request is received the `service()` method is invoked to handle each request in a separate thread. The `service()` method takes two parameters, a request and a response. Only one of the parameters is used (the other parameter is set to `null`), depending on which of the parameters are used, one of two different methods are called. These methods are called `doRequest()` and `doResponse()`, these two methods in turn call a number of other methods to implement the request or response. The existence of a base collection of predefined methods makes it easy for developers to override specific methods for their servlets. For example, if

the servlet will handle SIP REGISTER requests, then the `doRegister()` method is overridden. Details of the available methods are found at in the SIP Servlet Specification [11].

5.5 Service Design

In order to deliver this example service, the application server must be able to do three things:

- process incoming requests
- communication with the HSS (to check if the subscriber is permitted to use this service and to get the subscriber's profile), and
- communicate with a web server on the Internet.

To process incoming requests, the AS must be able to send and receive SIP traffic. To be able to talk to the HSS, the AS must be able to send and receive Diameter SIP Application messages. Finally, the AS must be able to communicate with the "Oracle of Bacon" webpage using HTTP.

5.5.1 Overview

A user application running on the terminal creates a SIP message addressed to the AS with a string starting with `\bacon` followed by the name of an actor or actress. The message will be sent via the IMS signaling system to the AS. The AS will query the HSS for the subscriber's "permission", in this service the HSS will respond with the second actor of the Kevin Bacon Game. Given this information the AS creates a HTTP request and send it to the "Oracle of Bacon" web server. This web server will respond with a list of actors and movies. The list from the HTTP response is placed in a SIP message which is sent back to the user's application.

5.5.2 Node responsibilities

The HSS holds the service's "permissions" (and also the user's IFCs - if used). The HSS can store data for many services and users. Data is stored and identified by a combination of a service ID and the subscriber's public identity. When requesting data for a service, a specific public user identity or a wildcarded subscriber identity can be used to retrieve a permission for a specific user or a group of users. This service will use a single Diameter SIP command, the `User-Data` request (UDR), which requests specific service data for a specific user or group of users. The UDR carries a couple of important AVPs (see Table 5.1).

The AS communicates with three entities: the CSCFs, the HSS, and a web server on the Internet using SIP, Diameter, and HTTP respectively.

Table 5.1. AVPs carried in the User-Data request

AVP	Code	Content
User-Identity	700	“sip:alice@open-ims.test”
Service-Indication	703	“test”
Data-Reference	704	“0” (Repository-Data)

5.6 Service Implementation

Almost all functionality required by the service is already offered by the IMS-network. The additional functionality is simply the parts of the AS that shall handle the SIP request and response, the Diameter query to the HSS, and the HTTP request and response to/from the web server. This missing functionality will be implemented in a SIPservlet that will run on the AS.

This SIPServlet must be able to handle `SIP MESSAGE` requests that are addressed to the AS and that contain the string `"\bacon"` (as shown in the first line in Listing 5.2). When a request is received the content is parsed and an actor's name is extracted. This is one of the two actors' names that will be used in the "Kevin Bacon Game". The other actor's name is stored in the HSS and will be fetched by using the Sh-interface.

The AS creates a Diameter SIP `User-Data` request with AVPs containing the public identity of the user that sent the SIP request, a service indication, and data reference. (See Table 5.1 for more details about the AVPs). When the UDR is sent an UDA response will be received. If the UDA reports success, then an actor name is read from the `User-Data` AVP (as shown in Listing 5.1). The actor's name is read from the XML-string and used together with the user provided actor's name to query a web server.

```
<Sh-Data>
    <RepositoryData>
        <ServiceIndication>test</ServiceIndication>
        <SequenceNumber>0</SequenceNumber>
        <ServiceData>Kevin Bacon</ServiceData>
    </RepositoryData>
</Sh-Data>
```

Listing 5.1. Example contents of a `User-Data` AVP in a Diameter `User-Data` answer message.

After the AS has both actors names it creates a `HTTP GET` request and sends it to the "Oracle of Bacon" web server. In response the AS will receive a `HTTP OK` response containing a webpage. The content is parsed and a list of actors and movies are extracted. The AS creates a new `SIP MESSAGE` addressed to the subscriber containing a formatted list of actors and movies. An example is shown in Listing 5.2.

```
Alice: /bacon Peter Stormare

AS: Peter Stormare
    starred in
Fargo (1996)
    together with
William H. Macy
    starred in
Murder in the First (1995)
    together with
Kevin Bacon
```

Listing 5.2. Example of using the service. (Note that this request can return different actors as the connecting link since many actors has worked together with Kevin Bacon and Peter Stormare.)

5.6.1 Fetching and Parsing HTML

In order to generate the SIP message shown above, the SIPServlet creates a URL object with the "Oracle of Bacon" webpage address and the actor's names. The URL (shown in Listing 5.3) is opened using a URLConnection and then read with a InputStreamReader. The InputStream is scanned line by line and from all lines with `<span>` tags with the "class" attribute set to "film" or "actor" the name of the anchor tag is put into one of two lists, one for actors and one for movies. The movies and actor lists are compiled into a string such as the one shown in Listing 5.2 above.

```
http://oracleofbacon.org/cgi-bin/movielinks?game=0&firstname=Kevin+Bacon&secondname=
    Peter+Stormare&using=1&start_year=1850&end_year=2050&dir=0&use_genres=1&g0=on&g4=
    on&g8=on&g12=on&g16=on&g20=on&g24=on&g1=on&g5=on&g9=on&g13=on&g17=on&g21=on&g25=on
    &g2=on&g6=on&g10=on&g14=on&g18=on&g22=on&g26=on&g3=on&g7=on&g11=on&g15=on&g19=on&
    g23=on&g27=on
```

Listing 5.3. An URL to fetch a webpage that shows the link from Peter Stormare to Kevin Bacon.

5.7 Implementation Message flow

Figure 5.2 shows the message flow for this example service. In Figure 5.2 `SIP MSG1` refer to a message similar to line 1 of Listing 5.2 and `SIP MSG2` contains a list of actors and movies, similar to line 3 through 11 in Listing 5.2.

1. The User Agent sends a `SIP MESSAGE`, addressed to the AS, to its designated P-CSCF.
2. The P-CSCF sends the message on to the S-CSCF that is responsible for this subscriber.
3. The S-CSCF, who knows the location (IP address) of the AS, forwards the `SIP MESSAGE`.
4. The AS acknowledges that it received the message with a `SIP OK`.
5. The `OK` is passed back to the original sender.
6. The `OK` is passed back to the original sender.

7. The AS sends a `User-Data` request (UDR) to the HSS.
8. The HSS answers with a `User-Data` answer back to the AS.
9. The AS creates and sends a `HTTP GET` request to the "Oracle of Bacon"-web server.
10. The requested webpage is returned in a `HTTP OK` response.
11. The AS creates a new `SIP MESSAGE`, addressed to the user, and sends it to the I-CSCF to route it to the user.
12. The I-CSCF queries the HSS for the user's allocated S-CSCF with a Diameter SIP `Location-Information` request.
13. The HSS sends the name of the user's designated S-CSCF.
14. The I-CSCF forwards the message to the S-CSCF.
15. The S-CSCF forwards the message to the P-CSCF responsible for this user.
16. The P-CSCF delivers the `SIP MESSAGE` to the user.
17. The user agent acknowledges the message with a `SIP OK`.
18. The `OK` is passed back to the S-CSCF.
19. The `OK` is passed back to the I-CSCF.
20. The `OK` is passed back to the AS.

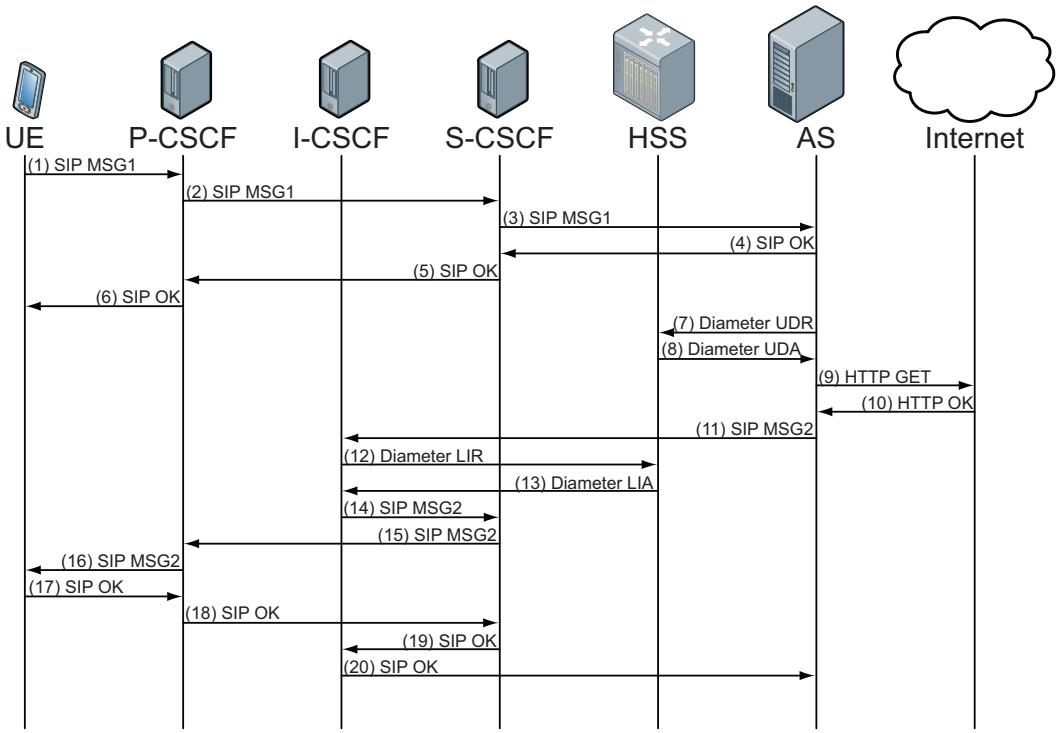

Figure 5.2. Message flow of the service.

5.8 Thoughts for a commercial version of the service

The service as it is currently implemented does not take advantage of the user data subscription abilities of the SIP Diameter Application. By saving the fetched subscriber data in the AS and subscribing to future changes, the number of queries to the HSS will be reduced from one query for every use to two for every new user and then one every time data is updated in the HSS.

When a service request comes from a subscriber, one `User-Data` request (UDR) is sent to the HSS in order to get the initial subscriber data. The AS stores the data and subscribes to changes in it by sending a `Subscription-Notification` request (SNR). All service requests from that subscriber will not generate new requests to the HSS since the subscriber data is locally stored (cached) in the AS. When the subscriber's data is updated in the HSS, then a `Push-Notification` request (PNR) will be sent to the AS to update the cache. This flow is shown in Figure 5.3.

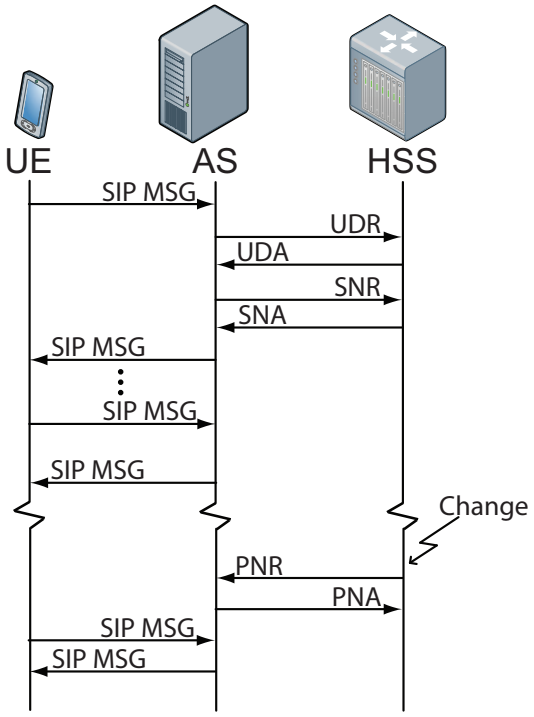

Figure 5.3. Example of message flow with AS caching and subscribing to user data to maintain cache coherence.

Chapter 6

Load tests

In this chapter load tests of the HSS from Fokus [17] are done. These tests were made to determine how well the HSS scales with load. This scaling is important since all ASs can make requests to the HSS for information, hence if the HSS scales poorly then AS based services will have a performance bottleneck.

6.1 FoHSS - a HSS from Fokus

The service implemented in the previous chapter queries the HSS for information regarding specific subscribers. As a result a important question for anyone who wants to deploy an IMS service such as the one described in the previous chapter in a commercial scenario is: How many concurrent users can my HSS handle? This leads to the following set of questions: Will the HSS be a bottleneck when using the service? If so, does the performance depend on the specific implementation of the HSS or does it depend on the database server used by the HSS?

6.2 Test environment

The testbed consists of the FoHSS [18] and a test client. They are connected to a 100Mbit Cisco FastHub 300 Series together with a computer running Wireshark [46], as shown in Figure 6.1. Tests are run and statistics collected from the client. Wireshark can be used to check message contents and to calculate average service response time.

6.2.1 HSS settings

The HSS is running on a PC with Ubuntu Desktop 9.04 64-bit. This PC is fitted with an Asus A8N-SLI motherboard with the nForce4 chipset, equipped with an AMD Athlon 64 3500+ processor running at 2.2 GHz, 2048 MB of memory, and a 41GB Hitachi IDE drive model IC35L040AVER07-0. Both the motherboard's integrated ethernet, and IDE controllers were used or the tests. The HSS has 2000

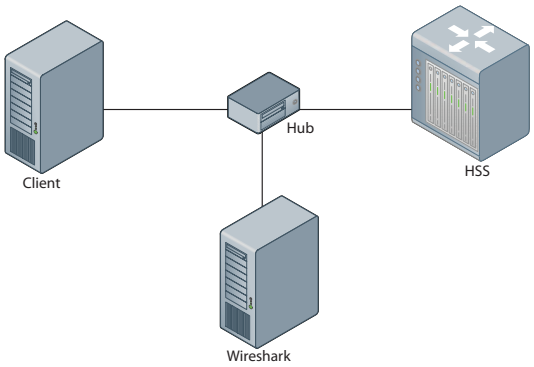

Figure 6.1. Test environment.

users stored in a MySQL version 5.0.75 [37] database. Flags for the servers Java SE 1.6.0 64bit VM are "-Xmx1500m" (which suggests to the Java VM to use 1500 MB Memory).

6.2.2 Test Client

The test client is a Java application that uses JavaDiameterPeer [30] to query the HSS. The client can send Diameter UDR or LIR messages with specified intervals, for a random user (that exists in the HSS database). The client measures the time taken from sending the request to receiving the answer. The client creates a trace with end-to-end identifiers, time from sending to receiving, and time since the client was started. The trace is kept in memory until the run is completed and is then written both to a file, and a graph.

If the client computer has more than one virtual or physical network interface connected to the specified test network it can create one test client thread per interface.

6.2.2.1 Limitations of time precisions

The test client can not measure the sending interval with more precision than 1ms. This is a limitation in the Microsoft Windows platform and impacts the Java VM. To achieve intervals with greater precision more threads with lower interval will be used. For example, instead of having one thread sending with an interval of 4.6ms five threads with 23ms interval will be used.

6.3 Tests

6.3.1 Idle system tests

Testing began with one client querying the HSS with 5 UDR requests/second. The graph in Figure 6.2 shows a steady response time with an average of 10.0ms and a

standard deviation of 6.5.

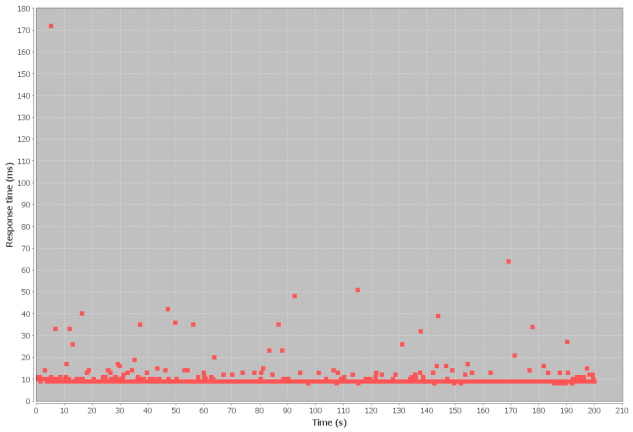

Figure 6.2. Response times of 1000 `User-Data` requests at 200ms intervals.

6.3.1.1 HSS event polling

When idling the HSS makes database queries over the loopback interface every 10 seconds, this can be monitored with an instance of Wireshark running on the HSS. These queries seems to check for updates in subscriber data that might affect any subscriptions on the Sh or Cx interfaces. If data has been changed, then there are some Diameter messages that need to be sent to the subscribing nodes. For the Sh interface the message is the `Profile-Notification` request and on the Cx interface its either a `Registration-Termination` request (RTR) or a `Profile-Push` request (PPR) depending on what change was made to the subscribers data or subscription status. For details about the MySQL queries, see Listing 6.1.

```
SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITED
SET autocommit = 0
SELECT * FROM sh_notification WHERE hopbyhop = 0 LIMIT 1
COMMIT
ROLLBACK
SET autocommit = 0
SELECT * FROM cx_events WHERE hopbyhop = 0 LIMIT 1
COMMIT
ROLLBACK
SET autocommit = 1
```

Listing 6.1. Example of HSS idle database traffic.

6.3.2 Heavy load tests

To find the upper limit of how many messages per second the HSS can handle, the number of messages were increased until the response times increased faster than the rate at which the load was increased.

In the following tests, the client runs five threads that send `User-Data` requests with a given interval.

In Figure 6.3 the interval for each of these five clients is 23ms, which at the HSS becomes 4.6ms between requests or about 217 messages/second. The Cx/Sh checks for data updates are now done every 5 seconds, this can be seen to impact the response times - as the peak delays have a periodicity of 5 seconds. This is because the database is busy processing these queries and is not processing the client requests. The figure shows that the client requests are delayed, but not lost as the delay for client requests decreases after these peaks.

As seen in the graph the standard deviation is 51.8 due to the Cx/Sh checks, average response time is 70.5ms. By using Little's Law $L = \lambda * W$ where $\lambda = 1/4.6$ms and $W = 70$ms the average number of users in the system L, is 15. This suggests that the servers message buffer is somewhere around 15 messages.

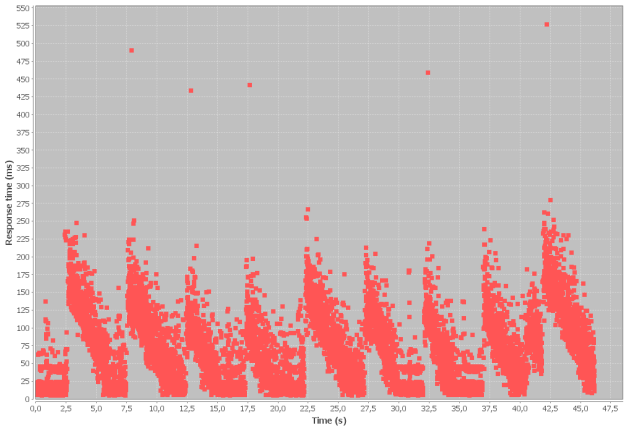

Figure 6.3. Response times of `User-Data` requests at 4.6ms intervals over 60 seconds.

When the load is increased to 227 messages/second, as in Figure 6.4, the response times does not decrease after the interrupts from the Cx/Sh activity and it looks more like a denial of service attack.

The critical point seems to be when the interarrival time is less than 4.6ms (i.e., more than 217 messages/second).

6.3.3 Heavy load test with database in memory

By putting the database files in a RAM-disk partition instead of on a disk, it is possible to see if disk access limits the performance of the database - that in turn is capping the HSS performance or if it is the HSS itself that is the bottleneck.

The same test as in Figure 6.3 is shown in Figure 6.5. The recovery after a Cx/Sh check is faster. As a result, if the interarrival time is decreased from 4.6ms to 4.2ms (i.e., increasing the message arrival rate from 217 to 238 messages/second) a similar graph as with the disk based database and interarrival time of 4.6ms is

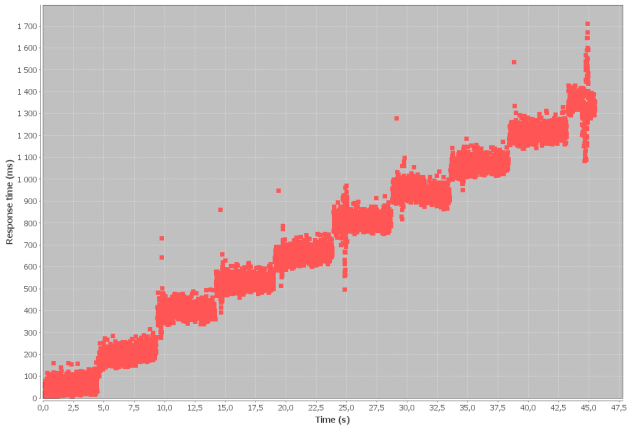

Figure 6.4. Response times of `User-Data` requests at 4.4ms intervals over 60 seconds.

presented.

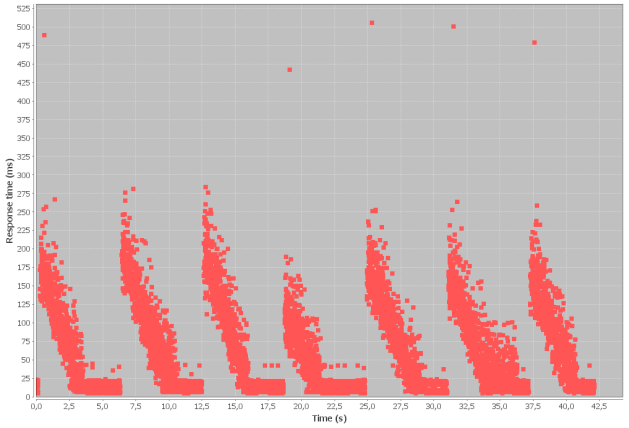

Figure 6.5. Response times of `User-Data` requests at 4.2ms intervals over 60 seconds with database in memory.

It looks as the memory resident database makes it possible to increase the interarrival time by 0.4ms (corresponding to an increased load of 11 requests per second) to get the same behavior as without the database in memory. The HSS shows the same behavior in Figure 6.4 using a disk based database as in Figure 6.6 were the database is stored in memory.

6.3.4 Tests with periodic Sh/Cx checks set to 30 min

The periodic Cx/Sh checks can be configured in the HSS settings. The default interval of 10 seconds was set to 1800 seconds (30 minutes) and a test with 4.4ms intervals (equal to the test in Figure 6.4) was done. The resulting graph is shown in

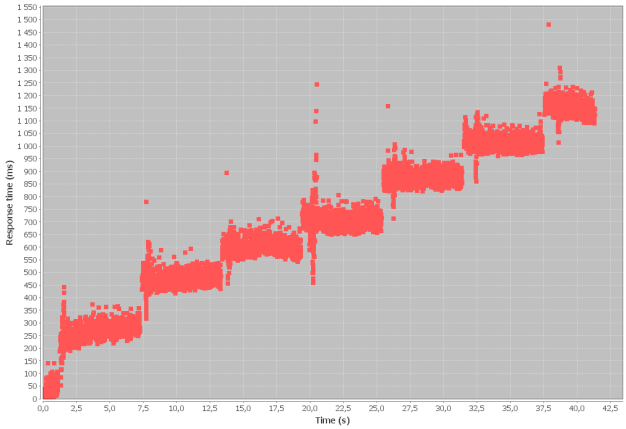

Figure 6.6. Response times of `User-Data` requests at 4.0ms intervals over 60 seconds with database in memory

Figure 6.7. The HSS performance has increased compared to the previous test but still has interruptions. These interruptions are not related to the Cx/Sh checks.

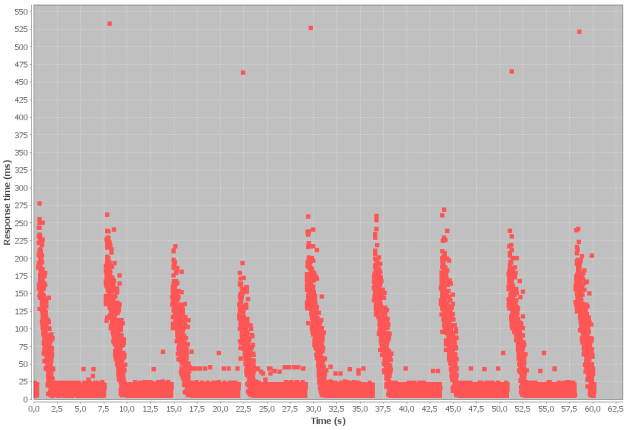

Figure 6.7. Response times of `User-Data` requests at 4.4ms intervals over 60 seconds with Cx/Sh checks set to 30 minutes

As in the previous tests, with the database in memory, the performance is increased. The performance increase makes the HSS able handle interarrival times that are 0.4ms lower (or 11 messages/second more) than with the Cx/Sh checks every 10 seconds. Figure 6.8 shows the response times the test with an interarrival time of 4.0ms and the Cx/Sh tests set to every 30 minutes.

6.4 Tests summary

The tests with a memory database and those with Cx/Sh checks disable shows that the HSS performance is depending on the database performance. This is

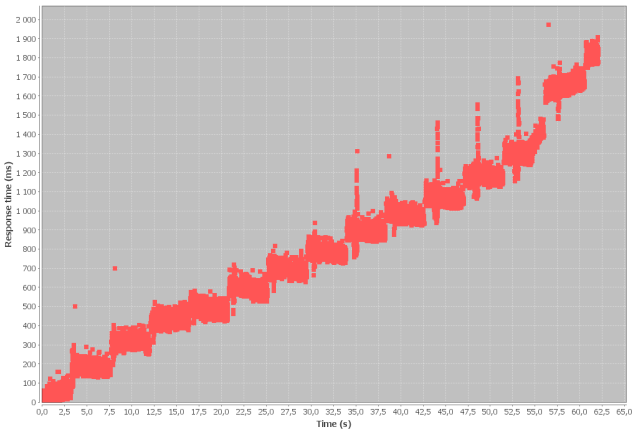

Figure 6.8. Response times of `User-Data` requests at 4.0ms intervals over 60 seconds with Cx/Sh checks set to 30 minutes

most likely not because the database performance was bad, its more likely that the unoptimized database queries made by the HSS made it seem like the database was a bottleneck. Doing a read operation in a transaction - that is later rolled back - is raising questions. If a Cx/Sh check that returns no new events interrupt processing of messages that much, how much processing time will a check that need to send subscription messages to other nodes take?

Since the HSS is written with the purpose to provide an IMS core reference implementation for IMS testing and application prototyping, it is not for aimed at commercial use and not optimized. For a commercial version of this IMS core node another programming language might be better suited to handle the task more efficiently. Using Java for this task might work as Java can be and is used for large complex systems but Ericsson AB [8] has for example chosen C++ for their HSS implementation. A programming language that is aimed at big telecom applications, like the HSS where availability and concurrency is important, is Erlang [9]. Erlang outperforms other programming languages when it comes to concurrent programming (as shown by [22]). It would be interesting to see how an Erlang implementation of the HSS would perform.

Chapter 7

Conclusions

This chapter contains the results of the service creation and load tests.

7.1 Service creation

IMS aims to make it easier and cheaper to create and deploy new services. The IMS architecture offers a platform that handles communication and message routing when creating a new service, it also offers access to subscriber data, and presents ways to make a service interact with and take part in a subscriber's session in a standardized way. This makes the IMS better than older cellular networks with respect to creating and deploying new services and bring more Internet-like services to mobile phones. Using IMS it is no more difficult to create new services than to create Internet services. It is possible to cache or store subscriber data in the AS and to subscribe to updates in order to lower the amount of traffic to the HSS. When using subscriptions, the HSS must of course use processing time to create and send subscription notifications, but if the amount of updates are lower than the service requests, the impact on service rate will be lower.

7.1.1 Comparing with an internet based alternative

If this service were implemented as a webservice instead of an IMS service there are some parts that now are handled by IMS but isn't available to a webservice and needs to be added, there are also some parts that are no longer needed.

In IMS the client terminal must be equipped with a SIP client capable of creating text messages in order to use the service. The webservice does not need to handle any SIP or Diameter messages at all but will instead need a HTML based webform where the user can enter his search terms and authenticate himself against a user database to be allowed access. In IMS authentication and authorization is handled by the IMS framework, while a webservice might have to use some decentralized standard for authenticating like OpenID [19] or implement a separate solution. The webservice can query the "Oracle of Bacon" over HTTP in a similar way to the IMS

service and then present the result to the user. Another aspect is that IMS has an accounting framework used to collect information for billing subscribers using the network and its services, such functionality is not as easily available for a webservice.

A simple performance comparison in terms of complexity between the IMS service and the webservice shows that the webservice, that is a the same very simple service, does not depend on any other nodes or network functions to work (excluding common internet functionality that both services use). The IMS version relies on both SIP and Diameter as well as the nodes in the IMS core, and that does add a overhead when it comes to performance. While the webservice sends four[1] HTTP messages, the IMS service sends twenty messages in total (see Figure 5.2).

7.2 HSS load tests

In older cellular networks the subscriber database node that is responsible for storing user subscription data has always been a protected database and access has only been granted to trusted nodes in the operator's own network. The IMS counterpart, the HSS, will be open not only to trusted CSCF nodes, but also to application servers and most importantly, to trusted third party application servers. The HSS has high availability demands. Both the load on and the number of connections to the HSS will grow as more nodes have access to it. If the HSS cannot support this extra load, then other parts of the network will have service disruptions when communicating with the HSS or these other nodes may be negatively impacted by increased load on the HSS. This might be solved by distributing the HSS over several nodes, by using solutions presented in the IMS specifications for load balancing different functions [6].

With a service like the one in this thesis the load on the HSS is unnecessarily large. By caching data for subscribers using the service and subscribing to updates for that data, the service can probable reduce its load on the HSS (depending on the amount of service request compared to the amount of subscriber data updates).

7.2.1 The FoHSS

The FoHSS is, as stated in the description [18], was not built for performance, but for conformance. This makes it a poor choice for a commercial IMS system where it would soon become a bottleneck. For a smaller system, mainly used for testing and educational purposes, it is very good. If the number of subscribers for a service reaches the limits of the HSS's capabilities of handling Diameter requests even with a light load the service can perhaps be optimized in some way to lighten the load on the HSS (such as the caching suggested in section 5.8). Or perhaps the HSS can be distributed to handle the traffic better. High availability and high performance might be provided by Open SAF (see § 2.6 of [38]).

[1] These four HTTP messages are, one from the client to the webservice, two back and forth between webservice and "Oracle of Bacon" and then one back to the client.

During testing of the FoHSS one thing stuck out as strange. The periodic access to the database that checks for updated subscriber data has a notable impact on performance. Why are these checks done using polling and not using some notification from the procedure that does the update to the procedure sending the update notifications? For example, the updates could be collected in a list (similar to a transaction log) and used to directly generate the notifications of updates.

7.3 Future work

In this thesis the event checking for the Cx and Sh interface impacts on HSS performance, a deeper investigation on the impact of subscription message handling would be of interest.

An investigation of the statistical distribution for the traffic to the HSS could be made. Something similar has been done to user generated SIP traffic in an IMS network [33]. With the results of such an investigation, better and more realistic test programs could be created, helping developers optimize their HSS implementation with something that simulates real Diameter traffic.

Bibliography

[1] 3GPP. 3G security; Network Domain Security (NDS); IP network layer security. TS 33.210, 3rd Generation Partnership Project (3GPP), September 2007. URL http://www.3gpp.org/ftp/Specs/html-info/33210.htm.

[2] 3GPP. Cx and Dx interfaces based on the Diameter protocol; Protocol details. TS 29.229, 3rd Generation Partnership Project (3GPP), December 2007. URL http://www.3gpp.org/ftp/Specs/html-info/29229.htm.

[3] 3GPP. IP Multimedia (IM) session handling; IM call model; Stage 2. TS 23.218, 3rd Generation Partnership Project (3GPP), December 2007. URL http://www.3gpp.org/ftp/Specs/html-info/23218.htm.

[4] 3GPP. IP Multimedia Subsystem (IMS) Sh interface; Signalling flows and message contents. TS 29.328, 3rd Generation Partnership Project (3GPP), December 2007. URL http://www.3gpp.org/ftp/Specs/html-info/29328.htm.

[5] 3GPP. IP Multimedia Subsystem (IMS); Stage 2. TS 23.228, 3rd Generation Parnership Proejct (3GPP), December 2007. URL http://www.3gpp.org/ftp/Specs/html-info/23228.htm.

[6] 3GPP. Network architecture. TS 23.002, 3rd Generation Partnership Project (3GPP), December 2007. URL http://www.3gpp.org/ftp/Specs/html-info/23002.htm.

[7] 3GPP. Sh interface based on the Diameter protocol; Protocol details. TS 29.329, 3rd Generation Partnership Project (3GPP), December 2007. URL http://www.3gpp.org/ftp/Specs/html-info/29329.htm.

[8] Ericsson AB. URL http://www.ericsson.com/.

[9] Ericsson AB. URL http://www.erlang.org/.

[10] B. Aboba and M. Beadles. The Network Access Identifier. RFC 2486, Internet Engineering Task Force, January 1999. URL http://www.rfc-editor.org/rfc/rfc2486.txt.

[11] SIP Servlet API, mar 2003. URL http://jcp.org/en/jsr/detail?id=116.

[12] P. Calhoun, T. Johansson, C. Perkins, T. Hiller, and P. McCann. Diameter Mobile IPv4 Application. RFC 4004, Internet Engineering Task Force, August 2005. URL http://www.rfc-editor.org/rfc/rfc4004.txt.

[13] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol. RFC 3588, Internet Engineering Task Force, September 2003. URL http://www.rfc-editor.org/rfc/rfc3588.txt.

[14] P. Calhoun, G. Zorn, D. Spence, and D. Mitton. Diameter Network Access Server Application. RFC 4005, Internet Engineering Task Force, August 2005. URL http://www.rfc-editor.org/rfc/rfc4005.txt.

[15] Open IMS Core. internet, 2008. URL http://www.openimscore.org/.

[16] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, Internet Engineering Task Force, April 2006. URL http://www.rfc-editor.org/rfc/rfc4346.txt.

[17] The Fraunhofer Institute for Open Communication System, June 2008. URL http://www.fokus.fraunhofer.de/.

[18] Fraunhofer Institute for Open Communication Systems. URL http://www.fokus.fraunhofer.de/en/fokus_testbeds/open_ims_playground/components/osims/fhoss.

[19] OpenID Foundation. URL http://openid.net.

[20] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617, Internet Engineering Task Force, June 1999. URL http://www.rfc-editor.org/rfc/rfc2617.txt.

[21] M. Garcia-Martin, M. Belinchon, M. Pallares-Lopez, C. Canales-Valenzuela, and K. Tammi. Diameter Session Initiation Protocol (SIP) Application. RFC 4740, Internet Engineering Task Force, November 2006. URL http://www.rfc-editor.org/rfc/rfc4740.txt.

[22] Ali Ghodsi and Joe Armstrong. Apache vs. yaws. URL http://www.sics.se/~joe/apachevsyaws.html.

[23] Miguel A. Garcia-Martin Gonzalo Camarillo. *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*. John Wiley & Son, second edition, December 2005. ISBN 0470018186.

[24] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889, Internet Engineering Task Force, January 1996. URL http://www.rfc-editor.org/rfc/rfc1889.txt.

[25] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. RFC 2608, Internet Engineering Task Force, June 1999. URL http://www.rfc-editor.org/rfc/rfc2608.txt.

[26] Heino Hameleers and Christer Johansson. IP Technology in WCDMA/GSM core networks, January 2002. URL http://www.ericsson.com/ericsson/corpinfo/publications/review/2002_01/files/2002012.pdf. The PDF file is date 19 April 2002.

[27] H. Hannu, J. Christoffersson, S. Forsgren, K.-C. Leung, Z. Liu, and R. Price. Signaling Compression (SigComp) - Extended Operations. RFC 3321, Internet Engineering Task Force, January 2003. URL http://www.rfc-editor.org/rfc/rfc3321.txt.

[28] R. Housley. Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). RFC 4309, Internet Engineering Task Force, December 2005. URL http://www.rfc-editor.org/rfc/rfc4309.txt.

[29] Internet Assigned Numbers Authority (IANA). internet, 2008. URL http://www.iana.org.

[30] JavaDiameterPeer. internet, 2008. URL http://www.openimscore.org/docs/JavaDiameterPeer/index.html.

[31] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, Internet Engineering Task Force, November 1998. URL http://www.rfc-editor.org/rfc/rfc2401.txt.

[32] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, Internet Engineering Task Force, December 2005. URL http://www.rfc-editor.org/rfc/rfc4301.txt.

[33] Ivan I. Kuzmin and Olga A. Simonina. Signaling flows distribution modeling in the ims. In *EUROCON 2009, EUROCON '09. IEEE*, pages 1866–1869, May 2009.

[34] Mahsa Nakhjiri Madjid Nakhjiri. *AAA and Network Security for Mobile Access*. John Wiley & Son, December 2006. ISBN 0470011947.

[35] M. Mealling. Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database. RFC 3403, Internet Engineering Task Force, October 2002. URL http://www.rfc-editor.org/rfc/rfc3403.txt.

[36] M. Mealling and R. Daniel. The Naming Authority Pointer (NAPTR) DNS Resource Record. RFC 2915, Internet Engineering Task Force, September 2000. URL http://www.rfc-editor.org/rfc/rfc2915.txt.

[37] MySQL. URL http://www.mysql.com/.

[38] Dan Peterström. Ip multimedia for municipalities: The supporting architecture. Master's thesis, Royal Institute of Technology (KTH), School of Information and Communication Technology, aug 2009. URL http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/090818-Dan_Peterstrom-with-cover.pdf.

[39] R. Price, C. Bormann, J. Christoffersson, H. Hannu, Z. Liu, and J. Rosenberg. Signaling Compression (SigComp). RFC 3320, Internet Engineering Task Force, January 2003. URL http://www.rfc-editor.org/rfc/rfc3320.txt.

[40] A. B. Roach. Session Initiation Protocol (SIP)-Specific Event Notification. RFC 3265, Internet Engineering Task Force, June 2002. URL http://www.rfc-editor.org/rfc/rfc3265.txt.

[41] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002. URL http://www.rfc-editor.org/rfc/rfc3261.txt.

[42] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, Internet Engineering Task Force, July 2003. URL http://www.rfc-editor.org/rfc/rfc3550.txt.

[43] Sun GlassFish Enterprise Server, June 2008. URL https://glassfish.dev.java.net/.

[44] Inc Sun Microsystems. URL http://www.oracle.com/sun. Sun was aquired by Oracle, January 27, 2010.

[45] Wikipedia. internet, 2008. URL http://en.wikipedia.org/wiki/Servlet.

[46] Wireshark. internet, 2008. URL http://www.wireshark.org.

TRITA-ICT-EX-2010:29

www.kth.se