

D4.1.4



Consolidated Network Service Provisioning Concept

Version 1.0

WP 4.1 Mobile Network Architecture,
Design & Implementation

Dissemination Level: Public

Lead Editor: Nuno Inácio, IT-Aveiro

24/08/2007

Status: Approved by QM

SIXTH FRAMEWORK PROGRAMME
PRIORITY IST-2002-2.3.1.18



Information Society

Grid for complex problem solving
Proposal/Contract no.: 004293

This is a public deliverable that is provided to the community under the license Attribution-NoDerivs 2.5 defined by creative commons <http://www.creativecommons.org>

This license allows you to

- to copy, distribute, display, and perform the work
- to make commercial use of the work

Under the following conditions:



Attribution. You must attribute the work by indicating that this work originated from the IST-Akogrino project and has been partially funded by the European Commission under contract number IST-2002-004293



No Derivative Works. You may not alter, transform, or build upon this work without explicit permission of the consortium

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

This is a human-readable summary of the Legal Code below:

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

- "**Collective Work**" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
- "**Derivative Work**" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
- "**Licensor**" means all partners of the Akogrino consortium that have participated in the production of this text
- "**Original Author**" means the individual or entity who created the Work.
- "**Work**" means the copyrightable work of authorship offered under the terms of this License.
- "**You**" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

- to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
- to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works.
- For the avoidance of doubt, where the work is a musical composition:
 - Performance Royalties Under Blanket Licenses.** Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.
 - Mechanical Rights and Statutory Royalties.** Licensor waives the exclusive right to collect, whether individually or via a music rights society or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).

- d. **Webcasting Rights and Statutory Royalties.** For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Derivative Works. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

- a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any credit as required by clause 4(b), as requested.
- b. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or Collective Works, You must keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; the title of the Work if supplied; and to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

5. Representations, Warranties and Disclaimer. UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE MATERIALS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

- a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
- b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

- a. Each time You distribute or publicly digitally perform the Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
- b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
- c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
- d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Context

Activity 4	Detailed Architecture, Design & Implementation
WP 4.1	Mobile Network Architecture, Design & Implementation
Dependencies	This deliverable depends on and has influence on (to some extent) work carried out in WP 3.1, 3.2, 4.2, 4.3, 4.4, 5.1, 5.2

Approved by: QM**Contributors:**

Nuno Inácio (IT-Aveiro)
 Vicente Olmedo (UPM)
 Juan Enrique Burgos (TID)
 Noelia de la Peña Alonso (TID)
 Bienvenido Gómez (TID)
 Antonio Cuevas Casado (UPM)
 Patrick Mandic (USTUTT)

Reviewers

Internal review by WP4.1 Participants.
 Review by Akogrimo partners external to WP 4.1:
 Cristian Morariu (UniZh)
 Fredrik Solsvik (Telenor)
 David Lutz (UStutt)
 Jesus Movilla (TID)
 Peter Racz (UniZh)

Version	Date	Authors	Sections Affected
0.1	19.06.07	All	Document created based on D4.1.3.
0.2	27.07.07	Juan Burgos, Patrick Mandic, Nuno Inácio	Update of chapter 5 and References. Rewriting/updating of parts of chapter 7.
0.3	06.08.07	All	First update of document regarding comments produced by the external review.
0.4	19.08.07	All	Further update of document with regards to the comments by external reviewers.
0.5	22.08.07	Juan Burgos, Nuno Inácio	Update of SIP sections in chapter 7. Preparation for final delivery.
1.0	24.08.07	Nuno Inácio	Final update of the document taking into account comments from QM.

Executive summary

This document presents the evolution of the Network Layer Architecture described in D4.1.1. That document depicted the fundamentals and architecture of this particular view of Next Generation Networks (NGN), in which collaboration and integration with Grid infrastructures is considered a must.

After laying the basis for this architecture and corroborating the design by means of a first implementation while taking into account the feedback of the commission on the previous work, a redesign to polish some aspects of the architecture was undertaken and summarized in D4.1.3, which focused on the changes performed in the architecture and paying special attention to the most innovative issues such the interaction between Grids and Networks – Grid using Network QoS, SIP with SOAP, etc. In addition to these topics, security was also strongly prioritized, focusing on the following fields:

- Security aligned to the research done in the project i.e. security issues arisen within the context of the new functionalities developed (e.g. SIP with SOAP interactivity).
- Security as a whole, in order to provide guidelines for the security of the complete platform developed. (e.g. provide authentication and Single Sign On (SSO)).
- Security for the tools used for implementation - e.g. Java stubs. (This one given the least importance in favour of the other two)

This document is an evolution of D4.1.3, with an improved security section, general updates to some of the components reflecting changes made to the Akogrimo architecture either due to feedback from the various demonstrations performed or to new requirements from the Disaster Handling and Crisis Management scenario. Throughout the entire document, additional minor changes were performed to clarify some issues that were considered to be not properly explained.

Table of Contents

Executive summary.....	5
Table of Contents.....	6
List of Figures.....	9
List of Tables.....	11
Abbreviations.....	12
1. Introduction.....	16
2. Architecture.....	17
3. Mobility.....	18
3.1. Introduction.....	18
3.2. User Mobility.....	18
3.3. Session Mobility.....	18
3.3.1. Architecture.....	18
3.3.2. Interaction with other subsystems.....	20
3.3.3. Signalling.....	21
3.4. Terminal Mobility.....	23
3.4.1. Architecture.....	23
3.4.2. Interaction with other subsystems.....	24
3.4.3. Signalling.....	24
3.4.4. Security.....	26
4. Quality of Service.....	27
4.1. Introduction.....	27
4.2. QoS Architecture.....	27
4.2.1. Overview.....	27
4.2.2. Components involved in QoS subsystem.....	28
4.2.3. QoS Bundles.....	28
4.3. QoS subsystems and interactions.....	29
4.3.1. QoS Broker.....	29
4.3.2. Access Router.....	30
4.3.3. Mobile Terminal.....	32
4.3.4. Policies/PBNM interactions.....	32
4.3.5. User profiles/A4C interactions.....	33
4.4. Signalling.....	33
4.4.1. QoS Broker <-> Access Router.....	33
4.4.2. Access Router <-> Mobile Terminal.....	33
4.5. QoS and EMS interoperation.....	36

4.5.1.	Architecture	36
4.5.2.	EMS Interface.....	36
4.5.3.	Monitoring Interface	37
4.6.	Security.....	37
5.	The Akogrimo SIP framework.....	38
5.1.	Introduction	38
5.2.	SIP Architecture	38
5.2.1.	Mobile Terminal: SIP Module.....	38
5.2.2.	Network components.....	40
5.3.	SIP-SOAP synergies	42
5.3.1.	Services SIPification framework.....	43
5.4.	Implementation	46
5.4.1.	MT SIP Module.....	47
5.4.2.	SIP Server.....	51
5.4.3.	SIP Broker.....	53
6.	PBNM.....	55
6.1.	Introduction.....	55
6.2.	Architecture.....	55
6.3.	Sample scenarios	56
6.4.	Documents, standards, protocols & interfaces.....	57
6.4.1.	Example: mapping policies to CIM.....	58
6.5.	Implementation	58
7.	Security.....	62
7.1.	Introduction.....	62
7.2.	Attack model.....	62
7.3.	Network layer attack scenarios.....	63
7.3.1.	Mobile IPv6	64
7.3.2.	Attacks to Access Router.....	64
7.3.3.	QoS Broker	66
7.3.4.	PBNM.....	66
7.3.5.	Attacks to SIP signalling	66
7.4.	Defences.....	71
7.4.1.	Assumptions	71
7.4.2.	Perimeter security.....	72
7.4.3.	Mobile IPv6	74
7.4.4.	PBNM Defence.....	74

7.4.5.	SIP Defence	74
7.5.	Security Architecture Overview	77
	References	80
Annex A.	Access security at L3 example	82
Annex B.	SER (SIP Express Router) configuration file (ser.cfg).....	84
Annex C.	Network User Profile.....	88
Annex D.	gSDP formats	90
D.1.	gSDP session.....	90
D.2.	gSDP notification.....	90
Annex E.	SIP security mechanisms	92
E.1.	Transport and Network Layer Security	92
E.1.1.	IPSec.....	92
E.1.2.	TLS	93
E.2.	SIP URI Scheme.....	93
E.3.	HTTP Digest	94
E.4.	S/MIME.....	95

List of Figures

Figure 1 – General architecture.....	17
Figure 2 – User-triggered Session Transfer.....	19
Figure 3 – Grid-triggered Session Transfer.....	20
Figure 4 – SIP Session Mobility (session handoff mode). User triggered	22
Figure 5 – SIP Session Mobility (session handoff mode). Grid triggered	23
Figure 6 – Fast Handover	25
Figure 7 – QoS subsystem components	27
Figure 8 – Akogrimo QoS Broker	29
Figure 9 – Akogrimo Access Router	31
Figure 10 – Implicit QoS request.....	34
Figure 11 – explicit QoS request using RSVP.....	35
Figure 12 – QoSB Grid Gateway.....	36
Figure 13 – D.413 MT SIP module architecture	39
Figure 14 – Revised MT SIP module architecture	40
Figure 15 – SIP Server architecture	41
Figure 16 – AN SIP proxy architecture	41
Figure 17 – SIP Broker architecture	42
Figure 18 – Intermediaries and interfaces setup	44
Figure 19 – SIPification session establishment and service invocation	44
Figure 20 – User disconnection and reconnection from another location.....	45
Figure 21 – Change on services information	46
Figure 22 – gSDP session termination (from EMS)	46
Figure 23 – Akogrimo MT SIP application. Main function structure.....	50
Figure 24 – PBM Architecture. Relation with other components	55
Figure 25 – A declaration of a QoS policy in CIM	57
Figure 26 – Representation of a CIM class in XML	58
Figure 27 – Representation of a CIM class in MOF.....	58
Figure 28 – Communication between the PBNM Server and a PDP	59
Figure 29 – CIM Remote Providers	59
Figure 30 – GUI for the CIMOM (PBNM Server).....	60
Figure 31 – QoSBroker Policy database	60
Figure 32 – “SIP with SOAP” possible attacks	68
Figure 33 – Network Authentication	73
Figure 34 – General architecture.....	78
Figure 35 – HTTP Digest security mechanism.....	94

Figure 36 – S/MIME encryption and decryption96
Figure 37 – S/MIME signature.....97

List of Tables

Table 1 – QoS bundles and usage. Values are in kilobytes per second.....	28
Table 2 – QoS-EMS interface methods.....	37
Table 3 – MT SIP module implementation: libraries and packages	47
Table 4 – MT SIP module implementation: available interfaces.....	48
Table 5 – MT SIP module implementation: location of the utility classes	49
Table 6 – gSDP application 1 st prototype vs. final version	50
Table 7 – MT gSDP application implementation: libraries and packages	50
Table 8 – MT gSDP application implementation: real services interface	51
Table 9 – a4c.so exported functions.....	52
Table 10 – SIP Broker implementation: libraries and packages	53
Table 11 – Illustration of a process of mapping a policy from system level to instance level.....	58
Table 12 – Attack model.....	62
Table 13 – OpVO member’s token content	76
Table 14 – Encrypted headers with S/MIME	95

Abbreviations

A4C	Authentication, Authorisation, Accounting, Auditing and Charging
AA	Authentication and Authorisation
AAA	Authentication, Authorisation and Accounting
Akogrimo	Access To Knowledge through the Grid in a Mobile World
AN	Access Network
AoR	Address of Record
API	Application Programming Interface
AR	Access Router
BA	Binding Acknowledgement
BE	Best Effort
BU	Binding Update
CIM	Common Information Model
CIMOM	CIM Object Manager
CM	Context Manager
COPS	Common Open Policy Service Protocol
CPU	Central Processing Unit
DiffServ	Differentiated Services
DoS	Denial of Service
DDoS	Distributed Denial of Service
DMTF	Distributed Management Task Force
DSCP	Differentiated Service Code Point
EMS	Execution Management Service
EPR	End Point Reference
FHO	Fast Handover
FBU	Fast Binding Update
FNA	Fast Neighbour Advertisement

gSDP	Grid Session Description Protocol
GUI	Graphical User Interface
GUP	Generic User Profile
HA	Home Agent
HoA	Home Address
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IntServ	Integrated Services
IP	Internet Protocol
IPSec	IP Security Protocol
IPv6	Internet Protocol version 6
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MIPv6	Mobile IPv6
MN	Mobile Node
MOF	Managed Object Format
MT	Mobile Terminal
MTIHO	Mobile Terminal Initiated Handover
nAR	New Access Router
NIHO	Network Initiated Handover
oAR	Old Access Router
OpVO	Operational Virtual Organization
PANA	Protocol for carrying Authentication for Network Access
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PBNM	Policy Based Network Management
PBNMS	Policy Based Network Management System

PDA	Personal Digital Assistant
PGP	Pretty Good Privacy
QoS	Quality of Service
QoSB	Quality of Service Broker
QoSGGw	Quality of Service Grid Gateway
RSVP	Resource-Reservation Protocol
RTP	Real-Time Protocol
RTSP	Real Time Streaming Protocol
SDP	Service Discovery Protocol
SER	SIP Express Router
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
SSO	Single Sign On
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA	User Agent
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VO	Virtual Organization
WBEM	Web Based Enterprise Management
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network

WP	Work Package
WPA	Wi-Fi Protected Access
WS	Web Service
XML	Extensible Markup Language

1. Introduction

As already described in previous documents, the Network layer provides means to keep the transport of information up and optimized at all times always taking into account the context and the nature of the communication, the parties involved and external factors such as saturation or breakdown.

This document is an updated version of D4.1.3, which provided a refined architecture of the different components and services provided by the Network.

Chapters 3 and 4, Mobility and QoS respectively, have had minor updates to clarify some situations, while chapter 5 has had some more changes due to updated functionality.

PBNM, described in chapter 6, has suffered only minor updates from D4.1.3.

Chapter 7 makes a survey of possible attacks and their importance and describes the most appropriate solutions according to impact, probability and interest to the project's goals. It has been significantly improved taking into account input from the last project review.

2. Architecture

The following picture describes the general architecture inside WP4.1 (inside the blue box) and its relations to components in different WPs, the arrows among them indicating the existent interfaces. The protocol utilized for each interface to communicate is also indicated.

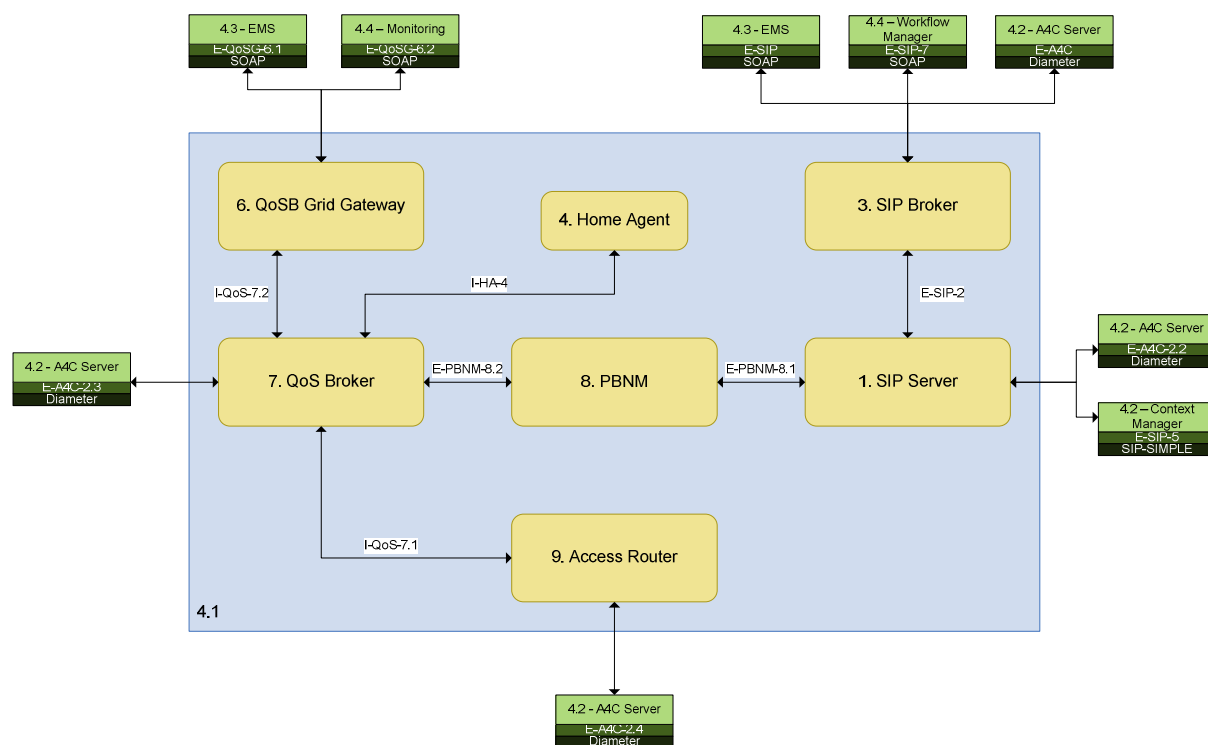


Figure 1 – General architecture

Regarding network quality of service, the main components involved are the Access Router (AR), QoS Broker (QoSB) and PBNM. The QoSB is the equivalent to a Policy Decision Point (PDP), which receives general network policies from the PBNM, and also user profile information from the A4C Server.

The AR is the equivalent to a Policy Enforcement Point (PEP). It controls all network flows that originate or are destined to mobile terminals connected to that AR. When a new flow is detected, the AR sends a message to the QoSB questioning if that flow is allowed or not. The QoSB, based upon the information it gathered from the PBNM and A4C Server, makes a decision and responds to the AR.

The SIP Server provides advanced support for multimedia sessions, session mobility, user presence and context and also support for interactions with web services. Web services may use the SIP Server to enable grid sessions with session management and mobility, session transfer or 3rd party grid session control.

The QoSB Grid Gateway and the SIP Broker are Web Services interfaces to the QoS Broker and the SIP Server, respectively.

Security issues have also been taken into account in the architecture. They have however not been represented in the description of the architecture at module level since security needs to be handled in much broader sense. Therefore, this issue is treated separately in chapter 7.

3. Mobility

3.1. Introduction

Different types of mobility may be considered; in this chapter we present the three most important in the Akogrimo project.

3.2. User Mobility

As defined in [D411], User Mobility enables the user access to network services or resources independently of the user's terminal. A user-oriented security and authentication framework (described in WP42 deliverables) is responsible for this. For more details see [D424].

3.3. Session Mobility

Session Mobility is the ability, supported by the underlying infrastructure, to transfer an ongoing session from the device the user is currently using (the source Mobile Terminal) to another device (the target Mobile Terminal). Of course, this must be done without the session being interrupted.

Such ability could be used to continue a session in a more suitable device when it is discovered, or when the source device becomes unavailable, e.g. running out of battery. In Akogrimo's eHealth scenario, session mobility is used to transfer an already established media session to a larger display, which is more suitable for media visualisation than the small, low resolution screen of a wearable computer.

Session transfers are usually initiated by the user when needed, e.g. need to switch from a fixed to a mobile terminal (*user triggered session transfer*). However, many of the times the need of a session transfer can be anticipated by analysing user context. Taking this into account, and taking advantage of Akogrimo's infrastructure support for context, session transfers can be initiated on behalf to the user (*Grid triggered session transfer*), suggesting more suitable target MT's in the user surroundings. This functionality is demonstrated in the eHealth scenario, where a doctor is warned about the availability of a better display and the session transfer is initiated by the platform. This operation, of course, requires new entities to be defined in the platform to perform the needed behaviour, as well as to offer these services as suitable interfaces to upper layers.

In Akogrimo, session mobility support will be provided by the SIP infrastructures. In the following subsections, the architecture will be presented and the involved components described, as well as their interactions with other subsystems in the platform. Finally, some MSCs are analyzed to clarify the different roles and to give an overview of the signalling involved.

3.3.1. Architecture

As D4.1.1 mentions [D411], there are two main methods to perform the session transfer using SIP:

- Mobile Node Control mode, based on SIP 3PCC (Third-Party Call Control).
- Session Handoff mode, based on SIP REFER.

After analysing the pros and cons of each method, it was decided the second one is more appropriated to the Akogrimo project. For example, it does not involve the terminal which

initiated the transfer after it takes place, so a MT could transfer an ongoing session to another device if its battery is running out. For a more detailed explanation, please refer to [D411].

As mentioned in the previous section, session mobility in Akogrimo can take place in two scenarios:

- A user decides to move an ongoing session from one terminal to another (user-triggered session transfer).
- Some workflow decides, based on user context and other parameters from the workflow execution, to inform a user about the convenience of transferring an ongoing session to another device (grid-triggered session transfer). This situation typically can take place if a context-change occurs after a grid-initiated SIP call (e.g. in an eHealth environment, there is a grid-initiated SIP call between a user and a doctor. The doctor is using some device not capable to visualise a medical application, but he/she is moving and now there is a reachable beamer near him/her).

The following drawing shows the user-triggered session transfer, in the way it was implemented for the prototype in the first project phase.

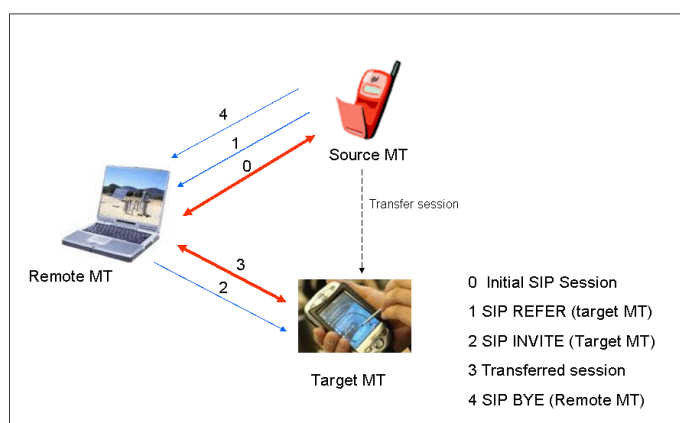


Figure 2 – User-triggered Session Transfer

In this scenario two users (Source MT and Remote MT) have established one SIP session, and one of them (Source MT) wishes to transfer his session to another device (Target MT). So, the Source MT sends a REFER SIP message to the Remote MT, including the Target MT SIP URI in this message. After the Remote MT receives this message, it can start a standard SIP session setup process with the Target MT, and the session will continue now from the Remote MT to the Target MT.

Basically the difference regarding to the scenario described at D4.1.1 (pg 67), is where the REFER message is sent. Sending it to the remote MT being involved in the communication (instead of the target MT, as described in D411) avoids the inclusion of the “Replaces” header field in the SIP request and simplifies the process. Section 3.3.3 describes the message sequence in detail.

The other scenario refers to the grid-initiated session transfer. In this context, there is a SIP session established between two users (Source MT and Remote MT), when the Workflow Manager (based on the user context information and some workflow execution parameters) decides to inform to the SIP architecture (SIP Broker) about the convenience to perform a session transfer, from the device being used (source MT, e.g a PDA) to a more suitable device (target MT, e.g. a beamer).

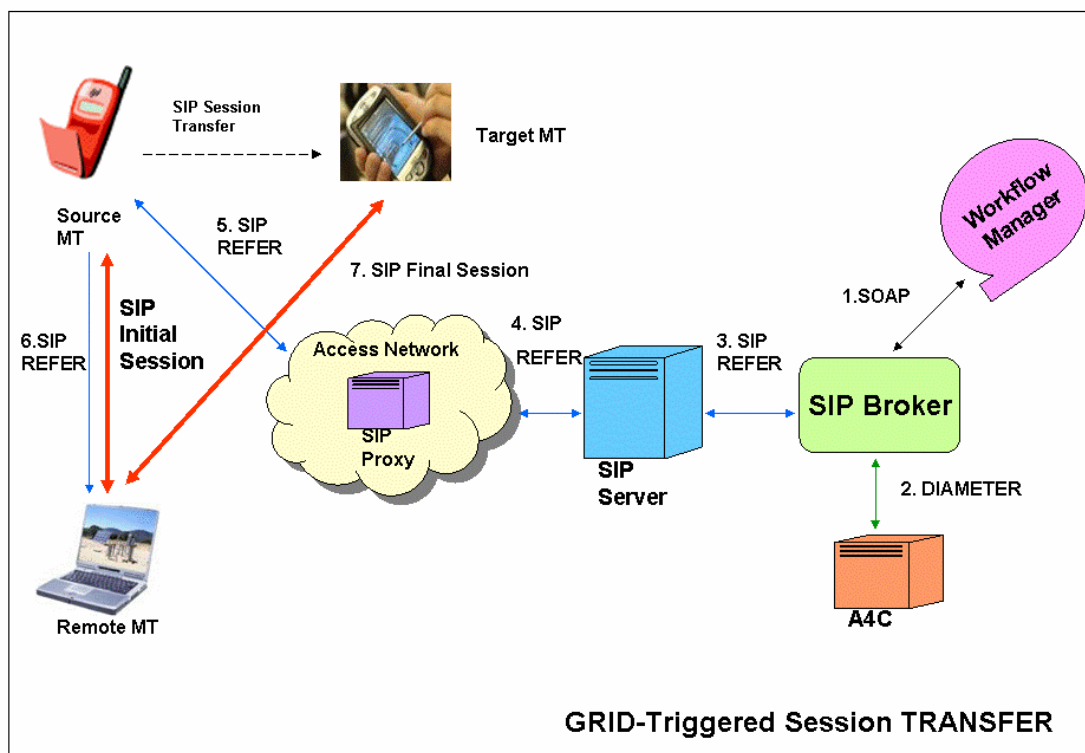


Figure 3 – Grid-triggered Session Transfer

As shows the figure above the components involved in this scenario are the following:

- **Workflow Manager:** this component is in charge of sending the SOAP request (1) to the SIP Broker, in order to indicate it that one session transfer should be started from Source MT to Target MT. The SOAP request (1) includes three parameters: the user identity of the participants of the ongoing session and the target MT's SIP AoR.
- **SIP Broker:** once it receives the SOAP request (1) from the Workflow Manager, it retrieves the SIP URIs of the participants from the A4C (i.e. concerning the source and remote MT's being used), then it builds the REFER SIP message and sends it to the SIP Server.
- **SIP Server:** it receives the REFER message from the SIP Broker and forwards it to the SIP Proxy which is supporting to the Source MT.
- **SIP Proxy:** it is placed in the Access Network, and is in charge of forwarding the REFER message to the Source MT.
- **MT:** it is the end user device. Once the Source MT receives the SIP REFER message in order to transfer its current session to another device (more suitable), it starts the Session Transfer process described before (Figure 2)

3.3.2. Interaction with other subsystems

The SIP module offers a Java interface, which enables the communication from/to the end user application running on top. Considering the point of view of the application running on top, this interface can be divided in two main parts:

- The public interface that the SIP Module component offers to the application in order to initiate different SIP requests.

- A Java interface that the applications have to implement in order to receive notifications from the SIP UA. These notifications can be a response to a process initiated by the application, an incoming request from another SIP entity, or intermediate notifications on the status of a transaction in progress.

So as part of the middleware offered to the MT SIP-aware applications, the SIP module offers a set of functions to initiate, accept, reject, and so on, a SIP session transfer, even if not initiated by another MT. In fact, the SIP Broker WS interface includes a method to notify a user on the availability of a more suitable device for an ongoing session. This method can be invoked from a grid entity (typically the Workflow Manager) to induce a session transfer.

3.3.3. Signalling

SIP signalling for session transfer in Akogrimo is based on the concepts and scenarios presented in [SHACH03]. The proposed mechanisms have been adapted to Akogrimo platform specific requirements. In the draft, two possibilities to perform session transfer are presented: Mobile Node Control Mode and Session Handoff (SH) Mode. The main difference between them is that the Mobile Node Control Mode allows the Source MT to retrieve a previously transferred session. Session Handoff Mode is not as flexible as Mobile Node Control Mode, but it simplifies signalling and implementation. SH Mode is the approach selected in Akogrimo, because its capabilities satisfy session transfer requirements for Akogrimo scenarios, so there is no real advantage in making signalling and implementation more complex.

In the Session Handoff Mode, the session is completely transferred to the Target MT. The Source MT is no longer related with the session being held. As stated before, this is how sessions are transferred in Akogrimo.

Figure 4 shows the signalling involved in a user-triggered session transfer. In the MSC, the Source MT has established a session with the Remote MT. Then, the user decides to transfer the session to a Target MT for any reason. Session transfer is started with a REFER message, which tells the Remote MT to transfer the session to the Target MT. The REFER message also creates a subscription relationship between the Remote MT and the Source MT. This subscription is used by the Remote MT to inform the Source MT about the progress of the transference by means of NOTIFY messages. The Remote MT then establishes a new session with the Target MT according to a standard SIP session setup. Once the new session is established, the Remote MT informs the Source MT, which sends a BYE message to the Remote MT to tear down the previous session.

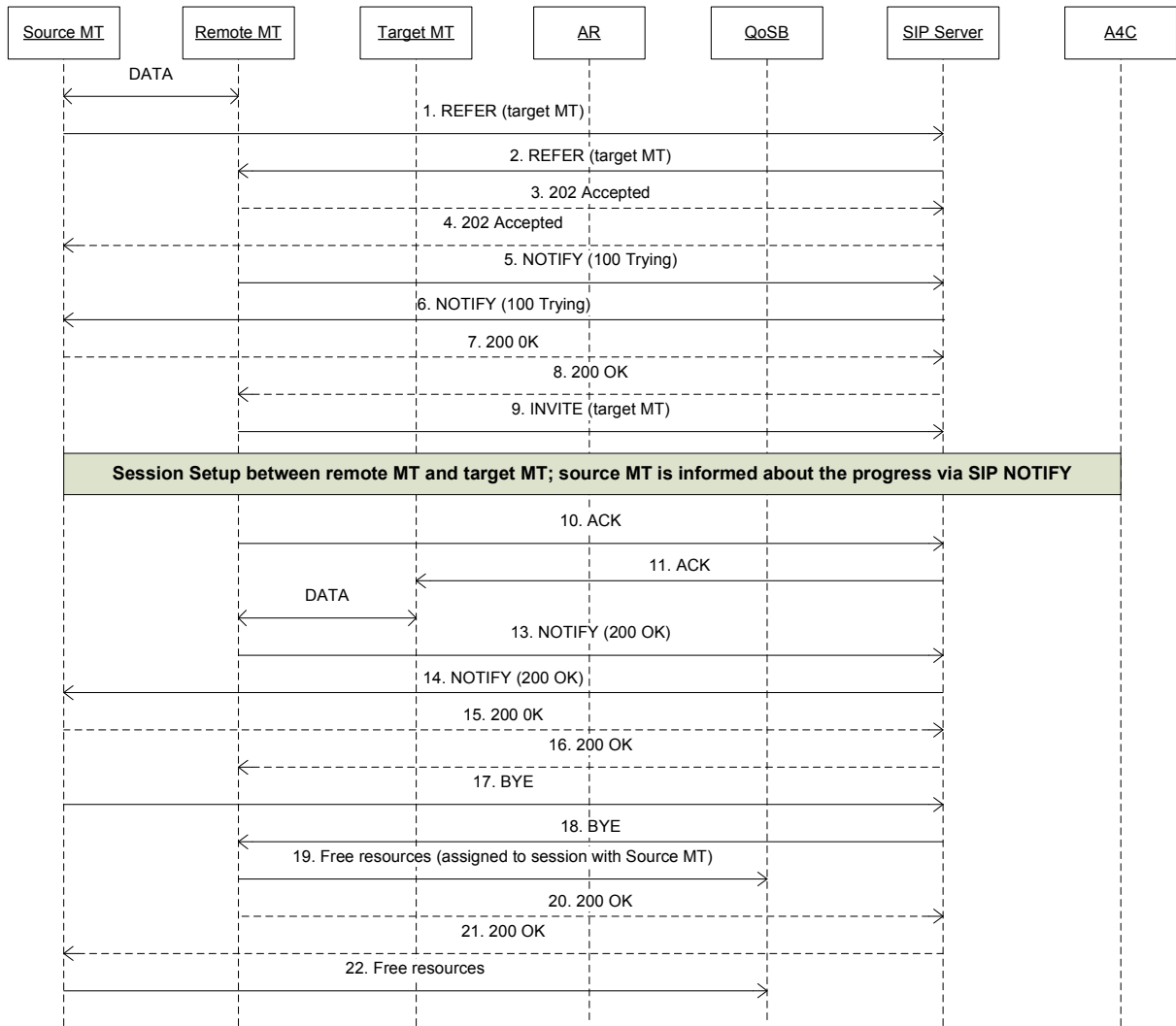


Figure 4 – SIP Session Mobility (session handoff mode). User triggered

Note that the signalling presented in the last figure is a bit different from the one proposed in [SHACH03], where the Source MT sends the REFER message to the Target MT instead of the Remote MT. The Target MT then contacts the Remote MT and establishes a session with it. As the Remote MT does not maintain an ongoing session with the Target MT, some mechanism to distinguish the session transfer from an independent new session is needed. This is achieved via the *Replaces* header, defined in [RFC3891]. This header is included by the Source MT in the first REFER and copied by the Target MT in the INVITE it sends to the Remote MT. Its presence in the INVITE message indicates a session transfer, and the header contains the needed data for the Remote MT to know which session is actually being replaced.

The signalling proposed in Figure 4 for Akogrimo simplifies the messages exchanging and UA behaviour, its drawback being that the Target MT does not know it is engaged in a session transfer, as it receives a standard INVITE from the Remote MT. Whether this is important or not depends on each application. In the Akogrimo’s eHealth scenario, session transfer is used for a multimedia videoconference application, and when the session is transferred the application running on the remote MT just continues sending audio and video coming from the microphone and the webcam, but to another device. Target MT does not need to be aware of a session being transferred. This issue should be taken into account for future Akogrimo applications using SIP. If these applications need remote party awareness of session transfer, more complex signalling should be implemented.

Figure 5 shows the signalling needed to perform a Grid-triggered session transfer. In this scenario, the SIP Broker makes use of the REFER message to ask the Source MT to transfer the session. This should not be strange, as the REFER method can be used to make a remote UA send any other method, not only INVITE. In this case, it is used to cause another REFER, and this is why this mechanism is known as *Nested REFER*.

Apart from this, the signalling involved is the same as in the previous scenario. The REFER sent by the SIP Broker also creates a subscription in the Source MT, which will inform the SIP Broker about the progress of the transfer via NOTIFY messages.

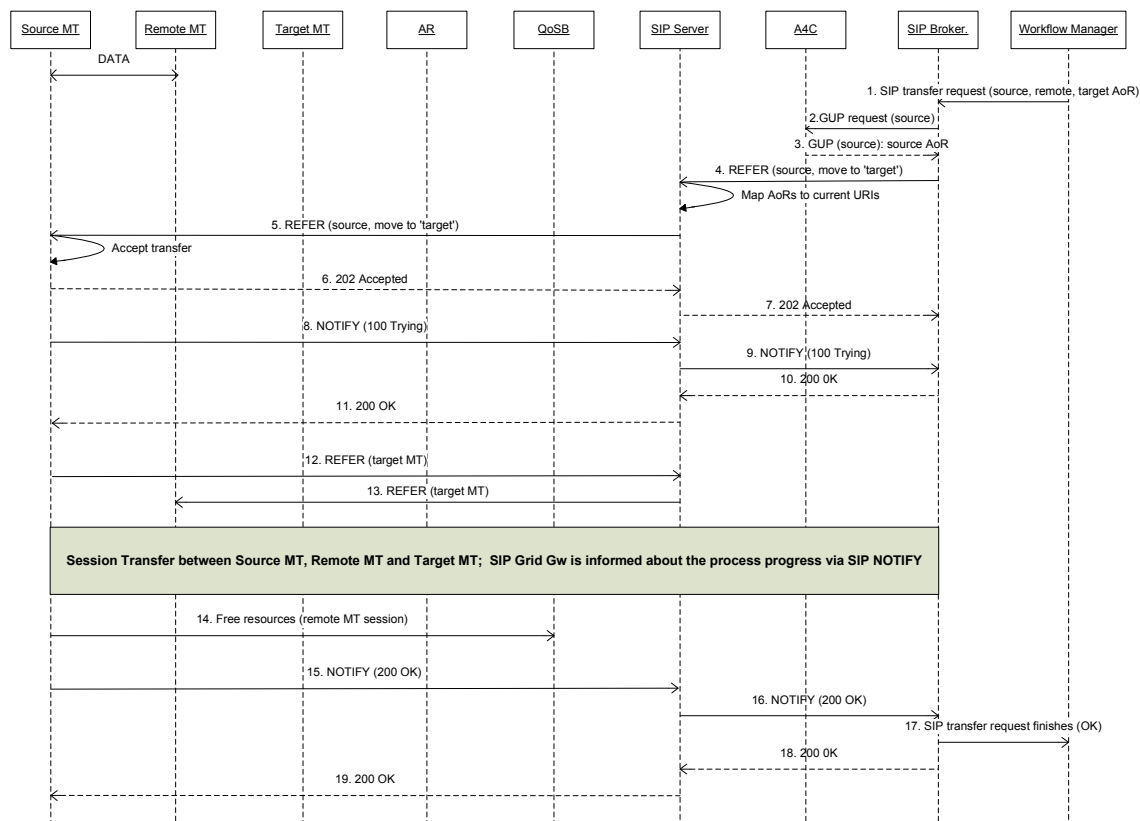


Figure 5 – SIP Session Mobility (session handoff mode). Grid triggered

3.4. Terminal Mobility

Terminal mobility refers to the ability of a terminal to remain connected to the network even if it changes access point or access technologies.

SIP alone is not enough to guarantee seamless terminal mobility, for if a non-Mobile IPv6 capable terminal changes access point, it loses connection to its previous network and gains a new IP address in the new network. In practice, that means that all its ongoing communications before the change, are stopped and have to be restarted.

3.4.1. Architecture

Terminal mobility in Akogrimo is achieved with the usage of Mobile IPv6 [MIPv6]. The Mobile IPv6 protocol enables mobility at layer 3 in a transparent manner, i.e. higher layers don't need to know, and in fact are not aware if the terminal they are communicating with is using mobile IPv6 or not.

For implementing Mobile IPv6 in a network, one new entity has to be added to the network:

- Home agent (HA): A node that resides at the MT's home network, which is in charge of forwarding traffic directed to the MT while it is away from its home network so that it looks as though the MT is virtually at his home network. To this end, the HA must be aware of the MT's current binding.

Additionally, mobile terminals have to be modified in order to have support for Mobile IPv6. The Mobile IPv6-enabled terminal may be described as

- Mobile terminal (MT): A node with the ability to change its point of attachment to the network keeping connections alive.

Any entity communicating with the MT is called the correspondent node.

- Correspondent node (CN): Any peer node which a MT is communicating with. A CN does not necessarily have to implement mobility i.e., a CN may be either mobile or stationary, with or without mobile IPv6 support.

When the MT is away from his home network the HA will act on behalf of it forwarding all the packets addressed to the MT at its Home Address (HoA) to its current Care of Address (CoA) by means of a tunnel. The CN is not aware that the MT is roaming unless it supports Mobile IPv6. If this is the case, the CN will realize that its traffic is being tunnelled by the HA and may perform a route optimisation by directly communicating with the MT by means of the MT's CoA instead of using the tunnel provided by the HA. This should be the standard mechanism because it notably increases the performance. In order for the MT to communicate its position to the HA and all the CNs it is communicating with, a procedure is launched in which a packet called Binding Update (BU) is sent with the address currently used and a packet called Binding Acknowledgement (BA) is received by the MT as a confirmation.

3.4.2. Interaction with other subsystems

Mobile IPv6 is transparent for higher layers. However, in some situations the QoS Broker must know whether or not a terminal is outside its home network. For that purpose, Akogrimo's Home Agent includes a module which will communicate with the QoS Broker.

That module's function is solely to translate any HoA into CoA, and vice-versa. The results of that translation will allow the QoS Broker to better perform its decisions.

3.4.3. Signalling

3.4.3.1. Fast Handover

The goal of Mobile IPv6 is to achieve seamless mobility of devices. Due to its movement, the MT may have to change its point of attachment to the network by using a new access router that is closer to its current position. This change of point of attachment to the network is called handover and involves acquiring a new CoA and communicating the HA and CNs its new location before it can be fully operative again and the communication can be re-established. Voice applications, for example, are very sensitive to delays and need the handover to be fast enough in order to be seamless.

Fast Handover [FHO] reduces handover latency by anticipating the handover by means of the link layer. It allows the MT to start sending packets on a new network link as soon as it is detected, and also allows the MT to receive packets as soon as its attachment to the new link is detected. This way, while maintaining the previous connection intact and sending packets through it, the MT also sends packets through the new network link (effectively duplicating the

packets). This process, also known as bicasting, avoids the loss of packets during the handover process and consequent degrading of quality in applications that are making use of the network.

3.4.3.1.1. Mobile Terminal Initiated Handover

Figure 6 depicts a scenario in which a Mobile Terminal is moving from the Old Access Router (oAR) to the New Access Router (nAR). Numbered messages show how the FHO is executed. The explanation of the messages is given below.

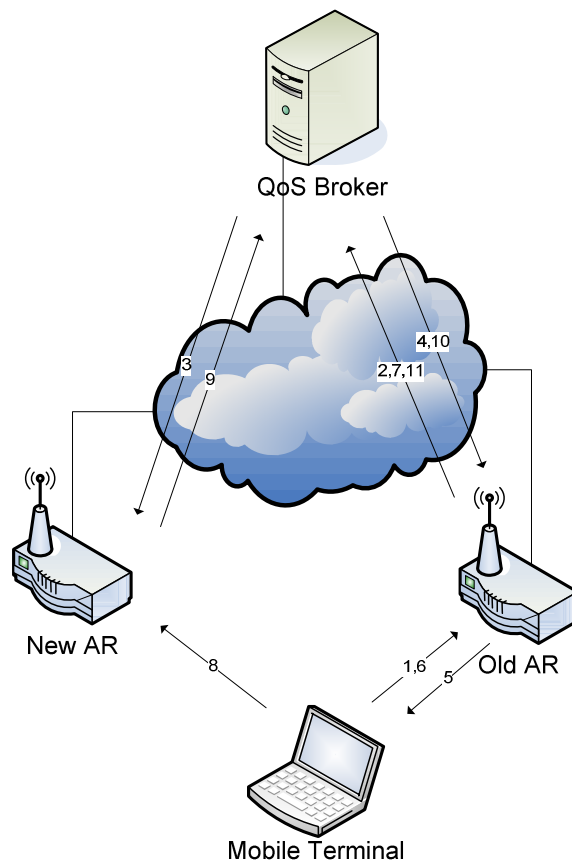


Figure 6 – Fast Handover

The Mobile Terminal detects a new access network and decides that it should move there. The reason may be related to better signal in the new network or user preferences. The MT begins the FHO process by sending a *HandoverRequest* message to the current AR (1). Upon receiving this message the AR sends a request for approval of the handover to the QoS Broker (2). The QoSB, after verifying that the requested access network has available resources to match the user's requirements, sends a *HandoverDecision* message to both the oAR and nAR (3,4).

The oAR informs the MT that it can move to the new access network by sending it a *HandoverResponse* message (5). When the MT receives that message, it performs some internal checks and sends a *FastBindingUpdate* to the oAR (6) which is subsequently reported to the QoSB (7). The MT then configures its layer 2 connection to the new link and sends a *FastNeighbourAdvertisement* message in order to populate the nARs neighbour cache (8).

Having received the FNA, the nAR notifies the QoSB that the handover was successful (9) and the QoSB informs the oAR of the handover's success (10). Finally, the oAR informs the QoSB that information about this MT has been deleted and that MT's control has been handed to the nAR (11).

3.4.3.1.2. *Network Initiated Handover*

In this scenario, it is the network which decides that the MT must perform a handover to a new access network. The QoSB sends the oAR a *HandoverDecision* unsolicited message specifying that the MT must roam to the new access network. The oAR sends a *HandoverResponse* to the MT, which upon its reception must perform the necessary steps to make the handover to the nAR.

In case the new access network is no longer available to the MT, it can request aborting the procedure by sending a FBU with a negative acknowledge.

The steps of the NIHO are the same as the MTIHO, except the former does not include steps 1 and 2.

3.4.4. Security

The use of Mobile IPv6 should not make the network less secure. In order to do that, MIPv6 has already some security mechanisms. Others mechanisms are also used to further improve security.

Binding Update protection is necessary for preventing an attacker from disrupting or hijacking the communication between two nodes.

The use of IPSec extension headers allows the protection of messages exchanged between the MT and the HA. It also protects the authenticity and integrity of Mobile Prefix Solicitations and Advertisements.

Protection of messages between MT and the correspondent node is achieved through a procedure called Return Routability.

Security is developed further in chapter 7 of this deliverable.

4. Quality of Service

4.1. Introduction

Quality of service can be defined in a very basic sense as the consistent and predictable delivery of data. The Akogrimo network must be able to provide such diverse services as interactive audio (e.g. a SIP audio session) or large data transfers (e.g. some Grid service which requires large portions of data to be transferred among different machines for computation). These diverse services have completely different requirements on a network level. The former requires a relatively small amount of bandwidth (e.g. 64Kbps) but very strict latency requirements, in order to allow a normal conversation without interruptions. The latter requires large amounts of bandwidth (e.g. 512Kbps or more) but is tolerant when it comes to latency, since a delay does not affect the computation of the data. Quality of service is used in this case to accommodate the different network flows according to their requirements and at the same time optimize the use of the network resources. QoS also has an impact on the reduction of network infrastructure costs.

This section will present the functioning of the QoS system and the interactions among the major components involved in QoS.

4.2. QoS Architecture

4.2.1. Overview

Figure 7 shows the 4.1 work package components which are responsible for managing QoS in the Akogrimo network. All other components, including possible QoS clients which can make QoS requests but are not involved in managing it, are dimmed.

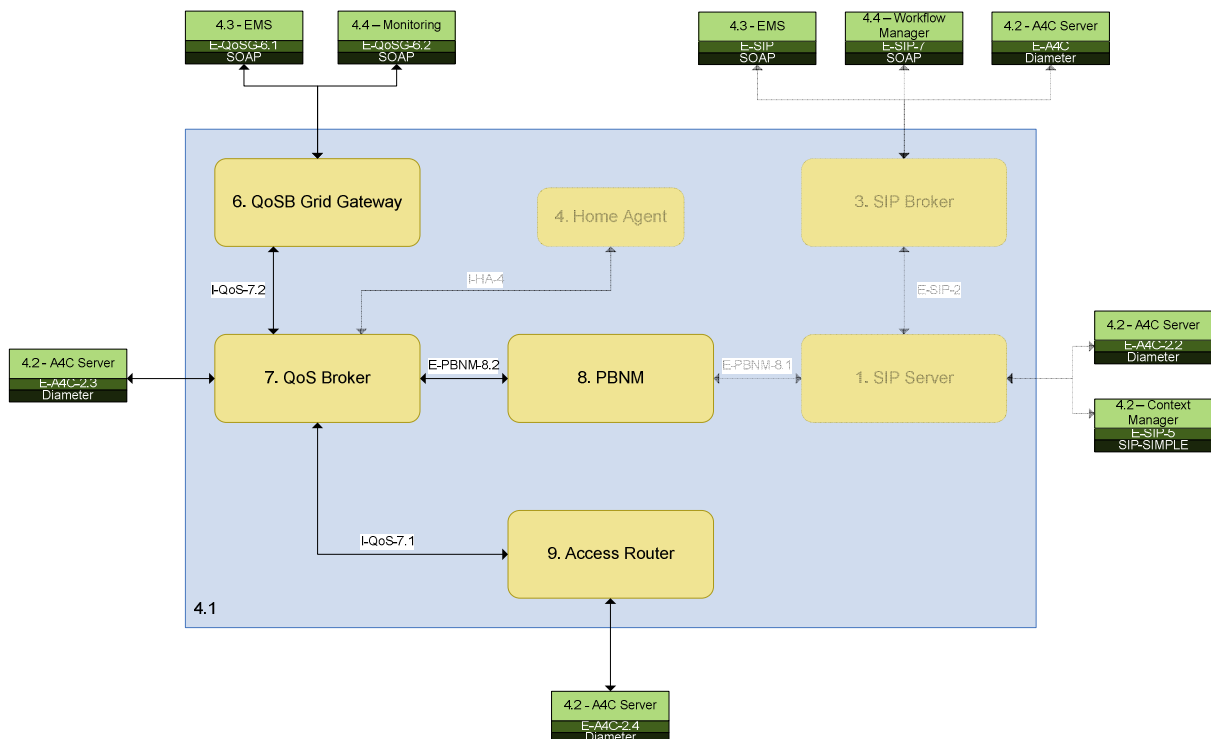


Figure 7 – QoS subsystem components

4.2.2. Components involved in QoS subsystem

The PBNM system is responsible for distributing policies to the QoS Broker. It is discussed in chapter 6.

The QoS Broker manages all network resources that belong to the network(s) it is responsible for. It knows at every moment the usage level of the network resources and the users; with that knowledge combined with the policies it receives from the PBNM, it is able to make informed decisions whenever a request from a user arrives. Additionally, the QoS Broker can receive requests from the Execution Management Service (EMS) through the QoS Grid Gateway. The QoS Broker is described in chapter 4.3.1.

The Access Router, besides providing access to users of either wireless or wired technology, will enforce the QoS rules it gets from the QoS Broker. Whenever a user starts a network flow with a determined QoS level, the AR checks if the user is allowed that QoS level or otherwise and correspondingly allows or blocks the network flow. The Access Router is described in chapter 4.3.2.

4.2.3. QoS Bundles

To make an effective use of network resources and to assure that a user can utilize the services he is entitled to without disruptions, the Mobile Network Layer will implement end-to-end Quality of Service, for all types of mobility. The QoS system allows fine-grained QoS reservations at the access networks, but aggregates different flows with the same QoS requirements in the core network.

For implementation and scalability reasons, the network supports well defined QoS bundles, strongly influenced by the existing models for mobile technologies (UMTS). Each of the three defined bundles is designed for a specific usage profile, audio, video and data. A QoS Bundle is comprised of several well defined services, which the user may choose from when using the Akogrimo network. Table 1 presents these bundles.

Bundle 1	Bundle 2	Bundle 3
Mix audio + data	High data + video	Mostly voice
10 – Interactive	20 – Interactive	10 – Interactive
100 – Data	1000 – Data	1 – Priority
1 – Priority	200 – Priority	1 – Signalling
1 – Signalling	1 – Signalling	250 – Best Effort
250 – Best Effort		

Table 1 – QoS bundles and usage. Values are in kilobytes per second

Signalling: This is traffic needed so as to maintain and support the network infrastructure; therefore it is the highest priority. It is time-critical and, in fact, essential to network operations as a whole. Typically its bandwidth requirements are very low.

Interactive real-time: This is time-critical traffic that will be used mainly for video conferencing or audio communications. Interactive multimedia applications are very sensitive in regards to delays. The delays allowing for optimal functioning of interactive applications are less than 100ms. Latency, jitter and out-of-order packets also affect adversely voice communications.

Priority: This type of traffic is not time-critical, but it is important, such as multimedia streaming, or some grid application data exchange. It is higher priority than Data Transfer, but has lower bandwidth available typically.

Data Transfer: Data transfer is somewhere in between Priority and Best Effort (BE). This type of traffic is not time-critical but may be loss-sensitive. Furthermore, out of order packets are typically not a concern for applications that use Data Transfer.

Best Effort: As the name implies this service offers best effort. If the network conditions are good, this should be fine for most applications. If the network is heavily loaded, BE will be the most affected. This is basically what Internet provides.

4.3. QoS subsystems and interactions

4.3.1. QoS Broker

The QoS Broker is the network component which effectively manages all network resources. It receives global network policies from the PBNM system. Requests for QoS from access routers have to be approved by the QoS Broker which then sends appropriate configurations to the ARs. It also exchanges information with other QoS Brokers.

Mobility is also dependent on the QoS Broker, since a user may not change Access Network (AN) if, for example, the new AN has all its resources already occupied. When a user moves from an access network to another then both brokers are involved in the process, assuming that the different access networks are supported by distinct QoS Brokers.

Figure 8 shows the QoS Broker functional architecture:

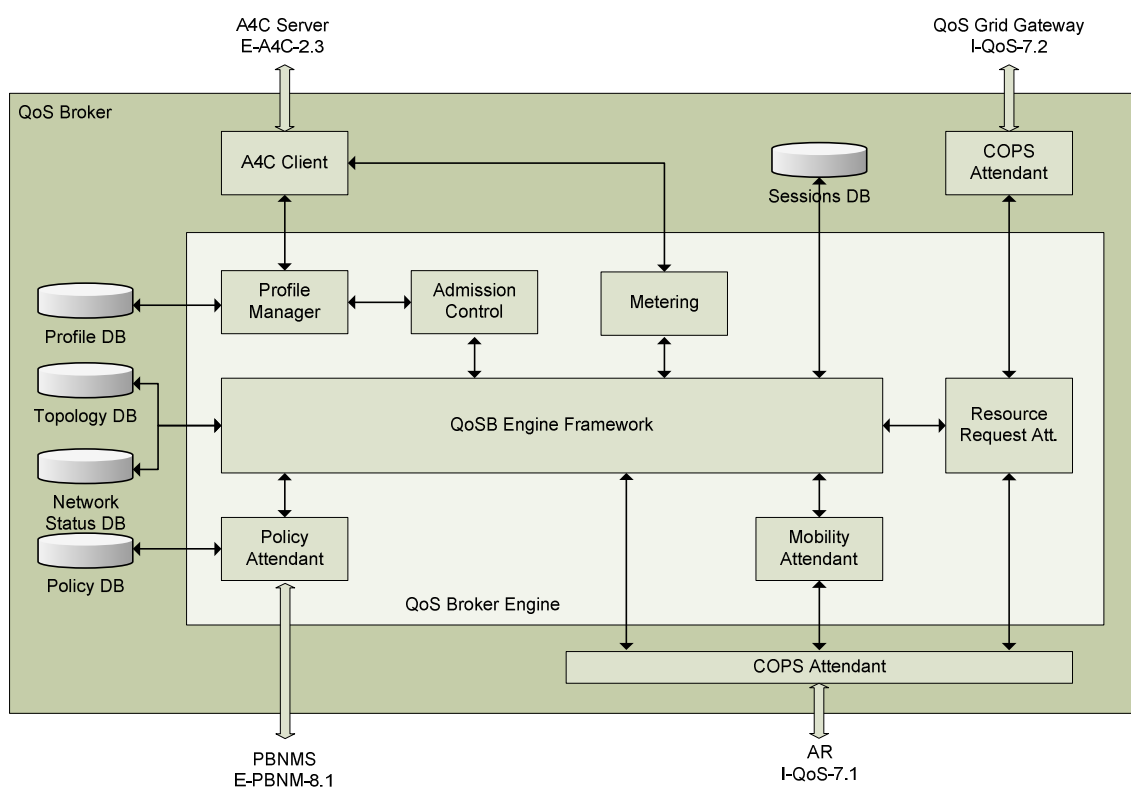


Figure 8 – Akogrimo QoS Broker

The QoS Broker has a Policy Attendant which communicates with the PBNMS and stores the policies it receives in a policy database. It is also possible for the QoS Broker to request policies by itself.

The COPS Attendant module allows it to communicate with the AR using the COPS protocol [COPS]. It is through this interface that the QoS Broker will receive QoS requests and other information from the Access Routers. It will also be used to send appropriate configurations to the ARs.

The A4C Client module allows the QoS Broker to retrieve user profiles from the A4C Server. Those profiles are then cached in the Profile database for efficiency reasons. This eliminates the

need to do a request to the A4C Server each time the QoS Broker needs information from the user's profile. The A4C Client module is also used to communicate metering data, received from the Access Routers, to the A4C Server after having been processed by the QoS Broker. For a sample user profile, see Annex C.

A COPS Attendant module is responsible for receiving QoS requests from the QoS Web Service. This module also sends QoS status notifications, for SLA monitoring purposes. These messages are transmitted to the QoS WS which then communicates with the SLA Monitor.

The Topology, Network Status and Sessions databases are used for QoS Broker internal operation:

- Topology DB – holds information about routers and respective interfaces. It allows the QoSB to find an actual route for a data flow.
- Network Status – allows the evaluation of whether there are available network resources for establishing a new session.
- Sessions DB – holds information about ongoing sessions.

4.3.1.1. Differences to previous version

There are two main differences between the present architecture and the previous version which was presented in deliverable D4.1.1.

- **Accounting:** in the previous version accounting was only present in the Access Router. Upon the implementation of the actual software it was realised that it would be beneficial to move the accounting module to the QoS Broker. Having the AR do all the accounting work would imply added communications between the AR and the QoS Broker for retrieving user information which is necessary to correlate the network flows and the users responsible for them. It would also mean that it would be necessary to add logic to do that operation. The present architecture is simpler, albeit introducing an extra step in the accounting communications.
- **QoS Grid Gateway:** while in the previous version a web service interface was presented as a solution for communicating with the EMS, that interface has evolved into a separate component. This component has a web service based interface to the EMS. That interface is in fact not limited to the EMS – it can also be used by any other service that wishes to do so. For communicating with the QoS Broker it uses a COPS interfaces, much like the Access Router.

4.3.2. Access Router

4.3.2.1. Overview

The Access Router acts as an interface between wired/wireless terminals and the core network. A single AR may support different access technologies. The rest of the network will be able to communicate with different mobile terminals in the same way, regardless of the technology they are using for network access.

The Access Router also provides network management components (such as QoS Broker, for example) with a uniform way of managing a heterogeneous network environment such as the one which will form the Akogrimo network.

The AR resides on an Access Network at the edge of the Akogrimo network, and thus, any attempt to unlawfully use network resources without proper permission will be blocked at the AR.

A user is required to authenticate himself to the provider before he can use any resource of a provider's network. User authentication requests are intercepted by the AR and then forwarded to the A4C Server.

A user is immediately authorised once he has successfully authenticated. That authorisation allows him to use basic network resources such as moving between different ANs.

Figure 9 shows a functional view of the AR.

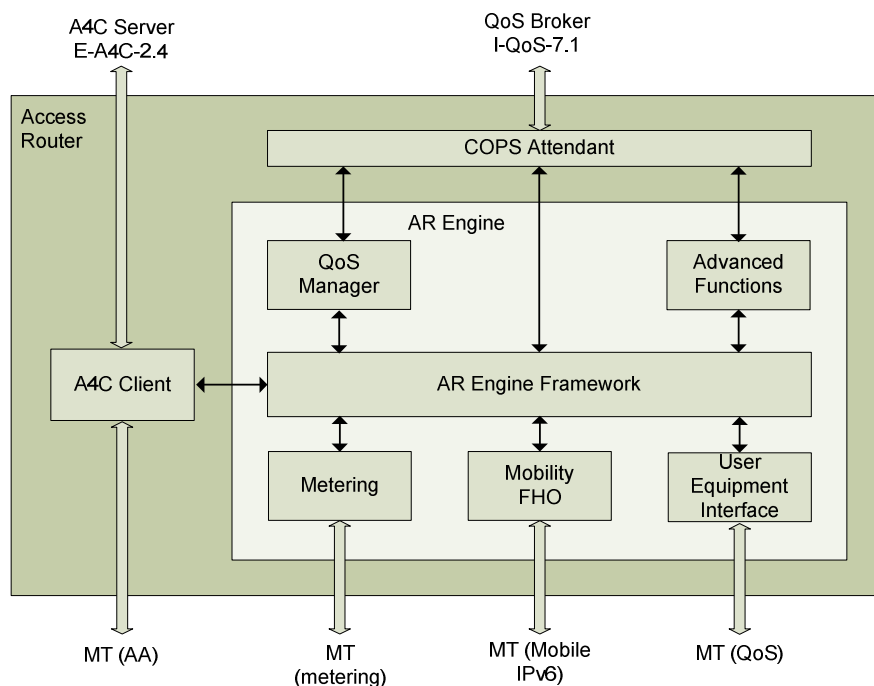


Figure 9 – Akogrimo Access Router

The Access Router is comprised of the AR engine which provides most of its functionalities. It also incorporates an A4C Client module which communicates, using the Diameter protocol, with the A4C Server for authenticating users. The AR engine is, in its turn, formed by the following modules:

- **COPS Attendant:** The COPS Attendant is the main interface of the AR. It is through this interface that the AR communicates with the QoS Broker. QoS requests from the mobile terminal will be translated into COPS messages and then delivered to the QoS Broker. QoS requests may come in the form of explicit RSVP messages or DSCP packet marking; whatever the case, and assuming that the AR is properly configured to handle those options, the AR will process them and deliver the appropriate COPS message to the QoS Broker.
- **QoS Manager:** The QoS Manager module is responsible for managing per-user flows established in the AN, as well as the per-hop behaviour of the AR.
- **Metering:** The metering module is able to do per-flow metering. Its results are communicated to the QoS Broker, which then does some processing for relating flows with users and finally delivers the results to the A4C Server.
- **Mobility+FHO:** The Mobility and FHO module will deal with users' mobility requests.
- **Advanced Functions:** Advanced Functions is a group of modules that implement certain advanced mechanisms which may be used by the AR:
 - **QoS Trans-signalling:** The translation of external QoS signalling into internal DiffServ signalling, appropriate for the CN, is done by the QoS Manager module. This translation is then sent to the QoS Broker, where it is validated and communicated to the rest of the

network. Having the AR do the translations and the QoSB the validations and communications allows the distribution of processing effort among ARs and QoSB's.

- **Advanced Signalling Processing:** This module is capable of processing signalling messages from layers above layer 3, such as SIP. It can inspect packets and retrieve QoS related information, such as session bandwidth from SDP packets. With this information, the AR may setup a temporary resource allocation which would then be used in case the session setup is successful, thus diminishing the time necessary compared with normal setup.
- **Legacy QoS:** This module provides support to legacy applications that are unaware of the Akogrimo network QoS framework.
- **User Equipment Interface:** The User Equipment Interface is in fact a group of modules that deal with communications to and from the mobile terminal. These communications can be done through a signalling mechanism, be it explicit or implicit. Alternatively, a mechanism such as connection tracking can be used. The modules that compose the UEI are:
 - **DSCP:** The DSCP module is capable of recognizing and controlling network flows marked with a specific DSCP code. It is also capable of re-marking the packets, should the need arise.
 - **RSVP:** This module extracts information from RSVP requests and passes that information to other modules.
 - **SIP:** If required, SIP requests are intercepted by the ARs Advanced Functions module and they are processed, else they are simply forwarded to the SIP Proxy. Processing of SIP messages is done to allow the extraction of relevant QoS information or even change the messages themselves.
 - **Connection Tracking:** This module is capable of tracking TCP sessions. Other transport protocols may also be tracked, allowing customised services on a per-flow/session basis.

4.3.2.2. Differences to previous version

There is one significant difference between the present architecture and the previous version presented in deliverable D4.1.1.

- **Accounting:** as stated in the previous chapter, the AR is no longer responsible for sending accounting information to the A4C Server. Its responsibilities in that matter are limited to gathering metering data and providing it to the QoS Broker, which will handle them appropriately.

4.3.3. Mobile Terminal

The mobile terminal runs a Linux kernel patched with Mobile IPv6 [MIPL] and Netfilter. The MIPv6 patch obviously enables Mobile IP functionality. The Netfilter patch is necessary for implicit signalling to work. It permits

- the marking of packets with a specific DSCP
- the matching of packets with a specific DSCP

4.3.4. Policies/PBNM interactions

The network components must be configured according to the network operator's requirements. Since configuring large networks can be tedious and error-prone, Akogrimo will use Policy Based Network Management system, which will allow network administrators to use a simple interface for configuring global network policies that will be propagated to the rest of the network.

The PBNM system is responsible for providing network policies and configurations to the QoS Brokers, which will then configure the Access Routers. For more information on PBNM, see chapter 6.

4.3.5. User profiles/A4C interactions

The user profile is stored in the A4C Server. When a user connects to the network, the QoS Broker will request the user's profile from the A4C Server, and will cache it during the user's session in order to avoid repeating requests for the user profile.

The user profile holds information about the services which the user has at his/her disposal. With that information, the QoSB is able to perform the decisions on whether or not the user is allowed access to a specific service.

A sample skeleton of a user profile is available in Annex C.

4.4. Signalling

This section will present the signalling mechanisms which QoSB, AR and MT use to communicate with one another.

4.4.1. QoS Broker <-> Access Router

Signalling between the QoS Broker and the Access Router is done with the COPS protocol which provides all necessary functionality as well as the possibility of being extended to accommodate new functionalities.

The Common Open Policy Service protocol (COPS) is a client/server policy exchange protocol. The clients (Policy Enforcement Point or PEP) send requests to the server (Policy Decision Point or PDP). The PDP then makes a decision and sends it to the PEP. The COPS protocol is extensible; it can recognize new self-identifying objects and supports client specific information without changes to the protocol. It uses TCP for transport, provides message level security for authentication, replay protection and message integrity. It can also use IPsec [IPSEC] or TLS [TLS] for authorisation and securing communications between the PDP and PEP.

4.4.2. Access Router <-> Mobile Terminal

The signalling between the Mobile Terminal and the Access Router may be handled in two different ways – implicit signalling and RSVP [RSVP]. The functionality provided by the multiple different ways is, however, equivalent. In Akogrimo only implicit signalling has been used; that choice was based on its simplicity and performance when compared with RSVP, although at the expense of some flexibility. The next sections present the different signalling schemes which can be adopted.

4.4.2.1. Implicit signalling

4.4.2.1.1. Overview

Implicit signalling is done by having the Mobile Terminal mark its packets according to the QoS class required, or by having the AR automatically mark the packets of a flow for which a given policy exists.

Implicit signalling is closely tied to the notion of well-known services. Akogrimo well known services that require a specific QoS may be used without requiring any signalling from the part of the MT.

Implicit signalling is also used for support of legacy applications, which are unaware of the Akogrimo QoS-enabled network.

In any case, two distinct possibilities exist for implicit signalling usage:

- The MT marks the packets that applications send. In this case the MT is pre-configured for handling application specific packets and marking them accordingly.
- The AR recognizes a particular flow for which appropriate policies exist which require that flow to be treated with a particular QoS level. In this case the AR re-marks the packets from the default value to the desired value, thus enabling QoS on that flow.

4.4.2.1.2. Packet Marking

Packet marking means setting the Traffic Class field of an IPv6 packet to a specific value. The Traffic Class field is composed by eight bits; the six most significant bits are used for the value of the DSCP (Differentiated Services Code Point), while the remaining two bits are used for ECN (Explicit Congestion Notification). In Akogrimo the ECN bits are also used as part of the DSCP.

The Access Router analyses each and every packet that is sent by the Mobile Terminal. Knowing the DSCP of a packet, the AR then matches it with its internal rules, which have been previously setup by the QoS Broker. If a packet does not match to a rule, it is dropped. Else the packet is sent to its destination through the core network and using the appropriate QoS.

4.4.2.1.3. Message sequence

The following picture shows a message sequence depicting two mobile terminals communicating with each other using network QoS.

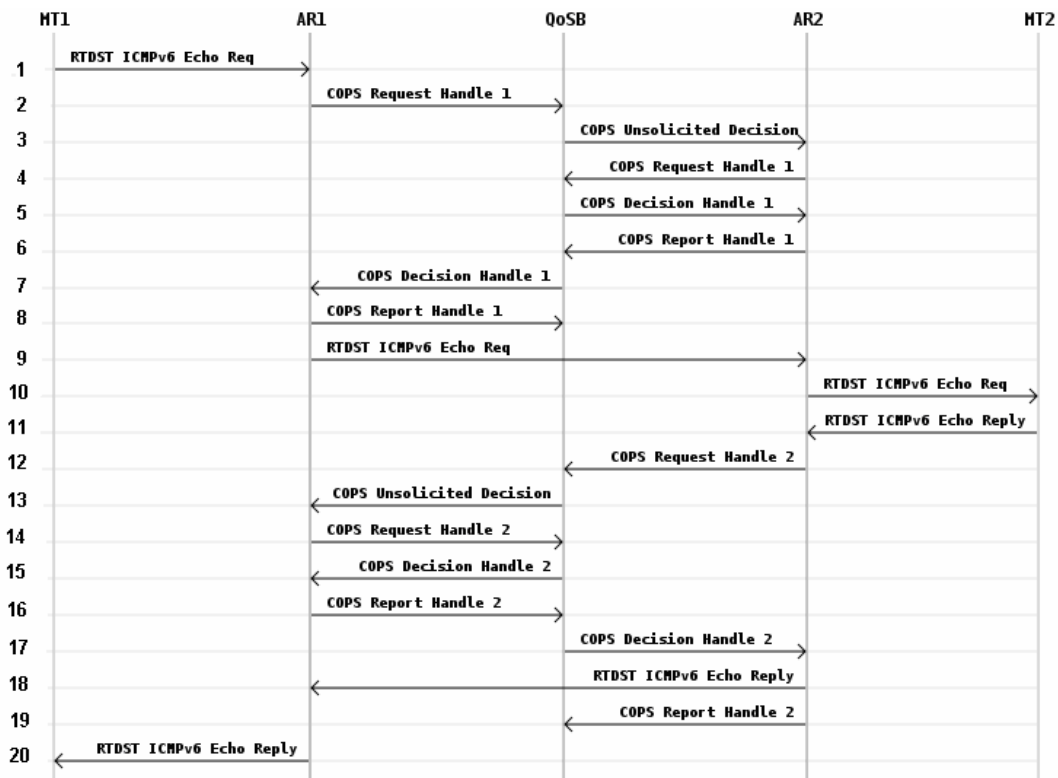


Figure 10 – Implicit QoS request

This example is very simple and uses an ICMPv6 Echo Request/Reply (i.e. a ping packet) for simplicity sake. Any other flow of packets will produce the same results, assuming the network is configured accordingly.

MT1 initiates the communication with a ping packet which was marked in the mobile terminal (1). AR1 analyses the packet and recognises by its DSCP that it requires a determined QoS level. Since it does not know if the user is allowed that QoS or not, it contacts the QoS Broker (2). The QoS Broker finds out that MT1 is indeed allowed to use that QoS for that flow, as is MT2. Furthermore it will contact the Home Agent to find the access network MT2 is located in (not depicted in the picture), thus it will know the AR at the endpoint where MT2 is located. Before responding to the AR1, it will inform AR2 that MT1 and MT2 will be having a QoS session and that it must reserve resources for that (3). AR2 then receives the configuration and sends a report stating that all went OK (4, 5, 6). Only then does AR1 receive its configuration for the flow (7, 8). The ping packet then traverses the core network and is delivered by AR2 to MT2 (9, 10). The ping reply will trigger the equivalent message sequence, but in the inverse direction.

It is worth pointing out that all this signalling occurs only for the first packet of a given flow. Other packets will be processed according to the configuration that was received before and require no signalling overhead.

4.4.2.2. **RSVP signalling**

It is possible to use RSVP for QoS requests thanks to the QoS trans-signalling capabilities of the AR. Figure 11 shows an RSVP request message sequence.

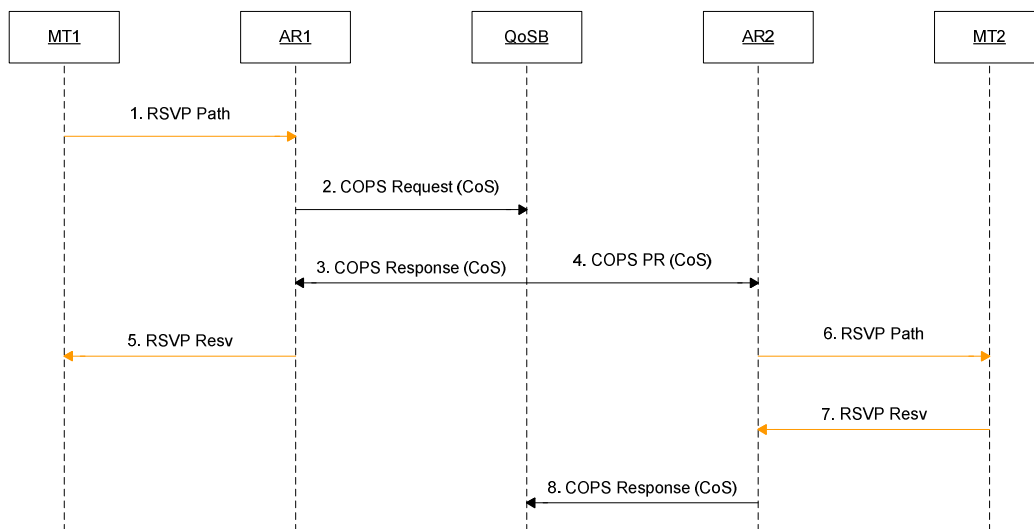


Figure 11 – explicit QoS request using RSVP

In this scenario MT1 issues an RSVP Path request that is detected by AR1 (1). AR1 extracts relevant information and requests the QoS Broker an appropriate configuration and mapping of that RSVP reservation into DiffServ Class of Service (CoS) (2). The QoS Broker then responds to AR1 (3) with appropriate configurations and filters. AR1 then sends a RSVP Resv message to MT1 (5). At the same time, the QoS Broker sends a COPS provision message to AR2 (4) with information that will be used to build a new RSVP Path message and also for configuring AR2. The RSVP message built by AR2 is then sent to MT2 (6) which will respond accordingly (7) with a RSVP Resv message. Finally AR2 responds to the QoSB provisioning message (8).

4.5. QoS and EMS interoperation

The interoperation between QoS Broker (QoSB) and Execution Management Services (EMS) is provided by the QoSB Grid Gateway. This component exposes some of the QoSB’s functionality as Web Services, enabling the EMS to interact with the QoSB.

4.5.1. Architecture

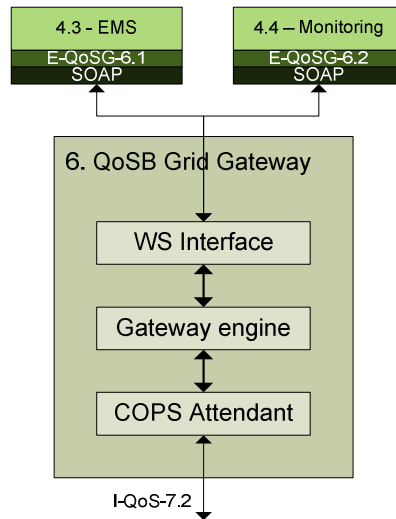


Figure 12 – QoSB Grid Gateway

Figure 12 presents a functional view of the QoSB Grid Gateway. It is composed of three software modules.

- The WS Interface is responsible for communication with other Web Services, namely the EMS and the Monitoring service. It is a web service developed with the Globus Toolkit.
- The Gateway Engine is where most of the logic resides and is developed with the C++ programming language. It communicates with the WS Interface via standard internet sockets, using a simple purpose-specific protocol that allows the Gateway Engine to receive QoS requests and to send the responses to the WS Interface.
- The COPS Attendant is responsible for communicating with the QoS Broker. It is a C++ program that makes reuse of some of the software classes already developed for the AR and the QoS Broker. Communication between COPS Attendant and the Gateway engine is done through normal function calls (API).

4.5.2. EMS Interface

The QoSB Grid Gateway exposes the methods described below. Note that the interface is not limited to the EMS - its use in the methods description is just for simplicity.

Method	Arguments	Purpose
check_qos_availability	user qos_bundle qos_service	This method is used by the EMS to check if the network is able to accommodate the new flow required by the EMS. Note that no reservation is performed by this method.

set_qos	user SLAid qos_bundle qos_service	This method is used by the EMS to perform the actual QoS reservation.
---------	--	---

Table 2 – QoS-EMS interface methods

The *set_qos* method allows the EMS to select not only the *qos_service* (real-time, data, etc.) but also to request the QoS Broker to permit the user access to a better QoS Bundle.

4.5.3. Monitoring Interface

The QoSB Grid Gateway also has an interface to the Monitoring Service. The QoSB Grid Gateway will be the information producer and the monitoring service is the information consumer.

An appropriate notification will be sent if an SLA is not being complied with by the network infrastructure. Since the QoSB will always check the status of the networks, this situation only arises when the network is at or near full load and a high priority network flow, such as a request from a doctor for patient data, arrives at the QoSB.

4.6. Security

Security at the network level involves a wide array of problems, most of which are not Akogrimo specific. As such our focus has been in matters more relevant to Akogrimo.

One of the major concerns is user's safety and data confidentiality. For this reason, IPSec will be used between MT and AR to provide data integrity and confidentiality. It is not possible to use end-to-end tunnel mode, since that would prevent the AR from inspecting packets sent by the MT.

Network entities may use IPSec for communication among themselves. This will prevent an intruder who gains access to the core network from posing as a given machine or eavesdropping on the communication between other machines.

More details can be found in chapter 7.

5. The Akogrimo SIP framework

5.1. Introduction

The Session Initiation Protocol (SIP) is an application-layer protocol that provides mechanisms to establish, modify and terminate sessions. In conjunction with other IETF protocols it can provide advanced support for multimedia sessions, although it is not restricted to multimedia. These protocols are SDP, Session Description Protocol for describing multimedia sessions; RTP, Real-time Transport Protocol for transporting real time traffic and RTSP, Real Time Streaming Protocol for controlling delivery of streaming data. It can provide also user and session mobility. This section is intended to describe how SIP infrastructures cooperate with the rest of the network infrastructures to provide such kind of services.

The Akogrimo SIP-related components are the Mobile Terminal (MT) SIP Module, the SIP Server and the SIP Broker (the old SIP Grid Gateway). A detailed description of them can be found in D4.1.1 [D411] and it was updated on D4.1.3 [D413]; however, the new requirements from the DHCM scenario have introduced some additional changes on the architecture, so an analysis of this evolution will be performed, both at architecture and implementation levels.

5.2. SIP Architecture

5.2.1. Mobile Terminal: SIP Module

The internal architecture of the MT SIP Module (which was previously updated in D413 [D413] in order to avoid dependencies with the type of session being handled) has been revised and simplified again, with the purpose to adopt the new requirements from the DHCM scenario. Figure 13 shows the D413 version of the component, focussed only on the session control part (subcomponents related to the SIP presence handling have not been depicted).

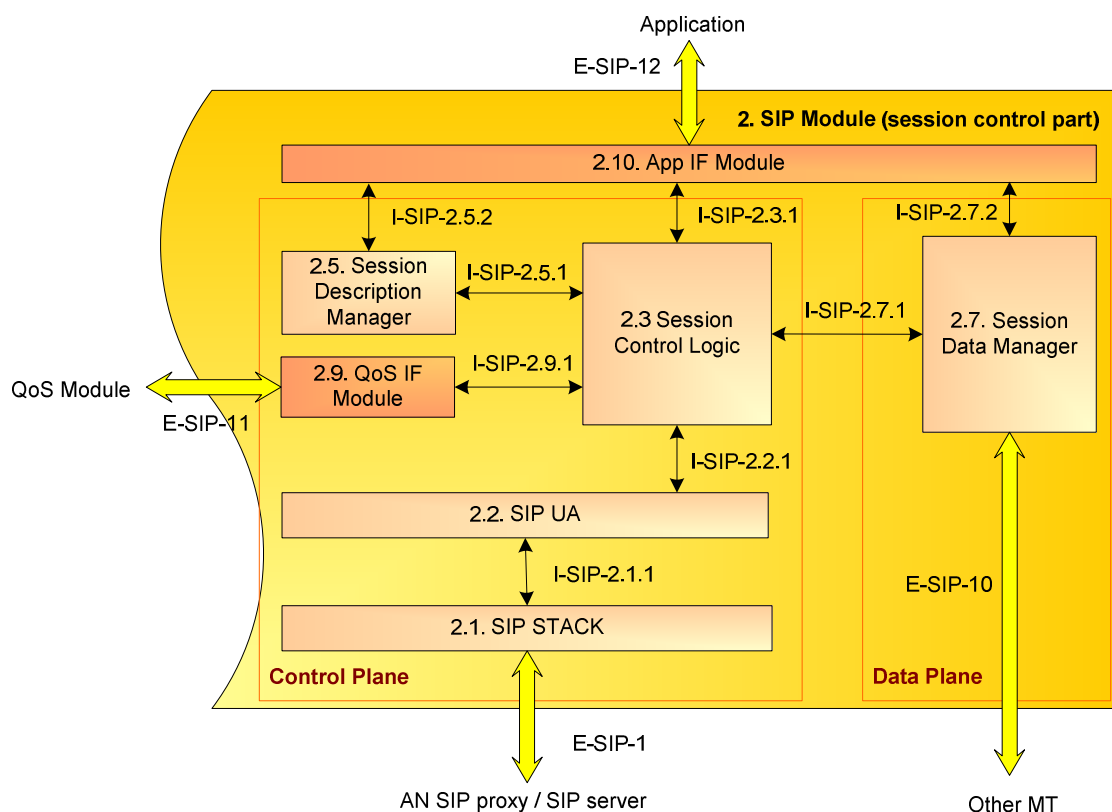


Figure 13 – D.413 MT SIP module architecture

The control plane comprises all subcomponents related only to signalling, which are independent from the type of application being handled. The core part of this subset of components is the Session Control Logic. This component interacts with the data plane core part through a dedicated internal interface (I-SIP-2.7.1) which is used, for example, to inform that a problem with the data exchange has occurred, or to stop this data transmission if the user ordered it.

In order to consider the application-dependant nature of the session handling, in this architecture each application was responsible for the definition of its specific session data handlers, which had to be registered in the MT SIP Module. All registered session handlers were managed by a single component named Session Data Manager, which substituted the old D.4.1.1 Media Manager IF module and offered a generic interface to the signalling core part.

This architecture fits in a suitable way the requirements from the eHealth scenario, in which a few SIP-aware applications had to handle with a single session (one multimedia session, one gSDP session). But in the DHCM scenario, there can be multiple sessions of the same type handled by different applications (e.g. the silver command clerk can have an active multi-conference with several participants, while he is using the surveillance application to monitor some static video-cameras). Since SIP applications need the control on how and where to configure, start or stop the data handlers, and this could be different from an application to another, we have decided not to increase the complexity of I-SIP-2.7.1 and let the applications to get the complete control on the data plane, so the Session Data Manager and related interfaces have been removed from the SIP Module architecture; however, the SIP module offers some utility classes to avoid the developers the building and handling of session data handlers, as will be mentioned in the implementation subsection.

Additionally, and since the session description parsing is also very application-dependant, we have also removed the Session Description Manager sub-module, but providing some utilities to the developers as well.

Considering all of this, the final SIP Module architecture is depicted in Figure 14, which also reflects how Session Description Manager and Session Data Manager are moved to the application level.

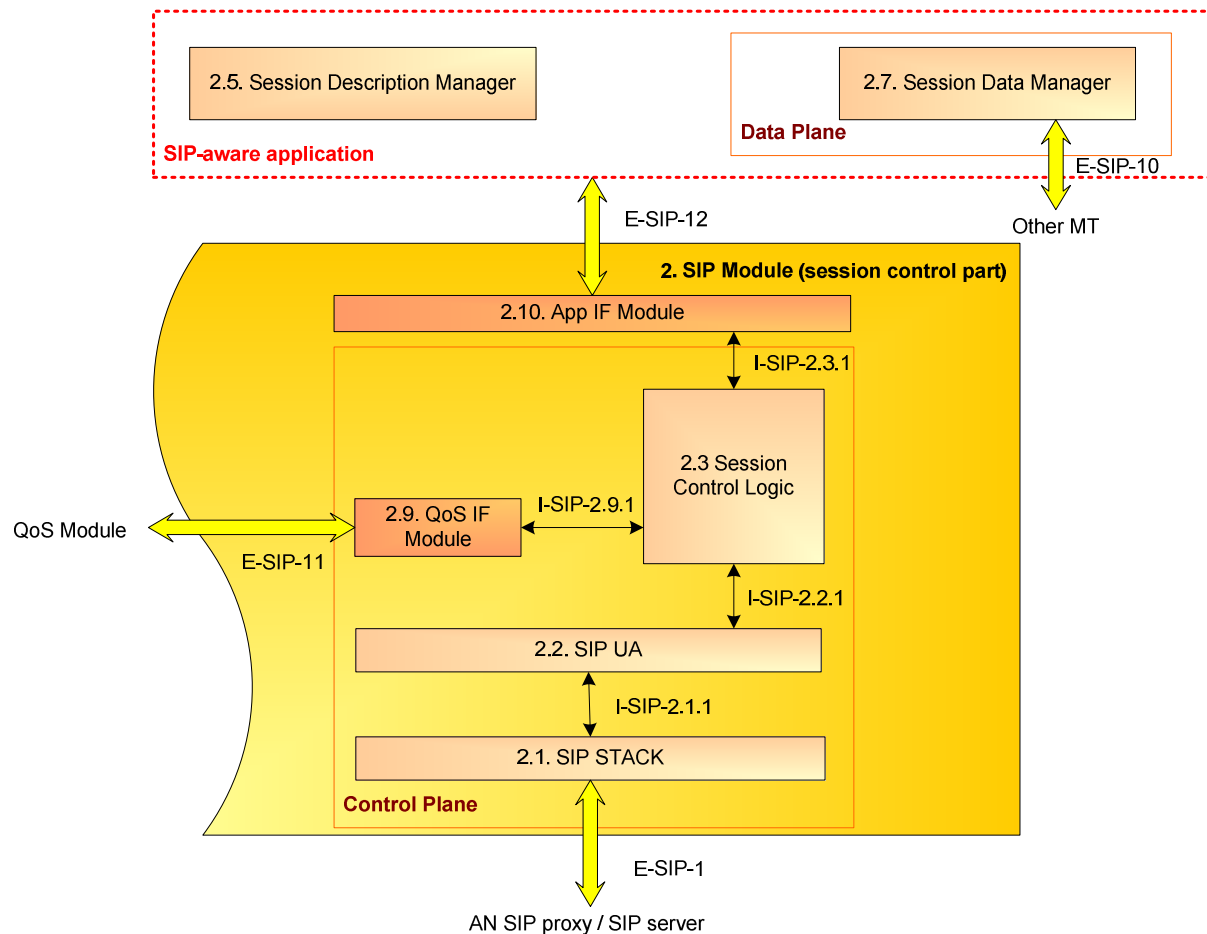


Figure 14 – Revised MT SIP module architecture

Only internal interfaces are affected by this change on the architecture.

5.2.2. Network components

5.2.2.1. SIP Server

No significant changes from the architecture described in D4.1.1 [D411].

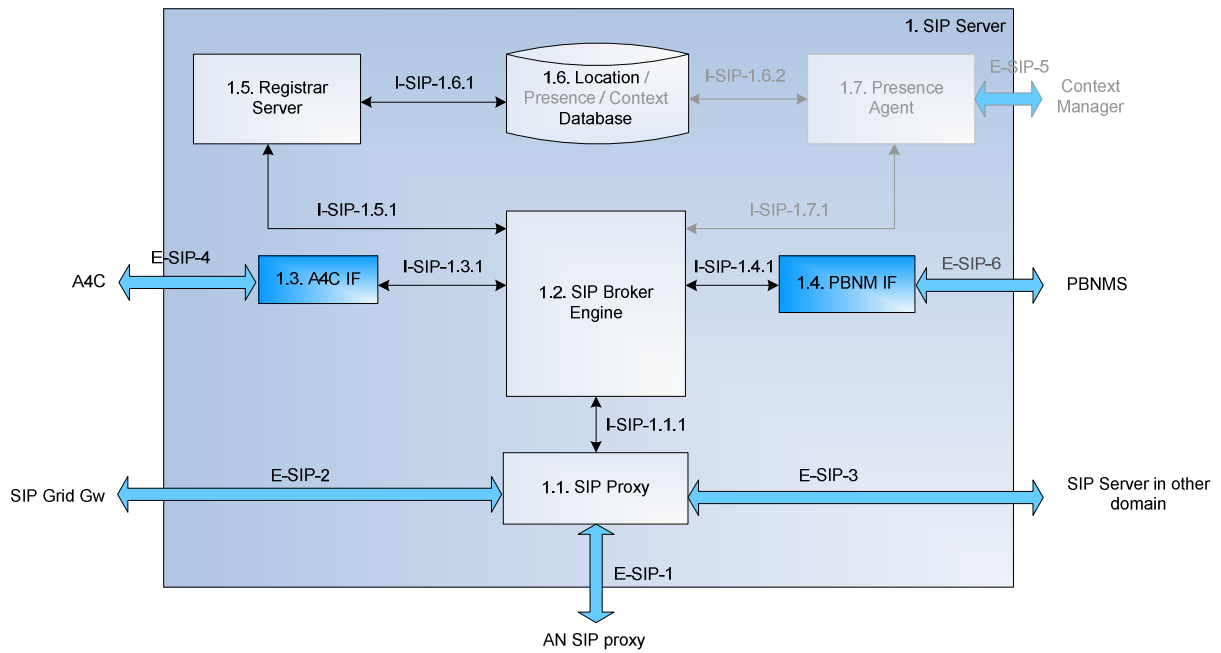


Figure 15 – SIP Server architecture

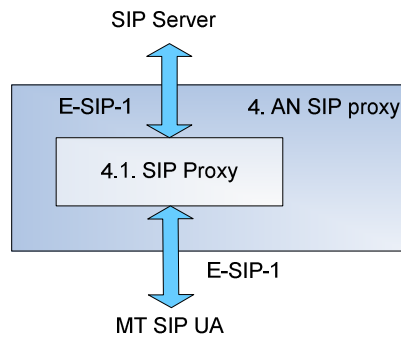


Figure 16 – AN SIP proxy architecture

5.2.2.2. SIP Broker

The SIP Broker can be understood as an application server (following the IMS terminology) which offers SIP-based services (e.g. grid initiated SIP calls and session transfers, gSDP session establishment and termination...) to the upper layers, exposing them as web services.

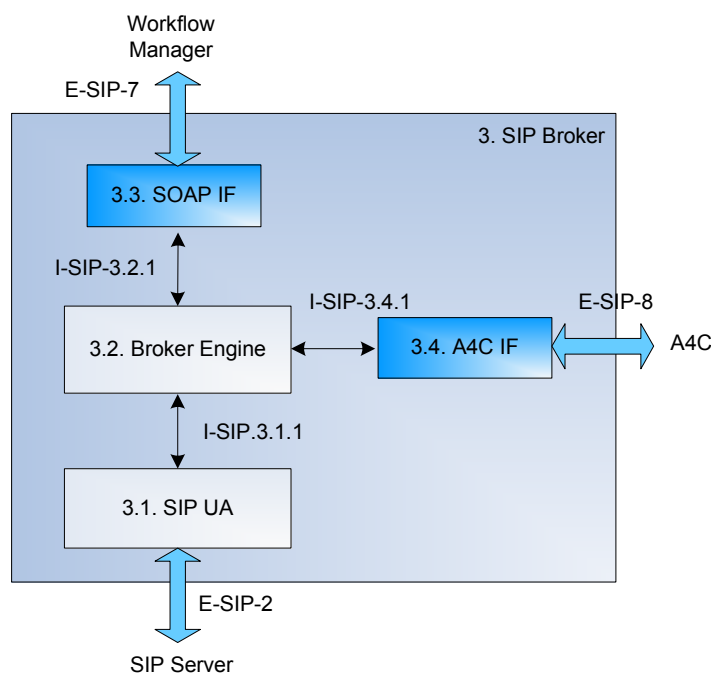


Figure 17 – SIP Broker architecture

5.3. SIP-SOAP synergies

As it has been defined in the updated architecture documents, it is necessary to address the integration of the GRID environment in a ubiquitous and mobile context where the different actors (users and services) can move to a different terminal (with a different network address), or appear/disappear suddenly.

The agreed approach is to integrate SOAP-based Web Services with SIP signalling protocols in a framework for addressing this mobility and ubiquity feature in non-multimedia sessions like Grid sessions, problem also known as the *“SIP with SOAP interaction”* where the main protocols (SIP for ubiquity and SOAP for Web Services) have to interact.

In the proposed approach, GRID applications will be able to:

- Establish a GRID session with a mobile/ubiquitous service without a prior knowledge of its current location.
- Use the SIP mechanisms to manage sessions, like transfer a session, or control a Grid session by a 3rd party.
- Using the presence and context management mechanisms established by the SIP SIMPLE infrastructure.

From a service provisioning point of view, it is needed to provide an interface to support the functionality needed by Grid applications, which includes the following:

- At start up phase, mobile/ubiquitous services will be associated to a certain SIP URI, which has been registered in the SIP registrar with the SIP REGISTER method. This SIP URI can be simply the URI of the user that started the MT where the services are running, or an URI representing the services. Within the context of the SIP-SOAP synergies, from now on we will refer to it as “SIP service URI”.
- The applications, in order to know the necessary service URI to build an EPR to a resource (from now on, WS URI) , should use in the establishment phase a SIP service URIs for

identifying mobile/ubiquitous services, so they will send a SIP INVITE to the SIP service URI.

- In the SIP answer, by means of a specific “Grid Session Description Protocol” (from now on, gSDP), the Grid application will get the needed Grid session information to start the Grid invocations (i.e. the current WS URI for the service in that computer).

Additionally, Grid applications could include the adequate logic needed to:

- Use further session management features provided by SIP like session save/load/restore, useful also for session transfer. There are already some proposals for this feature (Web Services Continuity), like the approach presented in [WS CONT].
- React to context changes which are notified via SIP.

This service provisioning can be done in two ways:

- Provide a SIP API to the Grid applications. Grid applications will handle SIP interactions by themselves, managing registrations and invitations. This is the cleanest approach in terms of architecture, but implies that Grid applications have to be modified in order to use the interfaces offered by the SIP UA or the Session Control Logic for SIP protocol handling.
- Provide some intermediate modules, which will be invoked by the Grid applications and manage the needed SIP interactions on behalf of the Grid applications, even establishing a “Grid session” on behalf of them. These modules will offer their capabilities through standard Web Service interfaces. This approach will ease the implementation of the Grid applications, since they will need to use just SOAP messages even for SIP-related methods. Thanks to these modules, SIP information such as the current location as well as notifications about the status of mobile services running at registered MTs on the SIP Server, are propagated to the Grid layer and vice-versa. In this way, the Grid layer can keep track of the mobile users and readjust itself to changes on the fly.

Finally the second approach (the usage of intermediate modules) was adopted in order to minimise the changes of the grid applications; however, the first option is also a valid approach if future projects are intended to continue on this research area.

Next section describes how the chosen solution (named “Services SIPification” works within the Akogrimo context.

5.3.1. Services SIPification framework

As mentioned before, the adopted solution involves some intermediaries in which the grid applications delegates in order to handle all the underlying SIP interactions. Depending on the role and the nature of the involved Grid services, two types of intermediaries are envisaged:

- For the mobile/ubiquitous grid services running at MTs, considering that these MTs already have a SIP module, the simplest solution is to have a specialised application handling the gSDP sessions. This MT gSDP application should be active in the MTs where the service to be invoked is running and it will offer some API to the services in order to modify the services information. In order to interact with the MT SIP module, it must follow the general rules for building Akogrimo SIP applications described in section 5.4.1.
- For grid services not running at the MTs (typically the EMS service), the current SIP Broker module WS interface will be extended, adding a WSRF-compliant WS developed on top of GT4 that acts as a gateway service in front of the SIP infrastructure. Using these new facilities, a grid service will be able to retrieve the current WS URI of another service running on a MT, be aware of the service availability, etc.

The way in which all these elements interact is described in the following MSCs.

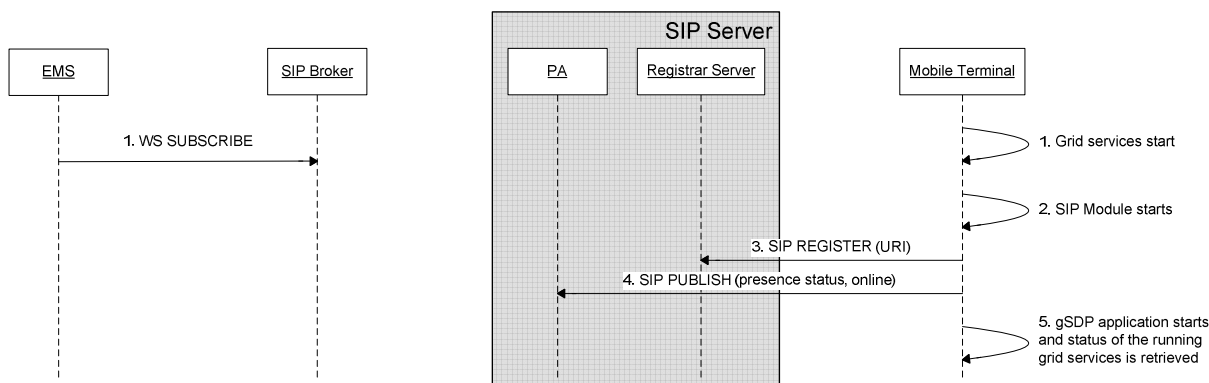


Figure 18 – Intermediaries and interfaces setup

During the MT startup, the grid services, the SIP module and the gSDP application which will act in behalf of the grid services are started following the depicted sequence in the right side of the picture. The SIP module startup involves a SIP registration and a SIP presence publication, in order to make visible the MT for other grid services that could be interested in the grid applications running on it. After the gSDP application starts, the MT will be ready to provide information about the grid services running on it. The SIP service URI to be used will be the SIP URI being used during the SIP registration.

The interface of the intermediate module running on the SIP Broker has also to be initialised, as shown in the diagram (left side). After the initialisation, the consumer service (e.g. the EMS in the sequence) will be ready to receive WS notifications.

Next sequence shows how the WS URI of a mobile/ubiquitous service can be retrieved using this framework, in order to invoke it.

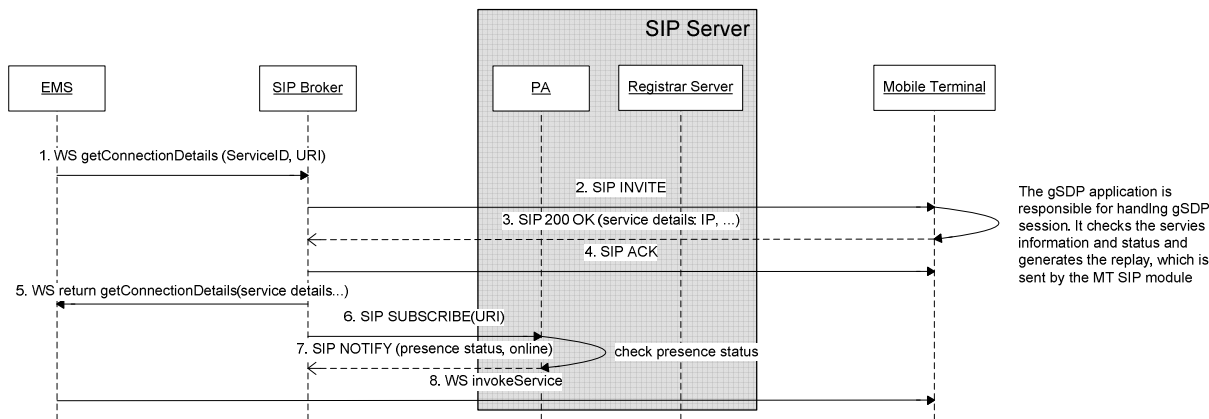


Figure 19 – SIPification session establishment and service invocation

When the EMS needs to know the WS URI of a service running on a MT, it invokes the operation `getConnectionDetails` offered by the SIP Broker, indicating which service it is interested in and the SIP URI being used by the MT in which the service is expected to be running (1). The EMS can also request details about all the services running in the specified terminal. The SIP Broker starts a SIP gSDP session setup with the required MT (2, 3, 4), whose status basically provides the information the EMS needs to invoke the services. Finally, the method `getConnectionDetails` returns with the required information (5). Annex D.1 describes the agreed format used by the SIP gSDP session, which is also used as return value to the EMS. At this point, the EMS knows the current WS URI to be used in order to invoke the

service or services it is interested in (8). Note that an EPR to a service resource is created using the WS URI that is returned to EMS from the SIP Broker.

Additionally, the SIP Broker performs a subscription to the SIP presence information of the SIP service URI (6, 7); in this way, it will be aware on the availability status of the whole terminal and it will be able, for example, to inform the EMS if the user disconnects from one location and then connects from a different one. Using this feature, the EMS can take an appropriate decision, like getting the WS URIs at the new services location. This is the behaviour described in the figure below.

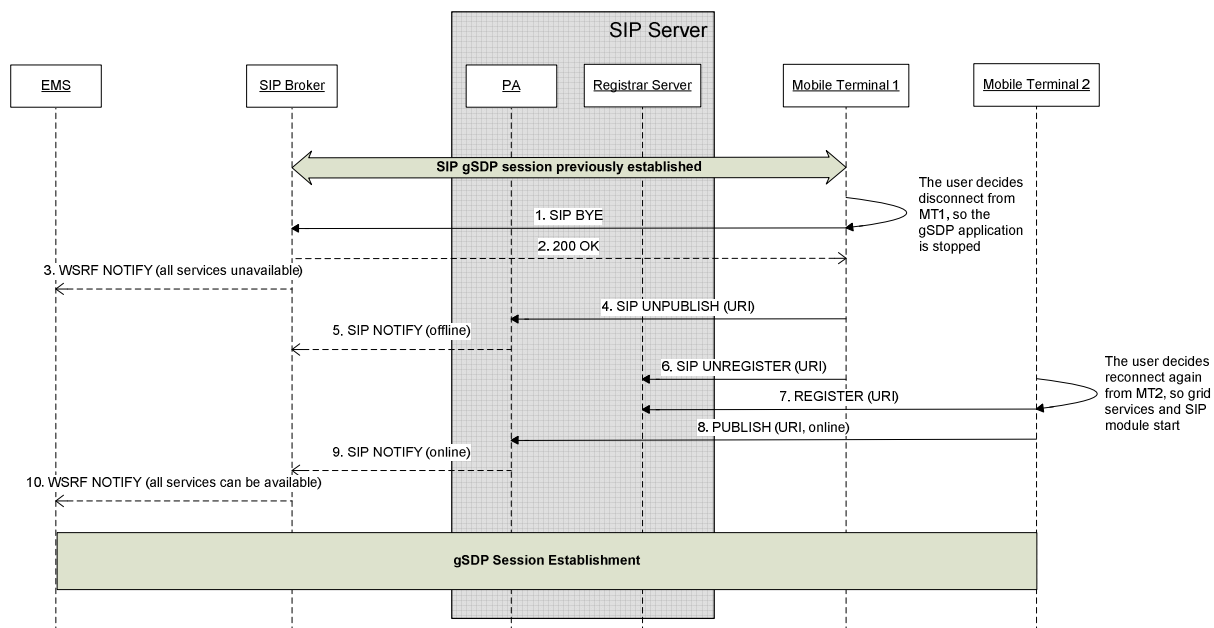


Figure 20 – User disconnection and reconnection from another location

Steps from 1 to 7 represent a controlled disconnection from certain location. From the point of view of the services SIPification, most relevant steps are:

- The SIP Broker informs the EMS about the gSDP session termination using a WS notification (3), indicating that all services running at the MT become unavailable. The format of the content of this notification is described in Annex D.2.
- The Presence Agent informs about changes on the presence status while the presence subscription done during the gSDP session setup phase remains active (5).

Then the SIP service URI is used to connect again from another location (steps from 7 on). Thanks to the active presence subscription, the SIP Broker detects that the presence status of the SIP service URI becomes online again (9) and it can inform on this to the EMS (10), which typically will request again for the service details at the new location.

Even if an accidental disconnection takes place, the EMS will be aware on this situation, since the SIP Broker will receive a SIP presence notification indicating that the presence status publication of the SIP service URI has expired. This event will be interpreted as an incidental disconnection and the EMS will be informed about the complete services unavailability at the corresponding URI.

Formally, the presence information is not needed, since it is used only as an indirect mechanism to detect a further reconnection (i.e. a SIP re-registration). The SIP Broker would be aware on the registration status of the SIP service URI using the method proposed in RFC3680 [RFC3680], also based on the SIP event framework but for the registration event itself.

Additionally, incidental disconnections could be prevented by monitoring the status of the established gSDP session, as described in RFC4028 [RFC4028].

The SIP session established will be used as the main mechanism to inform the EMS about changes on the MT grid services availability or WS URIs. So, if the information related to some service changes, the MT gSDP application will send a reINVITE containing the new gSDP session properties. The EMS will be informed on this through a WS notification, whose content indicates a change on the services using the format described in Annex D.2. This sequence is depicted in Figure 21.

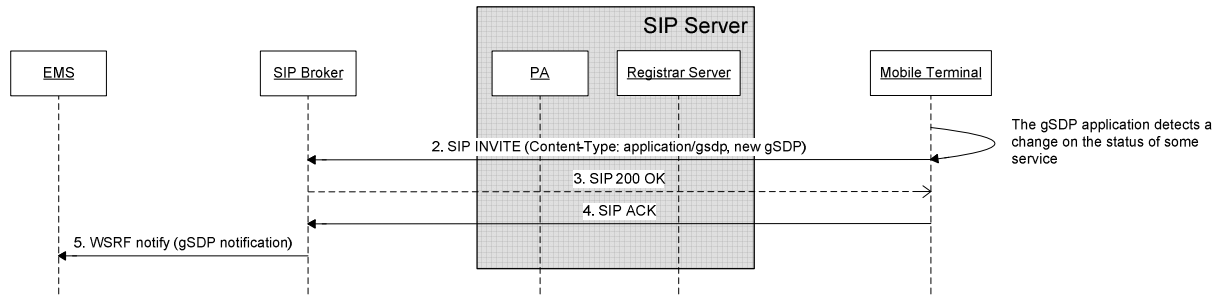


Figure 21 – Change on services information

When the information related to the services running on certain MT is no longer needed, the EMS can terminate an existing SIP gSDP session. In this case, the SIP presence subscription is also terminated. See next figure for more details.

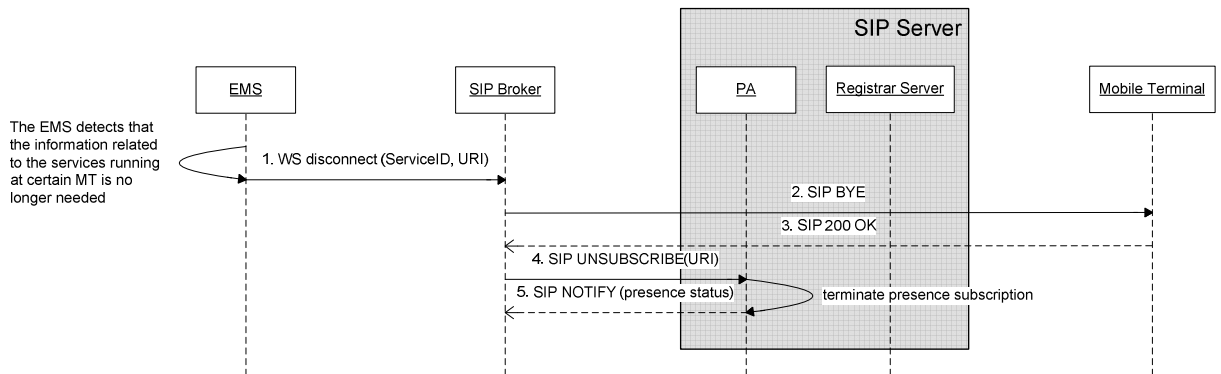


Figure 22 – gSDP session termination (from EMS)

5.4. Implementation

Main technologies being used for the SIP architecture development are Web Services, Java and C. In the MT all components have been developed in Java, while in the SIP Server C was used (SIP Server was developed using and extending SER).

The SIP Broker functionalities are exposed by means of the SOAP IF subcomponent (see Figure 17). It comprises two WS-Resources components: the SIP Broker WSRF service, which is invoked from the Workflow Manager or the EMS during their operations, and the SIP Broker Notification Producer, which is responsible for the managing of the WS-Notifications. See D4.4.3 [D443] and further updates for more details.

Other SIP Broker subcomponents (Broker Engine and SIP UA) were implemented with Java.

5.4.1. MT SIP Module

The MT SIP module has been developed in Java, and it has been designed to support multiple and simultaneous SIP applications and sessions running over it. Applications interact with the SIP Module process using Java RMI technology, but the way in which the SIP Module has been designed hides these details to the application developers.

Each application may be responsible for the handling of certain types of session, so they have to define both the content type of the sessions they are responsible for, and the number of permissible simultaneous sessions they can support: for example, the multimedia application manages “application/sdp” sessions while the gSDP application is responsible for the handling of “application/gsdp” sessions. In both cases, due to the nature of the sessions themselves, only one session is allowed.

When the SIP Module starts, it checks if the login application was started in order to make use of the current login information, which includes the SIP Address of Record, or main SIP URI (SIP AoR) to be used during the SIP registration. If yes, the SIP module is successfully registered; if not, a GUI is shown to request for the login data.

Once the SIP Module has been successfully started, SIP applications can be initiated. The final version of the SIP module could be configured to automatically start some applications after a successful registration, if required. In the other hand, since SIP applications cannot work if the SIP module is not running, all applications running on top will be notified when stopped.

The corresponding debian package install several .jar files that can be used to build an Akogrimo-compliant SIP application following certain rules, as will be described below.

The mapping between the functional components and their implementation is described in the table below.

Component	.jar file	Package
2.1 Sip Stack	JainSipApi1.1.jar nist-sip-1.2.jar	These libraries contain the JAIN SIP specification and RI, and a JAIN-compliant SIP Stack developed by NIST. They are used by the SIP UA.
2.2 SIP User Agent	SIPUA2.0.jar	org.akogrimo.sip.ua
2.3 Session Control Logic	sipmodule.jar	org.akogrimo.sip.sipmodule.sessioncontrol
2.5 Session Description Manager	sipmodule.jar	org.akogrimo.sip.sipmodule.sessiondescription
2.7 Session Data Manager	sipmodule.jar	org.akogrimo.sip.sipmodule.sessiondata
2.9 QoS IF module	sipmodule.jar	org.akogrimo.sip.sipmodule.qos
2.10 App IF module	sipmodule.jar	org.akogrimo.sip.sipmodule

Table 3 – MT SIP module implementation: libraries and packages

These libraries offer a middleware that can be used by the programmers to build a SIP-aware Akogrimo-compliant application, under certain preconditions:

1. The MT SIP application must extend from the abstract class SIPModuleConnector, (located in org.akogrimo.sip.sipmodule package), which provides an implementation of the key utilities

that all Akogrimo SIP applications need in order to make use of the SIP Module and a remote interface to it.

2. The MT SIP application must define its own listeners in order to receive notifications from the SIP UA, depending on the type of events the application is interested in, by implementing the appropriate interfaces. These interfaces (contained in `org.akogrimo.sip.module.event` package) are described in the table below.

Interface	Purpose
<code>IRegistrationListener</code>	Receive notifications about the registration status of the SIP module. Since the abstract class <code>SIPModuleConnector</code> implements it, the main class also do.
<code>ISessionListener</code>	Receive notifications within the context of an established session (from the session establishment to its termination: transfer requests, properties change requests...).
<code>ISessionRequestListener</code>	Receive session-related requests outside the scope of an established session (e.g. an incoming session requests have arrived, or a request to start a session).
<code>IEventListener</code>	Receive SIP-events framework related information (status of publication of subscriptions, or the corresponding notifications).

Table 4 – MT SIP module implementation: available interfaces

Additionally, and for session-oriented applications, the SIP module offers some utility classes to build the session parsers and the data handlers.

1. Each application must define how to parse its specific session description data from and to SIP message bodies. This can be done by extending from the generic parser base class, named `SessionDescriptionParser`.
2. As mentioned before, each application must define its own session data handler, which can be done by extending from the `SessionDataHandler` base class.

If these utility base classes are used, each application must define its own session object that implements the generic session interface, named `ISession`. This empty interface will be used in all interactions with the generic base classes; the application will be responsible to perform an appropriate casting from the generic type to the specific one, and vice versa.

There is also another utility class, named `SessionDataManager`, which provides some utilities to synchronously or asynchronously start/stop one or several data handlers, and how to configure them. The relevance of this utility consists on the fact that some class extending from `SessionDataManager` and implementing the mentioned `ISessionListener` interface comprises all the required functionality to handle a single session.

Next table summarises these classes and the SIP Module library package which contains them.

Class/Interface	Package (located at)
<code>ISession</code>	<code>org.akogrimo.sip.sipmodule.sessiondescription</code>

Class/Interface	Package (located at)
SessionDescriptionParser	org.akogrimo.sip.sipmodule.sessiondescription
SessionDataHandler	org.akogrimo.sip.sipmodule.sessiondata
SessionDataManager	org.akogrimo.sip.sipmodule.sessiondata

Table 5 – MT SIP module implementation: location of the utility classes

After that, the main function of the application must do the following:

3. Connect to the SIP Module. Once connected, the class extending from SIPModuleConnector contains a valid reference to the SIP Module (SIPModuleConnector.sipModule), and can perform a request to it.
4. Register the listener(s) at the SIP module, in order to receive notifications from it. Listeners are indexed by a string key, so typically a session-oriented application should register itself by using the ‘Content-Type’ it processes (e.g. “application/sdp” for multimedia, or “application/gsdp” for the sipification intermediary on the MT) and an event-oriented application should use the ‘Event’ it processes (e.g. “presence” or “sipsd”).

Only objects implementing ISessionRequestListener and IEventListener have to be registered. Note that the SIP Module allows specifying which session listener will be aware on the session related events, once established. So developers can decide on having a single object implementing both ISessionRequestListener and ISessionListener interfaces (which is appropriate for applications like gSDP, where only a single session will be established), or having one session request listener object handling all incoming requests, but multiple session listener objects. Each session listener takes care on one single session related events (this is the case of the multimedia application, since only one bidirectional communication can be established, but at the same time it can only receive video from another sources).

After these two steps, the bidirectional communication between the application and the SIP Module is setup.

The structure of the main function is depicted in Figure 23. Note that the base class SIPModuleConnector already provides the implementation of the required methods, so they are available to the MT SIP application subclasses.

```

public static void main (String[ ] args)
{
    myApp app = new myApp();

    1. // SIPModuleConnector implements connectToSIPModule, so available to myApp
    String strSIPModulePath = path_to_sipmodule.xml_file;
    app.connectToSIPModule(strSIPModulePath , "application/mySDP");

    .....
    mySessionRequestListener = new mySessionRequestListener();

    2. // Register listeners. SIPModuleConnector contains a ISIPModule object, so available
    // to myApp
    app.sipModule.addListener(mySessionRequestListener, "application/mySDP");

}

```

Figure 23 – Akogrimo MT SIP application. Main function structure

Just to mention that the connectToSIPModule method is responsible for creating the RMI connection between the application process and the SIP Module, but these details remain hidden to the application developers. This makes Akogrimo SIP applications clearer and simpler, by avoiding the developers to know the concrete details of the RMI connections.

5.4.1.1. MT gSDP application

As mentioned earlier, a specialised SIP session-oriented application has been developed to act in behalf of the grid services running at the MTs. The first version was just a proof-of-concept prototype oriented to demonstrate the feasibility of the Akogrimo SIP-SOAP synergies approach, so not all the required functionality has been included; however, missing features have been included in the final version. Next table shows a comparison between the main characteristics of the first prototype and the final version:

Current prototype	Final version
Started by the SSO login application after the SIP Module	Started by the SIP Module itself when successfully logged in
Interface with the grid services simulated; the services information is read from a preloaded xml file	Real interface with the grid services, which will be able to use the gSDP application. Additional GUI for admin. and testing purposes.
Console application to simulate services input	

Table 6 – gSDP application 1st prototype vs. final version

The mapping between the installed functional components and their implementation is described in the table below.

Component	.jar file	Package
gSDP application	gSDPApp.jar	org.akogrimo.sip.gsdppapp

Table 7 – MT gSDP application implementation: libraries and packages

The corresponding debian package installs a shell script (gsdp.sh) which implements the real interface with the grid services (and also allows handling the gSDP application from the GUI),

and an xml file (services.xml) with the services information. The format of this xml file is identical to the gSDP session specification defined in Annex D.1.

The real services can interact with the gSDP application by invoking `gsdp.sh` in the way described in the table below:

Method	Purpose
<code>gsdp register serviceName serviceURI availability</code> Parameters: String serviceName String serviceURI String availability (“available” or “unavailable”)	Register a new service and its availability.
<code>gsdp update serviceName serviceURI availability</code> Parameters: String serviceName String serviceURI (empty value = no update) String availability (empty value = no update)	Updates a registered service URI and/or availability

Table 8 – MT gSDP application implementation: real services interface

5.4.2. SIP Server

The whole SIP Server has been built using SIP Express Router (SER) implementation as the basis [SER]. SER is a powerful, easily configurable, C-written and free SIP server implementation. SER can be easily configured to acts as a proxy, registrar or redirect server. Most relevant characteristics of SER are:

- **Speed:** SER is able to process thousands of calls per second even on low-cost platforms.
- **Extensibility:** using SER it is possible to link new C code to redefine, or extend, current logic. New code, which can be developed independently on SER core, is linked to it in run time. In this way, new functionality is included by adding SER modules (*.so libraries) keeping the SER core really fast and compact. This has been the adopted solution to add the interface modules with the A4C and the PBNMS.
- **Flexibility:** routing decisions and SIP method processing can be modified by means of textual scripts that administrators may write. By modifying the main script (named `ser.cfg`) it is possible to control and modify lot of parameters (e.g. expiration time of registration requests) and introduce new logic (loading new modules, and invoking exported functions when necessary).
- **Portability:** written in ANSI C, and deeply tested on PC/Linux, Sun/Solaris platforms.
- **Interoperability:** SER is based in open SIP standards. SER core implements basically RFC 3261.
- **Small size:** core footprint is 300KB, add-on modules take up to 630KB.

As mentioned before, SER has been built to be configured using a configuration file, named `ser.cfg`. It is a C-Shell script which defines how incoming SIP requests are processed. During the server startup, this file is translated into an internal binary representation, because the direct interpretation of this file would make SER very slow. Considering the functional description

provided in D4.1.1 [D411], the main configuration file and the main part of the SER core implements both the SIP Proxy and the SIP Broker Engine modules. The appropriate manipulation of this file enables the delivery of incoming SIP requests to the specific module that implements a more concrete functionality, like the Registrar or the A4C interface. Annex B includes the SER configuration file (ser.cfg) used for the demo.

Extensions to implement the required interfaces are described in next subsections.

5.4.2.1. Akogrimo A4C IF SER module

The Akogrimo A4C IF sub-module has been implemented as a SER extension. The resulting dynamic C library is named a4c.so.

The basic functionality of this module is to check and validate the A4C Token included in a SIP request, acting as a filter for the corresponding SIP transaction. In order to validate the token, the C library makes use of the A4C C++ client library. If the validation finishes with success, such transaction is allowed. D4.1.1 [D411] provides detailed information on how this mechanism works.

Next table summarises the new functionality that this library exports, which have to be invoked from the ser.cfg configuration file in order to validate the corresponding transaction. Annex B shows a usage example for the validation of a SIP REGISTER.

Table 9 – a4c.so exported functions

Function	Description	Parameters
aa_request	This function checks and validates the A4C token included in a SIP transaction	None when invoked from ser.cfg file ¹

The library offers some configurable parameters. Most significant are:

- The detail level of the logs (for debugging purposes a traces mechanism – independent from the SER traces framework – has been included. This allows the library debugging without “interferences” from the rest of the SER code).
- The name of the non-standard SIP header field in which the A4C token will be included.
- The executable command for validation (a wrapper for the A4C C++ client library to be invoked from the a4c.so code).

5.4.2.2. Akogrimo PBNMS IF SER module

As the PBNM section describes (section 6), policies related to SIP proxies or the SIP Server may also exist. The PBNMS IF sub-module has been implemented as a SER extension which communicates with the PBNM Server in order to get policies for some relevant SIP Server interactions.

¹ The SER framework passes implicitly the incoming SIP message structure to the function.

5.4.3. SIP Broker

The SIP Broker is a statefull Web Service (WS-Resource) deployed on an application server which exposes SIP-based functionality. To be precise, in the prototype version two functionalities were offered:

- Grid-initiated SIP Call, which enables the possibility to arrange a grid call between two SIP users.
- Grid-initiated SIP transfer, which enables the possibility to inform a user he/she can move an ongoing session to a most suitable device.

Other functionalities (e. g. a WS interface to expose the SIP SD facilities) are being considered to be included in next versions. A detailed description of the offered interface is described in section 3.3.3.

The internal structure of the SIP Broker comprises four differentiated sub modules, following the functional description provided in Figure 17:

- The SOAP interface, which accepts incoming WS requests from the “grid world”: to be precise, from the Workflow Manager and EMS.
- The A4C IF, an embedded overlay of the A4C Java client used to request the A4C Server for the generic user profile (GUP) of the users to be contacted. The GUP contains (apart from other information) the SIP AoR of the user, which is crucial for all SIP interactions.
- The Broker Engine. This Java component is the core of the SIP Broker. Its main task is to convert the WS request parameters (typically the Akogrimo IDs of the user to be put in contact, or the device URI the user is allowed to move to) into a suitable format that the SIP UA can understand. This is made through the facilities provided by the A4C IF submodule. For efficiency purposes, and considering that only one SIP UA and one A4C client are needed, it has been implemented as a separate Java process that can be accessed through Java RMI.
- The SIP UA, which handles the SIP process itself. It has been implemented as a Java library based on JAIN SIP that other components can use.

The mapping between the functional components and their implementation is described in the table below.

Component	.jar file	Package
3.1 SIP UA	JainSipApi1.1.jar nist-sip-1.2.jar	These libraries contain the JAIN SIP specification and RI, and a JAIN-compliant SIP Stack developed by NIST. They are used by the SIP UA.
	SipUA2.0.jar	org.akogrimo.sip.ua
3.2 Broker Engine	SIPBrokerEngine.jar	org.akogrimo.sip.broker.brokerengine
3.3 SOAP IF	sgg.war (deployed on Tomcat5.0.X)	akogrimo.sipBroker
3.4 A4C IF	liba4cclient.jar	This library (UniZh) contains several utility packages that are used by the Broker Engine.

Table 10 – SIP Broker implementation: libraries and packages

Most of the parameters of the SIP Broker can be configured through an xml file (named sipbroker.xml) which is installed together with the mentioned libraries. Parameters that can be configured using this file are:

- SIP Broker Engine RMI port and policy file.
- SIP UA related parameters (IP address, port, transport...).
- SIP proxy server parameters (IP address and port).
- A4C related parameters (A4C client library location, A4C Server IP address...).

The SOAP IF component receives the request of arranging a SIP Call or transferring a session from one device to other. The SOAP-IF then translates the SOAP request to a SIP request. The invocation of the corresponding SIP Broker Engine is made via RMI technology. The SOAP IF implements a client whose URL contact is stored on a properties file that belongs to the SOAP IF component.

This component has been deployed using GT4 java WS container as WS-Resources. In our case java WS-container has been used to host the WS-Resources interfacing SIP Broker with Grid services. See D4.4.3 [D443] and further updates for more details.

6. PBNM

6.1. Introduction

Policy Based Network Management (PBNM) goal is to help bridge part of the conceptual gap between a human policy maker and a network element that is configured to enforce the policy. When a human business executive defines network policy, it is usually done using informal business terms and language. For example, a human may come with a policy statement that reads: “traffic generated by our human-resources application should have higher priority for making it through to its destination compared to traffic generated by people browsing the WEB during their lunch breaks” (abstract policy). While this statement clearly defines QoS policy for the network, the network itself cannot enforce it. Translation to “network terms and language” (concrete configuration) is required.

Clearly this wide gap implies several translation levels, from the abstract to the concrete. Two levels will be analysed here: CIM ([CIM]) closer to human natural language and used in many Akogrimo systems – Web Services will use WSDL ([WSDL])- and COPS ([RFC2748]) (and its extension [RFC3084]) closer to configuration commands and used in QoS System.

6.2. Architecture

Figure 24 depicts the PBM model that can be followed. It is divided in PDPs and PEPs (Policy Decision/Enforcement Points). PDPs take decisions and control and configure the PEPs who enforce the policies. PDPs and PEPs may communicate using e.g. COPS or DIAMETER protocols. The actual protocol depends on the system considered: COPS for QoS (see section 4.4), DIAMETER for A4C. PDPs can gather the policies from a PBNM Server (PBNMS). PDPs should communicate with the PBNMS all using the same protocol e.g. CIM (XML based) over HTTP. PBNMS will store the policies (e.g. in a LDAP repository) and offer a Graphical User Interface (GUI) for the manager to introduce the policies. Policy conflict is detected in the PBNMS.

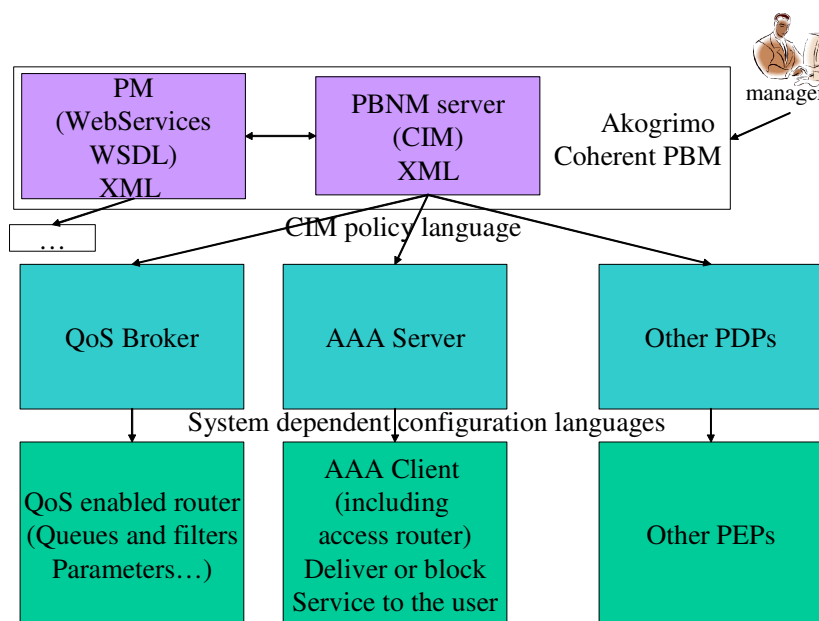


Figure 24 – PBM Architecture. Relation with other components

6.3. Sample scenarios

PBNM covers a wide range of scenarios and situations. For instance we have “policy conflict” cases. This is better explained by an example. We have two policies: policy 1 “users under 18 years old, have a 50% discount in MMS between 20:00 and 8:00” and policy 2 “gold profile users have a 60% discount in MMS between 21:00 and 9:00”. If a user is gold and under 18, which policy should be applied ? Mechanisms handling these conflicts must be designed, for instance assigning priorities to policies, or policy “set-union” rules.

Other scenarios, more related to QoS, may be as following. Let’s consider the following policy:

“On any interface to which these rules apply, guarantee at least 30% of the interface bandwidth to transport QoS class conversational, and at least 40% of the interface bandwidth to transport QoS class transactional.”

Policies in the system level can be grouped in *PolicySet* CIM class. This class can also contain information about the roles of the entities to which these policies apply. In our example the desired role is an interface, because we want to allocate for QoS Classes an amount of bandwidth available through the interface. The resources assigned to specific roles are described in *PolicyRoleCollection* CIM class.

In the following scenario another aspect of PBNM system occurs. Let’s consider the following policy:

“If the EF traffic load between A and B exceeds 70% of the allocated bandwidth, allocate more bandwidth to the EF traffic from other classes.” (1)

This is an example of dynamically applied (reconfiguration) policy. In order to initiate this policy, the QoS system (namely the QoSBroker) needs to know from where and how to get more bandwidth. It means that in PBNM system other policies (describing any conditions) must have been defined earlier and sent to the QoSBroker. They can be configuration or reconfiguration policies.

Firstly, QoSBroker must have information from the measurement system (this system can be the same as the QoS System) about the actual situation in network. This implies that a policy should be supported by the measurement system. This policy could be:

“Provide measurements of the EF traffic between A and B every 10 minutes and send the trigger to QoS Broker if it exceeds 70% of the allocated bandwidth”. (2)

Additionally, another policy defining how to get more bandwidth for EF traffic load is required. It can be defined in the following way:

“If the AF traffic load is under 40% you can take up to 30% of its unused bandwidth.” (3)

Abovementioned policy is a typical network view level policy. It seems simple but in fact it engages a lot of aspects of network management. In order to enforce the policy it must be translated into lower levels of abstraction. It also interacts with other policies on the same (network) level.

Policies related to SIP proxies may also exist. One may think in a policy that reads:

“If the user profile is not “high priority” and the network load is above 50% then, in any audio-video call, suppress the video codecs and force to make an audio-only call. (1)

This policy involves others systems besides the SIP proxy, namely the QoSBroker and the A4C system. As in the example above, policies should be defined for these systems. For instance, for the QoSBroker one may define a policy such as:

“If the network load is above 50% then on any reservation request prevent the SIP Proxy about such a status and send an alarm to the SIP Proxy.(2)

6.4. Documents, standards, protocols & interfaces

The communication between PDPs and PEPs is well defined in standards. In our PBM model this communication can be different between the several systems, the coherence is achieved in the PBNM Server level. For QoS the PDP-PEP communication is based on COPS-PR [RFC3084] and the data representation on PIBs [RFC3571].

At the abstract end of the policies (closer to human language level) is CIM. CIM [CIM] is defined by DMTF [DMTF]. CIM facilitates a formal representation of network business rules, thus providing the first concretization level: formally representing humanly expressed policy. CIM is based on UML (Microsoft Visio files are available) and XML representation exists. CIM policies are based on Policy conditions (if ...) and actions (then ...). CIM covers a wide range of policies. Figure 25 represents an example of QoS policies definition. Due to the innovative nature of Akogrimo some new policies have been defined.

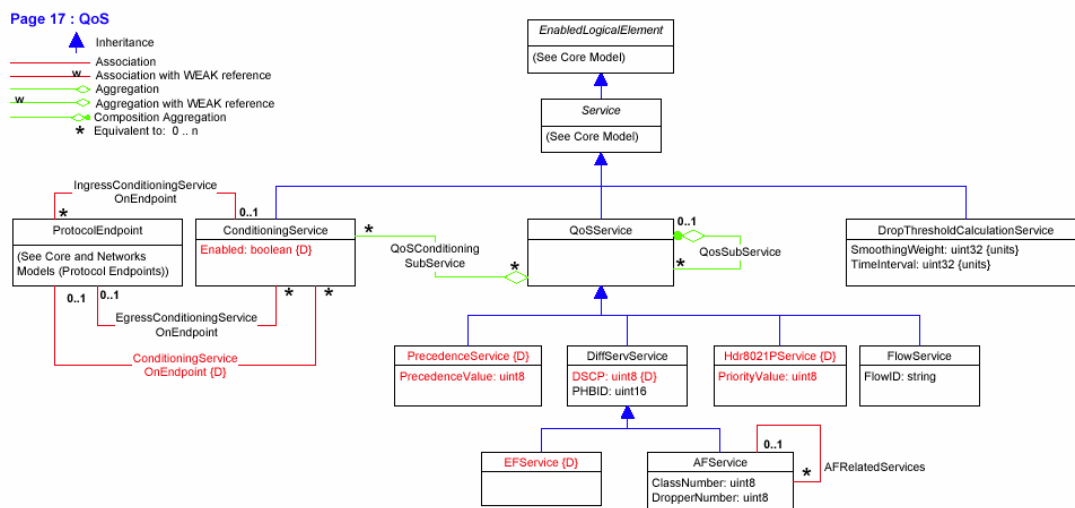


Figure 25 – A declaration of a QoS policy in CIM

As we said CIM is based on UML but XML representation exists. Figure 26 shows the QoS Service Class present in Figure 25 but defined in XML. The communication between the PBNM Server and the PDPs is based on this XML representation which is transported over HTTP.

```
<?xml version="1.0" ?>
<CIM CIMVERSION="2.2" DTDVERSION="2.1">
<DECLARATION>
<DECLGROUP>
<VALUE.OBJECT>
<CLASS SUPERCLASS="CIM_Service" NAME="CIM_QoSService">
<QUALIFIER TOSUBCLASS="false" TRANSLATABLE="true" NAME="Version" TYPE="string">
<VALUE>2.7.0</VALUE>
</QUALIFIER>
<QUALIFIER TRANSLATABLE="true" NAME="Description" TYPE="string">
<VALUE>This is a concrete class that represents ....</VALUE>
</QUALIFIER>
</CLASS>
</VALUE.OBJECT>
```

```
</DECLGROUP>
</DECLARATION>
</CIM>
```

Figure 26 – Representation of a CIM class in XML

We recall that for Web Services and service management [WSDL] will be the high (closer to human natural language) level language.

6.4.1. Example: mapping policies to CIM

We should see in this section, how one of the policies presented in the scenarios in section 6.3 can be mapped into CIM classes. Let’s recall the policy:

“On any interface to which these rules apply, guarantee at least 30% of the interface bandwidth to transport QoS class conversational, and at least 40% of the interface bandwidth to transport QoS class transactional.”

The proposed mapping of this policy for different level of abstraction is presented in Table 11

Levels		Sample Objective	Sample objects	Relevant model (see 6.4)
Domain	System	On any interface to which these rules apply, guarantee at least 30% of the interface bandwidth to transport QoS class Conversational, and at least 40% of the interface bandwidth to transport QoS class Transactional.	Conversational Transport Class	CIM (PCIM, PCIME, QPIM)
	Network	If (DSCP value is 01) THEN guarantee 30% of available BW If (DSCP value is 02) THEN guarantee 40% of available BW	Conversational DiffServ Service	CIM (PCIM, PCIME, QPIM)
Device	Device	A possible implementation of these rules within a PEP (router in this case) would be to use a Weighted-Round-Robin scheduler with 3 queues. The first queue would be used for DSCP01 traffic, the second queue for DSCP02 traffic and the third queue for the rest of the traffic. The weights of the Weighted-Round-Robin scheduler would be 30% for the first queue, 40% for the second queue and 30% for the last queue.	Routers, Interfaces, Queues	CIM (PCIM, PCIME, QDDIM)
	Instance	Specific commands configuring the queues of the devices.	Objects representing PIBs	PIB, COPS

Table 11 – Illustration of a process of mapping a policy from system level to instance level

6.5. Implementation

PBNMS-PDPs communications we will use CIM and the OpenWBEM [OPENWBEM] tool. It is free source code and not based on XML but on Managed Object Format (MOF) , but this should cause no problems. Indeed MOF is also an accepted representation of CIM (Figure 27).

```
// Copyright (c) 2005 DMTF. All rights reserved.
// =====
// CIM_QoSService
// =====
[Version ("2.7.0"), Description (
    "This is a concrete class that represents...")]
class CIM_QoSService : CIM_Service {
};
```

Figure 27 – Representation of a CIM class in MOF

Note that other open source tools for managing PBM CIM framework are available, such as OpenPegasus [OPENPEGASUS]. OpenPegasus and OpenWBEM offer very similar characteristics, indeed both could be employed.

The interface between the PBNMS and the management entities (PDPs) can be implemented as a CIM OpenWBEM provider running in the PBNMS. Currently this provider main functionality is to handle Akogrimo policy extensions and, based on the type of policy considered, “forward” it to the appropriate PDP(s). HTTP and CIM-XML (over IPv6) are used for this communication, see for instance Figure 28, where the PBNMS address is 2001:...:a5df and the PDP’s (a QoS Broker) one is 2001:...:693d.

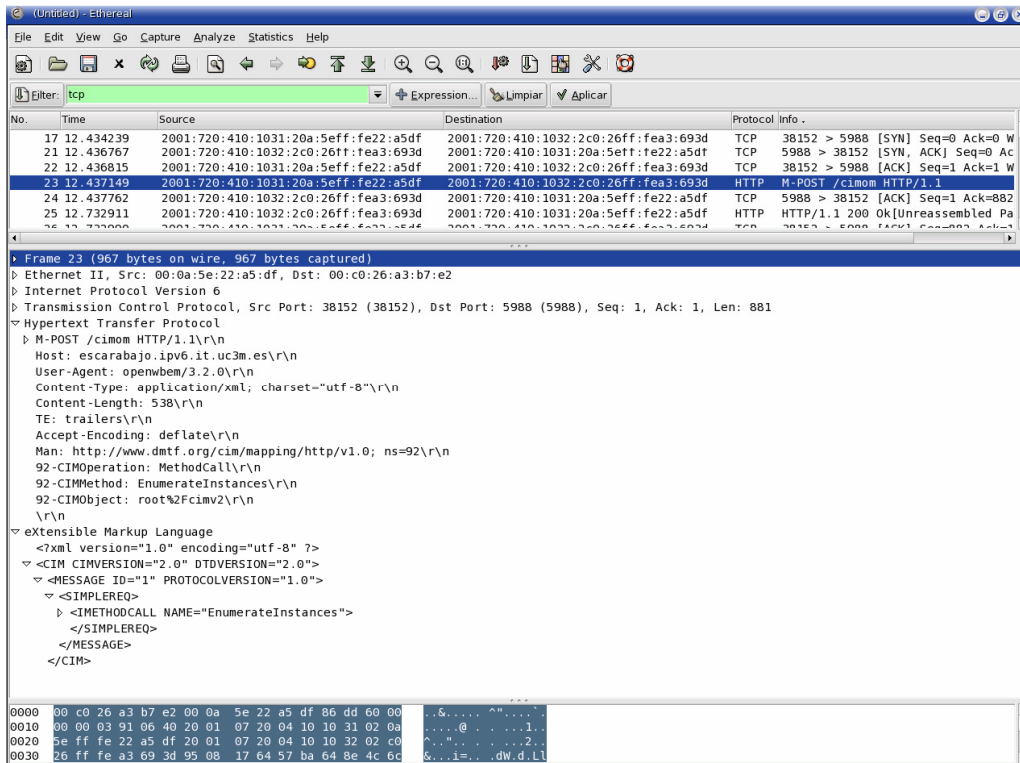


Figure 28 – Communication between the PBNM Server and a PDP

OpenWBEM provider implementation is based on a CIMOM (CIM Object Manager) loadable library, a UNIX .so file, completely controlled by the CIMOM. OpenWBEM proposes a remote provider concept: remote providers use a light-weight CIMOM implemented in the managed entity (the PDP) that, making use of a loadable library, implements the provider interfaces to the managed entity. In the managed entity (the PDP) the provider is in charge of translating the CIM policy to PDP understandable policies. Figure 29 illustrates the remote provider concept.

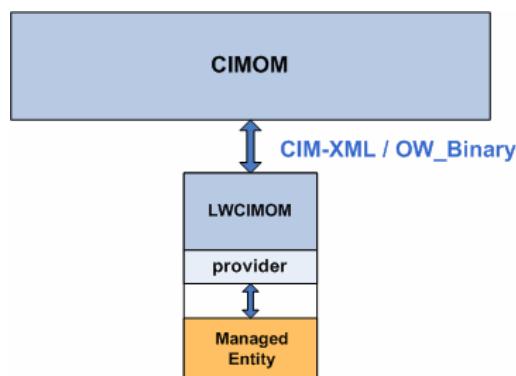


Figure 29 – CIM Remote Providers

The communication between both CIMOMs is performed in CIM-XML over HTTP. The light-weight CIMOM interfaces the managed entity. This is exploited in the implementation of some units. OpenWBEM CIMOM is able to store policies on a LDAP repository and offers a GUI (Figure 30) for controlling them. We can check there the Akogrimo Policy extensions (termed D_***) that will be handled by the implemented provider in the PBNM Server and sent to the appropriate provider(s) in the PDP(s).

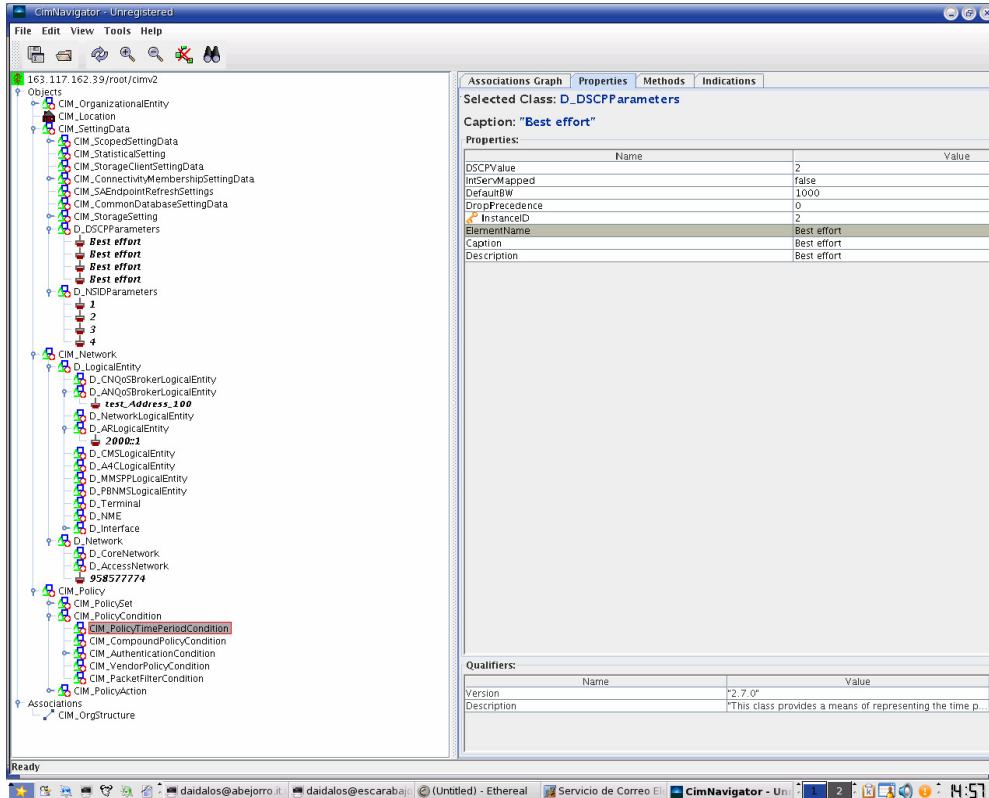


Figure 30 – GUI for the CIMOM (PBNM Server)

In the QoS Broker, its provider will “translate” the policies into a mysql database understandable by the QoS Broker. The interface between the provider in the QoS Broker and the PBNMS’ Provider is bidirectional: the QoS brokers will request the PBNMS for a configuration at startup time, and along the time it will send alarms/notifications to the PBNMS; the PBNM Server (PBNMS) will send a policy definition whenever it is (re)defined by the human operator or by automatic application triggers.

The QoS Broker configuration information may include the network topology description, the QoS description of the services offered by the operator, and the initial resource distribution that must be defined in its domain. The services available are described by the PBNMS in terms of the DSCP code, the required resources both for the downlink and the uplink flows. It is very important for the operator to have a central network entity defining, in an integrated way, the service QoS parameters. A central service parameter definition avoids network configuration inconsistencies as well as it makes the configuration job much easier. Furthermore, when a service redefinition occurs the PBNMS performs an almost instantaneous QoS entities configuration. Figure 31 illustrates the QoS Broker policy database.

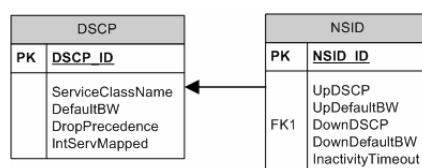


Figure 31 – QoSBroker Policy database

The QoS network configuration is performed accordingly to the network usage. The broker can also use historical network usage information namely at start-up. The initial resource distribution is sent by the PBNMS. Several CIM classes have been developed in order to extend CIM model in order to support the QoS Broker model.

7. Security

7.1. Introduction

Security tackles a wide range of different fields and one of its characteristics is that is something we can't get enough of. No matter how many security mechanisms are provided and how strict they are there is always a something left out. The reason being that security is a trade off with usability. Usually the more secure a system is, the less usable it is. The objective is always to provide a fairly secure system optimizing usability. There is a need to define how far one would like to go in terms of security and draw a line delimiting an acceptable level of security that is considered more than necessary for the purposes aimed. For instance, a user could be requested to be authenticated in order to make a transaction of a considerable amount of money, but just be requested to show a token in case of a small amount purchase, making it less secure but faster.

7.2. Attack model

In order to be able to deliver a consistent security architecture, first the targets and severity of attacks that can be performed need to be identified in relation with their impact on the architecture.

		Attacker →						
		User	Terminal	OpVO	BaseVO	Customer Domain	Network Provider	Service Provider
Target ↓	User	4/4/4 relevant but out of scope	4/2/2	1/2/1	2/3/2	4/2/2	4/3/3 Not specific to Akogrimo	1/1/1 Accounting; SLA
	Terminal	4/3/2	4/3/2	4/3/2	4/3/2	4/3/2	4/3/2	4/3/2
	OpVO	1/1/1	1/1/1	1/2/1	1/3/1	1/2/1	4/4/4	1/3/1
	BaseVO	1/1/1	1/1/1	1/2/1	4/4/4	1/2/1	2/3/2	1/2/1
	Customer Domain	4/1/1	2/2/2	1/3/1	1/3/1	4/3/3	4/3/2 Not specific to Akogrimo	1/3/1
	Network Provider	4/3/3	4/2/2	2/3/2	2/3/2	4/4/4	4/3/2 Not specific to Akogrimo	4/4/4
	Service Provider	2/2/2	3/3/3	1/1/1	1/3/1	1/2/1	1/3/1	1/3/1

Relevance in Ako/Likelihood in Ako/Impact
1 high - 4 low

Table 12 – Attack model

Table 12 shows the attack relations between the main Akogrimo actors. The most relevant attacks fields are coloured in red whereas secondary ones are coloured in green; the rest of the fields are considered out of scope and left white.

The three numbers in each square are an estimate of the relevance, likelihood and impact to Akogrimo of a particular class of attacks. As an example, a user attacking a user is certainly a

possible scenario, but it is one particular scenario which has been dealt with before and which occurs in practically any environment or project. Thus, those type of attacks were considered not relevant to Akogrimo. On the other hand, a user attacking an OpVO is considered highly relevant because it is a new kind of attack that occurs because of Akogrimo new infrastructure.

From this table one can infer what fields need to be tackled in WP4.1 in terms of security. The provisioning of security in WP4.1 will therefore be aligned to the following directions:

- Provisioning of security aligned to the research done in the project and therefore analyzing in detail the security issues arisen within the context of the new functionalities developed (e.g. SIP with SOAP interactivity). From the table one can easily infer that this is the issue with more preference, as most fields related to OpVO and BaseVO are considered of high priority. Regarding network attacks, although direct attacks to the infrastructure are not considered as important, one has to keep in mind that the network is the first contact the user has with the system, and therefore its first line of defence. E.g. users are authenticated at the network layer and afterwards authorization is performed that understands the former authentication as true. A lot of higher layer attacks can be triggered from not having a secure network substrate.
- Provisioning of security as a whole, in order to provide guidelines for the security of the complete platform developed. (e.g. provide authentication and Single Sign On (SSO)) and avoid the introduction of insecurities that come to the hand of the technologies used (e.g. weaknesses introduced by the use of MIPv6). In such an architecture cross-layer security becomes a must.
- Provision of security on the tools used for implementation. (e.g. Java stubs). This point is of lower priority as it is not Akogrimo specific.

In very general terms, the security mechanisms deployed should strive to cover the following issues:

- Provide privacy
- Avoid impersonation, man-in-the-middle, replay attack by means of authentication
- Establish a robust authorisation system.
- Provide integrity of network messages.
- Avoid DoS, DDoS attacks
- Provide non-repudiation and attacks using false information or internal attacks by means of auditing.

7.3. Network layer attack scenarios

This chapter lists relevant attacks to the Network substrate. Some of them are more relevant than others in the Akogrimo context and therefore will be treated in more detail. It is worth mentioning, that this study of attack scenarios does not intend to be exhaustive, it tries to give the reader an overview of possible attacks to ease the identification of security breaches. Security can not be scenario driven, because the possibilities are infinite, but a description of possible attacks and their importance is a good exercise in order to design a more generic security architecture. The corresponding defences explained in the next section not only cover the presented attack scenarios but also are intended to provided defences to attacks that might not be envisaged.

7.3.1. Mobile IPv6

The use of MIPv6 introduces new insecurities over regular IPv6 that need to be palliated. Among them we find: MN impersonation, HA impersonation, MiM or DoS. They all exploit the fact that the mobile terminal needs to update its position when doing a handover, by manipulating, blocking this information or sending false frames.

7.3.2. Attacks to Access Router

The threats that could suffer the Access Router in the Access Network could be divided in two sections, external and internal. External refers to the AR's interface to the access network and internal refers to the interface to the core network.

7.3.2.1. External attacks

The Access Router, being the network component that connects the users with the core of the network, is a prime target for attacks from the outside of the network. Most of these attacks fall outside the scope of the Akogrimo project. They are typically well understood and some counter measures can be taken in order to minimize their impact.

The most important class of attacks are Denial of Service (DoS) attacks. They are also the most difficult to defend from since they are typically brute-force attacks that aim to spend all the resources of the target, such as CPU, memory or network bandwidth.

Distributed DoS attacks are basically the same as DoS attacks, main difference is in the scale. In DoS attacks an attacker or several attackers working in close cooperation perform the attack using few machines. In a DDoS attack, the attacker or attackers have third party machines at their disposal, often through viruses or worms, and can use those machines to perform the real attack.

7.3.2.1.1. Low level attacks

One major issue wireless networks have is that the signal travels through an uncontrolled physical medium – air. Any attacker can use special devices to fill up the spectrum used by wireless networks with a random signal that makes no sense whatsoever. Depending on the strength of the attacker's signal and its position relative to the components communicating in the Akogrimo network, this kind of attack can lower significantly the throughput of the network or even completely stop any communication whatsoever.

There is currently no way to defend against these attacks.

7.3.2.1.2. Denial of Service attacks

In this category are the attacks made with the objective of using up the AR's resources. These attacks are typically performed by sending a large amount of network packets to the target machine. These attacks can cause problems in two different areas:

- Bandwidth consumption – the attack uses network bandwidth which will obviously not be available for legitimate users
- CPU/Memory consumption – sophisticated attacks can send packets which are carefully crafted to provoke a bigger consumption of CPU time or system memory than ordinary packets

Not very sophisticated DoS attacks, that target mainly the network bandwidth, can be successful with a large number of attacking machines just by exhausting the bandwidth available for

legitimate users. Although simple and not requiring a lot of knowledge from the attacker, this kind of attack can be very effective and disrupt the network.

A more sophisticated approach can be undertaken by attackers with a greater knowledge of network protocols, or the internal workings of the Akogrimo network.

In the previous attack described above, the content of the packets sent to the network is not very important since the attack's main objective is to use all the available bandwidth. Those packets will be dropped by the AR with minimal processing performed on them. However the attacker may produce packets specifically designed with the purpose of maximizing the processing that must be done. These attacks will typically exhaust either CPU or memory or even both resources faster than the previous kind of attacks. Thus, although a much higher knowledge of the system being attacked is necessary, fewer machines will be needed to perform the attack.

7.3.2.1.3. Impersonation attacks

In this class of attacks the attacker tries to trick the target into believing that the attacker is some other person he is not.

Most of these attacks involve “spoofing” some kind of data in order to masquerade as another person. A simple example is spoofing IP packets. An attacker may be watching legitimate communication between the AR and a legitimate user, and then construct IP packets with that user's IP address and possibly also MAC address. Without any other security mechanisms in place, the attacker would effectively be seen as the legitimate user by the system.

Although in practice things would be more complicated and would require a significant degree of knowledge, time and patience to fully explore this technique, it is imperative that some kind of defence against these attacks is in place.

7.3.2.2. Internal Attacks

Internal attacks are considered to be attacks originating from within the core network. Attacks originating from the core network would typically involve some rogue component masquerading as some other legitimate component or being used as a platform for DoS attacks or attempting to break in to other components. Some of the possible attacks are:

- Malicious QoS Broker sends wrong rules, for benefiting or hurting a user
- Malicious A4C server can send fake user information
- Malicious QoS Broker can send fake rules; can benefit a certain user, giving him QoS level which he would not get normally
- Any component can cause DoS attacks
- Any component can be used as a basis to launch further attempts of break in to other components

The AR has a few basic safety measures, such as requiring a configuration for each of the components that are able to communicate with it. That coupled with authentication of the components between them reduces risks considerably, as a bad misuse or attack can more easily be attributed to a concrete component and therefore restrict its access or take any other measure against it.

7.3.3. QoS Broker

The QoS Broker receives requests from the Access Routers and the EMS Web Service. With the user's profile information it gets from the A4C Server, and also information about the current network status, it decides if a user is allowed a network flow at a given QoS level.

7.3.3.1. External attacks

Although the QoS Broker is not directly exposed to the access networks, an attacker can provoke some form of attacks indirectly.

An authenticated user can provoke an artificially high rate of requests to the QoS Broker. Multiple users in cooperation can effectively perform a DoS attack on a QoS Broker.

Since the AR gets rules installed when a new flow is requested, this will only cause problems if the user starts a large number of different flows. With typical Akogrimo software this is not possible, so the user's MT would have to have been compromised and special software installed on it.

7.3.3.2. Internal attacks

Like the AR, the QoS Broker is also subject to attacks originating from the core network:

- Malicious A4C server can send fake user profile
- Malicious AR can
 - use any kind of data collected from innocent users
 - Perform a DoS attack on the QoSB
- Malicious EMS can also benefit a certain user; can cause repeated QoS requests
- Any component can cause DoS attacks

As in the AR, the QoS Broker has basic safety measures that coupled with authentication of the components between them reduces these risks considerably.

7.3.4. PBNM

The PBNM is mostly sensible to internal attacks performed by another Akogrimo component such as:

- PBNM manager impersonation
- Fake Profiles obtained
- Policies not enforced correctly

7.3.5. Attacks to SIP signalling

The SIP entities that are involved in the Akogrimo SIP infrastructure are the Mobile Terminal, the Proxy SIP (in the Access Network) and finally the SIP Server and the SIP Broker in the Core Network.

The possible attacks that could suffer each one of these SIP entities could be the following.

7.3.5.1. **SIP with SOAP security problem**

The SIP with SOAP interaction in Akogrimo takes place in the component called “SIP Broker”, which basically exposes some SIP functionalities as a WS (typically invoked by the Workflow Manager and the EMS). To be precise, in the prototype version two functionalities were offered:

- Grid-initiated SIP Call, which enables the possibility to arrange a grid call between two SIP users.
- Grid-initiated SIP transfer, which enables the possibility to inform a user he/she can move an ongoing session to a most suitable device.
- Request for information about services running on a mobile or nomadic computer.

Other functionalities (e.g., a WS interface to expose the SIP SD facilities) could be considered by other projects that adopt the Akogrimo’s SIP with SOAP approach, so we can consider that the this approach in Akogrimo mainly relies on the capabilities that this component offers.

The internal structure of the SIP Broker consists mainly on three differentiated main submodules:

- The **SOAP interface**, which accepts incoming WS requests from the “grid world”.
- The **Broker Engine**. This component is the core of the SIP Broker. Its main task is to convert the WS request parameters (typically the Akogrimo IDs of the user to be put in contact, or the device URI the user is allowed to move to) into a suitable format that the SIP UA can understand. This is made through an A4CServer’s user profile request (using an embedded A4C client), that returns – apart from other information - the SIP AoR of the user. For efficiency purposes, and considering that only one SIP UA and one A4C client are needed, it has been implemented as a separate Java process that can be accessed through Java RMI.
- The **SIP UA**, which handles the SIP process itself. It has been implemented as a Java library that other components can use.

In order to avoid that some non-authorized entity can make use of the facilities that the SIP Broker can provide, security mechanisms must be implemented to protect each possible access path to this component. In next section both the authorized way to use the SIP Broker and the possible attacks are described.

Apart from attacks, another crucial consideration must be taken into account. The currently implemented interactions (grid triggered SIP calls and SIP session transfers) take place within the context of a certain VO, i.e. some entity decides that two members of an OpVO should have a conversation, or one of the members should be requested to transfer an ongoing session it has with another OpVO component to a most suitable device. The distributed perimeter security model adopted by the OpVO means that interactions between entities can only occur in the context of a specific OpVO instance. This issue is considered in section 7.4.5.1.

7.3.5.1.1. **SIP Broker threat model**

Figure 32 shows both the authorized way to use the SIP Broker and the potential attacks we have identified.

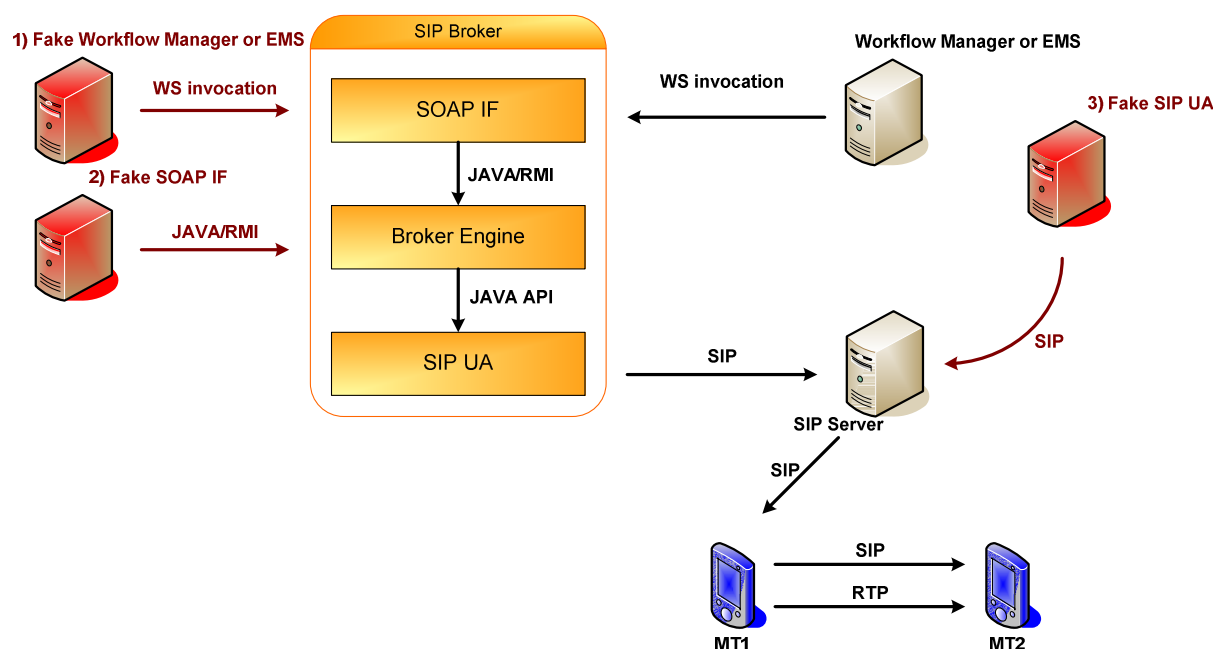


Figure 32 – “SIP with SOAP” possible attacks

During a normal (authorised) operation, the Workflow Manager will request for a SIP process using the interface that the SOAP interface subcomponent provides; then it invokes the corresponding method of the Broker Engine using Java RMI, which will invoke the Java API provided by the SIP UA to start the corresponding SIP mechanism (arrange a call or request for a transfer in the current prototype). These interactions have been depicted with black arrows in the figure.

We have identified three types of potential attacks, depending on the faked entity.. Non authorised elements and interactions have been depicted in Figure 32 – “SIP with SOAP” possible attacks with numbers 1), 2) and 3).

1. If the SOAP interface does not check the identity of the invoking element (i. e. the Workflow Manager or the EMS), it could be possible that a non-authorized element makes use of the exposed WS interface to make SIP call or SIP transfer requests. So in order to avoid this kind of attacks, this WS interface should implement authorisation mechanisms. This is a **pure WS interface security problem**.
2. The broker engine is exposed as a RMI daemon, so it could be invoked from a remote machine. So if the identity of the calling entity is not guaranteed, some entity which knows the RMI URL could fake the identity of the SOAP interface and make non-authorized requests; to solve this, the JAVA RMI interface must guarantee that only the real SOAP interface is allowed to use the broker engine. This is a **pure Java RMI security problem**.
3. Finally, some malicious SIP entity could pose as the whole SIP Broker (to be precise, posing as the SIP UA) to send false SIP requests to end user’s MTs. If neither the SIP Server nor the end user terminal checks for the request authenticity, this non-authorized entity could send a request to an MT to initiate a call or to perform a transfer. Anyway, this is a **pure SIP security problem**.

So all possible attacks in which some entity poses as the SIP Broker (of any of its components) can be reduced to a technology-related security problem (security with WS, RMI or SIP interactions). Chapters 7.4.5.2 and 7.4.5.3 describe these different security mechanisms.

7.3.5.2. Registration Hijacking

The SIP registration mechanism allows a user agent to identify itself to a registrar as a device at which a user is located, in Akogrimo project the SIP address are as *user@domain*. A registrar assesses the identity asserted in the “From” header field of a REGISTER message to determine whether this request can modify the contact addresses associated with the address-of-record in the “To” header field. While these two fields are frequently the same in a REGISTER message, in a normal SIP REGISTER transaction (except in the case that the registration is done by a third party entity), there are many valid deployments in which a third-party may register contacts on a user's behalf.

The “From” header field of a SIP request, however, can be modified arbitrarily by the owner of a UA, and this opens the door to malicious registrations. An attacker that successfully impersonates a party authorised to change contacts associated with an address-of-record could, for example, de-register all existing contacts for a URI and then register their own device as the appropriate contact address, thereby directing all requests for the affected user to the attacker's device.

This threat type belongs to a family of threats that rely on the absence of cryptographic assurance of a request's originator. Any SIP UAs that represents a valuable service, for instance the SIP Broker, might want to control the access to its resources by authenticating requests that it receives. Even an end-user (MI) is interested in ascertaining the identities of originator of request.

This threat demonstrates the need for security services that enable SIP entities to authenticate the originators of requests.

7.3.5.3. Impersonating a Server

The domain to which a request is destined is generally specified in the Request-URI. UAs commonly contact a server in this domain directly in order to deliver a request. However, there is always a possibility that an attacker could impersonate the remote server, and that the UA's request could be intercepted by some other party.

In Akogrimo project this attack type is especially important, in case of impersonating a SIP Broker (see the third type of attack described on section 7.3.5.1.1), since an attacker could send a REFER message to an end user (for instance the doctor) in order to establishes a grid call with other user (the patient).

7.3.5.4. Tampering with Message Bodies

Consider a UA that is using SIP message bodies to communicate session encryption keys for a media session. Although it trusts the proxy server of the domain it is contacting to deliver signalling properly, it may not want the administrators of that domain to be capable of decrypting any subsequent media session. Worse yet, if the proxy server were actively malicious, it could modify the session key, either acting as a man-in-the-middle, or perhaps changing the security characteristics requested by the originating UA.

This family of threats applies not only to session keys, but to most conceivable forms of content carried end-to-end in SIP. These might include MIME bodies that should be rendered to the user, SDP, or encapsulated telephony signals, among others. Attackers might attempt to modify

SDP bodies, for example, in order to point RTP media streams to a wiretapping device in order to eavesdrop on subsequent voice communications.

For these reasons, the UA might want to secure SIP message bodies, and in some limited cases header fields, end-to-end. However, since many header fields are legitimately inspected or altered by proxy servers as a request is routed, not all header fields can be secured end-to-end.

The security services required for message bodies include confidentiality, integrity, and authentication. These end-to-end services should be independent of the means used to secure interactions with intermediaries such as proxy servers.

7.3.5.5. Tearing Down Sessions

Once a session has been established by initial messaging, subsequent requests can be sent that modify the state of the dialog and/or session. It is critical that principal participants in a session can be certain that such requests are not forged by attackers.

Consider a case in which a third-party attacker captures some initial messages in a dialog shared by two parties in order to obtain the parameters of the session (“To” header, “From” header, and so on) and then inserts a BYE request into the session. The attacker could opt to forge the request such that it seemed to come from either participant. Once the BYE is received by its target, the session will be torn down prematurely.

Similar mid-session threats include the transmission of forged re-INVITEs that alter the session.

The most effective countermeasure to this threat is the authentication of the sender of the BYE message. In this instance, the recipient needs only to know that the BYE came from the same with whom the corresponding dialog was established. Also, if the attacker is unable to learn the parameters of the session due to confidentiality, it would not be possible to forge the BYE message. However, some intermediaries (like proxy servers) will need to inspect those parameters as the session is established.

7.3.5.6. Denial of Service and Amplification

Denial-of-service attacks focus on rendering a particular network element unavailable, usually by directing an excessive amount of network traffic to its interfaces. A distributed denial-of-service attack allows one network user to cause multiple network hosts to flood a target host with a large amount of network traffic.

SIP creates a number of potential opportunities for distributed denial-of-service attacks that must be recognized and addressed by the implementers of SIP systems.

Attackers can create bogus requests that contain a falsified source IP address and a corresponding “Via” header field that identify a targeted host as the originator of the request and then send this request to a large number of SIP network elements, thereby using hapless SIP UAs or proxies to generate denial-of-service traffic aimed at the target.

Similarly, attackers might use falsified “Route” header field values in a request that identify the target host and then send such messages to forking proxies that will amplify messaging sent to the target.

“Record-Route” could be used to similar effect when the attacker is certain that the SIP dialog initiated by the request will result in numerous transactions originating in the backwards direction.

A number of denial-of-service attacks open up if REGISTER requests are not properly authenticated and authorised by registrars.

Attackers could de-register some or all users in an administrative domain, thereby preventing these users from being invited to new sessions. An attacker could also register a large number of contacts designating the same host for a given address-of-record in order to use the registrar and any associated proxy servers as amplifiers in a denial-of-service attack. Attackers might also attempt to deplete available memory and disk resources of a registrar by registering huge numbers of bindings.

The use of multicast to transmit SIP requests can greatly increase the potential for denial-of-service attacks.

These problems demonstrate a general need to define architectures that minimize the risks of denial-of-service, and the need to be mindful in recommendations for security mechanisms of this class of attacks.

7.4. Defences

Spotting the most important fields in which security work needs to be done in Akogrimo helps us to focus on the targets that need to be secured. In addition to that, the review of possible relevant attacks is used as a guidance to define a generic defence architecture to a broader range of attacks related to the previously defined targets.

7.4.1. Assumptions

To focus even more on the most relevant parts of security for this project we will assume the following premises:

- No attacks will be executed by an Akogrimo component from the core network towards another Akogrimo component. In other words, this kind of attack can be seen as not happening, because of contracts, SLA’s and Monitoring. The case of having the QoSB possible, does not seem to be quite probable and therefore it will not be considered, as typically network operators consider their core network secure.
- The core is considered secure, communications inside the core are not expected to be eavesdropped or manipulated as long as they don’t leave the network’s administration area. However, if communications transverse different administration areas, or imply two user end points or a user endpoint and the network, this is considered as communication outside of the core and therefore efficient security mechanisms should be developed.
- Misuse from part of the user that can compromise his own security is not contemplated. That is, cases like the user loosing his private key or its password, physical and social engineering, user forgetting etc... are out of scope.
- Users need to be previously registered as Akogrimo users before making use of it.
- The user’s device is considered to be the security endpoint. Usually that implies no more than one *simultaneous* user per handled (device). This could be extended in the future.
- Security solutions at L2 are not considered since the Akogrimo architecture is all-IP access technology independent. So WEP, WPA or any other L2 access technology dependent

security mechanism is neither taken for granted nor expected and will not be considered as part of this work. Although, in principle, not necessary, its use can be optional as additional security means.

- Insecurity related to host internal software security such as software bugs, rootkits, viruses, OS exploits as well as physical access attacks fall out of scope.
- Highly sophisticated and complicated cryptographic attacks that are highly improbable are not considered. If a cryptographic algorithm is considered secure by the community so will we. Some algorithms that have been broken can be still considered secure such as the SHA-1 hash, considered broken but not necessary vulnerable. (brute force attack to find collisions would take fewer than 2^{80} operations whereas using a know attack “only” 2^{69} operations)

7.4.2. Perimeter security

In order to maintain the core secure, the perimeter of the core needs to be efficiently secured. This is the first contact that the user has with the system. At this point the users are authenticated and granted permissions according to their profiles. To this end, traffic coming from the access network or the Internet will have to be correspondingly secured and unauthenticated traffic dismissed.

7.4.2.1. Perimeter defence

It will be enforced that all access through the core or to elements in the core, will have to be authenticated, authorised and audited. This reduces possible attacks to well known and auditable users. In order to do this, we need means to authenticate users, enforce that they communicate just with whom they are allowed to and encrypt communications between users and the network. On the other hand, the user will also want to make sure that the network he is using is the one he thinks he is using in order to avoid attacks towards users by someone trying to pretend to be an access network. This implies then bi-directional authentication.

7.4.2.1.1. User Authentication

For authentication, the A4C will be used as shown in Figure 33. PANA will be used to convey the messages from the MT to the access network and Diameter will convey the messages inside the A4C infrastructure.

Upon successful bidirectional authentication the A4C will securely convey to the AR and the MT enough cryptographic material to create corresponding Security Associations and therefore bootstrap an IPsec tunnel between both. Connections between A4C and AR are considered secured as they both are inside the core network and connections between A4C and MT are encrypted and the material is only provided after successful authentication.

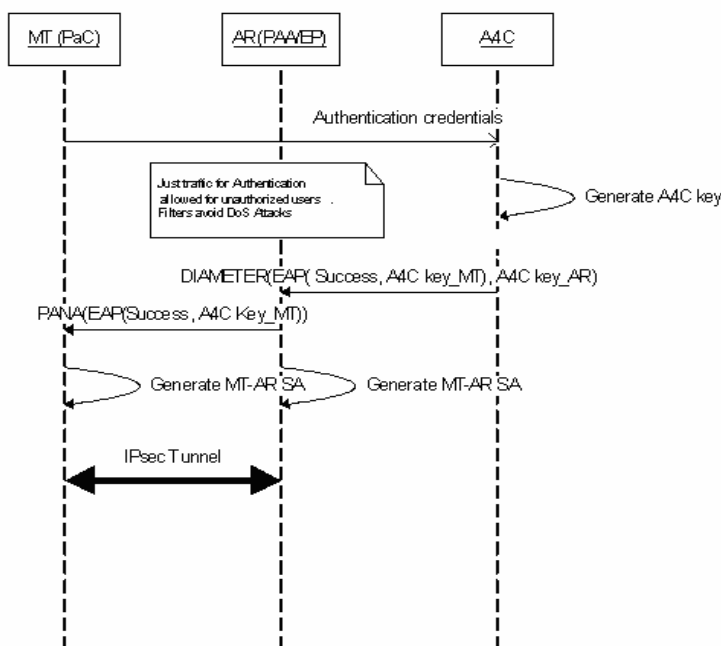


Figure 33 – Network Authentication

7.4.2.1.2. Access Router Defences

The AR is the first line of defence of the Akogrimo network. Its role is to prevent unauthorised users from accessing the Akogrimo network, and to forward valid packets from authorised users to their destination. The AR uses, at its core, Linux kernel netfilter, which is given a set of rules, determining whether or not to drop (or block) a packet. A first basic rule that would need to be enforced is that no traffic without corresponding authentication or encryption will be let in.

In this section we present a short explanation of the main basic principles that will allow a better understanding of how the AR works internally, and then proceed to analyse what attacks can be made and what protections the AR has.

- **Network flows**

A network flow is identified by protocol, source and destination addresses, their respective port numbers and DSCP code.

- **Rules**

Netfilter is linux’s packet filter. It inspects packet headers and then decides whether or not to drop packets based on rules with which can be configured

The AR gets information about how to configure its netfilter rules from the QoS Broker.

A rule identifies a network flow and has information on what to do with it. Since the actual comparisons and dropping and accepting of packets are done at kernel level, and by looking at just specific parts of the packet’s header, these actions are very fast.

- **Encryption**

IPSec in transport mode is possible. Packets will be encrypted from the MT to the AR, where they are decrypted, allowing the AR to perform its operations. Then, after traversing the core network and reaching the AR that serves the destination MT, they are once again encrypted and finally delivered to that MT.

- **AR General Characteristics**

- Only accepts packets from authenticated users. All other packets are promptly dropped. There is minimal CPU use.
- When a packet from a new flow arrives, the AR checks with the QoS Broker if that flow is allowed. If indeed it is allowed, a specific rule for that flow is installed by the QoSB in the AR.
- When a packet from an already known flow arrives, filters are already in-place and the packet is dealt without further requests to the QoS Broker.

These characteristics, although not eliminating the possibility of DoS attacks on both the AR and other core network components, pose significant challenges to an attacker.

7.4.3. Mobile IPv6

The use of MIPv6 in case of interdomain mobility reserves some attention in terms of security.

The Mobile IPv6 protocol itself (e.g. Return Routability Procedure) already has some provisions in order to minimise some of its inherent weaknesses.

To avoid some of the other weak points MIPv6 introduces, the use of IPsec security mechanisms in transport mode, as described in [DuCo05, RFC3776], to ensure integrity in Binding Updates (BU) and Binding Acknowledgements (BA) is mandatory.

7.4.4. PBNM Defence

The PBNM just communicates with other components inside the same core network and therefore no special network security mechanisms are contemplated. It is suggested, nevertheless, the use of IPsec in transport mode and the use of the Akogrimo PKI for authentication. The weak points here are not considered relevant and therefore work will be undertaken on other more important tasks instead.

7.4.5. SIP Defence

In the following the most relevant candidates for SIP defence are presented.

Rather than defining new security mechanisms specific to SIP, SIP reuses wherever possible existing security models derived from the HTTP and SMTP space. Several security mechanisms are analysed and describe at Annex E of this doc, such as, IPsec, TLS, S/MIME, HTTP Digest and so on.

The implementation in Akogrimo project of some certificate-based security mechanism, like TLS [TLS], will provide data integrity and confidentiality, preventing from MiM attacks. Other authentication mechanisms, such as HTTP Digest could be used, in order to provide good DoS attacks protection.

By using IPsec security mechanisms direct attacks on the MTs could be avoided making use of peer-to-peer nature of SIP, therefore avoiding for instance that the attacker sends a request to the MT, impersonating both the SIP Server and the SIP Broker.

Finally the implementation of S/MIME provides the following cryptographic security services for electronic messaging applications:

- Authentication, message integrity and non-repudiation of origin (using digital signatures), and data confidentiality (using encryption).

- Since the general use of S/MIME is to secure the message bodies, such as the SDP rather than message headers, it seems most logical, that S/MIME enables to avoid attacks like those described in chapter 7.3.5.4 “Tampering with message bodies”.

It is worth noting as a drawback that S/MIME (the same as IPSec) generates considerable overhead in SIP messages, not to mention the time delay introduced during the encryption and validation processes.

So as conclusion, each one of these security mechanisms has some advantage and some drawbacks. Therefore as a general rule, IPsec will be used as it is used in other parts of the architecture, and S/MIME will be used only when necessary e.g. several users are using the same terminal, or the same session key is using from different terminals.

Besides these mechanism related to own SIP architecture, Akogrimo will implement another security mechanism “SIP&SOAP Security Model”, which will provide an additional security level, for a specific part of the architecture.

7.4.5.1. SIP&SOAP Security Model (VO Security)

The adopted solution relies on the OpVO security mechanisms and takes advantage on the extensibility and on the backward compatibility of the SIP protocol. Since the SIP specification allows the definition of new header fields to be included as part of the SIP messages (as an extension), and it also specifies that it must be ignored by an entity that no recognizes it (which assures the backward compatibility), a new header field containing the OpVO membership token have been defined. This token will be included into the SIP signalling messages associated to the grid-triggered SIP call, the SIP transfer, the addition of a new user to an ongoing multivideoconference session or during the request for services information (i.e. all the transactions affected by the SIP with SOAP approach). This new header will be processed at application level at the MTs, providing end-to-end security. Anyway, it will have a meaning only within the Akogrimo context; any other "non-Akogrimo" SIP entities are requested to ignore it, as SIP protocol specifies.

So using this approach, when a MT is requested to be the initiator of a SIP call, it can check if the request coming from the SIP Broker contains a valid OpVO token prior to start the communication. In this way, the initiating MT can check if the call is authorised. The only requirement is, this information should be included somehow in the Workflow Manager or EMS request to the SIP Broker, because the SIP transaction is sent to the MTs by the SIP Broker in response to a WS request from these entities.

At the other hand, and in the case of a grid triggered SIP call or transfer, the initiator MT will include its OpVO token in the request, so the receiving MT can be sure the call is authorised. MT will obtain its OpVO tokens by connecting to the corresponding VO User Agent during the VO login. The User Agent is created by the workflow to represent a particular user to the workflow. The user "connects" to the User Agent from a particular MT (authentication, selection of OpVO, authorisation, etc). At this point, the MT is given its OpVO token.

The best aspect of this approach is that the SIP network entities do not need to be secured from the OpVO security model perspective - being outside the OpVO firewall they are not a security threat (other than from a denial of service). Of course, self protection for the SIP broker and the SIP Server would be beneficial, and these could use the same mechanism (i. e. the baseVO dynamic perimeter) to ensure they only respond to valid Akogrimo OpVOs and MTs.

7.4.5.1.1. OpVO's security model in Akogrimo

Essentially, any user/service participating in an OpVO will be given an OpVO token to prove their membership of that OpVO instance. In this way, the user/service should only accept

instructions/data from an entity that also possesses a similar token. This token can be used to provide end-to-end security for the communications.

The VO security mechanism operates as follows: all participants in an OpVO are given membership tokens signed by the OpVO's Manager and containing their identity. Note that this mechanism depends on the security strategies adopted at VO level in order to avoid that an attacker could use an eavesdropped token to impersonate a valid VO member.

An OpVO token will contain the following information:

OpVOManager's Token	BaseVO name BaseVO public key OpVO name OpVOManager's public key
OpVO Token	OpVOManager's token member's name member's public key member's role

Table 13 – OpVO member's token content

Each participant only responds to messages that contain such a token. The message content can be protected using the token and message sequencing mechanisms. Thus the OpVO members communications within this OpVO are secured against external attack. This basic mechanism can be extended to provide security against insider attacks by enforcing role-based authorisation, or even checking with an authorisation engine (which would have to be in the OpVO).

For example, we create an eHealth OpVO, O1, with doctor D, patient P, and two services S1 and S2. Now all of these are provided with a membership token for O1, containing their name. O1 (or rather its workflow instance) will instruct S1 to invoke S2. S1 checks the message came from O1 and has an O1 membership token (if not, the message is ignored). It then invokes S2, passing its own membership token. S2 checks the message is from S1 and that S1 has a membership token. It then accepts the invocation. Note that how S2 gets the token and how the validation is done is out of scope of WP41, since it is described on the corresponding security sections of WP44 deliverables.

In the context of the SIP with SOAP interactions being considered in Akogrimo (grid triggered SIP calls and SIP session transfers), Mobile Terminals should be able to process tokens, signatures, encryption, message sequencing, etc... However, if that were not the case for a certain component, then the SIP Gateway will take over the responsibility of operating the "distributed perimeter". On receiving an instruction to connect two Mobile Terminals, it will have to check that the message has been authorised before acting on it, i.e. both Mobile Terminals are members of the same OpVO as the sender. It will have to access the Participant's Registry to check what OpVOs the MTs belongs to, so needs to be a member of the same BaseVO as the OpVO. It may also have to check with the OpVO that this particular communication is authorised (the insider attach protection mentioned above)

In the example, if a service were to be a doctor doctor being used by a patient, the call should only be possible if the doctor is treating the patient, i. e. in the context of an OpVO. When the patient gets the call, he should be sure that the caller is their own doctor, because the call is known to belong to the OpVO.

The inclusion of this mechanism in the SIP with SOAP security model has been described in section 7.4.5.1.

7.4.5.2. WS interaction path

As described earlier, in principle one attacker could impersonate the identity of the Workflow Manager and make a non-authorised use of the WS exposed by the SIP Broker. Apart from this, the WS communication path can be intercepted and the exchanged messages modified by a non-authorised entity. In order to prevent this, specific WS security mechanisms are envisaged.

WS-Security core specification [WS Sec] defines several mechanisms to provide protection against SOAP attacks. It includes also mechanisms to detect if some message has been manipulated (XML signature) and encryption mechanism as well (XML encryption).

However, the adoption of the OpVO security model provides a higher level of protection, because the attacker should include a valid OpVO token in the request; in any other case it will be rejected.

7.4.5.3. RMI path

Internal communications between the SOAP interface and the SIP Broker broker engine is made by means of an RMI interface. This means that remote clients could access the classes and resources that the RMI server (the broker engine in our case) makes accessible. So if some malicious entity knows how to invoke the corresponding methods, it could impersonate the identity of the real SIP Broker SOAP interface and could make unauthorised requests.

To prevent these kinds of attacks, we must take advantage on the Java Security Model [Java Sec]. Using this, you can restrict the actions that remote entities can perform. By default, an RMI program does not have a security manager installed, and no restrictions are placed on remotely loaded objects. Anyway, the `java.rmi` package provides a default security manager implementation that can be used, which requires the specification of a security policy file containing specific permissions regarding which operations are granted and from which locations (machines, ports) [RMI]. So the best way to proceed is to specify that only requests from the machine in which the SIP Broker SOAP interface is running are allowed.

The current implementation of the SIP Broker enables the configuration of the policy file to be used. No security policies are applied if this file is not specified.

7.5. Security Architecture Overview

Having a view on the overall security concept, four crucial security concerns must be considered: The user's identity must be validated within his home domain as well as in a foreign domain, the belonging of users and services to Virtual Organizations must be stated clearly, all communication related to private data must be secured and all actions must be monitored to detect misuse.

The user's identity is secured with his Identity Token (IDToken), which is issued after the initial login to the platform by the SAML Authority at the A4C Server in his network provider domain. The IDToken is a string, composed of a SAML Artifact, which is a pointer to a SAML Authentication Assertion, a serial number, a random number, a UserID and a signature. Each time the user uses his token, it must first be updated. That implies incrementing the serial number and generating a new random number before adding a new signature to the token. An IDToken can only be validated at the user's home A4C Server, which sends Identity Information, e.g., the user's attributes, to requesting components via the A4C infrastructure. Since the security mechanisms of the token guarantee a nearly impossible to forge token and the trust between

different A4C servers is established by offline signed contractual agreements, all Akogrimo members can trust the user’s token after it’s validation.

The belonging to VOs is secured with VO tokens. A VO token is an XML file containing information about the VO, the membership and the bearer. It is signed by the VO Manager and issued whenever a participant joins the VO.

The communication between the A4C servers is secured in two ways: Each server has a configuration file that states those servers it is allowed to communicate. Also, the connection between servers is made using a PKI infrastructure and SSL. The communication inside of VOs is secured also by the user of PKI and the related X.509 certificates, but here the VO tokens enhance the security once more.

The following figure depicts the security additions to the architecture that need to be accomplished in order to realize the defence mechanisms shown above. The assumption of taking the core network as secure is taken in order to give preference to security mechanisms that revolve around innovative aspects of the project rather than standard out-of-the-box security mechanisms.

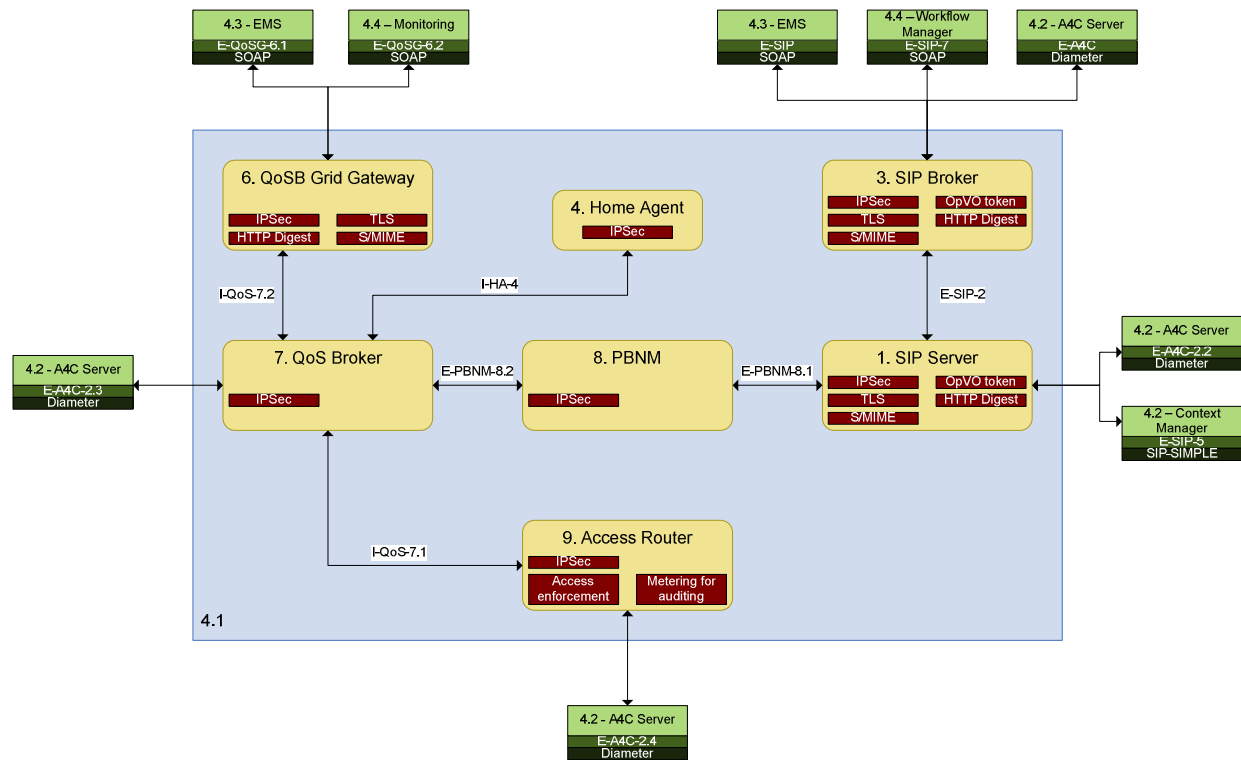


Figure 34 – General architecture

The main security components at network level are the following:

The AR:

The AR is the enforcement point that restricts access to external users or lets them through depending on the policy linked to them after they have been authenticated. Across the same domain, reauthentication is not necessary and it would be enough for a user to show a token to regain access from another part of the network, thus minimizing the impact of a handover over real-time communication. Metering is also performed in order to detect misuses.

The A4C:

It is responsible to authenticate the user. It links the user to a certain profile and provides the user with the necessary material to access the network and other resources of the system. The use of these and other services is based on Single Sign On, allowing higher layers to not have to worry about authentication issues.

The PBNM:

It collaborates with the A4C to provide information as to whether a user is allowed to make use of a resource and to what extent. This liberates each service and corresponding resources of having to track user profiles and facilitates the management of user accesses making at the same time for a more secure architecture. The policies stored in the PBNM can be changed whenever is necessary depending on the current scenarios or necessities, and these are propagated to all users using the system.

The QoSB

It is the brain that upon reception of the information from the A4C regarding an authentication and PBNM regarding the general network policies to apply, instructs the AR to enforce a specific QoS ranging from zero bandwidth in case the user is not allowed in to specific QoS bundles depending on the policy and resource contacted.

These components together with other security resources such as the IPsec, OpVO tokens or S/MIME (See E.4) compose the security architecture at network level.

These building blocks, which collaborate with security, even if indirectly, together with the overall VO-centric security concept and the corresponding security measures studied above to secure communications, strive to provide a uniform and robust cross-layer security approach

References

- [CIM] Common Information Model (CIM) Standards <http://www.dmtf.org/standards/cim/>
- [COPS] D. Durham et al., RFC 2748 “The COPS (Common Open Policy Service) Protocol”, <http://www.ietf.org/rfc/rfc2748.txt>
- [DMTF] Distributed Management Task Force, <http://www.dmtf.org/>
- [D411] Deliverable D4.1.1, “Mobile Network Architecture, Design and Implementation”, <http://www.akogrimo.org/modules.php?name=UpDownload&req=getit&lid=43>
- [D413] Deliverable D4.1.3, “Final Network Service Provisioning Concept”, <http://www.akogrimo.org/modules.php?name=UpDownload&req=getit&lid=117>
- [D424] Deliverable D4.2.4_Consolidated Integrated Services Design and Implementation Report <http://www.akogrimo.org/modules.php?name=UpDownload&req=getit&lid=120>
- [DuCo05] F. Dupont, J-M. Combes. “Using IPsec between Mobile and Correspondent IPv6 Nodes” <draft-ietf-mip6-cn-ipsec-02.txt>, IETF draft, December 2005.
- [FHO] R. Koodli, RFC4068 “Fast Handovers for Mobile IPv6”, <http://www.ietf.org/rfc/rfc4068.txt>
- [Java Sec] Security in Java 2 SDK 1.2; Security Features Overview. <http://java.sun.com/docs/books/tutorial/security1.2/overview/index.html>
- [MIPV6] D. Johnson, C. Perkins, J. Arkko, RFC3775 “Mobility Support in IPv6”, <http://www.ietf.org/rfc/rfc3775.txt>
- [OPENWBEM] Open [Web Based Enterprise Management](http://www.openwbem.org/) <http://www.openwbem.org/>
- [OPENPEGASUS] "C++ CIM/WBEM Manageability Services Broker" <http://www.openpegasus.org/>
- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999. <http://www.ietf.org/rfc/rfc2246>
- [RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998. <http://www.ietf.org/rfc/rfc2401>
- [RFC3084] COPS Usage for Policy Provisioning (COPS-PR). K. Chan, March 2001. <http://www.ietf.org/rfc/rfc3084>
- [RFC3261] Rosenberg J., Schulzrinne H. et al., “SIP: Session Initiation Protocol” RFC 3261. <http://www.ietf.org/rfc/rfc3261.txt>
- [RFC3268] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.
- [RFC3571] “Framework Policy Information Base for Usage Feedback” D. Rawlins, August 2003. <http://www.ietf.org/rfc/rfc2571>
- [RFC3680] Rosenberg J., “A Session Initiation Protocol (SIP) Event Package for Registrations” RFC 3680. <http://www.ietf.org/rfc/rfc3680.txt>
- [RFC3776] J. Arkko, V. Devarapalli, F. Dupont. “Using IPsec to Protect Mobile IPv6 Signalling Between Mobile Nodes and Home Agents”. RFC3776, IETF, June 2004. <http://www.ietf.org/rfc/rfc3776>
- [RFC3891] Mahy, R., Biggs, B., Dean, R., “The Session Initiation Protocol (SIP) "Replaces" Header”, RFC 3891, IETF, September 2004. <http://www.ietf.org/rfc/rfc3891>

- [RFC4028] Rosenberg J., Donovan, S., "Session Timers in the Session Initiation Protocol (SIP)" RFC 4028. <http://www.ietf.org/rfc/rfc4028.txt>
- [RFC2617] J. Franks, , P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC2617, June 1999. <http://www.ietf.org/rfc/rfc2617.txt?number=2617>
- [RMI] RMI tutorial. <http://java.sun.com/docs/books/tutorial/rmi/index.html>
- [RSVP] R. Braden et al., RFC2205 "Resource ReSerVation Protocol (RSVP)", <http://www.ietf.org/rfc/rfc2205.txt>
- [SER] SIP Express Router homepage. <http://www.iptel.org/ser/>
- [SHACH03] Shacham, R., Schulzrinne, H., et al. "Session Initiation Protocol (SIP) Session Mobility", IETF draft-shacham-sipping-session-mobility-03, Nov 2006.
- [SIPSOAP] Secure SIP with SOAP, Akogrimo Security Task Force, <http://bscw.hlrz.de/bscw/bscw.cgi/d111221/SIPwithSOAPsec.doc>
- [TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999. <http://www.ietf.org/rfc/rfc2246>
- [WS CONT] Dorn, C., Dustdar, S.. "Achieving Web Service Continuity in Ubiquitous Mobile Networks the SRR-WS Framework". 4th International Workshop on Ubiquitous Mobile Information and collaboration Systems (UMICS), co-located with CAiSE 2006, 5 - 6 June 2006.
- [WSDL] Web Services Description Language (WSDL) <http://www.w3.org/TR/wsdl>
- [WS Sec] Web Services Security: SOAP Message Security 1.1 (WS-Security 2004) OASIS Standard Specification, 1 February 2006. <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

Annex A. Access security at L3 example

Secure wireless access at 3-layer (IPsec). Just access with IPsec is allowed. This is an example implemented using IPv4 to show the functionality. The future implementation in IPv6 is analogous.

MN: 192.108.37.99

AR: 192.108.37.97

```
# apt-get install racoon # Automatic keying
Choose direct mode in the menu
# modprobe crc32c
# modprobe ipcomp6
```

/etc/racoon/setkey.conf

```
-----
#!/usr/sbin/setkey -f
```

```
flush;
spdflush;
```

```
# server out to anything wireless: unrestricted
spdadd 192.108.37.97/32 192.168.37.96/29 any -P in none;
```

```
# anything wireless into server: unrestricted
spdadd 192.168.37.96/29 192.108.37.97/32 any -P out none;
```

```
# anything else going to laptop must tunnel from server to laptop
spdadd 0.0.0.0/0 192.108.37.99/32 any -P in ipsec
        esp/tunnel/192.108.37.97-192.108.37.99/require;
```

```
# anything else coming from laptop must tunnel from laptop to server
spdadd 192.108.37.99/32 0.0.0.0/0 any -P out ipsec
        esp/tunnel/192.108.37.99-192.108.37.97/require;
```

```
-----
In the server is the same but changing all "in and outs"
```

/etc/racoon.conf

```
-----
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";
remote anonymous {
    exchange_mode main;
#    generate_policy off;
#    certificate_type x509 "my-cert.pem" "private.pem";
#    my_identifier asn1dn;
#    peers_identifier asn1dn;
    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
#        authentication_method rsasig;
        authentication_method pre_shared_key;
        dh_group modp1024;
    }
}
sainfo anonymous {
    pfs_group modp768;
    encryption_algorithm 3des;
    authentication_algorithm hmac_md5;
```

D4.1.4, 1.0

```
        compression_algorithm deflate;
    }
-----
```

```
/etc/racoon/psk.txt
# IPv4/v6 addresses
192.108.37.97    I did it my way
#Add here the rest of the nodes in the wireless link if you want to
communicate with them...
```

```
#172.16.1.133    0x12345678
#194.100.55.1    whatcertificatereally
#3ffe:501:410:ffff:200:86ff:fe05:80fa    mekmitasdigoat
#3ffe:501:410:ffff:210:4bff:fea2:8baa    mekmitasdigoat
# USER_FQDN
#foo@kame.net    mekmitasdigoat
# FQDN
#foo.kame.net    hoge
-----
```

```
On both machines:
# setkey -f /etc/racoon/setkey.conf
# racoon -F -f /etc/racoon/racoon.conf
-----
```

```
Filter non authorized access to the network i.e. everything that is
not a esp tunnel.
```

```
(AR)
```

```
#!/bin/bash
```

```
EXT_IF="wlan0"
INT_IF="eth0"
IPTABLES="/sbin/iptables"
```

```
$IPTABLES -F
$IPTABLES -t nat -F
$IPTABLES -t mangle -F
```

```
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT ACCEPT
```

```
# Mark VPN packets
```

```
$IPTABLES -t mangle -A PREROUTING -i $EXT_IF -p esp -j MARK --set-mark
1 #VPN
```

```
$IPTABLES -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A INPUT -i $INT_IF -j ACCEPT
$IPTABLES -A INPUT -p icmp -j ACCEPT
$IPTABLES -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT #SSH
$IPTABLES -A INPUT -i $EXT_IF -p udp -m udp --dport 500 -j ACCEPT #VPN
$IPTABLES -A INPUT -i $EXT_IF -m mark --mark 1 -j ACCEPT
```

```
$IPTABLES -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPTABLES -A FORWARD -i $EXT_IF -m mark --mark 1 -j ACCEPT
$IPTABLES -A FORWARD -i $INT_IF -j ACCEPT
```

Annex B. SER (SIP Express Router) configuration file (ser.cfg)

```

#
# $Id: ser.cfg,v 1.4 2006/03/08 12:22:25 jeb1231 Exp $
#
# simple quick-start config script
#

# ----- global configuration parameters -----

#debug=          # debug level (cmd line: -ddddddddd)
#fork=yes
#log_stderr=no   # (cmd line: -E)

# Uncomment this line to enter debugging mode
debug = 3

# If log_stderr is set to 'yes', traces are sent to the console; if set
# to 'no', traces are sent
# to the corresponding trace file in folder /temp
log_stderr = yes

# Uncomment these lines for multithreaded
fork = yes
children = 3

# Here goes the real IP of the machine hosting the SIP server
listen = 192.108.37.72
listen = 2001:638:202:e00:213:d4ff:fe36:dab1
port = 5060

check_via=no    # (cmd. line: -v)
dns=no          # (cmd. line: -r)
rev_dns=no      # (cmd. line: -R)

fifo="/tmp/ser_fifo"
alias = "akogrimo.org"
alias = "test.akogrimo.org"

# ----- module loading -----
# Uncomment this if you want to use SQL database
#loadmodule "/usr/local/lib/ser/modules/mysql.so"

loadmodule "/usr/lib/ser/modules/sl.so"
loadmodule "/usr/lib/ser/modules/exec.so"
loadmodule "/usr/lib/ser/modules/tm.so"
loadmodule "/usr/lib/ser/modules/rr.so"
loadmodule "/usr/lib/ser/modules/maxfwd.so"
loadmodule "/usr/lib/ser/modules/usrloc.so"
loadmodule "/usr/lib/ser/modules/registrar.so"
loadmodule "/usr/lib/ser/modules/esc.so"
loadmodule "/usr/lib/ser/modules/a4c.so"

# Uncomment this if you want digest authentication
# mysql.so must be loaded !
#loadmodule "/usr/lib/ser/modules/auth.so"

```

D4.1.4, 1.0

```
#loadmodule "/usr/lib/ser/modules/auth_db.so"

# ----- setting module-specific parameters -----

modparam("registrar","default_expires",86400)
# -- esc params (presence agent module) --
modparam("esc", "trace_level", 2)
modparam("esc", "check_expires_timer_interval", 600)
modparam("esc", "max_subscription_expires_value", 3600)
modparam("esc", "max_publish_expires_value", 3600)
modparam("esc", "auth_mode", 0)
#modparam("esc", "single_authorized_subscriber_uri",
"contextmanager@akogrimo.org")

# -- a4c params --
modparam("a4c", "trace_level", 2)
modparam("a4c", "AC4_token_hf", "X-A4C-Token")
modparam("a4c", "A4C_client_command", "/usr/lib/ser/modules/A4CIFmodule")

# -- usrloc params --
modparam("usrloc", "db_mode", 0)

# Uncomment this if you want to use SQL database
# for persistent storage and comment the previous line
#modparam("usrloc", "db_mode", 2)

# -- auth params --
# Uncomment if you are using auth module
#
#modparam("auth_db", "calculate_ha1", yes)
#
# If you set "calculate_ha1" parameter to yes (which true in this config),
# uncomment also the following parameter)
#
#modparam("auth_db", "password_column", "password")

# -- rr params --
# add value to ;lr param to make some broken UAs happy
modparam("rr", "enable_full_lr", 1)

# ----- request routing logic -----

# main routing logic

route
{

    # initial sanity checks -- messages with
    # max_forwards==0, or excessively long requests
    if (!mf_process_maxfwd_header("10"))
    {
        sl_send_reply("483", "Too Many Hops");
        break;
    };
    if ( msg:len > max_len )
    {
        sl_send_reply("513", "Message too big");
        break;
    };

    # we record-route all messages -- to make sure that
    # subsequent messages will go through our proxy; that's
```

D4.1.4, 1.0

```
# particularly good if upstream and downstream entities
# use different transport protocol
record_route();
# loose-route processing
if (loose_route())
{
    t_relay();
    break;
};

# if the request is for other domain use UsrLoc
# (in case, it does not work, use the following command
# with proper names and addresses in it)
if (uri==myself)
{

    if (method=="REGISTER")
    {

        # Uncomment this if you want to use digest
        # authentication
        /*
        if (!www_authorize("hi.inet", "subscriber")) {
            www_challenge("hi.inet", "0");
            break;
        };
        */
        # Uncomment this if A4C authorisation will not be
        # used and comment lines below
        /*
        log(1, "REGISTER transaction\n");
        save("location");
        break;
        */
        # Uncomment this to use A4C authorisation and comment
        # lines above
        if (aa_request())
        {
            log(1, "REGISTER transaction\n");
            save("location");
        }
        break;

    };

    # Presence agent behaviour
    if (method=="SUBSCRIBE")
    {
        if (!t_newtran())
        {
            log(1, "newtran error\n");
            sl_reply_error();
        };
        handle_subscription();
        break;
    };
    # Inform about not registered destinations
    if (!lookup("location"))
    {
        log(1, "user not found\n");
        sl_send_reply("404", "Not Found");
    }
}
```

D4.1.4, 1.0

```
        break;
    };
    if (method=="PUBLISH")
    {
        # The following piece of code creates a new
        # transaction.
        # MANDATORY IF THE MODULE PROCESSING THIS METHOD
        if (!t_newtran())
        {
            log(1, "newtran error\n");
            sl_reply_error();
        };
        handle_publish();

        break;
    };
}
# Proxy behaviour (forward to current URI using stateful
# forwarding)
log(1, "Proxy transaction\n");
if (!t_relay())
{
    log(1, "newtran error\n");
    sl_reply_error();
};
}
```

Annex C. Network User Profile

This is a skeleton XML file of the Akogrimo Network User Profile.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="ANUP">
    <xs:annotation>
      <xs:documentation>Akogrimo Network User Profile</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UID" type="xs:string">
        </xs:element>
        <xs:element name="PRIO" type="xs:short">
        </xs:element>
        <xs:element name="MAC" type="xs:string">
        </xs:element>
        <xs:element name="DSCP" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:unsignedInt">
              </xs:element>
              <xs:element name="Macro">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="MaxUpBW" type="xs:unsignedLong">
                    </xs:element>
                    <xs:element name="MaxDownBW" type="xs:unsignedLong">
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="Layout" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Filter">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="FromAddr" type="xs:string">
                          </xs:element>
                          <xs:element name="ToAddr" type="xs:string">
                          </xs:element>
                          <xs:element name="FromPort" type="xs:unsignedInt">
                          </xs:element>
                          <xs:element name="ToPort" type="xs:unsignedInt">
                          </xs:element>
                          <xs:element name="Protocol" type="xs:string">
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="QoSBundle" type="xs:unsignedLong">
                    </xs:element>
                    <xs:element name="PeriodOfDay" maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:sequence>
```



```
        <xs:element name="Start" type="xs:string">
        </xs:element>
        <xs:element name="End" type="xs:string">
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Annex D. gSDP formats

This annex describes the different formats being used for the services SIPification framework

D.1. gSDP session

The gSDP sessions are described using an xml document. The related xml schema is:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!-- definition of simple types -->
<xs:simpleType name="avstatus">
  <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="AVAILABLE"/>
    <xs:enumeration value="UNAVAILABLE"/>
  </xs:restriction>
</xs:simpleType>
<!-- definition of attributes -->
<xs:attribute name="id" type="xs:string" use="required"/>
<xs:attribute name="reference" type="xs:string" use="required"/>
<xs:attribute name="availability" type="avstatus"/>
<!-- definition of complex types -->
<xs:element name="service">
  <complexType>
    <xs:attribute ref="id"/>
    <xs:attribute ref="reference"/>
    <xs:attribute ref="availability"/>
  </complexType>
</xs:element>
<xs:element name="services">
  <complexType>
    <sequence>
      <xs:element ref="service" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</xs:element>
</xs:schema>
```

D.2. gSDP notification

The content of the gSDP notifications are also based on xml. Its related xml schema is:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- definition of simple types -->
  <xs:simpleType name="avstatus">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="AVAILABLE"/>
      <xs:enumeration value="UNAVAILABLE"/>
    </xs:restriction>
  </xs:simpleType>
  <!-- definition of attributes -->
  <xs:attribute name="mobileuser" type="xs:string" use="required"/>
  <xs:attribute name="service" type="xs:string" use="required"/>
  <xs:attribute name="status" type="avstatus" use="required"/>
  <xs:element name="message">
    <complexType>
      <xs:attribute ref="mobileuser"/>
      <xs:attribute ref="service"/>
      <xs:attribute ref="status"/>
    </complexType>
  </xs:element>
</xs:schema>
```

Annex E. SIP security mechanisms

From the threats described above, the fundamental security services required for the SIP protocol are: preserving the confidentiality and integrity of messaging, preventing replay attacks or message spoofing, providing for the authentication and privacy of the participants in a session, and preventing denial-of-service attacks. Bodies within SIP messages separately require the security services of confidentiality, integrity, and authentication.

Rather than defining new security mechanisms specific to SIP, SIP reuses wherever possible existing security models derived from the HTTP and SMTP space.

Full encryption of messages provides the best means to preserve the confidentiality of signalling - it can also guarantee that messages are not modified by any malicious intermediaries. However, SIP requests and responses cannot be naively encrypted end-to-end in their entirety because message fields such as the “Request-URI”, “Route”, and “Via” need to be visible to proxies in most network architectures so that SIP requests are routed correctly. For instance the Proxy Servers need to modify some features of messages as well, such as adding values to “Via” header

To this purpose, low-layer security mechanisms for SIP are recommended, which encrypt the entire SIP requests or responses on the wire on a hop-by-hop basis, and that allow endpoints to verify the identity of proxy servers to whom they send requests.

SIP entities also have a need to identify one another in a secure fashion. A cryptographic authentication mechanism is provided in SIP to address this requirement.

An independent security mechanism for SIP message bodies supplies an alternative means of end-to-end mutual authentication, as well as providing a limit on the degree to which user agents must trust intermediaries.

E.1. Transport and Network Layer Security

Transport or network layer security encrypts signalling traffic, guaranteeing message confidentiality and integrity.

Oftentimes, certificates are used in the establishment of lower-layer security, and these certificates can also be used to provide a means of authentication in many architectures.

Two popular alternatives for providing security at the transport and network layer are, respectively, TLS [RFC2246] and IPSec [RFC2401].

E.1.1. IPSec

IPSec is a set of network-layer protocol tools that collectively can be used as a secure replacement for traditional IP. IPSec is most commonly used in architectures in which a set of hosts or administrative domains have an existing trust relationship with one another. IPSec is usually implemented at the operating system level in a host, or on a security gateway that provides confidentiality and integrity for all traffic it receives from a particular interface (as in a VPN architecture). It can also be used on a hop-by-hop basis.

In many architectures IPSec does not require integration with SIP applications; It is perhaps best suited to deployments in which adding security directly to SIP hosts would be arduous. UAs that have a pre-shared keying relationship with their first-hop proxy server are also good candidates to use IPSec. Any deployment of IPSec for SIP would require an IPSec profile describing the protocol tools that would be required to secure SIP.

The IP security (IPsec) provides confidentiality, integrity, data origin authentication services as well as traffic analysis protection.

IPsec was introduced to provide security services such as:

- Encrypting traffic (So it can not be read in its transmission)
- Integrity validation (Ensuring traffic has not been modified along its path)
- Authenticating the Peers (Both ends are sure they are communicating with a trusted entity the traffic is intended for)
- Anti-Replay (Protect against session replay)

For more detail see related documentation [RFC2401]

E.1.2. TLS

TLS provides transport-layer security over connection-oriented protocols, for the purposes of this document, so signifying TLS over TCP. The primary goal of the TLS Protocol is to provide privacy and data integrity between two communicating applications

TLS is most suited to architectures in which hop-by-hop security is required between hosts with no pre-existing trust association. For example, Alice trusts her local Proxy Server, which after a certificate exchange decides to trust Bob's local Proxy Server, which Bob trusts, hence Bob and Alice can communicate securely.

TLS must be tightly coupled with a SIP application. Note that transport mechanisms are specified on a hop-by-hop basis in SIP, thus a UA that sends requests over TLS to a proxy server has no assurance that TLS will be used end-to-end.

SSL/TLS has many of the advantages of IPsec and the successful introduction of the protocol in the wired Internet has proved its usability and effectiveness. Likewise, SSL/TLS runs above TCP/IP and below higher-level protocols such as HTTP or FTP and consequently the TCP header is not encrypted.

In contrast to IPsec, TLS does not assume any trust relation among communication parties, so, it is more suitable for Inter Domain Authentication.

SSL/TLS provides protection against several known attacks (including man in the middle attacks), prevents the eavesdropping, the tampering, and message forgery. However is susceptible to a number of denial-of-service (DoS) attacks. In particular, an attacker who initiates a large number of TCP connections can cause a server to consume large amounts of CPU doing RSA decryption.

Because TLS runs over TCP, it is also susceptible to a number of denial-of-service attacks on individual connections. In particular, attackers can forge RSTs, thereby terminating connections, or forge partial TLS records, thereby causing the connection to stall. These attacks cannot in general be defended against by a TCP-using protocol. Implementors or users who are concerned with this class of attack should use IPsec.

On the other hand, the biggest drawback with SSL/TLS is does not run over UDP. In addition, keeping up many TCP connections simultaneously may be too heavy for Proxy Servers.

E.2. SIP URI Scheme

The use of a SIPS URI of the form “sips:alice@akogrimo.org “ an INVITE message requires that TLS should be used on the whole path to the destination. Since each hop may add route information to the SIP message header, TLS protection must be realized on a hop-by-hop basis

on each segment of the path. The use of TLS requires the use of TCP as a transport protocol (TCP/SIP) and necessitates a public key infrastructure.

E.3. HTTP Digest

SIP provides a challenge capability, based on HTTP authentication that relies on the 401 and 407 response codes and header fields for carrying challenges and credentials as well. Without significant modification, the reuse of the HTTP Digest authentication [RFC 2617] scheme in SIP allows for replay protection and one-way authentication.

In the SIP context, the “server” entity authenticating the “client” entity can be a proxy, registrar, a redirect server or another user agent. The basic mechanism is similar in all cases: both client and server have a shared secret (password), which must be known in advance (e.g. using out of bands mechanisms). When the server receives a request, it challenges the client to provide valid credentials; then the client provides a valid name and demonstrates it knows the password.

One of the advantages of the HTTP Digest mechanism is that the password does not have to be sent in clear text, like in HTTP Basic Authentication. For this purpose the Digest mechanism uses hash algorithms like MD5 or SHA1 (a hash function is a one way functions that provide a fixed length output) and nonces (an arbitrary and random value which is used only once).

Next figure describe how this mechanism works. Note that the name of the header to be include and the challenge number depends on the nature of the server entity: Registrar Servers and UAS use the “WWW-Authenticate” and “Authorisation” header, and the 401 reply, while proxies use the “Proxy-Authenticate”,and “Proxy-Authorisation” headers, and 407 reply.

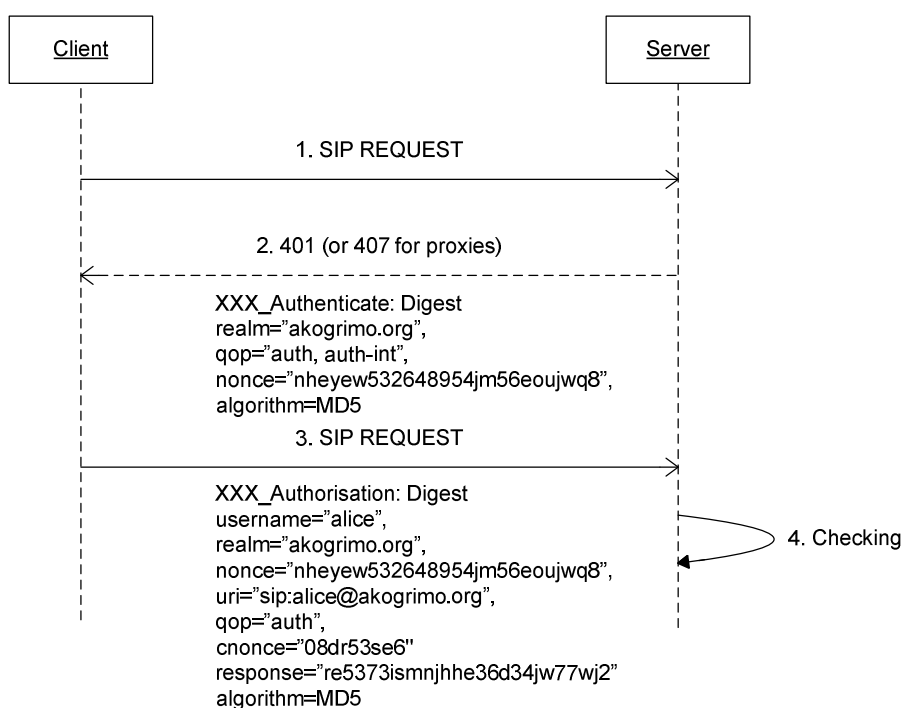


Figure 35 – HTTP Digest security mechanism.

Client sends a SIP request to the Server (REGISTER, INVITE...)

Server challenges the request with a 401 (407) reply, including a “XXX_Authenticate” header field with several parameters:

- realm: the domain of which the share secret (password) is required.

- Qop (quality of protection): indicates if integrity protection applies for either the request line (auth) or both request line and message body (auth-int).
- nonce, algorithm: hash function and random nonce to be used in the client side for credential presentation (will be also used in the server side to check the reply)

The client replies with a new request with a “XXX_Authorisation” header field containing several parameters. The most important is response, which contains a hash comprising the username, password, the server’s nonce, the client’s nonce (specified in the cnonce parameter) and the request line (also the body if auth-int was chosen).

The server calculates another hash value using the same input. If the result matches with the response parameter of step 3, the request is processed; otherwise, the request will be challenge again.

The security properties of this mechanism are the following:

- The usage of random nonces prevents again reply attacks (if some malicious entity obtains the correct has value for a particular nonce, it will not be able to access the server because a different one will be used in a different challenge).
- Provides user authentication.
- Provides limited integrity protection (i.e. the request line – method and URI – and the body, if the proper qop parameter is selected). But considering that not all headers fields are protected, this schema is sensible to MitM attacks.
- DoS attack protection (during the challenge period, the server remain stateless, so an attacker can not block resources for a long time).

This strategy is commonly used in UAC because typically they will not implement advanced and secure certificate-based security mechanism. These site certificates are commonly used for network entities.

The HTTP Digest authentication drawbacks are as follows:

- Does not support message Integrity and Confidentiality.

E.4. S/MIME

S/MIME can also be used to ensure both integrity and confidentiality of SIP messages. It basically consists on a certificate-based mechanism than can be used to sign or encrypt some parts of SIP messages, typically SIP bodies. Encryption of an entire SIP message to provide end-to-end confidentiality is not appropriate because some network entities need to view certain headers (as shows Table 13) to properly route the message; however, S/MIME can provide some kind of protection for headers fields, using a mechanism named “SIP message tunnelling”, which basically consists on the encapsulation of a SIP message as the body of an “outer” SIP message. This body, containing the whole message, is then encrypted using S/MIME.

Header	Request URI	From	To	Via	Record Route	Record	Call Id	Cseq
Modification Allow	YES	NO	NO	YES	YES	YES	NO	NO

Table 14 – Encrypted headers with S/MIME

As mentioned earlier, S/MIME is a certificate-based mechanism. These certificates are associated with the keys that are used to sign or encrypt message bodies. Currently there is no a

consolidated authority today that provides certificates for end user applications, so most usual solutions are:

- Usage of pre-configured certificates previously trusted.
- Usage of self-signed certificates (that is, a certificate that cannot be verified by the recipient), which are also sent in the requests as a multipart MIME body (different sets of data combined in a single body, as defined in RFC 2046 [RFC2046]). In this case, the recipient end user application is responsible for accepting or rejecting the request, based on the information the certificate carries.

Each user agent supporting S/MIME must have a key ring for end-user certificates in order to map between AoRs and their corresponding certificates. Anyway, SIP can also be used as a mean to distribute these certificates. The S/MIME specification enable the inclusion of the certificate itself together with the proper content to be protected as part of a multipart MIME body, so when a new request including a certificate arrives, the user agent can validate it and add the new certificate to its local database.

The security that this process provides relies on the usage of a random key which is sent together with the protected content. This random key is encrypted using the public key of the recipient, so when receiving a message, it will make use of its private key to a) decrypt the content encryption key and b) use this decrypted key to decrypt the message itself. This process is depicted in the figure below:

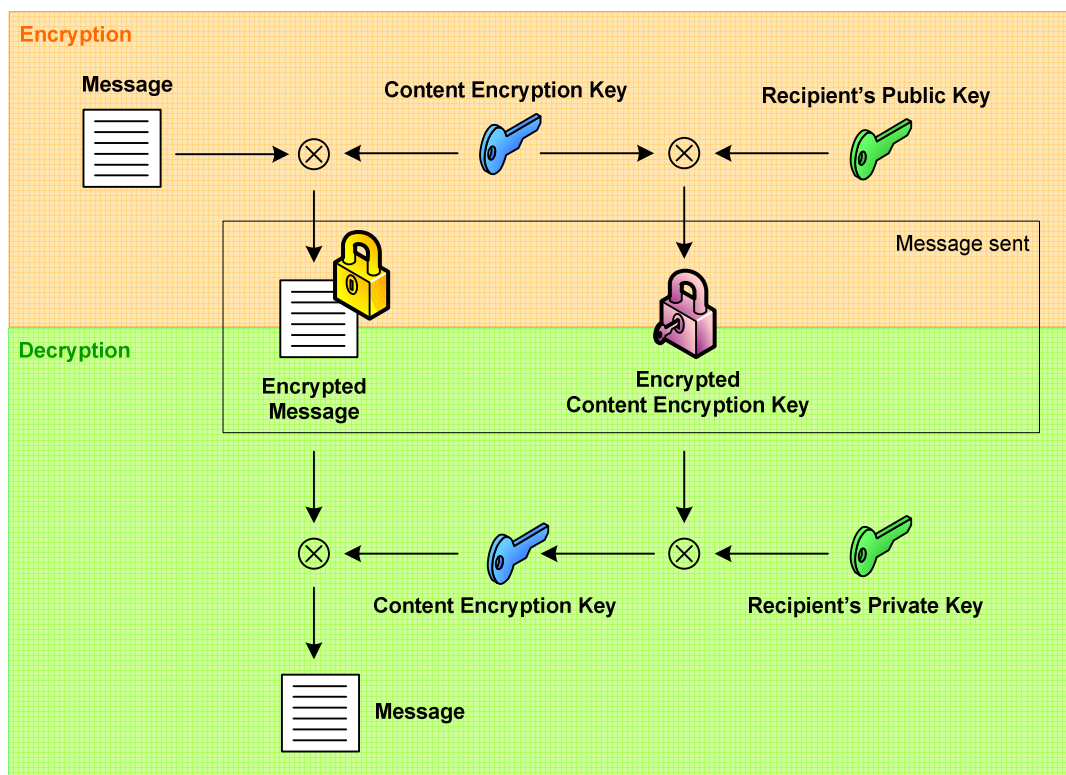


Figure 36 – S/MIME encryption and decryption

The usages of a symmetric key (the same key will be used in both sides) have a significant impact on the speed of the encryption/decryption process. Obviously, senders must have foreknowledge of the public key of recipients, so mechanisms to access this information must be implemented.

In order to provide sender's authentication, messages can be also signed. However, in this case the sender's private key is used, as next figure describes. Again, the recipient needs to know in advance the public key of the opposite side, in order to compare the decrypted message hash it

has received with the calculated one by itself; but as mentioned before, senders may include it with the message as appropriate.

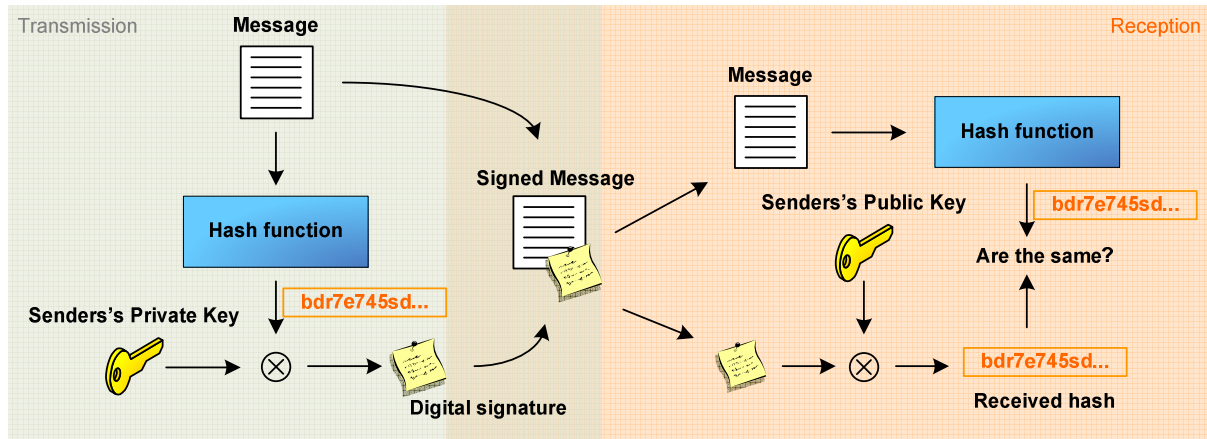


Figure 37 – S/MIME signature

Finally, just to mention that both processes can be nested: encrypted messages can be signed, and signed messages can be encrypted.

S/MIME provides the following cryptographic security services for electronic messaging applications:

- Authentication, message Integrity and non-repudiation of origin (using digital signatures), and Data Confidentiality (using encryption).
- Since S/MIME is mostly used for securing message bodies, such as the SDP rather than message headers, S/MIME avoids tampering with message bodies' attacks.

It is worth noting as drawback that S/MIME (the same than IPSec) generates considerable overhead in SIP messages, and the lack of PKI is an additional restriction for the operation of S/MIME in SIP.