European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable D2.3
Virtual Laboratory Integration Report

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
International University Bremen, IUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universitat Politecnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, CETIM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University of Surrey, UniS, UK
University of Pitesti, UniP, Romania

For more information on this document or the EMANICS Project, please contact:

Dr. Olivier Festor
Technopole de Nancy-Brabois — Campus scientifique
615, rue de Jardin Botanique — B.P. 101
F—54600 Villers Les Nancy Cedex
France
Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title: Virtual Laboratory Integration Report

Type: Public

Editor(s): Jürgen Schönwälder, Ha Manh Tran, Iyad Tumar

E-mail: j.schoenwaelder@jacobs-university.de

Author(s): WP2 Partners

Doc ID: D2.3.doc

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V0.1	May 24, 2007	Jürgen Schönwälder	Initial version
V0.2	June 1, 2007	Fabian Hensel, Gregor Schaffrath, David Hausheer	Included UniZH input
V0.3	June 8, 2007	Joan Serrat, Bala Karpagavinayagam	Included UPC input, Included INRIA input
V0.4	June 12, 2007	Jürgen Schönwälder, Ha Manh Tran	Included IUB input
V0.5	June 15, 2007	Gabi Dreo Rodosek, Frank Eyermann, Iris Hochstatter Aiko Pras	Included UniBwM/CETIM input Included UT/KTH input
V0.6	June 17, 2007	Ralf König, Feng Liu	Input LMU
V0.7	June 18, 2007	Ralf König, Feng Liu	Minor changes on trace collection text
V0.8	June 29, 2007	Jürgen Schönwälder, Ha Manh Tran, Iyad Tumar	Updated introduction, collaboration, and conclusion sections; editorial changes to improve the presentation
V0.9	July 4, 2007	Jürgen Schönwälder	Added some more introductionary texts
V1.0	July 4, 2007	Jürgen Schönwälder	Added two more trace locations to section 5.2

Table of Contents

1 Executive Summary

This third “Virtual Laboratory Integration Report”, also the last report of the first EMANICS phase, presents the on-going activities of the virtual laboratory and common test-beds work package (WP2). The work package currently entertains four activities:

1. Creation and maintenance of a VoIP management test-bed (VOIP)
2. Network management trace collection and analysis (TRACE)
3. Trace collection for network replay (REPLAY)
4. Resource usage data collection (ABLOMERS)

Comparing with the first and second “Virtual Laboratory and Integration Report”, submitted six and twelve months ago, these activities have been stretched on many aspects including the degree of collaboration, the scalability and diversity of activities and the achievement of activities. This report provides the detailed updates of these activities during the last six-month period, which further emphasize the achievement and collaboration. In particular, the VOIP project has seen several essential evolutions of the already-built VoIP infrastructure of the various partners. During the project meeting in February 15-16 in Munich, several sub-projects were defined that take advantage of the infrastructure. The TRACE project has proposed a systematic approach to trace analysis. The project has received contributions and supports from several other partners, who are interested in trace collection and analysis. The REPLAY project has made sound progress in both research on a decentralized and asynchronous protocol, namely A-GAP, and collection of real traffic traces from various locations. The main contribution of this project is a trace repository used by researchers within EMANICS but also outside of EMANICS. The ABLOMERS project has implemented an open source monitoring approach for resource management in large-scale networks, performed various experiments in a real scenario and obtained a lot of promising results.

In general, this work package has not only made significant progress according to the originally defined objectives, but also addressed several issues found in the previous deliverables:

(i) The VoIP test-bed reached a state where it can be utilized thoroughly for research activities.

(ii) The demand of good analysis tools is crucial due to the huge amount of traces and their complexity; collaboration on the creation of such analysis tools is progressing well.

(iii) Confidentiality issues needed further discussion between EMANICS partners in order to come up with effective ways to share data.

More importantly, the work package has achieved closer collaboration among the EMANICS partners involved in each project. Several partners outside EMANICS have expressed their interests in contributing to the projects and some are already collaborating with EMANICS partners, which essentially increases the degree of collaboration achieved.

2 Introduction

The objectives of this work package are the integration of existing laboratories, the establishment of a collaboration environment and the creation and maintenance of trace repositories for research and educational purposes. The first and second objectives are highlighted by constructing a Voice over IP (VoIP) test-bed and sharing trace repositories among partners, whereas developing various tools for collecting and analyzing traces and establishing trace repositories emphasize the third objective.

The last deliverable of the first EMANICS phase reports the detailed updates of the four projects. While the first deliverable focuses on defining projects, gathering partners and establishing basic collaboration environments among partners; the second deliverable concerns implementing the projects, collaborating partners and addressing standing issues, this deliverable aims at documenting the achievements of the projects, the resolution of issues and epitomizing the degree of collaboration. This deliverable not only updates the projects' activities, it also discusses the potential trends of these projects, which open up more intensive and larger projects for the second EMANICS phase in terms of integration, research and education, and collaboration.

The rest of the deliverable is structured as follows. Section 3 documents the evolution of the VoIP Management Test-bed (VOIP), concentrating on advanced configuration mechanisms, the management of dial plans, performance and security monitoring, context management, and interfacing to traditional PBX systems. The management trace collection and analysis activities (TRACE) are described in Section 4, detailing the evolution of the SNMP / NETFLOW / SIP trace collection tools and the experiments to use GRID middleware for the analysis of traces. Section 5 describes the packet trace replay support activity before Section 6 details the evaluation of the resource usage data collection (ABLOMERS) software on a large Grid computing infrastructure. Section 7 discusses the collaboration achieved in the project. The report concludes in Section 8 with some remarks on WP2 activities in the second phase of the project.

3 VoIP Management Test-bed (VOIP)

This section describes the evolution of the VoIP test-bed. Section 3.1 and 3.2 discuss the integration of the VoIP infrastructure of the University of Zurich into the test-bed and how configuration management has been automated at the University of Zurich. The two following Section 3.3 discusses the integration with ENUM while Section 3.4 presents distributed dial plan management using DUNDi. Performance monitoring and security testing tools are described in Section 3.5 before context support in VoIP networks is discussed in Section 3.6. Section 3.7 discusses how common PBX functions can be realized in a VoIP network and a short survey about VoIP usage carried out at the University of Zurich is presented in Section 3.8.

3.1 Evolution of the VoIP Infrastructure

The Communication Systems Group (CSG) at the Institute for Informatics (IFI) is maintaining the Voice-over-IP (VoIP) infrastructure at the University of Zurich (UniZH). The VoIP infrastructure consists of two servers, one serving as an experimental platform while the other provides production-use VoIP services to people working at the department. The two systems are completely independent of each other, only connected by an IAX trunk. The development and testing of new services is done on the experimental server before they are moved to the productive system. Both systems are located in the CSG test-bed. Figure 1 shows the physical setup of the infrastructure with different types of clients accessing it.

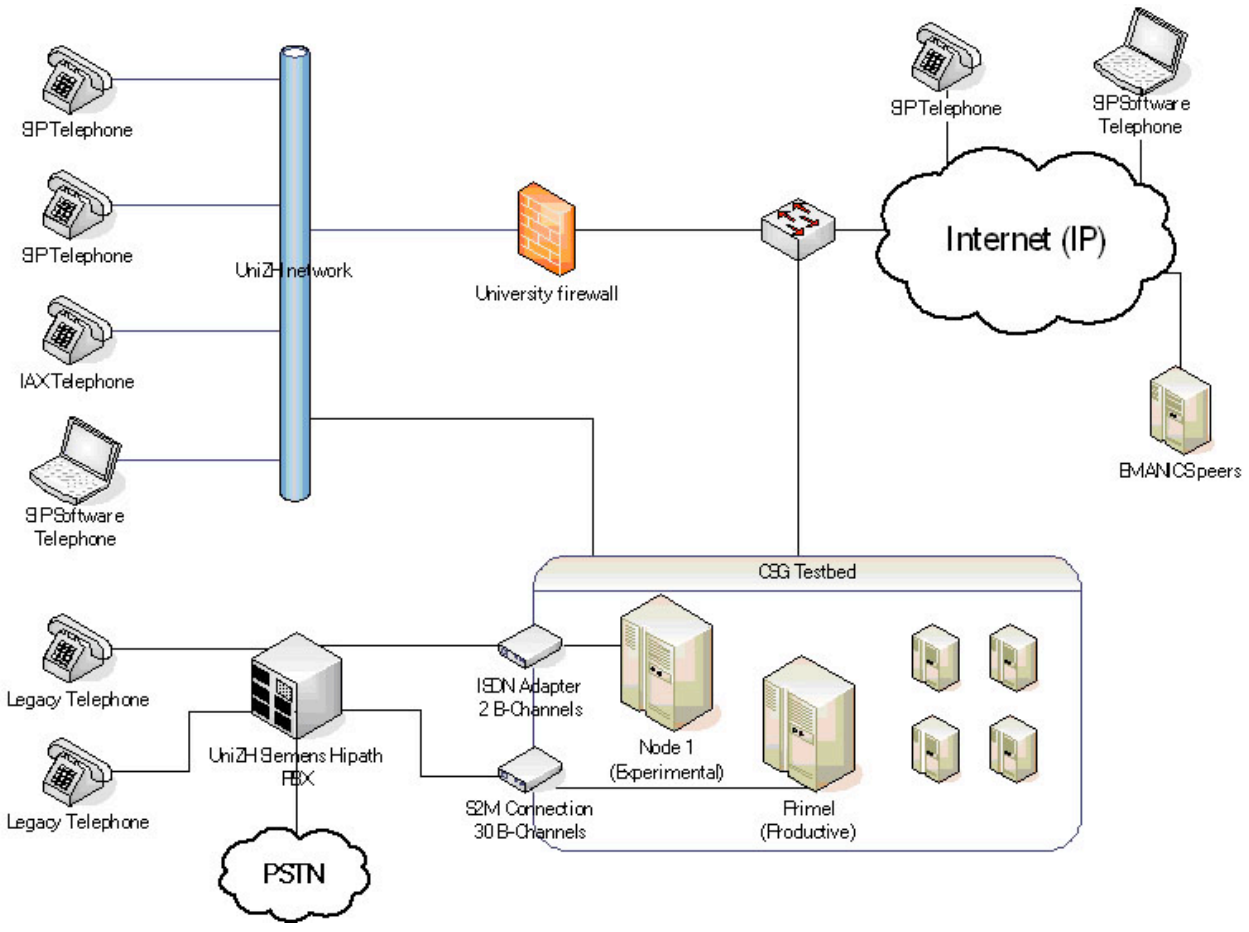


Figure 1: Physical setup of the UniZH VoIP infrastructure

3.1.1 UniZH Experimental System

The experimental server, named “Node1”, was installed in September 2006. The dedicated Dell PowerEdge 850 (Pentium 4 3.6GHz, 1GB RAM) runs Debian/GNU Linux. The Asterisk PBX software version 1.2.7 provides the VoIP capabilities. The machine has two network connections, one directly to the Swiss research network SWITCH, as well as a direct connection to the University LAN. The multihomed configuration is disregarded but currently the only solution to an issue with the central University firewall prohibiting the correct handling of SIP traffic. The machine additionally has a connection to the public switched telephone network (PSTN) via a HFC-chipset based ISDN-BRI adapter. The freely available “bristuff”-package for Asterisk provides a channel driver (zaphfc) for this card. The BRI card is connected to a Siemens Optiset ISDN Adapter of a University Siemens phone. This is necessary because the University Siemens Hipath PBX uses proprietary protocols. Limitations of this connection are a maximum of two concurrent calls from or to the PSTN and only a single public dialable phone number (not yet registered with ENUM) for the whole server. Further, there are IAX peerings with the other VoIP test-bed participants CETIM, IUB, UPI and INRIA over which successful test- and conference calls were made.

All members of the CSG group at UniZH have an account on the experimental server. Additionally, there is a varying number of test accounts. Tested hardware and software VoIP clients are: Cisco 7940 Phone, Grandstream GXP-2000 Phone, xLite, eyeBeam and iaxComm.

A number of applications have been deployed on the server:

- Every user has his voicemail, with optional e-mail notifications. The DISA service, running on the BRI-line makes the VoIP clients reachable from the PSTN.
- Several tests with Music-On-Hold, possibly making a Radio-over-Phone service feasible, were made.
- The server also implements an ENUM lookup script, first checking for alternate connection possibilities before switching to PSTN. There are no ENUM records for directing incoming calls via VoIP to the server, as there is only a single E.164 number available.
- DNS SRV records for the csg.uzh.ch domain enable incoming VoIP (SIP, IAX2) calls from third-parties.
- Some tests have been conducted with SIP Notify. This feature allows for presence indication to other users. Similar to instant messaging systems, the current status (available, busy, away) is presented to the other users. It is supported by a limited number of SIP devices such as the Grandstream GXP-2000 or the eyeBeam software.
- The installed OpenH323 package makes connections from and to H323 possible.
- Furthermore, the Meetme conference application is installed for use with other EMANICS peers.
- The call detail records (CDR) saved to a MySQL database can be accessed by the web-based “Asterisk CDR Analyzer” to generate graphical reports.

- Videotelephony is enabled by activating the H.261, H.263 and H.263+ codecs in Asterisk. Successful tests were made with xLite.

3.1.2 UniZH Productive System

The productive server, named “Primel”, was installed in February 2007. Its primary goal is to provide reliable VoIP services to the members of the UniZH Department of Informatics (IFI) and to run advanced services and management tools that have previously been tested on the experimental system at larger scale.

Hardware specifications are identical to the experimental server, only differing in the PSTN connection interface. An ISDN-PRI (S2M) interface card is used to connect to the University PBX and further on to the PSTN. The type of card used is a Sirrix PCI2E1 card, providing two independent PRI, each serving up to 30 concurrent calls simultaneously. Because the card is rather new on the market, there were several hardware compatibility issues with the PCI bus of the Dell servers which forced the use of other server hardware until those issues were resolved. On the University PBX side, a corresponding Siemens interface card is installed. With the provided EuroISDN DSS1 protocol, standard protocols are used that can easily be traced and debugged. In total, 300 public reachable (E.164) phone numbers are routed directly to the server. It was also possible to preserve the user-to-user signaling feature, allowing the transmission of the caller’s name along with the caller ID.

The productive server is running the Ubuntu Server 6.06 operating system with the newly released Asterisk version 1.4. The new version features some new functionality, but the amount of addons supporting it is still limited. Additionally, the server hosts a MySQL database and an Apache Webserver. Asterisk is configured to access parts of its configuration data directly from the database by using the Asterisk Realtime Interface instead of the configuration files. This way, changes to extensions and routing settings in the database come to effect immediately. LDAP solutions (LDAP realtime) were also tested, but the support and flexibility were found to be insufficient. Instead, it will be possible to implement a connecting logic in the future, replicating LDAP data into the MySQL database. The MySQL database will be moved to a central data storage facility in order to achieve higher security and reliability in the event of a system crash.

The productive system currently only hosts the MeetMe conferencing application together with WebMeetMe. This web frontend allows for easy scheduling and management of conferencing calls. Several conferences have been successfully hosted on the server, combining participants via SIP and PSTN. In order for MeetMe to work, a timing driver had to be installed. On the experimental server this is provided by the BRI-card and was, therefore, never an issue. The productive system lacking such a card uses the Ztdummy driver that would only work if the systems ACPI capabilities were turned off.

Billing of PSTN calls was not possible in the beginning, because no billing information (Advice of charge) is transmitted to the system. The University Siemens Hipath PBX is unable to do so. Instead of receiving this information directly with the calls, it is now collected on the Siemens PBX and regularly retrieved via FTP, further processed and then forwarded to the responsible accounting department of the University. Nevertheless, call detail records are logged without the cost parameter in the MySQL database for potential future analysis.

Tested hardware and software VoIP clients are: Cisco 7940 Phone, Grandstream GXP-2000 Phone, xLite, eyeBeam and iaxComm as well as the Nokia E61i mobile phone.

The Nokia E-Series phones have an integrated SIP client and WiFi, allowing mobile VoIP connectivity throughout the whole University campus.

3.2 Configuration Management

This section documents two approaches to manage configurations that have been developed and tested at the University of Zurich.

3.2.1 UniZH Experimental System

New applications are frequently installed, making changes in the dialplan necessary. Therefore, the experimental system stores its whole configuration data in the Asterisk configuration files.

The users configured in the system are limited to the members of the Communication Systems Group. The extensions match the ones of their legacy phones, preceded by the digit 2. Incoming calls are signaled via VoIP and the legacy phone.

In order to comply with the EMANICS specification of the VoIP test-bed, all users are also reachable under their corresponding EMANICS extension of the form 900 XXX, where XXX matches the assigned EMANICS extension. Outgoing calls to other EMANICS peers is possible by dialing the digit 9 and then the EMANICS extension of the person to be reached. A small macro translates the caller's caller id to match the specification.

The following table gives a quick overview of the experimental system's dialplan:

- 2[5-digit-LegacyPhoneExtension] - will call the registered user via SIP, IAX and his legacy phone
- 3[5-digit-LegacyPhoneExtension] - will connect to University office phones via ISDN
- [UserAlias] - UserAliases issued will be corresponding usernames on the CSG systems
- 4[InternationalPhoneNumber] - will call the international number via SIP/IAX2/H.323 if a corresponding ENUM entry is available or respond with an error signal otherwise.
- 500 - will connect to own VoiceMailBox
- 502[5-digit-LegacyPhoneExtension] - will connect to the user's VoiceMailBox
- 53[extension] - will join MeetMe voice conference as user, but not create new conferences
- 54[extension] - will join MeetMe voice conference number [extension] as administrator, allowing for more options and definition of a conference PIN to be entered upon user connect. This will create the conference, if it does not yet exist.
- 55[extension] - will join MeetMe voice conference as administrator and keep a record of the conference
- 812 - Text to speech system test
- 813 - 1000Hz 0db test tone for jitter testing
- 823 - Play "MusicOnHold", currently attached to an internet radio station
- 866 - Echo test for latency testing
- [PSTN-number] - will connect to any phone number in the public telephone network (password protected).
- 9[EMANICS-routing-number][EMANICS-extension] - will connect to people at other institutions participating in the EMANICS VoIP test-bed.

3.2.2 UniZH Productive System

The dialplan of the productive system was designed with respect to stability but with the possibility of easy user and application management. In order to gently introduce the users to the new VoIP technology without forcing them, all members of the department with a phone line got assigned an additional, separate number for VoIP. Because of the flexibility of Asterisk's call diversions features, users can choose to publish their new number and let their existing legacy phone ring together with their VoIP clients. The DISA feature on a dedicated extension can easily be programmed on a speed dial button of a legacy phone, allowing outgoing VoIP calls as well. If a user does not want to use the VoIP infrastructure at all, no action is required.

There were also a number of political issues to be fulfilled. In total there are 300 E.164 numbers available. Every research group of the Department of Informatics at the University of Zurich is assigned a block of 20 or 30 consecutive numbers. Some numbers are reserved for lecture halls and other rooms equipped with phones, the last block of 20 numbers is reserved for applications. Because these numbers are University internal extensions, calls from and to University legacy phones are free of charge, allowing a broader range of cost effective routing possibilities.

In order to offload the management of the VoIP infrastructure to the users themselves, a user friendly interface is required. As stated earlier, LDAP support in Asterisk, especially version 1.4 is insufficient for productive use whereas MySQL support is very stable and extended over the Asterisk Realtime interface. MySQL databases can also very easily be accessed by web scripting languages such as PHP, which can be used to design the user friendly web frontend to the VoIP infrastructure. Many of the already available open-source products were tested, but none would perfectly fit the needs, so an individual solution was to be implemented. The web frontend is divided into two levels: the user settings panel and the administrative control panel. The user settings panel enables each individual user to set their preferences and incoming call routing, while the administrative control panel serves the department secretary, who is in charge of the phone administration, to add and delete users.

Not all of Asterisk's settings are retrieved from the database. Currently, only the SIP user, IAX user and Voicemail user (sip.conf, iax.conf and voicemail.conf) configuration is served from the database, because these are the primary settings that have to be changed for user management. The MySQL interface of Asterisk Realtime is enabled in the "extconfig.conf" file. The fields in the respective "sippeers", "iaxpeers" and "voicemail" database tables are shown in Figure 2. They correspond to the parameters normally set in the respective configuration files and are not necessarily all set (NULL value).

The call detail records (without advice of charge information, see above) is also stored in a database table with corresponding fields, by configuring the "cdr_mysql.conf" file.

The dialplan, as the central element of call handling in the VoIP infrastructure, is primarily left in the extensions.conf file. Even though Asterisk Realtime offers this file also to be served from the database, changes are made rarely. Instead, specific settings that depend on the user and that can change often are stored in separate database tables that are queried from the dialplan with the MYSQL() command. Currently, there are five such tables configured that are queried from different contexts with the dialplan. For a reference of the defined dialplan contexts, see below.

- The “aliases” table contains the SIP URI aliases, so users can be called by their known e-mail address (enabling SIP.edu specifications). It is queried from the “sip-in” context.
- In the “groups” table the owner (research group) of extension blocks is maintained. There is a database entry for every extension. Extensions that are currently not assigned have an owner, too. Upon creating a new user, a free extension owned by the group can be selected. This parameter is mainly used for the administrator control panel.
- The “legacyphones” table serves the purpose of keeping track between each users VoIP extension and the extension of his legacy phone. This is mainly used for restricting direct access to the DISA service and, therefore, queried in the “macro-disa” macro. If the “blocked” parameter is set, access to the DISA service is denied. If only “mustauth” is set, the user must authenticate with his voicemail password. The purpose of this functionality is to restrict DISA access from public accessible phones within the institute. The table is also queried when accessing the voicemail from the legacy phone. In order to directly access the voicemail, the corresponding VoIP extension for the calling legacy phone extension is looked up, allowing for a faster access to the voicemail menu.
- The “enum” table stores whether a user wants to do an ENUM lookup when calling out or if he prefers a direct PSTN connection. This was introduced because of the long call setup time when making ENUM lookups. It is queried form the “pstn-callexternal” context.
- The “diversions” table is the central element of call routing. It is queried by a complex macro within the “voip-calluser” context. Each “extension” is assigned a number of different destinations (“destination” field). The “tech” field contains the respective technology used to reach that destination (SIP, IAX2, PSTN). The “timeout” field defines the timeout after which the call moves on to the next “priority”. The macro is implemented as two loops, the outer one processing the entries from priority 0 upwards, the inner one generating Asterisk dialstrings (dial() command parameters) of entries having the same priority and the “activated” field set to true. Users can, therefore, configure the VoIP infrastructure to let several phones ring simultaneously, when they receive a call. Further, the “fixed” parameter defines if the respective table entry can be fully edited from the user control panel (set false) or if editing the entry is limited to priority and timeout (set true). The “simultaneous” parameter is deprecated. It was used in order enable simultaneous ringing of different phones, which is now achieved by setting equivalent priority entries.

Figure 2: MySQL database design

The dialplan itself is structured into different contexts. Primarily, authenticated and unauthenticated calls are distinguished as shown in Figure 3. Further differentiated are real, special and virtual extensions. Real extensions correspond to actual E.164 numbers, reachable from the PSTN, that map onto an individual VoIP user. Special numbers are also E.164 numbers, but running special applications, such as conferencing applications or permitting voicemail access. Virtual numbers are not reachable from the PSTN and only used for call forwarding. They were introduced in order to have a single VoIP user maintain several SIP accounts, if one owns several SIP devices (e.g. VoIP enabled mobile phone, desk phone and software client). When only using one SIP account, only one device can register concurrently with Asterisk that will signal incoming calls. Users will be able to create virtual numbers by using the web frontend.

The dialplan context design is as follows:

The “pstn-in”, “sip-in” and “iax-in” are the first contexts for incoming, unauthenticated calls of the respective technology. For simplicity, the “sip-in” context is included in the “iax-in” context. The “sip-in” context features an alias lookup script, allowing calls in e-mail address format. Numeric destinations are forwarded into the “pstn-in” context, which implements a caller-ID adaption script, as the leading 0 used in the University PBX for external calls, is automatically dialed by the VoIP infrastructure. Apart from that it includes the “pstn-special”, “all-special”, “all-virtual” and “voip-calluser” contexts. The only “pstn-special” application running is DISA (“macro-disa”), because it shall only be accessible from the PSTN and not from VoIP systems. The “all-special” context defines a number of test applications (echo test, jitter test), as well as the MeetMe conferencing application. The “all-virtual” context remains empty at this time. It is only implemented for completeness, but not accessible because virtual numbers are not E.164 numbers. The “voip-calluser” context contains the database querying logic for the “diversions” table.

Authenticated SIP and IAX clients as well as authenticated users accessing the VoIP infrastructure by using their legacy phone via DISA, are placed in the Asterisk “default” context. All extensions that are accessible by unauthenticated users can also be used by authenticated users. Additionally virtual numbers from the “authed-virtual” context and special numbers restricted to VoIP users in “voip-special” are available. These two contexts are both empty at this time. Finally, the “pstn-callinternal” and “pstn-callexternal” contexts permit outgoing calls via PSTN. Internal numbers are differentiated because they can be called free of charge. The context for external calls uses the ENUM macro, if the user has enabled this feature.

There are several smaller macros implemented to ensure correct translation of the caller id name into the user-to-user signaling string used in the ISDN-PRI connection and vice-versa.

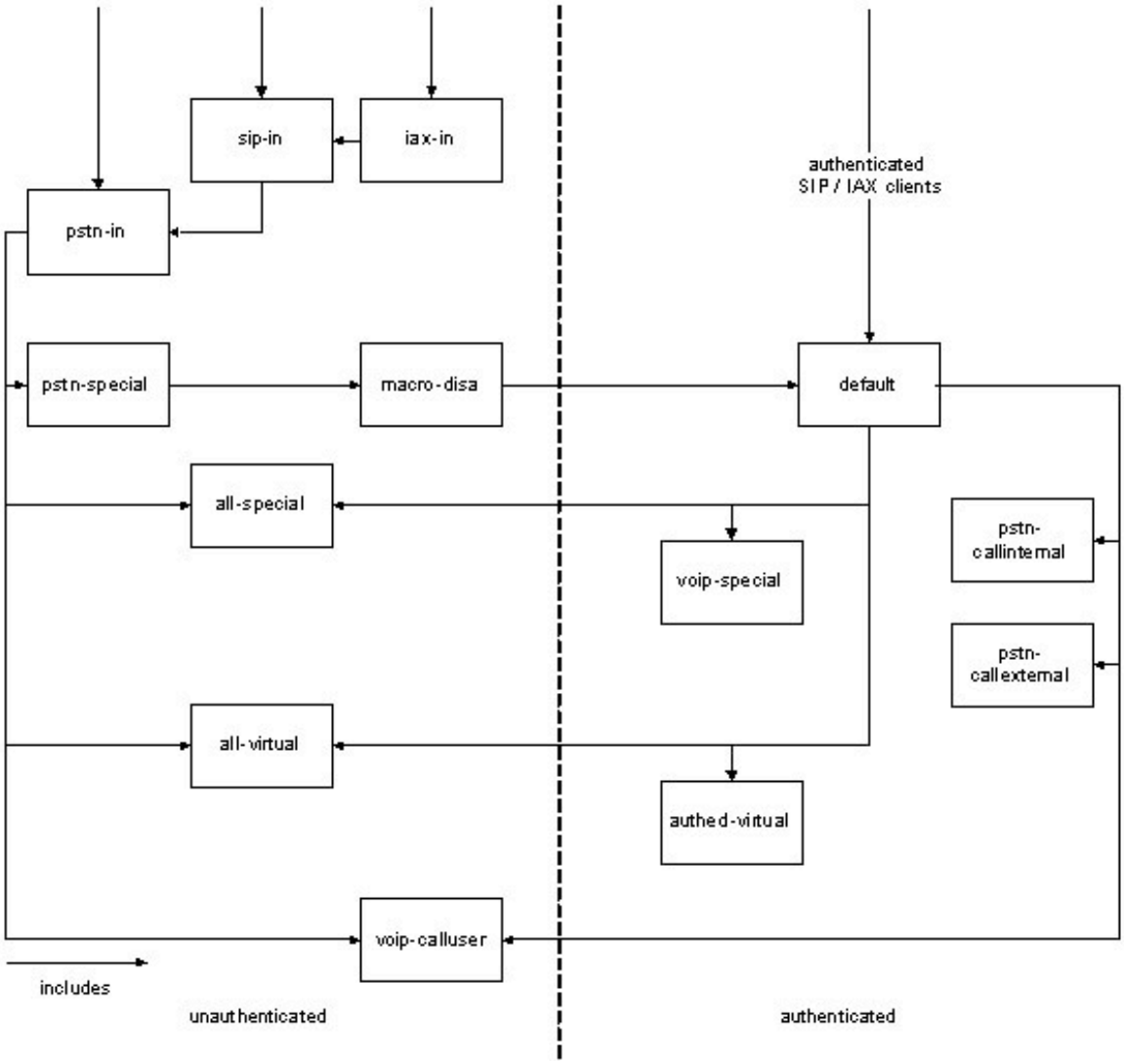


Figure 3: Asterisk dialplan contexts

The web frontend of the system does not interact directly with Asterisk. It is basically a customized frontend for the MySQL database. While basic features of the user settings panel are already implemented, the administrator control panel is not realized yet. The administrative tasks are currently maintained with the PHPmyadmin tool.

The user settings panel, shown in Figure 4, consists of four PHP scripts: Login, Logout, Settings, and Diversions. User login verification is done by checking the user input against the “name” and “secret” fields in the “sippeers” database table. This enables a Single-Sign-On (SSO) architecture for all VoIP services. The settings page displays information about the VoIP account, allows activating or deactivating the account, setting a new password, voicemail PIN (also for DISA access), the SIP alias and enabling/disabling voicemail notifications via e-mail and ENUM lookups on outgoing calls.

The diversions table, shown in Figure 5, allows the user to configure call routing options. Incoming calls are routed in the order of ascending priorities, starting with zero. Entries with the same priority ring at the same time. Each entry can be activated and deactivated (do not have to be deleted when not used). The technology field is used to select the respective protocol used for the destination (SIP, IAX2, PSTN). The timeout field specifies after what time the next priority shall be processed. If there are none left, Asterisk forwards the caller to voicemail. The “NEW” field allows making new entries in

the list; the delete checkbox is used to delete entries. The diversions table always consists of at least two entries: The SIP and IAX2 client for the extension itself. It makes no sense to deactivate or delete those entries. If no client is registered with Asterisk, they are simply skipped. Editing these fields is, therefore, limited in the frontend.

In order to ensure the correct functioning of Asterisk and prevent SQL injection, user input to the web frontend is carefully validated on client- and server-side.

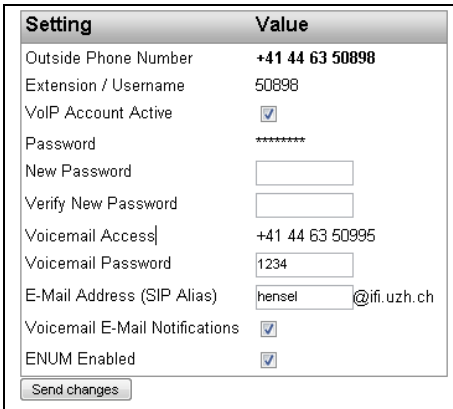


Figure 4: User settings

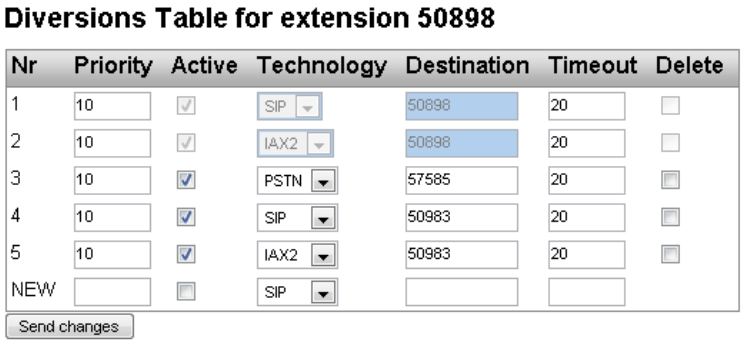


Figure 5: User diversions

3.3 ENUM Support

The University of Zurich supports ENUM mappings of phone numbers via the Domain Name System (DNS). The implementations is scetched in this section.

3.3.1 UniZH Experimental Server

Outgoing calls via ENUM are possible by predialing 4. The implemented script makes DNS NAPTR lookups in the domains e164.info, e164.org and e164.arpa. Several tests to ENUM test numbers in Austria have shown the setup working. Problematic is the long call setup time, occurring when DNS lookups are slow.

DNS SRV configuration is enabled for the csg.uzh.ch domain, both for SIP and IAX protocols. Because of the University firewall issues, two such entries had to be defined. The _sip._udp.csg.uzh.ch with priority 10 points to the University internal interface of the experimental server, while priority 20 points to the external interface. This way, SIP clients supporting SIP SRV records can automatically determine the correct interface to contact the server. Additionally, SIP SRV records allow CSG members to be reached by using SIP URI in the form SIP:alias@csg.uzh.ch.

The interfaces and SRV entry (csg.uzh.ch) priorities are the following:

```
Internal interface:

_sip._udp.csg.uzh.ch    SRV service location:
          priority       = 10
          weight         = 0
          port           = 5060
          svr hostname   = voip-gw-csg.ifi.uzh.ch
          IP             = 130.60.156.212

_iax._udp.csg.uzh.ch    SRV service location:
          priority       = 10
          weight         = 0
          port           = 4569
```

```
          svr hostname   = voip-gw-csg.ifi.uzh.ch
          IP             = 130.60.156.212

External interface:

_sip._udp.csg.uzh.ch    SRV service location:
          priority       = 20
          weight         = 0
          port           = 5060
          svr hostname   = voip-gw.csg.uzh.ch
          IP             = 192.41.135.197

_iax._udp.csg.uzh.ch    SRV service location:
          priority       = 20
          weight         = 0
          port           = 4569
          svr hostname   = voip-gw.csg.uzh.ch
          IP             = 192.41.135.197
```

3.3.2 UniZH Productive System

An identical DNS SRV configuration has been setup for the ifi.uzh.ch domain, pointing to the interfaces of the productive server. The SRV records are also a requirement for successful incoming calls via ENUM.

The productive system, hosting a total of 300 E.164 numbers, is fully ENUM enabled for incoming and outgoing calls. Delegation for the corresponding e164.arpa zones was requested and granted free of charge by SWITCH, providing DNS services for Switzerland. The NAPTR records for the three zones `7.0.5.3.6.4.4.1.4.e164.arpa`, `8.0.5.3.6.4.4.1.4.e164.arpa` and `9.0.5.3.6.4.4.1.4.e164.arpa` are set as follows:

```
8.9.8.0.5.3.6.4.4.1.4.e164.arpa naptr =
200 10 "u" "E2U+sip" "!^\\+414463(508.*)$!sip:\\1@ifi.uzh.ch!"
8.9.8.0.5.3.6.4.4.1.4.e164.arpa naptr =
100 10 "u" "E2U+iax2" "!^\\+414463(508.*)$!iax2:guest@ifi.uzh.ch/\\1!"
```

The regular expressions adapt the called number accordingly and form a SIP URI that subsequently is resolved by the SRV records. The incoming calls are then handled by the “sip-in”/”iax-in” context, cropping unnecessary digits from the number.

The interfaces and SRV entry (ifi.uzh.ch) priorities are the following:

```
Internal interface:

_sip._udp.ifi.uzh.ch    SRV service location:
          priority       = 10
          weight         = 0
          port           = 5060
          svr hostname   = voip-gw.ifi.uzh.ch
          IP             = 130.60.156.115

_iax._udp.ifi.uzh.ch    SRV service location:
          priority       = 10
          weight         = 0
          port           = 4569
          svr hostname   = voip-gw.ifi.uzh.ch
          IP             = 130.60.156.115

External interface:
```

```
_sip._udp.ifi.uzh.ch    SRV service location:
          priority       = 20
          weight         = 0
          port           = 5060
          svr hostname   = voip-gw-ifi.csg.uzh.ch
          IP             = 192.41.135.201

_iax._udp.ifi.uzh.ch    SRV service location:
          priority       = 20
          weight         = 0
          port           = 4569
          svr hostname   = voip-gw-ifi.csg.uzh.ch
          IP             = 192.41.135.201
```

3.4 Distributed Dialplan Management

The EMANICS VoIP-Test-bed consists of several Asterisk Servers which are currently operated and managed independently. In the current setup every administrator has to manually configure links to other servers, resulting in a fully meshed overlay network, and a lot of manually performed actions.

Currently, two different approaches exist to simply the location of remote resources. The older and more widely known one is ENUM (t**E**lephone **NU**mber **M**apping), defined in RFC 3761 [35]. ENUM defines the format of DNS records that point from an E.164 [36] compatible phone number to an IP address, hence a SIP server. E.164 is the identification of the ISO document defining the international public telecommunication numbering plan.

ENUM has several drawbacks. By definition ENUM can only be used for E.164 numbers, i.e. only for public phone numbers. However, the EMANICS test-bed was designed to use a proprietary numbering plan, which prohibits the use of ENUM.

Furthermore, the DNS records need to be registered in the e164.apra domain. This has to be done with a commercial provider offering registration services for this domain. The problem, however is that this is not suitable for a dynamic test-bed.

Last but not least, SPIT (Spam over Internet Telephony) is getting a problem for extensions announced in ENUM. Internet phones are very cheap to connect to making it attractive for marketers to call people and advertise their products. ENUM can provide these spammers a directory with “cheap” extensions.

To address these problems, Digium, the company that also developed Asterisk, proposed DUNDi. DUNDi [4] is the acronym for ”Distributed Universal Number Discovery” and is a protocol for resolving phone numbers into Internet resources for contacting those phone numbers. Unlike the ENUM standard, DUNDi is fully-distributed, thus eliminating the need for a central authority.

A set of nodes creates a so-called context, which is nothing else but a collection of numbers and a virtual numbering plan. Different contexts can be created for different purposes and communities. However, only the context with the name e164 is reserved for public reachable e.164 numbers and is only allowed to be used by the *DUNDi trust group*, a set of organizations which signed the General Peering Agreement (GPA) [37]. The GPA has the aim to protect the integrity of entries in the e164 context and to avoid

spam over VoIP. DUNDi facilitates strong encryption algorithms (like RSA and AES) for authentication and protection privacy.

3.4.1 DUNDi in the EMANICS Test-bed

The context “emanics” has been created to be used within the test-bed. It has to be noted that only EMANICS partners are allowed in this context. This is enforced by a RSA key authentication. All partners are allowed to search in a pool of numbers and to establish calls. This makes the dialplan smaller and easier to read because routing information is supplied by DUNDi and does not need to be configured manually.

Configuring manually would mean, that each server know the prefix and the address of all other server; resulting in a fully-meshed network of connections between server. With DUNDi, the number of connections could be drastically reduced. Only one connection to any of the other servers is necessary.

Figure 6 shows the proposed DUNDi overlay. Three servers form the DUNDi core. Each of these servers may exchange information with the other two servers. All other servers form a ring around this core and are configured to exchange information with at least two of the three servers in the core. This setup ensures availability of the DUNDi service even in case of one core server being not reachable.

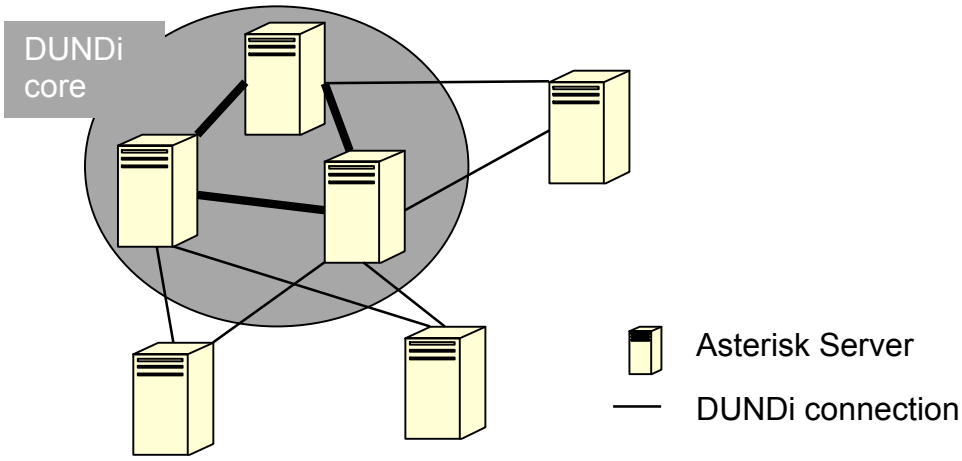


Figure 6: DUNDi overlay topology

The configuration for enabling DUNDi is shown in the following section. Basically, four steps are necessary. The first one is the generation of the RSA key for mutual authentication, followed by the definition of the own identity. Local resources are published in the third step by defining the DUNDi contexts. In the last step connections to other peers are defined.

3.4.2 Creating RSA Keys

A key pair for authentication can be generated with the `astgenkey` tool and placed in „`/var/lib/asterisk/keys/`“. Figure 7 shows the complete syntax.

```
cd /var/lib/asterisk/keys
/path/to/astgenkey -n sipgw3.informatik.unibw-muenchen.de
```

Figure 7: Creation of the RSA key pair

The flag „-n“ of `astgenkey` generates a key pair which is not encrypted with a password. If keys are password encrypted, then the password needs to be entered each time Asterisk is started. “`sipgw3.informatik.unibw-muenchen.de`” will be the name of the key.

Two files are created. The one with the extension “.key” is the private key file and has to be kept secret. The file with the extension “.pub” holds the public key and needs to be exchanged with the partners.

3.4.3 Defining the Own Identity

The own identity is defined in the general section of the file `dundi.conf`. Figure 8 shows the configuration of the UniBw server.

```
[general]
department=IIS
organization=Universitaet der Bundeswehr in Muenchen
locality=Neubiberg
country=DE
email=frank.eyermann@unibw.de
phone=+498960042404
entityid=00:13:72:5D:A4:D0
bindaddr=0.0.0.0
port=4520
cachetime=3600
ttl=32
autokill=yes
secretpath=dundi
```

Figure 8: dundi.conf general section

The first eight parameters are different for each partner. The last five should be left default. The parameter *entityid* has to be unique for all DUNDi servers; here it is set to the MAC address of the first network interface.

3.4.4 Configuring DUNDi Mapping Contexts

As already stated, for the EMANICS VoIP test-bed only one context, namely “emanics”, is used. DUNDi contexts are defined in the “mappings” section of the file `dundi.conf`. In this section, one to many contexts from the dialplan are published in one DUNDi context. The mapping furthermore defines, with which technology the published local extensions could be reached. Figure 9 shows the mappings of the UniBw server.

```
[mappings]
emanics => emanics_in,0,IAX2,emanics:${SECRET}@${IPADDR}/${NUMBER}, nounso-
      licited,nocomunsolicited,nopartial
```

Figure 9: DUNDi mappings

`emanics_in` is the context in the dialplan which is published via DUNDi. In this context all extensions are defined that should be reachable from within the EMANICS test-bed.

In the `iax.conf` file, a user named `emanics` which stores its password in the Asterisk database in the key dundi/secret needs to exist. The actual password is regenerated every 3600 sec from asterisk and advertised in the ${SECRET} variable.

3.4.5 Configuring Connections to other DUNDi Servers

Connections between DUNDi peers are configured in the file `dundi.conf`, too. For each peer that the server should exchange information with, it is necessary to include a section analogous to Figure 10.

```
[00:14:22:7B:DE:E7] ; This is the MAC/EID of UniZH
model = symmetric
host = voip-gw.csg.unizh.ch
inkey = EMANICS-UniZH
outkey = sipgw3.informtik.unibw-muenchen.de
include = emanics
permit = emanics
qualify = yes
dynamic = yes
```

Figure 10: DUNDi peer connection

The public key of the peer needs to be stored in /`var/lib/asterisk/keys/` with the filename specified under *inkey*. The own private key needs to be in the same folder; the *outkey* parameter determines the filename of the key.

When searching a particular context, the peer is included in the search if the context is mentioned in an *include* parameter. All contexts which are named in *permit* parameters may be searched by this peer.

3.4.6 Adding DUNDi to the Dialplan

In the final step, DUNDi is integrated in the local dialplan, having Asterisk search DUNDi resources when routing calls. A DUNDi-switch, that needs to be added to the context, routes outgoing calls. Figure 11 shows an example.

```
[default]
exten => _800.,1,Goto(emanics_numbers,${EXTEN:3},1)
switch => DUNDi/emanics
```

Figure 11: DUNDi switch in dialplan

In this example, outgoing calls are handled in the “default” context. Local calls (all UniBw extensions start with the prefix 800) are sent to the context “emanics_numbers”, all other destinations are sent to the DUNDi switch.

3.5 Performance and Security Management

This section documents activities that were carried-out in the test-bed for the progression of the performance and security management.

3.5.1 VoIP-IRC Bots

VoIP Security Management is in its budding stage. As a first step towards to understand the need for security management, INRIA has developed a tool that can perform different tests on a VoIP networks in an automated way, called the “VoIP-IRC (Internet Relay Chat) bot”. The concept of the tool is simple yet powerful. To have the tool operational, we first need to connect to an IRC server and create a chat room. Then we install the tool in a PC and start operating the tool via the IRC chat room, executing SIP tests or attacks.

The characteristics and description of the tools are as follows: The bot is a piece of code written in Java (1.5) that can be transported via malware, such as a virus or a worm. The bot manager can command an army of bots via the already existent infra-

structure of an IRC network (see figure below). The implementation currently supports the SIP protocol only.

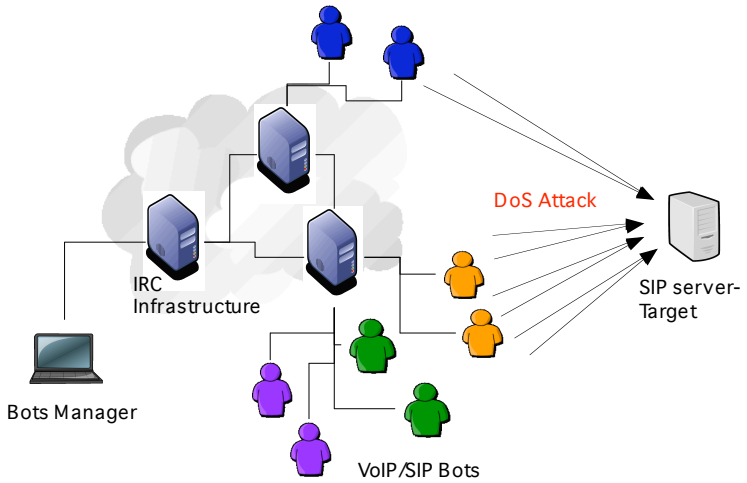


Figure 12: Reliability Check of VoIP Server

The bot supports various functionalities:

- **Automated Voice Message**: This is achieved by sending media (audio) to some SIP user. All we need to know the *SIP user name* and its *IP address* where it can be reached, i.e., the *IP address* of the SIP phone or the SIP server (domain) where the SIP user is registered.
- **User and Service Discovery:** This functionality automatically finds SIP users. This is achieved by sending INVITE messages to a list of SIP server destinations. Depending on the response obtained from the server, we identify the existing users that can be attacked.
- **Assessing Security Configuration:** This functionality involves cracking a user's password. Once an existing user name has been identified, we try to crack the password. Note that, if the user employs a digest user name which is different from his SIP username, it will be harder to crack it since we also have to know the digest user name.
- **Reliability Check:** This functionality involves executing a Denial of Service (DoS) attack. To check the reliability of VoIP entities, we send successive SIP INVITE messages with different transactions to the targets (IP phones or SIP server). In order to paralyze a SIP server, we might need to employ many bots.
- **Fraudulence Consistency:** With the above functionalities, if we find the SIP username and its respective password, we can use that information to register it ourselves. Note that, the current version of the bot can not receive calls, i.e., it is not a complete phone. In our next version, we will add to the bot the ability to transfer calls, so social engineering attacks can be analyzed.

For further information about the VoIP-IRC bot and its availability, see Table 1.

Resource	Location
VoIP-IRC bot source code	http://www.loria.fr/~nassar/javabot.zip
README file	http://www.loria.fr/~nassar/readme.html
VoIP-IRC bot discussion in the research community	http://voipsa.org/blog/2007/05/07/ready-or-not-here-come-the-irc-controlled-sipvoip-attack-bots/ http://www.blueboxpodcast.com/2007/05/blue_box_58_the.html (Audio)

Table 1: Resouce location of VoIP-IRC bot

3.5.2 Asterisk Infrastructure Monitoring based on Nagios

The goal of this work is to offer a unified view on the functioning of the VoIP test-bed from a single point. By using monitoring software like Nagios, the partners can have a clear look on the status of all VoIP servers and services in real-time. And with the help of such tools, the system administrators can be notified automatically by email or instant messenging when problems occur.

3.5.2.1 General Overview

The architectural diagram of the test-bed with the Nagios monitoring is shown below.

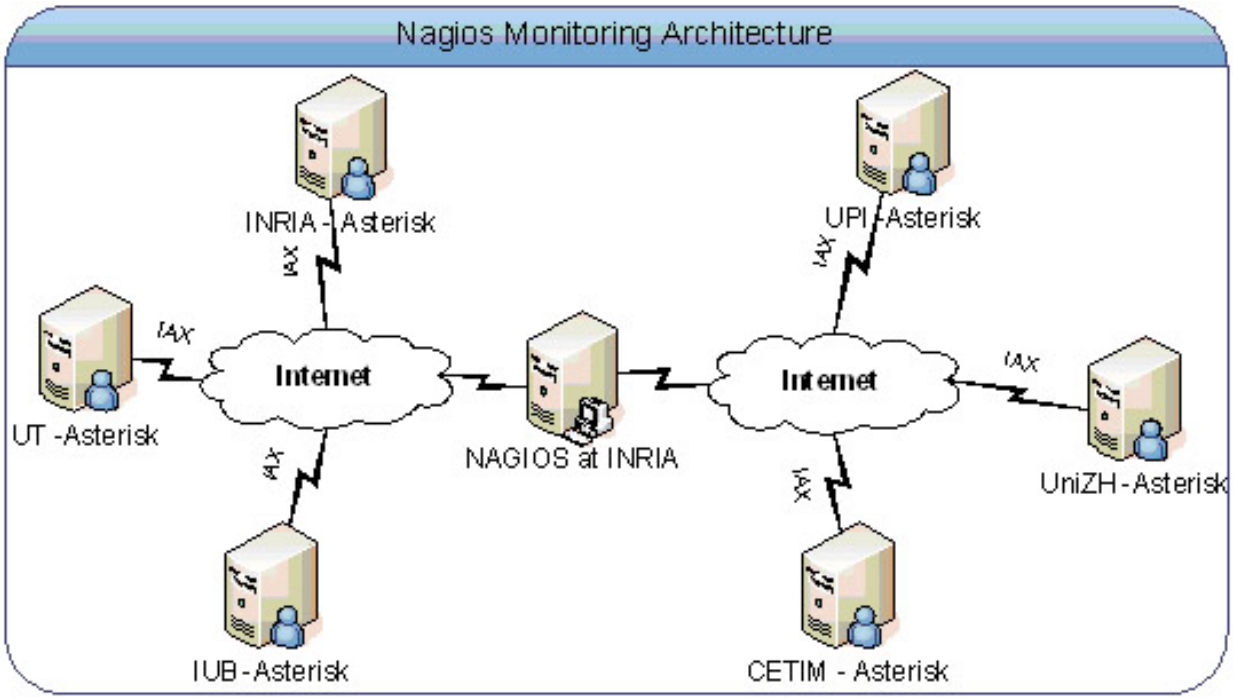


Figure 13: Nagios Monitoring Architecture

The first step in designing a monitoring infrastructure is to know the exact details of the network to be monitored: devices, topology, services, etc. The next step is to define, what information we want to obtain and what is the monitoring software expected to carry out or perform. There are many programs alike which can satisfy our goals like BigBrother, SysMon, OpenView and OpenNMS to name just a few. We however choose Nagios due to its flexibility and power.

Formerly known as NetSaint and released under the logo “only NSA monitors more”, Nagios is an application that runs as a daemon and periodically checks the services defined on the desired hosts and generates a log. Besides that, it has a nice web interface that shows clearly the state of all services, the tactical overview of the network and problems that are present at every moment. It does also show reporting information, such as availability and trends on a given period of time. It can use direct service check, SNMP and local resource verification. And more, it has the ability to check services on remote hosts by connecting through SSH or by the Nagios Remote Plugin Executor (NRPE). If network security demands this, it can act like a passive agent and receive information from plug-ins that run on the remote host and send results to the Nagios Service Check Acceptor (NSCA).

Plug-ins are distributed in a separate package, downloadable from *nagiosexchange.org*, the home for all sorts of Nagios plug-ins. The plug-ins are not developed by Ethan Galstad, the main Nagios author, but by many programmers spread across the world. The plug-ins are usually Perl scripts, named check_* and are called by the CGIs. Beside the return values for the daemon interface (0 - OK, 1 - WARNING, 2 – CRITICAL, 3 - UNKNOWN), many of them also provide performance data such as response time, bytes received, or the exact answer received from the server.

The implementation of the Nagios for monitoring the VoIP test-bed based on asterisk had two possibilities. The first one is the conventional Nagios software with a plug-in for the asterisk - configuration to be done manually. The second one is called Oreon System – an automated tool to install and configure Nagios easily. We used both methods to implement Nagios for monitoring the test-bed. Both methods were modified for the implementation of the tool in the test-bed.

To integrate the conventional Nagios software to monitor the Asterisk based VoIP test-bed, we customized one of the existing plug-ins. The plug-in has two operating modes, one for the Asterisk web interface (manager) and one for IAX connectivity. Since not all the partners have the web manager installed or provide access to the web server port from outside (the port is usually blocked for security reasons), we decided that IAX is the best since we have inter-connection between partners. The figure below shows the main user interface of the Nagios monitoring tool adapted for the test-bed.

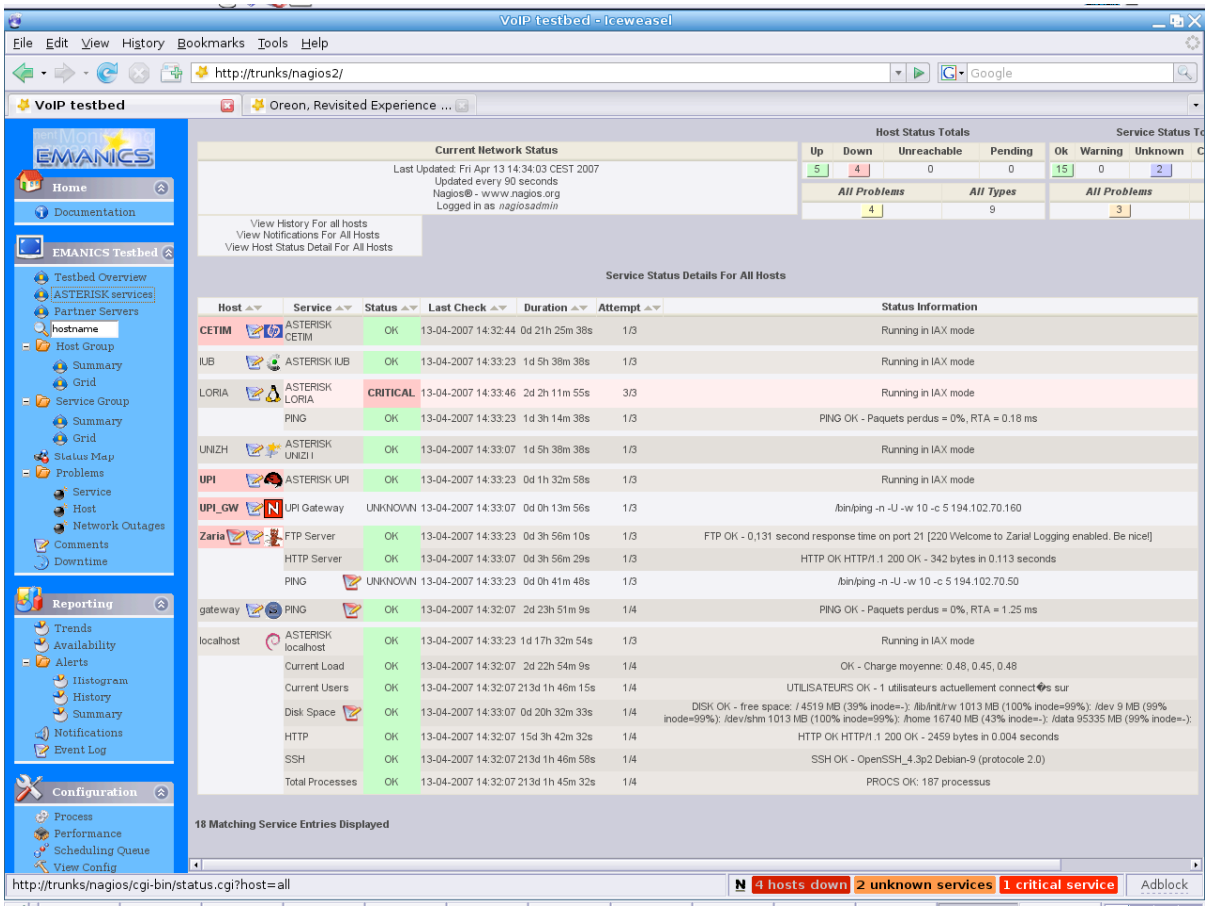


Figure 14: Nagios Monitoring Interface for VoIP test-bed

3.5.2.2 Configuration of the Monitoring tool

There are various tasks that are accomplished by the monitoring tool. The first is the monitoring of the test-bed. In order to accomplish this, we did the following configurations.

- **Hosts:** This is the configuration file where we declare the hosts (asterisk servers) to be monitored, i.e., we configure checks if the host or server is alive.
- **Services:** This configuration file controls the different service checks that should be executed on the asterisk servers, the main functionality of Nagios. Presently we only check the IAX connections. Later, if needed, we could also implement checks for other services like SIP.
- **Dependencies:** This option portrays the network structure (like a tree). Though this is not a mandatory part of the configuration of Nagios, this dependency information would be helpful particularly when the network gets expanded. It is very much useful to identify the problems quickly. The screenshot of this option can be viewed in the below diagram. However, in our present test-bed, the entities are independent, so we do not have a big tree.

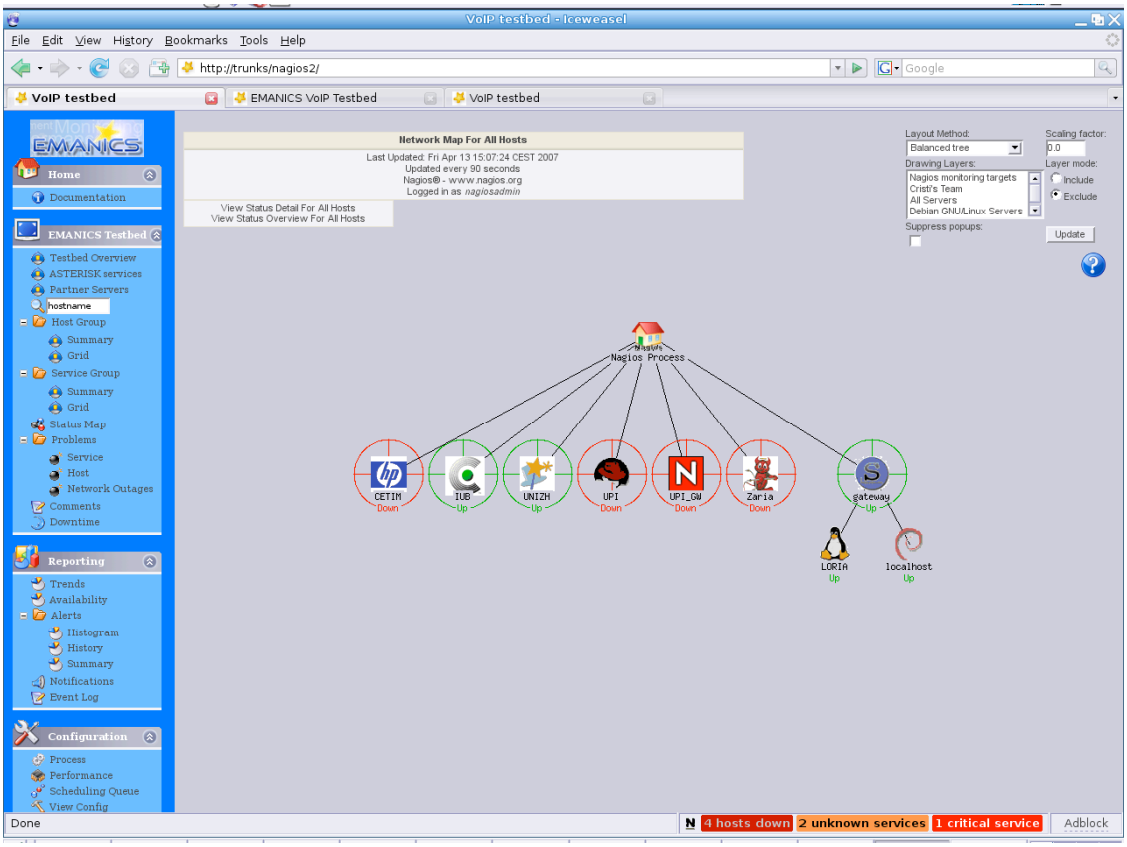


Figure 15: Dependency details shown in the tool

- **The Alerting Mechanism:** Nagios has a built-in alerting system, that can be configured to send various types of messages (email, SMS, instant message via Jabber), to react on problems. The monitoring tool was implemented to notify the administrators, in case of problems. The mechanism has various properties like
  - Alerting a single person (like administrator)
  - Alerting a group of people (partners and different sites)
  - Time periods (set time for each group or person when he could be notified)
  - Escalations (in case of persistent problems, the problem could be escalated to higher level support)
- **Statistics and Performance Report**: For the enhancement of the test-bed, the tool was integrated with other plug-in modules to collect statistics about the performance of the test-bed. With these modules, we could have the report and graph for each individual asterisk server in the test-bed. The screenshot of the performance graph for one of the asterisk server in the test-bed is given below.

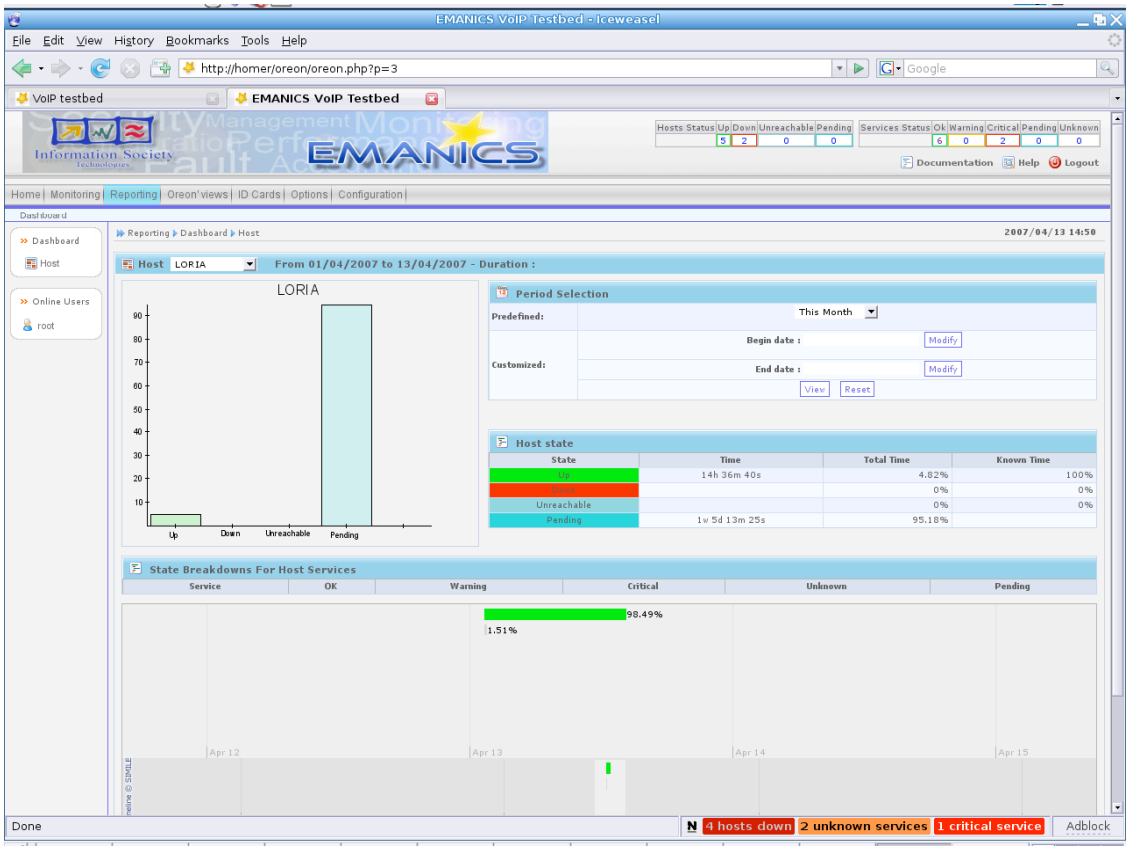


Figure 16: Performance Graph Option

Apart from the regular report on the test-bed, we could also have individual reports on the asterisk servers. Some of the services are CPU load, traffic on the host, its uptime, etc. We can generate reports based on the time for better analysis of the test-bed performance. The screenshot below shows the traffic on one of the host at various interval of time.

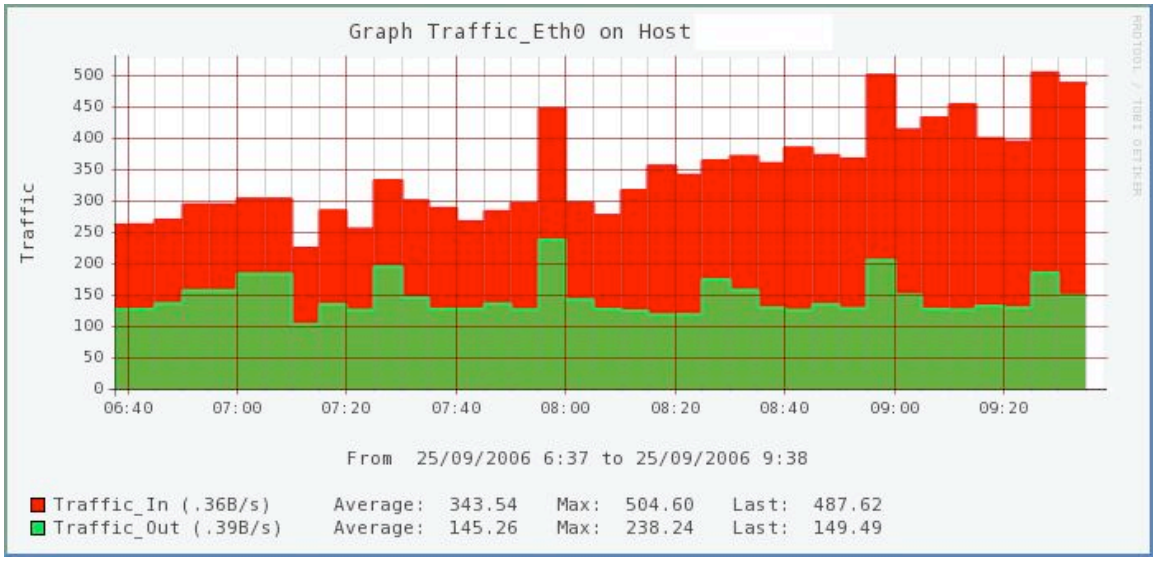


Figure 17: Traffic on a monitored host at various intervals of time

3.5.3 Summary

There were two major activities that were done as part of the Performance and security management in the VoIP test-bed. The VoIP-IRC bots was developed to better understand the security management needs in VoIP domain. The tool has already gained attention in the research community for its capabilities to test the security management issues.

The other major activity in the test-bed was the integration and implementation of a real-time monitoring tool based on Nagios for Asterisk infrastructure. This tool provides an overall performance monitoring report for the VoIP test-bed and also for each entity in the test-bed. This tool helps in understanding the issues that affect the performance of the VoIP test-bed .

3.6 Context Management

This section discusses the importance of context-awareness in VoIP networks and outlines some ideas how context-awareness can be supported in the VoIP test-bed.

3.6.1 Context and Context-Awareness

Context has traditionally been an important element of human languages. Text that has been produced in a given language does not have a distinct meaning itself, but has to be interpreted in order to eventually construct the intended meaning. This interpretation is based on what comes with the text, namely the context [32]. Schilit and Theimer first note context in the field of Ubiquitous Computing in [29] when they noticed the challenges that frequently changing execution environments posed to mobile computing. Here, context is identified mainly as location, thereby deriving information such as nearby people and objects, their identities as well as changes to these objects. Later, Schilit et. al. claim that important aspects of context are where the user is located, who the user is with, and what resources are nearby. They categorize examples of context and distinguish between [30]:

- Computing environment : available processors, devices accessible for user input and display, network capacity, connectivity, and costs of computing
- User environment : location, collection of nearby people, and social situation
- Physical environment : lighting and noise level

Dey's understanding of context includes information about the user's attention, emotional state, location and orientation, date and time, and objects and people in the user's environment [26]. Location still is probably the most important context parameter today. But definition by example is too specific as context is about an entire situation relevant to an application and its set of users. Hence, a definition of context cannot be achieved by simply listing relevant aspects of all situations possible as those aspects constantly change over time and from situation to situation as Dey and Abowd analyze in [27] . Desiring a more operational approach, they give their own, now widely used, definition:

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves.*

Dey and Abowd also categorize both, primary context types, such as location, identity,

time, and activity when characterizing the situation of a particular entity as well as secondary types. Such secondary context is derived from primary types for an entity itself as well as for other relevant entities [27]. Accordingly, all information that describes the situation of an entity is context.

Winograd emphasizes this operational approach to context and focuses on its role in communication in [32]:

> *Context is an operational term: something is context because of the way it is used in interpretation, not due to its inherent properties. (. . .) Features of the world become context through their use.*

He also differentiates between setting and context. Setting being people, places, and things that the user is situated in, and context being only the aspects that are relevant to or used in communication or human-computer interaction, respectively.

One of the first researches on context-aware computing was the work on the Active Badge Location System carried out by Want et. al. and described in [31]. The system's purpose was to determine the location of people in an office building. This was achieved by badges worn by people that provided information about their location to a centralized location service by transmitting signals through a network of sensors.

Schilit and Theimer form the term context-aware computing in [29] when they describe their active map service that keeps clients informed of changes in their environment:

> *Context-aware computing is the ability of a mobile user's applications to discover and react to changes in the environment they are situated in.*

As opposed to the earlier understanding of context-awareness, where applications, such as the Active Badge System, are simply informed about environmental information, the context-aware application here is defined as not only discovering but reacting to this context. This process of an application or a service reacting to context is also called adaptation. Context-awareness does not only mean the adaptation to context but also and more likely embraces its elementary usage. Consequently, most of the definitions generally focus on either adaptation or usage and thus can be categorized.

Pascoe et. al. have context usage in mind as they consider context-awareness the capability of a device to detect and sense, interpret and respond to aspects of a user's local environment. Therefore, their application is aware of time and location, and uses these pieces of context to derive secondary context [28].

Dey limits context-awareness in [26] to the human-computer interface. The CyberDesk system he presents uses context information in order to integrate software modules dynamically.

Other researchers such as Brown et. al. focus on the ability of an application to adapt to context. Here, context-aware applications "change their behavior according to the user's context" [25]. Later, Brown regards context-aware applications as applications that "automatically provide information and/or take actions according to the user's present context as detected by sensors" [24]. Thereby, such an application is able to tailor the process of provisioning information to the current status of the user. This includes the mere presentation of information as well as the execution of an application or the configuration of a graphical layout.

The definition of Dey and Abowd is a more general approach towards context-

awareness and implies both usage and adaptation [27]:

> *A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.*

Dey and Abowd claim that their definition is more general because it does not exclude existing context-aware applications that do not change the behavior of the application. Additionally, it only requires the response to context whereas detection and interpretation could be done by other entities.

3.6.2 Scenarios

For the integration of context information into a PBX system, many types of information can potentially be interesting to include. Thus, we will develop a generic framework, where context from any source can be included transparently. The starting points for this work are two use cases of context-aware VoIP communication: call routing to the physical location of the user and incorporating the user's social situation. Consider a normal company where employees work at their desks most of the time. They can be called there but once somebody leaves his office, all arriving phone calls are routed to his voice box irrespective whether he will be returning in a couple of minutes or months. Very popular subsets of context-aware services are location-based services, where the only context taken into account is the position of entities. A lot of research has been done on sensing the location of users. Those location sensing techniques will allow us to virtually follow an employee in a company (or any other person using the service) and forward calls accordingly to the nearest phone in his physical environment. Thus, a person can always be reached by a single phone number. Further enhancing the seamless communication service, we will use further context information about a user to be able to determine whether an incoming call will disturb him in his current situation. The presence information, nowadays often used for instant messaging services, can tell callers something about the state of mind of the callee as well as whether the call will be disturbing the callee. Combining the location of an employee with his calendar information will tell the PBX if the employee is currently in a meeting and may not be disturbed unless the urgency of the call is very high. The social relation between caller and callee will tell the PBX when and where to route the call to and which voice box message to play then the user cannot be reached.

From the scenarios described above, we will further pursue the integration of context into PBX with the example of Asterisk. More precisely, we will develop a framework that allows for easy context integration into Asterisk that can then be used with a wide variety of sensing technology and different services.

3.7 Implementing common PBX functions in Asterisk

If users should adapt the new Voice-over-IP technology, it is necessary that the new technology provides the same functions as the traditional phone system. A set of functions is common for almost all PBX systems. This section is a how-to, showing how to implement those common PBX functions in Asterisk.

The functions which are presented are:

- Blind transfer
- Attended transfer
- Call parking

- Call Pickup
- Music on Hold
- Call Forwarding (on Busy, on no Answer, unconditional)
- Direct Line
- Do not disturb
- Speed dial

3.7.1 General Guidelines

Asterisk supports a wide variety of channels, i.e. technologies to connect to phones, networks, other PBXs or the PSTN. The most popular channel types include ZAP (to connect to analogous and digital phone networks, CAPI (Common ISDN Application Programming Interface) to connect to ISDN networks, SIP (Session Initiation Protocol) for Voice-over-IP calls and IAX (Asterisk Interchange Protocol), an Asterisk-proprietary Voice-over-IP protocol.

Somebody trying to integrate different channel types in a dynamic dialplan will quickly note that each channel type has its own syntax. In general it is not sufficient to only store a number (e.g. to forward a call to). In order to dynamically produce the right syntax for the dial command, it is necessary to also know the channel type.

This extra information can be omitted, if the channel type is codified in the extension numbering plan. For example, all extensions reachable via SIP start with 4, all extensions reachable via ZAP start with 5.

The following examples assume such a numbering of extensions. Besides the easier handling, another great benefit is that users do not need to be aware of the channel type. They only know their phone number, as they are used to.

3.7.2 Blind Transfer

Blind transfer is the passing of a call without notifying the receipt. The caller is immediately disconnected from the original caller after the transfer is initiated.

This function is already built into Asterisk. The key combination to initiate the transfer is set in the file features.conf:

```
…
[featuremap]
blindxfer => #1
…
```

Afterwards it needs to be activated in the dialplan (file extensions.conf):

```
…
[globals]
…
DYNAMIC_FEATURES => blindxfer
…
```

In order to transfer a caller to extension *number*, the callee needs to press #1*number* on his phone.

Please note the timeout of 500ms between two consecutive digits. This requires a rather quick typing. If the timeout occurs between the first and second digit (i.e. between # and 1) nothing will happen, as Asterisk will not recognize the sequence. If the timeout

happens somewhere later in the number, the caller will be transferred to the number entered so far.

The timeout can be change in the file features.conf. The following example sets the timeout to 1000ms:

```
…
[general]
…
featuredigittimeout = 1000; timeout in ms between two consecutive digits.
…
```

3.7.3 Attended Transfer

When using attended transfer, the callee and the transferee can talk to each other, before the actual call is transferred. The call is transferred to the transferee only after the callee hung up. It is also possible, that the transferee does not want to take the call. In this case the callee can get back to the caller.

Configuring Attended transfer is analogous to blind transfer. The default key combination is *2:

```
…
[featuremap]
…
atxfer => *2
…
```

The activation in the dialplan is shown in the next figure:

```
…
[globals]
…
DYNAMIC_FEATURES => atxfer
…
```

If more than one function listed in “[featuremap]” should be activated, they need to be separated with a # sign in the dialplan:

```
…
[globals]
…
DYNAMIC_FEATURES => blindxfer#atxfer
…
```

3.7.4 Call Parking

Call parking requires blind transfer to be enabled and activated. The process of parking and retrieving a call is:

- The callee transfers the caller to the park extension (e.g. #1700, if the sequence for blind transfer is #1 and the park extension is 700). There is only one park extension all callers to be parked are sent to.
- The system determines an available park position, parks the caller and announces the park position of the caller to the callee
- The callee can get back to the caller by calling the caller’s park position

For configuring blind transfer refer to Section 3.7.2. The park extension and the extensions for the park positions are configured in the file features.conf:

```
[general]
parkext => 700
```

```
parkpos => 701-720      ;20 positions to park calls. Range may be extended
context => parkedcalls
;parkingtime => 45      ; max time in sec. to be parked
```

The parkingtime parameter sets an upper bound for the time a caller might be parked. If this time is expired, the caller will be automatically connected to the original caller. The context parameter defines a context, which has to be included in the dialplan. Include this context in the dialplan wherever the park extension and the park positions should be reachable.

```
 [default]
…
include => parkedcalls
…
```

Parked calls can be retrieved from any phone connected to the asterisk system.

3.7.5 Call Pickup

The Asterisk Pickup Application enables a user to pickup any ringing phone. Therefore the user needs to dial an escape sequence and the number of the phone ringing. The following example defines ** as this escape sequence:

```
[default]
…
exten => _**4X.,1,PickupChan(SIP/${EXTEN:2})
exten => _**5X.,1,PickupChan(ZAP/g1/${EXTEN:2})
exten => _**X.,2,Hangup
```

Please note, as defined in Section 3.7.1 all extensions starting with 4 are connected via SIP, all extensions starting with 5 via ZAP.

3.7.6 Music on Hold

In order to ease probable waiting periods, it is possible to play music to callers. Which music to play is configured in the file musiconhold.conf

```
[default]
…
mode = files
directory = / var/lib/asterisk/mohmp3/
random = yes
```

Music to play out has to be stored in the folder defined by the directory directive (here /var/lib/asterisk/mohmp3/). The default file type for music on hold is gsm.

3.7.7 Call Forwarding

Three cases for call forwarding are defined in this howto:

- no Answer: after trying some time to reach an extension, an alternative number is tried
- busy: if an extension is busy, the caller is connected with an alternative number
- unconditional: a caller is immediately connected to an alternative number

If, and how a call should be forwarded could be determined by the user himself. This prohibits a static configuration in the dialplan. Instead a state is stored in the Asterisk database. Depending on this state actions are performed.

The state may be set by dialling special numbers. The following prefixes are assigned in this howto:

- *91: Deactivate call forwarding on busy
- *91*number*: Forward calls when busy to extension *number*
- *92: Deactivate call forwarding on no answer
- *92*number*: Forward calls on no answer to extension *number*
- *93: Deactivate unconditional call forwarding
- *93*number*: Forward all calls immediately to extension *number*
- *94: Deactivate do not disturb mode
- *941: Forward all calls immediately to extension voicemail box
- *942: Announce callers you are on do not disturb mode

Setting of state is handled by the following entries in the dialplan:

```
;Set call forwarding on busy
exten => _*91,1,DB_DELETE(fwdstatBusy/${CALLERID(num)})
exten => _*91,2,Playback(ForwardOnBusyDel)
exten => _*91X.,1,Set(DB(fwdstatBusy/${CALLERID(num)})=${EXTEN:3})
exten => _*91X.,2,Playback(ForwardOnBusyActivated)
exten => _*91.,3,Hangup

; Set call forwarding on no answer
exten => _*92,1,DB_DELETE(fwdstatNA/${CALLERID(num)})
exten => _*92,2,Playback(ForwardOnNADel)
exten => _*92X.,1,Set(DB(fwdstatNA/${CALLERID(num)})=${EXTEN:3})
exten => _*92X.,2,Playback(ForwardOnNAActivated)
exten => _*92.,3,Hangup

; Set unconditional call forwarding
exten => _*93,1,DB_DELETE(fwdstatAll/${CALLERID(num)})
exten => _*93,2,Playback(ForwardOnAllDel)
exten => _*93X.,1,Set(DB(fwdstatAll/${CALLERID(num)})=${EXTEN:3})
exten => _*93X.,2,Playback(ForwardOnAllActivated)
exten => _*93.,3,Hangup

; Set DND (do not disturb), i.e. forward all calls to Mailbox
exten => _*94,1,DB_DELETE(fwdstatDND/${CALLERID(num)})
exten => _*94,2,Playback(ForwardOnDNDDel)
exten => _*941,1,Set(DB(fwdstatDND/${CALLERID(num)})=1) ; to mailbox
exten => _*942,1,Set(DB(fwdstatDND/${CALLERID(num)})=2) ; only announcement
exten => _*94X,2,Playback(ForwardOnDNDActivated)
exten => _*94.,3,Hangup

; Delete state
exten => _*90,1,DB_DELETE(fwdstatBusy/${CALLERID(num)})
exten => _*90,n,DB_DELETE(fwdstatNA/${CALLERID(num)})
exten => _*90,n,DB_DELETE(fwdstatAll/${CALLERID(num)})
exten => _*90,n,DB_DELETE(fwdstatDND/${CALLERID(num)})
exten => _*90,n,Playback(fwdDelAll)
exten => _*90,n,Hangup
```

In the Asterisk database family “fwdstatBusy” holds all extensions, for which call forward on busy is set. The key for the database is the extension; the value stored at this key is the number to forward to. The Asterisk database family “fwdstatNA” and “fwdstatAll” work analogous for call forwarding on no answer and unconditional call forwarding respectively.

If a user does not want to be disturbed he has two options. Dialling *941 on his phone will result in all calls be redirected to his voicebox. If the user has dialled *942, all callers will be announced, that the user is not available at the moment.

Several sound files are mentioned in the dialplan shown above. These sound files need to be generated and must exist.

Before a caller is connected to a callee, the state of the callee has to be checked. The following dialplan fragment checks the state and establishes connections respectively.

```
; Test for unconditional call forwarding
exten => _[01234568].,1,GotoIf(${DB_EXISTS(fwdstatAll/${EXTEN})}:10)
exten => _[01234568].,2,Goto(dial_out,${DB(fwdstatAll/${EXTEN}),1}

; Then try to call
exten => _[01234568].,10,Goto(dial_out,1)

[dial_out]
; Test for DND
exten => _X.,1,Set(nastat=${DB(fwdstatNA/${CALLERID(num)})})
; nastat is empty if forwarding is not set
exten => _X.,2,Set(timeoutNA=${IF($["foo${nastat}"="foo"]:45)})
exten => _X.,3,GotoIf(${DB_EXISTS(fwdstatDND/${EXTEN})}?DND,${EXTEN},1)
exten => _4X.,4,Dial(SIP/${EXTEN},$timeoutNA,tj)
exten => _5X.,4,Dial(ZAP/g1/${EXTEN},$timeoutNA,tj)
; Dial only returns if timeout is set
exten => _X.,5,Set(timeoutNA=)
exten => _X.,6,Goto(dial_out,$nastat,3)  ;respect DND, but no call forwarding
; Number is busy
exten => _X.,105,Set(busystat=${DB(fwdstatBusy/${CALLERID(num)})})
exten => _X.,106,Set(timeoutNA=)
exten => _X.,107,GotoIf($["foo${busystat}"="foo"]:${busystat},3))
exten => _X.,108,Busy()

[DND]
exten => _X.,1,Set(dndstat=${DB(fwdstatDND/${CALLERID(num)})})
exten => _X.,2,GotoIf($[${dndstat}=1]:10)
exten => _X.,3,VoiceMail(${EXTEN})
exten => _X.,10,PlayBack(userDND)
```

In the first step, it is checked whether unconditional call forwarding is set. If yes, Asterisk directly tries to connect to the number from the database. If no database entry exists, the original number is tried.

In the dial_out context it is checked first if call forwarding on no answer is activated. The variable nastat will hold the number a caller should be forwarded to. The variable timeoutNA is set to “45” if forwarding on no answer is enabled, otherwise the variable is empty. If the timeout is blank, the dial command will ring the channel infinitely, otherwise for the given amount of seconds.

Before a user is dialed, his DND-status is checked and respective actions are performed. As dialling depends on the channel type, different dial commands are used. If timeoutNA is set and the caller does not take the phone within 45 seconds, the dial command will exit and it will be tried to contact the new number. In order to avoid infinite loops, no check for possible call forwarding is performed, but DND is checked.

If an extension is busy, the processing of the dialplan continues at priority 105. If a number to try is stored in the database is number will be called.

In the DND context, a caller is sent to the voicebox or is played out an announcement respectively.

3.8 Tests and Surveys

In order to determine the interest in VoIP and related technologies before implementing the productive system, the CSG conducted a survey on all members of the Department of Informatics of the University of Zurich. 49 people out of a possible 130 filled out the questionnaire, resulting in a quite good feedback. The survey leads to the following results:

Everyone has already heard or read about VoIP, while more than half of the participants have already used VoIP technology. 30% are using it daily and 15% is familiar with the technical details. For 75%, Skype is their preferred VoIP tool. MSN Messenger and Google Talk are used by more than 10%, while other SIP and H.323 clients are used only by a minority. According to this survey, open standard VoIP infrastructures are used rather rarely, while proprietary solutions are spread widely. The most frequently used functionality on legacy phones is call diversion, followed by voicemail.

A number of features were proposed that the participants had to rate, considering their usefulness. The following features had an above average rating (highest ratings first):

- Reverse “Helpdesk” loop: Users can hang up as soon as they are in the queue. The VoIP infrastructure detects when the call agent is available at which moment the phone rings, allowing the continuation of the call. This way the user is not distracted during the waiting time.
- Web administered conference calls
- One-number-concept - Always reachable with the same telephone number, regardless of location
- Presence notification (buddylists)
- Setting up a phone call with a chat partner when using instant messaging
- Calling by using e-mail addresses instead of phone numbers
- Video telephony and video conferencing
- Voicemail with e-mail notification
- Filtering rules and profiles, as commonly used to sort incoming e-mail
- Automatic diversion by querying user’s electronic calendar
- Notification of incoming calls via instant messaging in do-not-disturb situations
- Privacy feature: Let callers without caller id identify themselves before the phone rings.

For using the systems diversion features, for most participants it is sufficient to filter incoming calls by caller name, number, date/time and their current location. As opposed to how the VoIP infrastructure is currently implemented, more than 75% of the participants wish to have VoIP functionality transparently available in the already existing legacy phone system, although most of them accept having to press a speed-dial button with the preprogrammed DISA access number to access the VoIP infrastructure.

For more than half of the participants, it is not acceptable to have lower than mobile phone quality when using VoIP. Therefore the use of G.711 codecs should be considered where possible.

In general, quite a number of people are interested in the subject and committed additional ideas on how the infrastructure could be enhanced. Context awareness features, privacy considerations, emergency number availability and stability were the main comments given in the free text part of the survey.

4 Network Management Trace Collection and Analysis (TRACE)

This section documents the progress made on various network management trace collection and analysis activities during the last six months. Section 4.1 documents formats used to exchange SNMP traces and defines a relational database schema for storing intermediate analysis results. The following Section 4.2 describes the NETFLOW traces that are made available by the Leibniz Computing Centre in Munich while Section 4.3 describes a SIP trace collection activity and supporting tools. Finally, Section 4.4 describes a GRID infrastructure for trace analysis based on the GLOBUS toolkit.

4.1 SNMP Trace Collection and Analysis

This section documents the progress made in during the last few months towards a common platform for the analysis of SNMP traces. In addition, work continues to collect traces from more operational production networks.

4.1.1 XML Format

SNMP messages are BER encoded and therefore somewhat difficult to process. An XML representation has been developed, which keeps all information associated with SNMP messages. The XML format is formally specified in RELAX NG compact notation:

```
# Relax NG grammar for the XML SNMP trace format.
#
# Published as part of RFC XXXX.
#
# Note that we do not use the IANA namespace registry since RFC 3688
# seems to restrict it to IETF documents (and this specification is
# originating from the IRTF).
#
# $Id: snmptrace.rnc 2375 2007-06-28 20:03:29Z schoenw $

default namespace = "http://www.nosuchname.net/nmrg/snmptrace"

start =
  element snmptrace {
    packet.elem*
  }

packet.elem =
  element packet {
    element time-sec  { xsd:unsignedInt },
    element time-usec { xsd:unsignedInt },
    element src-ip    { ipaddress.type },
    element src-port  { xsd:unsignedInt },
    element dst-ip    { ipaddress.type },
    element dst-port  { xsd:unsignedInt },
    snmp.elem
  }

snmp.elem =
  element snmp {
    length.attrs?,
    message.elem
  }
```

```
message.elem =
  element version   { length.attrs, xsd:int },
  element community { length.attrs, xsd:hexBinary },
  pdu.elem

message.elem |=
  element version { length.attrs, xsd:int },
  element message {
    length.attrs,
    element msg-id         { length.attrs, xsd:unsignedInt },
    element max-size       { length.attrs, xsd:unsignedInt },
    element flags          { length.attrs, xsd:hexBinary },
    element security-model { length.attrs, xsd:unsignedInt }
  },
  usm.elem?,
  element scoped-pdu {
    length.attrs,
    element context-engine-id { length.attrs, xsd:hexBinary },
    element context-name      { length.attrs, xsd:string },
    pdu.elem
  }

usm.elem =
  element usm {
    length.attrs,
    element auth-engine-id    { length.attrs, xsd:hexBinary },
    element auth-engine-boots { length.attrs, xsd:unsignedInt },
    element auth-engine-time  { length.attrs, xsd:unsignedInt },
    element user              { length.attrs, xsd:hexBinary },
    element auth-params       { length.attrs, xsd:hexBinary },
    element priv-params       { length.attrs, xsd:hexBinary }
  }

pdu.elem =
  element trap {
    length.attrs,
    element enterprise        { length.attrs, oid.type },
    element agent-addr        { length.attrs, ipv4address.type },
    element generic-trap      { length.attrs, xsd:int },
    element specific-trap     { length.attrs, xsd:int },
    element time-stamp        { length.attrs, xsd:int },
    element variable-bindings { length.attrs, varbind.elem* }
  }

pdu.elem |=
  element (get-request | get-next-request | get-bulk-request |
           set-request | inform-request | snmpV2-trap | response | report) {
    length.attrs,
    element request-id        { length.attrs, xsd:int },
    element error-status      { length.attrs, xsd:int },
    element error-index       { length.attrs, xsd:int },
    element variable-bindings { length.attrs, varbind.elem* }
  }

varbind.elem =
  element varbind { length.attrs, name.elem, value.elem }

name.elem =
  element name { length.attrs, oid.type }
```

```
value.elem =
  element null              { length.attrs, empty } |
  element integer32         { length.attrs, xsd:int } |
  element unsigned32        { length.attrs, xsd:unsignedInt } |
  element counter32         { length.attrs, xsd:unsignedInt } |
  element counter64         { length.attrs, xsd:unsignedLong } |
  element timeticks         { length.attrs, xsd:unsignedInt } |
  element ipaddress         { length.attrs, ipv4address.type } |
  element octet-string      { length.attrs, xsd:hexBinary } |
  element object-identifier { length.attrs, oid.type } |
  element opaque            { length.attrs, xsd:hexBinary } |
  element no-such-object    { length.attrs, empty } |
  element no-such-instance  { length.attrs, empty } |
  element end-of-mib-view   { length.attrs, empty }

# The blen attribute indicates the number of bytes used by the BER
# encoded tag / length / value triple. The vlen attribute indicates
# the number of bytes used by the BER encoded value alone.

length.attrs =
  ( attribute blen { xsd:unsignedShort },
    attribute vlen { xsd:unsignedShort } )?

oid.type =
  xsd:string {
    pattern =
      """[0-2](\.[0-9]+)+"""
  }

# The types below are for IP addresses. Note that SNMP's buildin
# IpAddress type only supports IPv4 addresses; IPv6 addresses are only
# introduced to cover SNMP over IPv6 endpoints.

ipv4address.type =
  xsd:string {
    pattern =
      """[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"""
  }

ipv6address.type =
  xsd:string {
    pattern =
      """(([0-9a-fA-F]+:){7}[0-9a-fA-F]+)|(([0-9a-fA-F]+:)*[0-9a-fA-
F]+)?::(([0-9a-fA-F]+:)*[0-9a-fA-F]+)?"""
  }

ipaddress.type = ipv4address.type | ipv6address.type
```

4.1.2 CSV Format

The comma separated values (CSV) format has been design to capture only the most relevant information about an SNMP message. The CSV format uses the following fields:

1. Time-stamp in the format seconds.microseconds since 1970, for example "1137764769.425484".
2. Source IP address in dotted quad notation (IPv4), for example "127.0.0.1", or compact hexadecimal notation (IPv6), for example "::1".
3. Source port number represented as a decimal number, for example "4242".

4. Destination IP address in dotted quad notation (IPv4), for example "127.0.0.1", or compact hexadecimal notation (IPv6), for example "::1".
5. Destination port number represented as a decimal number, for example "161".
6. Size of the SNMP message (a decimal number) counted in bytes, for example "123". The size excludes all transport, network, and link-layer headers.
7. SNMP message version represented as a decimal number. The version 0 stands for SNMPv1, 1 for SNMPv2c, and 3 for SNMPv3, for example "3".
8. SNMP protocol operation indicated by one of the keywords get-request, get-next-request, get-bulk-request, set-request, trap, snmpV2-trap, inform-request, response, report.
9. SNMP request-id in decimal notation, for example "1511411010".
10. SNMP error-status in decimal notation, for example "0".
11. SNMP error-index in decimal notation, for example "0".
12. Number of variable-bindings contained in the varbind-list in decimal notation, for example "5".
13. For each varbind in the varbind list, three output elements are generated:
    a. Object names given as object identifiers in dotted decimal notation, for example "1.3.6.1.2.1.1.3.0". Object names are separated by commas.
    b. Object base type name or exception names, that is one of the following: null, integer32, unsigned32, counter32, counter64, timeticks, ipaddress, octet-string, object-identifier, opaque, no-such-object, no-such-instance, and end-of-mib-view.
    c. Object values are printed as numbers if the underlying base type is numeric. IPv4 addresses are printed in the usual decimal notation and IPv6 addresses in the usual hexadecimal notation. Octet string values are printed in hexadecimal format while object identifiers are printed in dotted decimal notation. Exceptions are encoded by their name, that is no-such-object, no-such-instance, and end-of-mib-view.

Note that the format does not preserve the information needed to understand SNMPv1 traps. It is therefore recommended that implementations are able convert the old SNMPv1 trap format into the new trap format used by SNMPv2c and SNMPv3, according to the rules defined in RFC 3584. The activation of trap format conversion should be the user's choice.

4.1.3 Trace Naming Conventions

A trace's name is determined by the location (organization) it was taken from and the number of other traces taken at that location. Each location has a number assigned to it, which uniquely identifies it. Formally, a trace name has the following form:

l<location number>t<trace number>

Numbering starts with 1. A central registry is used to assign location numbers and location specific trace numbers. For example, the first trace taken on location 6 would have the name “l06t01”.

At the next lower level, the SNMP flows are named based on the trace they are part of,

the type of the flow, and the IP addresses of the endpoints. We currently distinguish two flow types: command flows, which are initiated by a command generator (or manager), and notification flows, which are initiated by a notification originator (or agent). Formally, a flow name has the following form:

<trace name>-<cg|no>-<IP of initiator>-<IP of responder>

As an example, the command flow between 192.168.1.1 and 192.168.1.123 of trace 'l06t01' will be named:

l06t01-cg-192.168.1.1-192.168.1.123

In the database we keep trace and flow information in two fields, named "trace_name" and "flow_name", which are common to most of the tables (see below).

On disk, all files belonging to a trace are kept under the same directory, named after the name of the trace. This directory includes the original trace data in PCAP format and CSV format and additional files generated by the processing scripts, containing statistics or SQL statements that can be imported into the database. Since traces can be split into flows, a "flows" directory is used for storing all flow files.

4.1.4 Database Schema for Intermediate Results

We have created a database named "snmptrace" using MySQL as a database server to store the information computed by some of our scripts, like "snmpwalks.pl" and "snmpstats.pl". The major advantage of using a database is that we can easily extract data by running different queries on the data stored.

To keep the information consistent and handle the cases where a script is run more than one time, we label every record from the database with a trace or flow name, so that whenever we need to re-run a script that should update the database information for a particular trace or flow, we know exactly what records to replace.

The diagrams presented in this and the following sections are Database Model Diagrams.

4.1.4.1 Trace/Flow META Information

The "snmpstats.pl" script extracts meta information from trace files or flow files. The information we collect at the moment includes start and end timestamps of the trace/flow, the number of messages seen and the number of managers, agents or unknown endpoints discovered (this only makes sense when the script is processing whole trace files). We use the following database table to store this information:

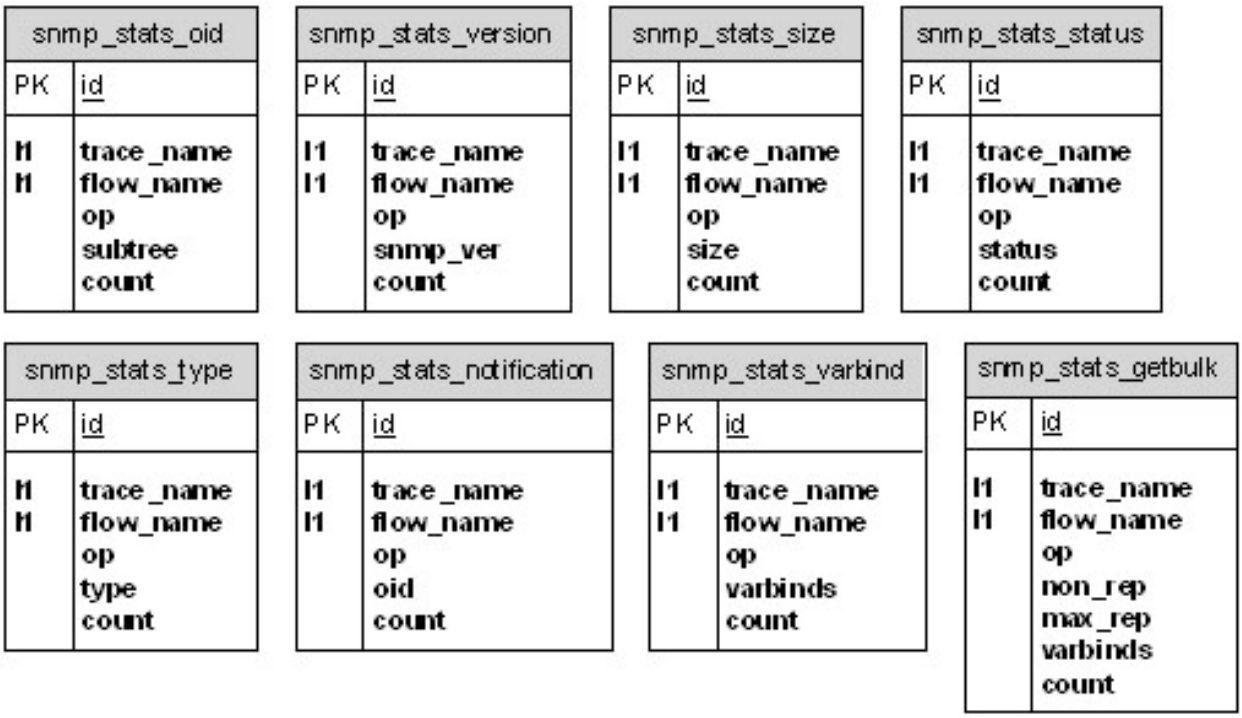
snmp_meta	
PK	id
l1 **l1**	**trace_name** **flow_name** **start_timestamp** **end_timestamp** **messages** **managers** **agents** **unknown**

An empty flow name in this table or any of the tables presented in the next section

indicates that the data represents the whole trace rather than a specific flow.

4.1.4.2 Trace/Flow Statistics

Additional trace/flow statistics are computed by the “snmpstats.pl” script and stored in the following tables of the database:

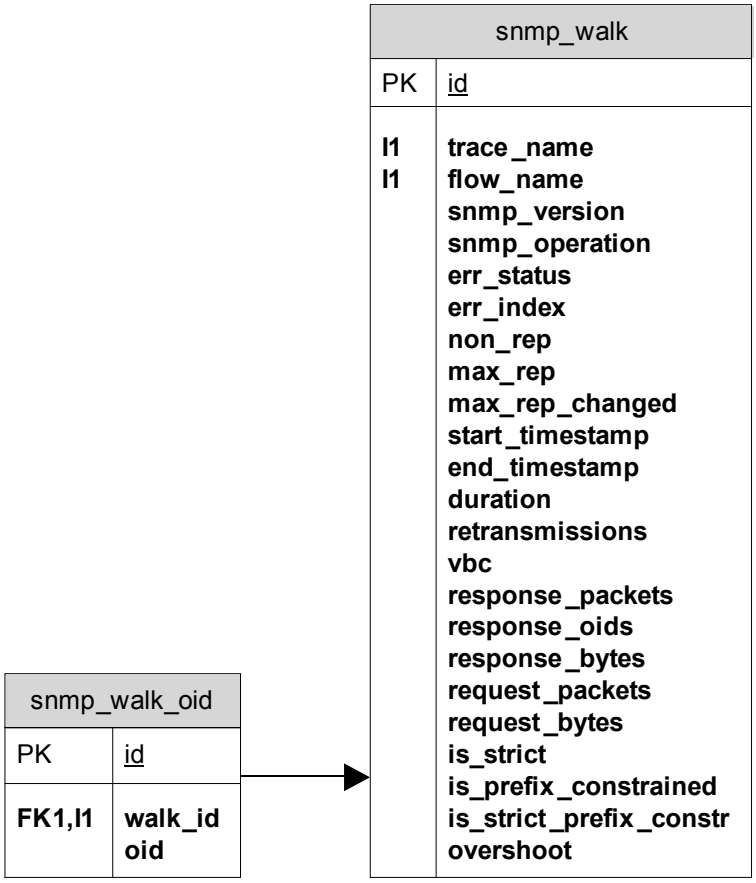


The “snmp_stats_oid” table shows a classification of OIDs based on their prefix, broken down by the SNMP operation. Table “snmp_stats_version” provides information on the SNMP operations, broken down by the SNMP version. The “snmp_stats_size” table stores the message size distribution for each SNMP operation. The distribution of status codes broken down by SNMP operation is stored in table “snmp_stats_status”.

The distribution of types used in SNMP operations is stored “snmp_stats_type”. The distribution of notifications types broken down by sub-tree is stored in “snmp_stats_notification”. The “snmp_stats_varbind” table shows the number of elements in the varbind list for each operation, while table “snmp_stats_getbulk” describes the parameters used in get-bulk operations.

4.1.4.3 WALK Information

The “snmpwalks.pl” script currently uses two database tables to store walk information computed from CSV trace files. Please see the model diagram below:



The "snmp_walk_oid" table is linked to the "snmp_walk" table through the column "walk_id", which represents the "id" column from "snmp_walk". It is a one to many relationship between the two tables (one walk has one or more starting OIDs).

Each discovered walk is represented by a record in the "snmp_walk" table. We use a second table, "snmp_walk_oid" to store the starting OIDs for every walk. The table below describes in more details the fields of "snmp_walk" database table:

Field Name	Description
Id	Auto increment and primary key for this table.
trace_name	The name of the trace to which this walk belongs to.
Flow_name	The name of the flow to which this walk belongs to.
snmp_version	The SNMP protocol version used.
snmp_operation	The SNMP protocol operation used (get-next-request or get-bulk-request).
err_status	The error status of the last response packet from this walk.
err_index	The error index of the last response packet from this walk.
Non_rep	The non-repeaters value from the last

	request packet of this walk.
max_rep	The max-repetitions value from the last request packet of this walk.
max_rep_changed	Indicates if the max-repetitions parameter was changed by the command generator during this walk.
start_timestamp	The timestamp of the first packet from this walk.
End_timestamp	The timestamp of the last packet from this walk.
Duration	end_timestamp – start_timestamp
Retransmissions	The number of retransmissions detected in this walk.
Vbc	The number of varbinds simultaneously retrieved in this walk, excluding any non-repeaters.
response_packets	Total number of response messages in this walk.
response_oids	The total number of objects contained in all response messages of this walk.
response_bytes	The total number of bytes in all response messages.
request_packets	The total number or request (get-next-request or get-bulk-request) messages in this walk.
request_bytes	The total number of bytes in all request messages.
is_strict	Indicates if this walk is strict (0/1).
is_prefix_constrained	Indicates if this walk is prefix constrained (0/1).
is_prefix_constrained_strict	Indicates if the OIDs of this walk never go out of the initial prefix (that is, the walk doesn't reach the end of the table).
hole_detection	If the manager detects holes in this walk this field will be "1", if holes are ignored the field will be "-1", otherwise "0".

4.1.4.4 Queries on the Database

This section describes a list of queries that can be executed on this database to extract information about the traces. All the queries presented can either be executed on data from all traces or just one single trace or even flow. All queries below will look at the

data for trace ‘l12t01’.

Determine the size of every operation within each flow of a trace:

SELECT flow_name, op, SUM(count * size) AS size from snmp_stats_size WHERE trace_name = 'l12t01' AND flow_name != '' GROUP BY flow_name, op;

Determine the level of usage for every SNMP version within each flow of a trace:

SELECT flow_name, snmp_ver, SUM(count) AS size from snmp_stats_version WHERE trace_name = 'l12t01' AND flow_name != '' GROUP BY flow_name, snmp_ver;

Determine the “busiest” flows in a trace (in terms or number of messages):

SELECT flow_name, messages AS size FROM snmp_meta WHERE trace_name = 'l12t01' AND flow_name != '' ORDER BY size DESC;

Determine the longest flows in a trace:

SELECT flow_name, ROUND((end_timestamp-start_timestamp)/3600, 1) AS size FROM snmp_meta WHERE trace_name = 'l12t01' AND flow_name != '' ORDER BY size DESC;

Determine the most queried MIB sub-trees on the whole trace:

SELECT subtree, SUM(count) AS size FROM snmp_stats_oid WHERE trace_name = 'l12t01' AND flow_name != '' GROUP BY subtree;

Determine the most queried MIB sub-trees in each flow of a trace:

"SELECT flow_name, subtree, SUM(count) AS size FROM snmp_stats_oid WHERE trace_name = 'l12t01' AND flow_name != '' GROUP BY flow_name, subtree;

Determine which SNMP versions are being used and how many walks use each one:

SELECT snmp_version, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY snmp_version;

Determine which SNMP operations are being used and how many walks use each one:

SELECT snmp_operation, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY snmp_operation;

Determine how common retransmissions are:

SELECT retransmissions, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY retransmissions;

Determine how many walks are strict, prefix constrained or have never reached the end of the table (in other words, they are strict prefix constrained):

SELECT is_strict, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY is_strict;

SELECT is_prefix_constrained, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY is_prefix_constrained;

SELECT is_strict_prefix_constr, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY is_strict_prefix_constr;

What is the distribution of response packets or total number or OIDs retrieved?

SELECT response_packets, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY response_packets;

SELECT response_oids, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY response_oids;

What is the distribution of walks among flows?

SELECT flow_name, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY flow_name;

Are there any managers that vary the number of max-repetitions (in get-bulk requests)?

SELECT max_rep_changed, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY max_rep_changed;

What is the ratio between the size of the requests and size of the responses? (This can further be evaluated on a per flow basis):

SELECT flow_name, SUM(request_bytes)/SUM(response_bytes) FROM snmp_walk GROUP BY flow_name;

What is the distribution of walks based on how many columns are retrieved at the same time?

SELECT vbc, count(*) FROM snmp_walk WHERE trace_name = 'l12t01' GROUP BY vbc;

What are the Top 10 starting positions among all walks?

SELECT t.prefix, count(*) AS c FROM (SELECT GROUP_CONCAT(t2.oid SEPARATOR ', ') AS prefix, count(*) FROM snmp_walk AS t1, snmp_walk_oid AS t2 WHERE t1.id = t2.walk_id GROUP BY t1.id) AS t GROUP BY t.prefix ORDER BY c DESC LIMIT 10;

4.2 NetFlow Trace Collection and Analysis

4.2.1 Trace Collection Infrastructure

The Leibniz Supercomputing Center (in German: Leibniz Rechenzentrum, LRZ) in Garching is the service provider for application and network services for almost all academic institutions in and around Munich, Germany, with more than 60.000 hosts and more than 100.000 users.

At LRZ, NetFlow traces are collected from 9 backbone routers on a 5 min basis which sums up to roughly 11-18 GB of NetFlow traces in compressed flow-tools format each day. The main reasons for the collection of these traces include performance, fault and security monitoring. All three of these monitoring activities need information about the traffic on LRZ backbone links, among them the 10 GBit/s uplink to the XWin, the backbone of the DFN (Deutsches Forschungsnetz - German Research Network).

Software from the flow-tools package is used to collect this data from the routers. Traces are stored at LRZ in compressed flow-tools format.

For special cases of external interest in an anonymized version of this data, an anonymization server performs anonymization of the NetFlow data. In an interest to make it harder for potential analysts of these traces to deduce data that can be associ-

ated with certain users in the network, IP addresses are replaced using the flow-xlate tool.

Anonymized traces are stored on a different server with about 700 GB of disk capacity. On this server, only the traces of the last 30 days are kept due to capacity constraints.

4.2.2 Trace Analysis

At the start of the EMANICS NoE, staff from the University of Twente (UT) had expressed interest to analyze NetFlow traces. This set the ball rolling to set up a point where EMANICS partners can get access to the NetFlow traces. Given the size and nature of the network operated by the LRZ, we assumed a certain interest from other partners as well.

UT's original interest had been in analyzing the frequency of large flows (``elephant'' flows) in network traffic of a large network service provider. Unfortunately, no analysis was actually performed in this direction. Instead, interest changed at UT. We have received a first informal request on analysing the data from the viewpoint of a distributed intrusion detection system.

Caused by the limited interest by other EMANICS partners so far, no additional effort was spent on coming up with agreement templates. For EMANICS partners, access to anonymized data can be given on an individual basis, the conditions will be negotiated on request based on requirements from both sides.

All in all, limited interest from other partners makes it necessary to reconsider the long-term availability of this service. So far, the work done to install and run this trace access by far exceeds the negligible external use of this data.

4.3 SIP Trace Collection and Analysis

4.3.1 Introduction

The SIP trace collection is one of the important means through which the research community can understand the behaviour of a VoIP network in real-time. The analysis of these trace patterns and behaviour helps in developing new methods for network management and security. There are different ways of studying the traces. The first method is the collection of traces from our VoIP test-bed. Though this method can be useful to some extend, however this is cannot be useful in understanding problems on real production networks. Therefore, we use another method in which we collect traces from existing production networks (this involves the ISP providers).

4.3.2 Trace Collection

The very important factor in trace collection is, from where it is collected (real production network, e.g., ISP provider). UPI has collected traces from RDS Pitesti (local ISP provider), the same place where the traces for SNMP were collected previously.

Traces were collected from a specific VLAN for about 83 days (15-03-07 to 07-06-07) using *tcpdump*. In addition, traces of the MGCP were collected and the analysis of them is work in progress.

4.3.3 SIP Analysis Tool

Since traces are captured using generic tools like *tcpdump*, they are stored in *pcap* files. Therefore, INRIA has developed a SIP trace analysis tool called "**SIP Tracer",** which can be used to analyze the SIP packets. The first step in the analysis will be to convert

the pcap files to a readable one. There are various tools that can perform this. In our tool we integrated the open source *pcapy*. Once we are able to read the captured packets, the next step will be to classify them. Since there are various messages available in the SIP packets, we need to classify them to understand the SIP transactions. To accomplish this, we have developed a SIP information model in our tool. Finally the last phase would be to analyze these traces according to our needs like (number of SIP packets, type of SIP packets, average length of the call, Standard deviation of the call, etc.) for various purposes. Presently we use these statistics for the enhancement of our VoIP bots. However the analysis tool can be used for various other studies.

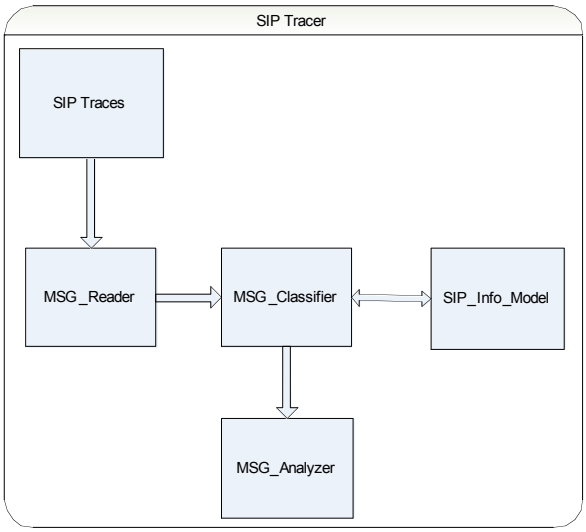


Figure 21: Block Diagram of SIP Tracer

As detailed above, we have various modules that perform the need for SIP trace analysis.

- **MSG Reader:** This module is used to read the trace files and collect the needed information for the classification of the SIP packets
- **MSG Classifier**: This module classifies the SIP packets (Provides the statistics information about the SIP packets.)
- **SIP Info Model**: This module provides the Information model for all the SIP messages and transactions.
- **MSG Analyzer**: It is the module that is used to perform various analysis like (Average Call duration, Standard deviation, etc.,)

4.3.4 Trace Analysis

In this section we present some initial results of the SIP trace analysis. However, to have more complete results, there are various additional analysis methods that need to be integrated into the tool.

As described in the previous section the first step in the analysis would be to analyze the different SIP packets. This is very useful in understating the reliability of the SIP network. For example, if we observe a large number of 4xx SIP packets, we could consider that the network is not configured correctly or we could find that there are intruders in the network trying to access some unauthorized services. We may be able to understand what is going on analysing other SIP packets.

SIP Statistics		
Packet Information		**Number of Packets**
Number of SIP Messages		7033
Number of resent SIP packets		3267
SIP Status codes in reply packets	407 Proxy Authentication Required	48
	484 Address Incomplete	3
	603 decline	2
	200 Ok	1012
	100 Trying	364
	401 Unauthorized	370
	180 Ringing	205
	403 Forbidden	5
	405 Method Not Allowed	118
	183 Session Progress	1
SIP Request Methods	INVITE	145
	BYE	17
	OPTIONS	4074
	ACK	16
	SUBSCRIBE	13
	REGISTER	32
	NOTIFY	811

Table 2: SIP Message Statistics

Apart from the SIP message, we analyzed which transports are being used. As we know that SIP can be used both over UDP and TCP, it is important to know which of the transports is used much for the SIP transaction.

Packet Classification according to transports	
Transport	**Percentage of packets**
UDP	36.46
TCP	63.54

Table 3: Transport Usage

The above results are just initial steps in the analysis of SIP traces. Presently, the tool is being enhanced with other modules to calculate for example the average call duration and its standard deviation. These results will be used in the enhancement of the VoIP bots that was developed at INRIA. The SIP Tracer tool will be released as open source in the near future.

4.4 GRID Infrastructure for Trace Analysis

Network traces are collected and stored by different trace providers of WP2 and shared with partners who are interested in trace analysis (trace analysts or users). Trace data from various sources is stored on servers of participating trace providers. In order to get access to those resources, trace analysts have to negotiate with different providers for

the access to trace data. Additionally, they also need to keep multiple accounts for data access on different remote servers. These processes introduce extra administrative overheads both for traces providers and trace analysts.

Furthermore trace providers mostly prefer to have control over traces they are sharing due to juristic and administrative reasons. Transferring traces to foreign hosts is undesirable for some trace providers. Although it is possible for trace analysts to ask an administrator of a trace provider side to execute their programs on the provider's host, an automated job submitting solution to simplify the process is still attractive.

In order to facilitate trace usage and sharing and to avoid above-mentioned problems, we suggest a grid-based infrastructure for traces analysis and trace sharing.

4.4.1 Motivation

Grid technology has been used successfully for distributed data storage and distributed computing in many research areas. It provides research communities with an environment to support collaborative activities. In this section, we show the major characteristics of grid infrastructures and how they help to address the problems that we have in the current scenario for sharing and analyzing traces.

In our trace storage and analysis scenario, data are distributed among trace providers, in order to get access to traces, trace analysts have to either download data or log into different remote servers. Grid infrastructure allows all trace activity's participants to form a so-called *Virtual Organization* (VO) in which participants share agreed resources with each other. Within this organisation, resources are shared not only by file exchanges but also by direct access to software and data. If trace analysts and trace providers belong to the same VO, analysts need merely log into their local grid node once and start their application based on remote data.

Based on the Globus Security Infrastructure (GSI), a user only needs to be authenticated once and he will then have access to all available grid resources. In this way, trace analysts who want to use multiple data from different providers don't have to log on to remote server multiple times for data access.

GridFTP is high-performance, reliable and secure file transfer protocol for grid environments. This protocol is based on the FTP protocol with extra considerations on performance, security, and stability. It supports PKI based security infrastructures to protect file transfers of grid applications. GridFTP allows third-party transfers, which means a client could initiate file transfers between two servers in addition to server-client transfers. Reliability of the GridFTP protocol is achieved through restart mechanism, which means a receiving server sends restart markers periodically to the sending party. If communication is broken due to certain failures, a client can, with help of restart markers, pick up the transfer where it left off and avoid complete new transfer of the file. With parallel transfers and large file support, GridFTP can transfer files efficiently.

Grid Security Infrastructure (GSI) is an implementation of existing and emerging standards. GSI bases its functionality on public key cryptography. The goals of GSI it to provide secure communication between grid participants in a distributed and inter-organisational environment. In the meanwhile, it avoids a centrally-managed security system. The core concept in grid GSI authentication is a digital certificate. Each participant is identified by a certificate, which contains vital information on this particular user. There are several ways to issue certificates, in order to simplify the process. In our test scenario, we use SimpleCA to set up certificates. SimpleCA provides a wrapper around

the OpenSSL CA functionality and it is sufficient for our test implementation. Following is an example of certificate information for a current user:

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 4 (0x4)
    Issuer: O=Grid, OU=GlobusTest, OU=simpleCA-project-emanics-lmu,
            CN=Globus Simple CA
    Validity:
      Not Before: May 8 15:17:53 2007 GMT
      Not After : May 7 15:17:53 2008 GMT
    Subject: O=Grid, OU=GlobusTest, OU=simpleCA-project-emanics-lmu,
                    OU=nm.ifi.lmu.de, CN=Liu Feng
```

Grid middleware offers possibility to monitor resource usages on nodes as well as job execution status (i.e. *ACTIVE*, *DONE*, *PENDING*, *FAILED*). Users can query information like CPU usage, total number of jobs and number of waiting jobs through a web-based interface. With corresponding job identification, users can query the status of their jobs through a command line interface at any time. Trace analysis is time- and resource-consuming. With simple monitoring mechanism provided by the grid middleware, trace analysts can keep track of their jobs and the resource usage on either remote or local nodes.

The job submission can be configured through an XML-based job description file, which indicates, for example, the location of the executable, possible parameters, or the location of data and results. Users can either manually submit jobs to various nodes according to their availability or they can rely on a meta scheduler to do the job distribution. In our usage scenario, trace analysts can send different jobs to different nodes regardless the locations of the data and application.

4.4.2 Grid-based Test-bed

Our test-bed is based on the Globus Toolkit (GT), which is the de facto standard middleware developed by the Global Grid Alliance for building grid systems and applications. It is open source software that is wildly used and supported by research communities. Grid middleware enables participants of VO to share data, computational power and other resources across institutional and geographic boundaries without losing their local autonomy. We compiled latest stable release of Globus Toolkit (4.0.4) from source and successfully installed it on our local server for testing purpose.

In order to run grid middleware, several applications are obligatory to support the operation:

- Apache Tomcat and Apache web server are required by web-based monitoring and discovery services (WebMDS).
- PostgresSQL or other JDBC compliant database is required to support GridFTP operation to store file transfer status.
- Java 2 Standard Edition SDK to support java-based web services.

Due to the fact that trace analysis activity is not our major undertaking, we currently installed Flow-tools for NetFlow traces analysis on our grid node for experimental purposes. Flow-tools is a collection of programs and libraries to collect, process, analyze as well as generate reports from NetFlow data. In current stage, user could logon to our grid node and submit jobs as grid jobs based on Flow-tools. If further interest exists, we could integrate their desired applications as grid services.

Since our current focus is testing and getting familiar with the Globus Toolkit, we started our experimental environment with a Linux based demonstration server dedicated to the EMANICS project. The current NetFlow trace repository exists on the same server. The current server has following configuration:

CPU	Intel Pentium 4 HT, 3.0GHz
Memory	1.024 MB
Hard Drive	700 GB
Operating System	SUSE Linux 10.0 with Kernel 2.6.16

4.4.3 Submitting Jobs to the Grid Test-bed

At current stage, a user who wants to submit test jobs has to first login to our demonstration server as local grid user. The grid user account has the right to access grid services as well as the trace analysis program `flow-tools`. A user has several options to submit his jobs to our grid test-bed. The standard way to submit a job is through the globus job submission client `globusrun-ws`. It is responsible for submitting and managing jobs on local or remote hosts. Here is a simple example:

```
globusrun-ws -submit -F emanics \
-streaming -so results.txt \
-c /var/flowtools/flow-print \
<~/testdata/ft-v05.2007-05-31
```

In the above example, a job to print out trace data `ft-v05.2007-05-31` with the command `flow-print` is submitted to the local server `emanics`. The result is saved in the file `results.txt`. Alternatively, a user could submit his job by using a job description file (JDF). A job description file is an XML file, which provides the grid job manager with necessary information about the job and the resources that are needed to carry out the operations. Following is an example of JDF (`simple-job.xml`):

```
<?xml version="1.0" encoding="UTF-8"?>
<job>
 <executable>/var/flowtools/flow-print</executable>
 <directory>/home/griduser</directory>
 <argument></argument>
 <stdin>~/testdata/ft-v05.2007-05-31</stdin>
 <stdout>results.txt</stdout>
 <stderr>error.log</stderr>
</job>
```

In the XML file above, the `<executable>` element tells the job manager where to find the program. The `<directory>` element specifies the current directory for the execution of the command on the node. Program-specific arguments are given by the `<argument>` element. The `<stdin>` element indicates the location of standard input. The `<stdout>` and `<stderr>` elements provide information where standard output and standard error should be redirected to. A job can be submitted with the following command:

```
globusrun-ws -submit -f simple-job.xml
```

If a user wants to operate on remote data without transferring them to the local node, slight changes are required to the JDF file (`remote-job.xml`) and its submission parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<job>
 <executable>/var/flowtools/flow-print</executable>
 <argument>&lt</argument>
 <stdin>/data/testdata/ft-v05.2007-05-31</stdin>
 <stdout>/tmp/results.txt</stdout>
 <stderr>/tmp/error.log</stderr>

 <fileStageOut>
  <transfer>
    <sourceUrl>
      /tmp/results.txt
    </sourceUrl>
    <destinationUrl>
      gsiftp://emanics:2811/home/griduser/results
    </destinationUrl>
  </transfer>
 </fileStageOut>
</job>
```

This example assumes that the required executable exists on the remote node and that the user knows both locations of the executable and the data. The `<fileStageOut>` element tells the job manager on the remote node that after job execution results should be sent with GridFTP to the job submitter. This job could be submitted with `globusrun-ws` with the extra parameter `-F` which indicates the address of the remote node. In addition to the `<fileStageOut>` element, there is also a `<fileStageIn>` element available, which indicates the location of data to be processed. Data staging is especially useful if a user wants to perform analysis operations regardless of the locations of programs as well as data.

Based on the web-based monitoring and discovery service (WebMDS), information on resources and their usage could be viewed online with a web browser. The following screenshot shows an example of a resource-monitoring page.

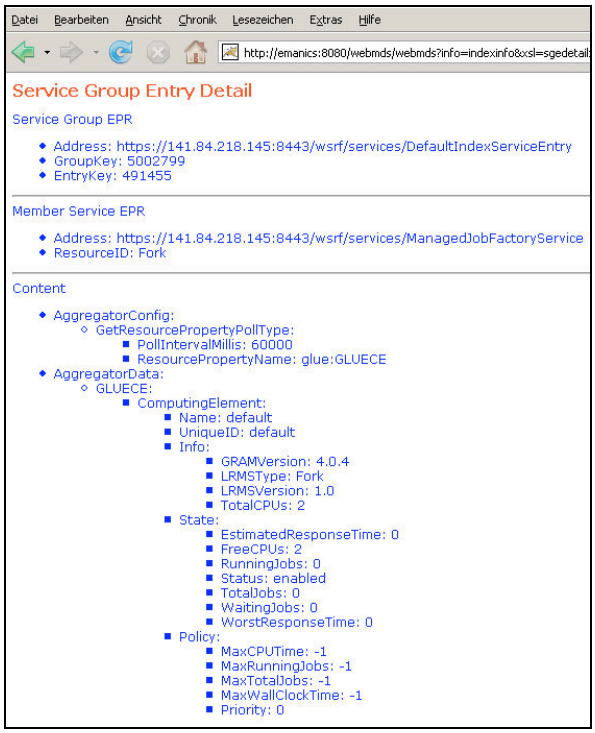


Figure 22: Service monitoring and discovering website

4.4.4 Collaboration and Future Plan

The idea behind grid technology is to facilitate collaboration of researchers. Therefore, the purpose of this project is to simplify and to encourage collaborations between trace providers and trace analysts involved in WP2 of the EMANICS project. Currently, we are collaborating with the University of Twente to test our grid infrastructure for trace analysis and we are trying to analyze their requirements on trace analysis applications. If further interests exist, we plan the following steps for future collaboration:

- Extending our current grid infrastructure by adding more nodes locally into the existing infrastructure. We currently consider using virtual machine as grid nodes for further testing purpose.
- Collaborating with interested partners to build further grid nodes and form a virtual organisation for traces analysis and traces sharing.
- Integrating trace analysis applications as grid services to meet the needs of our partners.
- Providing a web-based job submission interface to further simplify job the submission process.

5 Trace Replay (REPLAY)

The Trace Replay project is a joint project by the UT and KTH. The UT is responsible for making available its previously collected TCP-IP header traces and for collecting some new traces. KTH is using these traces for their A-GAP research project. This section of this deliverable provides information on the trace collection, the infrastructure needed to make these traces available, and on how others will be able to use these traces. The research performed by KTH is *not* discussed in this deliverable, since it is embedded within one of EMANICS research WPs.

5.1 Introduction

One of the interesting research projects within KTH is to develop a new adaptable protocol that can monitor large-scale networks in real-time. This protocol is called A-GAP. It can monitor network-wide metrics computed from device counters using aggregation functions, such as SUM, AVERAGE and MAX. Examples of such metrics include the total number of VoIP flows and the maximum link utilization in a network domain. A-GAP is a decentralized and asynchronous protocol that minimizes the generated overhead for a configurable accuracy of the estimation.

To evaluate the A-GAP protocol, real traffic traces collected from representative locations are needed. Such traces allow the validation of predicted results, which are based on the stochastic model that underlies A-GAP. The idea is to replay previously recorded IP packets in a test environment. Note that, for replaying, only TCP-IP header data is needed; packet payload is not needed. For privacy reasons the payload is therefore not included on the traces; for the same reason IP addresses need to be anonymized.

5.2 REPLAY Trace Collection

Within the REPLAY project the UT has made available its previously recorded anonymized traffic traces. These traces were collected at six locations:

- **Location 1:** On location #1 the 300 Mbit/s Ethernet link (a trunk of 3 x 100 Mbit/s) has been measured, which connects a residential network of a university to the core network of this university. On the residential network, about 2000 students are connected, each having a 100 Mbit/s Ethernet access link. The residential network itself consists of 100 and 300 Mbit/s links to the various switches, depending on the aggregation level. The measured link has an average load of about 60%. Measurements have taken place in July 2002.
- **Location 2**: On location #2, the 1 Gbit/s Ethernet link connecting a research institute to the Dutch academic and research network has been measured. There are about 200 researchers and support staff working at this institute. They all have a 100 Mbit/s access link, and the core network of the institute consists of 1 Gbit/s links. The measured link is only mildly loaded, usually around 1%. The measurements are from May - August 2003.
- **Location 3**: Location #3 is a large college. Their 1 Gbit/s link (i.e., the link that has been measured) to the Dutch academic and research network carries traffic for over 1000 students and staff concurrently, during busy hours. The access link speed on this network is, in general, 100 Mbit/s. The average load on the 1 Gbit/s

link usually is around 10-15%. These measurements have been done from September - December 2003.

- **Location 4**: On location #4, the 1 Gbit/s aggregated uplink of an ADSL access network has been monitored. A couple of hundred ADSL customers, mostly student dorms, are connected to this access network. Access link speeds vary from 256 kbit/s (down and up) to 8 Mbit/s (down) and 1 Mbit/s (up). The average load on the aggregated uplink is around 150 Mbit/s. These measurements are from February - July 2004.
- **Location 5:** Location 5 is a hosting-provider, i.e. a commercial party that offers floor- and rack-space to clients who want to connect, for example, their WWW-servers to the Internet. At this hosting-provider, these servers are connected at (in most cases) 100 Mbit/s to the core network of the provider. The bandwidth capacity level of this hosting-provider's uplink (that we have measured) is around 50 Mbit/s. These measurements are from December 2003 - February 2004.
- **Location 6:** On location #6, a 100 Mbit/s Ethernet link connecting an educational organization to the Internet has been measured. This is a relatively small organization with around 35 employees and a little over 100 students working and studying at this site (the headquarter location of this organization). All workstations at this location (~100 in total) have a 100Mbit/s LAN connection. The core network consists of a 1 Gbit/s connection. The recordings took place between the external optical fiber modem and the first firewall. The measured link was only mildly loaded during this period. These measurements are from May - June 2007.

In total we have captured around 850 TCP/IP packet traces. Each trace contains 15 minutes of TCP/IP header data; the sizes of these traces range from a few megabytes to a few gigabytes. In total some 500 gigabytes of uncompressed trace data was collected (the size of the compressed traces is around 55GB).

In addition to the existing traces, the UT has also collected new traces from two locations. Although collection of these new traces has not been fully completed yet, details on three of these traces are as follows:

- TCP only, duration: 1 week. The size of this trace is around 1.2 GB.
- TCP and UDP, duration 2 weeks: The size of this trace is 2GB.
- TCP and UDP, duration 8 days. The size of this trace is 1.5GB.

5.3 REPLAY Infrastructure

The funding received by the UT for the REPLAY project has been used for buying two systems: a file server and a compute server. In addition to these two systems, a number of other systems are also used for the purpose of trace collection and distribution. An overview of these systems is provided below:

- A Thecus N5200 High Performance NAS Server. This server is equipped with five 500 GByte disks, which are connected in RAID-5. The available disk-capacity for user data is therefore 2 TB. The server has multiple 1 Gbps interfaces.

Figure 23: Thecus file server

- An Intel SR1500 server system running a web-server to make the traces stored on the file server available via HTTP.
- An INTEL SR1500 server system for the collection and anonymization of traces. For anonymization, the public source *tcpdpriv* package is used (see: http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html). This system has two 400 GB hard disks.

Back-ups of the traces have been made on separate disks.

Figure 24: INTEL SR1500 Server

5.4 Downloading REPLAY Traces

The Thecus N5200 NAS Server supports SMB and NFS. Since the firewall at the entrance of the University of Twente (UT) blocks SMB traffic, direct SMB mounts to the NAS server are only possible from within the domain of the UT. The address to connect to is:

- smb://130.89.144.83/public (user: guest, password: guest).

In addition to SMB mounts, the traces can also be obtained via the website installed on the INTEL SR1500 server. Since HTTP traffic is able to pass the firewall of the UT, this will be the preferred way to obtain the traces for anyone outside the domain of the UT. The URL to obtain the traces is:

- http://traces.simpleweb.org/
- http://traffic-repository.ewi.utwente.nl/

Note that the traces are additionally available via: http://m2c-a.ewi.utwente.nl/

5.5 REPLAY Results

Although the results of the A-GAP project are reported within the EMANICS research WPS, it is interesting to note that the following papers by KTH have already been referencing our traffic repository.

- Gonzalez Prieto, R. Stadler: "A-GAP: An Adaptive Protocol for Continuous Network Monitoring with Accuracy Objectives", IEEE Transactions on Network and Service Management (TNSM), under review.
- F. Wuhib, M. Dam, R. Stadler, A. Clemm: “Robust Monitoring of Network-wide Aggregates through Gossiping,” 10th IFIP/IEEE International Symposium on Integrated Management (IM 2007), Munich, Germany, May 21-25, 2007.
- Gonzalez Prieto and R. Stadler "Adaptive Distributed Monitoring with Accuracy Objectives," ACM SIGCOMM workshop on Internet Network Management (INM 06), Pisa, Italy, September 11, 2006.
- Gonzalez Prieto, R. Stadler; “Distributed Real-time Monitoring with Accuracy Objectives,” IFIP Networking 2006, Coimbra, Portugal, May 15-19, 2006.

It is interesting to observe that also researchers outside EMANICS are already using these traffic traces (see for example Google Scholar).

6 Resource Usage Data Collection (ABLOMERS)

6.1 Introduction

Resource discovery and monitoring in large-scale distributed systems such as Grids [12], involves determining which computational resources are available to be assigned in order to execute a specific job, application, service, etc., without interfering to their owner's activities. It is very important that Resource Monitoring Systems [13] offer resources' availability information completely reliable so that subsequent resource management activities can keep resource load as balanced as possible throughout the network. Therefore, there is an inherent necessity for novel monitoring systems in charge of sensing computational resources (memory, processor, storage, etc.) as well as monitoring and tracking network-level end-to-end performance.

The challenge and complexity of monitoring in large-scale networks is quite different from others. Due to the fact that the amount of resources to monitor and discover have been drastically increased, the heterogeneity and diversity found in these systems is bigger and more extended. Monitoring systems for large-scale distributed systems have basically three main goals: Firstly, they should report resource performance activity when certain applications or services are running by means of graphical interfaces or threshold alarms. Secondly, they must support users in finding and keeping track of resources of interest. Finally, they should provide mechanisms to help the Resource Management System to allocate both computational and networking resources.

This chapter presents the motivation, overview, and evaluation of the ABLOMERS distributed monitoring system along with an account of experience gained through real deployment on a large-scale scenario such as the Grid5000 test-bed [14]. ABLOMERS aims to improve current distributed monitoring systems such as MonAlisa [15] or Ganglia [16] in terms of flexibility and heterogeneity. The experiments performed along this research show that ABLOMERS is a distributed monitoring system for large-scale networks with high levels of scalability and flexibility as well as a formidable heterogeneity to monitor a wide and diverse set of operating platforms. This chapter is organized as follows. The following section highlights the motivations to develop this research. The next section offers an overview of the monitoring system approach followed by the project status section. The remaining sections are related to the evaluation of the distributed monitoring system in terms of its performance, flexibility, heterogeneity and scalability.

6.2 Motivation

Large-scale Resource Monitoring and Discovery [17] is a novelty and challenging research area, which differs from traditional monitoring systems for distributed computing networks because computational and network resources dynamically join and leave, they are mainly heterogeneous and subject to frequent faults. Novel monitoring systems, for large-scale networks must be able to operate ubiquitously with respect to the resources and application components being monitored. They should facilitate their own scalability and they should reduce their performance overload in their hosting nodes. Large Scale Resource Monitoring is a completely different approach than traditional distributed systems monitoring. It is due to the fact that all monitoring activities should be made taking into account different administrative domains and network issues. These

varieties of distributed entities produce several problems due to heterogeneity in the way that similar resources are configured and administrated.

In general, monitoring data should be managed in a distributed fashion. Having a single, centralized repository for monitoring data causes two distinct performance problems. Firstly, a centralized repository for information or control represents a single point of failure for the entire system. Secondly, centralized data management can result in a performance bottleneck. Therefore, a centralized monitored information source simply can not handle the load generated by actively monitored resources at high scales. Basically, the challenges of monitoring large-scale networks are:

- No single point of observation
- No central point of monitoring information
- Diverse Hardware and Software Systems
- Different policies and decision making mechanisms
- Network monitoring is very important
- Larger monitoring data sets

6.3 Overview of the Proposed Monitoring System

The project ABLOMERS consists of a set of distributed resource-monitoring agents, which are constantly capturing end-to-end network and computational resource performance statistics (processor, memory, software, network and storage) in large-scale distributed networks. We have solved the scalability problem by the distribution of the monitoring system into a set of sub-monitoring instances, which are specific for each kind of computational resource to monitor. This approach reaches a high level of generality by means of the integration of the Simple Network Management Protocol (SNMP) and thus, it offers a wide ability to handle heterogeneous operating platforms. The proposed Agents for Balancing Load through Multi-Mode Resource Constrained Scheduling (ABLOMERS) solve the flexibility problem by the implementation of complex dynamic software structures, which are used to monitor systems ranging from simple personal computers to complex multiprocessor systems or clusters with multiple hard disk partitions. This approach is integrating an end-to-end network-level monitoring technology. The CISCO IOS® IP Service Level Agreements (CISCO IP SLAs) [18] allows for monitoring end-to-end network-level performance between switches, routers or from either remote IP network device.

ABLOMERS also introduces the concept of dynamic software structures, which are used to monitor from simple personal computers to complex multiprocessor systems or clusters with even multiple hard disk partitions. Therefore, it deploys a single software thread per type of resource to be monitored, independently of the amount of such resources. This is worthy to mention because, many monitoring systems fail when they try to handle new “hot-plug resources” that have been added to the system. Every resource is monitored by independent software threads that start again at certain lapse of time becoming an infinite cycle. The cycle-timing is defined by local or remote administrators through booting parameters at the beginning of its execution. ABLOMERS automatically re-configures their trapping times (calls to the SNMP agent) in an autonomous way. ABLOMERS increases or decreases the interval times between every trap based on the state of the monitored devices. This feature is important where the overload caused by any monitoring system is not affordable by the hosting nodes. Figure 25 reflects the dis-

tribution of our monitoring agents (triangular-shaped icons inside of any computer) in a real network.

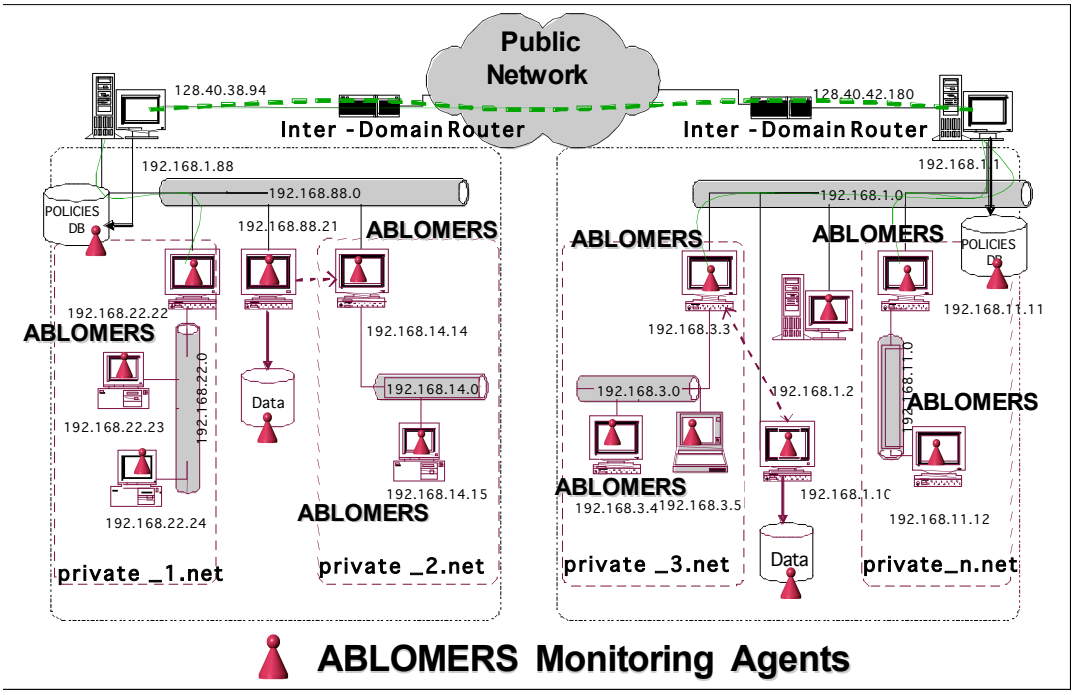


Figure 25. ABLOMERS Distribution in Virtual Organizations

6.4 Project Status

ABLOMERS has been completely implemented and is currently being tested. Unfortunately, it has not been possible to perform remote testing experiments with computational resources around many different EMANICS partners' networks because of the security policies in their laboratories or universities. Therefore, we decided to do testing in an experimental test-bed, namely the Grid5000 [14]. This is a French Grid initiative partially financed by INRIA, one of the EMANICS partners. This platform currently involves 3000 nodes located in 10 different sites in France. These nodes are linked through 1 and 10Gbits links (Figure 26).

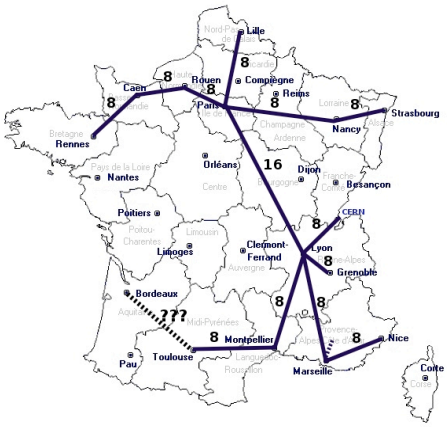


Figure 26:. Grid5000 Test-bed

We evaluated our approach on different Grid scenarios (micro Grid of nodes geographically located on one site, Enterprise small scale Grids with few dozens of nodes located

on a reduced number of sites and large scale Grids with 10 sites and few hundred of nodes). Also, different time frames were considered (from a few hours to a few days).

6.5 Overhead Evaluation

We evaluated system overheads installing ABLOMERS in a Pentium IV system with 512MB of RAM memory and the Windows XP operating system. The objective was to analyze the processor and memory consumption impact on the system performance. The following results are reported by means of Java Profiler. This programmer's tool can obtain a variety of information such as heavy memory allocation sites, CPU usage hot-spots, unnecessary object retention, and monitor contention, for a comprehensive performance analysis. Java Profiler helps software developers to find performance bottlenecks, pin down memory leaks and resolve threading issues. In our performance evaluation we will measure the CPU usage by ABLOMERS monitoring agents and how the system's memory is managed when the monitoring systems is running for long periods of time.

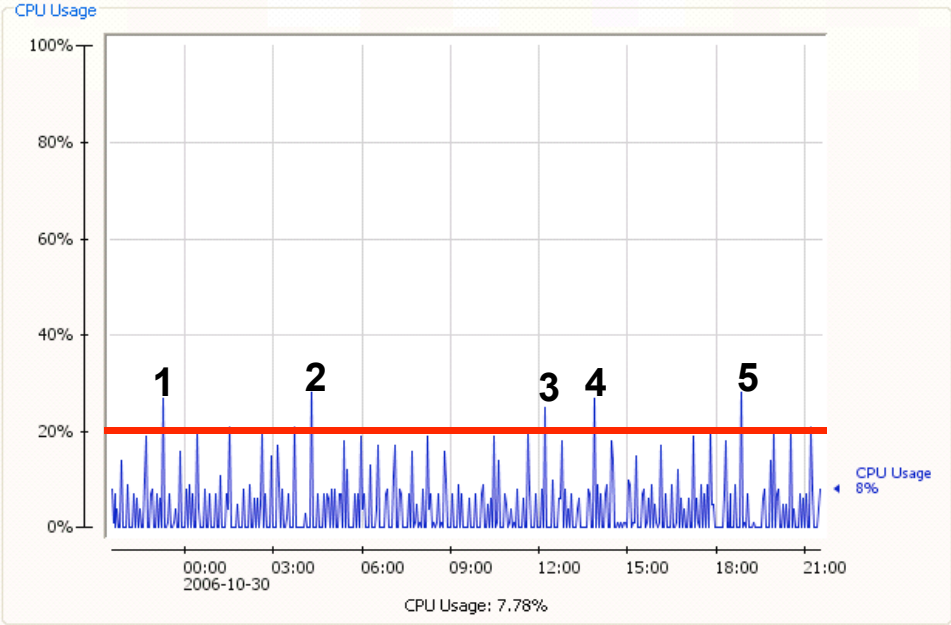


Figure 27: ABLOMERS CPU Usage

In Figure 27, we show the CPU usage of ABLOMERS monitoring agents. It is a snapshot obtained from Java Profiler. We have executed both, ABLOMERS monitoring agents and Java Profiler at the same time on the same workstation. It does not matter that other applications could be running on it because Java Profilers is just measuring resource usage by the ABLOMERS monitoring system. As can be observed, ABLOMERS causes an insignificant load impact on the system behavior. It is clear that only five times along a twenty-four hours experiment the ABLOMERS CPU usage is crossing the threshold of twenty percent of the total CPU capacity of the workstation. These changes in the normal ABLOMERS CPU usage are due to the fact that ABLOMERS generates resource availability reports in flexible intervals of time. ABLOMERS re-configures their trapping times (calls to the SNMP agent) automatically, based on the host node performance. Therefore, these changes are presented when the monitoring system performs these re-configurations. ABLOMERS CPU usage is normally oscillating between one and eight percent. In small intervals of time, system performance could be affected but in general it does not notice any overload.

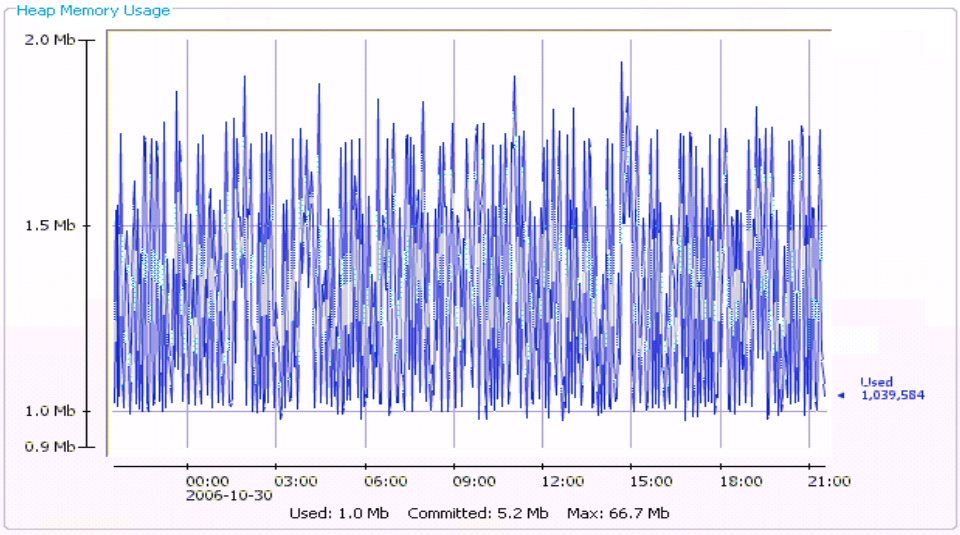


Figure 28: ABLOMERS Memory Usage

As far as the memory consumption by the ABLOMERS monitoring agents is concerned, Figure 28 reveals an increase from 1.0Mb to 2.0Mb occasionally. Figure 23 is also a snapshot obtained from the Java Profiler. As we have explained before, the fact that other applications may run on the same workstation where this experiment is being performed, do not affect the obtained results. Java Profiler assures that only ABLOMERS memory consumption is registered. The important information in this test is that memory consumption remains oscillating below this maximum for the whole test duration.

We have described that ABLOMERS is running a set of sub-monitoring agents, each per kind of resource and also per device to monitor. For instance, a total amount of five sub-monitoring agents are instantiated in a workstation with two hard disks, one of them contains two partitions; one bank of memory and only one CPU. Therefore, the amount of instances is growing directly proportional to the amount of devices to monitor. Then, it is important to show how ABLOMERS is controlling these instances, which basically are software threads. In Figure 29, the total amount of software threads are shown. It is clear that many threads are appearing and removing from the monitoring system. Therefore we conclude that ABLOMERS will never overload the host node capacity.

In summary, ABLOMERS monitoring agents do not affect system performance despite their continuous resource performance sensing. This statement is based on the fact that ABLOMERS is not consuming CPU and Memory resources that could affect the normal performance on its host node. It also controls efficiently the process of creating and removing instances (software threads) from the host node as we have shown in Figures 28 and 29.

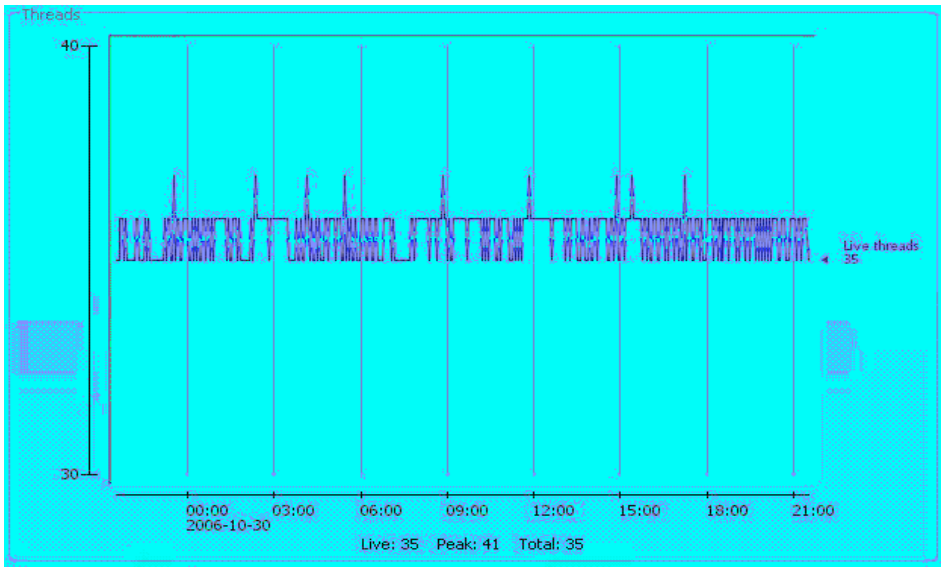


Figure 29: ABLOMERS Software Threads Management

6.6 Flexibility Evaluation

ABLOMERS reaches a high level of flexibility in two aspects: First, it implements dynamic software structures, which have been described during the overview section. In order to evaluate the reliability of these structures, we have deployed ABLOMERS in a storage server AthlonXP with three hard disks (Drive C, Drive D and Drive H). It is a common storage server with Linux Ubuntu as Operating System. We would like to show that ABLOMERS is able to start new sub-monitoring agents (as described in the overview section) when new resources are plugged into ABLOMERS' host node. In this experiment, we have connected an external storage device (i.e., a memory stick) for a period of half an hour; the device was disconnected and connected again one hour later.

In Figure 30, we show the total amount of devices available in this cluster and their performance. It shows four continues rows, which represent the percentage available (free space) in each of the hard disks available in this server. The memory stick was reconnected at 16:00 hours (brown line with small "x" items) and as the graphs show ABLOMERS is able to automatically identify this device and start the corresponding sub-monitor agent. This experiment could appear quite trivial, the importance is not just the ability of ABLOMERS monitoring agents to perform a new instance when the storage device appears and disappears. The crucial activity is that ABLOMERS keeps the statistical information of this resource in order to analyze if this device should be shared or not into the distributed network. We claim that ABLOMERS is novel in this activity since no resource monitoring so far offers this advantage.

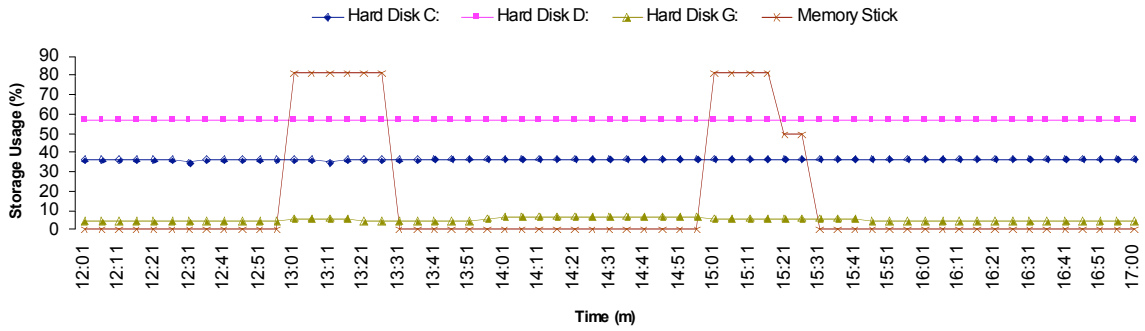


Figure 30: ABLOMERS New Sub-Monitoring Agents Deployment

The second aspect where ABLOMERS reaches flexibility is presented when it re-configures its trapping times automatically. In fact, ABLOMERS increases or decreases the interval times between every trap based on the state of the monitored devices. This feature of the ABLOMERS monitoring system was designed because current monitoring systems for distributed networks such as Ganglia have fixed trapping times, which causes that some information regarding resources behavior is just lost. It is caused because some events occur between two consecutive measurements and therefore cannot be detected. This problem is occurring when the monitoring system is not able to increase or decrease the gap between every call to SNMP-MIBs values. In ABLOMERS we have considered this issue. ABLOMERS is able to re-configure its interval times depending on the resource performance. It means that when a resource is being used quite frequently the time between every trap will be decreased and the amount of monitored information will be much more detailed. Similarily, when a resource is not being used for a long period of time, the time between every trap will be increased and then the amount of monitored information will be less. Moreover, the overload will be reduced significantly. The following experiments are supporting the before mentioned statements.

In Figure 31, we are plotting the CPU usage measured by the ABLOMERS distributed monitoring systems in one of the nodes from the GRID5000 test-bed (node-20.toulouse.grid5000.fr). This graph was done using fixed trapping times. It means that the interval time between calls to an SNMP agent was a fixed number during the experiment. In this graph the trapping time was twenty seconds. We have highlighted some intervals of time (i.e., between 13:40 and 14:40) where some information is just not detected by the monitoring system.

In Figure 32, we have performed the same experiment on the same Grid5000 node but we have activated the before explained self-configuration functionality by ABLOMERS. Here, it is quite clear that ABLOMERS is detecting resource usage that is not detected by a monitoring systems with fixed intervals between their trapping times. It may have important consequences when a monitoring system is not detecting some resource behavior information; because the results of the monitoring system are not successful.

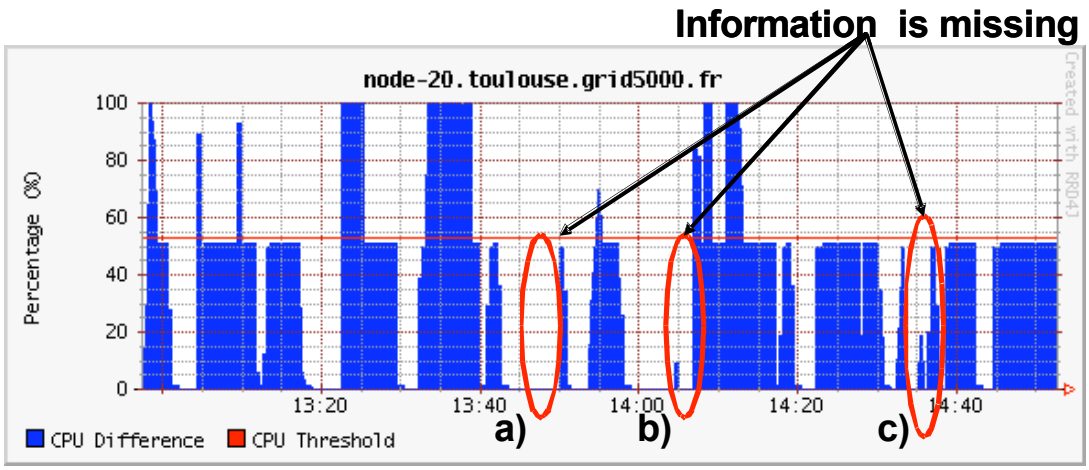


Figure 31: Fix Timing between Monitoring Traps

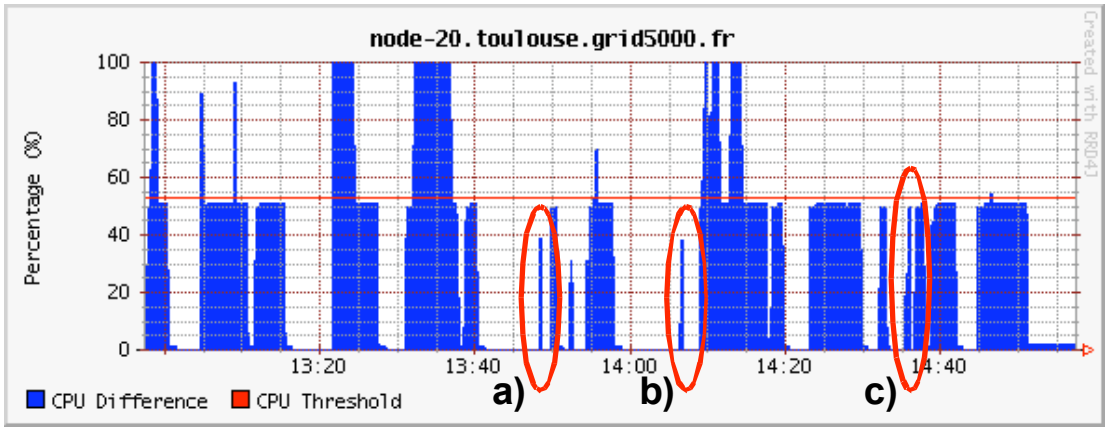


Figure 32: Auto-configuration Timing between Monitoring Traps

Obviously, the fact of decreasing interval times involves an impact on the performance of the hosting node. Therefore, we need to show that ABLOMERS monitoring systems can handle re-configuration changes without generating overload on its host node. The following experiment shows that ABLOMERS decreases its CPU usage when it increases its intervals times between every SNMP trap. ABLOMERS starts generating resource availability information every 10 seconds, then every five minutes it increase the trapping time in: 10, 20, 30, 40, 60 and 120 seconds. These values were chosen to show CPU usage in a slightly increment. Figure 33 shows ABLOMERS CPU usage versus time. Vertical red lines indicate the re-configuration moment for the trapping time. When ABLOMERS uses its shortest trap interval, the maximum CPU usage is less than 7% of the total capacity of its host node. This is supporting the statement that ABLOMERS is not an overload factor when it is monitoring resources, which normally present quite active behavior, such as CPUs or network interfaces.

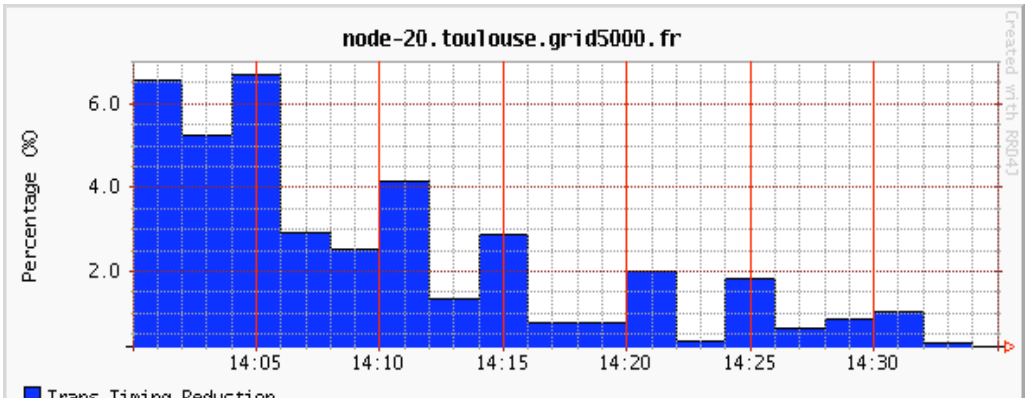


Figure 33: ABLOMERS Processor Overload in Auto-configuration Timing

6.7 Heterogeneity Evaluation

ABLOMERS claims to be a heterogeneous monitoring system. It is important to show this feature in real environments. Therefore, the following experiment was intended to collect resource availability information from three different operating systems: Windows XP, Linux Debian and Solaris 8. We pretend to show that ABLOMERS has been tested in different environments being a quite general solution for heterogeneous networks. In Figure 34, ABLOMERS is monitoring the CPU behavior in each node (every node is running under the before mentioned operating systems). These nodes were operating for a twenty-four hours period. ABLOMERS shows compatibility between these three different operating platforms. The heterogeneity in ABLOMERS is also demonstrated during the experiments on the Grid5000 test-bed. This statement is based on the fact that Grid5000 is a heterogeneous platform.

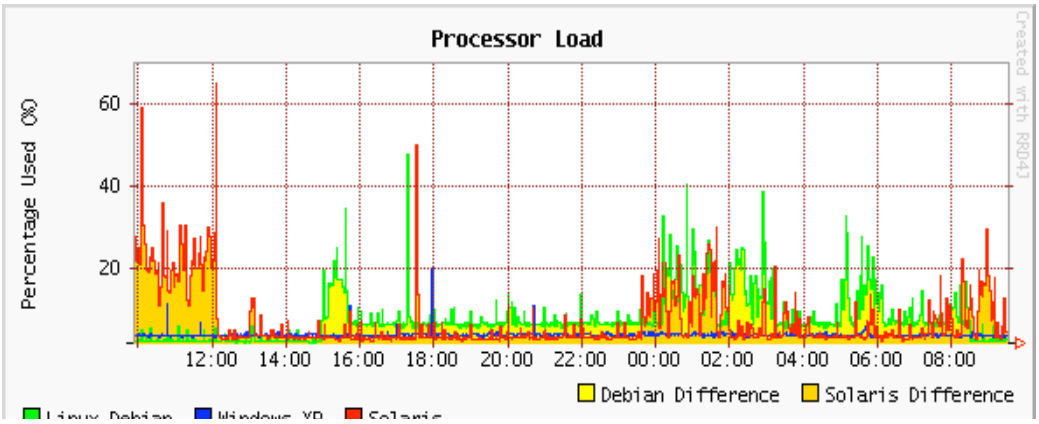


Figure 34: ABLOMERS Heterogeneity Snapshot

6.8 Scalability Evaluation

The scalability evaluation of ABLOMERS monitoring agents was performed in a real scenario, the Grid5000[1] test-bed. We have used 180 nodes belonging to this Grid with different architectures. A random number generator was used to dispatch processes to the network nodes in order to emulate normal “working day” conditions for all the nodes

[1]The information about Grid 5000 nodes architectures is found in the following web page: www.grid5000.fr

involved, so as to assure results approaching real Grid environments. ABLOMERS was previously installed in all nodes by means of remote scripts that copy, compile and execute the ABLOMERS code. The connectivity was done by means of “SSH” sessions. Figure 35 shows the performance activity of some nodes of the Grid5000.

Due to space limitations, it is not possible to show all graphs obtained in these experiments, we are just showing six of them. ABLOMERS has a web-based presentation of the monitoring information as we show in Figure 35. The graphs represent the resource activity along seven hours. Grid5000 is a high-demanded test-bed, where running long time experiments is quite hard to do. Anyway, the importance of this activity is to show that ABLOMERS is getting real-time percentage usability information about CPU, memory, storage and network activity at interface level (e.g. the amount of packets received and transmitted by the network cards of the nodes) from 180 nodes without any problem along the experiment.

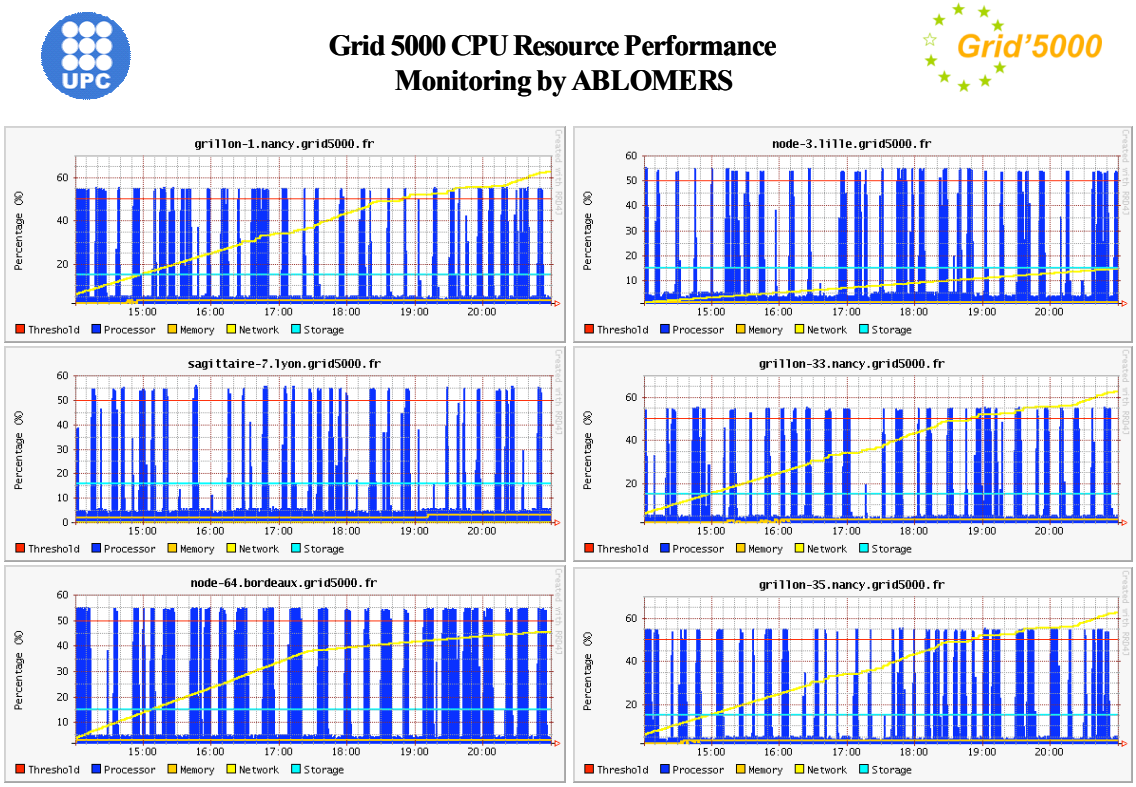


Figure 35: Grid5000 Resource Performance by ABLOMERS

The Grid5000 workflow is depicted in Figure 36. This sequence of activities is the same for every user of the test-bed. In our case, we followed this workflow to set our experiments up. The transmission of our experiments’ parameters is done by “SSH” connections. The execution of ABLOMERS and the collection of the information are done also by means of SSH connections. Unfortunately, there is not any alternative because of the security mechanisms implemented by the administrators of the Grid5000 test-bed.

Once we have our set of nodes ready to run our own experiments, we need to execute three activities to get information from the ABLOMERS distributed monitoring system. The first one is the configuration of the monitoring agents. In this phase, each node will receive the parameters used by ABLOMERS to configure its environment. These pa-

rameters are initial trapping times, activation of the flexibility mechanisms and number of traps needed to generate a statistical report.

The following phase is to send the activations command to every node where ABLOMERS has been configured. The last phase is to collect some resource behavior information from different nodes. These phases are actually part of ABLOMERS functionality. Therefore, we have tested how good the scalability is in each one of these phases. In Figure 37, we have measured the time required in the Grid5000 test-bed to configure all nodes running ABLOMERS. It is the first phase of the above mentioned. We have started with just five nodes and then we were incrementing the number of nodes in five until the amount of 115 nodes. We were not able to reserve more nodes in Grid5000. It is mainly because other researchers have reserved in advanced more nodes and we just were able to use these ones.

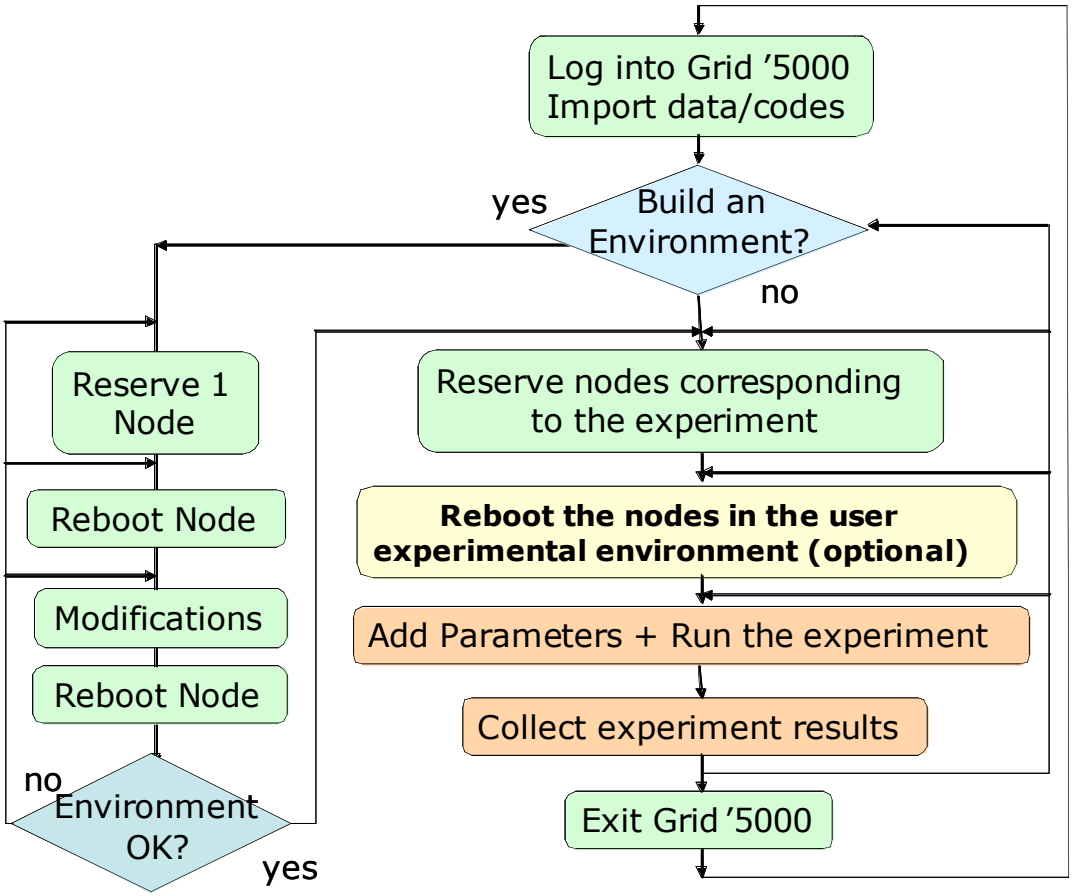


Figure 36: Grid5000 Workflow

The resulting graph shows that ABLOMERS is incrementing the time in a reasonable way. This steadily increment is due to the network traffic in the test-bed. We can not control the traffic between clusters which are forming the Grid5000. Fortunately, it is not affecting our results, as we show in the following graphs.

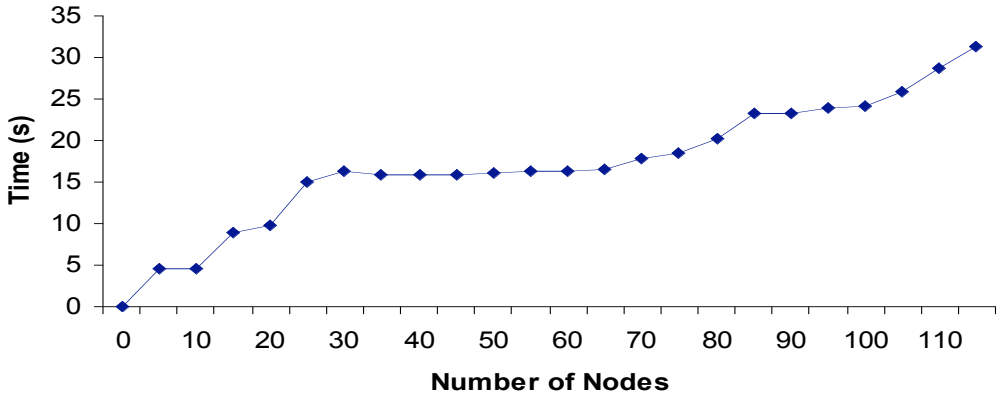


Figure 37: ABLOMERS Configuration Time in Grid5000

Regarding the activation phase, we have performed the same experiment. The following graph shows our results. In this phase, the stability of ABLOMERS is much more stable when the number of active nodes increases.

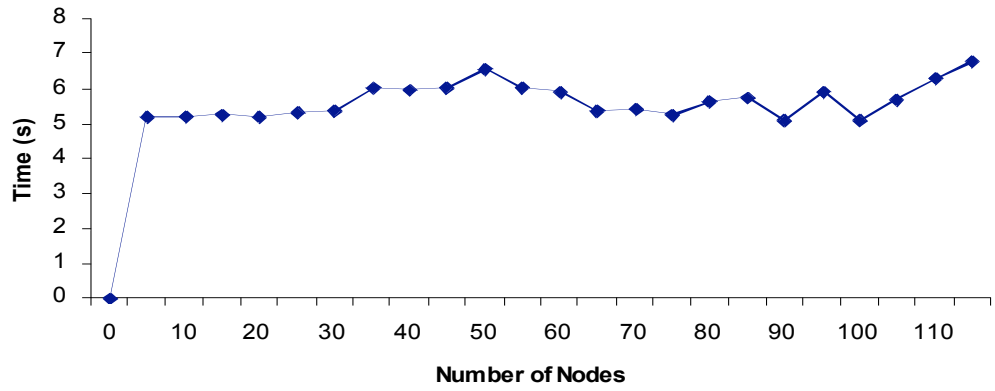


Figure 38: ABLOMERS Starting Time in Grid5000

Finally, we have also tested the scalability ABLOMERS when it is offering resource behavior information. This experiment has the same structure that the previous ones. In this case, the time that ABLOMERS consumes to offer specific resource behavior information is much less. Along this experiment, some values were longer that the expected because of network issues between ABLOMERS and the requesting entity. The requesting entity could be a user or administrator who wants to know resource behavior information in certain nodes (Figure 39). The time to get resource information is really short, around twenty and thirty milliseconds. This time remains steady regardless of the number of nodes in the experiment.

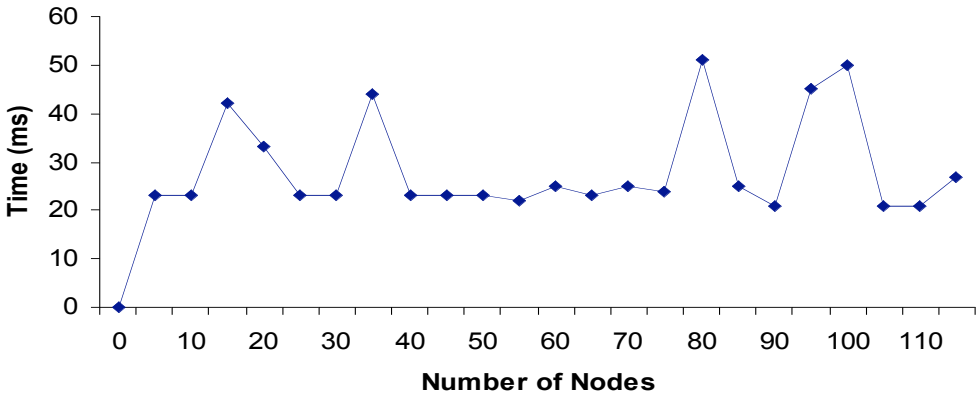


Figure 39: ABLOMERS Responding Time in Grid5000

6.9 Storage Evaluation

ABLOMERS is generating reports containing statistical resource availability information to be used by subsequent phases in the whole resource management process. These reports are done also in a flexible way, because they are directly proportional to the time between trapping times. When ABLOMERS is configured, it is necessary to specify the number of total traps required to generate a statistical report. When this number is small the number of reports will be bigger. For instance, in one hour there are a total of 360 traps (calls to the SNMP agent) whether statistical reports will be generated every 10 traps. The total number of statistical reports generated by ABLOMERS in one hour will be 36. On the contrary, when reports will be generated every 2 traps, the total number of statistical reports in one hour will be 180.

These reports are expressed in XML-based documents, which need certain space to be stored. Therefore, it is also important to know the total space used by these reports in a real scenario. In Table 4, we are presenting the kind of resource, time between each trap to MIB-OIDs values, the total amount of reports generated and the total space used by these reports in one of the nodes of the Grid 5000 test-bed. These results are reporting a time interval of twenty-four hours due to the fact that agents automatically clean memory buffers up after this period and they start to re-write from the oldest to the newest documents. Thereby, we avoid the possibility to fill system buffers and storage devices with monitoring reports.

Kind of Resources	Trapping Time (s)	Total Reports	Space Used (MB)
Processor	10	8640	3,52
Memory	60	1460	0,576
Network	30	2880	1,143
Storage	300	288	0,357
Software	1800	48	0,212

Table 4: Storage space used for resource monitoring DB

6.10 Conclusions

We have presented ABLOMERS, an open source monitoring approach for resource management purposes in large-scale networks. We have tested our distributed monitoring system approach in a real scenario, obtaining very promising results. The presented monitoring system would facilitate resource owners the provisioning of facilities for completing jobs execution in distributed systems. ABLOMERS presents an advantage in flexibility and scalability, due to the fact that it deploys several resource monitoring agents, which work independently from each other and offer real-time and statistical resource availability information.

We have performed several experiments to analyze ABLOMERS reliability in a large-scale scenario such as the Grid5000 test-bed. These experiments highlight the flexibility and scalability of our system. Moreover, ABLOMERS shows a great advantage in heterogeneity because it is able to monitoring heterogeneous operating platforms. We have done performance monitoring experiments on Grid5000. We have to mention that Grid5000 is not a homogeneous test-bed. Therefore, ABLOMERS shows a great capacity to perform the monitoring activities regardless of the architecture or complexity of the nodes forming the network.

The following phase in this project is to include end-to-end network-level monitoring information around geographically distributed networks. We aim to collect metrics such as average jitter, packet loss ratio, estimated bandwidth and average delay. We expect the collaboration from other EMANICS partners in order to realize this following research phase.

7 Collaboration

Partners in the VOIP project have attended a face-to-face meeting on February 15-16 in Munich. The goal of this meeting was to come up with proposals that can exploit efficiently the VoIP test-bed. The work carried out by UniZH, CETIM, and INRIA is based on a strong collaboration basis. In particular, UniZH has during the last six months integrated its VoIP infrastructure into the test-bed and tested conference call functions with other EMANICS partners. CETIM has improved the test-bed with the setup of a distributed dial plan system in collaboration with UniZH. INRIA has started to investigate performance and security issues in the test-bed. This work creates a better environment for research, education, and future collaboration.

One major discussion point in the TRACE activity during the last six months concerned confidentiality issues and the legal framework under which partners can share traces or contribute traces from their local sites. Several non-disclosure agreements have meanwhile been worked out between partners and working procedures have been established that seem to address the balance between the legal requirements and the desire of researchers to carry out their research projects. This achievement will likely pave the way to more collaboration in the TRACE activity. Partners from UT, INRIA, and IUB have met on April 27 in Bremen to discuss systematic approaches for SNMP trace analysis. While this meeting was primarily a WP7 meeting, it also served to discuss the progress on the SNMP trace collection infrastructure sponsored by WP2. The cooperation of INRIA and UPI is another example of fruitful collaboration, where UPI captures and provides SIP traces to INRIA, which develops SIP trace analysis tools to investigate these traces. In addition, LMU has installed a system to provide access to NETFLOW traces and a Grid-based platform for distributed trace analysis. UT has already indicated interests in the NETFLOW data sets.

The REPLAY and ABLOMERS projects are smaller and much more concerned with independent work. The collaboration in these projects is different from the two abovementioned projects. In the REPLAY project, KTH uses traces collected and maintained by UT to evaluate their A-GAP protocol. In ABLOMERS, UPC develops an open source tool for monitoring resources in large-scale distributed systems and requests partners to help in the evaluation of the approach. As documented in this deliverable, INRIA provided access to the French Grid5000 test-bed for the evaluation of ABLOMERS.

Without strong cooperation among EMANICS partners, the four projects sponsored by WP2 would not have made much progress and it would have been impossible to obtain the results documented in this deliverable. The collaboration between EMANICS partners achieved in this work package is high. Electronic communication mechanisms have helped coordinating the work on the various projects. However, the value of face-to-face meetings should not be underestimated for the stimulation of collaboration; independent of whether such meetings take place in specific WP meetings or more informally next to other events, such as IM 2007 or AIMS 2007.

8 Summary and Conclusions

The third “Virtual Laboratory Integration Report” documents the achievements of the four projects sponsored by WP2 during the last six-month period.

The VOIP project integrates EMANICS partners into the VoIP test-bed. UniZH has integrated and documented its Asterisk-based VoIP infrastructure and its configuration to connect to the local PSTN network as well as to the VoIP test-bed. They also documented the ENUM support and conference call setup to reach EMANICS partners. The experience of managing different hardware and software in a real platform can be useful for other partners. CETIM has evaluated the DUNDi approach for distributed dial plan storage in the VoIP test-bed. The idea is to use the DUNDi approach to configure automatically the dial plan and to provide routing information within the “emanics” context, which is restricted to EMANICS partners. CETIM has also started work on the integration of context awareness into the VoIP test-bed and provided a description of the implementation of common PBX functions with Asterisk. INRIA has developed VoIP-IRC bots to analyze and test VoIP test-beds.

The TRACE project is concerned with three different tasks: trace collection and analysis, analysis tool development, and distributed trace analysis. The TRACE project achieved the following milestones:

1. UT and IUB have defined a systematic approach and a common infrastructure for the collection and analysis of SNMP traces.
2. INRIA and UPI have collected SIP traces from the VoIP test-bed and from RDS Pitesti (a local ISP provider). INRIA has also developed a SIP trace analysis tool to scrutinize these traces.
3. LMU has collected NETFLOW traces from the Leibniz Supercomputing Center (LRZ) and installed a Grid-based server for distributed trace analysis. The Grid server is based on the Globus Toolkit, a commonly used Grid middleware.

The REPLAY project focuses on collecting TCP header traces to evaluate the A-GAP protocol developed by KTH for monitoring large-scale networks in real-time. UT made traces that have been collected at several locations available to KTH and additional traces are currently being collected. The results of testing the A-GAP protocol have been reported in other work packages.

The ABLOMERS project aims at investigating and evaluating the monitoring and load-balancing issues of large-scale distributed systems such as Grids. UPC has implemented a monitoring tool that discovers and monitors computational resources. The implementation has been tested on the French Grid5000 test-bed, made accessible by INRIA.

The collaboration of partners plays a vital role in the success of these projects. The VOIP project contains six partners, where UniZH, CETIM and INRIA actively proposed several extensions of the test-bed and attract the remaining partners to join experiments. The UniZH and CETIM groups’ work demands the collaboration of partners. Similarly, the TRACE project includes six partners. Due to the resolution of confidentiality issues, which took some time, partners have been reluctant at the beginning to share data. The cooperation of UT and IUB runs smoothly and constantly yields good results. INRIA has developed a SIP trace analysis tool to investigate SIP traces collected at UPI, another example for fruitful collaboration. LMU collects NETFLOW traces

at LRZ and proposes a Grid approach to distributed trace analysis. The approach is very promising for collaboration; however, the progress does not attract partners yet. The UT, leader in the REPLAY project, provides their trace collections for research activities in KTH. KTH has used UT's data sets already in several publications and also non-EMANICS partners have recently shown interest in these traces. The ABLOMERS project deals with the monitoring and load-balancing issues of large-scale distributed systems and is mainly driven by UPC while EMANICS partners provide mainly access to testing infrastructures.

WP2 has submitted three deliverables in the first phase of the EMANICS project. The first two deliverables focussed on documenting the available infrastructures, the definition of projects and goals, and the creation of collaborative environments; this deliverable more concentrates on the achievement of sponsored projects and the collaboration of partners that has emerged in the first phase of the project. All four projects sponsored by WP2 have achieved good results targeting the objectives of this work package; the degree of collaboration is strong and still increasing. In addition, some recognition of the sponsored projects outside of EMANICS has been reported lately.

The first deliverable of the second EMANICS phase is due in nine months. An open call for both new projects and continuation of existing projects has already been posted and discussions are underway between EMANICS partners to develop strong joint proposals. The selected projects will not only have to focus on the objectives of WP2, but they will also have to further strengthen collaboration within EMANICS. To this end, it is expected that more face-to-face meetings and exchanges will take place in the second phase to effectively support research and teaching activities through the common infrastructure developed in WP2.

9 References

[1] Asterisk, The open source PBX, http://www.asterisk.org/

[2] OpenSER, The open source SIP Server, http://www.openser.org/

[3] J. Rosenberg et al. SIP: Session Initiation Protocol, [RFC 3261], 2002, http://www.ietf.org/rfc/rfc3261.txt

[4] M. Spencer, Distributed Universal Number Discovery (DUNDi), [Internet-Draft], 2004, http://www.dundi.com/dundi.txt

[5] J. Schönwälder. SNMP Traffic Measurements. Internet Draft <draft-irtf-nmrg-snmp-measure-00.txt>, May 2006.

[6] R. Frye, D. Levi, S. Routhier, and B. Wijnen. Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Frameworks. RFC 3584, Aug. 2003.

[7] M. Havan and J. Schönwälder. Prefix- and Lexicographical-order-preserving IP Address Anonymization. In *10th IEEE/IFIP Network Operations and Management Symposium*, Apr. 2006.

[8] K. McCloghrie and F. Kastenholz. The Interfaces Group MIB. RFC 2863, June 2000.

[9] R. Raghunarayan. Management Information Base for the Transmission Control Protocol (TCP). RFC 4022, Mar. 2005.

[10] The EMANICS Web site. http://www.emanics.org

[11] The Simple-Web. http://www.simple-web.org

[12] I. Foster and C. Kesselman, “The GRID 2: Blueprint for a new Computing Infrastructure”. Morgan Kaufmann (Second Edition), May 2004. ISBN: 1-55860-933-4.

[13] J. Nabrzyski, J. M. Schopf and J. Weglarz, “Grid Resource Management State of the Art and Future Trends” Kluwer Academic Publishers. Boston, USA October 2004.

[14] F. Cappello, et al., “Grid'5000: A Large Scale, Reconfigurable, Controlable and Monitorable Grid Platform”, 6th IEEE/ACM Grid Computing, Grid'2005, Nov 13-14, 2005, Seattle, Washington, USA.

[15] I. Legrand, H. Newman, et al., “MonALISA: An Agent based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications” CHEP 2004, Interlaken, Switzerland, September 2004.

[16] M. Massie, B. Chun, and D. Culler. “The Ganglia Distributed Monitoring System: Design, Implementation and Experience” Parallel Computing, Vol. 30, Issue 7, July 2004.

[17] S. Zanikolas and R. Sakellariou, “A taxonomy of grid monitoring systems”. FGCS Journal. January 2005

[18] Cisco Systems, Inc. “The Cisco IOS IP Service Level Agreements – White Paper”. September, 2006.

[19] D. Harrington, R. Presuhn, B. Wijnen, “An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks”, RFC 3411, Dec. 2002.

[20] P. Barford, J. Kline, D. Plonka and A. Ron. A Signal Analysis of Network Traffic Anomalies, Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurements, Marseille, France, pp 71-82, 2002.

[21] J. Schönwälder, A. Pras, C. Ciocov, M. Harvan. “Walk or Crawl”. In progress.

[22] M. Rose, K. McCloghrie, and J. Davin, “Bulk Table Retrieval with the SNMP,” Performance Systems International, Hughes LAN Systems, MIT, RFC 1187, Oct. 1990.

[23] J. Schönwälder. SNMP Trace Analysis Update (slides). 22 NMRG at 68 IETF, Prague, March 2007.

[24] Brown, Peter J.: Triggering Information by Context. Personal Technologies, 2(1):1–9, 1998.

[25] Brown, Peter J., John D. Bovey and Xian Chen: Context-aware applications: from the laboratory to the marketplace. IEEE Personal Communications, 4(5):58–64, October 1997.

[26] Dey, Anind K.: Context-Aware Computing: The CyberDesk Project. Technical Report SS-98-02, AAAI 1998 Spring Symposium on Intelligent Environments, 1998.

[27] Dey, Anind K. and Gregory D. Abowd: Towards a Better Understanding of Context and Context-Awareness. In Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), The Hague, The Netherlands, April 2000.

[28] Pascoe, Jason, Nick Ryan and David Morse: Human-Computer-Giraffe Interaction - HCI in the Field. In Workshop on Human Computer Interaction with Mobile Devices, 1998.

[29] Schilit, Bill N. and Marvin M. Theimer: Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5):22–32, 1994.

[30] Schilit, Bill N., Norman I. Adams and Roy Want.: Context-Aware Computing Applications. In Proceedings of the Workshop on Mobile Computing Systems and Applications, pages 85–90, Santa Cruz, CA,USA, December 1994. IEEE Computer Society.

[31] Want, Roy, Andy Hopper, Veronica Falcao and Jon Gibbons: The Active Badge Location System. Technical Report, Olivetti Research Labs, January 1992.

[32] Winograd, Terry: Architectures for Context. Human-Computer-Interaction, 16(2), 2001.

[33] Globus Toolkit, The Globus Alliance. http://www.globus.org/

[34] Flow-Tools. http://www.splintered.net/sw/flow-tools/

[35] P. Faltstrom, M. Mealling, The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM), RFC 3761, April 2005

[36] International Telecommunication Union, The international public telecommunication numbering plan, E.164, Feb 2005

[37] DUNDi General Peering Agreement (GPA). http://www.dundi.com/gpa.pdf

10 Abbreviations

CGI	Common Gateway Interface
DoS	Denial of Service
DUNDi	Distributed Universal Number Discovery
ENUM	Telephone Number Mapping
IAX	Inter-Asterisk eXchange
IRC	Internet Relay Chat
ISDN	Integrated Services Digital Network
LDAP	Lightweight Directory Access Protocol
MIB	Management Information Base
NoE	Network of Excellence
NRPE	Nagios Remote Plugin Executor
NSCA	Nagios Service Check Acceptor
OID	Object Identification
PBX	Private Branch Exchange
PDU	Protocol Data Unit
PSTN	Public Switched Telephone Network
RADIUS	Remote Authentication Dial-In User Service
SIP	Session Initiation Protocol
SME	Small and Medium-sized Enterprise
SMS	Short Message System
SNMP	Simple Network Management Protocol
SSH	Secure Shell
VoIP	Voice over Internet Protocol

11 Acknowledgement

This deliverable was made possible due to the large and open help of the WP2 team of the EMANICS NoE. Many thanks to all of them.