Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Departement D-ITET

Computer Engineering and Networks Laboratory (TIK)

Semester Thesis

Call Route Discovery with Asterisk / DUNDi

André Wangler
awangler@ee.ethz.ch

September 5, 2007

Supervisor: Ulrich Fiedler

Prof. Dr. Bernhard Plattner

Institut für
Technische Informatik und
Kommunikationsnetze

Abstract

Starting point of the present work is a telephony system which is set up in a wireless mesh network and consists of interconnected PBXs (Public Branch eXchange). The PBXs provide the basic telephony services to the physical phones which are assigned to a specific PBX each and not allowed to move to another one. Moreover we have users which are mobile and can affiliate to arbitrary phones. After an affiliation to a phone the user is reachable at this phone under his personal number. To provide the mapping of the users logical number to a physical device, number and call route discovery is necessary. Already realized approaches using flooding does not meet the defined requirements with respect to traffic generation and time consumption. So a new event driven approach is introduced which is based on peer-to-peer trusted networks between the PBXs. For the implementation the open source PBX Asterisk as well as the number discovery protocol DUNDi are used and extended in a way that the dynamic requirements of our system can be fulfilled. The brought up procedure was tested on an emulated telephony network and it pointed out to be an efficient and well working solution for a future telephony system.

Contents

List of Figures

1 Introduction

In today's world communication is omnipresent. Mainly telephony is available everywhere. But one can think of situations where the present communication networks fail and backup systems have to be provided quickly. One of the problems in a dynamically built telephony system is the call routing within the changing topology of the telephony network.

The starting point of the current work is the scenario of a telephony system which is set up in a wireless mesh network that is characterized by its flat hierarchy whereby every node is similar and acts as host and as router.

In the assumed application there are three components which play an important role (see also Figure 1):

- First there are the nodes of the mesh network which act as PBX (private branch exchange). They are interconnected via a wireless interface and provide the necessary telephony functions such as call establishment or call forwarding to their subscribers. This PBX nodes are mobile which means that the topology of the network can change completely over time. The core of each PBX is its dialplan wherein every supported phone is listed and can be looked up by the local or a remote PBX.
- The second part of our system are the physical phones which are directly linked with the PBXs. Every phone is assigned to a specific PBX and it is not allowed to move to another one. The communication between the phones and the PBX can be managed through different VoIP (Voice over IP) protocols such as SIP (Session Initialization Protocol) or H.323.
- The last part is represented by the users. They are logical entities with an assigned number and they are allowed to move within the network. A user gets connected to the telephony system by affiliating itself or its number respectively to a physical phone. After an affiliation the telephony system is able to retrieve which logical number is assigned to a physical device and it will route future calls to the users number to this phone.

Let us have a look at an example of a possible scenario. One imaginable operational area is in the field of disaster recovery whereby no public communication channels are available. So one can erect antennas which are connected to boxes containing a PBX each. These are linked to multiple phones. The system allows a user to affiliate its logical number through the dial of the phone it wants to affiliate to. Then the user is reachable at this place up to the disaffiliation or up to the affiliation to another phone. Referring to Figure 1 this looks at follows: user A arrives at phone 11 which is connected to PBX node 1. User A affiliates itself to phone 11 which causes that it is reachable by the other users (e.g. B or C). Later user A moves to phone 23 (with or

without disaffiliation from phone 11) and affiliates to the new phone. The system now routes all calls to phone 23.

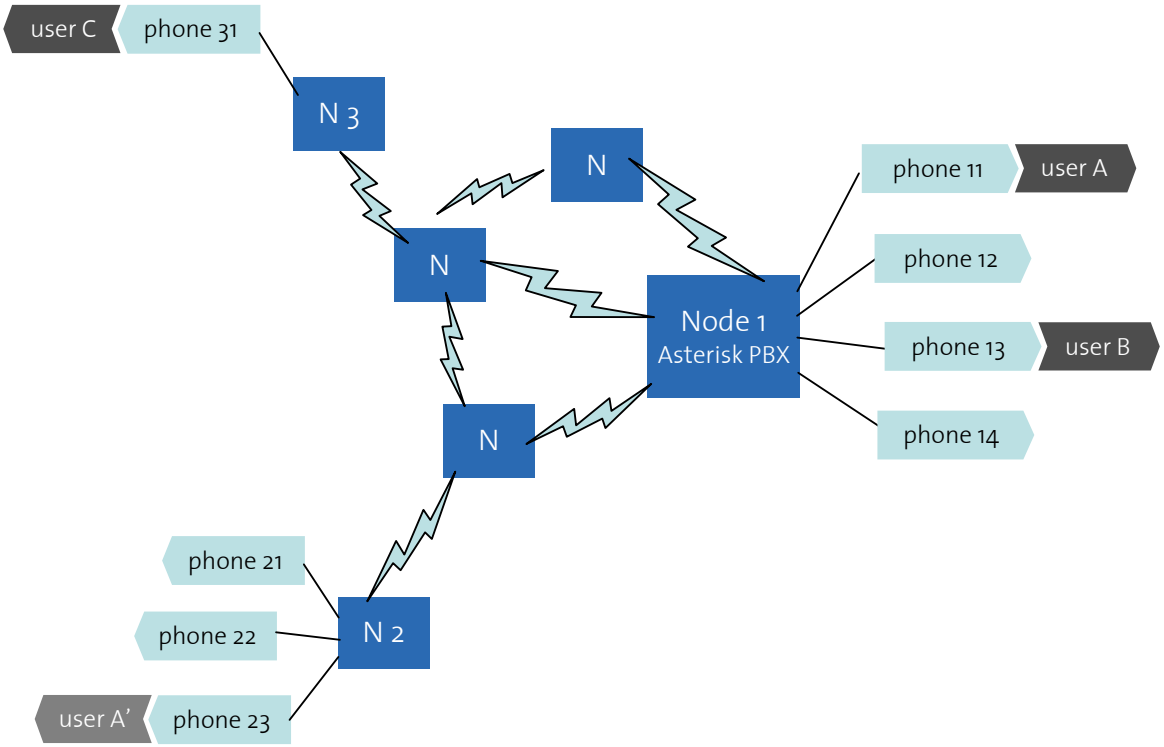

Figure 1: *The network we deal with consists of three parts: First the main nodes which acts as PBX and are organized as a wireless mesh network. The second part are the physical phones which are connected to a PBX and are not allowed to move. The last part are the users which affiliate to a specific physical phone. Users are mobile so they can affiliate to different phones at different time.*

To provide such a system one have to implement two network-side functions. The first one is number discovery, which means that the system maps a called logical number to a physical address or phone respectively. The second function is call route discovery, which performs the routing of a call to a physical phone through the network. These basically different tasks will be solved in the current work in a single step.

One can think of several algorithms to solve the call route discovery problem. One known approach is based on flooding. This one is already implemented by the industry partner (AS-COM Schweiz AG) and works as follows: The whole network is flooded to find a user. When the searched phone with the specific number is found a message is sent back and the system remembers the path to route the call. This is a low level approach and works without an IP like addressing of the different PBX nodes. The disadvantage of this attempt is on one hand that

flooding produces a large traffic overhead in the network. On the other hand this is a connection oriented approach which is not sufficient when one wants to implement for example data services in the same network.

To solve these problems and to improve the performance of the system, the goal of the current work is to bring up an event driven approach providing call route discovery in an IP based (packet switched) network wherein routing is done with OSPF. Since stability is one of the main requirements a completely distributed number lookup attempt is chosen which works on the basis of trusted partners which are organized in a peer-to-peer network. This allows us to do number discovery in an efficient way.

Furthermore several user-side functions should be implemented. So a user should be able to start the affiliation procedure by dialing a special code followed by his logical number. This procedure should be executed in a convenient time and afterwards one should be able to find the user from every phone in the network. Further a disaffiliation procedure should be implemented after that no one is able to reach the user any more. It should also be possible that a user is mobile without disaffiliating itself. This means at the affiliation the system should check whether the user is already affiliated at another place. Basically we can state that affiliation and disaffiliation have the goal to modify the PBX' dialplan depending on the users behavior. In the further paragraphs we will see where the problems of this simple attempt lie and how these will be solved.

For the implementation of the system we use the existing open source PBX Asterisk (see [1] and [2]) and its built in number discovery protocol DUNDi (Distributed Universal Number Discovery, see [4]). DUNDi as well as Asterisk are open source projects and are freely available. Furthermore the DUNDi protocol actually is independent of Asterisk but Asterisk is the only PBX which implements DUNDi up to now.

DUNDi performs number discovery exactly in the way we require. Namely it supports a completely distributed peer-to-peer event-driven approach to discover the location of a phone, based on trusted partners. It sends requests to remote Asterisk instances which look up the number in their dialplan. So DUNDi will deliver the address of the PBX to which the phone we are looking for is connected to. This address yields the implicit path to the target phone, so we have a convenient approach to solve the problem of call route discovery.

As already mentioned we use Asterisk to set up a PBX. Asterisk provides many telephony features by default. It allows us to set up a communication between different partners inside and outside the area of the local PBX, provides interfaces to telephony soft- and hardware and implements multiple security and encryption features. It also allows us a wide range of personal configuration possibilities: on the one hand in the programs source code and on the other hand

in the configuration files of Asterisk. From these configuration files at every reload of Asterisk the dialplan containing all subscribers and forwarding rules is generated anew. According to Asterisk all these subscribers are seen as physical phones with an assigned number, so it does not support our concept of phone independent users by default.

Since we want to dynamically adjust the dialplan and since every reload can affect ongoing calls and global variables, changing and reloading the dialplan can not fulfill our requirements. So the objective of the present work is to bring up a procedure, how a dynamic affiliation and disaffiliation (creating and deleting dialplan entries) as well as call establishment based on Asterisk could be done. Moreover we have to figure out how we can emulate mobile phone-independent users based on Asterisks device-oriented view of the system. Further we will have to investigate how DUNDi should be configured to gain the required performance.

In this work we will show an extension which uses a possibility Asterisk offers to SIP phones: they can be added dynamically to the dialplan by sending a SIP REGISTER message. But since we work with other protocols than just SIP, this alone will not help us. The fact that a SIP entity can work as redirection server will lead us to the solution that we are starting a virtual SIP phone for every affiliating user. Afterwards the virtual entity is responsible for forwarding every incoming call to the assigned physical phone.

Besides the implementation, a further main objective of the work is to test the gained procedure. Naturally it is too complex to build a real system. So we test our extensions in an emulated system built by UML instances (User Mode Linux).

UML is running on Linux PCs and simulates a whole Linux environment. Every instance is seen as a process by the host system so it is possible to start multiple UMLs. Furthermore one is able to setup network interfaces on a UML therewith we are communicating over channels simulated by link daemons. So we are able to emulate and test a whole telephony system of 17 PBXs, several phones and users on one PC.

To simulate the phones on the UML instances we use the call generator SIPp. It is able to send user defined sequences of SIP messages to other phones or PBXs. It is also possible to realize redirection servers with such SIPp scenarios. During the test the users which initiate affiliations, disaffiliations and test calls will be emulated by shell scripts which start SIPp instances which are responsible for this procedures.

The goal of these tests is to show the feasibility or the limitations of the chosen approach. It should be investigated whether the requirements could be fulfilled by utilizing Asterisk and DUNDi. Adjustments which should be done to Asterisk and DUNDi to work in a mature system should be found and reported. Furthermore the implemented procedures should be tested on

their functionality and their stability against network and link failures. In the end we want to have also a complete test environment that we can reuse for further tests.

The results will show us on one hand that we could have fulfilled the requirements of the aimed telephony network. But on the other hand we will also see some inaccurate implementations within the used open source projects which have led to unexpected results during the tests.

The rest of this thesis is structured as follows. Section 2 will explain the procedures of affiliation, disaffiliation and call setup in more detail. Furthermore, the adjustments which have been made to Asterisk and DUNDi are stated and the test setup is introduced. In Section 3 we will show the results and we will finish this thesis with the Conclusion in Section 4.

2 Implementation

2.1 Basic idea

As we have already seen that Asterisk is designed for static telephony setups whereby every phone and its owner is known at the initialization time of the system. To fulfill the requirement of a dynamic affiliation we profit from the way Asterisk handles SIP phones. The SIP protocol as well as Asterisk allow us to register SIP phones dynamically in the dialplan by sending a SIP REGISTER message to Asterisk (as stated in [3]). After the registration the number will be added to a specified context.

Since there are phones with other protocols than SIP connected to our Asterisk instance, we cannot directly make use of the dynamic SIP registration. So we implement a workaround by starting a virtual SIP phone for every affiliating entity. The initialized SIP phone would first register the logical number in the dialplan and will afterwards work as a redirection server for incoming calls. So we have an entry in the dialplan which forwards the number to the virtual SIP phone then the virtual destination knows the associated physical number and redirects the call to the physical device.

For our intension to simulate and verify this concept we take SIPp described in [9], which is a testing tool for the SIP protocol as well as for PBXs like Asterisk. There is the possibility to write own XML configuration scripts, which allows us to give this virtual phone the required behavior. Since SIPp can only be started either in server- or client mode we have to start SIPp multiple times for each affiliation, whereby the last instance takes the role of the redirection server and is running until the disaffiliation of the user.

The following sections show the basic ideas for the affiliation of the number 'abcd' to the physical device 1234. Because of simplicity, the physical device is represented as one instance, other than in the test scripts where we also have multiple SIPp instances taking the role of this phone.

For a final implementation of such a system, it would be a more proper solution to integrate the virtual SIP phone directly into Asterisk via the programming interface functions or by extending the source code of Asterisk itself.

2.1.1 Affiliation

Before we are able to start any affiliation during the tests, it is necessary to have a physical device to which the affiliation is done. Therefore we first register a SIP phone - realized as SIPp instance - as showed in Figure 2. This SIP phone stands for a physical phone communicating with an arbitrary protocol. For the test we use a SIP phone thus we can introduce a physical

phone dynamically, as well as the future virtual phones. In the test sequences we have to perform a physical phone which answers a call or a lookup respectively. So this 'physical' phone has the job to wait for calls, answer and finish them afterwards.

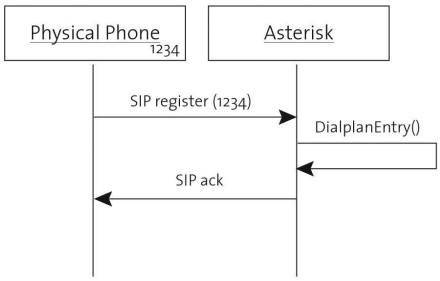

Figure 2: *Before an Affiliation can be established, we register a SIP phone which stands for an arbitrary physical phone. This SIP phone afterward has to initialize the affiliation procedure by calling a specified number.*

The user of the above registered SIP phone starts the affiliation process by dialing 871abcd from the physical phone 1234 (see Figure 3). Asterisk decodes the 871 prefix as an affiliation command and abcd as the users number. It starts a SIPp client, passing the number abcd. The SIP client then performs a SIP register procedure with the number abcd to Asterisk. On reception of the register, Asterisk adds a dynamically created entry for the number abcd to the dialplan. The SIP phone in client mode then terminates since it is no longer required.

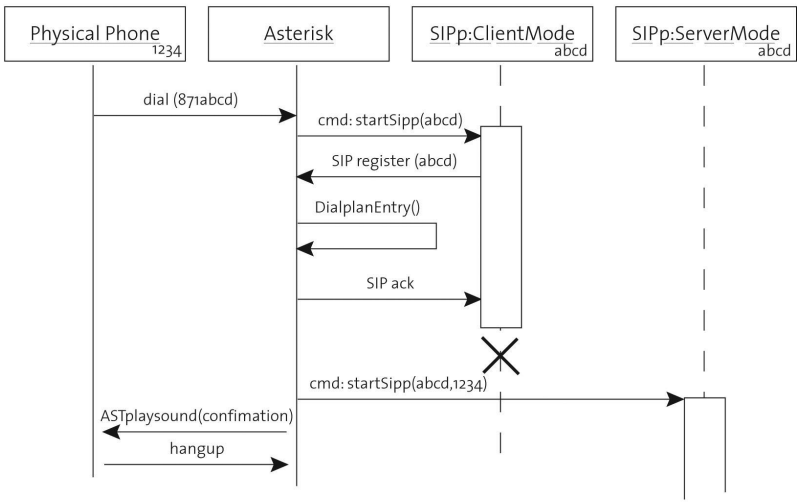

Figure 3: *The afflilation procedure started by a physical phone results in the initialization of a virtual SIP phone which registers to Asterisk. This is followed by adding an entry to the dialplan. After the start of the redirection server the user gets a confirmative feedback.*

Then a SIPp instance in server mode is started. This instance will be responsible for performing the actual redirection from the logical number to the physical device so at its initialization both numbers are passed. When the affiliation is finished a conformation sound is played to the user who then terminates the call.

2.1.2 Disaffiliation

The disaffiliation of a number is started according to the affiliation sequence by the code 870 and the logical number of the user (see Figure 4). Then Asterisk signalizes to the redirection server that it can stop. This can be done through a special SIP message or through a system call which kills the specific process. Afterwards we start a new SIPp instance in client mode which sends a further SIP registration message with an expiration time of zero. This has the effect that the number is removed from the dialplan. Finally an audible feedback is given to the user which acknowledges the disaffiliation procedure.

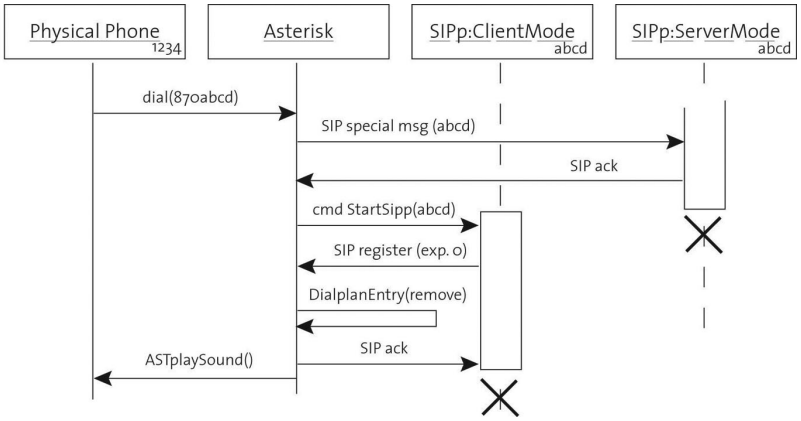

Figure 4: *According to the affiliation procedure the code 870 starts the disaffiliation of the number abcd. The redirection server is stopped and the dialplan entry is canceled by again sending a registration message, this time with expiration time zero.*

Since we have the mobility of the users within the network and we cannot expect the user to disaffiliate every time leaving a phone, we have to take into consideration what is happening when we have users which affiliate to different phones without disaffiliation. Of course it is required to find the latest affiliation for every user since the probability is very high that the user can be reached there (see Section 2.2.1).

2.1.3 Call setup

A call can be established by any phone in the network by dialing the logical number of the user (see Figure 5). The location determination is not considered in the shown sequence because it is done within the Asterisk object, which can also stand for multiple distributed Asterisk instances.

However, after dialing the number abcd, Asterisk forwards the call to the SIPp server which number is registered in the dialplan. The SIPp instance acts as a redirection server and replies with a "302 temporary moved" SIP message which contains the number of the physical phone - the user is affiliated to - in the contact field (for a SIP reference see [8]). So Asterisk forwards the call to the number 1234 which acknowledges the invitation. Finally the call information is returned to the originally inviting phone and the call is initialized.

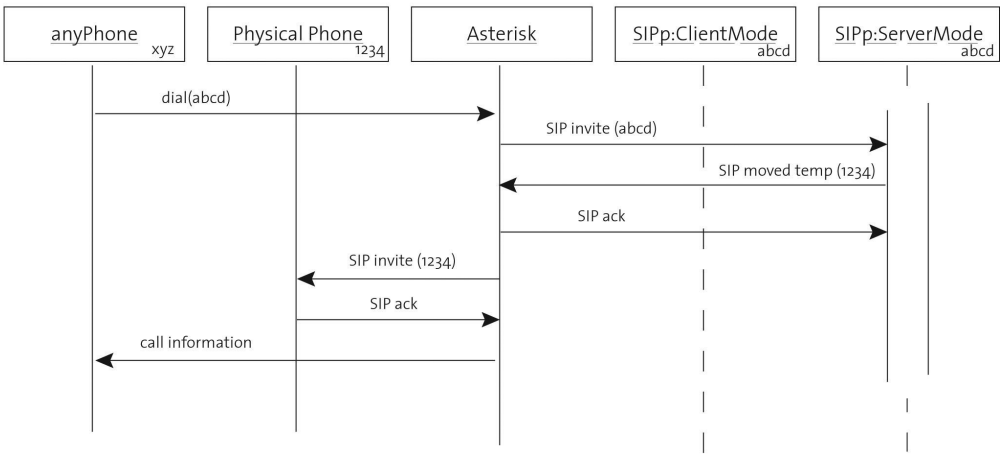

Figure 5: *When anyone in the network dials the number abcd it gets forwarded to the redirection server. It redirects the call to the physical phone the user abcd is affiliated to by sending a 302 SIP message.*

2.2 Node configuration

Every node in the network has the structure denoted in Figure 6. The nodes have one or multiple ethernet interfaces whereby to each an IP address is assigned. The routing between the nodes is done with an OSPF daemon. Communication protocols between the nodes are DUNDi, IAX (Inter Asterisk eXchange) and several VoIP (Voice over IP) protocols.

DUNDi is used for the dialplan discovery between the Asterisk instances. This protocol builds up a peer-to-peer system of trusted partners. Requests are forwarded to adjacent nodes until one knows the requested number or the whole network has been queried. IAX is responsible for transmitting calls between the Asterisk instances. Whatever protocol the subscribed devices use to communicate with Asterisk, the route between the participating Asterisk instances is bridged by the IAX protocol. DUNDi and IAX use the underlying UDP (User Datagram Protocol).

The core of Asterisk are the configuration files and the resultant dialplan. Asterisk is responsible for everything related to the handling of incoming and outgoing calls. They are generated by SIPp. The behavior of SIPp instances can be specified using XML scenarios.

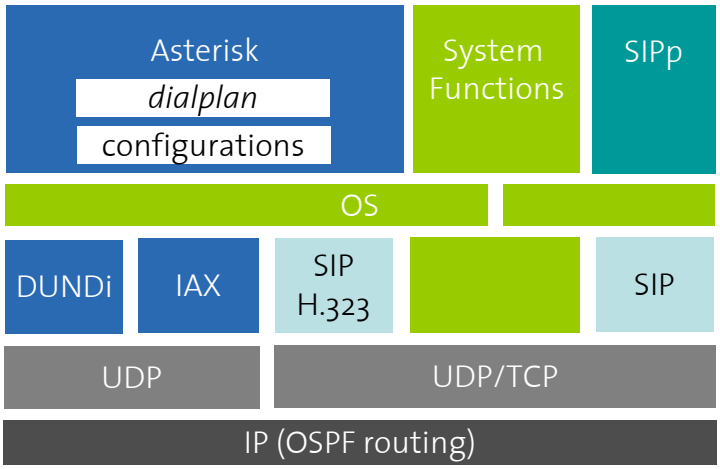

Figure 6: *On every node there is an Asterisk instance as well as some SIPp phones. Asterisk waits for events which are generated by SIPp phones through the SIP channel. SIPp phones themselves are controlled by system functions or shell scripts respectively.*

The last part is the system part of the node which is important when we want to execute predetermined test scripts. It controls the interaction between the different SIPp instances and Asterisk is starting the redirection server by calling a system function.

2.2.1 Asterisk

The core of Asterisk is the dialplan which is generated based on the configuration files. The most important one is the file `extensions.conf` (see Section A.1). Therein we specify all contexts, global variables, dialplan functions and macros.

Contexts are special parts of the dialplan which constitute independent sections which can be used in predefined situations by other parts of the dialplan like the SIP or the DUNDi part. For example the two DUNDi contexts in `extensions.conf` specifies the location where DUNDi can look up a number on the local or a remote host.

In the contexts either we include other contexts or we determine extensions. An extension denotes a step in a dialplan function sequence. Therefore in an extension we specify the (called) number or number pattern for which this extension should be executed, the sequence number of this extension and the in this step called function. So when we enter a context we search for extensions which are matching the called number, and execute the specified functions according to the sequence number of the extension.

A special context is the `[sipregistration]` context. It is specified as regcontext in the SIP configuration file `sip.conf` (see Section A.2). This means that a SIP phone which makes a registration is added to this `[sipregistration]` context at sequence number 1. Therefore the

sequence numbers in this context start at number 2. So when we call such a registered phone we enter the context `[sipregistration]` by using the before inserted extension with sequence number 1. The next matching extension is the one with sequence number 2 which matches for all five digit numbers in this context and which executes the call establishment to the required number.

The most important context is the `[affiliation]` context. Therein we specify what should be done when an affiliation is initiated by a phone calling the affiliation code 871 and the appended five digit number. Since a disaffiliation is optional when a user leaves a physical phone we have to make sure that we do not have multiple affiliations with the same logical number in the network. Therefore we add a loop to the affiliation procedure which is first looking up the number we have to affiliate. If the same number is found on other hosts in the network all affiliations are remotely disaffiliated and afterwards we can run the local affiliation of the users number (see Figure 7). Therefore we sequentially start two instances of a SIPp phone. The first one registers the logical number to Asterisk by sending a SIP REGISTRATION message. The second one is started in server mode and acts as redirection server for incoming calls.

For the tests Asterisk 1.4.4 is used. It could be installed as part of the debian standard distribution by running the aptitude command. To run Asterisk on UML it is important to first adjust the access authorization to all necessary directories. This is not done automatically in the UML file system.

2.2.2 DUNDi

The number discovery between the different PBXs is done with the DUNDi protocol. DUNDi is a fully distributed peer-to-peer system built as a network of communication servers. One server can look up a number at a trusted peer. If the queried peer does not know the number, it performs a query at its trusted peer and so on. A reply is sent back along the same route and every node on this route caches the result of this lookup. A further request then could be answered faster since more nodes know the location of the number.

The settings for DUNDi can be specified in the configuration file `dundi.conf` (see Section A.3). This file contains basically the following things: the identification data of the local host (e.g. the entity ID), the `[mappings]` context to specify where a DUNDi request is allowed to lookup numbers, and all data used to communicate with other DUNDi peers.

In the configuration there are several parameters which could be changed and have an effect on the performance of a DUNDi lookup. The cachetime (in seconds) can be set to a desired value so looked up numbers would be stored for that time in every node on the lookup route and a further lookup could be done in a shorter time using the cached information. Since we

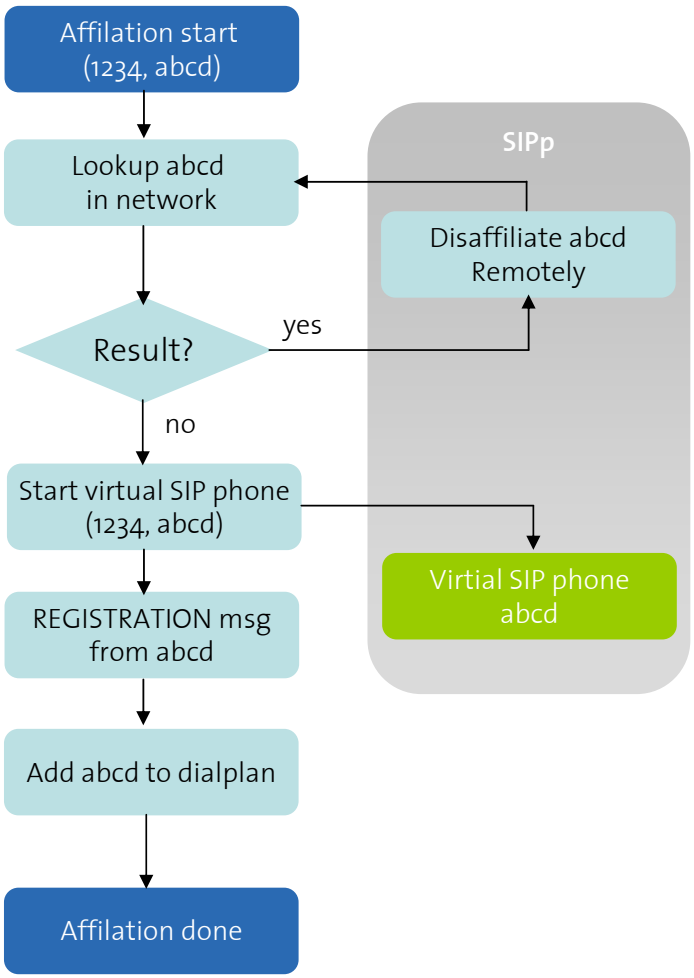

Figure 7: *Before affiliating a new number we check whether there are other affiliations with the same number and disaffiliate them. Afterwards the virtual SIP phone is started.*

have mobility in our network, we can not allow nodes to cache a destination which is probably invalid. So we set the cachetime to a low value.

A second parameter is the weight of a DUNDi mapping, which is the second parameter of a mapping context entry. This weight is used to decide which destination is called in the case of multiple DUNDi replies, whereby a smaller weight is preferred. A weight so far could be specified as static. The support of dynamic weight is already announced to be introduced in the next release of Asterisk (version 1.6). This would give us the possibility to transmit the expiration time of a SIP phone through a DUNDi response and we could automatically choose the latest registration of a number.

A third important parameter is the order of a trusted DUNDi peer. This can be set to primary, secondary, etc. A flat hierarchy yields a faster lookup of a number, i.e. a breadth-first search. The disadvantage of this setting is the generated network traffic and the less effective caching,

since we have shorter routes to a peer which knows the requested number.

It could be useful to declare every node in the network as DUNDi peer of a specific node, since in a real dynamic network we could not foresee which are our directly linked neighbors. While running the system only the directly connected nodes would become reachable for the local DUNDi system.

2.2.3 IAX

IAX is the Inter Asterisk eXchange protocol. Through this channel calls and call initialization messages are passed. The configuration is done in the configuration file `iax.conf` (see Section A.4). In this configuration it's important that IAX is binded to every network interface of the node. Otherwise Asterisk would not or just restricted be able to forward calls to destinations predetermined by DUNDi.

2.2.4 SIPp

SIPp is a testing tool for the SIP protocol and can act as a SIP server or client. This means that we can simulate every behavior of a SIP phone. It allows us to specify our own scenarios in simple XML files which contains the sequence of sent and received messages as well as the messages content.

In Section A.5 we see the scenario for the SIP redirection server which is started at the affiliation of a logical number. This scenario waits until it receives a SIP INVITE message for the assigned number at the specified port. Afterwards a SIP 100 Trying message followed by a SIP 302 Moved Temporarily message is sent which is containing the physical number in the contact field. The SIPp instance then waits for an acknowledgment. After receiving this, the scenario starts again and waits for an inviation message.

Starting multiple SIPp instances during the tests turned out to be a problem, since every instance has to be bound to a communication port. A solution for this problem is to specify the port as 0, so it takes the next free port by increasing ports from 5060. The solution for the required media port was to remove the checking algorithm in the SIPp source code. This was allowed because we don't need any media stream in our tests, since we just want to set up and terminate calls.

2.3 Tests

2.3.1 Network setup

The test network consists of 17 nodes whereby every node has one or multiple network interfaces according to Figure 8. There are also some subscribed physical phones which in the tests

would be initialized dynamically.

The network has the property that we can simulate many network failures by breaking a few links. So we reach a partition of the network or an isolated node by just deactivating one link.

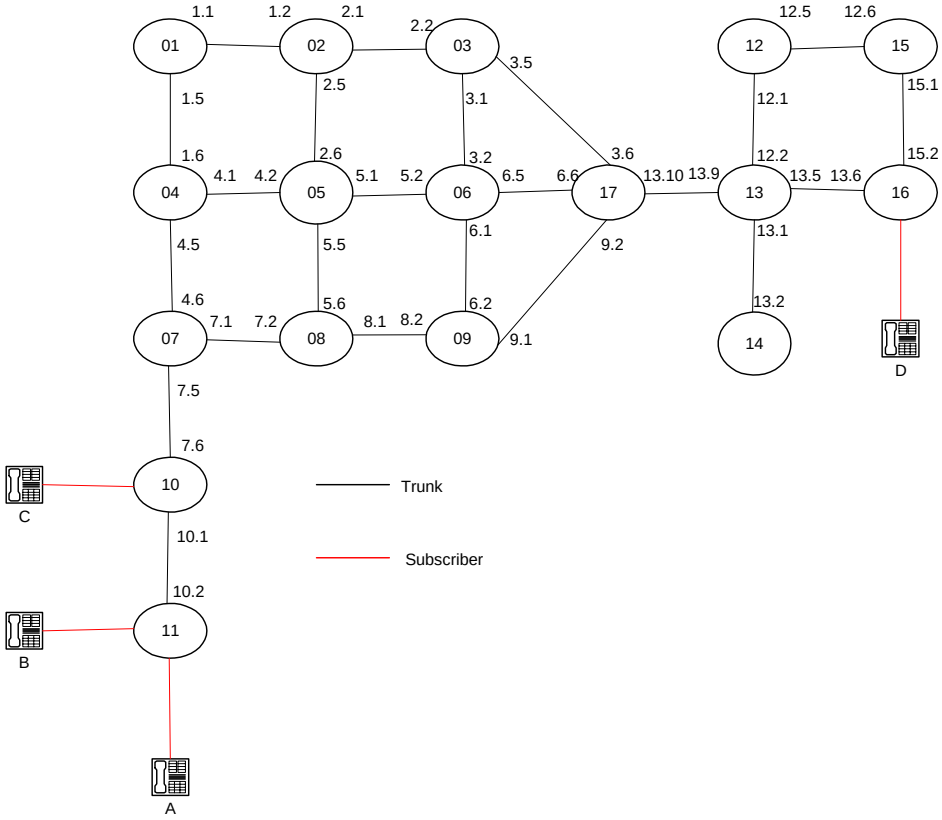

Figure 8: *The test network topology consists of 17 nodes and several subscribed phones.*

In the tests every node is represented by a UML instance and the links are implemented with a daemon which forwards every message to the adjacent node. On every node we install Asterisk and SIPp according to the description in Section 2.2. To make the installation efficient, we first install the programs to a dedicated file system. Afterwards we use this file system to start every single UML instance. So just the changes with respect to the original file system have to be saved for every instance, and we can install everything just once for all UMLs.

Since different UML instances have different configurations, after a first start of the UMLs we run an installation script which on one hand installs all required daemons and on the other hand copies all peer specific configuration files.

2.3.2 Testing types

The brought up approach should be tested on functionality and on robustness against node and link failures.

The first test sequence should determine the behavior of the system in executing the basic functions. The detailed functionality test sequences can be seen in Section B.1.1. We assume the test network as seen in Figure 8 without any failures.

We first test the number discovery without and with affiliation, so in the first case we should not get any result whereas in the second case we get the response with the users location. Further the disaffiliation procedure is tested whereby a user or a SIPp phone respectively affiliates its number. Then a disaffiliation and a new affiliation to another node is made. Independent of the users mobility we should get the latest affiliation when we look up the users number. Also the mobility of a user without disaffiliation is tested.

In the second test sequence we test the behavior in the case of isolated nodes, partitioned network and node and link failures (see the test sequences in Section B.1.2). An interesting situation is the mobility within partitions or over partition boundaries. Also interesting is the case when we have only a single way to the destination left because of link failures.

2.3.3 Test execution

To run the tests we first have to boot all UML instances on a PC. Then on every node Asterisk is started and allowed to reach steady state. We assume that before the start all installations as well as the configuration updates has been done.

The first test script we start on a node in the test network. So we can transmit all commands via a remote shell instruction which is a fast approach. In this script we start other shell scripts which make affiliations, disaffiliations and the establishment of calls. In the scripts mainly the interaction of the different SIPp instances is controlled.

Since we have network failures in the second test and we can not reach all other nodes any more, the second test script is executed on the host PC and the commands are submitted to each host through a TCP connection which is built up for every command we want to transmit.

To monitor the ongoing tests and to have a protocol afterwards, we write a log file on the host PC (see the test protocols in Section B.2). This file is written by the test script and the different SIPp instances. For instance at a lookup, the calling SIPp instance writes to the log file that it is starting a request. The called SIPp representing a physical phone logs its physical number

everytime it is called. The calling part then logs when it gets an acknowledge from the dialed partner.

So from the log file we will see what affiliations have been done, what numbers we tried to look up and to which physical numbers the requested numbers have been mapped. The comparison of the expected and the obtained number will give us the result of the test.

3 Results

3.1 General Remarks

We can state that we have successfully implemented the required functions affiliation and disaffiliation as well as that we have found a procedure to make the mapping of the logical to the physical phone numbers. So we were able to extend the static behavior of Asterisk and DUNDi with dynamic features. This is done by introducing a virtual SIP device for every affiliating user. The SIP phone takes the role of a redirection server which forwards an incoming call to the accurate physical phone where the user is located. So we could adopt the concept of a device independent user into the device-oriented view of Asterisk.

We also tested multiple settings in the usage of the DUNDi protocol and checked the effects of different parameter values. In the end we have a configuration which is a practical solution for a telephony system in wireless networks.

Further we have found a convenient way to emulate a test network and simulate subscribing phones on a Linux PC. The affiliation functions as well as the simulation of the SIP phones works with a reasonable time consumption. In the tests the establishment of the connections for transmitting the test commands to the UML instances and the timing between the different SIPp instances took most of the consumed time.

3.2 Test network

One goal of the work was the setup of the test network with UML. After several technical difficulties we now have a working setup which can easily be rebuilt for further tests. One of the main difficulty was the bounded memory of a UML instance to running processes. A workaround for this was to kill the started SIPp processes as soon they are not used any more during the tests. So it was possible to run the whole sequence in a row.

3.3 Test results

According to the test sequences in Section B.1 we have executed the test scripts. The generated log file can be seen in Section B.2. According to this protocol we see that the functionality test yields the correct results. In the case that no number is returned at a lookup, Asterisk 1.4 answers with a SIP 503 Service Unavailable message to the requesting phone, other than earlier releases of Asterisk which made a distinction between different request failures.

The second part of the tests could also be executed correctly except for two problem cases (therefore see Sections 3.3.1 and 3.3.2). These failures are based on incompletely implemented features in Asterisk and could be fixed with little effort by adjusting the source code of Asterisk.

3.3.1 Problem A: Static DUNDi weight

A first problem which appeared in the tests is the static DUNDi weight we have seen in Section 2.2.2. This becomes a real problem when we have a partitioned network (see Figure 9) whereby in at least two partitions an affiliation is done. The first affiliation is not removed since the destination is not reachable for the second host. So after restoring the network, at a lookup we have multiple DUNDi answers whereby in most cases the first incoming answer is taken since all DUNDi weights are static and equal. A fix of this problem is coming up in the next Asterisk release (version 1.6). The new feature would allow us to dynamically set the DUNDi weight for every new request. For example we can set it dependent of the expiration time so in the case of multiple answers we can find the latest affiliation in the network.

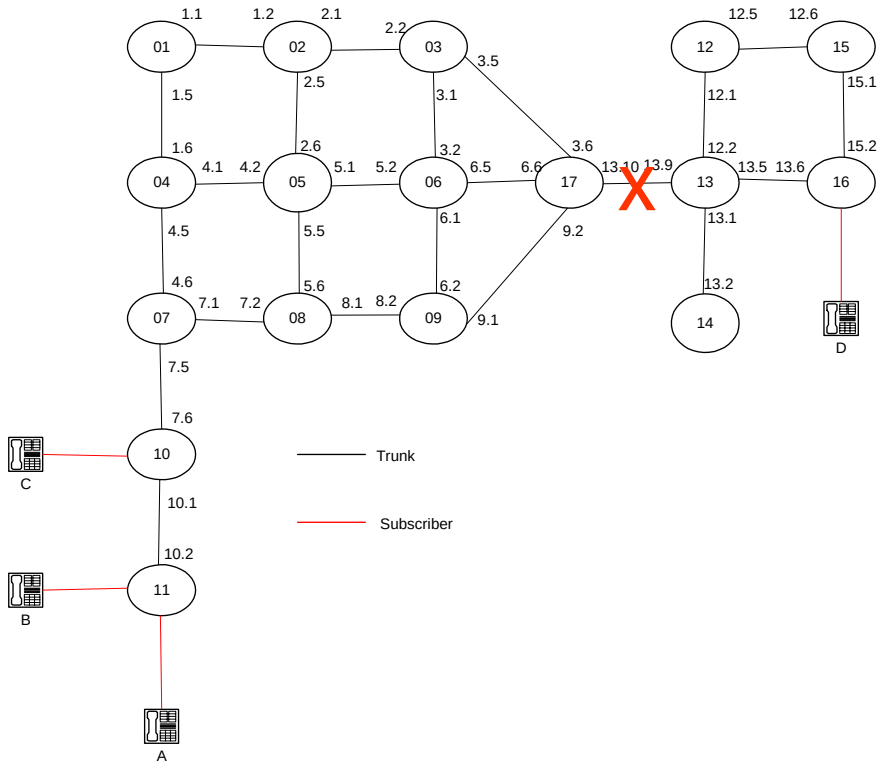

Figure 9: *Problem case A: At first the link between node 13 and 17 is broken. When we have mobility between the partitions and the failed links are restored, on a lookup we have multiple DUNDi answers which do not yield a basis of decision to chose the right one.*

3.3.2 Problem B: DUNDi protocol malfunction

The second problem case is based on the DUNDi protocol or the interpretation thereof by Asterisk respectively. We have a look at the situation showed in Figure 10.

For example node 7 starts a request of a number which is located at node 16. So the only possible route for node 7 to look up a number which is affiliated to this number is 7-4-1-2-5-8.

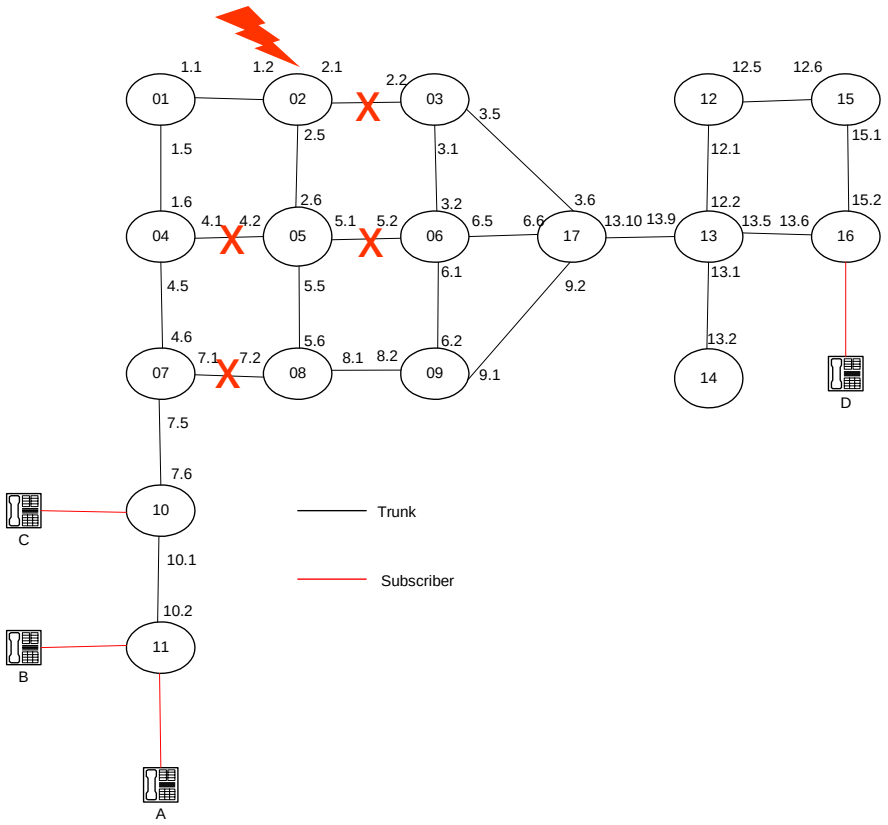

Figure 10: *Problem case B: A bug in Asterisk interpreting the DUNDi protocol leads in this situation to the problem that a request form node 7 is not forwarded at node 2 because of the presence of node 5 in the list of already queried nodes.*

According to the DUNDi protocol, when we send a request the protocol attaches the route the message has taken to the payload. So at every node an entry is made that the message has passed this specific node. Additionally the node adds the peers which are known as trusted partners and to which the request is forwarded to. This is done by reason of aborting the lookup flooding procedure when all nodes are reached.

The problem with this is now that Asterisk adds the neighbour peers even if they are unreachable or offline at the moment. So in the above example the flooding stops at node 2 because node 4 added node 5 to the already queried list. This leads to the malfunction in failure test 11 (see B.1.2). This problem must be a bug in Asterisk or in DUNDi respectively and may be solved by adjusting the source code of Asterisk.

3.4 Timing and time consumption

For the tests it turned out to be necessary to give enough time to the different events. Since the communication between the host and the UML instances for example is done over a TCP connection, it takes a while to initiate it. Another reason is that different programs are writing

in the log-files and we want to make sure that the order of this log messages doesn't get mixed up. So most of the used time has to be spent because of the testing environment.

Another aspect which takes some time concerns the communication between the Asterisk instances or the DUNDi protocol respectively. Namely when one or more links are stopped and restored, it takes a while till the adjacent nodes recognize the new DUNDi peers. This could take up to a minute and have to be considered within the test sequence.

A last point which takes time is the affiliation procedure in the Asterisk dialplan. Depending on how many other affiliations with the same number are found, we have to wait some time to finish the remote disaffiliation. The wait values in the affiliation procedure are chosen very tolerant. It would be possible to shorten the affiliation procedure by waiting just half of the specified time.

Taking into account these points the functionality test script takes about 15 minutes and the failover test script takes about 20 minutes. By testing the affiliation and call procedure manually in the test environment, one can see that affiliations are accomplished in a short acceptable time span and calls are established in an almost real time manner.

4 Conclusion

To provide the accessibility of a user in a telephony network, call route discovery is necessary. Additionally, when we have mobile users which are independent of the physical phones, we need number discovery which determines the phone the user with his number is affiliated to.

In this work we brought up an approach how these two things can be realized with the open source PBX Asterisk and the number discovery protocol DUNDi which is implemented in Asterisk. We have adopted the static device-oriented behavior of Asterisk to be able to handle users which are independent of physical phones and are mobile within the network. This was done by introducing a virtual SIP phone for every affiliating user. Since SIP phones are given the possibility to register dynamically to the Asterisk dialplan we use them as substitute for the users. The virtual SIP phones act as redirection server and forward the incoming calls to the accurate phone which does not have to use the SIP protocol.

This extension of Asterisk was necessary to make use of the DUNDi protocol which directly accesses the PBX' dialplan. So we now are able to discover the number through the DUNDi protocol. Aditionally we showed the effects of different DUNDi parameter settings to the behavior of the system. E.g. we can set the caching time of the PBXs on the path on which a lookup is performed. So we can adjust the sluggishness of the system and the reaction time on disaffiliating users. Further we can adjust for example whether DUNDi should perform a breadth-first or a depth-first search. So to perform a call to a user which is affiliated to a remote PBX, Asterisk starts a DUNDi request to its trusted partners and so is able to detect the current system state at every call. DUNDi yields the IP address of the PBX the user is affiliated to. With this address and the underlying OSPF routing we also have found the implicit route to the target phone.

We can state that it was possible to improve the existing flooding based approach by implementing an event driven procedure using DUNDI and Asterisk. This leads to a better performance with respect to traffic overhead and call setup delay.

Tests on the implementations functionality and stability were performed in an emulated network realized with UML instances on a single PC. We installed a test network of 17 nodes and several physical phones and users which affiliate and disaffiliate. The tests showed that we could meet the determined requirements. So our system manages for example multiple affiliations of the same user without disaffiliation in between. Also the number discovery was successful even in the case of multiple link or node failures. Two exceptions are represented by the pointed out incomplete implementations of the DUNDi protocol by Asterisk. These bugs could be fixed with little effort in the case of realizing the telephony system with the DUNDi protocol.

Future work has to be done in evaluating other approaches to perform call route discovery in wireless mesh networks. So one could use multicasting to map the logical user numbers to the

physical phone numbers. Advantages and disadvantages of the different approaches should be investigated and evaluated.

References

[1] Mark Spencer et. al., The Asterisk Handbook - Version 2, http://www.digium.com/-handbook-draft.pdf, last visited 07/2007.

[2] Leif Madsen et. al., The Asterisk Documentation Project: Volume One: An Introduction to Asterisk, http://www.asteriskdocs.org/modules/tinycontent/-content/docbook/current_v1/docs-pdf/vm1.pdf, last visited 07/2007.

[3] J.R. Richardson, Using DUNDi with a Cluster of Asterisk Servers, http://www.astricon.net/files/usa06/Friday-General_Conference/-JR_Richardson_Whitepaper.pdf, last visited 07/2007.

[4] Mark Spencer, DUNDi Internet Draft, http://www.dundi.com/dundi.txt, last visited 05/2007.

[5] Antti Kallaskari (ASCOM Finland), Mobile numbers in Access Node networks.

[6] Andrew Lunn (ASCOM Switzerland), Nomadic Telephony - Techniques for Locating Nomadic Users.

[7] Andrew Lunn (ASCOM Switzerland), Nomadic Telephony - Throw away Prototype System Test Plan.

[8] J. Rosenberg et. al., SIP: Session Initiation Protocol (Reference), http://www.ietf.org/rfc/-rfc3261.txt, last visited 07/2007.

[9] Olivier Jacques, SIPp reference documentation, http://sipp.sourceforge.net/-doc/reference.html, last visited 07/2007.

A Configuration files

A.1 extensions.conf

```
[general]
static=yes
writeprotect=no
autofallthrough=yes
clearglobalvars=no

[globals]
LOCALPORT=5200
MEDIAPORT=6011

[local]
include=>affiliation
exten=>_XXXXX,1,Dial(SIP/${EXTEN})
exten=>_XXXXX,2,Macro(dundi-lookup,${EXTEN})

[priv-local]
include=>sipregistration

[affiliation]
exten=>_871XXXXX,1,Set(DEST=${DUNDILOOKUP(${EXTEN:3},virtsip,b)})
exten=>_871XXXXX,2,GotoIf($["${DEST}" = ""]?6:3)
exten=>_871XXXXX,3,System(sh /etc/asterisk/sippscr/rdaff ${EXTEN:3} ${DEST})
exten=>_871XXXXX,4,Wait,4
exten=>_871XXXXX,5,Goto(1)
exten=>_871XXXXX,6,System(sh /etc/asterisk/sippscr/aff_init ${EXTEN:3} ${LOCALPORT}
   ${MEDIAPORT})
exten=>_871XXXXX,7,Wait,2
exten=>_871XXXXX,8,System(sh /etc/asterisk/sippscr/aff_serv ${EXTEN:3} ${CALLERID(num)}
   ${LOCALPORT} ${MEDIAPORT})
exten=>_871XXXXX,9,Set(GLOBAL(LOCALPORT)=$[${LOCALPORT}+1])
exten=>_871XXXXX,10,Set(GLOBAL(MEDIAPORT)=$[${MEDIAPORT}+10])
exten=>_871XXXXX,11,Answer
exten=>_871XXXXX,12,Wait,5
exten=>_871XXXXX,13,Hangup

[default]
include=>local

[sipregistration]
exten=>_XXXXX,2,Dial(SIP/${EXTEN},10)
exten=>_XXXXX,3,Hangup
exten=>_XXXXX,103,Hangup

[dundi-incoming]
include=>sipregistration
```

```
[dundi-lookup]
switch=>DUNDi/virtsip



;------------------------------------------------------------------
;MACRO-BLOCK
;------------------------------------------------------------------


[macro-dundi-lookup]
;check local context first, then lookup in a dundi context
exten=>s,1,Goto(${ARG1},1)
include=>priv-local
include=>dundi-lookup
```

A.2 sip.conf

```
[general]
context = local
allow=all
regcontext=sipregistration
bindport=5060
autocreatepeer=yes
insecure=very
```

A.3 dundi.conf

Code example from node 4 in the test network:

```
[general]
organization=ETHZ
locality=Zurich
country=CH
email=awangler@ee.ethz.ch

bindaddr=0.0.0.0
port=4520

entityid=FF:00:00:00:00:04

cachetime=2

ttl=32

autokill=yes
```

```
[mappings]
virtsip=>sipregistration,100,IAX2,dundi:${SECRET}@${IPADDR}/${NUMBER}

;04->01
[FF:00:00:00:00:01]
model=symmetric
host=192.168.1.5
outkey=keyuml4
inkey=keyuml1
include=all
permit=all
qualify=yes
order=primary

;04->05
[FF:00:00:00:00:05]
model=symmetric
host=192.168.4.2
outkey=keyuml4
inkey=keyuml5
include=all
permit=all
qualify=yes
order=primary

;04->07
[FF:00:00:00:00:07]
model=symmetric
host=192.168.4.6
outkey=keyuml4
inkey=keyuml7
include=all
permit=all
qualify=yes
order=primary
```

A.4 iax.conf

Code example for node 4 in the test network:

```
[general]
port=4569

bindaddr=192.168.1.6
bindaddr=192.168.4.5
bindaddr=192.168.4.1
```

```
[dundi]
type=friend
dbsecret=dundi/secret
context=sipregistration
disallow=all
allow=ulaw
allow=g726
allow=all

[priv]
type=friend
dbsecret=dundi/secret
context=dundi-incoming
disallow=all
allow=ulaw
allow=g726
allow=all
```

A.5 SIPp scenario of redirection server

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="Asterisk Wait INVITE">

  <recv request="INVITE" crlf="true">
  </recv>

  <send>
    <![CDATA[
  SIP/2.0 100 sip:[local_ip] SIP/2.0
  Via: SIP/2.0/UDP [local_ip]:[local_port]
  Max-Forwards:5
  [last_To:]
  [last_From:]
  Call-ID: [call_id]
  [last_CSeq:]
    ]]>
  </send>

  <send>

    <![CDATA[
  SIP/2.0 302 sip:[local_ip] SIP/2.0
  Via: SIP/2.0/UDP [local_ip]:[local_port]
  Max-Forwards:5
  [last_To:]
```

```
  [last_From:]
  Call-ID: [call_id]
  [last_CSeq:]
  Contact: <sip:[physnr]@[local_ip]>
  Expires: 10
  Content-Length:0
    ]]>
  </send>

  <recv response="200" optional="true">
  </recv>

  <recv request="ACK">
  </recv>

</scenario>
```

B Tests

B.1 Test sequences

B.1.1 Function testing

1. Not affiliated
   - The network is started and allowed to reach steady state.
   - Telephone A looks up the number 19990.

   ⇒ No physical telephone is returned
2. Affiliation on same switch
   - The network is started and allowed to reach steady state.
   - The number 19990 is affiliated to telephone B.
   - Telephone A looks up the number 19990.

   ⇒ The physical telephone number 11020 is returned
3. Affiliation on a neighbour switch
   - The network is started and allowed to reach steady state.
   - The number 29990 is affiliated to telephone C
   - Telephone A looks up the number 29990.

   ⇒ The physical telephone number 10010 is returned
4. Affiliation on a remote Switch
   - The network is started and allowed to reach steady state.
   - The number 39990 is affiliated to telephone D.
   - Telephone A looks up the number 39990.

   ⇒ The physical telephone number 16010 is returned
5. Mobility to the same switch with disaffiliation
   - The network is started and allowed to reach steady state.
   - The number 49990 is affiliated to telephone C.
   - The number 49990 is disaffiliated from telephone C.
   - The number 49990 is affiliated to telephone B.
   - Telephone A looks up the number 49990.

   ⇒ The physical telephone number 11020 is returned

6. Mobility to a neighbour switch with disaffiliation

   - The network is started and allowed to reach steady state.
   - The number 59990 is affiliated to telephone B.
   - The number 59990 is disaffiliated from telephone B.
   - The number 59990 is affiliated to telephone C.
   - Telephone A looks up the number 59990.

   $\Rightarrow$ The physical telephone number 10010 is returned

7. Mobility to a remote switch with disaffiliation

   - The network is started and allowed to reach steady state.
   - The number 69990 is affiliated to telephone B.
   - The number 69990 is disaffiliated from telephone B.
   - The number 69990 is affiliated to telephone D.
   - Telephone A looks up the number 69990.

   $\Rightarrow$ The physical telephone number 16010 is returned

8. Mobility to the same switch without disaffiliation

   - The network is started and allowed to reach steady state.
   - The number 79990 is affiliated to telephone C.
   - The number 79990 is affiliated to telephone B.
   - Telephone A looks up the number 79990.

   $\Rightarrow$ The physical telephone number 11020 is returned

9. Mobility to a neighbour switch without disaffiliation

   - The network is started and allowed to reach steady state.
   - The number 89990 is affiliated to telephone B.
   - The number 89990 is affiliated to telephone C.
   - Telephone A looks up the number 89990.

   $\Rightarrow$ The physical telephone number 10010 is returned

10. Mobility to a remote switch without disaffiliation

    - The network is started and allowed to reach steady state.
    - The number 99990 is affiliated to telephone B.
    - The number 99990 is affiliated to telephone D.
    - Telephone A looks up the number 99990.

    $\Rightarrow$ The physical telephone number 16010 is returned

B.1.2 Failover and Recovery Testing

1. Isolated switch, reachable affiliation

   - The network is started and allowed to reach steady state.
   - Node 10 is stopped
   - The number 18880 is affiliated to telephone B.
   - Telephone A looks up the number 18880.

   ⇒ The physical telephone number 11020 is returned

2. Isolated switch, unreachable affiliation

   - The network is started and allowed to reach steady state.
   - Node 10 is stopped
   - The number 28880 is affiliated to telephone D.
   - Telephone A looks up the number 28880.

   ⇒ No affiliation information is returned

3. Isolated pair of switches, reachable affiliation

   - The network is started and allowed to reach steady state.
   - Node 07 is stopped
   - The number 38880 is affiliated to telephone C.
   - Telephone A looks up the number 38880.

   ⇒ The physical telephone number 10010 is returned

4. Isolated pair of switches, unreachable affiliation

   - The network is started and allowed to reach steady state.
   - Node 07 is stopped
   - The number 48880 is affiliated to telephone D.
   - Telephone A looks up the number 48880.

   ⇒ No affiliation information is returned

5. Partitioned network, reachable affiliation

   - The network is started and allowed to reach steady state.
   - Link 1317 is removed
   - The number 58880 is affiliated to telephone C.

- Telephone A looks up the number 58880.

⇒ The physical telephone number 10010 is returned

6. Partitioned network, unreachable affiliation

   - The network is started and allowed to reach steady state.
   - Link 1317 is removed
   - The number 68880 is affiliated to telephone D.
   - Telephone A looks up the number 68880.

   ⇒ No affiliation information is returned

7. Isolated node, recovering from partitioning, no mobility

   - The network is started and allowed to reach steady state.
   - Link 1011 is removed and the network is allowed to reach steady state.
   - The number 78880 is affiliated to telephone D.
   - Link 1011 is restored and the network is allowed to reach steady state.
   - Telephone A looks up the number 78880.

   ⇒ The physical telephone number 16010 is returned

8. Isolated node, recovering from partitioning, with mobility

   - The network is started and allowed to reach steady state.
   - The number 88880 is affiliated to telephone C.
   - Link 1011 is removed and the network is allowed to reach steady state.
   - The number 88880 is affiliated to telephone D.
   - Link 1011 is restored and the network is allowed to reach steady state.
   - Telephone A looks up the number 88880.

   ⇒ The physical telephone number 16010 is returned

9. Recovering from partitioned network, no mobility

   - The network is started and allowed to reach steady state.
   - Link 1317 is removed and the network is allowed to reach steady state.
   - The number 98880 is affiliated to telephone D.
   - Link 1011 is restored and the network is allowed to reach steady state.
   - Telephone A looks up the number 98880.

   ⇒ The physical telephone number 16010 is returned

10. Recovering from partitioned network, with mobility

    - The network is started and allowed to reach steady state.
    - The number 08880 is affiliated to telephone C.
    - Link 1317 is removed and the network is allowed to reach steady state.
    - The number 08880 is affiliated to telephone D.
    - Link 1317 is restored and the network is allowed to reach steady state.
    - Telephone A looks up the number 08880.

    $\Rightarrow$ The physical telephone number 16010 is returned

11. Operation with many link failures

    - The network is started and allowed to reach steady state.
    - The number 17770 is affiliated to telephone B.
    - The links 0708, 0405, 0203, 0506, 0917, 0617 and 1316 are removed and the network is allowed to reach steady state.
    - The number 17770 is affiliated to telephone D.
    - Telephone A looks up the number 17770.

    $\Rightarrow$ The physical telephone number 16010 is returned

12. Operation with many node failures

    - The network is started and allowed to reach steady state.
    - The number 27770 is affiliated to telephone B.
    - The nodes 05, 06, 08, 09, 12, 14 and 15 are stopped and the network is allowed to reach steady state.
    - The number 27770 is affiliated to telephone D.
    - Telephone A looks up the number 27770.

    $\Rightarrow$ The physical telephone number 16010 is returned

B.2 Test protocols

B.2.1 Functionality testing

```
--------------------------------------------------
ASTERISK NUMBER AFFILIATION AND DISCOVERY TEST
Mon Jun 18 12:47:38 UTC 2007
--------------------------------------------------

DISAFFILIATION OF ALL RECENT NUMBERS
```

```
192.168.10.2 disaffiliated number 19990
192.168.7.6 disaffiliated number 29990
192.168.15.2 disaffiliated number 39990
192.168.10.2 disaffiliated number 49990
192.168.7.6 disaffiliated number 59990
192.168.15.2 disaffiliated number 69990
192.168.10.2 disaffiliated number 79990
192.168.7.6 disaffiliated number 89990
192.168.15.2 disaffiliated number 99990

1. NOT AFFILIATED
-----------------------------------------------------
expected number: no number
192.168.10.2 tries to look up 19990
lookup gives no result (503 service unavailable)

2. AFFILIATION ON SAME SWITCH
-----------------------------------------------------
expected number: 11020
192.168.10.2 affiliated 19990 to 11020
192.168.10.2 tries to look up 19990
192.168.10.2 mapped number to 11020
...lookup successful

3. AFFILIATION ON NEIGHBOUR SWITCH
-----------------------------------------------------
expected number: 10010
192.168.7.6 affiliated 29990 to 10010
192.168.10.2 tries to look up 29990
192.168.7.6 mapped number to 10010
...lookup successful

4. AFFILIATION ON REMOTE SWITCH
-----------------------------------------------------
expected number: 16010
192.168.15.2 affiliated 39990 to 16010
192.168.10.2 tries to look up 39990
192.168.15.2 mapped number to 16010
...lookup successful

5. MOBILITY TO THE SAME SWITCH WITH DISAFFILIATION
-----------------------------------------------------
expected number: 11020
192.168.7.6 affiliated 49990 to 10010
192.168.7.6 disaffiliated number 49990
192.168.10.2 affiliated 49990 to 11020
192.168.10.2 tries to look up 49990
192.168.10.2 mapped number to 11020
```

```
...lookup successful

6. MOBILITY TO A NEIGHBOUR SWITCH WITH DISAFFILIATION
-----------------------------------------------------
expected number: 10010
192.168.10.2 affiliated 59990 to 11020
192.168.10.2 disaffiliated number 59990
192.168.7.6 affiliated 59990 to 10010
192.168.10.2 tries to look up 59990
192.168.7.6 mapped number to 10010
...lookup successful

7. MOBILITY TO A REMOTE SWITCH WITH DISAFFILIATION
--------------------------------------------------
expected number: 16010
192.168.10.2 affiliated 69990 to 11020
192.168.10.2 disaffiliated number 69990
192.168.15.2 affiliated 69990 to 16010
192.168.10.2 tries to look up 69990
192.168.15.2 mapped number to 16010
...lookup successful

8. MOBILITY TO THE SAME SWITCH W/O DISAFFILIATION
-------------------------------------------------
expected number: 11020
192.168.7.6 affiliated 79990 to 10010
192.168.10.2 affiliated 79990 to 11020
192.168.10.2 tries to look up 79990
192.168.10.2 mapped number to 11020
...lookup successful

9. MOBILITY TO A NEIGHBOUR SWITCH W/O DISAFFILIATION
----------------------------------------------------
expected number: 10010
192.168.10.2 affiliated 89990 to 11020
192.168.7.6 affiliated 89990 to 10010
192.168.10.2 tries to look up 89990
192.168.7.6 mapped number to 10010
...lookup successful

10. MOBILITY TO A REMOTE SWITCH W/O DISAFFILIATION
--------------------------------------------------
expected number: 16010
192.168.10.2 affiliated 99990 to 11020
192.168.15.2 affiliated 99990 to 16010
192.168.10.2 tries to look up 99990
192.168.15.2 mapped number to 16010
...lookup successful
```

B.2.2 Failover and Recovery Testing

```
--------------------------------------------------
ASTERISK FAILOVER AND RECOVERY TEST
Mon Jun 18 18:09:11 CEST 2007
--------------------------------------------------

192.168.10.2 disaffiliated number 18880
192.168.7.6 disaffiliated number 28880
192.168.7.6 disaffiliated number 38880
192.168.7.6 disaffiliated number 58880
192.168.15.2 disaffiliated number 38880
192.168.15.2 disaffiliated number 28880
192.168.15.2 disaffiliated number 48880
192.168.15.2 disaffiliated number 68880
192.168.15.2 disaffiliated number 78880
192.168.15.2 disaffiliated number 88880
192.168.15.2 disaffiliated number 98880
192.168.15.2 disaffiliated number 08880
192.168.15.2 disaffiliated number 17770
192.168.15.2 disaffiliated number 27770

1. ISOLATED SWITCH, REACHABLE AFFILIATION
--------------------------------------------------
expected number: 11020
Node 10 stopped.
192.168.10.2 affiliated 18880 to 11020
192.168.10.2 tries to look up 18880
192.168.10.2 mapped number to 11020
...lookup successful

2. ISOLATED SWITCH, UNREACHABLE AFFILIATION
--------------------------------------------------
expected number: no number
Node 10 stopped.
192.168.15.2 affiliated 28880 to 16010
192.168.10.2 tries to look up 28880
lookup gives no relult (503 service unavailable)

3. ISOLATED PAIR OF SWITCHES, REACHABLE AFFILIATION
--------------------------------------------------
expected number: 10010
Node 7 stopped.
192.168.7.6 affiliated 38880 to 10010
192.168.10.2 tries to look up 38880
192.168.7.6 mapped number to 10010
...lookup successful
```

```
4. ISOLATED PAIR OF SWITCHES, UNREACHABLE AFFILIATION
-----------------------------------------------------
expected number: no number
Node 7 stopped.
192.168.15.2 affiliated 48880 to 16010
192.168.10.2 tries to look up 48880
lookup gives no relult (503 service unavailable)


5. PARTITIONED NETWORK, REACHABLE AFFILIATION
-----------------------------------------------------
expected number: 10010
Link 13-17 stopped.
192.168.7.6 affiliated 58880 to 10010
192.168.10.2 tries to look up 58880
192.168.7.6 mapped number to 10010
...lookup successful

6. PARTITIONED NETWORK, UNREACHABLE AFFILIATION
-----------------------------------------------------
expected number: no number
Link 13-17 stopped.
192.168.15.2 affiliated 68880 to 16010
192.168.10.2 tries to look up 68880
lookup gives no relult (503 service unavailable)

7. ISOL. NODE, RECOV. FROM PART. NETWORK
-----------------------------------------------------
expected number: 16010
Link 10-11 stopped.
192.168.15.2 affiliated 78880 to 16010
Link 10-11 restored.
192.168.10.2 tries to look up 78880
192.168.15.2 mapped number to 16010
...lookup successful

8. ISOL. NODE, RECOV. FROM PARTIT., WITH MOBILITY
-----------------------------------------------------
expected number: 16010
192.168.7.6 affiliated 88880 to 10010
Link 10-11 stopped.
192.168.15.2 affiliated 88880 to 16010
Link 10-11 restored.
192.168.10.2 tries to look up 88880
192.168.15.2 mapped number to 16010
...lookup successful
```

```
9. RECOVERING FROM PART. NETWORK, NO MOBILITY
-----------------------------------------------------
expected number: 16010
Link 13-17 stopped.
192.168.15.2 affiliated 98880 to 16010
Link 13-17 restored.
192.168.10.2 tries to look up 98880
192.168.15.2 mapped number to 16010
...lookup successful

10. RECOVERING FROM PART. NETWORK, WITH MOBILITY // problem case A
-----------------------------------------------------
expected number: 16010
192.168.7.6 affiliated 08880 to 10010
Link 13-17 stopped.
192.168.15.2 affiliated 08880 to 16010
Link 13-17 restored.
192.168.10.2 tries to look up 08880
192.168.7.6 mapped number to 10010
...lookup successful

11. OPERATING WITH MANY LINK FAILURES // problem case B
-----------------------------------------------------
expected number: 16010
192.168.10.2 affiliated 17770 to 11020
Links are stopped.
192.168.15.2 affiliated 17770 to 16010
192.168.10.2 tries to look up 17770
192.168.10.2 mapped number to 11020
...lookup successful

12. OPERATING WITH MANY NODE FAILURES
-----------------------------------------------------
expected number: 16010
192.168.10.2 affiliated 27770 to 11020
Links are stopped.
192.168.15.2 affiliated 27770 to 16010
192.168.10.2 tries to look up 27770
192.168.15.2 mapped number to 16010
...lookup successful
```