Advanced Techniques for Passive Testing of Communication Protocols

Baptiste ALCALDE

December 4, 2006

Contents

List of Figures

Thank you all

Introduction

The life-cycle of every product can be decomposed in several steps. Commonly, the product emmerge from the idea of a person or a group of person. Then the idea is put down in words, for instance in english. Afterwards, the text is read by people - often different than the one who wrote it - in order to give a reality to this idea, i.e. to product an implementation of it. Last but not least, when the product is supposed to be finished it undergoes a set of tests in order to insure that it behaves as expected. This sequence of actions applies also to the case of telecommunication protocols and services.
Everyone can agree that between the idea and the final product a succession of transformations, translations, and interpretations occures, and that is why the last phase of the process - namely testing - is altogether important. The testing activity is simultaneously a phase a provider can't get around but which is often done with few methodology in practice. Paradoxally, it exists various methodologies, algorithms and tools in order to process the testing activity automatically, such those based on formal specifications. The automatization of the testing is crucial for several reasons : first the telecommunication protocols are permanently and inexorably increasing in complexity and due to a huge hardware and sofware heterogeneity the control of such systems overcomes the human skills. Secondly, it enable the provider to save time and money in the process and it is far to be meaningless as the costs for testing are said to reach one third of the budget in average.
When we speak about testing that implies the presence of two entities : a specification and an implementation. Briefly, the implementation behaviours are observed after a possible stimulation and are then compared with the ideal system that is to say the specification. Testing is then an activity in which we try to guaranty the protocol or service to process without fault - at least ideally. The most known testing family is the active testing. It consists in generating testbeds - automatically - from the specification of the protocol in order to cover as much flaws as possible, and then execute them on an implementation of this protocol. The analysis of the implementation behaviours after this stimulation will give infomation about the implementation validity. There are even various algorithms to reduce the number or the size of the testbeds and simultaneously increase the fault coverage. But these tests are processed on a limited duration and only minimise the number of faults because the complete checking is impossible. The other testing family is the passive testing or also refered to as monitoring and is the topic of this work.
Last but not least, we must note that testing is a wide topic. Indeed, we can test different implementation architectures and for various purposes. The tests can process on

IUTs where we have access to the internal code — that activity is called *white box testing* — or on the contrary where we only have access to the interfaces of the system — the *black box testing*. The tests can focus on various problematics such as the performances, the robustness, the interoperability, etc... In this work we only take into account the black box conformance testing, where conformance testing refers to the activity of showing that the IUT executions follow all the properties defined in the corresponding specification.

In the present work we propose to study the passive testing techniques and give some improved methods.
There are two main known way to test passively a network protocol or service using formalisms : by interval determination and by invariant. Thus after studying the weak points of each we may be able to highlight few ways to resolve them.
The principle of passive testing is as follows : event traces (i.e. sequence of message exchanges) from the implementation are recorded and are automatically maped to the specification or properties of the specification (namely the invariants). No message is produced and given to the implementation under test (called IUT), and we suppose that the IUT is taken without knowledge of its internal state that is to say that we don't suppose the event trace record to start from the initial state or a predefined state of the EFSM.
The interval determination technique for passive testing is presented in [14]. The concept is simple. We take the first event couple of the trace, we search some transitions in the EFSM specification that hold these events. Then we process as if we fire these transitions and we record the results in structures called Candidate Configuration Sets (CCS). If no transition is found, a fault is detected i.e. this couple is not defined in the specification. If at least one transition is found we process it and we refine the intervals of the variables afterwards. If then an inconsistency appears in the intervals, an error is highlighted. The process is repeted starting from the CCS resulting of the last event analysis for the next events in the trace till the trace is shown to be faulty or till its end - in which case the trace is said to be valid.
The invariant approach is a bit different. It is presented in the works [1] and [4]. The idea is that a set of invariants represents the most relevant expected properties of an implementation. By definition an invariants is a property that must be verified at any time in the IUT process. These invariants can be automatically generated from the specification or — what is generally more interesting — given by an expert of the tested protocol or service. In the case the invariant is given by an expert, it should be firstly check on the specification. When we command some checked invariants we can then map them on any recorded event trace of the given IUT.

Our contribution in the domain of passive testing consists in giving new improvements for a better or faster error detection in network protocols or services. The work uses a systematic approach since we take both available passive testing techniques and search for improvements.
In the case of the invariant approach we show that it is not able to detect output errors in EFSM and then we present a technique using logical programming theory concepts in

order to extract constraints about the values of the variables for each invariants. Then the invariants contain both a control and a data portion to be matched in the implementation event traces, contributing in detecting more faults.
In the case of interval determination we prove that the approach is not able to detect every transfert errors. We present a backward approach based on the same idea i.e. trying to determine the intervals of the variables and check for unconsistencies. The backward checking technique is original and powerful. It is more costy in terms of complexity than the forward approach but it is exhaustive and in the worst case as complex as the best known exhaustive forward technique. In order to give an additional advantage to this approach a parallelized version of the algorithm is given, leading to the first exhaustive passive testing algorithm with a linear complexity.
Then, after the presentation of these theoretical improvements we show the feasibility of such approach by its application to a real complex communication protocol.
To this purpose we have chosen the IP Multimedia Subsystem (IMS). IMS is a standardised Next Generation Networking (NGN) architecture for telecom operators that want to provide mobile and fixed multimedia services, using a Voice-over IP (VoIP) implementation based on a 3GPP standardised implementation of Session Initiation Protocol (SIP), and runs over the standard Internet Protocol (IP). The IMS should enable all the current as well as the future services that the Internet provides.
In order to apply our techniques a formal model (EFSM) of the IMS has been extracted from the informal specifications (written in english) of the 3GPP. Next, we record real traces of the IMS thanks to the IMS Playground deployed at Fokus Fraunhofer. Last, we match the traces on the IMS formal specification with help of a Backward Checking algorithm implementation.

The present thesis is organised in five chapters. The first chapter present the basic concepts that will be used further. It begins with the presentation of the different testing families, i.e. active and passive testing, as well as their advantages and drawbacks. Then we define the error types and introduce the Extended Finite State Machine concept. Afterwards, we draw the state of the art of passive testing, explaining the value determination approach, the interval determination approach and the invariant-based one. Last, an exemple is given to underline the deficencies of each approaches.
The second chapter present a first possible improvement of passive testing, based on invariants. We define the notions that were inherited of the logical programming — namely substitution, unification, and normalization. We present the invariants and how to extract the contraints on them and we finally show an application example.
In the third chapter, the reader will find the second possible improvement : the backward checking approach. We give an overview of this approach before entering more in detail and show how to process an EFSM transition in a backward manner — which is the basis of processing a whole trace backward — and we prove the algorithm termination and compute the maximal complexity.
The fourth chapter is an improvement of what was found in the third one, in terms of time complexity and with the use of parallelization. We propose three algorithms in or-

der to parallelize the backward checking approach and compare the performances of these algorithms in order to make emerge the best one. Then we show the obtained gain in complexity by comparing with the sequencial backward checking algorithm and with a theoretical application on various communication protocols : SCP, INRES, OLSR, and OSPF.
In the last chapter before we take conclusion and give some paths for future works, we present backward checking tools and their application to the IMS protocol. This chapter is the occasion to see how the new approach reacts in the context of a real complex protocol, the use of an existing IMS platform — the IMS Playground at Fokus-Fraunhofer — and real traces.

Chapter 1

State of the art of the passive testing

To perform a conformance testing there is not only one way. It is commonly admitted to classify them in two main testing families : the active testing and the passive testing. Each of these two families encompasses various approaches, and each approach contains different techniques. The basic notions of the two testing families are introducted in the begining of this chapter, trying to underline the positive and negative aspects of them.

Then, in this chapter we study the different passive testing techniques. We begin in the section 1.3 with the presentation of the Finte State Machine (FSM) and Extended Finite State Machines (EFSM) formalisms. In the section 1.4, we take a look at a naive simple algorithm for error detection by determination of the value of variables. Then we present in 1.5 a more advanced algorithm based on determination of the interval of the variables. Finally, the invariant-based approach is shown in the section 1.6.

1.1 The two conformance testing families

1.1.1 The active testing

Principle

This family of conformance testing tries to show the conformance relation between an implementation and a specification by putting the implementation into specific situations and a known environment through processing chosen sequences of inputs. The results of this event sequences processed on the implementation is then compared with the expected behaviour, that is to say the processing of the same sequence on the specification. In the testing community, it is refered to black-box testing to qualify a test with no information about the intern structure of the implementation. For the problems we consider, we make the assumption that we are in this situation of black-box testing.

In a common active testing approach there are two steps :

- the (automated) generation of test cases with help of the specification,

- the processing of these test cases on the implementation and the conformance verdict is given after the analysis of the result.

The figure 1.1 models this process.

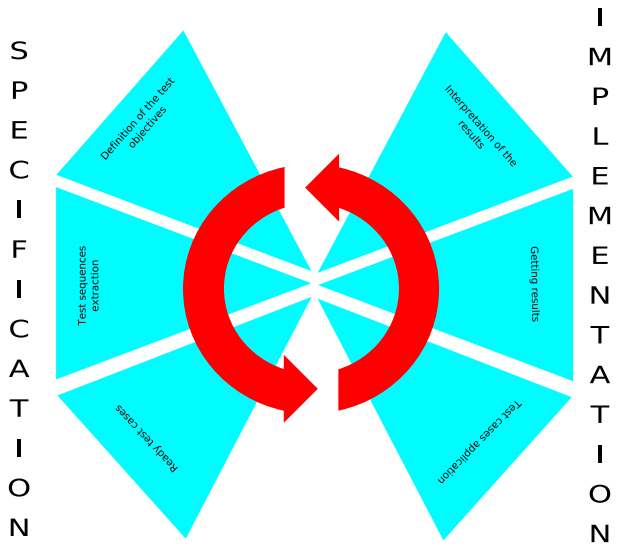

Figure 1.1: The two steps of an active testing approach

Advantages and drawbacks

The strength of the active testing is in the ability to focus on particular aspects of the implementation. Indeed, the test cases can be for instance limited to a specific type of error, of to an important state of the specification.
On the other hand, the test case choice and production can turn out to be complicated.

1.1.2 The passive testing

Principle

The passive testing consist in observing the input and output events of an implementation in run-time. It should not disturb the natural run-time of a protocol or service, that is why it is called passive testing. It is sometimes also refered to as monitoring. The record of the event observation is called an event trace. This event trace will be applied to the specification in order to determine the confomance relation between the implementation and the specification. We should keep in mind that when an event trace conforms to the

specification it doesn't mean that the whole implementation conforms to the specification, but on the other hand, if a trace doesn't conform then the implementation neither.

The figure 1.2 schematize this approach.

Advantages and drawbacks

The passive testing has the huge advantage not to influence the system under test. Contrary to active testing, there is no injected probe messages. That is crucial because this probe messages modify the environment, and may trouble or in a worse case cause the crash of the tested protocol or service. So, the passive testing approach seems to be an ideal tool for doing a control directly on the implementation in natural run-time conditions (monitoring). In addition in this approach the tests can be run all protocol life long in contrast with the active testing test campains that must be run on a limited duration.

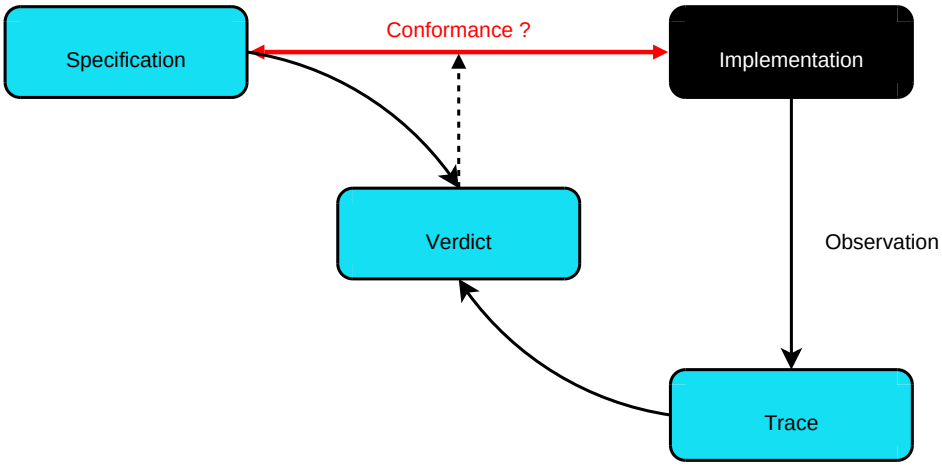

Figure 1.2: The passive Testing

Nevertheless, the available algorithms are not much efficient and the goal of the present work will be to give improvements of them.

1.1.3 A life-cycle philosophy for fault-free products

Each of the different testing families have their pro and contra as presented in the thereabove section, but the goal is not to oppose them. On the contrary, we must take advantage of both approaches in order to insure more reliability in the communication protocol or service.
At this point we can also underline the important role that formal methods can play. For

instance, many works have been made in active testing to automate the test case generation from formal specifications, and most of passive testing approaches are based on formal specifications. It has to be noted that verification techniques (which are outside the present work scope) also use formal methods.
Thus, a complete architecture can be proposed in order to provide fault-free protocols or services or at least to converge to this goal. Verification, active testing end passive testing could then act like successive filters : the verification phase showing conceptual faults (live-lock, deadlock, etc...), the active testing highlighting possible errors in the expected most usual scenari and in the majority or essential behaviours, and finally the passive testing finding inevitable residual flaws.

1.2 The different error types

The goal of conformance testing is to check that an implementation (the system under test) performs all what is described in its specification (that is supposed to be correct). In consequence, in order to realise the conformance testing, an implementation as well as a specification are needed. To minimise wrong interpretations of the specification it is preferably describe with help of a formal language or formal model.

The comformance testing is then similar to a comparison between two elements. When the elements are not corresponding the implementation is said to be not conform to the specification. The possible kind of errors are classified following the three types below (and also represented graphically with figure 1.3):

- **output error** : the output of a transition in the implementation differs from the corresponding one in the specification,
- les **transfert error** : the ending state of a transition in the implémentation differs from the corresponding one in the specification,
- les **mix error** : the output and ending state of a transition in the implementation differ from the corresponding ones in the specification.

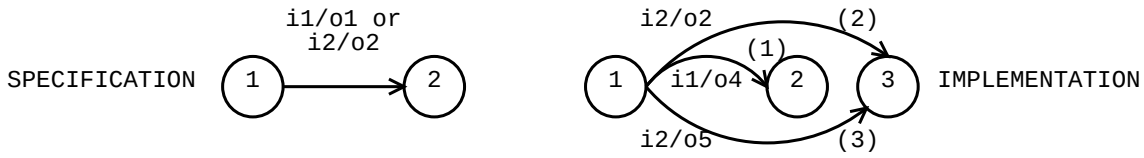

Figure 1.3: Error types

1.3 Finite State Machines (FSM)

The various passive testing methods that we consider are design with the help of specification formal models. The simplest formal model is the **Finite State Machine** representation (FSM) which is also called **automata**. It exists more complex representations of network protocols and services for instance **Extended Finite State Machines** (EFSMs). These notions are defined in the following chapter.

1.3.1 Definition of a FSM

A FSM can be intuitively seen as the representation of the possible changes of the system state to some other state, caused by the reception or sending of a message. The links between two states are called **transitions**.
There are two categories of FSMs :

1. the deterministic FSMs (**DFSM**), for a given state and a given event, there is only one reachable state,
2. the non deterministic FSMs (**NDFSM**), for a given state and a given event, it can exist several reachable states.

Formal definition of a DFSM

A DFSM M is a 7-tuple $M = (I, O, S, s_o, S_f, \delta, \lambda)$, where :

- I is a finite non empty set of input symbols
- O is a finite set of output symbols
- S is a finite non empty set of states
- s_o is the initial state
- S_f is the set of final states
- $\delta : S \times I \longrightarrow S$ is a transition function that brings the system into the state s_j when reading the event e in the state s_i $((s_i, e) \in S \times I)$
- $\lambda : S \times I \longrightarrow O$ is a transduction function that produces the output o when reading the event e in the state s_i $((s_i, e) \in S \times I)$

So, when the machine is in the state s and receive the input symbol a then it moves to the state define by $\delta(s, a)$ producing the output described by $\lambda(s, a)$.

Formal definition of a NDFSM

A NDFSM M is also a 7-tuple $M = (I, O, S, s_o, S_f, \delta, \lambda)$, but :

- I is a finite non empty set of input symbols
- O is a finite set of output symbols
- S is a finite non empty set of states
- s_o is the initial state
- S_f is the set of final states
- $\delta : S \times I \longrightarrow 2^S$ is a transition function that enables the system in a state s_i with the input e ($(s_i, e) \in S \times I$) to reach a set of states $S_j \in 2^S$
- $\lambda : S \times I \longrightarrow 2^O$ is a transduction function that associate a sequence of outputs to the state s_i and the input e ($(s_i, e) \in S \times I$)

So, when the machine is in the state s and receive the input symbol a then it moves to ONE of the state define by $\delta(s, a)$ producing the output described by $\lambda(s, a)$.

1.3.2 EFSM definition

The EFSM representation is an evolution of FSM. Variables are used to reduce FSM models and to make them more compact. Thus, each transition can contain :

- input and output events eventually with parameters
- a predicate (or guard) to satisfy
- a sequence of actions to perform

The Extended Finite State Machine M is formally a 7-tuple $M = (S, s_0, S_f, I, O, \vec{x}, T)$, where :

- S is a finite non empty set of states
- s_o is the initial state
- S_f is a finite set of final states

- I is a finite set of input symbols, with or without parameters
- O is a finite set of output symbols, with or without parameters
- $\vec{x} = (x_1, \ldots, x_k)$ is a vector denoting a finite set of variables
- T is a finite set of transitions

Each transition t is defined by a 6-tuple $t = (s_t, f_t, i_t, o_t, P_t, A_t)$, where :

- s_t is a starting state
- f_t is an ending state
- i_t is an input symbol
- o_t is an output symbol
- $P_t(\vec{x})$ is a predicate on the variables (boolean formula)
- $A_t(\vec{x})$ is a sequence of actions

Remark : The event parameter setting in the EFSM model constitute the major problem of passive testing since the variables can take whichever value and the system under test can be in whichever state before the trace record starts.

1.4 Passive testing by value determination

The EFSM model offers more confort and possibilities than the classical FSM model for specifications, and this with help of the event parameter setting, the predicates and the actions hold on the transitions. In counterpart, the passive testing methods should consider not only checking the correctness of event sequences but also the one of the values of the variables and parameters. The following subsections presents a simple way of value determination in order to find faults.

1.4.1 Principle

In the EFSM model, it is possible to deduct the values of the variables from an event trace. The scheme 1.4 shows us an example of this process. Let assume that we know the current state ① but that the value of x is unknown. If the next input/output couple of the trace is $a/1$ then we deduct that after the transition is fired the current states becomes the state ③ and x is equal to 0. It is on this property that the following algorithm is based. This algorithm is also deeper explained in [32]. According to this algorithm, a transition is fired if :

- the input/output couple of the trace map with the one of the transition,
- the transition predicate is true or can not be evaluated because of a lack of information (values are not yet known).

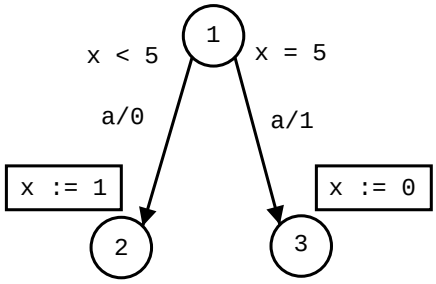

Figure 1.4: Deducting the values of the variables

1.4.2 The problem of information loss

We note that till the time it stays undefined variables (i.e. no value is found) there is a risk of losing the values already found for other variables. The figure 1.5 illustrate this phenomenon.

In this example we consider the current state to be ①, and that x has been already defined with the value 3. Since y is unknown we must for any case fire the two transitions ①⟶② and ①⟶③ since the I/O on both transitions are identical. Now, the two transitions give different values to x so that x becomes UNDEFINED !

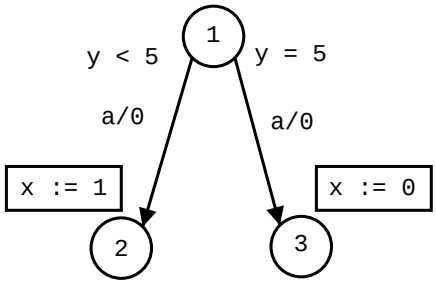

Figure 1.5: Information loss

1.4.3 The algorithm

The algorithm process in two main steps. The first step is called **homing phase** of the current state and the values of the variables. In this step, we use the following rules to change the values of the variables :

- for a given I/O couple, if it exists few possible transitions producing different values for a variable then this variable becomes UNDEFINED,
- a predicate is concerned with an UNDEFINED variable it is considered to be true (in the doubt),

The second step is called **fault detection phase** and controls the conformance of the trace remaining with the specification. In the following figures, we show respectively the simple algorithm pseudo-code (1.6), the $A(\vec{x})$ algorithm which add the action vector $\vec{x}$ to the list $newX$ (1.7), and the $Synt(NewX)$ algorithm which make the synthesis of the possible values (1.8).

1.5 Passive testing by interval determination

1.5.1 Principe

We saw that the simple algorithm suffers from an information loss phenomenon. We will see later that this information loss is highly prejudicial to the fault detection. Considering the deficiencies of the first algorithm, a more efficient (and also more complex) algorithm for fault detection was developped and presented in [14]. This algorithm make use of three tools :

1. **intervals** in order to refer to the set of values of the variables. It follows the [21] formalism,
2. **assertions**, that are boolean formula recording additional information about the variables,
3. **candidate configurations sets** formalising the analysed environment of the system under test.

This algorithm tries to determine the values of the variables not using anymore a single value assignation but a set (in the form of an interval) of possible values for each variable. It refine as much as possible the intervals in which the variables take their values.

1.5.2 The intervals

The **intervals** are a beginning of an answer to the information loss problem. Indeed, in the simple algorithm, a variable was not allowed to have more than one possible value. When few values were possible for a variable the variable was said to be undefined. With help of intervals, a variable v whose value is contained between two integers a and b will be defined by the interval $R(v)$ so that : $R(v) = [a; b]$. In the particular case where v is a constante value a, we have : $R(v) = [a; a]$. We then say that this variable v is **decided**.

In this algorithm we also need three operations on intervals :

Input : an EFSM specification and an event trace σ from the implementation

Output : verdict σ is conform to the specification or not

/* Homing Phase */

1. L $\longleftarrow$ $s_1, \ldots, s_n$; /*possible current states*/
2. L' $\longleftarrow$ $\emptyset$; k $\longleftarrow$ 1; end_homing $\longleftarrow$ FALSE;
3. **while** (end_homing = FALSE) **do**
4. .$newX \longleftarrow \emptyset$; /*list of the possible values*/
5. .**for each** state s of L **do**
6. . .**for each** transition t starting from state s with the output o_k **do**
7. . . .**if** $eval(P_t, \vec{x}) = TRUE$ **then do** /* P_t is the transition predicate */
8. . . . .add $\delta(s, a_k, P_t)$ in L'; /* a_k is the transition input */
9. . . . .add $A_t(\vec{x})$) to $newX$; /* cf. the adding algorithm */
10. .$L \longleftarrow L'$;
11. .$L' \longleftarrow \emptyset$;
12. .$\vec{x} \longleftarrow$ Synt($newX$); /* cf. the synthesis algorithm */
13. .**if**($|L| = 0$) **then do** /* empty set of current states */
14. . .**return** FALSE;
15. .**else if**(($|L| = 1$) and ($\nexists\, x_i \in \vec{x}$, x_i = UNDEFINED)) **then do**
16. . .end_homing $\longleftarrow$ TRUE;
17. .$k \longleftarrow k + 1$;

/* Fault Detection Phase*/

18. /*$L = s$*/
19. $r \longleftarrow 1$;
20. **while**(($o_{k+r} = \lambda(s, a_{k+r}, P_t)) \wedge (P_t(\vec{x})) \wedge$ (end_of_trace = FALSE)) **do**
21. .$s \longleftarrow \delta(s, a_{k+r}, P_t)$;
22. .$\vec{x} \longleftarrow A_t(\vec{x})$;
23. .$r \longleftarrow r + 1$;
24. **return** end_of_trace;

Figure 1.6: The simple algorithm

1. $\vec{y} \longleftarrow \vec{x}$;
2. **for each** y_i in $\vec{y}$ **do**
3. .**if**$((x_i = \text{UNDEFINED}) \wedge (\exists\, p \in P_t,\, p \equiv (x_i = val_i)))$ **then do**
4. . .$y_i \longleftarrow val_i$;
5. $\vec{y} \longleftarrow A_t(\vec{y})$;
6. add y to $newX$;

Figure 1.7: The adding algorithm

1. **for each** x_i in $\vec{x}$ **do**
2. .$V \longleftarrow$ list of variables y_i in $newX$
3. .**if** at least one possible value of x_i is UNDEFINED **then do**
4. . .$x_i \longleftarrow$ UNDEFINED;
5. .**if** at least two possible values of x_i are different **then do**
6. . .$x_i \longleftarrow$ UNDEFINED;
7. .**else do**
8. . .$x_i \longleftarrow val_i$; /* all the possible values are identical */

Figure 1.8: The synthesis algorithm

1. the **sum** of two intervals. We have : $[a;b] + [c;d] = [a+c; b+d]$,
2. the **substration** of two intervals. We have : $[a;b] - [c;d] = [a-d; b-c]$,
3. the **multiplication** of an intervals by an integer :
   $w \times [a;b] = [w \times a; w \times b]$ if $w \geq 0$
   $w \times [a;b] = [w \times b; w \times c]$ if $w < 0$

1.5.3 The assertions

Definition 1 *An assertion $asrt(\vec{x})$ is a boolean formula on the vector of variables $\vec{x}$ that must be true at the current state of the analysis.*

The **assertions** are used to record constraints on variables. These constraints can arise from transition predicates and actions. This way, if a transition is fired then its predicate is added to the assertion as well as updates (i.e. actions) that contain undecided variables in the right member of the equalty. For instance, the action $x_2 \longleftarrow x_1 + 1$ updates x_2. In this case every term of $asrt(\vec{x})$ containing x_2 must be deleted and the term $x_2 \longleftarrow x_1 + 1$ must be added to $asrt(\vec{x})$. As soon as we discover the value of x_1 we deduct trivially the x_2 one.

In order to manipulate it with ease, the assertions are given under a Disjunctive Normal Form (DNF).

Definition 2 *A logical boolean formula is considered to be in the Disjunctive Normal Form if and only if it is a disjunction (sequence of "or"s) of one or more conjunctions ("and"s) of one or more literals (i.e., statement letters and negations of statement letters).*

In our concern, a statement letter represent an inequation. Converting a boolean formula to DNF involves using logical equivalences of the classical logic, such as the Double Negative Law (a double negation is equivalent to no negation), De Morgan's Law, and the Distributive Law. We can note that all logical formulas can be converted into DNF.

Definition 3 *According to De Morgan's Law, let $\bigvee$ represent "or" , $\bigcup$ represent "and", and $\_$ represent "not". Then, for two statement letters E and F :*

- $(E \bigcup F) = \_E \bigvee \_F$ *and,*
- $(E \bigvee F) = \_E \bigcup \_F$

1.5.4 The Candidate Configuration Sets

Definition 4 *A Candidate Configuration Set (CCS) is a triplet $(s, R(\vec{x}), asrt(\vec{x}))$, where :*

- *s is the current specification state,*

- $R(\vec{x})$ *is the set of constraints on the variables (i.e. intervals),*
- $asrt(\vec{x})$) *is an assertion on* $\vec{x}$*.*

The **candidate configurations** model much more than the state in which the system under test is. They also show the different constraints on the variables. For instance, the following configuration $(1, \{R(x) = [1; 5]\}, x < 2 \wedge x > 2)$ means that the system is in the state ① and that the value of x is contained between 1 and 5 but not equal to 2.

The algorithm uses two lists Q_1 and Q_2, Q_1 being the set of current possible CCS and Q_2 the possible CCS of the previous step. From Q_1 and an event e, we must obtain the corresponding transitions. A transition t will be fired if it exists a configuration in Q_1 whose constraints (the intervals of the variables and the assertion) are compatibles with the predicate p of t.

There can be several possible configurations at a given moment. The algorithm consider fixing a threshold N of the maximal amont of possible configurations at a given time. Beyond this threshold few similar configurations can be combined in one. The tests have shown $3n$ - where n is the number of possible CCSs before the trace analysis - to be a reasonable threshold.

1.5.5 The algorithm

The lines 8 to 12 of the algorithm test the fulfilment of the $t.predicate$ predicate requirements by contradiction. According to the definition of an assertion, $c_t.asrt(\vec{x})$ is always true for $c_t.R(\vec{x})$. Then at the line 8 we suppose that $t.predicate$ is true. In this case the conjunction of $c_t.asrt(\vec{x})$ and $t.predicate$ must be true. Next we check the fulfilment of the conjunction requirements with the process **check_consistency** studied later on. If there exists an unconsistency then we deduct that $t.predicate$ is false and the transition cannot be fired.

1.5.6 Consistency checking and interval refinement

The goal of the **check_consistency** process is to refine the intervals of the variables functions of the constraints on these variables. The refinement allow us to control the existence of a configuration. Indeed if an interval becomes empty after refinement then it means that at least one statement letter of the studied assertion is false. Consequently the configuration is impossible.

Before presenting the **check_consistency** algorithm two notions should be introduced : the **normalized predicates** and the **interval refinement**.

Definition 5 *A statement letter is said to be a nomalized predicate if it has the form :* $a_1x_1 + a_2x_2 + \ldots + a_kx_k \sim Z$*, where :*

- $\sim$ *is a comparator in the set* $\{<, >, \leq, \geq, =, \neq\}$*,*

Input : An EFSM specification and a trace from its implementation

Output : notice of a fault

Q_1 : the set of possible current configurations

Q_2 : the set of possible configurations at the next step

$|Q_2|$: cardinal of Q_2

begin

1. $Q_1 \longleftarrow \{ccs_0, \ldots, ccs_{n-1}\}$, where for each state s_i,
   . . . $.ccs_i \longleftarrow (s_i, R(\vec{x}) = unknown, asrt(\vec{x}) = \emptyset)$
2. **while**(more event e) **do**
3. $.Q_2 \longleftarrow \emptyset$;
4. **.for each** CCS $c \in Q_1$ **do**
5. . **.for each** transition t, where $t.event = e \wedge t\_start\_state = c.state$ **do**
6. . . .replace the variables in $t.predicate$ by their values if they are decided;
7. . . .**if** $t.predicate$ is false **then do go to** 5;
8. . . .build a new CCS $c_t \longleftarrow c$ and add $t.predicate$ to $c_t.asrt(\vec{x})$;
9. . . .**check_consistency**(c_t);
10. . . .**if** there are unconsistencies in c_t **then do**
11. . . . .delete c_t, **go to** 5;
12. . . .**do_actions**(c_t, $t.action$);
13. . . .$c_t.state \longleftarrow t.end\_state$, $Q_2 \longleftarrow Q_2 \cup \{c_t\}$;
14. **.if** $Q_2 = \emptyset$ **then do**
15. . **.return** "Detected fault";
16. **.if** $|Q_2| > N$ **then do**
17. . .combine similar CCS;

end

Figure 1.9: The interval determination algorithm

- $x_1, x_2, \ldots, x_k$ *are variables*
- $a_1, a_2, \ldots, a_k, Z$ *are integer values*

We can also rewrite a normalized predicate under the form of an interval equation, i.e. : $\sum_i a_i R(x_i) = R(\sim Z)$, where $R(\sim Z)$ is the interval of $\sim Z$. For instance the interval of $\leq Z$ is $]-\infty; Z]$.

In order to normalize a statement letter the decided variables are replaced by their values. Next the constants are moved to the right member of the inequation and are moved to the left member. For instance, the predicate $7x_2 - x_3 < 9 + x_1$, with $x_3 = 2$ will be normalized to $(-1)x_1 + 7(x_2) < 11$. We rewrite this predicate with intervals this way : $(1)R(x_1) + 7R(x_2) =]-\infty; 10]$.

Reminder : the assertions are DNFs. In the assertions we consider that the statement letters are normalized predicates.

Definition 6 *Let the equation* $\sum_i a_i R(x_i) = R(\sim Z)$. $R'(x_j)$. *The refinement of* $R(x_j)$ *is defined as follows :*

$$R'_l(x_j) \longleftarrow \frac{R(\sim Z) - \sum_{i \neq j}(a_i R(x_i))}{a_j} \cap R_l(x_j)$$

The **check_consistency** algorithm is represented on the figure **??**.

The conjunctive terms are treated separetly (line 3) in this algorithm and the results are combined in the end (line 31). In each conjunctive term the predicates are checked and the intervals of the variables are refined in an iterative fashion. Indeed, each time an interval changes the algorithm replay the loop at the line 5. Thus, the algorithm ends when no interval modifiaction takes place. The refinement operation is assured to be a strictly monotonous operation and, since les values of the variables are integer, the algorithm finishes within a fintie number of steps.

A flag is used to note that at least one conjunctive tern is true. If the flag is set to false then the configuration is unconsistent. Else, the intervals of the variables are combined according to the following rule :

Assignation rule : for each variable definition domain, if there is a non-empty set of intervals issued from the treatment of the conjunctive terms then the new definition domain of the variable is the one whose lower bound (resp. upper bound) is the minimum (resp. maximum) of every interval lower bounds (resp. upper bounds).

1.5.7 Processing the actions

The variables can be modified when a transition is fired . The following process execute the transition actions modifiing the variables. A particular care should be taken with the actions such that $w \longleftarrow f(w, u, v, \ldots)$. After the action, $f^{-1}(w)$ will remplace w in $asrt(\vec{x})$.

Input : c, the configuration to be refined and checked

Output : FALSE if c is unconsistent, and TRUE else

begin

1. transform $c.asrt(\vec{x})$ into a DNF;
2. $flag \longleftarrow FALSE$;
3. **for each** conjunctive term D_l in $c.asrt(\vec{x})$ **do**
4. .$R_l(\vec{x}) \longleftarrow c.R(\vec{x})$;
5. .**for each** statement letter p in D_l **do**
6. . .$normalize(p)$; /* into a $\sum_i(a_i x_i) \sim Z$ form */
7. . .**if** $\sum_i(a_i R_l(x_i)) \subseteq R(\sim Z)$ **then do** /* p is TRUE */
8. . . .delete p from D_l; /* p gives no new constraint - we ignore it */
9. . . .**go to** 5;
10. . .**if** $\sum_i(a_i R_l(x_i)) \cap R(\sim Z) = \emptyset$ **then do** /* p is FALSE */
11. . . .delete $Rl(\vec{x})$ et D_l;
12. . . .**go to** 3;
13. . .**for each** x_j, $j = 1, \ldots, k$ **do** /* last case : refine $R_l(x_j)$ */
14. . . .$R'_l(x_j) \longleftarrow \frac{R(\sim Z) - \sum_{i \neq j}(a_i R(x_i))}{a_j} \cap R_l(x_j)$;
15. . . .**if** $R'_l(\vec{x}) = \emptyset$ **then do**
16. . . . .delete $Rl(\vec{x})$ and D_l;
17. . . . .**go to** 3;
18. .**if** $R'_l(\vec{x}) \subset R_l(\vec{x})$ **then do** /* there is at least one refinement */
19. . .$R_l(\vec{x}) \longleftarrow R'_l$;
20. . .**go to** 5;
21. .**else do**
22. . .$flag \longleftarrow TRUE$;
23. **if** $flag = TRUE$ **then do**
24. .combine every $R_l(\vec{x})$ produced in 3 to $c.R(\vec{x})$ according to the assignation rule;
25. **return** $flag$;

end

Figure 1.10: The check_consistency algorithm

Input : c_t - current configuration, $t.action$ action sequence

Output : c_t configuration after the action

begin

1. **for each** action of $t.action$ **do**
2. .remplace the defined variables of the right member by their values;
3. **if** the action is such that $w \longleftarrow constante$ **then do**
4. .$c_t.R(w) \longleftarrow constante$;
5. .delete every statement letter containing w from $c_t.asrt(\vec{x})$;
6. **if** the action is such that $w \longleftarrow f(w, u, v, \ldots)$ **then do**
7. .compute $c_t.R(w)$;
8. .remplace every occurrences of w in $c_t.asrt(\vec{x})$ by $f^{-1}(w)$;
9. **if** the action is such that $w \longleftarrow f(u, v, \ldots)$ **then do** /* w not in right member */
10. .compute $c_t.R(w)$;
11. .delete every statement letters containing w from $c_t.asrt(\vec{x})$;
12. .add the statement letter $w = f(u, v, \ldots)$ in $c_t.asrt(\vec{x})$;
13. **return** c_t;

end

Figure 1.11: The processing of actions

1.6 The Invariant-based approach

The process of invariant-based passive testing will be described in the present section. The foundations of this technique can be found in [5] and improved in [1] where several modifications were realised.

1.6.1 Principle

The leading idea of the invariant-based testing is simple. The specification is studied in order to disclose particular properties. The properties that we look for are true at any moment and that is why they are called *invariants*. Once invariants were found we just check them on the implementation traces. We remark that this is a two-step process :

- the invariant research (on the specification),
- the application of the invariants on the implementation trace.

Unfortunately, the search for invariants is a gruelling task. When we attempt to model a temporal property with invariants we are confronted to an obstacle : the trace size should be long enough to encompass the whole invariant.

For instance, let consider the invariant that says 'when we have the input i_1 then in the future we will have the input i_n'. Let also consider the recorded trace ends with i_1/o_1. Then the invariant checking cannot be applied.

Indeed, it is possible to build a temporal logic based on a given input/output couple because they are carried by an EFSM transition and then depend on one another. We have also an other possibility : to consider a whole sequence of input/output couples as we are going to see it know. We define here the different kinds of input/output (or I/O) invariants.

1.6.2 The I/O invariants

An I/O invariant contains two parts :

- the **test**, which is an input or an output symbol,
- the **preambule**, which is a the sequence that must be first found in the trace before trying to compare the *test*

There exists three types of invariants answering these requirements :

- the **output invariants**, when the *test* is an output symbol. The property is of the form "immediatly after the sequence *preambule* we must always have the output *test*". Here are some examples of output invariants :

  - $\underbrace{(i_1}_{preambule} / \underbrace{o_1)}_{test}$: which means "i_1 is always followed by o_1".
  - $\underbrace{(i_1/o_1)(i_2}_{preambule} / \underbrace{o_2)}_{test}$: which means "immediatly after the sequence (i_1/o_1) and the input i_2 we always have o_2". This example is said to be an invariant of length 2 because its preambule sprawl on 2 I/O couples.

- the **input invariants**, when the *test* is an input symbol. The property is of the form "immediatly before the sequence *preambule* we must always have the input *test*". Here are some examples of input invariants :

  - $\underbrace{(i_1}_{test} / \underbrace{o_1)}_{preambule}$: which means "o_1 is always immediatly preceded by i_1".
  - $\underbrace{(i_1}_{test} / \underbrace{o_1)(i_2/o_2)}_{preambule}$: which means "immediatly before the sequence $o_1, (i_2/o_2)$ we always have the input i_1".

- the **succession invariants**, that materialize complex properties such as loop problems. Its drawback is that there is nowadays no know technique to automate the search of the kind of invariants. We present know an exemple of succession invariant use :

  - $\underbrace{(i_1/o_1)(i_2}_{preambule} / \underbrace{o_2)}_{test}$
  - $\underbrace{(i_1/o_1)(i_2/o_2)(i_2}_{preambule} / \underbrace{o_2)}_{test}$
  - $\underbrace{(i_1/o_1)(i_2/o_2)(i_2/o_2)(i_2}_{preambule} / \underbrace{o_3)}_{test}$
  - the set of the three invariants creates the succession invariant. This invariant force the transition holding (i_2/o_2) to be fired twice before the transition with (i_2/o_3) is obligatorily fired. This kind of sequence is use to allow few attempts for a given operation (in our example two attempts) in a protocol before it returns a failure (o_3 can represent this failure).

With this formalism it becomes possible to extract properties from the specification.

1.6.3 Conclusion

The invariants are a powerful tool based on the properties of the a specification but as we have already evoked it the research of such properties is not an easy task. If we delegate this task to a human it is likely to take a big amount of time. On the other hand, the existing algorithms like the ones described in [5] are sensitive to non determinism of the EFSM specification for extraction of invariant with a length strictly greater than one. In addition this method doesn't allow the detection of every errors and is destined to be used in complementarity of other methods and then acts a bit like an error filter.

1.7 Application exemple of the existing methods

The present section show the application of the previously described methods on a exemple protocol : the Simple Connection Protocol (SCP). The functioning of this protocol is first briefly introduced. Then we do the needed preparative work in order to apply the different methods, that is to say the invariant extraction. Finally, two traces respectively containing the two kinds of errors (output and transfert errors) are given as input to the presented algorithms to see weither they are able to detect the errors or not.

1.7.1 The SCP specification

SCP allows us to connect an entity called *upper layer* to an entity called *lower layer* (Fig 1.12).

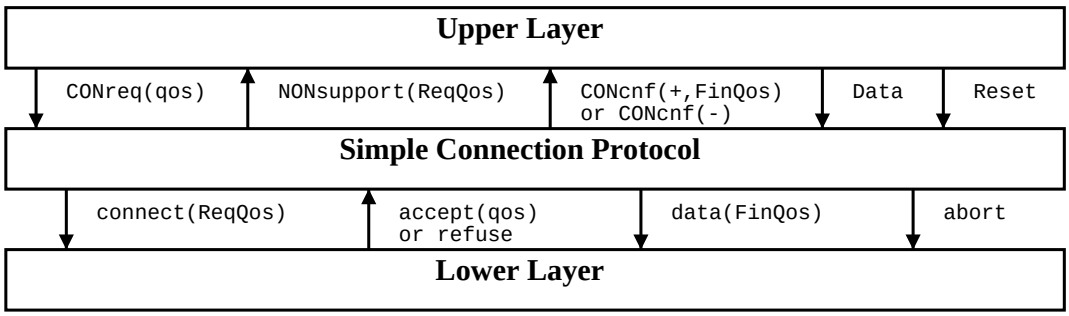

Figure 1.12: Layer view of the Simple Connection Protocol

The upper layer performs a dialogue with SCP to fix the quality of service desirable for the future connection. Once this negotiation finished, SCP dialogues with the lower layer to ask for the establishment of a connection satisfying the quality of service previously negotiated. The lower layer accepts or refuses this connection request. If it accepts the

connection, SCP informs the upper layer that connection was established and the upper layer can start to transit data towards the lower layer via SCP. Once the transmission of the data finished, the upper layer sends a message to close the connection. On the other hand, if the lower layer refuses the connection, the system allows SCP to make three requests before informing the upper layer that the connection attempts all failed. If the upper layer wishes again to be connected to the lower layer, it is necessary to restart the QoS negotiation with SCP from beginning. Every variable is defined in the interval $[0; 3]$. An EFSM specification of SCP is in the figure 1.13.

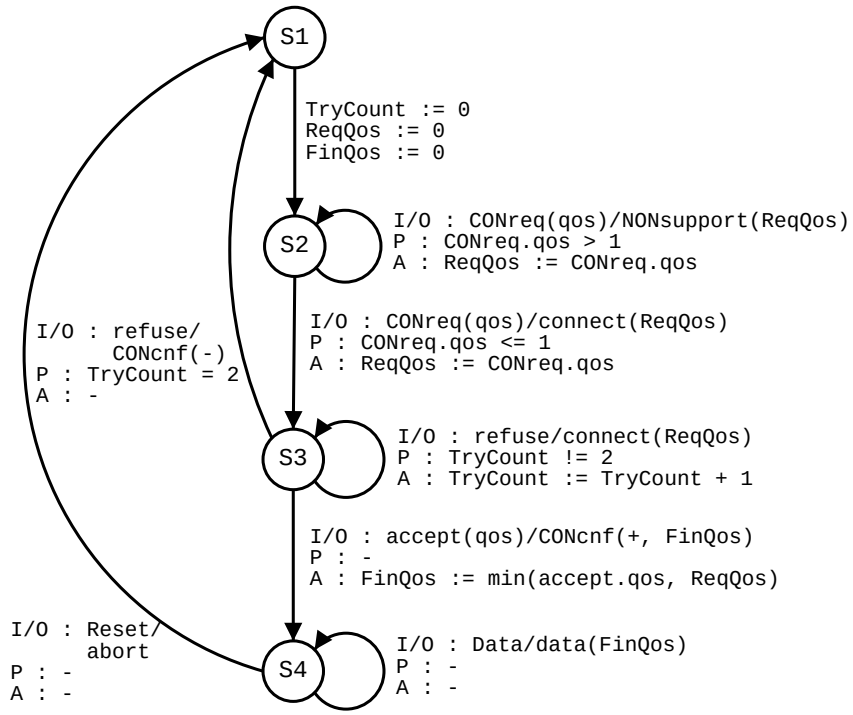

Figure 1.13: EFSM specification of the Simple Connection Protocol

1.7.2 Preparative work

Invariant extraction

Here are all the invariant with a length lower or equal to 2 that are possible to extract from the EFSM. In order to simplify the reading we write the *test* part of the invariants in bold. In addition, we adopt teh following nomenclature : IS, IE, ISuc respectively define output, input and succession invariants, whereas the two digits represent the length (for IS and IE) and the (arbitrary) number of the invariant.

IS-1-1 : accept(qos)/**CONcnf(+,FinQos)**

IS-1-2 : Data/**data(FinQos)**
IS-1-3 : Reset/**abort**
IS-2-1 : CONreq(qos)/connect(ReqQos), accept(qos)/**CONcnf(+,FinQos)**
IS-2-2 : refuse/connect(ReqQos), accept(ReqQos)/**CONcnf(+,FinQos)**
IS-2-3 : accept(qos)/CONcnf(+,FinQos), Data/**data(FinQos)**
IS-2-4 : accept(qos)/CONcnf(+,FinQos), Reset/**abort**
IS-2-5 : Data/data(FinQos), Data/**data(FinQos)**
IS-2-6 : Data/data(FinQos), Reset/**abort**

IE-1-1 : **CONreq(qos)**/NONsupport(ReqQos)
IE-1-2 : **refuse**/CONcnf(-)
IE-1-3 : **accept(qos)**/CONcnf(+,FinQos)
IE-1-4 : **Data**/data(FinQos)
IE-1-5 : **Reset**/abort
IE-2-1 : **CONreq(qos)**/NONsupport(ReqQos), CONreq(qos)/NONsupport(ReqQos)
IE-2-2 : **CONreq(qos)**/NONsupport(ReqQos), CONreq(qos)/connect(ReqQos)
IE-2-3 : **accept(qos)**/CONcnf(+,FinQos), Data/data(FinQos)
IE-2-4 : **Data/data(FinQos)**, Data/data(FinQos), Data/data(FinQos)
IE-2-5 : **accept(qos)**/CONcnf(+,FinQos), Reset/abort
IE-2-6 : **Data**/data(FinQos), Reset/abort

ISuc-1 : CONreq(qos)/connect(ReqQos), refuse/**connect(ReqQos)**
ISuc-2 : CONreq(qos)/connect(ReqQos), refuse/connect(ReqQos), refuse/**connect(ReqQos)**
ISuc-3 : CONreq(qos)/connect(ReqQos), refuse/connect(ReqQos), refuse/connect(ReqQos), refuse/**CONcnf(-)**

As explained previously, the succession invariant is used to model the three attempts to make an agreement for a connection preceding a possible failure and reinitialization of the protocol.

1.7.3 Output error

We present a trace containing an output error. For instance, this error can be realized by the modification of an action in order to produce a wrong variable update whose value will then be sent. The exemple trace can be seen as a trace comming from an implementation in which the action $ReqQos = CONreq.qos$ was (mistakably) replaced by $ReqQos = CONreq.qos - 1$ on the dotted transition of the figure 1.14. The corresponding output to the input $CONreq(3)$ is then $NONsupport(2)$ while $NONsupport(3)$ was expected.

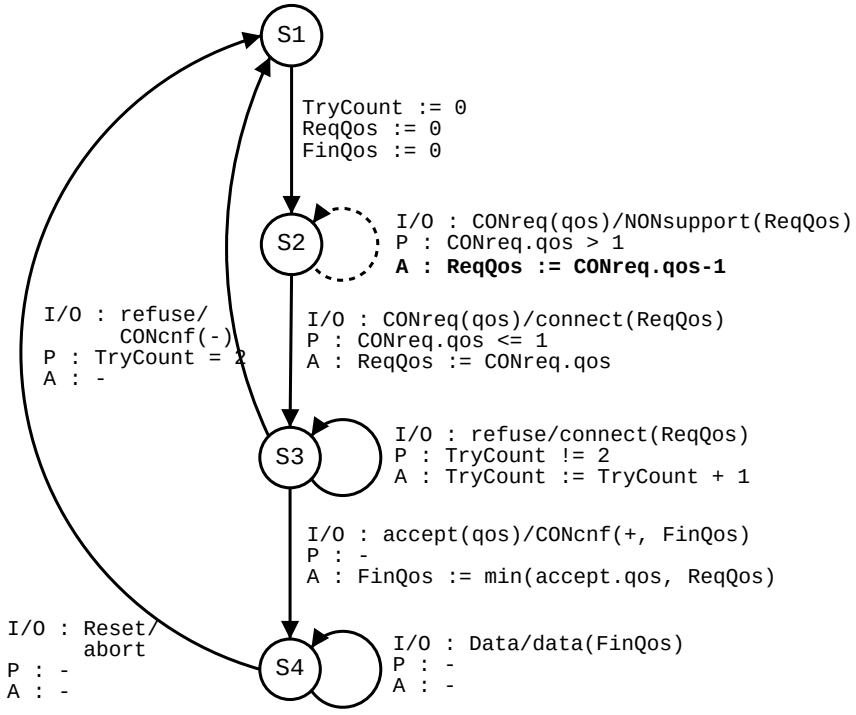

Figure 1.14: Implementation producing an output error

The following subsection present the study of the trace T_1 by the different methods :

$$T_1 = \begin{cases} CONreq(3)/NONsupport(2) \\ CONreq(1)/connect(1) \\ accept(2)/CONcnf(+,1) \\ Data/data(1) \end{cases}$$

First method : value determination

Let us see the process of the simplest algorithm as described in 1.4 on this trace. We note $T = \infty$ to say that T is undefined.

L=s_1, s_2, s_3, s_4, L'=$\emptyset$
fault_detected = $false$, **end_homing** = $false$
$\vec{x}$=(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)

- **k=1** : CONreq(3)/NONsupport(2)
  newX=$\emptyset$, L'=s_2

newX=[(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)]
L=s_2, L'=$\emptyset$
$\vec{x}$=(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)

- **k=2** : CONreq(1)/connect(1)
  newX=$\emptyset$, L'=s_3
  newX=[(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)]
  L=s_3, L'=$\emptyset$
  $\vec{x}$=(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)

- **k=3** : accept(2)/CONcnf(+,1)
  newX=$\emptyset$, L'=s_4
  newX=[(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)]
  L=s_4, L'=$\emptyset$
  $\vec{x}$=(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)

- **k=4** : Data/data(1)
  newX=$\emptyset$, L'=s_4
  newX=[(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)]
  L=s_4, L'=$\emptyset$
  $\vec{x}$=(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)

The algorithm fail to detect the output error because it was not able to associate a value to each variable. Indeed, the $\leq$ and ¿ operators of specification don't allow it.

2nd method : interval determination

The processing of the [14] algorithm is presented in the table 1.15.

step	event	configurations
0	-	$(S_1; < TC = [0;3]$, $RQ = [0;3]$, $FQ = [0;3]$, $Crq.qos = [0;3]$, $acc.qos = [0;3] >, \emptyset)$ $(S_2; < TC = [0;3]$, $RQ = [0;3]$, $FQ = [0;3]$, $Crq.qos = [0;3]$, $acc.qos = [0;3] >, \emptyset)$ $(S_3; < TC = [0;3]$, $RQ = [0;3]$, $FQ = [0;3]$, $Crq.qos = [0;3]$, $acc.qos = [0;3] >, \emptyset)$ $(S_4; < TC = [0;3]$, $RQ = [0;3]$, $FQ = [0;3]$, $Crq.qos = [0;3]$, $acc.qos = [0;3] >, \emptyset)$
1	CONreq(3) / NON-support(2)	$(S_2; < TC = [0;3]$, $RQ = 2$, $FQ = [0;3]$, $Crq.qos = 3$, $acc.qos = [0;3] >, \emptyset)$ **ERROR** RQ≠Crq.qos

Figure 1.15: Application of the interval determination algorithm for the output fault

We observe that the only possible transition for the I/O couple $CONreq(qos)/\ NONsupport(ReqQos$ leads the algorithm to detect the output fault.

3rd method : invariants

We bring together the invariants that are applicable on the trace in the table 1.16 where tests parts appear in bold.

Trace	CONreq(3)	NONsupport(2)	CONreq(1)	connect(1)	accept(2)	CONcnf(+,1)	Data	data(1)
IE-1-1	**CONreq(qos)**	NONsupport(RQ)						
IE-2-2	**CONreq(qos)**	NONsupport(RQ)	CONreq(qos)	connect(RQ)				
IS-2-1			CONreq(qos)	connect(RQ)	accept(qos)	**CONcnf(+,FQ)**		
IS-1-1					accept(qos)	**CONcnf(+,FQ)**		
IS-2-3					accept(qos)	CONcnf(+,FQ)	Data	**data(FQ)**
IE-1-3					**accept(qos)**	CONcnf(+,FQ)		
IE-2-3					**accept(qos)**	CONcnf(+,FQ)	Data	data(FQ)
IS-1-2							Data	**data(FQ)**
IE-1-4							**Data**	data(FQ)

Figure 1.16: Application of the invariant-based approach for the output fault

No error was discovered by the algorithm but we must also check that the variables respect their definition domains :

- CONreq.qos=3 then 1 → OK because CONreq.qos∈[0,3]
- ReqQos=2 then 1 → OK because ReqQos∈[0,3]
- accept.qos=2 → OK because accept.qos∈[0,3]
- FinQos=1 → OK because FinQos∈[0,3]

No domain inconsistency was found so the verdict of this algorithm for the current trace says there is no error.

1.7.4 Transfert error

On the figure 1.17 an other implementation modification was performed (doted transition). This modification leads to a transfert error. The algorithms are run with a faulty trace from this implementation.

The studied trace is :

$$T_1 = \begin{cases} CONreq(1)/connect(1) \\ refuse/CONcnf(-) \end{cases}$$

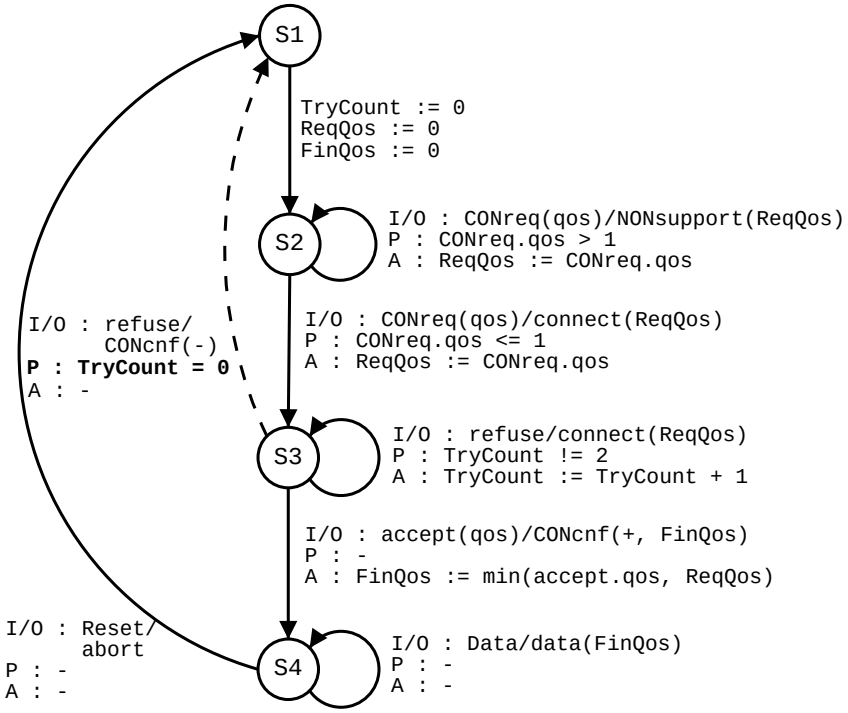

Figure 1.17: Implementation producing a transfert error

1st method : value determination

The algorithm gives us :

L=s_1, s_2, s_3, s_4, L'=$\emptyset$
fault_detected=false, **end_homing**=false
$\vec{x}$=(TryCount=∞,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)

- **k=1** : CONreq(1)/connect(1)
  newX=$\emptyset$, L'=s_3
  newX=[(TryCount=∞,ReqQos=1,FinQos=∞,CONreq.qos=1,accept.qos=∞)]
  L=s_3, L'=$\emptyset$
  $\vec{x}$=(TryCount=∞,ReqQos=1,FinQos=∞,CONreq.qos=1,accept.qos=∞)

- **k=2** : refuse/CONcnf(-)
  newX=$\emptyset$, L'=s_1
  newX=[(TryCount=2,ReqQos=∞,FinQos=∞,CONreq.qos=∞,accept.qos=∞)]
  L=s_1, L'=$\emptyset$

$\vec{x}$=(TryCount=2,ReqQos=1,FinQos=∞,CONreq.qos=1,accept.qos=∞)

This method fails in this case too.

2nd method : interval determination

The processing of the [14] algorithm is presented in the following table.

step	event	configurations
0	-	$(S_1; < TC = [0;3],\ RQ = [0;3],\ FQ = [0;3],\ Crq.qos = [0;3],\ acc.qos = [0;3] >, \emptyset)$ $(S_2; < TC = [0;3],\ RQ = [0;3],\ FQ = [0;3],\ Crq.qos = [0;3],\ acc.qos = [0;3] >, \emptyset)$ $(S_3; < TC = [0;3],\ RQ = [0;3],\ FQ = [0;3],\ Crq.qos = [0;3],\ acc.qos = [0;3] >, \emptyset)$ $(S_4; < TC = [0;3],\ RQ = [0;3],\ FQ = [0;3],\ Crq.qos = [0;3],\ acc.qos = [0;3] >, \emptyset)$
1	CONreq(1) / connect(1)	$(S_3; < TC = [0;3],\ RQ = 1,\ FQ = [0;3],\ Crq.qos = 1,\ acc.qos = [0;3] >, \emptyset)$
2	refuse / CONcnf(-)	$(S_1; < TC = 2,\ RQ = 1,\ FQ = [0;3],\ Crq.qos = 1,\ acc.qos = [0;3] >, \emptyset)$

Figure 1.18: Application of the interval determination algorithm for the transfert fault

In this trace, the fault stays undetected.

3rd method : invariants

We bring together the invariants that are applicable on the trace in the following table (1.19) where tests parts appear in bold.

Trace	CONreq(1)	connect(1)	refuse	CONcnf(-)
ISuc-1	CONreq(qos)	connect(ReqQos)	refuse	**connect(ReqQos)**
IE-1-2			**refuse**	CONcnf(-)

Figure 1.19: Application of the invariant-based approach for the transfert fault

We can stop right now raising a faulty behaviour of the implementation as shown by the inconsistency on the test part of **ISuc-1**. The transfert error is detected with help of

the succession invariant.

1.7.5 Summary of the main points

In order to see clearly the advantage of each methods we summarize the results of this application exemple in the hereinafter table (1.20)

	value determination	intervals determination	invariants
Output error	NO	YES	NO
Tranfert error	NO	NO	YES

Figure 1.20: Summary of the detection power of the various algorithms

Conclusion

As a conclusion for this chapter about the different existing passive testing techniques, we underline that no one of these is able to give correct answers , i.e. to detect every fault as shown with the application exemple. The value determination technique seems anyway totally outdated since it failed to find mistakes on the both traces we proposed for the study. It can also be seen as a particular case of the much more powerful interval determination technique.

We can propose two main possible research ways to explore in order to improve the passive testing domain.

The invariant approach seems interesting because of its abilities to focus on particular aspects of the specifiication and to give a quick answer. But firstly, the automatization of the invariant extraction is not easy - sometimes even not possible - and this step often needs the intervention of a expert (human) of the studied protocol or service. Secondly, this technique only studies the I/O events without consideration to the variables and then there is no fault detection on the variables themself. And last, if we work with traces that are shorter than the invariants then the invariants can't be applied and no fault can be detected.

The interval determination approach need less efforts than the invariant approach but is also more ressource/time consuming. It considers the variable values but as the invariant approach suffers from the size of the traces (which can be too short).

Improvements of these two approaches are presented in the following chapters as the scientific contribution of this thesis. The work is mainly focused on the interval determination but will also propose new concepts for the invariant approach. In particular this thesis will tries to resolve the trace length problem and then offers the true-false dual verdict where only the possible-false verdict could be given.

Chapter 2

Improvement of the invariant approach - A constrained invariant checking approach

2.1 Introduction

Most of passive testing methods are focused on the control parts of the system under test without considering the data parts, so it is sufficient for them to use finite state machines (FSMs) as the specification method. To take the data part of the protocols into account, extended finite state machines (EFSMs) are used to specify the system. EFSM uses parameterized input/output signals including variable parameters to encode data as well as predicates and actions to control the firing of the transitions by manipulating the relevant data. There are some methods proposed to perform passive testing using EFSM ([39], [14], [38]). These methods are based on exploring constraints of the variables and comparing the whole specification with the implementation regarding the constraints in a backward or forward manner (i.e. the naive approach).

In this chapter a method to perform passive testing based on the invariants on the EFSM is proposed. In our approach, first we have to extract the invariants intuitively from the protocol specification requirements regarding only the control portion of the protocols. After that, to take the data portion into account, we consider the invariant parameters as some variables.

We present two algorithms for finding the corresponding constraints over the variables of the invariants automatically. The algorithms use the unification method [2] for checking the correctness of the given invariants over the EFSM and finding the constraints over its variables. Having the invariants and their corresponding constraints in hand, we can check the execution trace with the invariants using pattern matching methods.

It should be noted that finding suitable control-driven invariants, especially with the help of an expert is relatively simple; also there are some methods to extract a limited set of invariants from an EFSM automatically [4]. We use the notion of invariant introduced in

[3] with little changes in the definitions to extract the control driven invariants intuitively.

The rest of the chapter is organized as follows : some preliminary concepts needed in the rest of the chapter are described then the notion of forward and backward invariants are described. Afterwards, our algorithms for checking invariants on a given EFSM and extracting corresponding constraints are presented and finally we report some experiments of the algorithms on the Simple Connection Protocol (SCP) to show the applicability of the method in detecting subtle errors in some given traces of the protocol.

2.2 Preliminaries

In this technique we use EFSM specification models, concepts that were previously defined but also some notions coming from the logic programming community.

2.2.1 Substitution and unification

Our algorithms use the unification method, so we borrow some definitions from the context of logic programming.

Definition 7 *A substitution θ is a set of bindings, each of the form V/T, such that V is a distinct variable and T is a term. θ is called a renaming if it map each variable to a new fresh variable.*

Applying a binding V/T to an expression E, replaces each free occurrence of V in E by T. Applying a substitution θ on an expression E denoted by Eθ applies all the bindings in θ to E simultaneously and independently.

Definition 8 *Let $\theta = \{V_1/T_1, \ldots, V_m/T_m\}$ and $\alpha = \{U_1/S_1, \ldots, U_n/S_n\}$ be substitutions. The composition of θ and α, denoted by $\theta \circ \alpha$ is defined as: $\theta \circ \alpha = \{V_1/T_1\alpha, \ldots, V_m/T_m\alpha\} \cup \{U_k/S_k \text{ — } U_k \notin \{V_1, \ldots, V_m\}\}$*

Definition 9 *A unifier of two simple expressions E and F is a substitution θ such that $E\theta = F\theta$. If two simple expressions have a unifier, they are said to be unifiable; we also say that E is unified with F by the unifier θ. A most general unifier, abbreviated as mgu, of two simple expressions E and F is a unifier θ that is more general than any unifier of E and F.*

As an example two expressions $P(X, f(X))$ and $P(b, f(a))$ are not unifiable, while the most general unifier of the expressions $P(X, f(a))$ and $P(b, f(Y))$ is $\{X/b, Y/a\}$.

2.2.2 Normalizing action sequences

We need in our algorithms to track the changes in the variable values made by the action sequences in each transition, so we define a special normal action sequence and present an algorithm to normalize a given action sequence. In normalizing an action sequence of

a transition, a special renaming substitution is produced which we name the *normalizer substitution* of that action. The normalizer substitution is then used to propagate the changes of the variable values in a transition to predicates and actions of the successive transitions.

Definition 10 *Let x be the set of variables in an EFSM, also let $A = (l_1 := r_1, \ldots l_n := r_n)$ be an action sequence of size n in a transition in the EFSM, in which $l_i \in x$ and r_i is an expression for $l \leq i \leq n$. Also suppose that $R_k = \{V \in x$ —V is used in expressions $r_1, r_2, \ldots, r_k\}$ for $l \leq k \leq n$. A is a normal action sequence if $l_j \notin R_j$ for $l \leq j \leq n$.*

The algorithm depicted in figure 2.1 change a given action sequence to a normalized one and returns its corresponding normalizer substitution. The algorithm renames the new appearances of the variables whose values are changed in an action sequence. The normalizer substitution of the action is in fact the set of variable renaming substitutions performed in the above process.

Input : An action sequence $A = (l_1 := r_1, \ldots, l_n := r_n)$ in the EFSM $M = (S, I, O, x, Tr, s_{in})$

Output : A normalizer substitution θ

Side effect : Action sequence A is changed to a normalized one

begin

1. $\theta := \emptyset$;
2. **for** $i := 1$ to n **do**
3. $\quad .R := \{V \in x | V$ is used in expressions $r_1, \ldots, r_i\}$
4. $\quad .$**if** $l_i \in R$ **then do**
5. $\quad . \quad .l'_i := newV$; /*a new variable name*/
6. $\quad . \quad .\theta := \theta \cup \{l_i / l'_i\}$;
7. $\quad . \quad .l_i := l'_i$
8. $\quad . \quad .$**for** $j := i + 1$ to n **do**
9. $\quad . \quad . \quad .l_j := l_j \theta$;
10. $\quad . \quad . \quad .r_j := r_j \theta$;

end

Figure 2.1: Normalizing an action sequence

2.2.3 The notion of invariant

In this section we represent the notion of invariants on EFSM as introduced in [11] with little changes in the definitions. An invariant represents a specific property (which should

be always true) on an EFSM which is in fact a statement about causal relationships between input/output pairs in the EFSM. Regarding the way of expressing the temporal relationships of the input/outputs in an EFSM, two types of invariants are introduced. We call them forward and backward invariants. Note that to define the invariants we only consider the control parts of the protocol so we do not speak about the values of the variables in input or output parameters. In the next section we represent algorithms to find corresponding constraints on the variables of a given invariant that makes it correct on the EFSM.

Forward invariants

A forward invariant is used to express properties in the EFSM such as "each time the implementation performs a specific execution trace like $i_1/o_1, \ldots, i_{n-1}/o_{n-1}, in$, the next observed output belongs to a specific set of output symbols". Based on this definition, we can assume that a forward invariant contains three elements: A preamble I/O sequence, a preamble input and a test output set. Intuitively a forward invariant is correct if for all paths in the EFSM matching with the preamble I/O sequence, and followed by an I/O pair containing an input equal to the preamble input, then the corresponding output essentially belongs to the test output set.

Definition 11 *Let $M = (S, I, O, x, Tr, s_{in})$ be an EFSM. We say that the $F(PIO, PI, TOS)$ is a forward invariant for M if the following conditions are respected :*

- *PIO is the preamble I/O sequence which is defined according to the following EBNF : $PIO ::= a/z, PIO | *, PIO | \epsilon$ in which $a \in I \cup \{?\}$, $z \in O \cup \{?\}$ and ϵ is the null sequence*
- *$PI \in I$ is the preamble input and $TOS \subseteq O$ is the test output set*
- *Each time that the sequence PIO is matched with any path in the EFSM, and it is followed by any transition with input PI, then we get essentially an output belonging to TOS*

Note that we deal with the wildcard ? as the standard one in pattern matching, while modify the usual meaning of the symbol $*$. The symbol $*$ replaces any sequence of input/outputs not containing any pair with input equal to PI.

Backward invariants

Using a backward invariant we can express more subtle properties such as "each time a specific output is produced by the implementation, then we must have that a specific trace had been produced before". So a backward invariant contains three elements : a preamble output set, a test input and a test I/O sequence. Intuitively a backward invariant is correct if any transition in the EFSM in which its output symbol belongs to the preamble output set, have an input equal to test input and essentially preceded by a path matching with the test I/O sequence.

Definition 12 *Let $M = (S, I, O, x, Tr, s_{in})$ be an EFSM. We say that the $B(TIO, TI, POS)$ is a backward invariant for M if the following conditions are respected :*

- *TIO is the test I/O sequence which is defined according to the following EBNF : $TIO ::= a/z, TIO|*, TIO|\epsilon$ in which $a \in I \cup \{?\}$, $z \in O \cup \{?\}$ and ϵ is the null sequence*
- *$TI \in I \cup \{?\}$ is the test input and $POS \subseteq O$ is the test output set*
- *All transitions of M with an output symbol belonging to POS must essentially have an input symbol equal to TI and proceed by a path matching with TIO*

Let us remark that, in contrast with forward invariants (for the case of preamble input symbol), we do not force the test input symbol here to be an input action (it can be also the wild-card character "?"). Furthermore, our matching method is modified such that the symbol $*$ replaces any sequence of input/outputs not containing any pair with input equal to TI.

2.3 Extracting invariant constraints

To use an invariant for passive testing, it is needed to assure at first about the correctness of the invariant on the specification. While checking invariants on a FSM simply returns a Boolean value showing the correctness or fail of the invariant, checking an invariant on an EFSM either returns simply a false Boolean value showing that the invariant is incorrect on the EFSM or returns a set of constraints on the variables of the invariant showing that the correctness of the invariant depends on the set of constraints. For passively testing the implementation, it is sufficient to match the execution trace with the invariant while its constraints regarding the value of the variables in the trace don't conflict. In this section we represent algorithms for checking forward and backward invariants on an EFSM and extracting their corresponding constraint set. The constraint extraction process is done once and off-line.

2.3.1 Forward invariant constraints

To check a given forward invariant on an EFSM, first we have to find the paths in the EFSM which are unifiable with the preamble part of the invariant. After that we should check if the invariant test set is reachable using all the unified paths or not, and if it is reachable, then what is the constraint set to make it true. The constraint set is in fact constructed during the unification of the preamble part with the paths in the EFSM.

Definition 13 *Let $\rho = i_1/o_1, \ldots, i_n/o_n$ be an input sequence of size n and $M = (S, I, O, x, Tr, s_{in})$ be an EFSM, we define U_n as the set of forward matchers of ρ containing quadruples (s, θ, C, δ) in which s is a state belongs to S, θ and δ are substitutions and C is a set of constraints (conjoined predicates) which is constructed inductively as follows :*

. *The initial forward matcher set is equal to* $U_0 = S \times \{\emptyset\} \times \{\emptyset\} \times \{\emptyset\}$

. *If* $t = (s, s', a, z, P, A) \in Tr$ *is a transition in* M*, and* U_{j-1} *contains a forward matcher quadruple* (s, θ, C, δ) *such that* $(a/z)\delta$ *is unifiable with* (i/o)*, then* U_j *contains quadruples* $(s', \theta', C', \delta')$ *in which* $\theta' = \theta \circ mgu((a/z)\delta, (i/o))$ *,* $C' = C \cup (P \wedge normalized(A))\delta$*, and* δ *is the normalizer of* A

Using the above definition we can describe our algorithm for checking the correctness of a given forward invariant on an EFSM and extracting its necessary constraints. Let $F(PIO, PI, TOS)$ be a forward invariant in which PIO is of size n. Suppose that U_n is the forward matcher set of PIO. If U_n is empty then the invariant is incorrect, else we check that for any transition labeled by the input PI, we receive an output unifiable with one of the items in the TOS. If there is no possible transitions, then the invariant is incorrect, else for each forward matcher quadruple (s, θ, C, δ) in U_n, if C is empty then the invariant is true, else the invariant is true constraint to $C\theta$. The set of constraints $C\theta$ can be simplified using the existing constraint simplification algorithms. The algorithm is depicted more formally and detailed in figure 2.2.

The algorithm deals with invariants containing the wildcard character $*$. Also we consider that both $i =?$ and $o =?$ hold. We have used some auxiliary functions : $head(I)$ returns the first i/o couple of the sequence I and $tail(I)$ removes the first i/o couple from I. the Boolean function $path(s, s', i)$ returns true if there exist a path $a_1/z_1, \ldots, a_r/z_r$ from s to s' and for any $1 \leq j \leq r$ we have $a_j \neq i$. Also the function $simplified(C)$ returns the simplified version of the constraint set C. In fact this function solves the constraints such that the most constraining predicates on a single variable are remained. We don't enter in the details of this function. There are some well known methods to do this in the literature [22].

2.3.2 Backward invariant constraints

To check a given backward invariant on an EFSM, first we have to find the set of transitions in the EFSM which have outputs unifiable with elements of the preamble output set in the invariant. After that, we should check whether the paths in the EFSM which are ended by the discussed outputs are unifiable with the test input and test I/O sequence in the invariant or not. And, if they are unifiable, what are the constraints on the variables of the invariant. The constraint set is in fact constructed during the unification of the test I/O sequence with the paths in the EFSM. We traverse the paths in the EFSM in a backward fashion to do the unification and extract the constraints.

Definition 14 *Let* $\rho = i_1/o_1, \ldots, i_n/o_n$ *be an input sequence of size* n *and* $M = (S, I, O, x, Tr, s_{in})$ *be an EFSM, we define* V_0 *as the set of backward matchers of* ρ *containing quadruples* (s, θ, C, δ) *in which* s *is a state belongs to* S*,* θ *and* δ *are substitutions and* C *is a set of constraints (conjoined predicates) which is constructed inductively as follows:*

. *The initial backward matcher set is equal to* $V_n = S \times \{\emptyset\} \times \{\emptyset\} \times \{\emptyset\}$

Input : $M = (S, I, O, x, Tr, s_{in})$, $I = F(PIO, PI, TOS)$

Output : $true$, $false$, or a set of constraints. Satisfaction of each constraint is sufficient ta satisfy the invariant.

begin

/* PIO matching : finding the paths in the EFSM which are unifiable with the PIO */

1 $I' := PIO$;
2 $U := S \times \{\emptyset\} \times \{\emptyset\} \times \{\emptyset\}$;
3 **while** $I' = \epsilon$ **and** $U \neq \emptyset$ *do*
4 .$first := head(I')$;
5 .$I' := tail(I')$;
6 .**if** $first \neq *$ **then do** /* first=i/o */
7 . .$T := Tr$;
8 . .$U' := \emptyset$;
9 . .**while** $T \neq \emptyset$ **do**
10 . . .$chooset \in T$; /* t = (s, s', a, z, P, A) */
11 . . .$T := T - \{t\}$;
12 . . .**if** $(s, \theta, C, \delta) \in U$ **and** $unifiable((a/z)\delta, i/o)$ **then do**
13 . . . .$\theta' := \theta_0 mgu((a/z)\delta, i/o)$;
14 . . . .$C' := C \cup (P \wedge normalized(A))\delta$;
15 . . . .$\delta' := normalizer(A)$;
16 . . . .$U' := U' \cup \{(s', \theta', C', \delta')\}$;
17 . .$U := U'$;
18 .**else do** /* first=* */
19 . .**while** $head(I') = *$ **do**
20 . . .$I' := tail(I')$;
21 . . .$first := head(I')$ /* first=i/o */
22 . . .$U := \{(s, \theta, C, \delta) | s \in S, \exists (s, \theta, C, \delta) \in U, p = path(s', s, i)\}$;

/* TOS checking : checking if TOS is reachable using the unified path or not and if so what is its constraints */

23 **if** $U = \emptyset$ then do
24 .**return** $false$;
25 **else do**
26 .$t_f := false$;
27 .$T := Tr$
28 .$CS := \emptyset$
29 .**while** $T \neq \emptyset$ **do**
30 . .$chooset \in T$;
31 . .$T := T - \{t\}$; /* t = (s, s', a, z, P, A) */
32 . .**if** $(s, \theta, C, \delta) \in U$ **and** $unifiable(a\delta, i_n)$ **then do**
33 . . .$\theta' := \theta_0 mgu(a\delta, i_n)$;
34 . . .**if** $o \in O \bullet (z\delta)\theta' = o\theta'$ **then do**
35 . . . .**return** $false$;
36 . . . .$t_f := true$;
37 . . . .$CS := CS \cup simplified(C \cup (P \wedge normalized(A))\delta)$;
38 **if** $t_f = false$ **then do**
39 .**return** $false$;
40 **if** $CS = \emptyset$ **then do**
41 .**return** $true$;
42 **else do**
43 .**return** CS;

end

Figure 2.2: Algorithm for checking forward invariants and finding its corresponding constraints

. *If* $t = (s, s', a, z, P, A) \in Tr$ *is a transition in* M*, and* V_{j+1} *contains a backward matcher quadruple* $q = (s', \theta, C, \delta)$ *such that* $(a/z)\delta$ *is unifiable with* (i_j/o_j)*, then* V_j *contains quadruples* $q' = (s, \theta', C', \delta')$ *in which* $\theta' = \theta \circ mgu((a/z)\delta, (i_j/o_j))$*,* δ' *is the normalizer of* A *and* $C' = C\delta' \cup (P \wedge normalized(A))$

. *Delete the quadruple* q *from* V_{j+1}.

Using the above definition we can describe our algorithm for checking the correctness of a given backward invariant on an EFSM and extracting its necessary constraints. Let $B(TIO, TI, POS)$ be a backward invariant. For all elements $o_m \in POS$ $(1 \leq m \leq |POS|$ in which $|POS|$ is the cardinality of $POS)$ we concatenate the pair TI/o_m to the TIO to generate a set of input/output sequences like $\rho_m =$ "$TIO, TI/o_m$". Let the size of ρ_m be n. For each $\rho_m (1 \leq m \leq |POS|)$ we try to find its backward matcher set. If after constructing $V_j (0 < j \leq n)$ in each iteration, V_j is empty or V_{j+1} is not empty, then the invariant is incorrect, else for each quadruple of the V_0, if C is empty, then the invariant is true without any condition, else the invariant is true constraint to simplified C_θ. The algorithm is depicted more formally and detailed in figure 2.3. Auxiliary functions used are the same as described for the forward invariant checking algorithm.

2.4 An example on SCP protocol

The Simple Connection Protocol was already presented in this work in 1.7.1. Please refer to this section for the description of the protocol.

2.4.1 Defining invariants

Let consider the EFSM specification of the Simple Connection Protocol depicted in figure 1.13 of section 1.7.1. We suppose that the values of $TryCount$, $ReqQos$, $FinQos$, $CONreq.qos$, and $accept.qos$ are defined in the interval $[0; 3]$. Suppose that we want to passively test an implementation of the SCP regarding the following properties of the specification which are described using the invariants:

- $I1 = B(< refuse/connect(x) >, (refuse), \{CONcnf(-)\})$, means that SCP fail to connect the two layers $(CONcnf(-))$ only if the lower layer refused the connection twice before $(refuse/connect(x), refuse/)$.

- $I2 = F(< CONreq(x)/connect(y) >, (accept(w)), \{CONcnf(+, z)\})$, means that if SCP accept to connect with the upper layer at his requested QoS $(CONreq(x)/$ $connect(y))$ and the lower layer accept it at a given QoS, then a connection must be realized between the two layers.

- $I3 = F(<>, (accept(x)), \{CONcnf(+, y)\})$, means that if the lower layer accept the connection $(accept(x))$, this connection must be realized $(CONcnf(+, y))$.

Input : $M = (S, I, O, x, Tr, s_{in})$, $I = B(TIO, TI, POS)$

Output : $true$, $false$, or a set of constraints. Satisfaction of each constraint is sufficient ta satisfy the invariant.

begin

/* PIO matching : finding the states in the EFSM which are unifiable with elements of the POS */

1 $T := Tr$;
2 $V := S \times \{\emptyset\} \times \{\emptyset\} \times \{\emptyset\}$;
3 $error := false$;
4 **while** $T \neq \emptyset$ **and** $error = false$ *do*
5 .$choose t \in T$; /* t = (s, s', a, z, P, A) */
6 .$T := T - \{t\}$;
7 .**if** $\exists o \in O \bullet unifiable((a/z)\delta), (TI/o))$ **then do**
8 . .$\theta := mgu(a/z, PI/o)$;
9 . .$\delta := normalizer(A)$;
10 . .$C = P \wedge normalized(A)$;
11 . .$V := V \cup \{(s, \theta, C, \delta)\}$;
12 .**else do** $error := true$;
13 **if** $V = \emptyset$ **then do** $error := true$

/* TIO matching : checking if the TIO is matched with all the paths ending to the states found in the previous step or not and if so wath is the constraint set */

14 $I' := reverse(TIO)$;
15 **while not** $empty(I')$ **and** $error = false$ **do**
16 .$V' := \emptyset$;
17 .$first := head(I')$;
18 .$I' := tail(I')$;
19 .**if** $first \neq *$ **then do** /* first=i/o */
20 . .$T := Tr$;
21 . .**while** $T \neq \emptyset$ **do**
22 . . .$choose t \in T$; /* t = (s, s', a, z, P, A) */
23 . . .$T := T - \{t\}$;
24 . . .**if** $(s', \theta, C, \delta) \in V$ **and** $unifiable((a/z)\delta, i/o)$ **then do**
25 . . . .$\theta' := \theta_0 mgu((a/z)\delta, i/o)$;
26 . . . .$C' := C\delta' \cup (P \wedge normalized(A))\delta$;
27 . . . .$\delta' := normalizer(A)$;
28 . . . .$V' := V' \cup \{(s, \theta', C', \delta')\}$;
29 . . . .$V = V - \{(s', \theta, C, \delta)\}$;
30 . . .**else do**
31 . . . .$error := true$;
32 .**else do** /* first=* */
33 . .**while** $head(I') = *$; **do**
34 . . .$I' = tail(I')$; /* skip a sequence of "*" */
35 . . .$first = head(I')$; /* first=i/o */
36 . . .$V' = \{(s, \theta, C, \delta) | s \in S, \forall (s', \theta, C, \delta) \in V \bullet path(s, s', o)\}$;
37 .**if** $V \neq \emptyset$ **then do** $error := true$;
38 .**else do** $V := V'$;
39 **if** $error = true$ **then do return** $false$
40 $CS = \emptyset$;
41 **while** $V \neq \emptyset$ **do**
42 .$choose v \in V$; /* v=(s, θ, C, δ) */
43 .$V = V - \{v\}$;
44 .$CS = CS \cup simplify(C)$;
45 **if** $CS = \emptyset$**then do return** $true$;
46 **else do return** CS;

end

Figure 2.3: Algorithm for checking backward invariants and finding its corresponding constraints

- $I4 = B(<>, (accept(x)), \{CONcnf(+, y)\})$, means that a connection is realized ($CONcnf(+, x)$), only if the lower layer accepted it before ($accept(y)$).

Note that $I1$ and $I4$ are forward invariants while $I2$ and $I3$ are backward invariants. In definition of the above invariants we have used a control driven approach i.e. in this stage, parameters of the signals are not important so we have used some variables instead of them.

2.4.2 Finding invariant constraints

Now, we apply our method on the invariants to find their corresponding constraints. Table 2.4 shows the trace of the algorithms. For each forward (backward) invariant, the value of the intermediate forward (backward) matcher set i.e. U (i.e. V) and the ultimate constraint sets CS over the variables of the invariants have been shown. (See the algorithms in figures 2 and 3). Applying the algorithms reveal that all the invariants are correct regarding the control part of the specification, but regarding the data part of the specification the invariants are true condition to some constraints which have been produced by the algorithms. For invariant I1, there is not any constraint over the variable of the invariant, so it should be matched by execution traces with any value for the variable x. For the other invariants, only such execution traces are matched with the invariants that the value of their input/output parameters does not causes any conflict with the corresponding constraints of the intended invariant.

2.4.3 Passive testing using the constrained invariants

Now suppose that the following execution traces are generated by a faulty implementation of the SCP :

- $Trace_1 = CONreq(1)/connect(1), refuse/CONcnf(-)$
- $Trace_2 = CONreq(1)/connect(0), accept(1)/CONcnf(+, 0)$

We know that a transition error has been occurred in the first trace because the specification forces two loops on state s_3 before eventual transition to s_1, corresponding to the three requests SCP must do before failing the connection. In this trace, the connection is said to be failed on first try.

For the second trace, there is an output error because the first I/O couple should be $CONreq(1)/connect(1)$. We can imagine that the trace comes from an implementation in which the action on the transition from s_2 to s_3 is $ReqQos := CONreq.qos - 1$ and then such a trace is produced. This error has for consequence to connect the upper and lower layers with a QoS equals to 0 when it could be (normally) equal to 1.

Tables 2.5 and 2.6 show the invariants used in the checking of the first and the second trace respectively.

We try to identify the constraints with the values of the variables extracted from the traces:

I_1 (Backward)	
V_1	$V_1 = \{(s_3, \theta_1, C_1, \delta_1)\}$ $\theta_1 = \emptyset$ $C1 = \{TryCount = 2\}$ $\delta_1 = \emptyset$
V_2	$V_2 = \{(s_3, \theta_2, C_2, \delta_2)\}$ $\theta_2 = \{ReqQos/x\}$ $C_2 = C_1\delta_2\ cup\{TryCount \neq 2, y = TryCount + 1\}$ $= \{y = 2, TryCount \neq 2, y = TryCount + 1\} = \{TryCount = 1\}$ $\delta_2 = \{TryCount/y\}$
Constraint set	$CS\_I_1 = C_2\theta_2 = \{TryCount = 2\}$

I_2 (Forward)	
U_1	$U_1 = \{(s_3, \theta_1, C_1, \delta_1)\}$ $\theta_1 = \{CONreq.qos/x, ReqQos/y\}$ $C_1 = \{CONreq.qos \leq 1, ReqQos = CONreq.qos\}$ $\delta_1 = \emptyset$
U_2	$U_2 = \{(s_4, \theta_2, C_2, \delta_2)\}$ $\theta_2 = \theta_1 \circ \{accept.qos/w, FinQos/z\}$ $= \{CONreq.qos/x, ReqQos/y, accept.qos/w, FinQos/z\}$ $C_2 = C_1 \cup \{CONreq.qos = min(accept.qos, ReqQos)\}\delta_1$ $= \{CONreq.qos \leq 1, ReqQos = CONreq.qos, FinQos = min(accept.qos, ReqQos)\}$ $\delta_2 = \emptyset$
Constraint set	$CS\_I_2 = C_2\theta_2 = \{x \leq 1, y = x, z = min(w, y)\}$

I_3 (Forward)	
U_1	$U_1 = \{(s_3, \theta_1, C_1, \delta_1)\}$ $\theta_1 = \{accept.qos/x, FinQos/y\}$ $C_1 = \{y = min(x, ReqQos)\}$ $\delta_1 = \emptyset$
Constraint set	$CS\_I_3 = C_1\theta_1 = \{y = min(x, ReqQos)\}$

I_4 (Backward)	
V_1	$V_1 = \{(s_4, \theta_1, C_1, \delta_1)\}$ $\theta_1 = \{accept.qos/x, FinQos/y\}$ $C_1 = \{y = min(x, ReqQos)\}$ $\delta_1 = \emptyset$
Constraint set	$CS\_I_4 = C_1\theta_1 = \{y = min(x, ReqQos)\}$

Figure 2.4: Using the algorithms to extract required constraints for the example

$Trace_1$	CONreq(1)	connect(1)	Refuse	CONcnf(-)
I_1	Refuse	connect(x)	Refuse	CONcnf(-)

Figure 2.5: Using invariant I_1 to check the execution trace $Trace_1$

$Trace_2$	CONreq(1)	connect(0)	Accept(1)	CONcnf(+,0)
I_2	CONreq(x)	connect(y)	Accept(w)	CONcnf(+,z)
I_3			Accept(x)	CONcnf(+,y)
I_4			Accept(x)	CONcnf(+,y)

Figure 2.6: Using invariant I_2, I_3 and I_4 to check the execution trace $Trace_2$

- $Trace_1$: Since the analysis found $CONcnf(-)$ in the trace and failed looking for the couple $refuse/connect(x)$, then the trace is erroneous regarding the invariant I_1. Note that the found error is control driven, so it is not needed to look at the constraints at all

- $Trace_2$: There are three invariants which are candidate for this trace. Matching the trace with the invariants shows that there is not any control driven error, so we use constraints and the value of the variables in the trace to decide about the possibility of data driven errors

$CS\_I_2 \cup \{x = 1, y = 0, w = 1, z = 0\} = \{x \leq 1, y = x, z = min(w, y)\} \cup \{x = 1, y = 0, w = 1, z = 0\} = \{1 \leq 1, 0 = 1, 0 = min(1, 0)\}$
$1 \leq 1$ is true, $0 = 1$ is false and $0 = min(1, 0)$ is true, so the invariant I_2 is false on $Trace_2$

$CS\_I_3 \cup \{x = 1, y = 0\} = \{y = min(x, ReqQos)\} \cup \{x = 1, y = 0\} = \{0 = min(1, ReqQos)\}$
0 is the minimum of 1 and $ReqQos$ only if $ReqQos$ is equal to 0, so the invariant I_3 is true on the trace $Trace_2$ if $ReqQos = 0$.

$CS\_I_4 \cup \{x = 1, y = 0\} = \{y = min(x, ReqQos)\} \cup \{x = 1, y = 0\} = \{0 = min(1, ReqQos)\}$
0 is the minimum of 1 and $ReqQos$ only if $ReqQos$ is equal to 0 so the invariant I_4 is true on the trace $Trace_2$ if $ReqQos = 0$.

As we found an inconsistency in the checking of the invariant I_2 with the trace $Trace_2$ we conclude that the trace $Trace_2$ is false. Checking the other invariants on the trace is not necessary but figure here as an example of variable simplification.

2.5 Conclusion

Passive testing methods for network protocols can be classified into naive and invariant based approaches. In the naive approach the implementation trace which is recorded during the execution of the protocol under test is compared with total of the specification in a forward or backward manner. This is where, in the invariant based approach only critical properties of the specification (i.e. invariants) which are extracted by an expert are compared with the implementation trace. By using invariant based approach, not only a lot of extra processing is reduced, but also we can focus on the critical properties of the program under test.

Passive testing methods can be compared from another aspect. Some methods are limited to testing only control driven aspects of the implementation, i.e. the order of occurrences of the input/output signals, while other methods are capable of testing both control driven and data driven aspects of the implementation i.e. the values of the signal's parameters. For testing control driven aspects it is sufficient to use FSM for specification, while for data driven aspects it is needed to use EFSM.

In this chapter a new method for passive testing of both control driven and data driven aspects of the network protocols using an invariant based approach was presented. The intended properties of the specification are expressed using some control driven invariants given by an expert. After that, using the given algorithms, the invariants are checked on the specification off-line. Also to take the data driven aspects into account, for the correct invariants, some constraints over the variables of the invariants are extracted. For passively testing the implementation traces, it is sufficient to compare, on-line, the trace with the invariants regarding the constraints using pattern matching. A trace is correct while it is matched with the invariant and the invariant's constraints are not conflicted regarding the values of the signal's parameters.

To show the applicability of the presented method, passive testing of the Simple Connection Protocol (SCP) using the presented method was illustrated.

Chapter 3

Improvement of the interval determination approach - The backward checking approach

3.1 The backward checking : overview

As told before all the various passive testing approaches fail to detect faults because of the trace length. In addition, the approach presented in [14], that we take as a basis for the current chapter, has notably a major problem : it postulates that the variables at the beginning of the trace analysis are indefined, i.e. there possible values are equal to their definition domains, what is generally false.

We will apply a different reasoning : since the trace "arrivesÂ´Â´ somewhere it means we have already at this moment some information on this trace. Then, why not take the trace in the reverse way ; this has the huge advantage to make us starting correct information instead of the total unknown. Add there is a second huge possibility that offers us this technique : why would we stop at the end of the trace (i.e. the observed event sequence) if it was not clearly refused or accepted (i.e. because of the *possible* verdict) ? Indeed, it is possible to explore the past of the trace in the specification end to artificially lengthen it - that is exactly the problem we wanted to solve.

The Backward Checking algorithm is then an approach of passive testing on EFSMs derived from the testing by determination of the intervals of variables such as the precursor one of [14] (see 1.5). In this kind of passive testing, the variables are defined in intervals and the goal is to find inconsistency between these intervals and the information given in the trace exploration about these variables. We consider that we have a system under test on which we place an observation point. We suppose that this observation point records the event traces respecting their causal order. We assume that finding the order of these events is a well studied and resolved problem.

We also consider that the observation can start at any moment, without any preliminary operations. In particular we don't place the system under test into a known configuration because it would mean that we somehow control the system. If we control or interact with the system then we enter in an active testing architecture and the problem can be solved according to the works of this community.

The Backward Checking algorithm is composed of two main phases. First, it follows a given event trace backward to find the possible initial configurations at the beginning of the trace. Secondly, it starts from these configurations and explore backward every possible path of the specification, with help of prunning operations and a transition choice strategy, to reduce as much as possible the search.

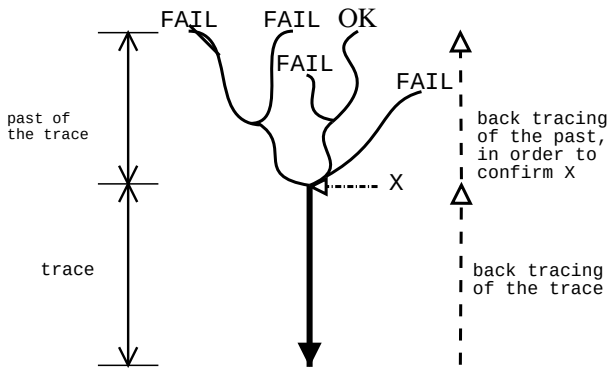

Figure 3.1: Overview of Backward Checking

3.1.1 In the trace

During the first phase, if an inconsistency is detected it means that the event trace is not correct, and then an error had been detected. If, in the end of the event trace analysis, we didn't find an inconsistency, it means the studied trace is possible with the obtained initial configurations. Then, we launch the second phase, that is to say the exploration of the past of this trace.

3.1.2 In the past of the trace

In the second phase we try to confirm the intervals in which the variables are defined according to the initial configurations. The algorithm finishes with a positive answer (trace is valid) at the first confirmed configuration, that is to say the first configuration in which every variables had been confirmed, or with a negative answer (invalid trace) if every branch of the exploration tree leads to an inconsistency. We can say that this algorithm is optimistic in the way that it will be fast to say that there is no error, but slower to say that there is one. This fact is coherent with reality because we suppose that an error in an event trace is an exceptional behaviour.

The different branches of the exploration tree are the possible successions of transitions, taken in a backward manner, from the initial configurations resulting from the first phase.

To avoid loops and reduce the size of the new configurations we use the *intersection* and *privation* operators which will be presented later in Sec.3.3.2 (Def.17 and 18).

In addition the transitions are choosen through a process of selection depending on several criteria. This strategy is the following : the transitions are classified by order or priority where the biggest priority means the most chances this transition can lead to a verdict. That is what will be presented in Sec.3.3.2 as the *Transition Choice Strategy*.

3.2 The backtracing of a transition

3.2.1 Intuitively

If we respectively note c_i and c_f the configurations in s_i and s_f, firing the transition t in a backward manner follows this operations :

- reverse the action sequence
- copy the intervals of variables that are not in an action left member from c_f to c_i
- treat each action according to its type (the three different action types are presented herafter)
- refine and control the consistency of the found values with the constraints contained in predicates, outputs, and inputs.

3.2.2 Validation of a trace

The algorithm must confirm only one of the starting configuration (i.e. one of the resulting configuration of the backtracing of the trace). It is only needed to fire transitions back (i.e. from their ending state to their starting state) till we reach a validation criterion. There are four validation criteria :

- the predicates
- the actions such as : $w \longleftarrow constant$
- the actions such as : $w \longleftarrow f(u, v, ...)$, where the variables in the action right member are themselves confirmed
- the transition inputs and outputs (depends if in the trace or in the past of the trace)

The figure 3.2 shows the example of unvalidation caused by a variable update. Let suppose that $(4, < x = 1 >, \emptyset)$ is the initial configuration of the tested event trace. We must confirm the value of x. When we arrive in state 2, we obtain the $(2, < x = 1 >, \emptyset)$

configuration. But the ①⟶② transition tells us that the value of x must be 0, not 1 (what we found in our current configuration) : then the $(4, < x = 1 >, \emptyset)$ starting configuration is not valid, and the hypothetical trace that produced it comes from a faulty implementation !

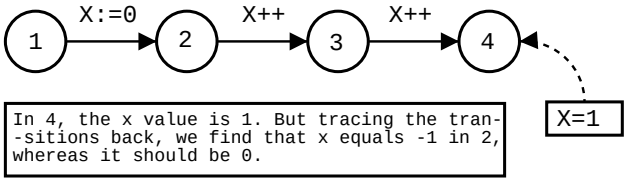

Figure 3.2: Contradiction in the past

We intuitively see that a set of variables must be *confirmed* or *validated*. We propose a definition of these particular variables that we name *determinants*. And then we extend the concept of CCS by the following definitions.

Definition 15 *A variable v is validated (also called confirmed) when we find in the processed transition the information saying it must be in a set of value I and we have the relation : $I \subset I_c$, where I_c is the current set of values of v.*
A variable is called determinant in the past of the trace if it was not validated yet.

Definition 16 *Let M be an EFSM. An Extended Candidate Configuration Set (ECCS) is a 4-tuple $(e, R, Assert, D)$, where :*

- *e is a state of M,*
- *R is an environment,*
- *$Assert$ is an assertion and,*
- *D is a set of determinant variables.*

A configuration is validated (also called confirmed) when it contains no more determinant (the set D is empty).

A variable v becomes a determinant one in the three following cases :

- after the trace analysis, the interval of v is strictly included in its definition domain
- during the past of the trace analysis, v appears in a predicat which already contains a determinant variable w
- during the past of the trace analysis, v is in a right member of an action whose left member is a determinant variable w

It can be noted that the second and third cases are situations in which the determinant property is transmitted to other variables. Indeed, there a dependance between the variables and all of them must then find a confirmation.

3.2.3 The inversed actions

A main difficulty is the application of the inverse action A^{-1}. The inverse actions will be processed in a reversed order. Hence the following normal ordered actions $\{a_1, \ldots, a_n\}$ will be processed in an order: $\{a_n, \ldots, a_1\}$.

Each inverse action depends on the type of the corresponding normal action. There are three types of actions:

1. $w \longleftarrow constant$
2. $w \longleftarrow f(u, v, \ldots)$ where w is not a variable of f
3. $w \longleftarrow f(w, u, v, \ldots)$ where w is also a variable of f

These three types of actions are assignations : they overwrite the value of the left variable w with the value of the right component. We note that the value of w is modified by an action, but the other variables after action keep the value they had before the action and that only the value of the variable w will be modified by back tracing a transition. Except for this, every type of action must be inverted:

1. Action of type 1. The value of w after the action is a *constant*. This gives us a first occasion of detecting an error. If the *constant* does not conform the current constraint then we are on an invalid path. Otherwise, we replace every occurrence of w with the *constant* and refine the constraints of other variables. However, it is impossible to know the value of w before the action; indeed, actions simply overwrite the former value of w. After the action back tracing the value of w is UNDEFINED;

2. Action of type 2. We could take that $R(w)$ is equal to $R(f(u, v, \ldots))$ but we can be more precise: it is $R(w) \cap R(f(u, v, \ldots))$. In order to keep as much information as possible, every occurrence of w will be replaced by $f(u, v, \ldots)$. However, the value of w before action remains UNDEFINED;

3. Action type 3. This action brings no new constraint refinement on the variable w (on the left side of the assignment) after the action (left member) but it gives a constraint on the variable w (on the right side of the assignment) before the action. Consequently, every occurrence of w will be replaced by $f^{-1}(w)$.

3.2.4 Final checking phase

The check_consistency process is from [14] and is able to detect inconsistency in the definition of the variables by refining the intervals of variables and its constraints.
There are no big differences between the transition back tracing algorithms for the trace and for its past, and we ignore in the trace algorithm what can happen to the set of determinants before the action. Indeed, in the trace we do not determine variables; we can only refine their values, and we invalid the trace if the constraints are not consistent. For the trace we must check the output before processing the inverse actions. After processing

every action we can determine the variables involved in the input if its constraint is consistent with what we found. Otherwise, we invalid the transition.
On the other hand, we must check that the variable values that we found are consistent with the predicates. Otherwise, the path is invalid. Therefore, in the checking we must determine if a transition is valid or not. We need a process called check_pred for the past of the trace to modify the set of determinants. In the case of back tracing, we just need to add the predicates to the set of assertions and process check_consistency - no specific operations are needed.

3.2.5 Algorithms

The algorithms to fire a transition in a backward manner in the trace and also in its past are presented in this subsection.

In the first algorithm, the output must be checked before reversing the action list. After every actions of the transition are performed, the input has to be checked, that is to say to control that the value of the input belongs to the interval that was computed during the backtracing of the transition. If there is an inconsistency, the transition is not valid.

In the second algorithm the output value is obsolescent (i.e. stays the same before and after the production of the output) and the input variables are directly validated (i.e. the hypothetical input has produced a valid value).

Here is presented few algorithms. First the back_trace_transition(t, c) algorithm (3.3) which is used for backtracing a transition during the trace processing. Then the back_past_transition(t, c) algorithm (3.4) which is used for backtracing a transition during the past trace processing, and the check_pred(P, c) algorithm (3.5) which controls the consistency of the variables with the predicates.

The control allowing to know if it stays variables to determine will be done in the main algorithm.

The check_consistency(c) algorithm (3.6) derives from the one presented in [14]. It tests configurations consistency, refines their constraints and delete all unused assertions. It returns the processed configuration if the initial one is consistent, or NULL if it is not.
Variable assignment Rule **(R)**: for each variable range if we have a set of non empty intervals from the processing of the conjunctive terms then the new variable range consists of an interval whose lower (upper) bound is the minimum (maximum) of all the interval lower (upper) bounds.

The $check_redundancy(c)$ algorithm (3.7) aims to deal with convergence cases, in order to solve the infinite loops problem.

3.2.6 Example

An example of this process is presented on the following figures (3.8, 3.9, 3.10, 3.11, 3.12, 3.13).

Input : a transition t and a configuration c

Output : returns c' if $c' \xrightarrow{t} c$ is possible, NULL if not.

begin

1. **if** $output.v \notin c.R(v)$ **then do return** $NULL$;
2. **else do**
3. .$c' = clone(c)$;
4. .$c'.R(v) = Def(v)$;
5. .replace every occurrence of v in $c'.Asrt$ by $output.v$
6. $inverselistofactions$;
7. **foreach** action a **do**
8. .**if** action a is : $w \longleftarrow constante$ **then do**
9. . .**if** $c'.R(w) \cap constante = \emptyset$ **then do return** $incorrecttrace$;
10. . .**else do**
11. . . .$c'.R(w) = Def(w)$;
12. . . .replace every occurrence of w in $c'.Asrt$ by $constante$;
13. .**if** action a is : $w \longleftarrow f(\vec{x})$ **then do**
14. . .replace every occurrence of w by $f(\vec{x})$ in $c'.Asrt$;
15. . .**if** $w \in \vec{x}$ **then do** $c'.R(w) = R(f^{-1}(\vec{x}))$;
16. . .**else do**
17. . . .$c'.Asrt = c'.Asrt \wedge (\underline{w} \leq f(\vec{x}) \leq \overline{w})$;
18. . . .$c'.R(w) = Def(w)$;
19. **foreach** predicate p **do**
20. .$normalize(p)$;
21. .$c'.Asrt = c'.Asrt \wedge p$;
22. **if** $(input.v \notin c'.R(v))$ **then do return** $NULL$;
23. **else do**
24. .$c'.R(v) = Def(v)$;
25. .replace every occurrence of v by $input.v$ in $c'.Asrt$;
26. $check\_consistency(c')$;
27. **return** c';
28. **end**

Figure 3.3: The back_trace_transition algorithm

1. $c' = clone(c)$
2. inverse list of actions
3. **foreach** action a **do**
4. .**if** action a is : $w \longleftarrow constante$ **then**
5. . .**if** $c'.R(w) \cap constante = \emptyset$ **then**
6. . . .**return** incorrect trace
7. . .**else**
8. . . .$c'.R(w) = Def(w)$
9. . . .replace every occurrence of w in $c'.Asrt$ by $constante$
10. . . .$D = D - w$ //w is validated
11. .**if** action a is : $w \longleftarrow f(\vec{x})$ **then**
12. . .replace every occurrence of w by $f(\vec{x})$ in $c'.Asrt$
13. . .**if** $w \in \vec{x}$ **then**
14. . . .$c'.R(w) = R(f^{-1}(\vec{x}))$
15. . .**else**
16. . . .$D = D - w$
17. . . .$c'.Asrt = c'.Asrt \wedge (\underline{w} - cst \leq f(\vec{x}) \leq \overline{w} - cst)$
18. . . .$c'.R(w) = Def(w)$
19. . .$D = D \cup \vec{y}$
20. check_consistency(c')
21. check_pred(p,c')
22. **return** c'

Figure 3.4: The back_past_transition algorithm

1. **for each** predicate $v = f(\vec{x}) \in P$ **do**
2. .**if** $c.R(v) \cap c.R(f(\vec{x})) \subseteq c.R(v)$ **then do**
3. . .$c.D = c.D - v$; // v is validated
4. . .$c.R(v) = c.R(v) \cap c.R(f(\vec{x}))$;
5. .**else do return** $FALSE$;
6. **return** $TRUE$;

Figure 3.5: The check_pred algorithm

Input : c, configuration that must be refined
Output : returns the refinment of c, or NULL
begin
1 $c' \longleftarrow c$;
2 transform $c'.Assert$ in DNF;
3 $S \longleftarrow \emptyset$;
4 $At \longleftarrow \emptyset$;
5 **for each** conjunctive term D_t of $c'.Assert$ **do**
6 .$dt\_true \longleftarrow TRUE$;
7 .$refine \longleftarrow TRUE$;
8 .**while** $refine = TRUE$ **do**
9 . .$refine \longleftarrow FALSE$;
10 . .$R_l \longleftarrow c'.R$;
11 . .$R'_l \longleftarrow \emptyset$;
12 . .**for each** predicate p of D_t **do**
13 . . .$normalize(p)$;
14 . . .**if** $\sum_i (a_i R_l(x_i)) \subseteq R(\sim Z)$ **then do** /* p is TRUE */
15 . . . .remove p from D_t;
16 . . . .**go to** 12;
17 . . .**if** $\sum_i (a_i R_l(x_i)) \cap R(\sim Z) = \emptyset$ **then do**
18 . . . .$dt\_true \longleftarrow FALSE$;
19 . . . .**go to** 28;
20 . . .**for each** x_j, $j = 1, \ldots, k$ **do**
21 . . . .$R'_l(x_j) \longleftarrow \frac{R(\sim Z) - \sum_{i \neq j} (a_i R(x_i))}{a_j} \cap R_l(x_j)$;
22 . . . .**if** $R'_l(x_j) = NULL$ **then do**
23 . . . . .$dt\_true \longleftarrow FALSE$;
24 . . . . .**go to** 35;
25 . . . .**if** $R'_l(x_j) \subset R_l(x_j)$ **then do**
26 . . . . .$refine \longleftarrow TRUE$;
27 . . . . .$R_l(x_j) \longleftarrow R'_l(x_j)$;
28 .**if** dt_true = FALSE **then do** remove D_t from $c'.Assert$;
29 .**else do**
30 . .**for each** variable v **do**
31 . . .$At \longleftarrow At \wedge (v \in R_l(v))$;
32 . . .$S(v) \longleftarrow$ combination of $S(v)$ and $R_l(v)$, according to **R**;
33 **if** $|S| = 0$ **then do return** $NULL$;
34 **else do**
35 .$c'.R \longleftarrow S$;
36 .$c'.Assert \longleftarrow c'.Assert \wedge At$;
37 .**return** c';
end

Figure 3.6: The check_consistency algorithm

Input : c, the configuration to be checked

Output : X, a set of configurations

V is the set of already-seen configurations and X' is an intermediate set of configurations

begin

1. $X \longleftarrow \{c\}$;
2. **for** each configuration $c_V \in V$ **do**
3. . $X' \longleftarrow \emptyset$;
4. . **for** each configuration $c_i \in X$ **do**
5. . . $c'_i \longleftarrow c_i \sqcap c_V$;
6. . . **if** c'_i =NULL **then do**
7. . . $X' \longleftarrow X' \cup \{c_i\}$;
8. . . . **goto** 4;
9. . . **else if** ($c'_i \neq$NULL)&($c'_i = c_i$) **then do**
10. . . . **goto** 4;
11. . . **else if** ($c'_i \neq$NULL)&($c'_i \neq c_i$) **then do**
12. . . . $(c_i^a, c_i^b) \longleftarrow c_i \backslash c'_i$;
13. . . . **if** $c_i^a \neq$NULL **then do**
14. . . . . $X' \longleftarrow X' \cup \{c_i^a\}$;
15. . . . **if** $c_i^b \neq$NULL **then do**
16. . . . . $X' \longleftarrow X' \cup \{c_i^b\}$;
17. . $X \longleftarrow X'$;
18. **return** X;

end

Figure 3.7: The check_redundancy algorithm

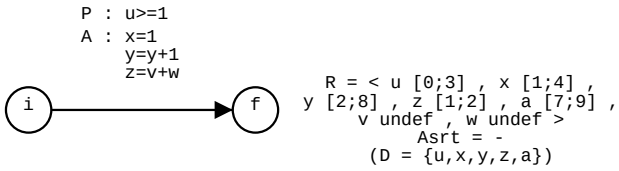

Figure 3.8: Starting point

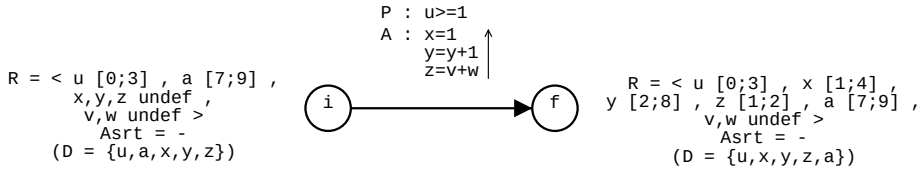

Figure 3.9: Unused variables are transfered

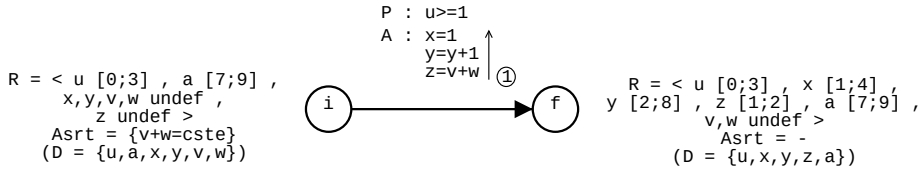

Figure 3.10: First type of action

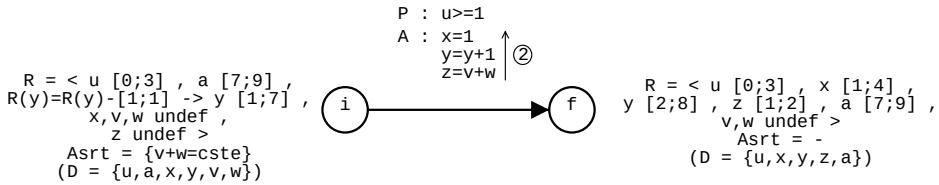

Figure 3.11: Second type of action

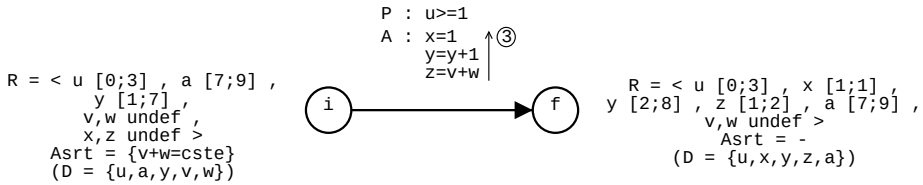

Figure 3.12: Third type of action

3.3 The main algorithms

Considering all the concepts defined above we can present in the following the main algorithms. The algorithm for the trace simply consists in backtracking transitions along

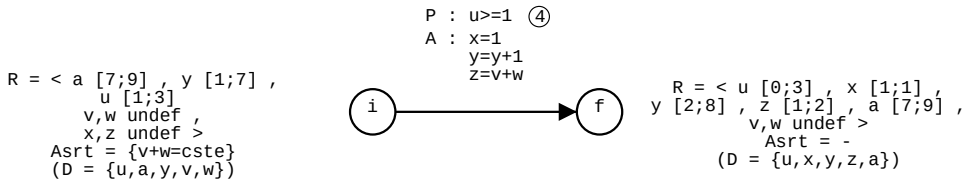

Figure 3.13: The *check_pred* and refinement

the trace, and in invaliding it if an inconsistency is found. The algorithm for the past of the trace is also a transition backtracking but without guidance of events (no more trace), associated with the operations to reduce search space and the Transition Choice Strategy.

3.3.1 Backward checking of a trace

The backward checking for a whole trace can be derived from the algorithm for back tracing a transition (Back_trace_transition) and it is presented in Fig.3.14.

3.3.2 Backward checking of the past of an event trace

The backward checking algorithm applied to the past of a trace consists of a breadth-first search in the past of the configurations, which are extracted from the back tracing of a trace due to the fact that one cannot use a variable value before it is assigned. In order to validate a trace, we only need to find a path binding a set of assignments or predicates to one of the configurations extracted from back tracing. We now proceed to the main algorithm. We first define the operations $\sqcap$ and $\setminus$ on the Extended Candidate Configuration Sets (ECCS) that will be used for pruning the exploration tree of the past.

The $\sqcap$ Operation.

It is an intersection between two configurations:

Definition 17 *Let be three configurations* $c_1 = (e, R_1, Assert_1, D)$, $c_2 = (e, R_2, Assert_2, D)$, *and* $c = (e, R, Assert, D)$. *We define the intersection operator* $\sqcap$ *as follows. If* $c = c_1 \sqcap c_2$, *then :*

1. *for each variable* v, $R(v) = R_1(v) \cap R_2(v)$ *where* $\cap$ *is the intervals intersection operator*
2. $Assert = Assert_1 \wedge Assert_2$ *where* $\wedge$ *is the boolean "and" operator*

Remark on $\sqcap$. The configuration states and the variable sets, which are not validated yet, are the same. If they are not, the "intersection" equals to NULL.

trace: The observed trace. *gettail*(*trace*) removes and returns the last i/o pair from *trace*.

X: Set of starting extended configurations from back trace of an event trace. Each configuration is a 4-tuple $(e, R, Assert, D)$

X': Current set of extended configurations

V: Set of known extended configurations

c': A new configuration

Returns TRUE if the sequence is correct, and FALSE otherwise

1. $V \longleftarrow X$
2. **while** $X \neq \emptyset$ & $i/o :=$ *gettail*(*trace*) **do**
3. .$X' \longleftarrow \emptyset$
4. .**for** each configuration $c \in X$ **do**
5. . .**for** each transition t where $t.end\_state = c.state$ and $t.i/o = i/o$ **do**
6. . . .$c' \longleftarrow$**Back_trace_transition**(t, c)
7. . . .$X' \longleftarrow X' \cup \{c'\}$
8. . . .$V \longleftarrow V \cup X'$
9. .$X \longleftarrow X'$
10. **return** FALSE

Figure 3.14: The trace backward checking algorithm

The $\setminus$ Operation.

It is a privation. Given two configurations c_1 and c_2, the result of $c_1 \setminus c_2$ is a couple (c_a, c_b). We obtain c_a by removing c_2 from c_1, but only in case of each variable is restricted to the intersection of the intervals c_1 and c_2, respectively. c_b is the rest of c_1.

Definition 18 *Given four configurations* $c_1 = (e, R_1, Assert_1, D)$, $c_2 = (e, R_2, Assert_2, D)$, $c_a = (e, R_a, Assert_a, D)$ *and* $c_b = (e, R_b, Assert_b, D)$, *we define the privation operator* $\setminus$ *as follows. If* $(c_a, c_b) = c_1 \setminus c_2$, *then :*

1. *for* c_a *:*
   (a) *for each variable* v, *we have got :* $R_a(v) = R_1(v) \cap R_2(v)$ *where* $\cap$ *is the intervals intersection operator*

(b) $Assert_a = Assert_1 \wedge \overline{Assert_2}$, *where* $\wedge$ *is the boolean "and" operator*

2. for c_b *:*

(a) $R_b = R_1$

(b) $Assert_b = Assert_1 \wedge (\bigvee_{i=0}^{|V|-1} (v_i \notin R_2(v_i)))$ *where* $\wedge$ *is the boolean "and" operator, and* $\vee$ *is the boolean "or" operator (be careful of priorities of parenthesis)*

Remark on $\setminus$. If $Assert_2$ equals to $\emptyset$, then c_a equals to NULL. Indeed $Assert_2$ means we have to keep all of the values that R_2 allows, yet on the contrary $\overline{Assert_2}$ implies that we must delete all of them.

General remark. The operations $\sqcap$ and $\setminus$ may return configurations, which are inconsistent. For example, the result of $c_1 \setminus c_1$ is not consistent. Moreover, some results may need to be refined. Indeed when two assertions are concatenated the constraints intervals of each variable may have to be changed. So we should use the **Check_consistency** procedure that has already been presented. For now, we consider that the results of $\sqcap$ and $\setminus$ are automatically checked and transformed by **Check_consistency**.

Let E_1 and E_2 be the cartesian products of the intervals of R_1 and R_2 respectively. The aim of the privation operator is to obtain the $E_1 \setminus E_2$ set - where $\setminus$ is the set minus symbol - possibly taking into account the assertion constraints. Therefore the ECCS c_a and c_b will contain the cases of c_1 that were not yet processed in c_2. c_a deals with the intersection of c_1 and c_2 whereas c_b is the rest of c_1 (cf Fig.3.15).

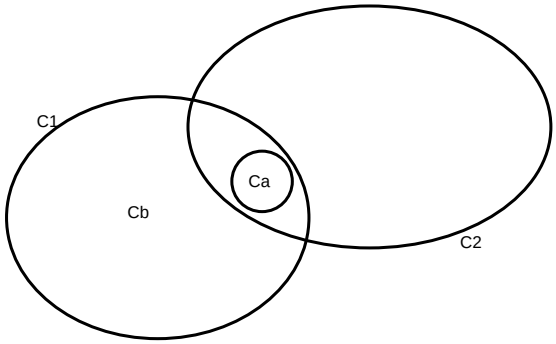

Figure 3.15: Privation operation scheme

Examples

Consider the configurations $c_1 = (e, < x = [0;5], y = [0;3] >, \_, \{x\})$ (where $\_$ means no assertion) and $c_2 = (e, < x = [0;2], y = [-1;1] >, \{x > y\}, \{x\})$, and three configurations c_i, c_a and c_b, which are defined as following :

- $c_i = c_1 \sqcap c_2$

- $(c_a, c_b) = c_1 \backslash c_2$

We first determinate c_i. R_i is defined as the intersection of R_1 and R_2, and $Assert_i$ is $Assert_1 \wedge Assert_2$. Then we have:
$c_i = (e, < x = [0; 2], y = [0; 1], \{x > y\}, \{x\})$.

Determinating c_a and c_b is a little bit more complicated. R_a is the intersection of R_1 and R_2, and $Assert_a$ is $Assert_1 \wedge \overline{Assert_2}$. Then we have :
$c_a = (e, < x = [0; 2], y = [0; 1] >, \{x \leq y\}, \{x\})$.

At last for c_b, we have the following properties. R_b equals R_1, and we must add $x < 0 \vee x > 2$ and $y < 0 \vee y > 1$ to $Assert_b$. Then we have :
$c_b = (e, < x = [0; 5], y = [0; 3] >, \{(x < 0 \vee x > 2) \wedge (y < 0 \vee y > 1)\}, \{x\})$.

Note that the two last configurations c_a and c_b are not refined as it was defined in [14]. If we apply the **Check_consistency** procedure, we obtain :
$c_a = (e, < x = [0; 1], y = [0; 1] >, \{x \leq y\}, \{x\})$ and $c_b = (e, < x = [3; 5], y = [2; 3] >, \_, \{x\})$.

Path Convergence

Consider a step r of our algorithm. If we find a configuration c that we have already found earlier, in a previous step or earlier in the step r, we have got a *path convergence* phenomena.

Definition 19 *Two paths P_1 and P_2 are convergent (in the past!) if they lead to the same configuration c.*

Consequently both P_1 and P_2 have the same past. So we will obtain the same information if we explore the common past from P_1 or from P_2. Consider that we have first followed P_1. When we find that P_2 converges toward c, we do not continue the exploration: we prune P_2 at c. The pruning enables us to deal with the infinite exploration paths.
Unfortunately extended configurations make convergences hard to be detected; they are non-empty intersections of configurations. We proceed as follows. Given three configurations c, c_1 and c_2, let c be equal to $c_1 \sqcap c_2$. Suppose that c_2 has been found before c_1. Then we have the following:

- $c =$NULL. c_1 and c_2 are independent and the respective pasts of c_1 and c_2 must be explored;
- $(c \neq$NULL$) \wedge (c = c_1)$. c_1 is included in c_2 and we must delete c_1;
- $(c \neq$NULL$) \wedge (c \neq c_1)$. c_2 is included in c_1 and we must substitute c_1 by $c_1 \backslash c_2$

The algorithm **Check_redundancy** described in 3.7, deals with the convergence cases.

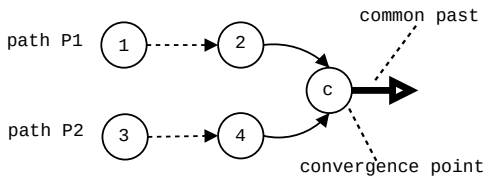

Figure 3.16: Example of Path Convergence

Algorithm of backward checking of the past of a trace

The Backward_checking_past algorithm backward traces the past of a trace in order to validate it. The input is the set of starting extended configurations, which we extracted from the trace back tracing.

Note that if the start configuration is invalid (not reachable from the initial configuration set) then we have to explore backward all the configurations to tell whether there is no valid path from the initial configuration set. However, if it is indeed valid, finding a valid path is enough. In most cases of passive testing, the traces do not contain faults and it is desirable to use a heuristic method to find a valid path. We now present such a procedure.

Transition Choice Strategy (TCS)

In addition the transitions are choosen through a process of selection depending on several criteria. This strategy is the following: the transitions are classified by order or priority where the biggest priority means the most chances this transition can lead to a verdict. The three criteria are :

- the starting state of the transition is the initial state of the EFSM. It is obvious that if we arrive at the initial state of the specification, the variables will be confirmed or leading to an inconsistency;
- the starting state of the transition is a state that was never seen in a former configuration. In this case we open more paths if the transition is successfully processed;
- the transition "determines" a variable, that is to say there is an action involving the variable v and v is the left member of the action. It is clearly an attempt to confirm the variable v.

For each criterion, a weight is given (empirically). Lets consider that the weights are named respectivelely w_1, w_2 and w_3 for the three criteria seen above and a is the number of actions in the transition. The global priority W_t is then given by the following formula :

$$W_t = w_1 + w_2 + a.w_3$$

In order to guide the heuristic search, we have figured out the end configurations. A configuration set c is an end configuration set if it satisfies one of the following conditions:

1. $c \cap c\_init \neq \emptyset$ where c_init is the initial configuration set of the machine
2. $c.D = \emptyset$
3. c is contained in another configuration set that has been explored

The second criteria is valid, since $c.state$ is reachable from the initial state of the machine, and there must be a valid path from the initial configuration.

We now present a heuristic search. We assign a weight for each configuration-transition pair $< c, t >$. Since we want to trace back to the initial configuration or reduce $c.D$, we increase the weight of such pairs. A priority queue Q contains all the configuration-transition pairs to be explored, sorted by their weights. The pair with the highest weight is placed in the head of Q and will be selected first.

The weight wgt of a configuration-transition pair $< c, t >$ with an initial value 0 can be incremented by the following rules :

1. if $t.start\_state = c\_init.state$, $wgt \mathrel{+}= w1$
2. if $t.start\_state$ has not been explored, $wgt \mathrel{+}= w2$
3. if $t.action$ defines k variables in $c.D$, $wgt \mathrel{+}= w3 * k$

The first two rules guide the search towards the initial state of the specification while the third one is to reduce the set of determinants. It is important to remark that we don't need to reach the initial state itself, and that a transition determining every variables left in the set of determinants is enough to conclude on the correctness of the explored path. This explains the importance of the third rule (we can note that the initial state is a particular case of it as it is supposed to determine every variables).

The values of $w1, w2$, and $w3$ can be given after practical usage.

The following is the procedure where

- Q: Set of configuration-transition pairs to be explored
- V: Set of already-explored Extended Configurations
- : Returns TRUE if the trace is correct, and FALSE otherwise.

In the worst case, this algorithm will explore backward all the possible configurations. When Q becomes empty no valid path is possible from the trace information from the passive testing and "FALSE" is returned - there are faults in the protocol system under test.

3.3.3 Exemples

The following examples show the process of the Backward Checking Approach on the Simple Connection Protocol (SCP).

begin

1. initialize Q, V
2. **while** $Q \neq \emptyset$ **do**
3. .take the first item $< c, t >$ from Q
4. .build a new configuration c' : $c' \longleftarrow$**Back_past_transition**(t, c)
5. .**if** $c' == NULL$ **then do**
6. . .**goto** 2;
7. .**if** $c'.D = \emptyset$ **then do**
8. . .**return** TRUE;
9. .c' =**Check_redundancy**(c', V);
10. .**if** $c' \neq \emptyset$ **then do**
11. . .$V \longleftarrow V \cup c'$;
12. . .**for** each transition t where $t.end\_state = c'.state$ **do**
13. . . .calculate the weight of $< c', t >$;
14. . . .insert $< c', t >$ into Q by its weight;
15. **return** FALSE;

end

Figure 3.17: The backward checking algorithm for the past of the trace

Validation of a trace

Let w be an event trace **CONreq(1)/connect(1), refuse/connect(1), refuse/connect(1), refuse/CONcnf(-)** of the implementation under test. We show how the backward checking algorithm give the verdict that this trace is correct.

After the backtracing of the trace we have the following configuration :
$(S2, < TryCount = 0; ReqQos = [0;3]; FinQos = [0;3]; CONreq.qos = [0;3] >, \emptyset, \{TryCount\})$

The figure 3.18 represente the execution of the algorithm in the past of the trace. The transition phase produce a configuration whose determinant list is empty. That means all the variables have found a confirmation of their values and then the trace is correct. Each line shows the composition of c, tr and c', which are respectively the current configuration of X, the current transition, and the new configuration of **Back_past_transition**(t,c). Several lines show X, X' or V when they are updated. They are respectively defined as the set of starting configurations (but also the set of current configurations after each step of the *while* loop), the current auxiliary set of configurations, and the set of already seen configurations.

step				
0	current conf.	$(S_2, < TC = 0; RQ = [0;3]; FQ = [0;3]; Crq.qos = [0;3] >, \emptyset, \{TC\})$		
	seen conf.	$(S_2, < TC = 0; RQ = [0;3]; FQ = [0;3]; Crq.qos = [0;3] >, \emptyset, \{TC\})$		
	transition $s_2 \longleftarrow s_2$	backtracing	next conf.	$(S_2, < TC = 0; RQ = [0;3]; FQ = [0;3]; Crq.qos = [0;3] >, \emptyset, \{TC\})$
		check redundancy	next conf.	$\emptyset$
			seen conf.	$(S_2, < TC = 0; RQ = [0;3]; FQ = [0;3]; Crq.qos = [0;3] >, \emptyset, \{TC\})$
	transition $s_2 \longleftarrow s_1$	backtracing	next conf.	$(S_2, < TC = [0;3]; RQ = [0;3]; FQ = [0;3]; Crq.qos = [0;3] >, \emptyset, \emptyset)$
	validation	it exists a conf. c with $c.D = \emptyset$: return TRUE		

Figure 3.18: Execution of the backward checking approach on a valid trace

Consider a false implementation of SCP, that has been used in [**?**] : the predicate of the transition $s_3 \longrightarrow s_1$ is replaced by $TryCount = 0$. The figures 3.19 and 3.20 show the executions of the backward checking algorithm (trace and past) on the trace **CONreq(1)/ connect(1), refuse/CONcnf(-)**. We have seen in 1.15 that this error is not detected by the algorithm presented in [14]. The trace is left "possible" for it.

We obtain the configuration $(S2, < TryCount = 2; ReqQos = 1; FinQos = [0;3]; CONreq.qos = 1 >, \_, \{TryCount; ReqQos; CONreq.qos\})$ from the back tracing of the

trace (Fig. 3.19) and we continue in the past. In the past, after the first step of the *while* loop, X is empty because the transition $\textcircled{s2} \longrightarrow \textcircled{s2}$ leads to a contradiction between $CONreq.qos$ value (= 1) and the predicate $CONreq.qos > 1$, and the transition $\textcircled{s1} \longrightarrow \textcircled{s2}$ is also invalid due to a contradiction between $ReqQos$ value (= 1) and the action $ReqQos = 0$. Then there is no more configuration to backtrack and the algorithm terminates, returning $FALSE$ - there are faults in the protocol implementation.

step	event	configurations
0	-	$(S_i; < TC = [0;3],\ RQ = [0;3],\ FQ = [0;3],$ $Crq.qos = [0;3],\ acc.qos = [0;3] >,\ \_)$ (for each i state number)
1	refuse / CONcnf(-)	$(S_3; < TC = 2,\ RQ = [0;3],\ FQ = [0;3],$ $Crq.qos = [0;3],\ acc.qos = [0;3] >,\ \_)$
2	CONreq(1) / connect(1)	$(S_2; < TC = 2,\ RQ = 1,\ FQ = [0;3],$ $Crq.qos = 1,\ acc.qos = [0;3] >,\ \_)$

Figure 3.19: Back Tracing the Trace

step				
0	current conf.	$(S_2, < TC = 2;\ RQ = 1;\ FQ = [0;3];\ Crq.qos = 1 >,\ \_\ ,\ \{TC;RQ;$ $Crq.qos\})$		
	seen conf.	$(S_2, < TC = 2;\ RQ = 1;\ FQ = [0;3];\ Crq.qos = 1 >,\ \_\ ,\ \{TC;RQ;$ $Crq.qos\})$		
	transition $\textcircled{s2} \longleftarrow \textcircled{s2}$	back tracing	next config.	$\emptyset$
	transition $\textcircled{s2} \longleftarrow \textcircled{s1}$	back tracing	next config.	$\emptyset$
	validation	there is no more configuration : return FALSE		

Figure 3.20: Back Tracing the Past of the Trace

3.4 Algorithm termination and complexity

In the first part of the algorithm (backtracking of the trace) there is no problem of termination because we follow the trace, so this step finishes when the trace finishes. The problem we had and we solved is in the second part of the algorithm (in the past of the trace). We present these problems in the following subsection.

3.4.1 Loop termination

There are two problems that we must solve : infinite paths, and infinite number of paths. These problems are often caused by loops.
A first infinite path case occurs when a path infinitely often reachs a configuration. This problem is solved thanks to the detection of path convergence (see 3.3.2), and ECCS operations that prevents exploring more than once in a configuration.

A second case occurs when a variable is infinitely increased or decreased. In this case the loop is limited by the upper or lower bound of the interval of definition of the variable. There are two cases when we have an infinite number of paths. First, a configuration has an infinite number of parents. Secondly there is an infinite path from which several paths start. But if the configurations number is finite, then a configuration can not have an infinite number of parents.
We proved the termination of the algorithm, and we present in the next subsection a study of the algorithm complexity.

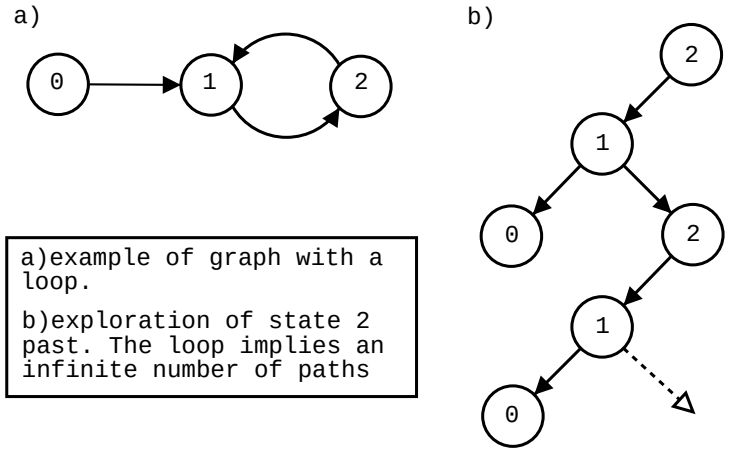

Figure 3.21: Infinite past

3.4.2 Complexity

In the first part of the algorithm (trace) the complexity depends on the trace. We have :
Proposition 1. Suppose that the observed event trace is of length l, then the complexity of the first part of the presented algorithm is proportional to l.

For the second part (past of the trace) the complexity depends on the number of possible configurations. A configuration includes a state number, interval of definition of variables, and a list of determinant variables. The complexity of the second part of the algorithm is :
Proposition 2. Let n_s be the number of states in the EFSM of the specification, $|R(x_i)|$ the number of values the variable x_i can take (in the interval of definition), and n the

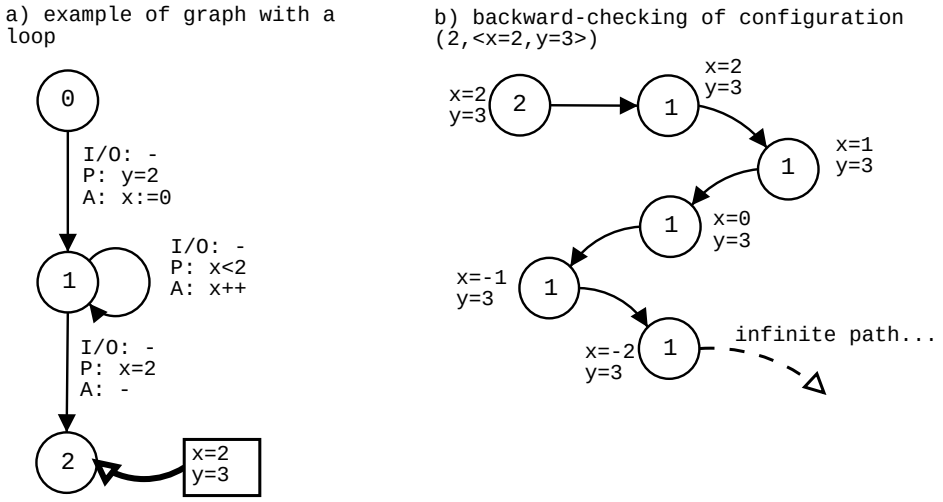

Figure 3.22: Infinite path produced by increasing (1)

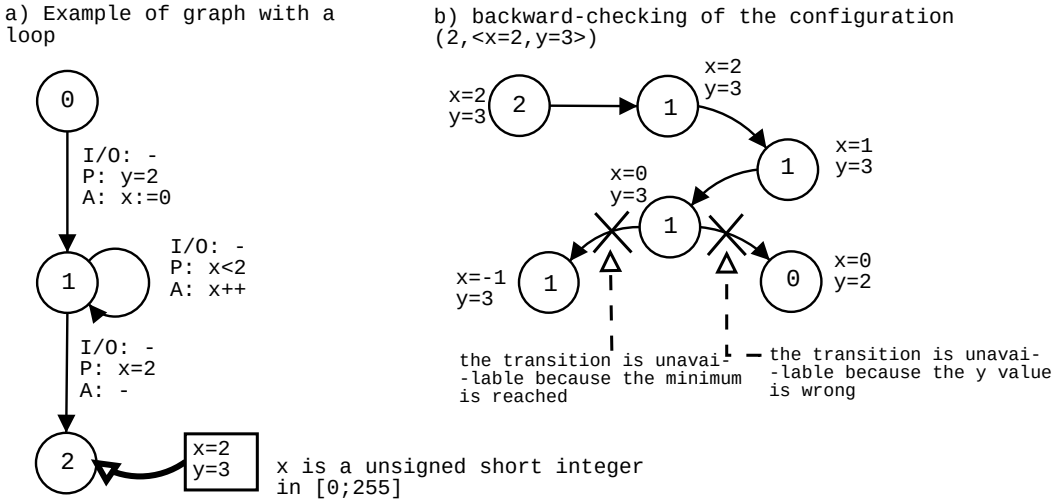

Figure 3.23: Infinite path produced by decreasing (2)

number of variables, then there is in $O(n_s(\prod_i |R(x_i)|)(2^n - 1))$ possible configurations.

We must balance this complexity with the power of the algorithm. The worst case of this algorithm is the case where there is an error because we must check every path of the past. When there is no error our algorithm gives a sure answer (in constrast with former algorithms) at the first correct path we meet (that is supposed to be fast using the heuristic). Anyway, the backward checking - if we consider only the trace analysis - is an improvement of former algorithm, and has the same complexity.

Nevertheless, this algorithm seems to be more efficient if the few properties are respected :

1. the number of variables is low
2. the definition domain $Def(v)$ of each variable v is narrow
3. le number of entering transitions in each state of the specification EFSM is small. Indeed, bigger is this number, more there are possible paths to be processed in the past of the trace

Chapter 4

Parallelized Backward Checking

In this chapter we discuss the parallel version of Backward Checking algorithm and its benefits on the complexity.

In order to improve furthermore the efficiency of the backward checking algorithm we present the following algorithms. As we can deduct from the study of the backward checking algorithm, the first part (the trace) will be treated in parallel but without reducing the complexity given the fact that it is closely bound to the trace. Actually, we saw it is proportional to the length of the trace, so we can only reduce the proportion multiplicator with the parallel version. The most interesting problematic is to parallelize the second part of the algorithm (past of the trace). In the previous sections we presented the process of the algorithm in the past and we presented the operations on the configurations. We have to make an important remark at this point of the work : the operations provide us a way to get independent configurations. In fact, if two configurations are dependent on one another it means that a total or partial inclusion exists between them - this will be detected by intersection and privation operations. The study of configurations is then possible in parallel. First, we present three variants of the parallel approach, and then we discuss their advantages and drawbacks.

For each of the following algorithms we have a parallel loop that computes the predecessor set of an ECCS (cf. Def.16 Sec.3.2.2). It means that if we dispose of a set of ECCS $(c_1, \ldots, c_n)$, all these configurations are processed in parallel instead of being processed sequencially. Nevertheless, we must check out and erase eventual redundancies after obtaining the predecessor set.

We can distinguish two methods to treat the parallelism: the first is a parallel algorithm by round and the second is a parallel algorithm without round that we call here direct algorithm. The method by round treats a set of transitions in parallel with a system of synchronization after each round whereas the direct method doesn't use synchronization.

For the redundancy checking in the method by round we consider two main ways to do. Either we check each new configuration with all the configurations seen till the last round, and then check the configurations of the current round between themselves, or we do the same in the opposite order.

So, finally, there are three main ways to parallelize the process : two by round and one

direct. Now, we present these algorithms.

For the following algorithms we consider that Q is the set of configuration-transition pairs to be explored, V is the set of already-seen Extended Configurations, and X is the set of Extended Configurations seen in the current round. Also $Back\_past\_transition(t, c)$ is a function that takes a configuration c and backtracks transition t returning the new configuration c', $Check\_redundancy(c', V)$ takes a configuration c' and checks its inclusions with each configurations of the set V, and $Check\_redundancy\_till(c', X, i)$ compares c' only with the i first configurations of the set X (i excluded). More details about these operations can be found in [38]. Also, $c'.D$ denotes the determinant set D of the configuration c'. The algorithms return TRUE if the trace is correct, FALSE if it is not.

4.1 The three algorithms

The first presented algorithm (4.1) is the one proposed for a processing with rounds and checking first the current round found configurations with the configurations seen till the last round, the second one (4.2) is by round and checking first the current round found configurations between themselves. The third presented algorithm is the direct one (4.3).

4.2 Complexity

In the present section we discuss about the advantages of the different algorithms and about the complexity of both normal and parallel algorithms.

4.2.1 Compared study of the proposed algorithms

In the previous subsection we proposed three algorithms. Each one has advantages and drawbacks that the present subsection wants to expose in order to define which method looks the most interesting according to the specification.

First, we can emit a few comments about the number of needed processors for the two methods (round and direct).

For a round method, the number of processors can be limited to the number of transitions of the EFSM. Indeed, at a given round we can't have more transitions to backtrace than the number of transitions contained in the EFSM. So it's not usefull to use more than $|T|$ processors (where $|T|$ is the number of transitions of the EFSM). Neverthless, for a correct processing the architecture has to prevent a transition from not having a corresponding processor. Finally, we say the method by round needs exactly $|T|$ processors.

For the direct method, as there is no synchronization, we can only limit the number of processors to the number of ECCS the EFSM can produce. This number is big and such a requested architecture is surely not available in today's world for complex protocols or services. However, we don't need so many processors and in the limit case where we have access to only one processor, the process is equivalent to the former sequencial process

begin

1. initialize Q, V, X;
2. **while** $Q \neq \emptyset$ **do**
3. .**parallely for each** i-processor **do**
4. . .take the i-th item $< c, t >$ from Q;
5. . .$c' \longleftarrow$**Back_past_transition**(t, c);
6. . .**if** $c' \neq \emptyset$ **then do**
7. . . .**if** $c'.D = \emptyset$ **then do**
8. . . . .**return** TRUE;
9. . . .$c' =$**Check_redundancy**(c', V);
10. . . .**if** $c' \neq \emptyset$ **then do**
11. . . . .$X \longleftarrow X \cup c'$;
12. .**parallely for each** i-processor **do**
13. . .$c' = X_i$;
14. . .$c' =$**Check_redundancy_till**(c', X, i);
15. . .**if** $c' \neq \emptyset$ **then do**
16. . . .$V \longleftarrow V \cup c'$;
17. . . .**parallely for each** transition t where $t.end\_state = c'.state$ **do**
18. . . . .calculate the weight of $< c', t >$;
19. . . . .insert $< c', t >$ into Q by its weight;
20. **return** FALSE;

end

Figure 4.1: The first algorithm by round

begin

1. initialize Q, V, X
2. **while** $Q \neq \emptyset$ **do**
3. .**parallely for each** i-processor **do**
4. . .take the i-th item $< c, t >$ from Q;
5. . .$c' \longleftarrow$**Back_past_transition**(t, c);
6. . .**if** $c' \neq \emptyset$ **then do**
7. . . .**if** $c'.D = \emptyset$ **then do**
8. . . . .**return** TRUE;
9. . . .$X \longleftarrow X \cup c'$;
10. .**parallely for each** i-processor **do**
11. . .$c' = X_i$;
12. . .$c' =$**Check_redundancy_till**(c', X, i);
13. . .**if** $c' \neq \emptyset$ **then do**
14. . .$c' =$**Check_redundancy**(c', V);
15. . . .**if** $c' \neq \emptyset$ **then do**
16. . . . .$V \longleftarrow V \cup c'$;
17. . . . .**parallely for each** transition t where $t.end\_state = c'.state$ **do**
18. . . . . .calculate the weight of $< c', t >$;
19. . . . . .insert $< c', t >$ into Q by its weight;
20. **return** FALSE;

end

Figure 4.2: The second algorithm by round

begin

1. initialize Q, V;
2. **while** $Q \neq \emptyset$ **do**
3. .**parallely for each** processor **do**
4. . .take the first item $< c, t >$ from Q;
5. . .$c' \longleftarrow$**Back_past_transition**(t, c);
6. . .**if** $c' \neq \emptyset$ **then do**
7. . . .**if** $c'.D = \emptyset$ **then do**
8. . . . .**return** TRUE;
9. . . .**sequencially do**
10. . . . .$c' =$**Check_redundancy**(c', V);
11. . . . .**if** $c' \neq \emptyset$ **then do**
12. . . . . .$V \longleftarrow V \cup c'$;
13. . . .**if** $c' \neq \emptyset$ **then do**
14. . . . .**parallely for each** transition t where $t.end\_state = c'.state$ **do**
15. . . . . .calculate the weight of $< c', t >$;
16. . . . . .insert $< c', t >$ into Q by its weight;
17. **return** FALSE;

end

Figure 4.3: The direct algorithm

(without any gain in performance). So to have an effective gain we need at least two processors, and in order to have an optimal gain we propose to use an architecture with at least $max|T\_in_i|$, the maximal number of entering transitions of a state from the EFSM specification - this to cope with the last parallel loop of the algorithm.

On the other hand, it is interesting to compare the three algorithms in order to show which one is the most efficient in a given situation. Let n be the number of already seen configurations till the i-th round (for the methods by round) or at a given time (for the direct method). We know that in this case the maximal number of transitions to be analysed is equal to the number of transitions contained in the EFSM. Let T be this number. The question we want to answer is which technique will take the less time to analyse t transitions (where $t \in [0;T]$).

For the direct method, we process one transition per processor, then we eventually add new non-empty ECCS in the already seen ECCS set and add the next transitions to be backtracked according to their priority weight. In other words, the already seen configuration set increases incrementally after each processed transition that gives a new non-empty ECCS. Then, the maximal time needed to process the t transitions can be given by :

$$\sum_{k=0}^{t-1}(n+k) = tn + \sum_{k=0}^{t-1} k (= tn + \frac{t(t-1)}{2})$$

For the first method by round (with comparison of ECCS of the current round between themselves first), the maximal number of comparison is :

$$\sum_{k=0}^{t-1} k + xn (= xn + \frac{t(t-1)}{2}),$$

where x is the number of refined ECCS remaining after comparison with $x \in [1;t]$

For the second method by round the formula is :

$$tn + \sum_{k=0}^{y-1} k (= tn + \frac{y(y-1)}{2}),$$

where y is the number of refined ECCS remaining after comparison with $y \in [1;t]$

On the base of the above statements, we can compare the different methods. First, we compare the direct method with the first method by round. The direct method is more efficient for t transitions iff :

$$\begin{gathered} tn + \sum_{k=0}^{t-1} k \leq \sum_{k=0}^{t-1} k + xn \\ tn \leq xn \end{gathered}$$

From the above lines we conclude that the direct method is never better than the round one.

In the same way we determine that the direct method is more efficient than the second method by round iff :

$$\begin{gathered} tn + \sum_{k=0}^{t-1} k \leq tn + \sum_{k=0}^{y-1} k \\ \sum_{k=0}^{t-1} k \leq \sum_{k=0}^{y-1} k \\ t \leq y \end{gathered}$$

From these results we deduct that the direct method is always slower than any method by round.

We must also compare the two techniques by round and we obtain :

$$\begin{array}{c} \sum_{k=0}^{t-1} k + xn \leq tn + \sum_{k=0}^{y-1} k \\ \sum_{k=y-1}^{t-1} k \leq (t-x)n \\ \frac{t(t-1)-y(y-1)}{2} \leq (t-x)n \end{array}$$

This relation shows us that the first technique is the fastest since there is at least one refinement ($x < t$) and n is big. The number n will increase as the algorithm processes, and its upper bound can be high. Then the first technique seems to be more promising. But if $x = t$ (no refinement) then the first technique is better only if :

$$\sum_{k=y-1}^{t-1} k (= \frac{t(t-1)-y(y-1)}{2}) = 0,$$

that is to say $y = t$ (no refinement in the second technique also). Unfortunately, there is no way to predict the number of refinement in first nor second technique, and we count on experiments to show which one is statistically more interesting.

4.2.2 Some remarks

First we must note that the techniques by round don't really benefit from the Transition Choice Strategy as every transition of a round is backtracked at the same time. It can then make a sense if the processors are different and classified from the most to the less performant. Then the first transition of the list - the most promissing one - is processed by the most efficient processor and in case the transition brings a positive conclusion ($TRUE$) the parallel algorithm stops immediately.

The direct method suffers from its sequencial part. On the other hand we can remark that this method uses optimally the Transition Choice Strategy. In addition, it can be efficient in case there are lots of unconsistent transitions since the sequencial part is then not processed.

4.3 Complexity of the sequential Backward Checking

In the first part of the algorithm (trace) the complexity depends on the trace. We have according to [38]:

Proposition : Suppose that the observed event trace is of length l, then the complexity of the first part of the presented algorithm is proportional to l.

For the second part (past of the trace) the complexity depends on the number of possible configurations. A configuration includes a state number, interval of definition of variables, and a list of determinant variables. The complexity of the second part of the algorithm is :

Proposition : Let n_s be the number of states in the EFSM of the specification, $|R(x_i)|$ the number of values the variable x_i can take (in the interval of definition), and n the number of variables, then there is in $O(n_s(\prod_i |R(x_i)|)(2^n - 1))$ possible configurations.

We must balance this complexity with the power of the algorithm. The worst case of this algorithm is the case where there is an error because we must check every path of the past. When there is no error the backward checking algorithm gives a sure answer (in constrast with former algorithms) at the first correct path we meet (that is supposed to be fast using the transition choice strategy). Anyway, the backward checking - if we consider only the trace analysis - is an improvement of former algorithm, and has the same complexity.

4.3.1 Gain in complexity

Here, we present the maximal complexity of our parallel algorithm.

Obviously, concerning the trace itself the complexity of normal and parallel algorithms are comparable. They are both proportional to the length of the trace. The difference is that the normal algorithm processes a transition in N_t where N_t is the number of transitions holding a given event couple, while the parallel algorihtm does it in 1.

With regards to the past of the trace, the complexity of the parallel algorithm is much lower. Actually, its maximal complexity depends on the number of states in the EFSM because the longest path without loops between two states goes through all the other states once. Nevertheless, a loop, in other words the path from a state to the same state through a non empty sequence of transitions, does not imply that the two considered configurations are equal. So the complexity also depends on the values of the variables, and more specially of the one with the largest definition interval.

Proposition : Let $|R(x_i)|$ be the number of values the variable x_i can take (in the interval of definition), and N_s the number of states in the EFSM. The complexity of the parallel algorithm is then given by the following fomula :

$$max|R(x_i)|.N_s$$

This result is very important because it makes from the parallel backward checking algorithm the first ever made linear algorithm for exhaustive error detection on EFSM. Then a total real-time error detection is possible, and its applications in terms of implementation correction and also in certain cases in Intrusion Detection System (IDS) is fundamental (cf. [40]).

4.4 Application examples

In this section we propose to illustrate the theoretical advantages of the parallel algorithm. For this purpose we introduce four network protocols with various characteristics. Two of them are small ones : Simple Connection Protocol (SCP), and Initiator-Responder Protocol (INRES). The third, Optimized Link State Routing Protocol (OLSR) is used in

ad-hoc networks, and the fourth is the Open Shortest Path First Protocol (OSPF) used in wired networks.

Then we present the compared performances of the algorithms, emit remarks and conclude.

4.4.1 SCP

SCP allows us to connect an entity called *upper layer* to an entity called *lower layer* after a negociation of the Quality of Service desired for the connection. SCP layer is between the two layers and has a role of intermediary for the connection establishment. The upper layer communicates the desired QoS to SCP layer and SCP layer gives three tries to the lower layer to accept the QoS. If the three tries fail, the upper layer must reinitiate the protocol. Elsewise, a connection is open between the upper and lower layer, and they can exchange data. An EFSM specification of this protocol can be found in [4].

4.4.2 INRES

The INRES System is a simplified service and protocol - as SCP - that tries to establish a connection followed by a data exchange between two processes : an initiator and a responder. We consider in the study the EFSM of the initiator process as described in [7]. In this system, the initiator sends a first message and waits a limited time after which it must reinitiate the protocol. If it receives the appropriate message, the connection is established and the two process can exchange messages.

4.4.3 OLSR

The OLSR protocol [23] is a link-state proactive protocol designed specifically for mobile ad-hoc networks. OLSR manages to diffuse routing information through an efficient flooding technique. The key innovation of this protocol is the concept of Multi Point Relays (MPRs). A node multipoint relay is a subset of its neighbors whose combined radio range covers all nodes two hops away. In order for a node to determine its minimum multipoint relay set based on its two-hop topology, periodic broadcasts are required. Similar to conventional link-state protocols the link information updates are propagated throughout the network. However, in OLSR when a node has to forward a link update it only forwards it to its MPR set of nodes. Finally, the distribution of topological information is realized with the use of periodic topology control messages and has as a result each node knowing a partial graph of the topology of the network that is further used to calculate the optimal routes. An EFSM of this protocol is available in [40].

4.4.4 OSPF

OSPF [24] is a very widely spread intra-domaine routing protocol using the Open Shortest Path First Algorithm. An OSPF Neighbour State Machine is used to maintain connec-

tions between two OSPF neighbour routers, and to exchange information on the link state through Link State Advertisement (LSA). The variables, like sequence numbers, are used for recording the present connections state. Such an EFSM is presented in [14].

4.4.5 Compared Performances

Given the EFSM of the pre-cited protocols, we can compare the performances of the different algorithms on them. To have a clear view of the results we sum them up in the following tables. N_s is the number of states, N_v the number of variables, V number of values that each variable can take, C the complexity, L_i the largest variable interval, and N_p the number of required processors.

protocol	N_s	N_v	V	C
SCP	4	4	4; 4; 4; 4	15360
INRES	4	3	5; 6; 2	1680
OLSR	4	6	2; 2; 2; 2; 5; 5	100800
OSPF	8	8	1; 2; 2; 2; 2; 2; 2; 2	261120

Figure 4.4: Complexity of the Sequencial Algorithm

protocol	L_i	N_s	N_p	C
SCP	4	4	8	16
INRES	6	4	28	24
OLSR	5	4	23	20
OSPF	2	8	88	16

Figure 4.5: Complexity of the Parallel Algorithm

In the two tables we present respectively all the characteristics needed to compute the complexity of the non-parallel and parallel algorithms. The figures in the tables last columns are not in seconds nor milliseconds but in transition processing time. It allows us to compare the results and can be abstracted to whichever unit of time. For instance we can read from these tables : if we need 261120ms (more than 4 minutes) to finish a trace analysis from OSPF with sequential algorithm, we need only 12ms with the parallel algorithm.

There are few other details to be explained about the tables data. For the number of values that a variable can take we made a simplification in two cases :

- in the INRES protocol one variable is not bounded (*old_data*) but as it is used only for a simple comparison (no intern modification) we can abstract it to a boolean;

- in the OSPF two variables are defined on the IP address domain ($[0; 2^{32} - 1]$) but are used for simple comparison, so we abstracted them too.

We must note that the parallelization implies the use of multi-processors architectures. Although there is no problem to organize a cluster of machines in wired world, it isn't the same situation for wireless networks in terms of bandwidth and security. The only valid possible parallel architecture for wireless protocols would be a massively multi-processor machine treating the protocol and its control through our parallel algorithm at the same time. For instance, it would need a 23 processors portable machine to control OLSR in real situation, and that is the reason why the OLSR example is nowadays not experimentally possible.

If we compare the protocols we can see that there is no doubt about the gain the parallel algorithm brings : it is 70 (INRES) to more than 20000 (OSPF) times faster and it processes the four protocols with an average of less than 20 transitions process, which is very low for an exhaustive analysis.

However, we can wonder when we look at the INRES protocol. Indeed, its complexity is 10 times lower than the SCP's one for the non-parallel algorithm and 1.5 time bigger for the parallel one. In addition, it needs even more processors than OLSR which is commonly considered as more complex. This can be explained by the fact that SCP, INRES, and OLSR are comparable in terms of number of states, but INRES uses variables that allows more values. In practice, the limitating factor tends to be the number of values allowed to the variables because EFSMs of real protocols are rarely composed of more than a ten of states.

4.5 Conclusion and future work

In this chapter we proposed few methods of parallelization of the backward checking pasive testing algorithm and the study of each of these methods in terms of complexity and performance. Finally we have shown the undeniable power of the parallel version illustrated by the application to four well known and used communication protocols. This study has proven the linear complexity of the parallel algorithm, then making from it the first ever linear exhaustive algorithm for passive testing, and a serious mean of real-time monitoring for real life protocols and services.

Nevertheless, the application examples show the expected theoretical results. In other words, it is not yet possible to evaluate the impact of different facts such as the parallel schedule management or the synchronizations, which were not taken into account in this theoretical chapter.

Chapter 5

The passive testing tools and the IMS experience

5.1 The tools

5.1.1 Overview

In order to give experimental results and to show the efficiency of the backward approach through the study of common network protocols, a tool was developped.
The tool can be decomposed in two main parts. The first part is the core of the algorithm. It is composed of a set of files written with C language regrouping all the structures and operations on these structures that the main algorithm uses.
The final algorithm uses XML models that are easier to understand and produce than a C code. But the use of XML is also stimulated by the fact that we could then use the algorithm as a network service.
Finally, on the upper level of abstraction, a graphical interface (written with GTK+) for EFSMs is provided. The user can then draw the EFSM and convert the graphical representation into various format including the XML model use by the backward checking algorithm or graphical formats (like ".png" for instance). We expect this interface to improve the confort of use of the users and the quickness of EFSM model production.

5.1.2 The Backward Checking implementation

The implementation of the backward checking approach was developped with the C language. We choose this language because of the performance it can provide. The different structures used by the backward checking technique and the operations on these structures are separated in different files. The reader can find on the figure 5.2 the organization of these files and their dependences. An directed arrow on the graph can be translated as "<arrow start> is used by <arrow end>". The implementation will be refered to as *BC-Tool* is the remaining of the work.

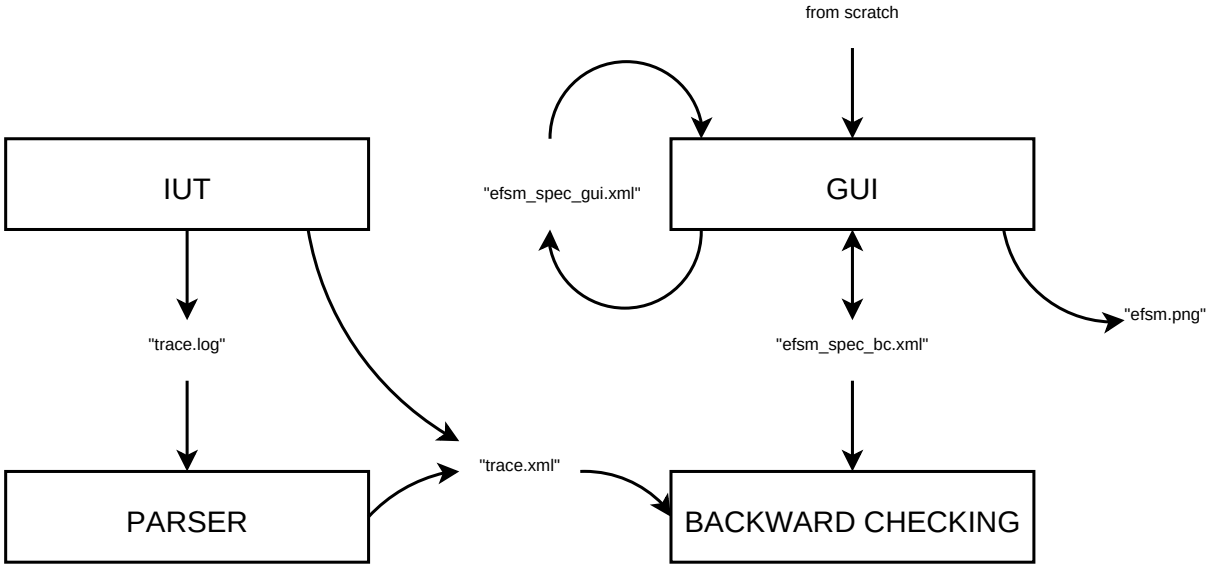

Figure 5.1: Global architecture of the testing tool

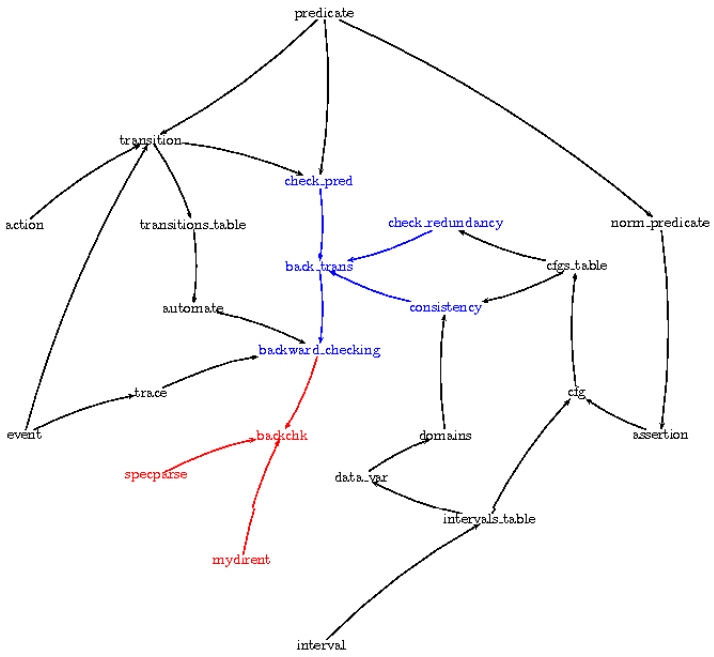

Figure 5.2: Scheme of the C files

The labels in black denotes the basic structures of the algorithm, whereas the blue one are the main algorithms, and the red ones are used to provide a graphical interface. Now

we describe briefly the content of the files :

- action : the action structure is used in the transitions
- predicate : the predicates are boolean formulas hold by transitions
- event : the input and output structures both found in automata transitions and traces
- norm_predicate : a transformation of the predicate structure in order to execute refinement algorithms
- interval : to define into which borders variables are defined
- intervals_table : hash table of intervals, regroup all the variable intervals
- domains : initial definition domains (intervals) of the variables
- assertion : used to record dependencies between the variables, see as a DNF of norm_predicates
- transition : an automata transition as defined in 1.3.2
- transitions_table : a set of transition. Is used to denote all the transitions of a given automata
- automate : EFSM as defined in 1.3.2
- cfg : ECCS (Extended Candidate Configuration Set) as defined in 16
- cfgs_table : set of ECCS. Is used to record all the already seen configurations during the backward checking process
- trace : a sequence of input and output events
- data_var : provide different variable types
- check_pred : functions for the check_pred algorithm
- check_redundancy : functions for the check_redundancy algorithm (that checks ECCS inclusions)
- consistency : functions for the check_consistency algorithm (that checks the consistency of a ECCS)
- back_trans : functions for the back_transition algorithms (track back e transition in the trace or in te past)
- backward_checking : the main algorithm

- specparse : parser for the convertion XML to C code

- mydirent : tool for reading directories. Used for the graphical interface backchk

- backchk : a simple graphical interface (in a console) for the use of BC-Tool.

In a first version of the implementation, only the integer values are taken into account. The SCP protocol (cf. 1.7.1) was succesfully tested with this version. This version can be found at [35].
Nevertheless, an improved version was necessary in order to handle other types of variables - for instance characters strings - that are used by complex communication protocols such as IMS.

5.1.3 A graphical interface for EFSM drawing

The GUI is known as the EFSM-XML project. It uses the GTK2+ libraries. GTK2+ is an object oriented graphical library that depends on GLIB (among others). Thus, the user only need to install GTK2+ (with its dependencies - present by default on most of the recent Linux distributions) to be able to run both EFSM-XML and BC-Tool.
It is a simple GUI for EFSM drawing. Its purpose is to provide a way to build the specifications in a graphical environment. Thus we leave the possibiity to the user to convert his automata into various file format. The available formats to this date are :

- the XML format for backward checking : contains all the needed information in order to give the saved automata as an input to the backward checking implementation

- the proprietary XML format : contains graphical information (i.e. layout information) in addition to the information of the above XML format

- PNG format : give the opportunity to use the automata for other purposes e.g. include them in research papers, or convert it to other graphical format throught the use of an image editor programm (such as The Gimp) etc...

When the application is launched, we obtain a new window (see 5.3). On the left side of the windows we see three icons that correspond respectively, from up to down, to an object (state and/or transition) selector, a state drawer, and a transition drawer, in the center is the drawing area, and on the top is a menu.

To create a state, the user clicks on the state drawer icon, and then he clicks once in the drawing area. To create a transition the user clicks on the transition drawer icon and then once at the desired starting location and once at the desired ending location. If either starting or ending location is in the area of a state, the transition is linked with this state. Note that a state can also be linked a posteriori to a transition edge.
When the selector button is on, the selected objets appear surronded by a red zone as show on the figure 5.8.

The interface propose also a menu to perform essential operations. In the “file” menu the user finds the usual functions such as openning a new or existing file, saving into a file,

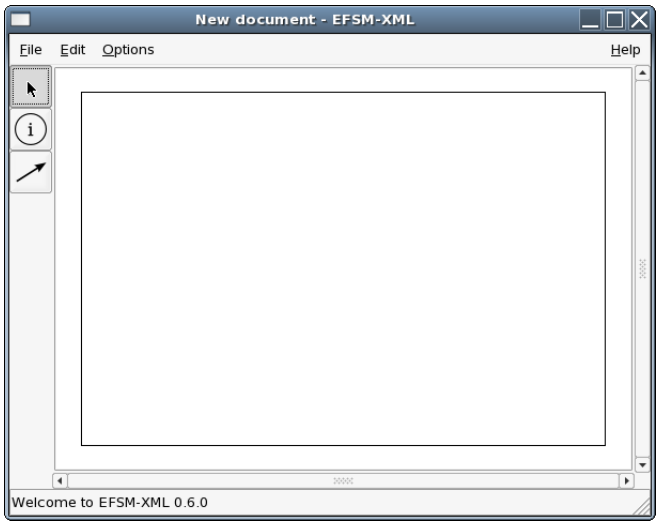

Figure 5.3: The first window of the GUI

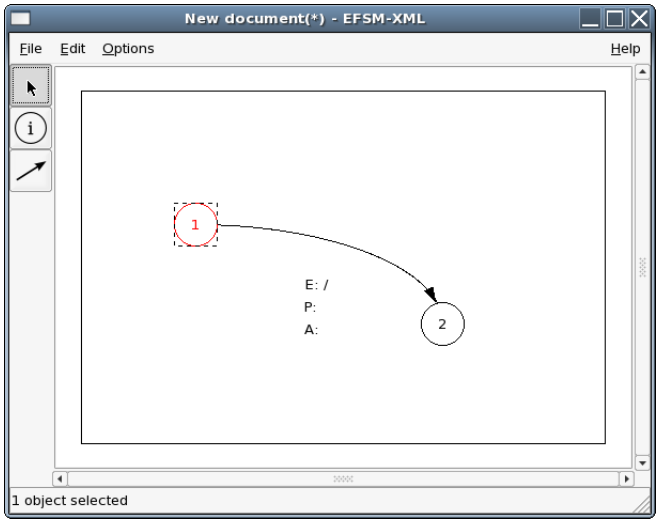

Figure 5.4: Selector example : selecting the state 1

exporting the file for BC-Tool (.xml) or as a picture (.png)...
The 'edit' menu is currently not of great use but is deemed to receive 'cut-copy-paste' functions, such as the very useful 'undo' function.
The 'options' menu is more interesting. The user can check the 'autodelete' box in order to erase - at the saving process - transitions that are not connected to a starting and ending state or states that don't have at least one entering or exiting transition. The user can also choose various automata layouts in the 'geometry' submenu to see automata that where recorded without layout information (which is the case of .xml format for BC-Tool)

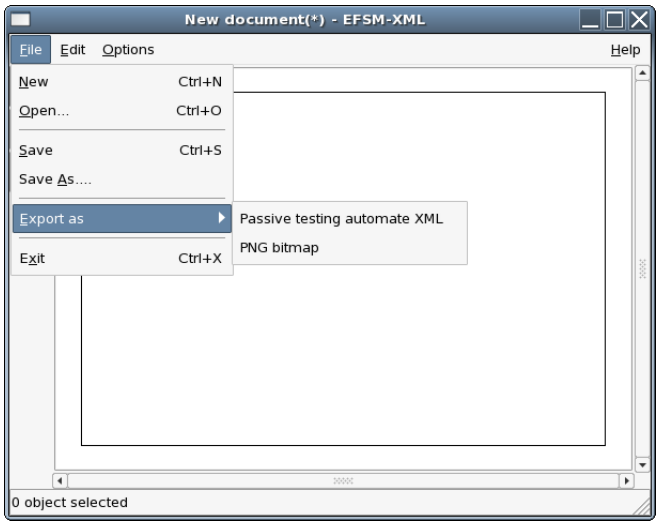

Figure 5.5: File menu

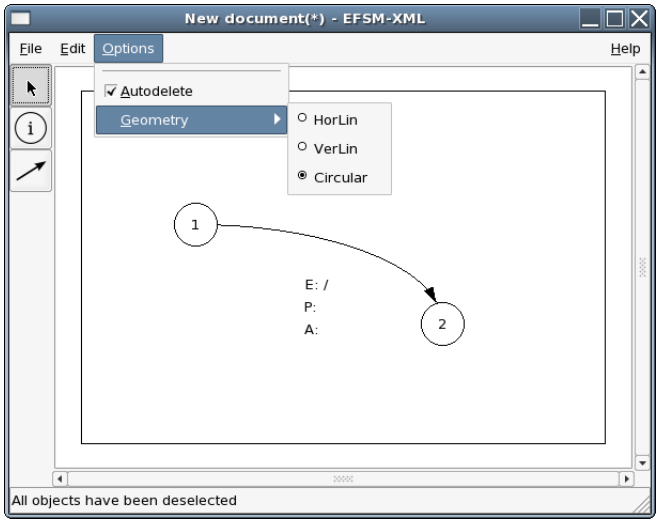

Figure 5.6: Options menu

If the user right-clicks after having selected an object (state or transition), an additional edit window can be asked. For the state, it is then possible to change the name of to define the state as an initial or final state os the EFSM. For a transition it is possible to give the input and output symbols (with or without parameters), to add predicates (simple inequations) and actions (simple updates).

This GUI is in a early version but provide the most essential functions the user will need to draw easily his EFSMs specifications. More functions will be provided in the future in order to facilitate further the usability of the tool. The current version of the GUI is

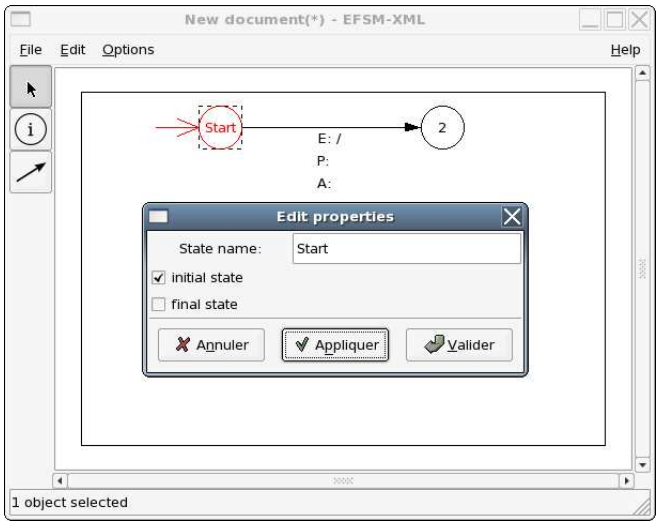

Figure 5.7: Editing a state

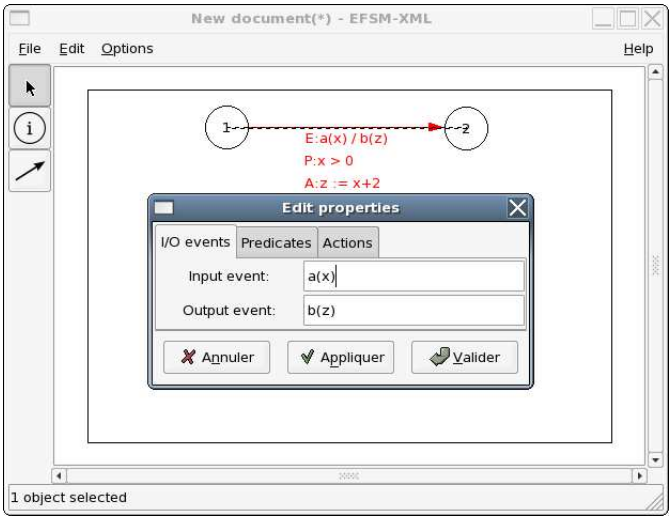

Figure 5.8: Editing a transition

available at [35].

5.2 The IMS

5.2.1 Overview of the IMS specification

The IP Multimedia Subsystem (IMS) is a standardised Next Generation Networking (NGN) architecture for telecom operators that want to provide mobile and fixed multimedia ser-

vices. It uses a Voice-over-IP (VoIP) implementation based on a 3GPP standardised implementation of SIP, and runs over the standard Internet Protocol (IP). Existing phone systems (both packet-switched and circuit-switched) are supported. The aim of IMS is not only to provide new services but all the services, current and future, that the Internet provides. In addition, users have to be able to execute all their services when roaming as well as from their home networks. To achieve these goals, IMS uses open standard IP protocols, defined by the IETF. So, a multimedia session between two IMS users, between an IMS user and a user on the Internet, and between two users on the Internet is established using exactly the same protocol. Moreover, the interfaces for service developers are also based on IP protocols. This is why IMS truly merges the Internet with the cellular world; it uses cellular technologies to provide ubiquitous access and Internet technologies to provide appealing services.
The IMS provides various interesting possibilities such as :

- **Access independence** : IMS will eventually work with any network (fixed, mobile or wireless) with packet-switching functions, such as GPRS, UMTS, CDMA2000, WLAN, WiMAX, DSL, cable, ... Older circuit-switched phone systems (POTS, GSM) are supported through gateways. Open interfaces between control and service layers allow elements and calls/sessions from different access networks to be mixed.

- **Different network architectures** : IMS allows operators and service providers to use different underlying network architectures.

- **Terminal and user mobility** : The mobile network provides terminal mobility (roaming), while user mobility is provided by IMS and SIP.

- **Extensive IP-based services** : IMS should make it easier to offer just about any IP-based service. Examples include voice over IP (VOIP), Push to talk over cellular (POC), multiparty gaming, videoconferencing, Messaging, community services, presence information and content sharing.

The IP Multimedia Core Network Subsystem is a collection of different functions, linked by standardized interfaces. A function is not a node (hardware box) : an implementer is free to combine 2 functions in 1 node, or to split a single function into 2 or more nodes. Each node can also be present multiple times in a network, for load balancing or organizational issues. Several roles of SIP servers or proxies, collectively called CSCF (Call Session Control Function), are used to process SIP signalling packets in the IMS.
The user can connect to an IMS network using various methods, all of which are using the standard Internet Protocol (IP). Direct IMS terminals (mobile phones, PDAs, computers, ...), can register directly into an IMS network, even when they're roaming in another network or country (the visited network). The only requirement is that they can use IPv6 (also IPv4 in 'Early IMS') and are running SIP User Agents. Fixed access (e.g., DSL, cable modems, Ethernet, ...), mobile access (W-CDMA, CDMA2000, GSM, GPRS, ...)

and wireless access (WLAN, WiMAX, ...) are all supported. Other phone systems like the POTS (the old analogue telephones), H.323 and non IMS-compatible VoIP systems are supported through gateways.
The global architecture or the Fokus-Fraunhofer IMS Playground is represented in the Figure 5.9.

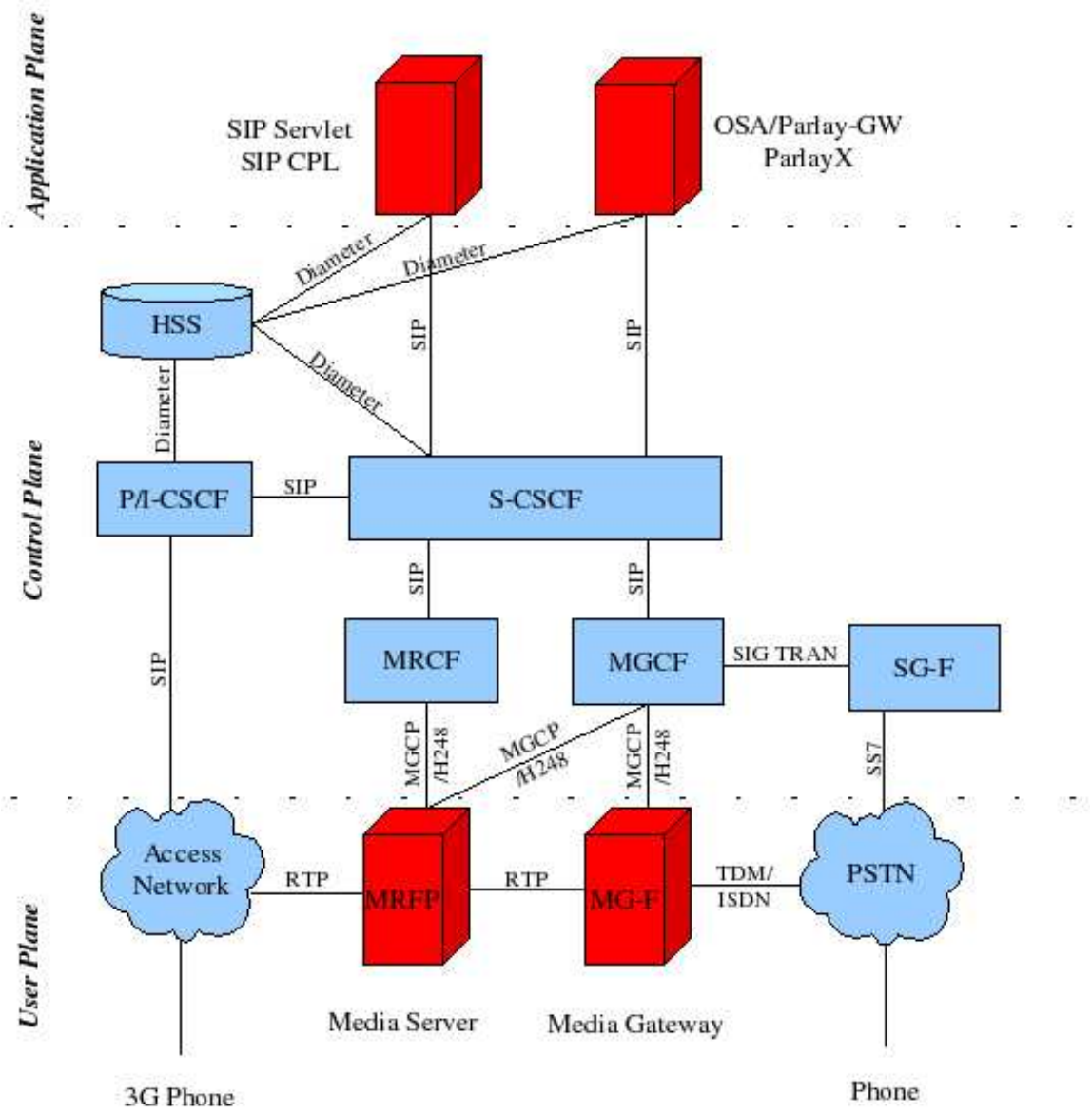

Figure 5.9: The IMS Architecture

In this architecture we decided to focus on the User entity that is to say the connection between the User and the Proxy-CSCF (P-CSCF). We study it further in the next subsec-

tions.

5.2.2 The IMS User Equipment

First of all, we suppose the following parameters were generated using the ISIM application or by derivation of a UICC :

- a private user identity
- at least one public user identity
- a home network domain name

These processes are described respectively in [11] and are then outside of the present work scope.
In the following subsections we describe the different processes that the IMS specification allows. We can divide structurally the subsections in three parts : in the first part a short description of the functionality is given, then we show a graphical view of the EFSM and the formal description of the involved transitions, followed by the original IMS specification in italic (taken from [10]). This way, the reader can understand intuitively the functionality and also see which translations were performed in order to transform the informal specification (written in english) into the formal one.

Initial Registration

The user begin with an initial registration process : "The UE can register a public user identity with its contact address at any time after it has acquired an IP address, discovered a P-CSCF, and established a IP-CAN bearer that can be used for SIP signalling". Then the user can receive four kinds of answers. If it receives a 200 , the registration is done and the user stores several parameters (expiration time, default PUI, list of service routes, security association lifetime). If it receives a 423 the registration is not succesful and can be retried with a longer expiration time. If it receives a 2xx then a subscribe request is sent and a 2xx answer to this subscribe is waited to extract the expiration time of it. If it receives a 401 then there was an authentication problem and then an authentication process has to be performed as described below (Fig. 5.12). The initial registration EFSM is presented in the figure 5.10.

Here is the description of the transitions :
1. - / RegisterReq(Authorization, From, To, Contact, Via, RequestURI, Supported)
P : IPAddr != NULL & PAddr != NULL & IPCan != NULL & PrivateUI != NULL & PublicUI != NULL & HomeNDN != NULL
A : Authorization.username = PrivateUI; From = PublicUI; To = PublicUI; InitAddr = IPaddr+":"+port; Contact.hostport = InitAddr; Via.send-by = InitAddr; Contact.expires

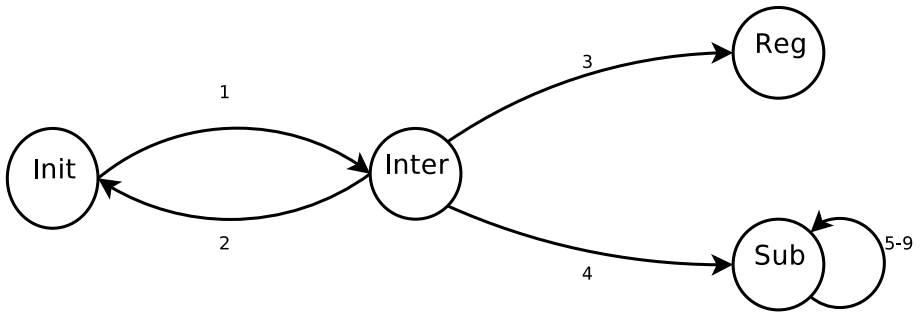

Figure 5.10: The initial registration

= 600000+c1; RequestURI = HomeNDN; Supported = Supported+"path";

2. RegisterRes(status, Min-Expires) / RegisterReq(Authorization, From, To, Contact, Via, RequestURI, Supported)
P : status = 423 & c2 >= 0
A : Contact.expires = RR_Min-expires+c2;

3. RegisterRes(status, To, P-Associated-URI, ServiceRoute) / -
P : status = 200
A : Expires = RR_To.expires; PublicUI = head(RR_P-Associated-URI); Service-Route = rrRR_Service-Route; Sal = max(Sal, RR_Sal+30);

4. RegisterRes(status) / SubscribeReq(RequestURI, From, To, Event, Expires, P-Access-Network-Info)
P : status = 2xx
A : RequestURI = PublicUI; From = PublicUI; To = PublicUI; Event = "reg"; Expires = 600000; Contact = InitAddr;

5. SubscribeRes(status, Expires) / -
P : status = 2xx & Expires > 1200
A : Expires = SR_Expires; TimeoutSubs = Expires-600;

6. SubscribeRes(status, Expires) / -
P : status = 2xx & Expires =< 1200
A : Expires = SR_Expires; TimeoutSubs = Expires/2;

7. - / SubscribeReq(RequestURI, From, To, Event, Expires)
P : TimeoutSubs = 0 & Expires > 1200
A : TimeoutSubs = Expires-600;

8. - / SubscribeReq(RequestURI, From, To, Event, Expires)
P : TimeoutSubs = 0 & Expires =< 1200
A : TimeoutSubs = Expires/2;

9. - / -
P : TimeoutSubs > 0
A : TimeoutSubs = TimeoutSubs-1;

5.1.1.2 Initial registration
The UE can register a public user identity with its contact address at any time after it has aquired an IP address, discovered a P-CSCF, and established an IP-CAN bearer that can be used for SIP signalling. However, the UE shall only initiate a new registration procedure when it has received a final response from the registrar for the ongoing registration, or the previous REGISTER request has timed out.

The UE shall send only the initial REGISTER requests to the port advertised to the UE during the P-CSCF discovery procedure. If the UE does not receive any specific port information during the P-CSCF discovery procedure, the UE shall send the initial REGISTER request to the SIP default port values as specified in RFC 3261 [26].

The UE shall extract or derive a public user identity, the private user identity, and the domain name to be used in the Request-URI in the registration, according to the procedures described in subclause 5.1.1.1A. A public user identity may be input by the end user.

On sending a REGISTER request, the UE shall populate the header fields as follows:
a) an Authorization header, with the username field, set to the value of the private user identity;
b) a From header set to the SIP URI that contains the public user identity to be registered;
c) a To header set to the SIP URI that contains the public user identity to be registered;
d) a Contact header set to include SIP URI(s) containing the IP address of the UE in the hostport parameter or FQDN. If the REGISTER request is protected by a security association, the UE shall also include the protected server port value in the hostport parameter;
e) a Via header set to include the IP address or FQDN of the UE in the sent-by field. If the REGISTER request is protected by a security association, the UE shall also include the protected server port value in the sent-by field
NOTE 1: If the UE specifies its FQDN in the host parameter in the Contact header and in the sent-by field in the Via header, then it has to ensure that the given FQDN will resolve (e.g., by reverse DNS lookup) to the IP address that is bound to the security association.
NOTE 2: The UE associates two ports, a protected client port and a protected server port, with each pair of security association. For details on the selection of the protected port value see 3GPP TS 33.203 [19].
f) an Expires header, or the expires parameter within the Contact header, set to the value

of 600 000 seconds as the value desired for the duration of the registration;
NOTE 3: The registrar (S-CSCF) might decrease the duration of the registration in accordance with network policy. Registration attempts with a registration period of less than a predefined minimum value defined in the registrar will be rejected with a 423 (Interval Too Brief) response.
g) a Request-URI set to the SIP URI of the domain name of the home network;
h) the Security-Client header field set to specify the security mechanism the UE supports, the IPsec layer algorithms the UE supports and the parameters needed for the security association setup. The UE shall support the setup of two pairs of security associations as defined in 3GPP TS 33.203 [19]. The syntax of the parameters needed for the security association setup is specified in Annex H of 3GPP TS 33.203 [19]. The UE shall support the "ipsec-3gpp" security mechanism, as specified in RFC 3329 [48]. The UE shall support the HMAC-MD5-96 (RFC 2403 [20C]) and HMAC-SHA-1-96 (RFC 2404 [20D]) IPsec layer algorithms, and shall announce support for them according to the procedures defined in RFC 3329 [48];
i) the Supported header containing the option tag "path"; and
j) if a security association exists, a P-Access-Network-Info header set as specified for the access network technology (see subclause 7.2A.4).

On receiving the 200 (OK) response to the REGISTER request, the UE shall:
a) store the expiration time of the registration for the public user identities found in the To header value;
b) store as the default public user identity the first URI on the list of URIs present in the P-Associated-URI header;
NOTE 4: The UE can utilize additional URIs contained in the P-Associated-URI header, e.g. for application purposes.
c) treat the identity under registration as a barred public user identity, if it is not included in the P-Associated-URI header;
d) store the list of Service-Route headers contained in the Service-Route header, in order to build a proper preloaded Route header value for new dialogs and standalone transactions; and
e) set the security association lifetime to the longest of either the previously existing security association lifetime (if available), or the lifetime of the just completed registration plus 30 seconds.

When a 401 (Unauthorized) response to a REGISTER is received the UE shall behave as described in subclause 5.1.1.5.1.

On receiving a 423 (Interval Too Brief) too brief response to the REGISTER request, the UE shall:
- send another REGISTER request populating the Expires header or the expires parameter with an expiration timer of at least the value received in the Min-Expires header of the 423 (Interval Too Brief) response.

5.1.1.3 Initial subscription to the registration-state event package
Upon receipt of a 2xx response to the initial registration, the UE shall subscribe to the reg event package for the public user identity registered at the user's registrar (S-CSCF) as described in RFC 3680 [43]. The UE shall use the default public user identiy for subscription to the registration-state event package, if the public user identity that was used for initial registration is a barred public user identity. The UE may use either the default public user identity or the public user identity used for initial registration for the subscription to the registration-state event package, if the initial public user identity that was used for initial registration is not barred.

On sending a SUBSCRIBE request, the UE shall populate the header fields as follows:
a) a Request URI set to the resource to which the UE wants to be subscribed to, i.e. to a SIP URI that contains the public user identity used for subscription;
b) a From header set to a SIP URI that contains the public user identity used for subscription;
c) a To header set to a SIP URI that contains the public user identity used for subscription;
d) an Event header set to the "reg" event package;
e) an Expires header set to 600 000 seconds as the value desired for the duration of the subscription;
f) a P-Access-Network-Info header set as specified for the access network technology (see subclause 7.2A.4); and
g) a Contact header set to contain the same IP address or FQDN, and with the protected server port value as in the initial registration.

Upon receipt of a 2xx response to the SUBSCRIBE request, the UE shall store the information for the established dialog and the expiration time as indicated in the Expires header of the received response. If continued subscription is required, the UE shall automatically refresh the subscription by the reg event package, for a previously registered public user identity, either 600 seconds before the expiration time if the initial subscription was for greater than 1200 seconds, or when half of the time has expired if the initial subscription was for 1200 seconds or less.

5.1.1.4 User-initiated re-registration
The UE can reregister a previously registered public user identity with its contact address at any time. Unless either the user or the application within the UE has determined that a continued registration is not required the UE shall reregister the public user identity either 600 seconds before the expiration time if the initial registration was for greater than 1200 seconds, or when half of the time has expired if the initial registration was for 1200 seconds or less, or when the UE intends to update its capabilities according to RFC 3840 [62].

The UE shall protect the REGISTER request using a security association, see 3GPP TS 33.203 [19], established as a result of an earlier registration, if IK is available. The

UE shall extract or derive a public user identity, the private user identity, and the domain name to be used in the Request-URI in the registration, according to the procedures described in subclause 5.1.1.1A.

On sending a REGISTER request that does not contain a challenge response, the UE shall populate the header fields as follows:
a) an Authorization header, with the username field set to the value of the private user identity;
b) a From header set to the SIP URI that contains the public user identity to be registered;
c) a To header set to the SIP URI that contains the public user identity to be registered;
d) a Contact header set to include SIP URI(s) that contain(s) in the hostport parameter the IP address of the UE or FQDN and protected server port value bound to the security association;
e) a Via header set to include the IP address or FQDN of the UE in the sent-by field and the protected server port value bound to the security association;
NOTE 1: If the UE specifies its FQDN in the host parameter in the Contact header and in the sent-by field in the Via header, then it has to ensure that the given FQDN will resolve (e.g., by reverse DNS lookup) to the IP address that is bound to the security association.
NOTE 2: The UE associates two ports, a protected client port and a protected server port, with each pair of security associations. For details on the selection of the protected port value see 3GPP TS 33.203 [19].
f) an Expires header, or an expires parameter within the Contact header, set to 600 000 seconds as the value desired for the duration of the registration;
NOTE 3: The registrar (S-CSCF) might decrease the duration of the registration in accordance with network policy. Registration attempts with a registration period of less than a predefined minimum value defined in the registrar will be rejected with a 423 (Interval Too Brief) response.
g) a Request-URI set to the SIP URI of the domain name of the home network;
h) a Security-Client header field, set to specify the security mechanism it supports, the IPsec layer algorithms it supports and the new parameter values needed for the setup of two new pairs of security associations. For further details see 3GPP TS 33.203 [19] and RFC 3329 [48];
i) a Security-Verify header that contains the content of the Security-Server header received in the 401 (Unauthorized) response of the last successful authentication;
j) the Supported header containing the option tag "path"; and
k) the P-Access-Network-Info header set as specified for the access network technology (see subclause 7.2A.4).

On receiving the 200 (OK) response to the REGISTER request, the UE shall:
a) store the new expiration time of the registration for this public user identity found in the To header value;
b) store the list of Service-Route headers contained in the Service-Route header, in order to build a proper preloaded Route header value for new dialogs and standalone transactions;

and
NOTE 4: The UE can utilize additional URIs contained in the P-Associated-URI header, e.g. for application purposes.
c) set the security association lifetime to the longest of either the previously existing security association lifetime, or the lifetime of the just completed registration plus 30 seconds.

When a 401 (Unauthorized) response to a REGISTER is received the UE shall behave as described in subclause 5.1.1.5.1.

On receiving a 423 (Interval Too Brief) response to the REGISTER request, the UE shall:
- send another REGISTER request populating the Expires header or the expires parameter with an expiration timer of at least the value received in the Min-Expires header of the 423 (Interval Too Brief) response.

On receiving a 408 (Request Timeout) response or 500 (Server Internal Error) response or 504 (Server Time-Out) response for a reregistration, the UE shall perform the procedures for initial registration as described in subclause 5.1.1.2.

When the timer F expires at the UE, the UE shall:
1) stop processing of all ongoing dialogs and transactions and silently discard them locally; and
2) after releasing all IP-CAN bearers used for the transport of media according to the procedures in subclause 9.2.2, the UE may:
a) select a different P-CSCF address from the list of P-CSCF addresses discovered during the procedures described in subclause 9.2.1;
b) if no response has been received when attempting to contact all P-CSCFs known by the UE, the UE may get a new set of P-CSCF-addresses as described in subclause 9.2.1; and
c) perform the procedures for initial registration as described in subclause 5.1.1.2.
NOTE 5: It is an implementation option whether these actions are also triggered by other means than expiration of timer F, e.g. based on ICMP messages.
After a maximum of 5 consecutive initial registration attempts, the UE shall not automatically attempt any further initial registration for an implementation dependant time of at least 30 minutes.

User-initiated re-registration

The re-registration process should be performed periodically for each registered public user identity. There can be four different answers to this (re-)registration request : a 200 response meaning that the registration is made, a 401 response that force the user to perform an authentication (cf. Fig.5.12), the 408, 500, or 504 response to which the user should react by performing the initial registration operations (cf. Fig.5.10), and a 423 response

meaning that the request failed and has to be retried with a longer expiration time.
The re-registration EFSM is presented in Figure 5.11 , and the details of each transition is given hereafter.

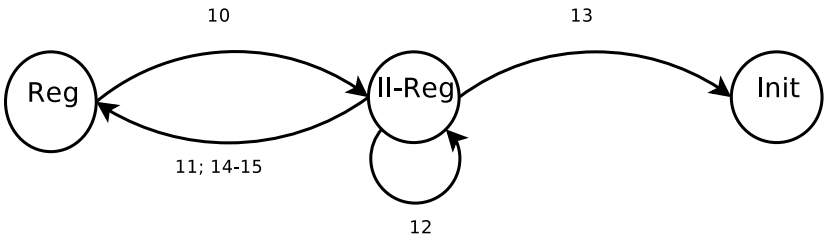

Figure 5.11: The user-initiated re-registration

10. - / RegisterReq(Authorization, From, To, Contact, Via, RequestURI, Security-Client, Security-Verify, Supported, P-Access-Network-Info)
P : TimeoutRegister = 0
A : Authorization.username = PrivateUI; From = PublicUI; To = PublicUI; InitAddr = IPaddr+":"+port; Contact.hostport = InitAddr; Via.send-by = InitAddr; Contact.expires = 600000; RequestURI = HomeNDN; Supported = Supported+"path";

11. RegisterRes(status, To, P-Associated-URI, ServiceRoute) / -
P : status = 200
A : Expires = RR_To.expires; Service-Route = RR_Service-Route; Sal = max(Sal, RR_Sal+30);

12. RegisterRes(status, Min-Expires) / RegisterReq(Authorization, From, To, Contact, Via, RequestURI, Security-Client, Security-Verify, Supported, P-Access-Network-Info)
P : status = 423 & c3 >= 0
A : Contact.expires = RR_Min-Expires+c3

13. RegisterRes(status) / -
P : status = 408 — status = 501 — status = 504

14. - / -
P : TimeoutF = 0 & cmp < 5
A : releaseall; initprocess; cmp = cmp+1

15. - / -
P : TimeoutF = 0 & cmp = 5 & NonAutoReg >= 180
A : releaseall; initprocess; cmp = 0; wait(NonAutoReg);

Authentication

An authentication process can be asked through a 401 response to a registration. In this process the user checks if the response is valid. If it is the case, a new registration request with security associations is sent and then either a 200 response is received and the registration is established with the given security associations either no 200 response is received within the timeout or a 403 response is received meaning that the registration failed and the user performs the initial registration process (cf. Fig.5.10). If the first 401 response is invalid the user sends a registration request without changing the security associations (i.e. sends the same registration request as formerly).
The authentication EFSM is presented in Figure 5.12 , and the details of each transition is given hereafter.

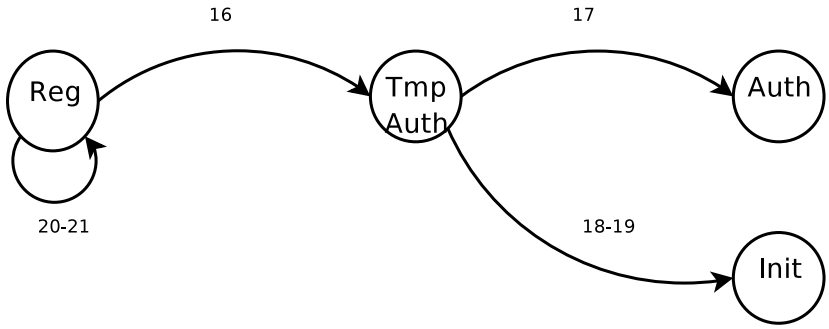

Figure 5.12: Authentication

16. RegisterRes(status, Security-Server) / RegisterReq(Security-Client, IK, Security-Verify, Call-ID)
P : status = 401 & XMAC = AUNT.MAC & AUNT.SQN = ok & Security-Server.preference != NULL & Security-Server.algorithm != NULL & Security-Server.protocol != NULL & Security-Server.mode != NULL & Security-Server.encrypt-algorithm != NULL & Security-Server.spi-c != NULL & Security-Server.spi-s != NULL & Security-Server.port-c != NULL & Security-Server.port-s != NULL
A : RES = compute_RES(); CK = derive_CK(RAND); IK = derive_IK(RAND); temp-Sec-Assoc = Security-Server; temp-SIP-lifetime = reg-await-auth-timer; Authorization = privateUI+challenge(RES); Security-Verify = Security-Server; Call-ID = RR_Call-ID;

17. RegisterRes(status) / -
P : status = 200
A : Sec-Assoc = temp-Sec-Assoc; SIP-lifetime = max(temp-SIP-lifetime, SIP-lifetime+30);

18. RegisterRes(status) / -
P : status = 403
A : Sec-Assoc = NUll;

19. - / -
P : temp-SIP-lifetime = 0
A : Sec-Assoc = NULL
20. RegisterRes(status, AUNT) / RegisterReq()
P : status = 401 & XMAC != AUNT.MAC
A : Authentication = NULL; AUTS = NULL; Sec-Assoc = old-Sec-Assoc; temp-Sec-Assoc = NULL;

21. RegisterRes(status, AUNT) / RegisterReq()
P : status = 401 & SQN >= SQN_min & SQN =< SQN_max
A : Authentication = NULL; Sec-Assoc = old-Sec-Assoc; temp-Sec-Assoc = NULL;

5.1.1.5 Authentication
5.1.1.5.1 General
Authentication is achieved via the registration and re-registration procedures. When the network requires authentication or re-authentication of the UE, the UE will receive a 401 (Unauthorized) response to the REGISTER request.

On receiving a 401 (Unauthorized) response to the REGISTER request, the UE shall:
1) extract the RAND and AUTN parameters;
2) check the validity of a received authentication challenge, as described in 3GPP TS 33.203 [19] i.e. the locally calculated XMAC must match the MAC parameter derived from the AUTN part of the challenge; and the SQN parameter derived from the AUTN part of the challenge must be within the correct range; and
3) check the existence of the Security-Server header as described in RFC 3329 [48]. If the header is not present or it does not contain the parameters required for the setup of the set of security associations (see annex H of 3GPP TS 33.203 [19]), the UE shall abandon the authentication procedure and send a new REGISTER request with a new Call-ID.

In the case that the 401 (Unauthorized) response to the REGISTER request is deemed to be valid the UE shall:
1) calculate the RES parameter and derive the keys CK and IK from RAND as described in 3GPP TS 33.203 [19];
2) set up a temporary set of security associations based on the static list and parameters it received in the 401 (Unauthorized) response and its capabilities sent in the Security-Client header in the REGISTER request. The UE sets up the temporary set of security associations using the most preferred mechanism and algorithm returned by the P-CSCF and supported by the UE and using IK as the shared key. The UE shall use the parameters received in the Security-Server header to setup the temporary set of security associations. The UE shall set a temporary SIP level lifetime for the temporary set of security associations to the value of reg-await-auth timer; and

3) send another REGISTER request using the temporary set of security associations to protect the message. The header fields are populated as defined for the initial request, with the addition that the UE shall include an Authorization header containing the private user identity and the authentication challenge response calculated by the UE using RES and other parameters, as described in RFC 3310 [49]. The UE shall also insert the Security-Client header that is identical to the Security-Client header that was included in the previous REGISTER request (i.e. the REGISTER request that was challenged with the received 401 (Unauthorized) response). The UE shall also insert the Security-Verify header into the request, by mirroring in it the content of the Security-Server header received in the 401 (Unauthorized) response. The UE shall set the Call-ID of the integrity protected REGISTER request which carries the authentication challenge response to the same value as the Call-ID of the 401 (Unauthorized) response which carried the challenge.

On receiving the 200 (OK) response for the integrity protected REGISTER request, the UE shall:
- change the temporary set of security associations to a newly established set of security associations, i.e. set its SIP level lifetime to the longest of either the previously existing set of security associations SIP level lifetime, or the lifetime of the just completed registration plus 30 seconds;and
- use the newly established set of security associations for further messages sent towards the P-CSCF as appropriate.
NOTE 1: In this case, the UE will send requests towards the P-CSCF over the newly established set of security associations. Responses towards the P-CSCF that are sent via UDP will be sent over the newly established set of security associations. Responses towards the P-CSCF that are sent via TCP will be sent over the same set of security associations that the related request was received on.

When the first request or response protected with the newly established set of security associations is received from the P-CSCF, the UE shall delete the old set of security associations and related keys it may have with the P-CSCF after all SIP transactions that use the old set of security associations are completed. Whenever the 200 (OK) response is not received before the temporary SIP level lifetime of the temporary set of security associations expires or a 403 (Forbidden) response is received, the UE shall consider the registration to have failed. The UE shall delete the temporary set of security associations it was trying to establish, and use the old set of security associations. The UE should send an unprotected REGISTER message according to the procedure specified in subclause 5.1.1.2 if the UE considers the old set of security associations to be no longer active at the P-CSCF.

In the case that the 401 (Unauthorized) response is deemed to be invalid then the UE shall behave as defined in subclause 5.1.1.5.3.

5.1.1.5.2 Network-initiated re-authentication
At any time, the UE can receive a NOTIFY request carrying information related to the reg

event package (as described in subclause 5.1.1.3). If:
- the state attribute in any of the <registration> elements is set to "active";
- the value of the <uri> sub-element inside the <contact> sub-element is set to the Contact address that the UE registered; and
- the event attribute of that <contact> sub-element(s) is set to "shortened"; the UE shall:
1) use the expiry attribute within the <contact> sub-element that the UE registered to adjust the expiration time for that public user identity; and
2) start the re-authentication procedures at the appropriate time (as a result of the S-CSCF procedure described in subclause 5.4.1.6) by initiating a reregistration as described in subclause 5.1.1.4, if required.
NOTE: When authenticating a given private user identity, the S-CSCF will only shorten the expiry time within the <contact> sub-element that the UE registered using its private user identity. The <contact> elements for the same public user identitity, if registered by another UE using different private user identities remain unchanged. The UE will not initiate a reregistration procedure, if none of its <contact> sub-elements was modified.

5.1.1.5.3 Abnormal cases
If, in a 401 (Unauthorized) response, either the MAC or SQN is incorrect the UE shall respond with a further REGISTER indicating to the S-CSCF that the challenge has been deemed invalid as follows:
- in the case where the UE deems the MAC parameter to be invalid the subsequent REGISTER request shall contain no authentication challenge response and no AUTS parameter;
- in the case where the UE deems the SQN to be out of range, the subsequent REGISTER request shall contain the AUTS parameter and not an authentication challenge response (see 3GPP TS 33.102 [18]). Whenever the UE detects any of the above cases, the UE shall:
- send the REGISTER request using an existing set of security associations, if available (see 3GPP TS 33.203 [19]);
- populate a new Security-Client header within the REGISTER request, set to specify the security mechanism it supports, the IPsec layer algorithms it supports and the parameters needed for the new security association setup; and
- not create a temporary set of security associations. A UE shall only respond to two consecutive invalid challenges. The UE may attempt to register with the network again after an implementation specific time.

User-initiated deregistration

The user can deregister one or more registered public user identities by sending the corresponding request, after releasing all the dialogs to these identities. The only awaited answer is the 200.
22. - / RegisterReq()
P :
A : release_dial(PublicUI); Authorization.username = PrivateUI; From = PublicUI; To =

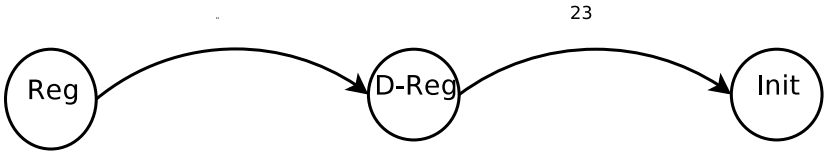

Figure 5.13: User-initiated deregistration

PublicUI; Contact = ; Via.sent-by = IPAddr+port; Contact.expires = 0; Request-URI = HomeNDN; P-Access-Network-Info = ; 23. RegisterRes(status) / -
P : status = 200
A : delete_all()

5.1.1.6 User-initiated deregistration
The UE can deregister a public user identity that it has previously registered with its contact address at any time.
The UE shall integrity protect the REGISTER request using a security association, see 3GPP TS 33.203 [19], established as a result of an earlier registration, if one is available.
The UE shall extract or derive a public user identity, the private user identity, and the domain name to be used in the Request-URI in the registration, according to the procedures described in subclause 5.1.1.1A.
Prior to sending a REGISTER request for deregistration, the UE shall release all dialogs related to the public user identity that is going to be deregistered or to one of the implicitly registered public user identities.

On sending a REGISTER request, the UE shall populate the header fields as follows:
a) an Authorization header, with the username field, set to the value of the private user identity;
b) a From header set to the SIP URI that contains the public user identity to be deregistered;
c) a To header set to the SIP URI that contains the public user identity to be deregistered;
d) a Contact header set to either the value of "" or SIP URI(s) that contain(s) in the hostport parameter the IP address of the UE or FQDN and the protected server port value bound to the security association;*
e) a Via header set to include the IP address or FQDN of the UE in the sent-by field and the protected server port value bound to the security association;
NOTE 1: If the UE specifies its FQDN in the host parameter in the Contact header and in the sent-by field in the Via header, then it has to ensure that the given FQDN will resolve (e.g., by reverse DNS lookup) to the IP address that is bound to the security association.
f) an Expires header, or the expires parameter of the Contact header, set to the value of zero, appropriate to the deregistration requirements of the user;

g) a Request-URI set to the SIP URI of the domain name of the home network; and
h) a P-Access-Network-Info header set as specified for the access network technology (see subclause 7.2A.4).

On receiving the 200 (OK) response to the REGISTER request, the UE shall remove all registration details relating to this public user identity.
If there are no more public user identities registered, the UE shall delete the security associations and related keys it may have towards the IM CN subsystem.
If all public user identities are deregistered and the security association is removed, then the UE shall consider subscription to the reg event package cancelled (i.e. as if the UE had sent a SUBSCRIBE request with an Expires header containing a value of zero).
NOTE: When the UE has received the 200 (OK) response for the REGISTER request of the only public user identity currently registered with its associated set of implicitly registered public user identities (i.e. no other is registered), the UE removes the security association established between the P-CSCF and the UE. Therefore further SIP signalling (e.g. the NOTIFY request containing the deregistration event) will not reach the UE.

Network-initiated deregistration

The network can ask a deregistration by sending a notification message during the subscription. The user has to do few operations and goes back to the initial state.

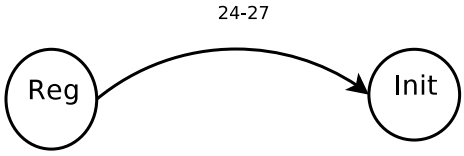

Figure 5.14: Network-initiated deregistration

24. NotifyReq() / -
P : (registration.state = terminated & registration.event = rejected) — (registration.state = activated & contact.state = terminated & registration.event = rejected)
A : delete_reg(PublicUI); release_dial(PublicUI);

25. NotifyReq() / -
P : (registration.state = terminated & registration.event = rejected) — (registration.state = activated & contact.state = terminated & registration.event = rejected) & List_PublicUI = NULL
A : delete_reg(PublicUI); release_dial(PublicUI); delete_Sec-Assoc;

26. NotifyReq() / -
P : (registration.state = terminated & registration.event = deactivated) & (registration.state = activated & contact.state = terminated & registration.event = deactivated)

A : delete_reg(PublicUI);

27. NotifyReq() / -
P : (registration.state = terminated & registration.event = deactivated) & (registration.state = activated & contact.state = terminated & registration.event = deactivated) & List_PublicUI = NULL
A : delete_reg(PublicUI); delete_Sec-Assoc;

5.1.1.7 Network-initiated deregistration
Upon receipt of a NOTIFY request on the dialog which was generated during subscription to the reg event package as described in subclause 5.1.1.3, including one or more <registration> element(s) which were registered by this UE with:
- the state attribute set to "terminated" and the event attribute set to "rejected" or "deactivated"; or
- the state attribute set to "active" and the state attribute within the <contact> element belonging to this UE set to "terminated", and associated event attribute element to "rejected" or "deactivated";
the UE shall remove all registration details relating to these public user identities. In case of a "deactivated" event attribute, the UE shall start the initial registration procedure as described in subclause 5.1.1.2. In case of a "rejected" event attribute, the UE shall release all dialogs related to those public user identities.

Upon receipt of a NOTIFY request, the UE shall delete the security associations towards the P-CSCF either:
- if all <registration> element(s) having their state attribute set to "terminated" (i.e. all public user identities are deregistered) and the Subscription-State header contains the value of "terminated"; or
- if each <registration> element that was registered by this UE has either the state attribute set to "terminated", or the state attribute set to "active" and the state attribute within the <contact> element belonging to this UE set to "terminated".

The UE shall delete these security associations towards the P-CSCF after the server transaction (as defined in RFC 3261 [26]) pertaining to the received NOTIFY request terminates.
NOTE 1: Deleting a security association is an internal procedure of the UE and does not involve any SIP procedures.
NOTE 2: If all the public user identities or contact addresses registered by this UE are deregistered and the security association is removed, then the UE considers the subscription to the reg event package terminated (i.e. as if the UE had sent a SUBSCRIBE request with an Expires header containing a value of zero, or a NOTIFY request was received with Subscription-State header containing the value of "terminated").
NOTE 3: When the P-CSCF has removed the security association established between the

P-CSCF and the UE, further SIP signalling (e.g. the NOTIFY containing the deregistration event) will not reach the UE.

Subscription and Notification

A subscribe response to a subscribe request can be either a 2xx response and the dialog is maintained or a 503 response and an other subscribe request is sent by the user after a given time (contained in the response). Notify requests are used by the network to inform the user of the identity status (active or terminated).

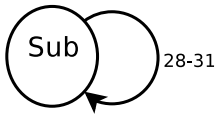

Figure 5.15: The subscribe and notification messages

28. SubscribeRes(status, Call-ID, To, From) / -
P : status = 2xx
A : dial = Call-ID + To + From

29. NotifyReq() / -
P : registration.state = active
A : registered_PublicUI = PublicUI + registered_PublicUI;

30. NotifyReq() / -
P : registration.state = terminated
A : registered_PublicUI = registered_PublicUI - PublicUI; deregistered_PublicUI = PublicUI + deregistered_PublicUI;

31. SubscribeRes(status) / -
P : status = 503 & Retry-After != NULL
A : wait(Retry-After);

5.1.2 Subscription and notification
5.1.2.1 Notification about multiple registered public user identities
Upon receipt of a 2xx response to the SUBSCRIBE request the UE shall maintain the generated dialog (identified by the values of the Call-ID, To and From headers).

Upon receipt of a NOTIFY request on the dialog which was generated during subscription to the reg event package the UE shall perform the following actions:
- if a state attribute "active", i.e. registered is received for one or more public user identities, the UE shall store the indicated public user identities as registered;

- if a state attribute "terminated", i.e. deregistered is received for one or more public user identities, the UE shall store the indicated public user identities as deregistered.
NOTE: There may be public user identities which are automatically registered within the registrar (S-CSCF) of the user upon registration of one public user identity. Usually these automatically or implicitly registered public user identities belong to the same service profile of the user and they might not be available within the UE. The implicitly registered public user identities may also belong to different service profiles. The here-described procedures provide a different mechanism (to the 200 (OK) response to the REGISTER request) to inform the UE about these automatically registered public user identities.

5.1.2.2 General SUBSCRIBE requirements
If the UA receives a 503 (Service Unavailable) response to an initial SUBSCRIBE request containing a Retry-After header, then the UE shall not automatically reattempt the request until after the period indicated by the Retry-After header contents.

5.1.2A Generic procedures applicable to all methods excluding the REGISTER method
5.1.2A.1 Mobile-originating case
The procedures of this subclause are general to all requests and responses, except those for the REGISTER method.

When the UE sends any request, the UE shall:
- include the protected server port in the Via header entry relating to the UE; and
- include the protected server port in any Contact header that is otherwise included.

The UE shall discard any SIP response that is not integrity protected and is received from the P-CSCF outside of the registration and authentication procedures. The requirements on the UE within the registration and authentication procedures are defined in subclause 5.1.1.

In accordance with RFC 3325 [34] the UE may insert a P-Preferred-Identity header in any initial request for a dialog or request for a standalone transaction as a hint for creation of an asserted identity within the IM CN subsystem. The UE may include any of the following in the P-Preferred-Identity header:
- a public user identity which has been registered by the user;
- a public user identity returned in a registration-state event package of a NOTIFY request as a result of an implict registration that was not subsequently deregistered or has expired; or
- any other public user identity which the user has assumed by mechanisms outside the scope of this specification to have a current registration.
NOTE 1: The temporary public user identity specified in subclause 5.1.1.1 is not a public user identity suitable for use in the P-Preferred-Identity header.
NOTE 2: Procedures in the network require international public telecommunication numbers when telephone numbers are used in P-Preferred-Identity header.

NOTE 3: A number of headers can reveal information about the identity of the user. Where privacy is required, implementers should also give consideration to other headers that can reveal identity information.
RFC 3323 [33] subclause 4.1 gives considerations relating to a number of headers.

Where privacy is required, in any initial request for a dialog or request for a standalone transaction, the UE shall set the From header to "Anonymous".
NOTE 4: The contents of the From header should not be relied upon to be modified by the network based on any privacy specified by the user either within the UE indication of privacy or by network subscription or network policy. Therefore the user should include the value "Anonymous" whenever privacy is explicitly required. As the user may well have privacy requirements, terminal manufacturers should not automatically derive and include values in this header from the public user identity or other values stored in or derived from the UICC. Where the user has not expressed a preference in the configuration of the terminal implementation, the implementation should assume that privacy is required. Users that require to identify themselves, and are making calls to SIP destinations beyond the IM CN subsystem, where the destination does not implement RFC 3325 [34], will need to include a value in the From header other than Anonymous.

The UE can indicate privacy of the P-Asserted-Identity that will be generated by the P-CSCF in accordance with RFC 3323 [33], and the additional requirements contained within RFC 3325 [34].

The UE shall insert a P-Access-Network-Info header into any request for a dialog, any subsequent request (except ACK requests and CANCEL requests) or response (except CANCEL responses) within a dialog or any request for a standalone method. The UE shall populate the P-Access-Network-Info header with the current point of attachment to the IP-CAN as specified for the access network technology (see subclause 7.2A.4).
NOTE 5: During the dialog, the points of attachment to the IP-CAN of the UE may change (e.g. UE connects to different cells). The UE will populate the P-Access-Network-Info header in any request or response within a dialog with the current point of attachment to the IP-CAN (e.g. the current cell information).

The UE shall build a proper preloaded Route header value for all new dialogs and standalone transactions. The UE shall build a list of Route header values made out of, in this order, the P-CSCF URI (containing the IP address or the FQDN learnt through the P-CSCF discovery procedures, and the protected server port learnt during the registration procedure), and the values received in the Service-Route header saved from the 200 (OK) response to the last registration or re-registration.

When a SIP transaction times out, i.e. timer B, timer F or timer H expires at the UE, the UE may behave as if timer F expired, as described in subclause 5.1.1.4.
NOTE 6: It is an implementation option whether these actions are also triggered by other

means.

5.1.2A.2 Mobile-terminating case
The procedures of this subclause are general to all requests and responses, except those for the REGISTER method.

When the UE sends any response, the UE shall:
- include the protected server port in any Contact header that is otherwise included.

The UE shall discard any SIP request that is not integrity protected and is received from the P-CSCF outside of the registration and authentication procedures. The requirements on the UE within the registration and authentication procedures are defined in subclause 5.1.1.

The UE can indicate privacy of the P-Asserted-Identity that will be generated by the P-CSCF in accordance with RFC 3323 [33], and the additional requirements contained within RFC 3325 [34].
NOTE 1: In the mobile-terminating case, this version of the document makes no provision for the UE to provide an P-Preferred-Identity in the form of a hint.
NOTE 2: A number of headers can reveal information about the identity of the user. Where, privacy is required, implementers should also give consideration to other headers that can reveal identity information. RFC 3323 [33] subclause 4.1 gives considerations relating to a number of headers.

The UE shall insert a P-Access-Network-Info header into any response to a request for a dialog, any subsequent request (except CANCEL requests) or response (except CANCEL responses) within a dialog or any response to a standalone method. The UE shall populate the P-Access-Network-Info header with its current point of attachment to the IP-CAN as specified for the access network technology (see subclause 7.2A.4).

Call Invitation

When registered, an user can try to connect to an other user via the invite process. The user sends an invite request with or without preconditions. The reception of a 503 response makes the user to wait a given time (contained in the response) before retrying an invite request. Otherwise, it answers the 200 response (invite is successful) and 380 response (for emergency call) by the sending of an Ack message. Then the user ends the invite process when it wants by sending a Bye message.
32. - / InviteReq()
P :
A :

33. InviteRes(status) / AckReq()

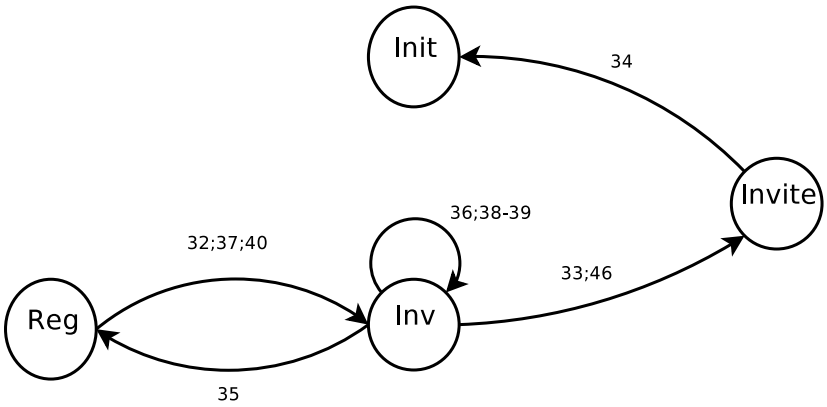

Figure 5.16: User-initiated invite

P : status = 200
A :

34. - / Bye()
P :
A :

35. InviteRes(status, Retry-After) / -
P : status = 503
A : wait(Retry-After);

36. InviteRes(status) / InviteReq(SDP)
P : status = 488
A :

37. - / InviteReq(Supported, Require)
P :
A :

38. InviteRes(status, Unsupported) / InviteReq(Request-URI, Supported)
P : status = 420
A : Supported =+ precondition; inactive_SDP(6.1);

39. InviteRes(SDP) / re-InviteReq(From, To, Call-ID)
P :
A : resource_resa(); active_SDP(6.1)

40. - / InviteReq(Supported)

P :
A : Supported =+ precondition;

46. InviteRes(status, bodyXML) / Ack()
P : status = 380 & InviteRes.bodyXML.altern_service.type = emergency
A :

5.1.3 Call initiation - mobile originating case
5.1.3.1 Initial INVITE request
5.1.3.1.1 General
Subclause 5.1.3.1 describe the procedures when the initial INVITE is sent by the originating UE. The default behaviour using the "integration of resource management and SIP" extension (herafter in this subclause known as the SIP precondition mechanism and defined in RFC 3312 [30] as updated by RFC 4032 [64], and with the request for such a mechanism known as a precondition) is described in subclause 5.1.3.1.2.1. Session without preconditions may be initiated:
- when the remote node does not support the precondition mechanism, as discovered in subclause 5.1.3.1.3; or
- when the specific service does not require the precondition mechanism, as described in subclause 5.1.3.1.4.
Editor's Note: The detailed criteria when to use the non-precondition procedures / resource reservation should be either derived from stage 2 or should be included as a reference to 3GPP TS 23.228.

The UE may indicate that proxies should not fork the INVITE request by including a "no-fork" directive within the Request-Disposition header in the initial INVITE request as described in RFC 3841 [56B].
NOTE 1: Table A.4 specifies that UE support of forking is required in accordance with RFC 3261 [26]. The UE can accept or reject any of the forked responses, for example, if the UE is capable of supporting a limited number of simultaneous transactions or early dialogs.

When a final answer is received for one of the early dialogues, the UE proceeds to set up the SIP session. The UE shall not progress any remaining early dialogues to established dialogs. Therefore, upon the reception of a subsequent final 200 (OK) response for an INVITE request (e.g., due to forking), the UE shall:
1) acknowledge the response with an ACK request; and
2) send a BYE request to this dialog in order to terminate it.

If the UA receives a 503 (Service Unavailable) response to an initial INVITE request containing a Retry-After header, then the UE shall not automatically reattempt the request until after the period indicated by the Retry-After header contents.

If the UE receives a 488 (Not Acceptable Here) response to an initial INVITE request, the UE should send a new INVITE request containing SDP according to the procedures defined in subclause 6.1.
NOTE 2: An example of where a new request would not be sent is where knowledge exists within the UE, or interaction occurs with the user, such that it is known that the resulting SDP would describe a session that did not meet the user requirements.

5.1.3.1.2 "Integration of resource management and SIP" required by originating UE
Upon generating an initial INVITE request using preconditions, the UE shall:
- indicate the support for reliable provisional responses and specify it using the Supported header mechanism;
- indicate the requirement for the preconditions mechanism and specify it using the Require header mechanism.

When the initial INVITE has been created and forwarded the forthcoming procedures are identical to the procedures described in subclause 5.1.3.1.1.

If the UE receives a 420 (Bad Extension) response to an initial INVITE request with "precondition" option-tag in the Unsupported header field, the UE shall either:
a) abort the session attempt and shall not resend this INVITE request with "precondition" option-tag in the Require header, or
b) try to complete the session by relaxing the requirement on the usage of the precondition mechanism and proceed with the procedures described in subclause 5.1.3.1.3 and subclause 6.1.

5.1.3.1.3 "Integration of resource management and SIP" required by originating UE and not supported by terminating UE
This procedure is initiated upon the reception of a 420 (Bad Extension) response to an initial INVITE request, the response containing the "precondition" option-tag in the Unsupported header field value.

When creating the new INVITE request the UE shall:
1) populate the Request-URI as per the initial INVITE request;
2) include the "precondition" option-tag in the Supported header;
3) set each of the media streams in inactive mode in SDP as described in subclause 6.1 in this specification in order to prevent the terminating end to send media whereas the resource reservation is not done at the originating side; and
4) forward the INVITE request as per regular procedures.

Upon receiving a provisional response or final response containing the remote SDP, the UE shall:
1) if required by the regular SIP procedures defined in RFC 3261 [26] and RFC 3262 [27], acknowledge, the SIP response; and

2) initiate the regular resource reservation mechanism.

When the above INVITE transaction is successfully completed, and the local resource reservation procedure is complete, the UE shall create and forward a re-INVITE request including:
1) the From, To, Call-ID headers as per a re-INVITE request; and
2) SDP in which the media streams previously set in inactive mode are set to active (sendrecv, sendonly or recvonly) mode, according to the procedures described in subclause 6.1 in this specification.

When the reINVITE request has been created and forwarded the forthcoming procedures are identical to the procedures described in subclause 5.1.3.1.1.

5.1.3.1.4 "Integration of resource management and SIP" not required by originating UE This procedure is initiated when the precondition mechanism is not required for a session by the origination UE. Upon generating the initial INVITE request the UE may indicate the support of the precondition mechanism by including the "precondition" option-tag in the Supported header.

When the initial INVITE request has been created and forwarded the forthcoming procedures are identical to the procedures described in subclause 5.1.3.1.1.

5.1.6 Emergency service
A UE shall not attempt to establish an emergency session via the IM CN Subsystem when the UE can detect that the number dialled is an emergency number. The UE shall use the CS domain as described in 3GPP TS 24.008 [8].

In the event the UE receives a 380 (Alternative Service) response to an INVITE request the response containing a XML body that includes an <alternative service> element with the <type> child element set to "emergency", the UE shall automatically:
- send an ACK request to the P-CSCF as per normal SIP procedures;
- attempt an emergency call setup according to the procedures described in 3GPP TS 24.008 [8].

The UE may also provide an indication to the user based on the text string contained in the <reason> element. As a consequence of this, a UE operating in MS operation mode C cannot perform emergency calls.

Call initiation - mobile terminating

The user can also receive an invite. In this case it reacts depending on few conditions, but can basically agree the invite (with or without sending provisional responses), or ask the inviter for more requirements.

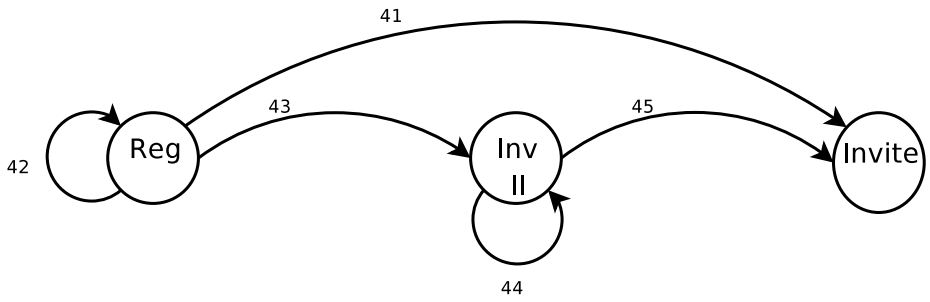

Figure 5.17: Network-initiated invite

41. InviteReq(Require) / InviteRes(status, Require)
P : resource_management = TRUE & InviteReq.Require = *+precondition
A : resource_resa(); status = 200;

42. InviteReq(Require) / InviteRes(status, Require)
P : resource_management = TRUE & InviteReq.Require != *+precondition & Req_precond = TRUE
A : status = 421;

43. InviteReq(Require) / -
P : resource_management = TRUE & InviteReq.Require != *+precondition & Req_precond != TRUE
A :

44. - / InviteRes(status)
P :
A : status = 183;

45. - / InviteReq(status)
P : call_accepted = TRUE
A : status = 200;

5.1.4 Call initiation - mobile terminating case
5.1.4.1 Initial INVITE request
5.1.4.1.1 General
The handling of incoming initial INVITE requests at the terminating UE is mainly dependent on the following conditions:

- the specific service requirements for "integration of resource management and SIP" extension (herafter in this subclause known as the precondition mechanism and defined in RFC 3312 [30] as updated by RFC 4032 [64], and with the request for such a mechanism known as a precondition); and
- the UEs configuration for the case when the specific service does not require the precondition mechanism.
Editor's Note: The detailed criteria when to use the non-precondition procedures / resource reservation should be either derived from stage 2 or should be included as a reference to 3GPP TS 23.228.

If an initial INVITE request is received the terminating UE shall check whether the terminating UE requires integration of resource management either due to the requested service or due to local configuration. If resource management is required at the terminating UE and:
a) the received INVITE request includes the "precondition" option-tag in the Require header, the terminating UE shall perform the actions as described in subclause 5.1.4.1.2;
b) the received INVITE request does not include the "precondition" option-tag in the Require header and the terminating UE, based on local configuration, requires the usage of the precondition mechanism in this case, the terminating UE shall perform the actions as described in subclause 5.1.4.1.3; or
NOTE 1: To configure the terminal to only accept calls with precondition is a rare case that will cause release of incoming calls from UE that will not indicate that precondition is required.
c) the received INVITE request does not include the "precondition" option-tag in the Require header and the terminating UE, based on local configuration, does not require the usage of preconditions in this case, the terminating UE shall perform the actions as described in subclause 5.1.4.1.4.

If resource management is not required by the terminating UE and:
a) the received INVITE request includes the "precondition" option-tag in the Require header, the terminating UE shall perform the actions as described in subclause 5.1.4.1.2, i.e. the terminating UE shall use the precondition mechanism in order to fulfil the requirement of the originating UE; or
b) the received INVITE request does not include the "precondition" option-tag in the Require header, the terminating UE shall perform the actions as described in subclause 5.1.4.1.4.
NOTE 2: Table A.4 specifies that UE support of forking is required in accordance with RFC 3261 [26].
Editor's Note: The above note needs further investigation.

5.1.4.1.2 "Integration of resource management and SIP" required by terminating UE and used by originating UE
Upon generating the first response to the initial INVITE request that indicated the "precondition" option-tag in the Require header, the UE shall indicate the requirement for reliable

provisional responses and specify it using the Require header mechanism.The UE shall send the 200 (OK) response to the initial INVITE request only after the local resource reservation has been completed and the call is accepted by the termination user.

5.1.4.1.3 "Integration of resource management and SIP" not used by originating UE
Upon receiving an initial INVITE request without the "precondition" option-tag in the Require header, and the precondition mechanism is required by the terminating UE, the terminating UE shall generate a 421 (Extension Required) response indicating the required extension in the Require header field value.

5.1.4.1.4 "Integration of resource management and SIP" not required by terminating UE and not used by originating UE
Upon receiving an initial INVITE request without containing the "precondition" option-tag in the Require header, if the terminating UE does not use the precondition mechanism, the UE shall:
1) send none or more provisional response(s) (eg. 183 Session Progress); and
2) send a 200 (OK) response, when the resources are available and the call has been accepted by the terminating user.

5.2.3 The entire EFSM specification for the IMS User Equipment

As a result to the gathering of the previous EFSMs we obtain the global process of an IMS User Equipment according to [10]. This is shown in the figure 5.18.

Some remarks

From the observation of the EFSM resulting of the process of translation from english specification to formal one, several remarks must be noted :

- The EFSM specification seems to be uncomplete. In particular, the states *Auth* and *Sub* are wells without issue. That is a problem that we propose to resolve by making a fusion between these states with the *Reg* state, and by adding two lists : for authorised and subscribed PUI. It is also incomplete in the way that a lot of message status are not handled in the document, and then are left to a very open interpretation.

- The translation is made more difficult by unconsistencies in the english specification. An example amongst others is the following : the standard document [9] states that the BYE message is optional for sending/reception in an UE whereas in the description of the basic functions of a UE this type of message is considered to be mandatory (otherwise there is no way to bring the protocol in its initial state). Thus, the EFSM specification is not a minimal - in the meaning that it contains all and only what is mandatory - but some wider specification.

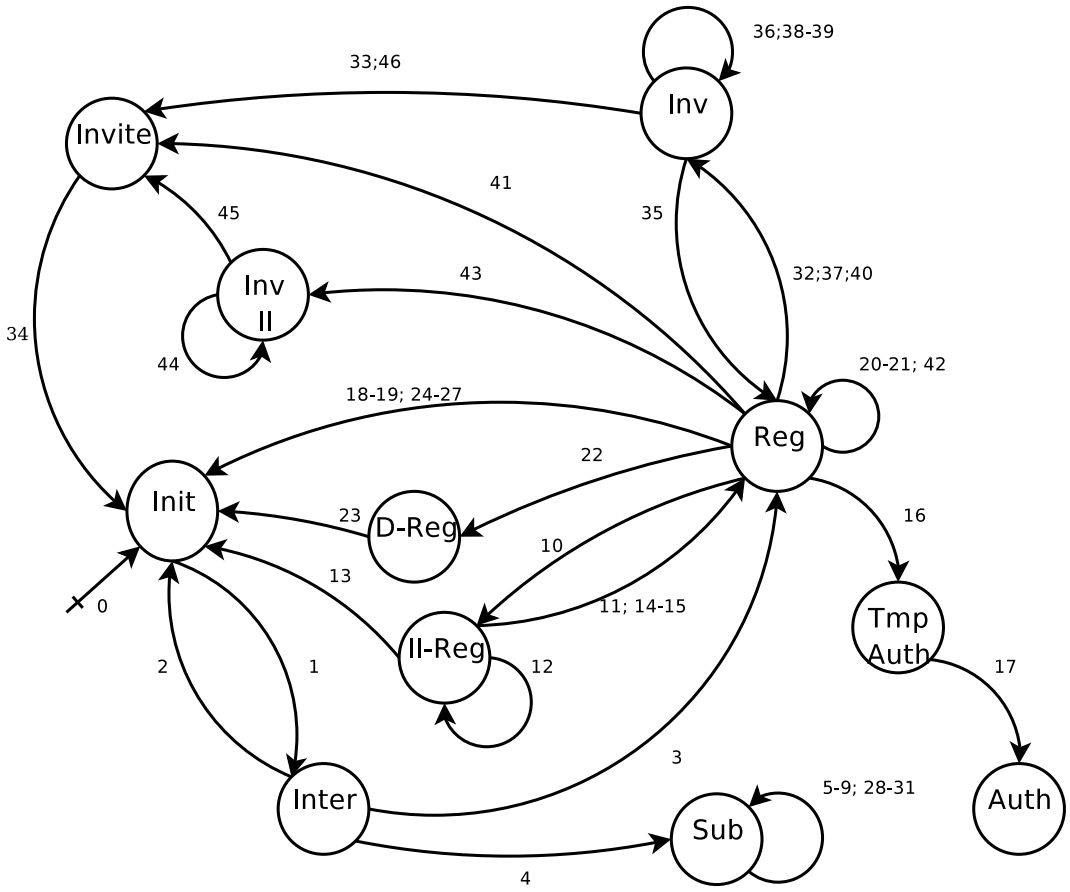

Figure 5.18: The IMS User Equipment EFSM specification

- The internal functions are supposed to process correctly. Anyway, we have no mean to see if it really does or not by only using the event trace.
- Some constants are added and initialised in the transition 0 thanks to the following predicate : c1≥0; c2≥0; c3≥0; cmp=0

5.2.4 The event traces - the final experiment

A pack of traces from a real IMS architecture deployed at FOKUS-Fraunhofer were recorded in order to inject them in the BC-Tool (cf. Sec.5.1.2) with the above (Fig.5.18) specification. The complete traces are given in the Appendix, but we make a brief description of them in the following paragraphs.
We recorded nine event traces that reflect the normal message exchanges in an IMS User to P-CSCF SIP link. These traces are of various length — between 17 and 61 — and propose various scenarii from a simple message exchange to a sequence of call invitations.
In the study of the traces we also see sereval message types that are not recognized by the

specification. That was expected as the EFSM specification is as minimal as possible. This point is resolved by assuming that these message exchange are included in some optional behaviours that will not affect the global variable but only variables dedicated for the considered complementary services. Thus, these messages can be momentaneously ignored.
The experiment has shown that the messages were partly uncomplete : several information fields were missing. One could say that the traces were then faulty. But, in order to discover other faults, the missing field of these messages were filled in with default values. The experiment was repeated afterwards, successfully.

Let's see a concrete example. The complete original trace can be found in the appendix. The following trace is the result of the filtering of the original trace, eleminating the unuseful information.

- / RegisterReq (Authorization.username="usim010000@umts-at-fokus.de", From= "sip:-subs010000@umts-at-fokus.de", To="sip:subs010000@umts-at-fokus.de", Contact.hosport ="193.175.133.125:5060", Via_Sent-by="193.175.133.125:5060", Contact.expires=1000000, RequestURI="sip:umts-at-fokus.de", Supported="path")
RegisterRes (status=401, Call-ID="968@193.175.133.125", Security-Server="NULL") / RegisterReq (Security-Client="NULL", IK="NULL", Security-Verify="NULL", Call-ID= "968@193.175.133.125")
RegisterRes(status=200) / -
RegisterRes(status=401, Call-ID="2315@193.175.133.125", Security-Server="NULL") / RegisterReq (Security-Client="NULL", IK="NULL", Security-Verify="NULL", Call-ID= "2315@193.175.133.125")
RegisterRes(status=200) / -
- / InviteReq()
InviteRes(status=100) / -
- / InviteReq()
InviteRes(status=200) / Ack()
- / Bye()
- / Bye()
Bye(status=200) / -
Bye(status=200) / -

In fact the trace which is checked is only the one presented below since the other events are not recognized by the specification and are considered to be optional services without influence on the main protocol process :

- / RegisterReq (Authorization.username="usim010000@umts-at-fokus.de", From= "sip:-subs010000@umts-at-fokus.de", To="sip:subs010000@umts-at-fokus.de", Contact.hosport ="193.175.133.125:5060", Via_Sent-by="193.175.133.125:5060", Contact.expires=1000000, RequestURI="sip:umts-at-fokus.de", Supported="path")
RegisterRes(status=200) / -

- / InviteReq()
InviteRes(status=200) / Ack()
- / Bye()

The verdict of the algorithm for this trace is "valid". We show here below the path taken by the trace in the EFSM specification.

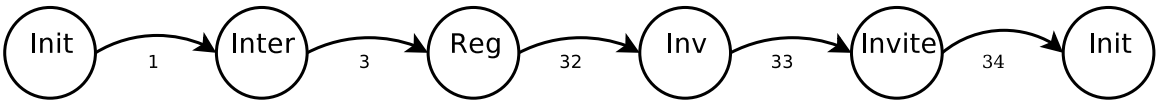

Figure 5.19: The path of the trace in the specification

Unfortunatly we can't establish a graph of the tool performances due to the quickness of the algorithm. If we use a command such as *time* under Linux in order to see how much time expired in the algorithm process, we find a null time. A solution could be to place some waiting steps inside the algorithm — in the code itself — but it is far to be obvious to find where it may be relevant and, moreover, it would be dodgy to give an interpretation then.
An other important remark has to be done. The algorithm ignore lots of messages during the analysis process of the traces. If it doesn't seem to be contradictory with the applied technique, it could nevertheless be interesting to know to which option they belong and to check if the option is well processed. In order to do it we must extend the specification to the possible options that the protocol is able to provide. In the case of IMS these options are quasi unlimited. But if we consider the studied trace, and the ignored messages :
RegisterRes (status=401, Call-ID="968@193.175.133.125", Security-Server="NULL") / RegisterReq (Security-Client="NULL", IK="NULL", Security-Verify="NULL", Call-ID= "968@193.175.133.125")
RegisterRes(status=200) / -
RegisterRes(status=401, Call-ID="2315@193.175.133.125", Security-Server="NULL") / RegisterReq (Security-Client="NULL", IK="NULL", Security-Verify="NULL", Call-ID= "2315@193.175.133.125")
, we can find — not automatically, but as an expert — that it corresponds to the execution of the protocol defined in [28], about Authentication and Key Agreement (AKA). A formal EFSM specification of this service should then be found or built in order to test the conformance of the trace. What is true for this service is also true for every possible available service, and to test intergraly the IMS we should dispose of all these formal specifications, what looks like a pretty difficult challenge...

5.3 Conclusion

In this chapter we presented the tools for EFSM drawing and backward checking as well as the IMS protocol. Then we proposed a formal specification of the IMS protocol and we process the backward checking algorithm on real IMS event traces from the IMS Playground deployed at Fokus-Fraunhofer.
We show the inherent problems of every complex communication protocol : the translation from an informal specification to a formal one — here an EFSM modelization — is a time consuming and laborious task ; at least for a non expert...
But we also see that the backward checking algorithm, even though it was initially developed in order to test "integer-based protocols" like SCP, can also be successfully applied to "string-based" ones like IMS. This experiment was the occasion to redesign the tool that tolerated only integer values and then give a wider application domain to our tool.
Anyway, this experiment has highlighted the difficulty to produce an informal specification that will be unambiguous and that leave enough freedom to every product provider. In fact, we can easily understand that these two parameters are self-dependant, in the way that more the unformal specification is unambiguous, less it let freedom for an implementation, and conversely.
It also show, in the particular case of IMS which is a very open protocol with a multitude of options, that a minimal formal specification is not always the best solution since with have to ignore a considerable amount of traffic. But for our purpose, formalizing every option would be nearly without end as IMS should make it easier to offer just about any IP-based service...

Conclusion

In this work we studied the passive conformance testing problematic in the frame of black box testing of communication protocols and services. We proposed new methods for each of the two families of passive testing, namely an invariant-based approach and an interval determination based approach. The proposed passive testing techniques will enable us to determine the conformance of an implementation under test with respect to its specification. With this purpose, event traces are recorded by observation points at no particular time, without the knowlegde of being in a specific configuration, during the normal process of the tested protocol. The backward checking approach was then applied to the IMS thanks to the open IMS playground deployed at Fokus Fraunhofer.

In the chapter 1 we first presented briefly the active and passive conformance testing families and then the formal concepts that will be used. Afterwards, we show the traditional techniques of passive testing and we highlighted the flaws of each one in order to propose some solutions to them.

In the chapter 2 we explained how to include the data portion in invariant-based approaches — that was formerly not the case.
We have shown that the former invariant approaches based on EFSM where not efficient enough since they only focused on the control portion without taking into account the data portion, and then couldn't detect a set of faults.
The concept of this approach is not far of the classical invariant approach. We need to have properties on the specification — which we name invariant — and we check if every trace follows these properties. But before checking the invariants on the event trace, we apply the invariants on the specification in order to extract constraints on the values of the variables. Then we have shown how the checked invariants are used on the event traces with the help of an example on the Simple Connection Protocol.

In the chapter 3 we show a passive testing technique based on interval determination. We have identified in the first chapter that the forward technique problem is about the length of the trace. Indeed, if a trace was too short the forward technique could miss important information and lead to an erroneous verdict. To avoid the problem, we proposed a technique that goes backward in the trace, and in its past thanks to the analysis of the specification.

We defined the structures and operations that the algorithm uses and we present examples of the process on the Simple Connection Protocol. Finally, the termination and complexity of the algorithm is depicted.

In the chapter 4 we give a parallelized version of the backward approach. As the complexity of the sequencial backward approach was not satisfactory we wanted to narrow it down. We noted that each new configuration found during the backward process was independant of the other and then a parallelization could be easily applied. We proposed three ways and three corresponding algorithms — one called *direct* and two others *by round* — in this purpose. We analysed them in terms of complexity in order to distinguish which would be the best one.
Last, we applied theoretically the parallel technique to various protocols and we highlighted the gain in complexity brought by the parallelism. We conclude by noting that the parallelize backward checking is the first fault-exhaustive passive testing technique to process within a linear time complexity.

In the chapter 5 we applied the backward technique on the IP-Multimedia Subsystem (IMS). The IMS is a real complex and challenging communication protocol. In order to prove the efficiency of our technique we first implemented our algorithms bringing some modifications in order to cope with string arguments. We present the tool and the IMS architecure, then we show on an example trace the various problematics revealed in this case study.

Future works

Various improvements for the passive testing are presented in the current work but we can always think about other ones. First, the backward checking approach must be reprocessed for each trace, i.e. if we want to apply it for real-time monitoring, a thread is launched after each recorded event. That could be very time and resource consuming. To avoid it we could think about a methodology that works iteratively, i.e. which will not compute the whole process each time but will only try to combine the configuration found for the last event with the configurations of the already processing thread.

Secondly, we could record the configurations found at the border point (at the end of the backtracking of the trace) in order to create a kind of a posteriori reachability graph. Indeed, the backward analysis of the past of the trace is a particular case of the reachability problem. The minimal complexity for determining the reachability graph of any EFSM is equal to the total number of possible configuration, which is also the worst case of the backward checking approach. It is obvious that if a reachability graph is provided, then the backward checking approach will be more or less as powerful as the forward approach. The reachability problem could also be interesting for the constrained invariant approach, since it gives the most precise constraints.

Furthermore, the most important point would be to focus on the fault location and correction problems. It exists some works on fault location on FSMs and CFSMs (cf. [18]) but no one on EFSMs. We could then try to extend these concepts and then, after identification and location of the errors, we could propose automatic corrections. That would open a path to what is commonly refered to as automonous networking.

Bibliography

[1] J.A. Arnedo, A. Cavalli, and M. Núñez, *Fast Testing of Critical Properties through Passive Testing*, Lecture Notes on Computer Science, vol. 2644/2003, pages 295-310, Springer, 2003.

[2] F. Baader, and W. Snyder, *Unification Theory, Handbook of Automated Reasoning*, Alan Robinson, Andrei Voronkov eds., Vol.1, Chapter 8, pp. 446-533.

[3] E. Bayse, A. Cavalli, M. Núñez, and F. Zaidi, *A Passive Testing Approach Based on Invariants : Application to the WAP*, Journal on Computer Network, 2004.

[4] A. Cavalli, C. Gervy, and S. Prokopenko, *New approaches for passive testing using an extended finite state machine specification*, Concordia Prestigious Workshop on Communication Software Engineering, pages 225-250, 2001.

[5] C. Gervy, *Application du test passif aux FSM étendues*, thèse de DEA, Université d'Evry, 2001.

[6] R. Hao, D. Lee, R.K. Sinha, and D. Vlah, *Testing IP routing protocols - from probabilistic algorithms to software toll*, Proceedings of FORTE-2000, pages 249-264, October 2000.

[7] D. Hogrefe, *Report on the Validation of the INRES System*, Technical Report IAM-95-007, Universitat Bern, November 1995.

[8] C.M. Huang, Y.C. Lin, and M.Y. Jang, *An executable protocol test sequence generation method for EFSM-specified protocols*, IWPTS'95 International Workshop on Protocol Test System, September 1995.

[9] *3GPP TS 24.228 (v5.13.0) : Signaling Flows for the IP Multimedia Call Control Based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)*, Stage 3 (release 5), 2005-6.

[10] *3GPP TS 24.229 (v7.1.1) : IP Multimedia Call Control Protocol Based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)*, Stage 3 (release 7), 2005-10.

[11] *3GPP TS 23.003 : Numbering, addressing and identification.*

[12] *3GPP TS 33.203 : Access security for IP based services.*

[13] R. Lai, *A survey of communication protocol testing*, Journal of Systems and Software, 62:21-46, 2002.

[14] D. Lee, D. Chen, R. Hao, R.E. Miller, J. Wu and X. Yin, *A formal approach for passive testing of protocol data portions*, Proceedings of the IEEE International Conference on Network Protocols, ICNP'02, 2002.

[15] D. Lee, K.K. Ramakrishnan, W. Melody Moh, and A. Udaya Shankar, *Protocol Specification Using Parameterized Communicating Extended Finite State Machines - a Case Study of the ATM ABR Rate Control Scheme*, International Conference on Network Protocols (ICNP) 1996, pp. 208-217, 1996.

[16] D. Lee, and M. Yannakakis, *Principes and methods of testing finite state machines - a survey*, Proceedings of the IEEE, 84(8):1090-1123, August 1996.

[17] R.E. Miller, *Passive testing of networks using a CFSM specification*, Proceedings of the IEEE International Performance Computing and Communication Conference, pages 111-116, February 1998

[18] R.E. Miller, and K.A. Arisha, *On fault location in networks by passive testing*, Technical Report NÂ°4044, Departement of Computer Science, University of Maryland, College Park, August 1999.

[19] R.E. Miller, and K.A. Arisha, *Fault identification in networks by passive testing*, 34th Simulation Symposium, SS'01, pages 277-284, Computer Society Press, 2001.

[20] R.E. Miller, and K.A. Arisha, *Fault coverage in networks by passive testing*, International Conference on Internet Computing, IC'02, pages 413-419, CSREA Press, 2001.

[21] R.E. Moore, *Methods and Applications of Interval Analysis*, Society for Industrial & Applied Mathematics, Philadelphia, 1979.

[22] K. Marriott, and P.J. Stuckey, *Programming with Constraints : an Introduction*, MIT Press.

[23] T. Clausen and P. Jacquet, *IETF RFC 3626 - Optimized Link State Routing Protocol (OLSR)* , The Internet Society, October 2003.

[24] J. Moy, *IETF RFC 2328 - OSPF Version 2* , The Internet Society, April 1998.

[25] D. Lee, A.N. Netravali, K. Sabnani, B. Sugla, A. John, *Passive testing and applications to network management*, IEEE International Conference on Network Protocols, ICNP'97, pages 113-122. IEEE Computer Society Press, 1997.

[26] T. Ramalingom, A. Das, and K. Thulasiraman, *A unified test case generation method for the EFSM model using context independent unique sequence*, IWPTS'95 International Workshop on Protocol Test System, September 1995.

[27] K.Sabnani, and A. Dahbura, *A protocol test generation procedure*, Computer Networks and ISDN Systems, 15:285-297, 1988

[28] *RFC 3310 : Hypertext Transfert Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)*

[29] *RFC 3261 : SIP : Session Initiation Protocol*, June 2002.

[30] *RFC 3262 : Reliability of provisional responses in Session Initiation Protocol (SIP)*, June 2002.

[31] *RFC 3265 : Session Initiation Protocol (SIP) Specific Event Notification*, June 2002.

[32] M. Tabourier and A. Cavalli, *Passive testing and application to the GSM-MAP protocol*, Journal of Information and Software Technology, 41:813-821, 1999.

[33] M. Tabourier, A. Cavalli, and M. Ionescu, *A GSM-MAP protocol experiment using passive testing*, World Congress on Formal Methods in the Development of Computing Systems, FM'99, LNCS 1708, pages 915-934, Springer, 1999

[34] J. Wu, Y. Zhao, and X. Yin, *From active to passive : Progress in testing of internet routing protocols*, Proceedings of FORTE 2001, pages 101-116, August 2001.

[35] http://membres.lycos.fr/baptistealcalde/

[36] http://developer.gnome.org/doc/API/glib/

[37] http://www.llnl.gov/computing/tutorials/parallel_comp/

[38] B. Alcalde, A. Cavalli, D. Chen, D. Khuu, D. Lee, *Network Protocol System Passive Testing for Fault Management - a Backward Checking Approach*, Lecture Notes on Computer Science, vol. 3235, pages 150-166, Springer, 2004.

[39] B. Tork Ladani, B. Alcalde and A. Cavalli, *Passive Testing - a Constained Invariants Checking Approach*, Lecture Notes on Computer Science, vol. 3502, Elsevier, 2005.

[40] J.M. Orset, B. Alcalde and A. Cavalli, *An EFSM-based Intrusion Detection System for Ad Hoc Networks*, ATVA'05, October 2005.

[41] B. Alcalde, A. Cavalli, *Parallel Passive Testing of System Protocols - Towards a Real-time Exhaustive Approach*, ICN'06 - International Conference on Networking, April 2006.

Appendix

A-1 The IMS traces

A-1.1 IMS simple register

REGISTER sip:umts-at-fokus.de:4060 SIP/2.0
Call-ID: 4607@193.175.133.125
Contact: <sip:subs0123450@umts-at-fokus.de>; expires=100; transport=UDP
Content-Length: 0
Cseq: 2 REGISTER
In-Reply-To: IUT <sip:subs0123450@umts-at-fokus.de>;
tag=7031
Mime-Version: 70
Unsupported: IUT <sip:subs0123450@umts-at-fokus.de>
Warning: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK6090

A-1.2 Call establishment and deregistration

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=“usim010000@umts-at-fokus.de, realm=umts-at-fokus.de,
nonce = ””, uri=”sip:pcscf.umts-at-fokus.de”,
response=””
Call-ID: 3007@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 2 REGISTER
Expires: 1000000
From: ”TESTER0” <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: ”TESTER0” <sip:subs010000@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK2233
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 3007@193.175.133.125
Cseq: 2 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-2e2e
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK2233
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "QQFh2317GD3UhnkZumVwEwAAAAAASAAAunWF4PrrKus=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010002@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 8938@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 6 REGISTER
Expires: 1000000
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK4724
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 8938@193.175.133.125
Cseq: 6 REGISTER
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-b1c6
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK4724
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))

Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "SXLXx+3wBcF3ftox5EtFRQAAAAAAIwAAYgr3pRtVq1s=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "QQFh2317GD3UhnkZumVwEwAAAAAASAAAunWF4PrrKus=",
response="69482e64b174510f", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 3007@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 3 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK2233
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 3007@193.175.133.125
Cseq: 3 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-ed09
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK2233
P-Associated-URI: <sip:subs010000@umts-at-fokus.de>
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0

Authorization: Digest username=”usim010002@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ”SXLXx+3wBcF3ftox5EtFRQAAAAAAIwAAYgr3pRtVq1s=”,
response=”4680a3a908bd1648”, algorithm=”AKAv1-MD5”, c
nonce = ”abcdefgh”, nc=”00000001”
Call-ID: 8938@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 7 REGISTER
Expires: 1000000
From: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK4724
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 8938@193.175.133.125
Cseq: 7 REGISTER
From: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
To: ”TESTER2” <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-622b
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK4724
P-Associated-URI: <sip:subs010002@umts-at-fokus.de>
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 ”Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3”

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010001@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ””, uri=”sip:pcscf.umts-at-fokus.de”,
response=””
Call-ID: 1596@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 4 REGISTER
Expires: 1000000

From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7243
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 1596@193.175.133.125
Cseq: 4 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-b415
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7243
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "5v0lNcOrZKRf4j5ccdWicQAAAAAAPgAAPgNk7v8N/t0=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 6651@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 8 REGISTER
Expires: 1000000
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8726
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 6651@193.175.133.125

Cseq: 8 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-a44a
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8726
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "q3HjboBawhjYi4ajEFRUkgAAAAAAIAAAL9MSd8M4HGo=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "5v0lNcOrZKRf4j5ccdWicQAAAAAAPgAAPgNk7v8N/t0=",
response="f4e99365d31d434a", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 1596@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 5 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7243
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 1596@193.175.133.125
Cseq: 5 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-2fbb
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7243
P-Associated-URI: <sip:subs010001@umts-at-fokus.de>
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿

Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "q3HjboBawhjYi4ajEFRUkgAAAAAAIAAAL9MSd8M4HGo=",
response="fc036247db34779c", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 6651@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 9 REGISTER
Expires: 1000000
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8726
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 6651@193.175.133.125
Cseq: 9 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-aabc
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8726
P-Associated-URI: <sip:subs010003@umts-at-fokus.de>
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

INVITE sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125

Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 6 INVITE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=2300
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7753

SIP/2.0 100 trying – your call is important to us
Call-ID: 10000000@193.175.133.125
Cseq: 6 INVITE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=2300
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7753
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:4060 "Noisy feedback tells: pid=25120
req_src_ip = 193.175.133.125 req_src_port=5062
in_uri=sip:subs010000@umts-at-fokus.de out_uri=sip:subs010000@umts-at-fokus.de via_cnt=1"

INVITE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 6 INVITE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=2300
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.78aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.eeb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.25e15d94.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.68aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7753
P-Asserted-Identity: <sip:subs010004@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280bd4a000000ba"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-

fokus.de”
P-Visited-Network-ID: ”Visited Network Number 1”
P-hint: outbound alias

INVITE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 6 INVITE
From: ”TESTER1” <sip:subs010001@umts-at-fokus.de>;
tag=2300
Max-Forwards: 13
To: ”TESTER0” <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.78aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.eeb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.25e15d94.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.68aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7753
P-Asserted-Identity: <sip:subs010004@umts-at-fokus.de>
P-Charging-Vector: icid-value=”abcd4280bd4a000000ba”; icid-gen-addr=”127.0.0.1”; orig-ioi=”umts-at-fokus.de”
P-Visited-Network-ID: ”Visited Network Number 1”
P-hint: outbound alias

INVITE sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 10000002@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 10 INVITE
From: ”TESTER3” <sip:subs010003@umts-at-fokus.de>;
tag=615
Max-Forwards: 70
To: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK5486

SIP/2.0 100 trying – your call is important to us
Call-ID: 10000002@193.175.133.125
Cseq: 10 INVITE

From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=615
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK5486
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:4060 "Noisy feedback tells: pid=25120
req_src_ip = 193.175.133.125 req_src_port=5063
in_uri=sip:subs010002@umts-at-fokus.de out_uri=sip:subs010002@umts-at-fokus.de via_cnt=1"

INVITE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000002@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 10 INVITE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=615
Max-Forwards: 13
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.2dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.ad0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.461a6cd3.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.1dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK5486
P-Asserted-Identity: <sip:subs010006@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280bd4a000000bb"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 INVITE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=2300
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5883
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.78aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.eeb03bc1.0

Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.25e15d94.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.68aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7753

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 INVITE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=2300
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5883
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7753

INVITE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000002@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 10 INVITE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=615
Max-Forwards: 13
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.2dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.ad0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.461a6cd3.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.1dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK5486
P-Asserted-Identity: <sip:subs010006@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280bd4a000000bb"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 10 INVITE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=615

To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=3692
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.2dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.ad0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.461a6cd3.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.1dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK5486

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 10 INVITE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=615
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=3692
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK5486

ACK sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 ACK
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=2300
Max-Forwards: 70
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5883
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7753

ACK sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 10 ACK
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=615
Max-Forwards: 70
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=3692
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK5486

BYE sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0

Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=2300
Max-Forwards: 70
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5883
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK5812

BYE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=2300
Max-Forwards: 13
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5883
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.317b23b4.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKd1fb.c0f764d6.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKd1fb.81d0e673.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.217b23b4.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK5812
P-Asserted-Identity: <sip:subs010004@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280bd88000000be"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

BYE sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 11 BYE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=615
Max-Forwards: 70
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=3692
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8013

BYE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000002@193.175.133.125
Content-Length: 0

Cseq: 11 BYE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=615
Max-Forwards: 13
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=3692
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.26d96961.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK906f.0ce71176.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK906f.b7d36675.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.16d96961.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8013
P-Asserted-Identity: <sip:subs010006@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280bd88000000bf"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

BYE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=2300
Max-Forwards: 13
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5883
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.317b23b4.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKd1fb.c0f764d6.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKd1fb.81d0e673.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.217b23b4.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK5812
P-Asserted-Identity: <sip:subs010004@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280bd88000000be"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

BYE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 11 BYE

From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=615
Max-Forwards: 13
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=3692
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.26d96961.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK906f.0ce71176.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK906f.b7d36675.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.16d96961.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8013
P-Asserted-Identity: <sip:subs010006@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280bd88000000bf"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=2300
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5883
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.317b23b4.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKd1fb.c0f764d6.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKd1fb.81d0e673.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.217b23b4.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK5812

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=2300
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5883
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK5812

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125

Content-Length: 0
Cseq: 11 BYE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=615
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=3692
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.26d96961.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK906f.0ce71176.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK906f.b7d36675.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.16d96961.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8013

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 11 BYE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=615
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=3692
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8013

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 8733@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Length: 0
Cseq: 12 REGISTER
Expires: 0
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK488
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 8733@193.175.133.125
Cseq: 12 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>

To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-180a
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK488
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "RjzY5IjC4bbqO1bpN2gq4wAAAAAAIQAATQFqPAVPsjU=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 7651@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>
Content-Length: 0
Cseq: 8 REGISTER
Expires: 0
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK640
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 7651@193.175.133.125
Cseq: 8 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-ce22
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK640
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",

nonce = "Q69focIOdlOYfcltyoTNsQAAAAAAPwAAlQAnud6vRS0=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "RjzY5IjC4bbqO1bpN2gq4wAAAAAAIQAATQFqPAVPsjU=",
response="ef86ee07a7ef3d4c", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 8733@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Length: 0
Cseq: 13 REGISTER
Expires: 0
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK488
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 8733@193.175.133.125
Cseq: 13 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-6c6a
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK488
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=0
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "Q69focIOdlOYfcltyoTNsQAAAAAAPwAAlQAnud6vRS0=",
response="731ae0b42ac3a146", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 7651@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>

Content-Length: 0
Cseq: 9 REGISTER
Expires: 0
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK640
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 7651@193.175.133.125
Cseq: 9 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-ff1c
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK640
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=0
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

A-1.3 The two messages acknoledged

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010002@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 8199@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 7 REGISTER
Expires: 1000000
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1294

AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 8199@193.175.133.125
Cseq: 7 REGISTER
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-25f0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1294
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "IYy14fVfMPFS/ls0JRVP0gAAAAAAAJQAAY+OkC6hK0nE=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 7524@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 3 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK1803
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 7524@193.175.133.125
Cseq: 3 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-081f
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK1803

Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "dRp4/dxatMeWC7DNc9uwTQAAAAAASgAAZ4ND7vEz6vU=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010002@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "IYy14fVfMPFS/ls0JRVP0gAAAAAAJQAAY+OkC6hK0nE=",
response="ef38a06960f8a84f", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 8199@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 8 REGISTER
Expires: 1000000
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1294
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 8199@193.175.133.125
Cseq: 8 REGISTER
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-32b8
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1294
P-Associated-URI: <sip:subs010002@umts-at-fokus.de>
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "dRp4/dxatMeWC7DNc9uwTQAAAAAASgAAZ4ND7vEz6vU=",
response="8ae0519a8876e176", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 7524@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 4 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK1803
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 7524@193.175.133.125
Cseq: 4 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-a0c8
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK1803
P-Associated-URI: <sip:subs010000@umts-at-fokus.de>
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 4457@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0

Cseq: 4 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK1441
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 4457@193.175.133.125
Cseq: 4 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-6c8f
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK1441
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "4Ews9mXAq3g4ENnpt84PeAAAAAAAQQAAGmO7IXf4rQ8=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 1184@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 8 REGISTER
Expires: 1000000
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK4205
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 1184@193.175.133.125
Cseq: 8 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-da37
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK4205
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "q0YEuMBpqk/7885q4Se5GwAAAAAAIwAA9+0AJ3NPpPA=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "4Ews9mXAq3g4ENnpt84PeAAAAAAAQQAAGmO7IXf4rQ8=",
response="cfe9f3a6988cac4f", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 4457@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 5 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK1441
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 4457@193.175.133.125
Cseq: 5 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-cfe0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK1441
P-Associated-URI: <sip:subs010001@umts-at-fokus.de>
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=1000000

Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

MESSAGE sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=7706
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK3775

MESSAGE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=7706
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.b8aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.0fb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.45e15d94.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.a8aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK3775
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280c429000000d9"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

MESSAGE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0

Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=7706
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.b8aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.0fb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.45e15d94.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.a8aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK3775
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280c429000000d9"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "q0YEuMBpqk/7885q4Se5GwAAAAAAIwAA9+0AJ3NPpPA=",
response="6da269e650d744fd", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 1184@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 9 REGISTER
Expires: 1000000
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK4205
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 1184@193.175.133.125
Cseq: 9 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>

To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-f93c
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK4205
P-Associated-URI: <sip:subs010003@umts-at-fokus.de>
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 MESSAGE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=7706
To: TESTER0 <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.b8aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.0fb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.45e15d94.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.a8aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK3775

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 MESSAGE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=7706
To: TESTER0 <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK3775

MESSAGE sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 10000002@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Length: 0
Cseq: 10 MESSAGE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=887
Max-Forwards: 70

To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6704

MESSAGE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000002@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Length: 0
Cseq: 10 MESSAGE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=887
Max-Forwards: 13
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.6dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.cd0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.661a6cd3.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.5dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6704
P-Asserted-Identity: <sip:subs010003@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd4280c42a000000db"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

MESSAGE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000002@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Length: 0
Cseq: 10 MESSAGE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=887
Max-Forwards: 13
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
User-Agent: IMSBenchmark/FOKUS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.6dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.cd0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.661a6cd3.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.5dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6704
P-Asserted-Identity: <sip:subs010003@umts-at-fokus.de>

P-Charging-Vector: icid-value="abcd4280c42a000000db"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 MESSAGE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=7706
To: TESTER0 <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.b8aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.0fb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.45e15d94.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.a8aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK3775

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 10 MESSAGE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=887
To: TESTER2 <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.6dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.cd0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.661a6cd3.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.5dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6704

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 10 MESSAGE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=887
To: TESTER2 <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6704

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125
Content-Length: 0

Cseq: 10 MESSAGE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=887
To: TESTER2 <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.6dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.cd0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.661a6cd3.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.5dfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6704

A-1.4 The 6 Call

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 4548@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 2 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK8752
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 4548@193.175.133.125
Cseq: 2 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-6920
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK8752
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",

nonce = ”LFd/z5eKRnl4ahwSOVm4HQAAAAAAPQAA6xfW8Y26/zw=”, algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010002@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ””, uri=”sip:pcscf.umts-at-fokus.de”,
response=””
Call-ID: 6808@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 6 REGISTER
Expires: 1000000
From: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1175
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 6808@193.175.133.125
Cseq: 6 REGISTER
From: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
To: ”TESTER2” <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-8a5d
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1175
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 ”Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3”
WWW-Authenticate: Digest realm=”umts-at-fokus.de”,
nonce = ”VVHF0cPUdzxm8ZWcHcz/oAAAAAAAGQAAXDj1Y/cCpXI=”, algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010006@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ””, uri=”sip:pcscf.umts-at-fokus.de”,
response=””
Call-ID: 1095@193.175.133.125
Contact: <sip:subs010006@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 14 REGISTER

Expires: 1000000
From: "TESTER6" <sip:subs010006@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER6" <sip:subs010006@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6034
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 1095@193.175.133.125
Cseq: 14 REGISTER
From: "TESTER6" <sip:subs010006@umts-at-fokus.de>
To: "TESTER6" <sip:subs010006@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-9528
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6034
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "fdTcJyNYtCOeXixpQduJdgAAAAAABAAAJY5THEBeEv4=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010004@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 6460@193.175.133.125
Contact: <sip:subs010004@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 10 REGISTER
Expires: 1000000
From: "TESTER4" <sip:subs010004@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER4" <sip:subs010004@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK8608
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE

Call-ID: 6460@193.175.133.125
Cseq: 10 REGISTER
From: "TESTER4" <sip:subs010004@umts-at-fokus.de>
To: "TESTER4" <sip:subs010004@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-7d41
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK8608
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "Yb3eRmEmhtC6P9eZEvf2uQAAAAAABAAA+vkWuy8/f7E=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010010@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 2355@193.175.133.125
Contact: <sip:subs010010@193.175.133.125:5065; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 22 REGISTER
Expires: 1000000
From: "TESTER10" <sip:subs010010@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER10" <sip:subs010010@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5065; branch=z9hG4bK5683
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 2355@193.175.133.125
Cseq: 22 REGISTER
From: "TESTER10" <sip:subs010010@umts-at-fokus.de>
To: "TESTER10" <sip:subs010010@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-af87
Via: SIP/2.0/UDP 193.175.133.125:5065; branch=z9hG4bK5683
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148

req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:stargazer.fokus.fraunhofer.de:6060 out_uri=sip:stargazer.fokus.fraunhofer.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "lTxsSrColj5AusO/LI2fXAAAAAAAAAQAAoddxpGPXkp4=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010008@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 9805@193.175.133.125
Contact: <sip:subs010008@193.175.133.125:5064; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 18 REGISTER
Expires: 1000000
From: "TESTER8" <sip:subs010008@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER8" <sip:subs010008@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5064; branch=z9hG4bK6542
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 9805@193.175.133.125
Cseq: 18 REGISTER
From: "TESTER8" <sip:subs010008@umts-at-fokus.de>
To: "TESTER8" <sip:subs010008@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-e6c1
Via: SIP/2.0/UDP 193.175.133.125:5064; branch=z9hG4bK6542
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "W3bOzsleLFURiRPfnfVC7wAAAAAABAAAuOcjxJrzeXU=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010002@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "VVHF0cPUdzxm8ZWcHcz/oAAAAAAAGQAAXDj1Y/cCpXI=",
response="a752925fa256ad0a", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"

Call-ID: 6808@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 7 REGISTER
Expires: 1000000
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1175
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 6808@193.175.133.125
Cseq: 7 REGISTER
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-1cb1
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1175
P-Associated-URI: <sip:subs010002@umts-at-fokus.de>
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "LFd/z5eKRnl4ahwSOVm4HQAAAAAAPQAA6xfW8Y26/zw=",
response="b8efa4cf3bca2410", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 4548@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 3 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>

User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK8752
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 4548@193.175.133.125
Cseq: 3 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-88ea
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK8752
P-Associated-URI: <sip:subs010000@umts-at-fokus.de>
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010004@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "Yb3eRmEmhtC6P9eZEvf2uQAAAAAABAAA+vkWuy8/f7E=",
response="076c734ef2782949", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 6460@193.175.133.125
Contact: <sip:subs010004@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 11 REGISTER
Expires: 1000000
From: "TESTER4" <sip:subs010004@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER4" <sip:subs010004@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK8608
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 6460@193.175.133.125
Cseq: 11 REGISTER

From: "TESTER4" <sip:subs010004@umts-at-fokus.de>
To: "TESTER4" <sip:subs010004@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-60cf
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK8608
P-Associated-URI: <sip:subs010004@umts-at-fokus.de>
Contact: <sip:subs010004@193.175.133.125:5062; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010006@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "fdTcJyNYtCOeXixpQduJdgAAAAAABAAAJY5THEBeEv4=",
response="20745a8a2bc84fb0", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 1095@193.175.133.125
Contact: <sip:subs010006@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 15 REGISTER
Expires: 1000000
From: "TESTER6" <sip:subs010006@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER6" <sip:subs010006@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6034
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 1095@193.175.133.125
Cseq: 15 REGISTER
From: "TESTER6" <sip:subs010006@umts-at-fokus.de>
To: "TESTER6" <sip:subs010006@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-2257
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK6034
P-Associated-URI: <sip:subs010006@umts-at-fokus.de>
Contact: <sip:subs010006@193.175.133.125:5063; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿

Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010010@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "lTxsSrColj5AusO/LI2fXAAAAAAAAAQAAoddxpGPXkp4=",
response="a2bc219bcc04b228", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 2355@193.175.133.125
Contact: <sip:subs010010@193.175.133.125:5065; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 23 REGISTER
Expires: 1000000
From: "TESTER10" <sip:subs010010@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER10" <sip:subs010010@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5065; branch=z9hG4bK5683
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 2355@193.175.133.125
Cseq: 23 REGISTER
From: "TESTER10" <sip:subs010010@umts-at-fokus.de>
To: "TESTER10" <sip:subs010010@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-2b4a
Via: SIP/2.0/UDP 193.175.133.125:5065; branch=z9hG4bK5683
P-Associated-URI: <sip:subs010010@umts-at-fokus.de>
Contact: <sip:subs010010@193.175.133.125:5065; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010008@umts-at-fokus.de", realm="umts-at-fokus.de",

nonce = "W3bOzsleLFURiRPfnfVC7wAAAAAABAAAuOcjxJrzeXU=",
response="6d62203f9ab59b92", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 9805@193.175.133.125
Contact: <sip:subs010008@193.175.133.125:5064; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 19 REGISTER
Expires: 1000000
From: "TESTER8" <sip:subs010008@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER8" <sip:subs010008@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5064; branch=z9hG4bK6542
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 9805@193.175.133.125
Cseq: 19 REGISTER
From: "TESTER8" <sip:subs010008@umts-at-fokus.de>
To: "TESTER8" <sip:subs010008@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-a3b8
Via: SIP/2.0/UDP 193.175.133.125:5064; branch=z9hG4bK6542
P-Associated-URI: <sip:subs010008@umts-at-fokus.de>
Contact: <sip:subs010008@193.175.133.125:5064; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 2544@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5066; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 4 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>

Max-Forwards: 70
To: ”TESTER1” <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5066; branch=z9hG4bK2702
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 2544@193.175.133.125
Cseq: 4 REGISTER
From: ”TESTER1” <sip:subs010001@umts-at-fokus.de>
To: ”TESTER1” <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-39af
Via: SIP/2.0/UDP 193.175.133.125:5066; branch=z9hG4bK2702
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 ”Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3”
WWW-Authenticate: Digest realm=”umts-at-fokus.de”,
nonce = ”32AEFJJ4R8fkpY3mihxE5wAAAAAANAAAOJS3VGMRlxI=”, algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010003@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ””, uri=”sip:pcscf.umts-at-fokus.de”,
response=””
Call-ID: 3351@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5067; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 8 REGISTER
Expires: 1000000
From: ”TESTER3” <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: ”TESTER3” <sip:subs010003@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5067; branch=z9hG4bK9437
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 3351@193.175.133.125
Cseq: 8 REGISTER

From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-9ccc
Via: SIP/2.0/UDP 193.175.133.125:5067; branch=z9hG4bK9437
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "XfuvJ4Ox0phf/zH8m5J4mAAAAAAAAFgAA00scmB8yJ1c=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010007@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 8408@193.175.133.125
Contact: <sip:subs010007@193.175.133.125:5069; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 16 REGISTER
Expires: 1000000
From: "TESTER7" <sip:subs010007@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER7" <sip:subs010007@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5069; branch=z9hG4bK9559
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 8408@193.175.133.125
Cseq: 16 REGISTER
From: "TESTER7" <sip:subs010007@umts-at-fokus.de>
To: "TESTER7" <sip:subs010007@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-dee9
Via: SIP/2.0/UDP 193.175.133.125:5069; branch=z9hG4bK9559
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "HeP//mKZXcXcwykMGAmaaQAAAAAABAAAdYOjNMSURcI=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010005@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 776@193.175.133.125
Contact: <sip:subs010005@193.175.133.125:5068; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 12 REGISTER
Expires: 1000000
From: "TESTER5" <sip:subs010005@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER5" <sip:subs010005@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5068; branch=z9hG4bK1681
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 776@193.175.133.125
Cseq: 12 REGISTER
From: "TESTER5" <sip:subs010005@umts-at-fokus.de>
To: "TESTER5" <sip:subs010005@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-72f7
Via: SIP/2.0/UDP 193.175.133.125:5068; branch=z9hG4bK1681
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "DSbt5dBnRBd9xpCpBDHyJgAAAAAABAAAfAdXFofUOmw=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010009@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 1927@193.175.133.125
Contact: <sip:subs010009@193.175.133.125:5070; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0

Cseq: 20 REGISTER
Expires: 1000000
From: "TESTER9" <sip:subs010009@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER9" <sip:subs010009@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5070; branch=z9hG4bK133
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 1927@193.175.133.125
Cseq: 20 REGISTER
From: "TESTER9" <sip:subs010009@umts-at-fokus.de>
To: "TESTER9" <sip:subs010009@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-658f
Via: SIP/2.0/UDP 193.175.133.125:5070; branch=z9hG4bK133
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "odVhgcmDyTDIkxTKRgzZHwAAAAAABAAAJ9ReUCCoX8k=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010011@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 5956@193.175.133.125
Contact: <sip:subs010011@193.175.133.125:5071; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 24 REGISTER
Expires: 1000000
From: "TESTER11" <sip:subs010011@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER11" <sip:subs010011@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5071; branch=z9hG4bK2215
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 5956@193.175.133.125
Cseq: 24 REGISTER
From: "TESTER11" <sip:subs010011@umts-at-fokus.de>
To: "TESTER11" <sip:subs010011@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-45bd
Via: SIP/2.0/UDP 193.175.133.125:5071; branch=z9hG4bK2215
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:stargazer.fokus.fraunhofer.de:6060 out_uri=sip:stargazer.fokus.fraunhofer.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "1ZVY3CFj4v/i7GDHIRdOLAAAAAAAAAQAA5ciHKem5iUg=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "32AEFJJ4R8fkpY3mihxE5wAAAAAANAAAOJS3VGMRlxI=",
response="f2f0cffb85c23e13", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 2544@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5066; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 5 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5066; branch=z9hG4bK2702
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 2544@193.175.133.125
Cseq: 5 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-8ec6
Via: SIP/2.0/UDP 193.175.133.125:5066; branch=z9hG4bK2702
P-Associated-URI: <sip:subs010001@umts-at-fokus.de>
Contact: <sip:subs010001@193.175.133.125:5066; transport=UDP>; expires=1000000

Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "XfuvJ4Ox0phf/zH8m5J4mAAAAAAAAFgAA00scmB8yJ1c=",
response="b1fc188aa37c7d24", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 3351@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5067; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 9 REGISTER
Expires: 1000000
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5067; branch=z9hG4bK9437
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 3351@193.175.133.125
Cseq: 9 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-2011
Via: SIP/2.0/UDP 193.175.133.125:5067; branch=z9hG4bK9437
P-Associated-URI: <sip:subs010003@umts-at-fokus.de>
Contact: <sip:subs010003@193.175.133.125:5067; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

A-1.5 A complete call

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 968@193.175.133.12
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 2 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK8079
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 968@193.175.133.125
Cseq: 2 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-e80f
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK8079
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "E+ASYyIYwUtQPvsa50Pg0gAAAAAANwAAmrl/EadBufM=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "E+ASYyIYwUtQPvsa50Pg0gAAAAAANwAAmrl/EadBufM=",
response="9dcd6cb7d30fae0f", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 968@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0

Cseq: 3 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK8079
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 968@193.175.133.125
Cseq: 3 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-b989
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK8079
P-Associated-URI: <sip:subs010000@umts-at-fokus.de>
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 2315@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 4 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK9202
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 2315@193.175.133.125
Cseq: 4 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-3b48
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK9202
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "lwmjRUnYbyU+srMdG70PZQAAAAAALgAARy/SswTZj/w=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "lwmjRUnYbyU+srMdG70PZQAAAAAALgAARy/SswTZj/w=",
response="a64a0bf244871c70", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 2315@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 5 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK9202
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 2315@193.175.133.125
Cseq: 5 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-9de6
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK9202
P-Associated-URI: <sip:subs010001@umts-at-fokus.de>

Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

INVITE sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 6 INVITE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=5772
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2287

SIP/2.0 100 trying – your call is important to us
Call-ID: 10000000@193.175.133.125
Cseq: 6 INVITE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=5772
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2287
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:4060 "Noisy feedback tells: pid=25120
req_src_ip = 193.175.133.125 req_src_port=5061
in_uri=sip:subs010000@umts-at-fokus.de out_uri=sip:subs010000@umts-at-fokus.de via_cnt=1"

INVITE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 6 INVITE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;

tag=5772
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.e6aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.1eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.54e15d94.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.d6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2287
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa2e000000004"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 INVITE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=5772
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5515
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.e6aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.1eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.54e15d94.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.d6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2287

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 INVITE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=5772
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5515
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2287

ACK sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 ACK

From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=5772
Max-Forwards: 70
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5515
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2287

BYE sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=5772
Max-Forwards: 70
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5515
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK3638

BYE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=5772
Max-Forwards: 13
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5515
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.407b23b4.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKd1fb.40f764d6.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKd1fb.01d0e673.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.307b23b4.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK3638
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa31d00000006"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE

From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=5772
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5515
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.407b23b4.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKd1fb.40f764d6.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKd1fb.01d0e673.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.307b23b4.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK3638

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=5772
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=5515
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK3638

A-1.6 A 2 call

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010002@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 5429@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 6 REGISTER
Expires: 1000000
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK5400
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 5429@193.175.133.125
Cseq: 6 REGISTER
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>

To: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-4e96
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK5400
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "gjG7wqlOe43/RqkJyuOvIAAAAAAAAGgAAjifh05HPqhM=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 5047@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 2 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK6598
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 5047@193.175.133.125
Cseq: 2 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-fb38
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK6598
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",

nonce = ”g7UxGkB4k7jjsMocCoM5pQAAAAAAPwAAMOR6lClwL+o=”, algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010002@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ”gjG7wqlOe43/RqkJyuOvIAAAAAAAGgAAjifh05HPqhM=”,
response=”26f26002cc614bad”, algorithm=”AKAv1-MD5”, c
nonce = ”abcdefgh”, nc=”00000001”
Call-ID: 5429@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 7 REGISTER
Expires: 1000000
From: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK5400
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 5429@193.175.133.125
Cseq: 7 REGISTER
From: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
To: ”TESTER2” <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-eab8
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK5400
P-Associated-URI: <sip:subs010002@umts-at-fokus.de>
Contact: <sip:subs010002@193.175.133.125:5061; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 ”Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3”

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010000@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ”g7UxGkB4k7jjsMocCoM5pQAAAAAAPwAAMOR6lClwL+o=”,
response=”839c138396fc2062”, algorithm=”AKAv1-MD5”, c
nonce = ”abcdefgh”, nc=”00000001”
Call-ID: 5047@193.175.133.125

Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 3 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK6598
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 5047@193.175.133.125
Cseq: 3 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-45d9
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK6598
P-Associated-URI: <sip:subs010000@umts-at-fokus.de>
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 8283@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 4 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7077

AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 8283@193.175.133.125
Cseq: 4 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-7add
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7077
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "eXAMNcmHAasVrSyYnJxhfAAAAAAAANQAALBXJhGmm0Ng=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 8092@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 8 REGISTER
Expires: 1000000
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8679
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 8092@193.175.133.125
Cseq: 8 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-a856
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8679

Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "+Agn1uB+mV5jQmZkMFXfjgAAAAAAFwAAurVrfRQ/OdA=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "+Agn1uB+mV5jQmZkMFXfjgAAAAAAFwAAurVrfRQ/OdA=",
response="088be1e3613dfad7", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 8092@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 9 REGISTER
Expires: 1000000
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8679
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 8092@193.175.133.125
Cseq: 9 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-9d3a
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8679
P-Associated-URI: <sip:subs010003@umts-at-fokus.de>
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "eXAMNcmHAasVrSyYnJxhfAAAAAAAANQAALBXJhGmm0Ng=",
response="1c08167c15871201", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 8283@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 5 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7077
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 8283@193.175.133.125
Cseq: 5 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-ff0f
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7077
P-Associated-URI: <sip:subs010001@umts-at-fokus.de>
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

INVITE sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 10000002@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 10 INVITE

From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=2492
Max-Forwards: 70
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK3435

SIP/2.0 100 trying – your call is important to us
Call-ID: 10000002@193.175.133.125
Cseq: 10 INVITE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=2492
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK3435
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:4060 "Noisy feedback tells: pid=25120
req_src_ip = 193.175.133.125 req_src_port=5063
in_uri=sip:subs010002@umts-at-fokus.de out_uri=sip:subs010002@umts-at-fokus.de via_cnt=1"

INVITE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000002@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 10 INVITE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=2492
Max-Forwards: 13
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.2cfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.2d0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.c51a6cd3.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.1cfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK3435
P-Asserted-Identity: <sip:subs010006@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fb3f600000041"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

INVITE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0

Call-ID: 10000002@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 10 INVITE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=2492
Max-Forwards: 13
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.2cfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.2d0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.c51a6cd3.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.1cfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK3435
P-Asserted-Identity: <sip:subs010006@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fb3f600000041"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

INVITE sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 6 INVITE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=4887
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7588

SIP/2.0 100 trying – your call is important to us
Call-ID: 10000000@193.175.133.125
Cseq: 6 INVITE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=4887
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7588
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))

Content-Length: 0
Warning: 392 193.175.133.122:4060 "Noisy feedback tells: pid=25120
req_src_ip = 193.175.133.125 req_src_port=5062
in_uri=sip:subs010000@umts-at-fokus.de out_uri=sip:subs010000@umts-at-fokus.de via_cnt=1"

INVITE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5062; transport=UDP>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 6 INVITE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=4887
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.77aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.6eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.a4e15d94.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.67aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7588
P-Asserted-Identity: <sip:subs010004@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fb3f700000042"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 10 INVITE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=2492
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=2189
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.2cfd85a5.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK806f.2d0e9be5.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK806f.c51a6cd3.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK806f.1cfd85a5.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK3435

SIP/2.0 200 OK

Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 10 INVITE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=2492
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=2189
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK3435

ACK sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 10 ACK
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=2492
Max-Forwards: 70
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=2189
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK3435

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 INVITE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=4887
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=6387
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.77aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.6eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.a4e15d94.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.67aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7588

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 INVITE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=4887
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=6387
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7588

ACK sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 6 ACK
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=4887
Max-Forwards: 70
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=6387
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK7588

BYE sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 11 BYE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=2492
Max-Forwards: 70
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=2189
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8657

BYE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 11 BYE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=2492
Max-Forwards: 13
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=2189
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.a5d96961.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK906f.cbe71176.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK906f.77d36675.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.95d96961.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8657
P-Asserted-Identity: <sip:subs010006@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fb43400000045"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

BYE sip:subs010002@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 11 BYE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=2492
Max-Forwards: 13
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=2189
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.a5d96961.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK906f.cbe71176.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK906f.77d36675.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.95d96961.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8657
P-Asserted-Identity: <sip:subs010006@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fb43400000045"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 11 BYE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=2492
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=2189
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.a5d96961.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK906f.cbe71176.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bK906f.77d36675.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK906f.95d96961.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8657

SIP/2.0 200 OK
Call-ID: 10000002@193.175.133.125
Content-Length: 0
Cseq: 11 BYE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=2492
To: TESTER2 <sip:subs010002@umts-at-fokus.de>;

tag=2189
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK8657

BYE sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=4887
Max-Forwards: 70
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=6387
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK4383

BYE sip:subs010000@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=4887
Max-Forwards: 13
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=6387
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.b07b23b4.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKd1fb.80f764d6.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKd1fb.41d0e673.0
P-Called-Party-ID: <sip:subs010000@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.a07b23b4.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK4383
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fb43400000046"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=4887
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=6387

Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.b07b23b4.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKd1fb.80f764d6.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKd1fb.41d0e673.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKd1fb.a07b23b4.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK4383

SIP/2.0 200 OK
Call-ID: 10000000@193.175.133.125
Content-Length: 0
Cseq: 7 BYE
From: TESTER1 <sip:subs010001@umts-at-fokus.de>;
tag=4887
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=6387
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK4383

REGISTER sip:subs010000@umts-at-fokus.de SIP/2.0
Call-ID:
Contact: *
Content-Length: 0
Cseq: 8 REGISTER
Expires: 0
From: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=1739
Max-Forwards: 70
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=6387
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK9659

SIP/2.0 403 Forbidden - HSS User Unknown
Call-ID:
Cseq: 8 REGISTER
From: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=1739
To: TESTER0 <sip:subs010000@umts-at-fokus.de>;
tag=6387
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK9659
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:5060 "Noisy feedback tells: pid=25140
req_src_ip = 193.175.133.122 req_src_port=4060
in_uri=sip:subs010000@umts-at-fokus.de out_uri=sip:subs010000@umts-at-fokus.de via_cnt=2"

A-1.7 Messaging

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 2542@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 2 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK9407
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 2542@193.175.133.125
Cseq: 2 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-e383
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK9407
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "SRxVbDUWuIZVs6E894EO4gAAAAAAOQAABG4P32qdERE=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "SRxVbDUWuIZVs6E894EO4gAAAAAAOQAABG4P32qdERE=",
response="71945e5ffaaf76ac", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 2542@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0

Cseq: 3 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK9407
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 2542@193.175.133.125
Cseq: 3 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-538e
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK9407
P-Associated-URI: <sip:subs010000@umts-at-fokus.de>
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 8303@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 4 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1771
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 8303@193.175.133.125
Cseq: 4 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-7f3d
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1771
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "JbqWb5IFlNC3R+7TBf044AAAAAAAMAAAV5/69r3noHU=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "JbqWb5IFlNC3R+7TBf044AAAAAAAMAAAV5/69r3noHU=",
response="525b37b993724c9a", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 8303@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 5 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1771
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 8303@193.175.133.125
Cseq: 5 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-0009
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1771
P-Associated-URI: <sip:subs010001@umts-at-fokus.de>

Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=25148
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

MESSAGE sip:subs010001@umts-at-fokus.de SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573

MESSAGE sip:subs010001@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.07aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.2eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.64e15d94.0
P-Called-Party-ID: <sip:subs010001@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.f6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa45b0000000e"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

MESSAGE sip:subs010001@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.07aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.2eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.64e15d94.0
P-Called-Party-ID: <sip:subs010001@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.f6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa45b0000000e"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

MESSAGE sip:subs010001@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.07aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.2eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.64e15d94.0
P-Called-Party-ID: <sip:subs010001@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.f6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa45b0000000e"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 408 Request Timeout
Call-ID: 10000000@193.175.133.125
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=0b83d27bb500dd054219024090597c3b-baa1
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:5060 "Noisy feedback tells: pid=25145
req_src_ip = 193.175.133.122 req_src_port=4060
in_uri=sip:subs010001@umts-at-fokus.de out_uri=sip:subs010001@umts-at-fokus.de via_cnt=0"

MESSAGE sip:subs010001@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.07aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.2eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.64e15d94.0
P-Called-Party-ID: <sip:subs010001@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.f6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa45b0000000e"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de" P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

MESSAGE sip:subs010001@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908

Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.07aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.2eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.64e15d94.0
P-Called-Party-ID: <sip:subs010001@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.f6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa45b0000000e"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

MESSAGE sip:subs010001@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.07aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.2eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.64e15d94.0
P-Called-Party-ID: <sip:subs010001@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.f6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa45b0000000e"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

MESSAGE sip:subs010001@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;

tag=8908
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.07aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.2eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.64e15d94.0
P-Called-Party-ID: <sip:subs010001@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.f6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa45b0000000e"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

MESSAGE sip:subs010001@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.07aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.2eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.64e15d94.0
P-Called-Party-ID: <sip:subs010001@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.f6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa45b0000000e"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

MESSAGE sip:subs010001@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 10000000@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>
Content-Length: 0
Cseq: 6 MESSAGE

From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
Max-Forwards: 13
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.07aaba24.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKc1fb.2eb03bc1.0
Via: SIP/2.0/UDP 193.175.133.122; branch=z9hG4bKc1fb.64e15d94.0
P-Called-Party-ID: <sip:subs010001@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bKc1fb.f6aaba24.0
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
P-Asserted-Identity: <sip:subs010001@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd427fa45b0000000e"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

SIP/2.0 408 Request Timeout
Call-ID: 10000000@193.175.133.125
Cseq: 6 MESSAGE
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=8908
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=a2b4eb3abc23cc763bd1273522a1e963-30ab
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2573
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:4060 "Noisy feedback tells: pid=25121
req_src_ip = 193.175.133.122 req_src_port=6060
in_uri=sip:subs010001@193.175.133.125:5061; transport=UDP out_uri=sip:subs010001@193.175.133.125:5061;
transport=UDP via_cnt=0"

A-1.8 Workbench

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 4329@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 4 REGISTER

Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2246
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 4329@193.175.133.125
Cseq: 4 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-bbc7
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2246
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "1x3LCn8/hGjyr1gDOeDvkQAAAAAABAAAlYCxPsQthUc=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "", uri="sip:pcscf.umts-at-fokus.de",
response=""
Call-ID: 9527@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 2 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK5540
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE

Call-ID: 9527@193.175.133.125
Cseq: 2 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-c300
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK5540
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "x+z3s+baQGyFe6EAAQpSpAAAAAAABAAA6Kv/ynBAFYA=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "1x3LCn8/hGjyr1gDOeDvkQAAAAAABAAAlYCxPsQthUc=",
response="c8bf0603a322b85c", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 4329@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 5 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2246
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 4329@193.175.133.125
Cseq: 5 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-9410
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK2246
P-Associated-URI: <sip:subs010001@umts-at-fokus.de>
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>

Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010000@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "x+z3s+baQGyFe6EAAQpSpAAAAAAAABAAA6Kv/ynBAFYA=",
response="2abf6ab23e4db947", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 9527@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 3 REGISTER
Expires: 1000000
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK5540
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 9527@193.175.133.125
Cseq: 3 REGISTER
From: "TESTER0" <sip:subs010000@umts-at-fokus.de>
To: "TESTER0" <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-6a16
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK5540
P-Associated-URI: <sip:subs010000@umts-at-fokus.de>
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0

Authorization: Digest username=”usim010002@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ””, uri=”sip:pcscf.umts-at-fokus.de”,
response=””
Call-ID: 2515@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 6 REGISTER
Expires: 1000000
From: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK8368
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 2515@193.175.133.125
Cseq: 6 REGISTER
From: ”TESTER2” <sip:subs010002@umts-at-fokus.de>
To: ”TESTER2” <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-015a
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK8368
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 ”Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3”
WWW-Authenticate: Digest realm=”umts-at-fokus.de”,
nonce = ”kXtu6bjb2Gu+3CO8/Or/0wAAAAAACwAApCOCEL45Rvw=”, algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010003@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ””, uri=”sip:pcscf.umts-at-fokus.de”,
response=””
Call-ID: 7622@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 8 REGISTER
Expires: 1000000
From: ”TESTER3” <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70

To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK4688
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 7622@193.175.133.125
Cseq: 8 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-df36
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK4688
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "hQjSljJidHrpsivvEYqH6wAAAAAACwAAIF2iPPLEzSk=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010002@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "kXtu6bjb2Gu+3CO8/Or/0wAAAAAACwAApCOCEL45Rvw=",
response="ec8e217705ee54f7", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 2515@193.175.133.125
Contact: <sip:subs010002@193.175.133.125:5062; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 7 REGISTER
Expires: 1000000
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK8368
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 2515@193.175.133.125
Cseq: 7 REGISTER

From: "TESTER2" <sip:subs010002@umts-at-fokus.de>
To: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-2d98
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK8368
P-Associated-URI: <sip:subs010002@umts-at-fokus.de>
Contact: <sip:subs010002@193.175.133.125:5062; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010003@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "hQjSljJidHrpsivvEYqH6wAAAAAACwAAIF2iPPLEzSk=",
response="91ce046a6e397e74", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 7622@193.175.133.125
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 9 REGISTER
Expires: 1000000
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK4688
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 7622@193.175.133.125
Cseq: 9 REGISTER
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>
To: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-02d8
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK4688
P-Associated-URI: <sip:subs010003@umts-at-fokus.de>
Contact: <sip:subs010003@193.175.133.125:5063; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿

Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"

INVITE sip:subs010003@umts-at-fokus.de SIP/2.0
Call-ID: 12345678@193.175.133.125
Contact: <sip:subs010003@umts-at-fokus.de>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 10 INVITE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=5270
Max-Forwards: 70
Route: <sip:stargazer.fokus.fraunhofer.de:4060¿ , <sip:orig@scscf.umts-at-fokus.de:6060¿
To: IUT <sip:subs010003@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825

SIP/2.0 100 trying – your call is important to us
Call-ID: 12345678@193.175.133.125
Cseq: 10 INVITE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=5270
To: IUT <sip:subs010003@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:4060 "Noisy feedback tells: pid=16775
req_src_ip = 193.175.133.125 req_src_port=5063
in_uri=sip:subs010003@umts-at-fokus.de out_uri=sip:subs010003@umts-at-fokus.de via_cnt=1"

INVITE sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 12345678@193.175.133.125
Contact: <sip:subs010003@umts-at-fokus.de>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 10 INVITE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=5270
Max-Forwards: 14

To: IUT <sip:subs010003@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK1b68.5f96e6c7.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK1b68.97cc1037.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK1b68.4f96e6c7.0
P-Called-Party-ID: <sip:subs010003@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825
P-Asserted-Identity: <sip:subs010003@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd42791dfb00000039"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

INVITE sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 12345678@193.175.133.125
Contact: <sip:subs010002@umts-at-fokus.de>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 8 INVITE
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=2853
Max-Forwards: 70
Route: <sip:stargazer.fokus.fraunhofer.de:4060¿ , <sip:orig@scscf.umts-at-fokus.de:6060¿
To: IUT <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437

SIP/2.0 100 trying – your call is important to us
Call-ID: 12345678@193.175.133.125
Cseq: 8 INVITE
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=2853
To: IUT <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:4060 "Noisy feedback tells: pid=16775
req_src_ip = 193.175.133.125 req_src_port=5062
in_uri=sip:subs010002@umts-at-fokus.de out_uri=sip:subs010002@umts-at-fokus.de via_cnt=1"

INVITE sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 12345678@193.175.133.125
Contact: <sip:subs010002@umts-at-fokus.de>
Content-Disposition: session

Content-Length: 12
Content-Type: application/sdp
Cseq: 8 INVITE
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=2853
Max-Forwards: 14
To: IUT <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK3df4.59dfd482.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK3df4.047743a5.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK3df4.49dfd482.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437
P-Asserted-Identity: <sip:subs010002@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd42791dfb0000003a"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

INVITE sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Call-ID: 12345678@193.175.133.125
Contact: <sip:subs010003@umts-at-fokus.de>
Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 10 INVITE
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=5270
Max-Forwards: 14
To: IUT <sip:subs010003@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK1b68.5f96e6c7.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK1b68.97cc1037.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK1b68.4f96e6c7.0
P-Called-Party-ID: <sip:subs010003@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825
P-Asserted-Identity: <sip:subs010003@umts-at-fokus.de>
P-Charging-Vector: icid-value="abcd42791dfb00000039"; icid-gen-addr="127.0.0.1"; orig-ioi="umts-at-fokus.de"
P-Visited-Network-ID: "Visited Network Number 1"
P-hint: outbound alias

INVITE sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Call-ID: 12345678@193.175.133.125
Contact: <sip:subs010002@umts-at-fokus.de>

Content-Disposition: session
Content-Length: 12
Content-Type: application/sdp
Cseq: 8 INVITE
From: ”TESTER2” <sip:subs010002@umts-at-fokus.de>;
tag=2853
Max-Forwards: 14
To: IUT <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK3df4.59dfd482.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK3df4.047743a5.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK3df4.49dfd482.0
P-Called-Party-ID: <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437
P-Asserted-Identity: <sip:subs010002@umts-at-fokus.de>
P-Charging-Vector: icid-value=”abcd42791dfb0000003a”; icid-gen-addr=”127.0.0.1”; orig-ioi=”umts-at-fokus.de”
P-Visited-Network-ID: ”Visited Network Number 1”
P-hint: outbound alias

NOTIFY sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKe8ce.a448e372.0
To: sip:subs010002@umts-at-fokus.de
From: sip:subs010002@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-2df5
CSeq: 10 NOTIFY
Call-ID: 19aa5dee-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version=”1.0”?>
<reginfo xmlns=”urn:ietf:params:xml:ns:reginfo” version=”0” state=”partial”>
<registration aor=”sip:subs010002@umts-at-fokus.de” id=”0xb5bde6d4” state=”active”>
<contact id=”0xb5bdbe4c” state=”active” event=”refreshed” expires=”1000000”>
<uri>sip:subs010002@193.175.133.125:5062; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0

Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKf8ce.7ac3e62.0 To: sip:subs010003@umts-at-fokus.de
From: sip:subs010003@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-812e
CSeq: 10 NOTIFY
Call-ID: 19aa5def-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version=”1.0”?>
<reginfo xmlns=”urn:ietf:params:xml:ns:reginfo” version=”0” state=”partial”>
<registration aor=”sip:subs010003@umts-at-fokus.de” id=”0xb5bde88c” state=”active”>
<contact id=”0xb5be58b4” state=”active” event=”refreshed” expires=”1000000”>
<uri>sip:subs010003@193.175.133.125:5063; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKe8ce.a448e372.0
To: sip:subs010002@umts-at-fokus.de
From: sip:subs010002@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-2df5
CSeq: 10 NOTIFY
Call-ID: 19aa5dee-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version=”1.0”?>
<reginfo xmlns=”urn:ietf:params:xml:ns:reginfo” version=”0” state=”partial”>
<registration aor=”sip:subs010002@umts-at-fokus.de” id=”0xb5bde6d4” state=”active”>
<contact id=”0xb5bdbe4c” state=”active” event=”refreshed” expires=”1000000”>
<uri>sip:subs010002@193.175.133.125:5062; transport=UDP</uri>
</contact¿
</registration¿

</reginfo¿

NOTIFY sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKf8ce.7ac3e62.0
To: sip:subs010003@umts-at-fokus.de
From: sip:subs010003@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-812e
CSeq: 10 NOTIFY
Call-ID: 19aa5def-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010003@umts-at-fokus.de" id="0xb5bde88c" state="active">
<contact id="0xb5be58b4" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010003@193.175.133.125:5063; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

CANCEL sip:subs010003@umts-at-fokus.de SIP/2.0
Call-ID: 3521@193.175.133.125
Cseq: 10 CANCEL
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=5270
Max-Forwards: 70
To: IUT <sip:subs010003@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825

SIP/2.0 404 Not Found and no voicemail turned on
Call-ID: 3521@193.175.133.125
Cseq: 10 CANCEL
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=5270
To: IUT <sip:subs010003@umts-at-fokus.de>;
tag=b27e1a1d33761e85846fc98f5f3a7e58.c91d
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825
Server: Sip EXpress router (0.8.12 (i386/linux))

Content-Length: 0
Warning: 392 193.175.134.84:5060 "Noisy feedback tells: pid=12385
req_src_ip = 193.175.134.85 req_src_port=5060
in_uri=sip:subs010003@umts-at-fokus.de out_uri=sip:subs010003@umts-at-fokus.de via_cnt=3"

SIP/2.0 200 OK
Call-ID: 12345678@193.175.133.125
Content-Length: 0
Cseq: 10 INVITE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=5270
To: IUT <sip:subs010003@umts-at-fokus.de>;
tag=8141
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK1b68.5f96e6c7.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK1b68.97cc1037.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK1b68.4f96e6c7.0
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825

SIP/2.0 200 OK
Call-ID: 12345678@193.175.133.125
Content-Length: 0
Cseq: 10 INVITE
From: TESTER3 <sip:subs010003@umts-at-fokus.de>;
tag=5270
To: IUT <sip:subs010003@umts-at-fokus.de>;
tag=8141
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825

SIP/2.0 200 OK
Call-ID: 12345678@193.175.133.125
Content-Length: 0
Cseq: 8 INVITE
From: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=2853
To: IUT <sip:subs010002@umts-at-fokus.de>;
tag=7025
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK3df4.59dfd482.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bK3df4.047743a5.0
Via: SIP/2.0/UDP 193.175.133.122:4060; branch=z9hG4bK3df4.49dfd482.0
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437

SIP/2.0 200 OK
Call-ID: 12345678@193.175.133.125

Content-Length: 0
Cseq: 8 INVITE
From: TESTER2 <sip:subs010002@umts-at-fokus.de>;
tag=2853
To: IUT <sip:subs010002@umts-at-fokus.de>;
tag=7025
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437

CANCEL sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 3197@193.175.133.125
Cseq: 8 CANCEL
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=2853
Max-Forwards: 70
To: IUT <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437

SIP/2.0 404 Not Found and no voicemail turned on
Call-ID: 3197@193.175.133.125
Cseq: 8 CANCEL
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=2853
To: IUT <sip:subs010002@umts-at-fokus.de>;
tag=b27e1a1d33761e85846fc98f5f3a7e58.8026
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437
Server: Sip EXpress router (0.8.12 (i386/linux))
Content-Length: 0
Warning: 392 193.175.134.84:5060 "Noisy feedback tells: pid=12389
req_src_ip = 193.175.134.85 req_src_port=5060
in_uri=sip:subs010002@umts-at-fokus.de out_uri=sip:subs010002@umts-at-fokus.de via_cnt=3"

ACK sip:subs010003@umts-at-fokus.de SIP/2.0
Call-ID: 3521@193.175.133.125
Content-Length: 0
Cseq: 10 ACK
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=5270
Max-Forwards: 70
To: IUT <sip:subs010003@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825

NOTIFY sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKe8ce.a448e372.0

To: sip:subs010002@umts-at-fokus.de
From: sip:subs010002@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-2df5
CSeq: 10 NOTIFY
Call-ID: 19aa5dee-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010002@umts-at-fokus.de" id="0xb5bde6d4" state="active">
<contact id="0xb5bdbe4c" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010002@193.175.133.125:5062; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKf8ce.7ac3e62.0
To: sip:subs010003@umts-at-fokus.de
From: sip:subs010003@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-812e
CSeq: 10 NOTIFY
Call-ID: 19aa5def-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010003@umts-at-fokus.de" id="0xb5bde88c" state="active">
<contact id="0xb5be58b4" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010003@193.175.133.125:5063; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

ACK sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 3197@193.175.133.125
Content-Length: 0
Cseq: 8 ACK
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=2853
Max-Forwards: 70
To: IUT <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437

ACK sip:subs010003@umts-at-fokus.de SIP/2.0
Call-ID: 3521@193.175.133.125
Content-Length: 0
Cseq: 10 ACK
From: "TESTER3" <sip:subs010003@umts-at-fokus.de>;
tag=5270
Max-Forwards: 70
To: IUT <sip:subs010003@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5063; branch=z9hG4bK825

ACK sip:subs010002@umts-at-fokus.de SIP/2.0
Call-ID: 3197@193.175.133.125
Content-Length: 0
Cseq: 8 ACK
From: "TESTER2" <sip:subs010002@umts-at-fokus.de>;
tag=2853
Max-Forwards: 70
To: IUT <sip:subs010002@umts-at-fokus.de>
Via: SIP/2.0/UDP 193.175.133.125:5062; branch=z9hG4bK6437

NOTIFY sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKe8ce.a448e372.0
To: sip:subs010002@umts-at-fokus.de
From: sip:subs010002@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-2df5
CSeq: 10 NOTIFY
Call-ID: 19aa5dee-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70

Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version=”1.0”?>
<reginfo xmlns=”urn:ietf:params:xml:ns:reginfo” version=”0” state=”partial”>
<registration aor=”sip:subs010002@umts-at-fokus.de” id=”0xb5bde6d4” state=”active”>
<contact id=”0xb5bdbe4c” state=”active” event=”refreshed” expires=”1000000”>
<uri>sip:subs010002@193.175.133.125:5062; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKf8ce.7ac3e62.0
To: sip:subs010003@umts-at-fokus.de
From: sip:subs010003@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-812e
CSeq: 10 NOTIFY
Call-ID: 19aa5def-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version=”1.0”?>
<reginfo xmlns=”urn:ietf:params:xml:ns:reginfo” version=”0” state=”partial”>
<registration aor=”sip:subs010003@umts-at-fokus.de” id=”0xb5bde88c” state=”active”>
<contact id=”0xb5be58b4” state=”active” event=”refreshed” expires=”1000000”>
<uri>sip:subs010003@193.175.133.125:5063; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKe8ce.a448e372.0
To: sip:subs010002@umts-at-fokus.de
From: sip:subs010002@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-2df5
CSeq: 10 NOTIFY
Call-ID: 19aa5dee-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))

Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010002@umts-at-fokus.de" id="0xb5bde6d4" state="active">
<contact id="0xb5bdbe4c" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010002@193.175.133.125:5062; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKf8ce.7ac3e62.0
To: sip:subs010003@umts-at-fokus.de
From: sip:subs010003@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-812e
CSeq: 10 NOTIFY
Call-ID: 19aa5def-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010003@umts-at-fokus.de" id="0xb5bde88c" state="active">
<contact id="0xb5be58b4" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010003@193.175.133.125:5063; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKe8ce.a448e372.0
To: sip:subs010002@umts-at-fokus.de
From: sip:subs010002@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-2df5
CSeq: 10 NOTIFY

Call-ID: 19aa5dee-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010002@umts-at-fokus.de" id="0xb5bde6d4" state="active">
<contact id="0xb5bdbe4c" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010002@193.175.133.125:5062; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKf8ce.7ac3e62.0
To: sip:subs010003@umts-at-fokus.de
From: sip:subs010003@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-812e
CSeq: 10 NOTIFY
Call-ID: 19aa5def-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010003@umts-at-fokus.de" id="0xb5bde88c" state="active">
<contact id="0xb5be58b4" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010003@193.175.133.125:5063; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKe8ce.a448e372.0
To: sip:subs010002@umts-at-fokus.de

From: sip:subs010002@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-2df5
CSeq: 10 NOTIFY
Call-ID: 19aa5dee-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010002@umts-at-fokus.de" id="0xb5bde6d4" state="active">
<contact id="0xb5bdbe4c" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010002@193.175.133.125:5062; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKf8ce.7ac3e62.0
To: sip:subs010003@umts-at-fokus.de
From: sip:subs010003@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-812e
CSeq: 10 NOTIFY
Call-ID: 19aa5def-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010003@umts-at-fokus.de" id="0xb5bde88c" state="active">
<contact id="0xb5be58b4" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010003@193.175.133.125:5063; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKe8ce.a448e372.0
To: sip:subs010002@umts-at-fokus.de
From: sip:subs010002@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-2df5
CSeq: 10 NOTIFY
Call-ID: 19aa5dee-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010002@umts-at-fokus.de" id="0xb5bde6d4" state="active">
<contact id="0xb5bdbe4c" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010002@193.175.133.125:5062; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKf8ce.7ac3e62.0
To: sip:subs010003@umts-at-fokus.de
From: sip:subs010003@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-812e
CSeq: 10 NOTIFY
Call-ID: 19aa5def-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010003@umts-at-fokus.de" id="0xb5bde88c" state="active">
<contact id="0xb5be58b4" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010003@193.175.133.125:5063; transport=UDP</uri>
</contact¿

</registration¿
</reginfo¿

NOTIFY sip:subs010002@193.175.133.125:5060; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKe8ce.a448e372.0
To: sip:subs010002@umts-at-fokus.de
From: sip:subs010002@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-2df5
CSeq: 10 NOTIFY
Call-ID: 19aa5dee-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010002@umts-at-fokus.de" id="0xb5bde6d4" state="active">
<contact id="0xb5bdbe4c" state="active" event="refreshed" expires="1000000">
<uri>sip:subs010002@193.175.133.125:5062; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

NOTIFY sip:subs010003@193.175.133.125:5061; transport=UDP SIP/2.0
Via: SIP/2.0/UDP 193.175.133.122:6060; branch=z9hG4bKf8ce.7ac3e62.0
To: sip:subs010003@umts-at-fokus.de
From: sip:subs010003@umts-at-fokus.de;
tag=2e263bcb7ba52145dbe495ff44a0e0fa-812e
CSeq: 10 NOTIFY
Call-ID: 19aa5def-0@193.175.133.122
Content-Length: 368
User-Agent: Sip EXpress router(0.10.99-dev3 (i386/linux))
Route: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Event: reg
Subscription-State: active; expires=70
Content-Type: application/reginfo+xml
Content-Length: 368
<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="0" state="partial">
<registration aor="sip:subs010003@umts-at-fokus.de" id="0xb5bde88c" state="active">

<contact id=”0xb5be58b4” state=”active” event=”refreshed” expires=”1000000”>
<uri>sip:subs010003@193.175.133.125:5063; transport=UDP</uri>
</contact¿
</registration¿
</reginfo¿

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010000@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ””, uri=”sip:pcscf.umts-at-fokus.de”,
response=””
Call-ID: 6372@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 1 REGISTER
Expires: 1000000
From: ”TESTER0” <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: ”TESTER0” <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK9444
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 6372@193.175.133.125
Cseq: 1 REGISTER
From: ”TESTER0” <sip:subs010000@umts-at-fokus.de>
To: ”TESTER0” <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-74eb
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK9444
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 ”Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3”
WWW-Authenticate: Digest realm=”umts-at-fokus.de”,
nonce = ”QK7HtY9THB83h+n84USpqAAAAAAABQAAodqzPMDPYQw=”, algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010000@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ”QK7HtY9THB83h+n84USpqAAAAAAABQAAodqzPMDPYQw=”,
response=”4948d4513db75c2a”, algorithm=”AKAv1-MD5”, c

nonce = ”abcdefgh”, nc=”00000001”
Call-ID: 6372@193.175.133.125
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 2 REGISTER
Expires: 1000000
From: ”TESTER0” <sip:subs010000@umts-at-fokus.de>
Max-Forwards: 70
To: ”TESTER0” <sip:subs010000@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK9444
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 6372@193.175.133.125
Cseq: 2 REGISTER
From: ”TESTER0” <sip:subs010000@umts-at-fokus.de>
To: ”TESTER0” <sip:subs010000@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-4f14
Via: SIP/2.0/UDP 193.175.133.125:5060; branch=z9hG4bK9444
P-Associated-URI: <sip:subs010000@umts-at-fokus.de>
Contact: <sip:subs010000@193.175.133.125:5060; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 ”Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3”

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username=”usim010001@umts-at-fokus.de”, realm=”umts-at-fokus.de”,
nonce = ””, uri=”sip:pcscf.umts-at-fokus.de”,
response=””
Call-ID: 5101@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 1 REGISTER
Expires: 1000000
From: ”TESTER1” <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: ”TESTER1” <sip:subs010001@umts-at-fokus.de>

User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1117
AllowEvents: presence
Event: registration

SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: 5101@193.175.133.125
Cseq: 1 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-6324
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1117
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"
WWW-Authenticate: Digest realm="umts-at-fokus.de",
nonce = "6G6RSuz64oyFqzs0eoVpUgAAAAAABQAAAhXFynIp5po=", algorithm=AKAv1-MD5

REGISTER sip:umts-at-fokus.de SIP/2.0
Authorization: Digest username="usim010001@umts-at-fokus.de", realm="umts-at-fokus.de",
nonce = "6G6RSuz64oyFqzs0eoVpUgAAAAAABQAAAhXFynIp5po=",
response="07fbb9765116793c", algorithm="AKAv1-MD5", c
nonce = "abcdefgh", nc="00000001"
Call-ID: 5101@193.175.133.125
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000; transport=UDP
Content-Length: 0
Cseq: 2 REGISTER
Expires: 1000000
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>
Max-Forwards: 70
To: "TESTER1" <sip:subs010001@umts-at-fokus.de>
User-Agent: kphone/4.0.5_IMS
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1117
AllowEvents: presence
Event: registration

SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: 5101@193.175.133.125
Cseq: 2 REGISTER
From: "TESTER1" <sip:subs010001@umts-at-fokus.de>

To: "TESTER1" <sip:subs010001@umts-at-fokus.de>;
tag=ebdeacfaf884a9ca19931da98b80f183-7820
Via: SIP/2.0/UDP 193.175.133.125:5061; branch=z9hG4bK1117
P-Associated-URI: <sip:subs010001@umts-at-fokus.de>
Contact: <sip:subs010001@193.175.133.125:5061; transport=UDP>; expires=1000000
Path: <sip:term@pcscf.umts-at-fokus.de:4060; lr>
Service-Route: <sip:orig@scscf.umts-at-fokus.de:6060¿
Server: Sip EXpress router (0.10.99-dev3 (i386/linux))
Content-Length: 0
Warning: 392 193.175.133.122:6060 "Noisy feedback tells: pid=16787
req_src_ip = 193.175.133.122 req_src_port=5060
in_uri=sip:scscf.umts-at-fokus.de:6060 out_uri=sip:scscf.umts-at-fokus.de:6060 via_cnt=3"