The "Always Best Packet Switching" architecture for SIP-based mobile multimedia services

Vittorio Ghini, Stefano Ferretti, Fabio Panzieri

Dept. of Computer Science, University of Bologna, Italy
Email: {ghini, sferrett, panzieri}@cs.unibo.it

Abstract – **This paper presents a distributed architecture for the provision of seamless and responsive mobile multimedia services. This architecture allows its user applications to use concurrently all the wireless network interface cards (NICs) a mobile terminal is equipped with. In particular, as mobile multimedia services are usually implemented using the UDP protocol, our architecture enables the transmission of each UDP datagram through the "most suitable" (e.g., most responsive, least loaded) NIC among those available at the time a datagram is transmitted. We term this operating mode of our architecture Always Best Packet Switching (ABPS). ABPS enables the use of policies for load balancing and recovery purposes. In essence, the architecture we propose consists of the following two principal components: i) a fixed proxy server, which acts as a relay for the mobile node and enables communications from/to this node regardless of possible firewalls and NAT systems, and ii) a proxy client running in the mobile node responsible for maintaining a multi-path tunnel, constructed out of all the node's NICs, with the above mentioned fixed proxy server. We show how the architecture supports multimedia applications based on the SIP and RTP/RTCP protocols, and avoids the typical delays introduced by the two way message/response handshake of the SIP signaling protocol. Experimental results originated from the implementation of a VoIP application on top of the architecture we propose show the effectiveness of our approach.**

Keywords – **mobile multimedia applications, SIP, multi-homing, cross-layer protocols, QoS provision.**

I. Introduction

The current generation of mobile telecommunication devices offers different means of accessing the Internet today. Mobile terminals are equipped with different Network Interface Cards (NICs) that allow the device to connect through different wireless networks, such as 3G and Wi-Fi networks. These technical innovations, and the opportunities related to the applications built on top of them, raise a tremendous interest on consumers, as well as on developers and practitioners. All technology-related blogs, Web sites and magazines keep mentioning the most recent advances related to 4G mobile services, novel wireless broadband communication technologies such as WiMAX, commercial and open initiatives to provide constant network access to nomadic users (e.g. Boing [1], FONERA [2]), and so on.

Several recent works from the mobile multimedia research community have been devoted to the interoperability in heterogeneous mobile networking contexts. These works usually provide support to session control in multi-homing scenarios, by resorting to all-IP network based solutions [6, 7, 10, 31]. However, open issues that still deserve attention are related to the optimized use of the different networks available at the mobile terminal.

A Mobile Node (MN) equipped with multiple heterogeneous wireless NICs should be enabled to use the "most appropriate" NIC, based on its actual context. The notion of "most appropriate" may depend on different criteria, such as economic cost, coverage, transmission rate, QoS, security and user preferences. In other words, some mechanism is to be made responsible for selecting automatically which is the NIC to exploit when a communication starts, when a vertical handoff (i.e. a change of networking technology) occurs, or even when the communication performance, provided by the NIC in use degrades below a certain threshold (and hence, another network is to be exploited).

The scenario depicted above suggests to employ an Always Best Connected (ABC) model [12] that determines the NIC to be used as single point of access to the Internet. Periodically, the MN elects its "best" NIC, based on the access points available for that type of network and, since that instant, the MN uses the wireless network connected with the preferred NIC. If the performance of that preferred NIC degrades, the MN elects a new preferred NIC (and the wireless network to connect) that replaces the previous one by means of a handover.

Focusing on the ABC model, there are several critical points to be mentioned. A problem is that handoffs might cause severe timing overheads which strongly affect mobile applications, especially those with strict QoS requirements (e.g. VoIP and Video on Demand - VoD) that require highly responsive interactions among the involved nodes. In addition to these communication interruptions, a change of network (and thus of IP address) imposes a reconfiguration phase, with a consequent exchange of messages at the application layer. An example is the exchange of handshake messages in the Session Initiation Protocol (SIP) [6, 7].

Another issue worthy of consideration is concerned with the selection of the available NICs and the economic and energy costs associated to the communications based on some NIC. When multiple networks are available, cheaper communication technologies might be preferred, such as IEEE802.11a/b/g/n medium-distance technology family, instead of 3G long-distance technologies.

Trying to summarize, the common implementation of an ABC approach forces the use of a single network at a time, switching from one network to another only when the current network becomes unusable. While this may appear to be reasonable, owing to the mentioned timing overheads associated to the configuration of a network, it prevents the simultaneous utilization of all the MN's NICs and does not allow one to take advantages, in terms of QoS and costs, of their different characteristics.

In this work we present a distributed architecture that copes with the issues mentioned above. It principally consists of a proxy-based distributed architecture implementing an approach we named Always Best Packet Switching (ABPS). The rationale is to allow the MN to utilize simultaneously all the available NICs so as to aggregate the capabilities of each NIC it has on board. It is the responsibility of (a software module inside) the MN to select, for each datagram to be transmitted, the best NIC to use at that particular transmission time. Our architecture embodies a set of protocols, services and APIs for the support of interactive multimedia services based on SIP/RTP/RTCP, while the MN moves across heterogeneous wireless networks, even if these are administered by different organizations, and regardless of the presence of firewalls and NAT systems. The MN monitor controls and configures the available NICs in background in order to allow the concurrent exploitation of different NICs, and to switch from a network to another, transparently to the application, without incurring in costly message handshakes at the application/session layers. Finally, this architecture allows each application to develop its own policies to provide its users with the desired QoS.

We describe a prototype implementation of the architecture we have developed for the Linux operating systems. Based on it, we provide results from an experimental assessment of the implemented architecture. Throughout the paper, we also discuss a main running example to show how the ABPS approach works in presence of different NICs active at a given MN, how it automatically configures these NICs, and how it exploits them in presence of handoffs and performance variations of NICs.

This paper is structured as follows. Section 2 illustrates the state of the art concerning both SIP-based services and mobility management architectures, and highlights the main limitations that currently prevent that mobile users be provided cheaply with multimedia services. Section 3 details the basic design principle of the ABPS architecture. Section 4 describes the enhancements we introduced in SIP and RTP protocols that operates between ABPS entities only. Section 5 describes the functioning of our architecture and proposes an example of VoIP service implemented by using our architecture. Section 6 presents issues concerned with a real implementation of the system in Linux-based mobile devices. Section 7 discusses an evaluation of our system. Finally, Section 8 proposes some concluding remarks.

II. Background and State Of the Art

This section introduces the background on host mobility management for mobile multimedia services, reviews the principal architectural solutions proposed in the literature that address issues on host mobility management, and highlights their limitations.

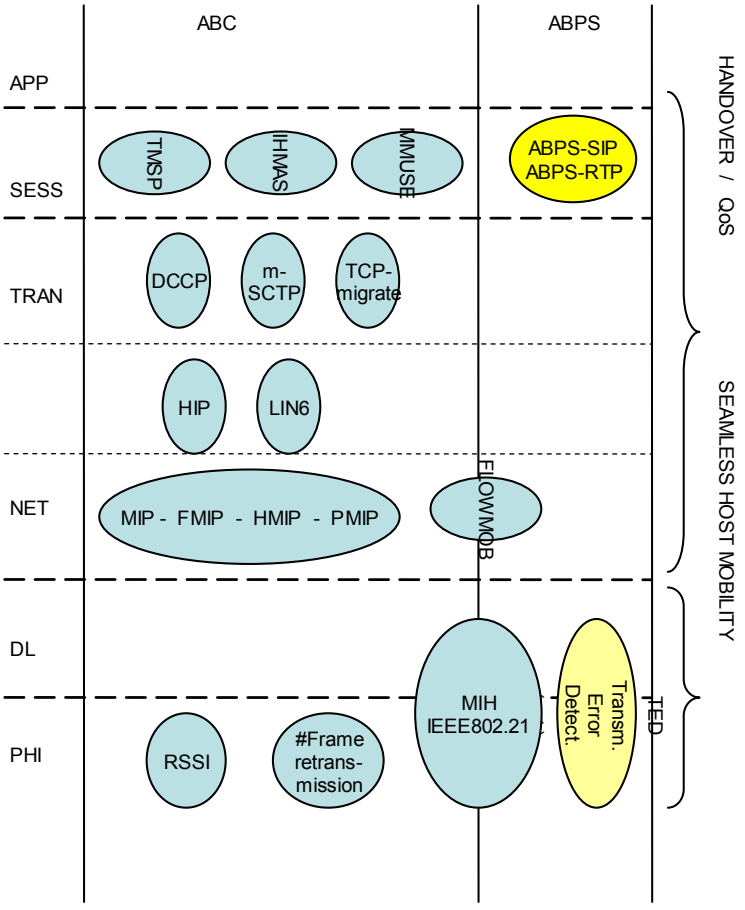

Figure 1: Architectures for mobility management.

II.1 Seamless Host Mobility Architectures

The aim of an host mobility architecture is to ensure that a MN can move across different access networks and seamlessly gain access to network services. Various sophisticated approaches aiming at providing seamless communications exploit multihoming as a basic solution to increase reliability, i.e. they assume that a MN is equipped with multiple NICs (and corresponding IP addresses) that it can use. Ideally, a seamless mobility architecture is responsible for: i) identifying univocally each given MN; ii) allowing each MN to be reachable from its correspondent nodes (CNs, for the purposes of this discussion, a CN is either a fixed or mobile node communicating with the MN); iii) monitoring the QoS provided by different networks to predict the need of a handoff and to select a new preferred NIC and its connected network access point; iv) performing the handoff seamlessly, ensuring the continuity of the communications.

In IP-based communications, the MN's IP address plays the twofold role of MN's identifier and MN locator, as it distinguishes uniquely the MN, and identifies its position in the network. When a MN joins a network, it is assigned an IP address that is valid within that network only. When that MN moves in a different network, it acquires a new IP address from the new network, loosing its identity. Hence it needs to inform CNs of its new identity and location. More generally, a mobility management architecture must ensure that two hosts can communicate even when they both move and change simultaneously their addresses.

To this aim, all the mobility management architectures adopt some mechanism that i) defines uniquely a MN identifier, independent from the location of the MN, and ii) provides a **localization service** that maintains a mapping between the MN identifier and the MN current location, even when the MN moves. The localization service is composed of a location registry, a service assumed to be always available. When a MN changes its IP address, it communicates with such registry, informing it of this change (**registration phase**). Once a CN wants to initiate a new communication with the MN, or when it wants to continue a communication with the MN that has changed its address, the CN starts a **lookup phase**, asking the location registry for the MN's current address, and then uses the obtained MN's address to contact directly the MN.

This is the general principle adopted by different architectures, and may be developed using very different algorithms and protocols. The architectural solutions examined below adopt this principle; however, they are implemented at different levels of the ISO-OSI reference model, as illustrated in Figure 1. In the rest of this Section, we shall examine these solutions in isolation, according to the ISO-OSI reference layer to which they belong.

II.2 Solutions at the Network Layer

Among the architectures working at the network layer, it is worth citing the efforts in the protocol Mobile IP version 6 (MIPv6) [14] and its optimizations, e.g. the Fast Handover Mobile IPv6 (FMIP) [17], Hierarchical Mobile IPv6 (HMIP) [26] and Proxy Mobile IPv6 (PMIP) [11]. All these approaches employ an Home Agent, i.e. an additional entity working inside the access network to which the MN belongs. This Home Agent plays the role of the location registry mentioned above, and routes datagrams towards the MN when such node is outside its "home network".

In order to work properly, the MIPv6-based approaches require that all the end-systems have IPv6 capabilities so as to insert in the IP datagrams some extension headers that transport both the MN's identifier (the home address) and the current MN address. A clear limitation is that these architectures only work on infrastructures with IPv6 capabilities. Moreover, the MIPv6 specification does not allow the simultaneous use of the multiple MN's NIC. For each given MN, the address of a single NIC is registered at the Home Agent. In addition, as demonstrated in [19], the handover latency results very high due to the numerous authentication messages.

II.3 Solution Between the Network and Transport Layers

Approaches have been presented in the literature that insert an intermediate layer between the network and transport layers of the protocol stack. This layer works on all the nodes involved in the communication, i.e., in a one-to-one communication as in VoIP, between the MN and its CN. Examples include the Host Identity Protocol (HIP) [21] and Location Independent Addressing for IPv6 (LIN6) [29]. Based on these solutions, the location registry is a DNS-like mapping function that operates as a service outside the access networks and associates host identifiers to host locations.

A limitation of these approaches is the requirement to modify the protocol stack at all end- nodes involved in a communication. While it is reasonable to ask a MN to install additional software to properly work while moving, its CN can be a fixed node that may not be interested in supporting the mobility of the MN.

II.4 Solutions at the Transport Layer

The common approach of the protocols working at the transport-layer, such as the datagram-oriented Datagram Congestion Control Protocol (DCCP) [18], the stream-oriented Mobile Stream Control Transport Protocol (m-SCTP) [22] and the TCP enhancement TCP-migrate, is very different: each given end-system plays the role of proactive location registry that directly informs the CN whenever its configuration changes. Unfortunately, this approach fails

when both the end-systems change simultaneously their IP configuration, because the two end-systems become mutually unreachable. Such a situation is not unusual, because it happens when both the end-systems are mobile ones and leave simultaneously their current network access point.

As in the previous approaches, solutions working at the transport-layer require modifications of the applications on both the MN and CN to invoke the services of the novel transport layer and to implement suitable recovery policies. This fact prevents the reuse of the existing applications.

II.5 Solutions at the Session Layer

When solutions are devised to let nodes communicate through different networks, a key role might be played by session protocols that control the dialogue between end-points and incorporate functionalities used by the localization service. Today, the Session Initiation Protocol (SIP) is the main protocol employed for controlling multimedia, multi-homed communication sessions [23, 28]. Since our approach employs SIP, as well as other solutions outlined in the rest of this section, we review now its basic properties and functioning.

SIP is a session-layer text protocol that uses a message/response handshake for signaling purposes. In particular, it is used to establish or change communication parameters such as IP addresses, protocol ports and audio/video codecs between the end-systems. The SIP specification is extensible and allows application-defined fields to be added to the SIP messages.

The SIP messages worthy of mention, in view of the discussion that follows, are *REGISTER*, *INVITE* and *re-INVITE*. The *REGISTER* message allows a given node to declare that it is available for communications; it is usually sent to a SIP server that works as the localization service. The *INVITE* message is used to establish a communication session between two nodes. Typically this message is sent from the user to the SIP server that replies with the address of the other end-node, together with some communication parameters. Then, the two end-nodes can communicate directly. A *re-INVITE* message may be used when communication parameters (such as the IP address) change.

Another important aspect is that the SIP protocol allows the presence of SIP proxies that can be transparent to the application (proxy agent) or can masquerade the end systems (back-to-back (B2B) user agent) working as an opaque relay.

Several proposals exist that employ SIP to control the session of a (multimedia) multi-homed communication. For instance, the Terminal Mobility Support Protocol (TMSP) [20] exploits an auxiliary SIP server, located outside the access networks, as location registry that maps a user identifier (e.g. ghini@cs.unibo.it) to the current user's location (the IP address of the user's MN). Each MN has a SIP user agent that sends REGISTER messages to the SIP server in order to update its current location. INVITE messages are sent to establish communications with the other nodes.

Similarly, [31] presents an architecture able to manage vertical handoffs, by using a SIP-based approach. The scheme complies to the IP Multimedia Subsystem (IMS), a standardized overlay architecture for session control, authentication, authorization and accountability in all-IP networks [7]. Another related proposal is that presented in [15], that only supports vertical handoffs from 3G networks to a Wi-Fi.

The session-layer solutions seem to be not efficient as they invoke an external localization service when an IP reconfiguration occurs. In particular, the SIP-based services introduce an additional delay due to their message/response behavior; in case of reconfiguration the MN interrupts the communication, sends a SIP signaling message to the CN and waits for the response before resuming the transmission. With this in view, the IHMAS work presented in [6] provides a solution to minimize handoff delays by exploiting a SIP-based, IMS compliant proactive mechanism that performs registration and renegotiation phases for novel connections while keeping the media flows active over old connections, if these are available.

II.6 Coping with NAT and Firewall Systems

Most of the previously described mobility management solutions do not take into account the possible presence of firewall and NAT systems in between end-nodes involved in a (multimedia) communication. Usually, in order to overcome this limitation, the applications resort to services offered by external STUN [24] or TURN [25] servers. The most restrictive firewalls require an intermediate application-layer relay server between the MN and its CN.

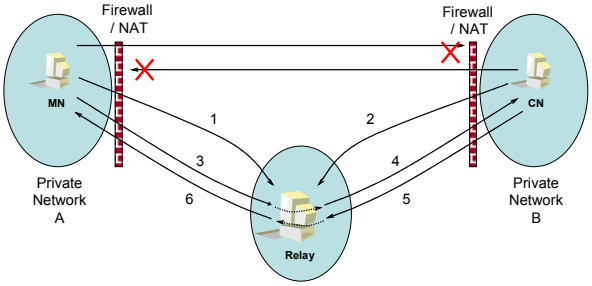

Figure 2: The data relay allows the undirected communication between two nodes behind different firewalls or NAT systems.

An example of system taking into account the possible presence of firewall and NAT systems is MMUSE [27]. Such system requires that an auxiliary SIP server (namely, the Session Border Controller, SBC) be located at the edge of the autonomous system inside which the MN will move. This autonomous system may be composed of several subnets using heterogeneous network technologies. While the MN moves across the subnets, each subnet provides the MN with a different IP address. The SBC aggregates the functionalities of SIP and RTP proxy, firewall and NAT systems, and intercepts the communications that enter and leave the network, in particular the SIP messages between the MN and its CN outside the network edge. Based on the outgoing SIP messages, the SBC sets up the firewall rules that allow the subsequent SIP, RTP and RTCP communications between MN and CN. Moreover, when the MN moves to a different subnet and changes its IP address, the SBC modifies the outgoing datagram in order to hide to the CN the current location of the MN.

The main limitation of the MMUSE solution is that the MN traffic needs to flow always through a given SBC that resides in the edge of the network. This implies that the MN may move only inside a given autonomous system, but it cannot move across different networks administered by different organizations.

II.7 Simultaneous Use of Different Network Interface Cards

It is worth to point out that all the previously described approaches do not allow the simultaneous utilization of all the NIC available on the MN. In other words, all the packets are transmitted using a NIC only.

An extension of MIPv6, namely Multiple Care of Address (MCoA), allows a MN to register its multiple IP addresses with its Home Agent. The Flow Mobility technique (FlowMob) described in [30] uses this extension to allow the MN to separate its outgoing traffic in different flows based on the protocols, port numbers and IP addresses, and forward each given flow using a selected NIC. This technique allows a flow-based switching through the MN's NICs. This approach shares the same limitations as the other MIPv6-based approaches: it needs that the network infrastructures be modified so as to add IPv6 capabilities, and the handover latency results very high owing to the large number of authentication messages.

II.8 Comparison between the State of the Art and Our Architecture

In contrast with the previously presented approaches, our ABPS-SIP/RTP architecture enables the transmission of each and every datagram through the most appropriate NIC. Our ABPS-SIP/RTP architecture operates at the session-layer using an auxiliary proxy server and allowing the MN to move across different autonomous systems. A cross-layer technique is employed to monitor all the concurrent NICs which are available, their performances and those that become active (inactive); based on their current status, automatic reconfiguration is performed in the MN.

Our solution uses SIP-compliant proxy servers which interact with additional software modules, installed on the proxies, working between the network and the transport layer; these modules are in charge of managing the application-layer data flows enabling their transmission through different NICs. In practice, our proxy decides on a per-packet basis which is the best NIC to use to transmit the data. We will show that the use of such a proxy avoids in a transparent way the typical delays introduced by the two way message/response handshake of the SIP signaling messages.

It is important to notice that all the approaches we have introduced earlier usually consider the problem of changing a NIC in use as soon as it becomes unavailable. Thus, the decision to change communication technology is not taken based on some particular QoS metric; rather, it is taken based on the failure of the currently adopted NIC. Vice versa, our approach takes into consideration QoS metrics to identify the best NIC to use at any given moment. In particular, a cross-layer mechanism (namely, Transmission Error Detector, TED) has been developed, that provides the applications with information about the successful (unsuccessful) datagram transmissions through a given wireless access point. At the moment, the implementation of TED is for WiFi NIC only.

Currently, to determine the "best" NIC to be used to transmit a given packet, the system adopts a simple "WiFi first" hybrid metric that takes into account both the energy consumption and the packet loss rate. In particular, at the MN, if all the available NICs are correctly configured and working, the proxy client selects the WiFi NIC, that consumes expensive than other (cellular) technologies, it sends the packet and waits for a few milliseconds to receive from the local TED a notification of the successful or unsuccessful packet transmission to the access point. In case of transmission error the packet is sent again using another (e.g. UMTS) NIC. The proxy server, instead, saves the sender IP address of the last-sent datagram received from the proxy client, and sends each packet directed to the proxy client exactly to that last saved address. If there is no traffic from the proxy client to the proxy server, the proxy client periodically sends to the proxy server an empty ABPS-SIP message in order to: i) keep open the path through the firewalls between proxy client and proxy server, and ii) implicitly give to the proxy server the IP address of the selected NIC.

The software module is extensible, in the sense that different metrics can be employed to decide which is the best NIC to use. Hence, the approach can be configured so as to take into consideration classic QoS criteria of network performance, such as costs, bandwidth, quality of the signal (i.e. Received Signal Strength Indicator - RSSI), indicators measured using the IEEE 802.21 Media Independent Handover [3] standard, or other solutions [16].

In essence, the advantages provided by our ABPS-SIP/RTP system are the following:

– our approach perfectly works over all IP based networks and does not require any modifications of the actual infrastructures;

– it is SIP compliant;

– it avoids delays occurring in classic SIP-based approaches when the MN changes its preferred NIC or its configuration. In this case, in fact, other schemes employ reconfiguration phases based on the exchange of INVITE messages, while our approach avoids this additional message exchange;

– it supports RTP-based applications such as VoIP and VOD services;

– it can cope with vertical handoffs, without introducing any additional delay during the passage from the use of a NIC to the next one, since as soon as this becomes available it is promptly configured to work;

– it optimizes the use of NICs, by deciding on a per-packet basis which is the best interface to be used, based on the monitored performances of the available networks and on the employed QoS metrics;

– it overcomes the presence of NATs and firewalls.

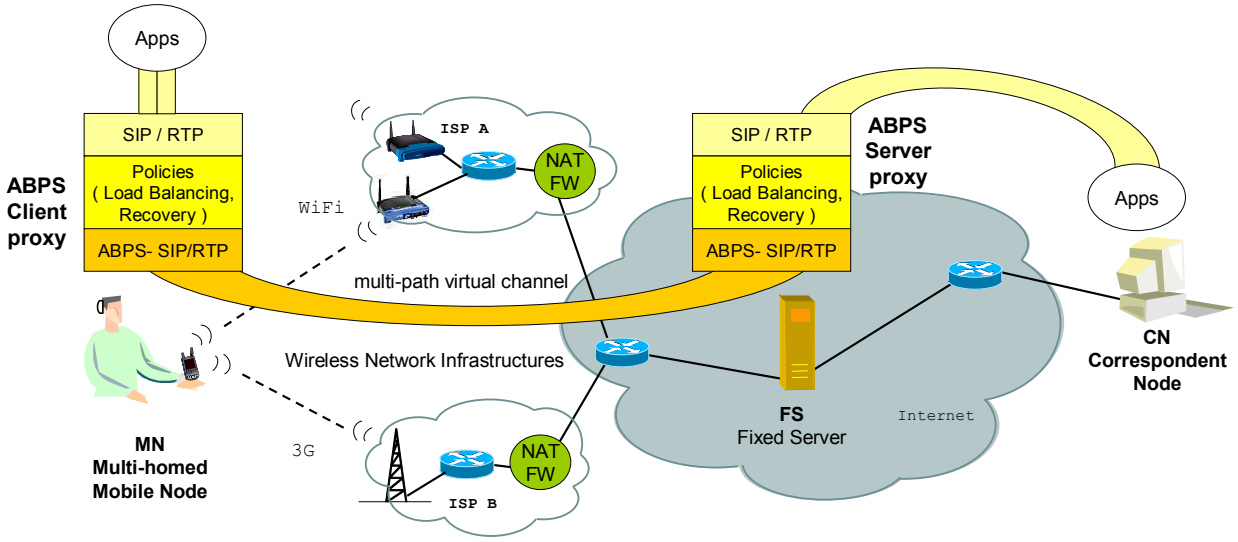

Figure 3: The ABPS architecture.

III. Proposed Architecture

The ABPS-SIP/RTP architecture we propose (see Figure 3) has been designed in order to meet the following three principal requirements:

- **To guarantee communication-continuity of service**. Our architecture incorporates a public infrastructure-independent anchor service (i.e, the so-called ABPS proxy server). This is an external service, independent of the access networks, which works in a fixed server (FS) with a public static IP address located outside any firewall and NAT system. The anchor service is essentially a modified SIP and RTP Proxy that, as illustrated in the Figure 3, operates as an intermediate data relay between the MN and its CN so as to maintain seamless communications. Each MN implements a modified SIP and RTP proxy agent (i.e., the so-called ABPS proxy client) that maintains a seamless multi-path communication channel with the ABPS proxy server, thus overcoming the presence of NAT and firewall systems and allowing the interoperability with the standard SIP and RTP based applications working on the MN.
- **To enable simultaneous use of all NICs**. Our ABPS architecture allows the MN's applications to use simultaneously all the available NICs, differentiating the choice of the NIC from datagram to datagram and introducing the ability to switch to the most suitable path depending on the characteristics of the datagram. The mechanism can retransmit lost datagrams through a NIC different from the one exploited before, without incurring in message duplication at the application level. The APBS client and server proxies collaborate and exploit policies for load balancing and recovery, hence maximizing the throughput, decreasing the loss rate and the economic costs. The APBS model enhances the traditional ABC model that requires that the MN identifies the best wireless network among those available through the NICs, and from that time on, uses that network as single point of Internet access until performance degrades.
- **To minimize delay via signaling shortcuts**. With the aim of minimizing delays upon changes of the communication parameters, the ABPS client and server proxies utilize some extensions of the SIP and RTP protocols that implement shortcuts of the commonly used signaling messages. In particular, these extensions introduce in the SIP and RTP packets some additional fields necessary to identify univocally and securely the sender of a message even when the sender changes its IP address. In SIP messages both a Proxy Identifier and a Fingerprint are used. The former univocally identifies the sender of the message. The latter ensures the integrity of the message. Such information is needed since all the SIP messages are multiplexed on a single port of each ABPS proxy. As to RTP flows, a standard

extension called Secure Real Time Protocol (SRTP) is used that includes an Authentication Tag that verifies the integrity of the messages. The SRTP messages do not include an identifier of the sender because each RTP flow uses its dedicated UDP port on each side of the flow. These ports are configured at the setup of the VoIP call and each proxy maintains a map that associates local UDP ports and sender at the other side. Thus, the receiver of a SRTP message deduces the identity of the sender using the port number on which the message has been received and verifies this identity by means of the Authentication Tag. These two extensions allow to identify the sender of a SIP and SRTP messages. In this way, the ABPS client can autonomously decide to send a given RTP datagram through a NIC, different from the one previously used, without using a preparatory SIP message that informs the SIP proxy server of the sender's IP address change. This represents the instantiation at the session layer of the ABPS principle.

For the sake of simplicity, Figure 3 shows only a single ABPS proxy server. In a real world scenario scalability issues can be taken into account deploying a set of geographically distributed ABPS proxy server that can deal effectively with requests coming from a large number of mobile users. To this end, cloud based technologies can be employed in order to dynamically allocate (deallocate) novel resources (i.e. ABPS server proxies), based on the number of active users [13]. Moreover, each MN may connect to a different ABPS proxy server so that the SIP/RTP communications between two MNs can flow through two ABPS server proxies. Hence, also the CN may be a mobile node. In this case, the techniques employed for the MN can be utilized on this node.

III.1 VoIP Data flows on ABPS architecture.

The ABPS server takes the place of the MN by exposing a SIP service to which SIP requests directed to the MN can be delivered. Moreover, while the MN performs a VoIP call the proxy server exposes a couple of UDP ports that receive RTP and RTCP messages directed to the MN. The ABPS proxy server uses a single UDP port, known by the client proxies, to receive (and send) ABPS-SIP messages from all the client proxies, and SIP messages from MNs, CNs and the SIP server. Thus ABPS-SIP messages include a field (termed ProxyclientID) that identifies the proxy client to which the messages belong. Instead, each RTP or RTCP flow uses a dedicated port pair on both the end sides thus the sender of a received RTP or RTCP packet is identified by means of the port on which the packet is received. At the MN, the management of multiple NICs imposes a little but substantial difference concerning the ports used for the RTP flows between client and server proxies: in fact, for each RTP flow the proxy client creates a different socket for each working NIC at the MN, bounds each socket to a NIC and assigns a different UDP port to each socket. This particular set of operations depend on the technique implemented to allow the selection of the specific NIC through which a given RTP packet will be sent (this technique will be explained in detail in the following Section VI).

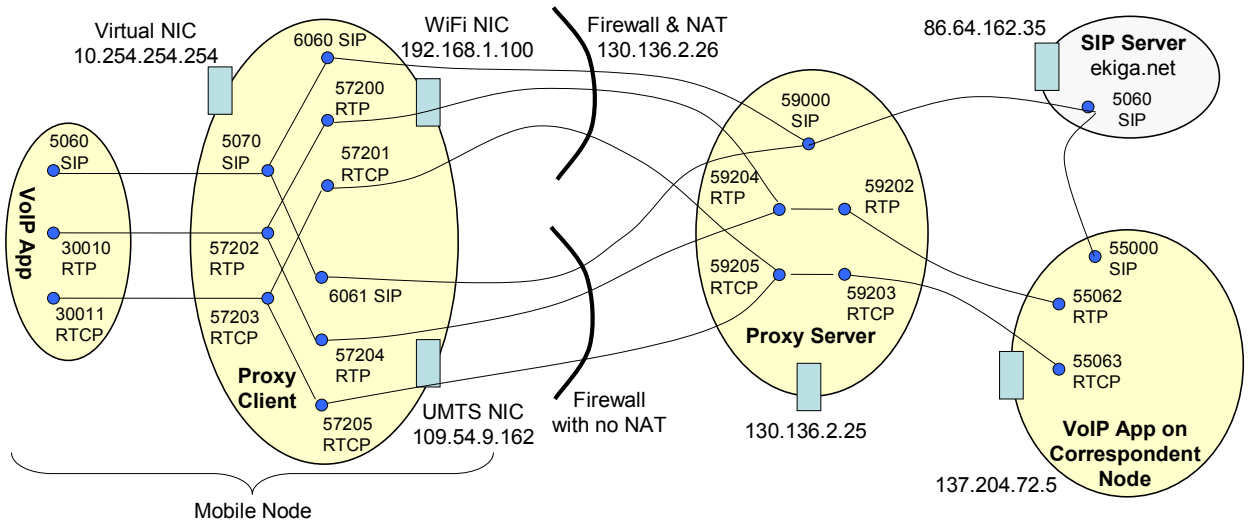

Figure 4: The data flows.

The Figure 4 above illustrates the complexity of the communication flows among a proxy server, a MN and a CN after a VoIP call has been established between a MN and a CN. This example describes the true experimental scenario in which we tested our architecture; the MN holds a WiFi NIC and a UMTS NIC, to which the IP addresses respectively 192.168.1.100 and 109.54.9.162 have been dynamically assigned. The WiFi NIC operates in a class-C private network protected by a symmetric firewall and NAT that exposes the public IP address 130.136.2.26. Instead, the UMTS NIC operates in a public network and is protected by a symmetric firewall with no network address translation. Moreover, on the MN we configured a virtual NIC with address 10.254.254.254 that operates as default gateway so as to hide network reconfigurations to the local applications. The correspondent node is reachable at the address 137.204.72.5. On the correspondent node operates a user fab11@ekiga.net that is registered at the SIP server of the domain ekiga.net. This

SIP server is reachable at the address 86.64.162.35. On the MN operates a user vic00@ekiga.net registered at the same SIP server. Each entity (i.e. MN VoIP application, proxy server, CN, SIP server), with the exception of the proxy client, use a UDP port for all the SIP messages (namely port numbers 5060, 59000, 55000, 5060 in Figure 4). The proxy client uses an UDP port (5070) to exchange SIP messages with the local MN VoIP application, and a set of UDP ports, one for each working NIC, for the SIP communications with the proxy server. In the example, the proxy client employs the port number 6060 with the WiFi NIC and the port number 6061 with the UMTS NIC. Differently from SIP messages, RTP and RTCP flows use dedicated UDP ports that are created during the INVITE procedure that establishes a two-way VoIP call between MN and CN. It is worth to point out that voice communication between MN and CN flows through the proxy server so that the voice communication is split in three hops: i) the first hop is a local one within the MN and connects the VoIP application (ports 30010 for RTP and 30011 for RTCP) and the proxy client (port 57202 for RTP and 57203 for RTCP); ii) the second hop connects proxy client and proxy server through two NIC: at the proxy client side two port (57200 and 57204) are dedicated to RTP messages exchanged with the proxy server on its port 59204; analogously for RTCP messages, two ports (57201 and 57205) are used on proxy client and another (59205) on proxy server; iii) the third hop connects the proxy server and the CN using the port pair 59202 for RTP and 59203 for RTCP on proxy server and the pair 55062 and 55063 on the CN.

IV. ABPS Extensions for SIP and RTP

The ABPS extensions designed for SIP and RTP aim at both identifying univocally the sender of a message even when it changes its IP address, and providing authentication, integrity, and optionally confidentiality between the ABPS client and server entities. We have designed two different solutions for SIP and RTP, due to their different requirements. In particular, since multimedia applications exchange a large number of small RTP datagrams, it is essential to limit both the size of the datagram and the CPU load for the scope of datagram encryption and decryption. In particular, the two solutions for SIP and RTP, can be described as follows.

- **ABPS-SIP**. The ABPS-SIP is an extension of SIP that establishes a secure SIP channel between the two ends of the multi-path virtual channel, i.e., the ABPS proxy client and server, and authenticates the SIP messages using a digest scheme. During the off-line configuration phase, each proxy client and server agree on a so called pre-shared key that they will use for secure communication purposes and that they will never transmit. Following the directives of the National Institute of Standards and Technology [8], the pre-shared key is used without transmitting it in a preliminary authentication phase in which authentication messages, and suitable *Key Derivation Functions* based on HMAC cryptographic hashing (keyed-Hash Message Authentication Code), generate a temporary one-session cryptographic key, shared between the client and server. For each given SIP message to be sent this key is used, together with HMAC functions, to generate a fingerprint of that message. For each SIP message to be sent, the ABPS-SIP adds a variable size textual header. This header transports the UID of the sender, a sequential number, and the fingerprint of the complete message (including the additional header but excluding the fingerprint itself). The UID is the identity of the sender, the sequence number is used to avoid replay-attach, and the fingerprint allows the receiver to verify the integrity of the message. In this way, ABPS-SIP ensures authentication and integrity in SIP messages.
- **ABPS-RTP**. The ABPS-RTP is designed as a set of application-layer standard protocols that cooperate in providing a secure RTP channel between two end systems. In particular, ABPS-RTP provides authentication on a RTP flow basis regardless of the IP address of the sender, integrity and optional confidentiality. In other words, the receiver distinguishes between different RTP flows and identifies the sender of a RTP message even when it changes its IP address. Thus, ABPS-RTP allows switching dynamically RTP messages through all the available paths between the ABPS client and server. To deliver messages of a given RTP flow, ABPS-RTP uses a standard extension of RTP, the Secure Real-time Transport Protocol (SRTP) [5]. SRTP adds a header with an authentication tag calculated using an encryption key that is created for that RTP flow only. Thus, before setting up each given protected RTP flow between two end systems, the SRTP protocol requires a preliminary phase in which a key-agreement protocol generates that encryption key. SRTP does not specify a particular key-agreement protocol among those available, such as MIKEY, SDES, DTLS, ZRTP. However, due to its performance we choose the key-agreement protocol named ZRTP [32]. ZRTP begins its operations when the two end-systems exchange the INVITE SIP messages to set up a given RTP flow: ZRTP calculates a hash value of a subsequent ZRTP message and inserts it as a field of the SDP body that describes the RTP flow. In this phase, since the conventional SIP does not provide authentication and integrity, it is possible that an attacker substitutes the end-system that wishes to setup the RTP flow. Our ABPS architecture overcomes this limit enabling INVITE messages to be delivered using ABPS-SIP that provides authentication and integrity. Note that SRTP introduces an overhead of 4 bytes to each RTP message.

The pair ABPS-SIP and ABPS-RTP works as follows: 1) the off-line configuration phase sets up the pre-shared key of the client and server. 2) When the MN starts up and configures its NICs, the client uses the pre-shared key to set up a one-session cryptographic key for the SIP session. 3) Each subsequent SIP message to be sent includes UID, sequence number, and fingerprint. 4) For each media flow to be established, the client builds a ZRTP Hello message with ZRTP parameters and options, and calculates the hash value of this message (i.e., Hello Hash). 5) The client sends the authenticated SIP INVITE message to the server with a SDP body that includes the Hello Hash. 6) The client receives

the authenticated SIP INVITE response with the Hello Hash of the server. 7) The client sends the ZRTP Hello message previously built to the server that computes the hash value of the message and compares it with the previously received Hello Hash. Attacker's messages are discarded. 8) The server replies sending its ZRTP Hello previously built to the client that computes the hash value of the message and compares it with the received Hello Hash. Attacker's messages are discarded. 9) The Diffie-Hellman exchange between the client and server sets-up the ZRTP encryption key. 10) The Client and server generate and exchange the SRTP encryption key. Finally, 11) the client and server start the transmission of the media encapsulating the data in SRTP messages.

V. System Setup and Calls Lifecycle

Basically, the interaction among the different components of out ABPS architecture passes through the following five different phases: i) an initialization phase (**proxy client registration phase**) between the APBS proxy client on the MN and the APBS proxy server on the fixed server that sets up a secure multi-path virtual channel between the two proxies, ii) a **Client registration phase** in which the user of the VoIP application on MN informs an external SIP server (for instance ekiga.net) that the user is available to perform VoIP calls; the proxy server itself may play the role of SIP server; iii) a **call setup phase** to begin a communication with a CN; this phase may be initiated by both the MN and the CN; iv) the communication phase where voice data are exchanged; and v) a possible update phase that may happen when configuration changes occur at the MN. In the following, these phases are discussed in isolation.

V.1 Initialization (Proxy Client Registration Phase)

A preliminary one-time-only off-line-registration procedure is in charge of registering the mobile user with the APBS proxy server. To this end, a Identifier of the proxy client (ProxyclientID) and a cryptographic key are shared between the MN and the ABPS proxy server, to be used when the MN accesses the system and registers to its ABPS server.

When the MN starts up and configures its NICs, the ABPS proxy client sets up a temporary security context with the ABPS proxy server using the pre-shared key. In this way, we construct a one-session cryptographic key shared between the ABPS client and server proxies without transmitting the pre-shared key through the network, thus meeting security requirements. After this phase, the MN owns all the necessary information to construct and correctly understand the messages of the ABPS-SIP/RTP protocols. In particular, the ABPS proxy server can identify the SIP or RTP messages being sent by the ABPS proxy client when the MN changes the IP address of all NICs.

After this, the ABPS proxy client sets up the multi-path channel with the ABPS proxy server. A main aspect to consider is that the MN has multiple IP addresses; however, some of the employed networks may deploy firewalls or NAT systems that can change the address contained in IP datagrams, i.e. the ABPS proxy server may receive datagrams with an IP sender address which is different than that inserted by the ABPS proxy client. To sort out this problem, we use the extension to the SIP for symmetric response routing, which defines a new parameter for the Via header field, called "*rport*". This parameter allows a client to request that the server sends the response back to the source IP address and port from which the request originated (see RFC 3581). In essence, this option tells the server to consider as IP addresses those contained in the IP header field, and not those specified as data in the SIP messages.

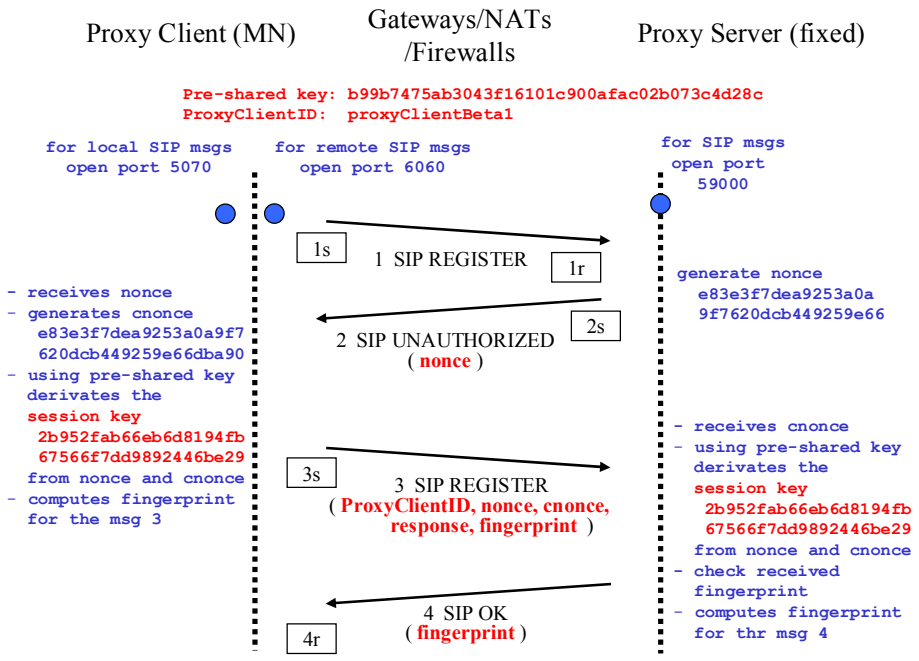

Figure 5: The message exchange during the Initialization phase.

Figure 6: Content of SIP messages in the Initialization phase.

Finally, the ABPS proxy client sends to the ABPS proxy server a REGISTER ABPS-SIP message. The figures 5 and 6 show the interaction between proxy client and proxy server during the proxy client registration, and the SIP messages that implement this phase, respectively. The strings (1s, 1r, 2s, 3s, 4r) enclosed in the rectangles near each SIP message allow the reader to verify. In Figure 6, the content of that message. In the figures, messages where specifically delivered through the WiFi NIC; it is possible to observe that the NAT of the private network modifies the IP addresses of the datagrams; in particular, in Figure 6, we show the first REGISTER SIP message before (1s, message 1 when it has been sent) and after (1r, message 1 when it has been received) the transmission through the NAT; as shown in this Figure, the NAT changed the IP sender address from 192.168.1.100 to 130.136.2.26.

V.2 Client Registration Phase

Typically, the mobile user makes VoIP calls through a SIP server, using a SIP account such as vic00@ekiga.net. A SIP server has the main role of maintaining both the status (online or not) and the contact information (actual IP address and SIP port) of the SIP client of its users. In this way, another user may ask the SIP server to get information and locate a given user. A user gives to the SIP server its contact information through a message exchange, called **client registration phase**, based on the REGISTER SIP message. In the ABPS architecture, the mobile user does not contact directly the SIP server but the proxy client. This contacts the proxy server that finally communicates with the SIP server. The user contact information, sent by the proxy server to the SIP server, consists of the IP address and SIP port of the proxy server itself; this allows the proxy server to take the place of the mobile user, from the standpoint of the SIP server, and hides the effective location of the mobile user. The ABPS client registration phase consists of multiple SIP messages and is depicted in Figure 7. As usual, the messages between the proxy client and the proxy server include the fingerprint field. Such information is exploited for security reasons and, indirectly, identifies the address of the MN. Figure 8, reports the content of the REGISTER SIP message (the message number 11) sent from the proxy server to the SIP server: note that the contact information (the field “Contact: <sip:vic00@130.136.2.25:59000>\”) of the user on the mobile node (vic00@ekiga.net) consists of the IP address and the SIP port of the proxy server. For the sake of completeness, in the Appendix A Section we report all the messages of the client registration phase.

Finally, it is worth to point out that when the proxy server receives a SIP message from the proxy client and checks successfully the identity of the sender by means of the fingerprint, the proxy server collects the IP address and the port of the sender in order to update the information about the localization of the proxy client.

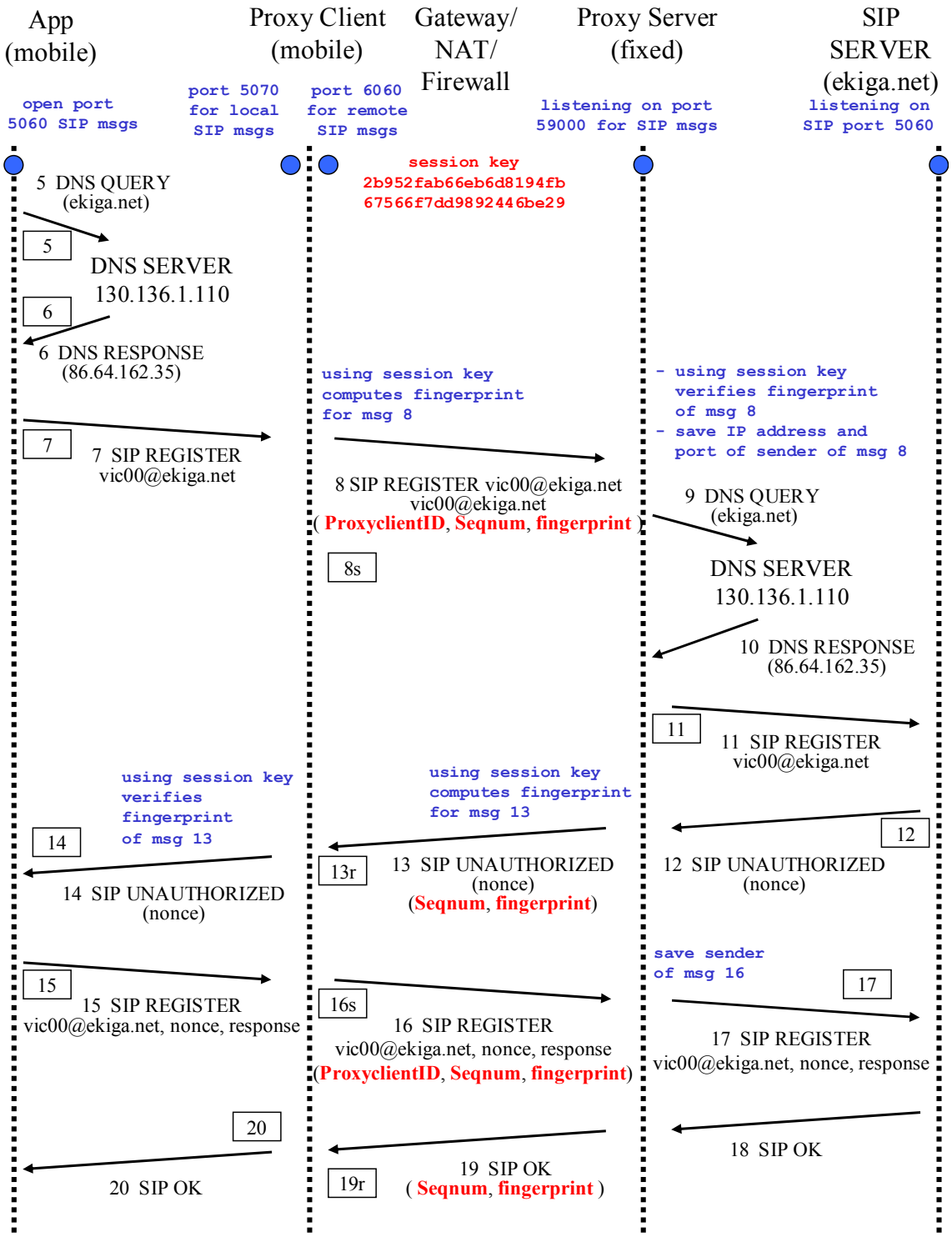

Figure 7: SIP messages of the Client Registration phase.

```
11

IP  -  Src 130.136.2.25  Dst 86.64.162.35  Len 637
UDP -  Src 59000  Dst 5060   Len 617
SIP -  REGISTER sip:ekiga.net SIP/2.0\r\n
    Via: SIP/2.0/UDP 130.136.2.25:59000;rport;
        branch=z9hG4bKe057af0ec22995f436e1c15196bf5719\r\n
    Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
        branch=z9hG4bKc5b2506810b25d2bcae49b3fff76a5f1\r\n
    Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
        branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
    From: <sip:vic00@ekiga.net>;
        tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
    To: <sip:vic00@ekiga.net>\r\n
    Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
    CSeq: 52060 REGISTER\r\n
    Contact: <sip:vic00@130.136.2.25:59000>\r\n
    Max-forwards: 68\r\n
    User-agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
    Expires: 100\r\n
    Content-Length: 0\r\n
    \r\n
```

Figure 8: REGISTER SIP messages sent from the proxy server to the SIP server.

V.3 Call Setup

In order to initiate a RTP-based communication with a given correspondent user at a given CN, an application at the MN starts a message exchange that involves the two end systems, the two ABPS proxies and the SIP server of the domain of the correspondent user at the CN. Figure 9 depicts the complete message exchange that sets up a VoIP call from a MN to a CN. The content of the messages labeled with a string included into a rectangle (for instance, 30r) are reported in the Appendix B Section. In addition, some of the most representative messages are summarized in figure 10, 11 and 12.

The application at the MN opens two local UDP ports to receive the RTP and RTCP messages; then, it delivers to the ABPS proxy client a SIP INVITE message directed to the SIP server of the domain of the user working on CN. The INVITE message includes a Session Description Protocol (SDP) body that contains the port numbers. In addiction, the SDP message includes different options for audio/video coding. The ABPS proxy client delivers the messages to the ABPS proxy server that relays the message to the SIP server that, in turn, relays the message to the CN. While the SIP server performs only the relaying of the message, both the client and server proxies, before transmitting the received messages to the next entity (i.e. the proxy server and the SIP server, respectively) execute the following four operations in order to obtain from the next entity both the SIP response and the RTP data: they

1) open a UDP port to receive RTP messages from the next entity and another one for RTCP messages;

2) insert in the SIP message their own IP address as an additional field;

3) substitute, in the Session Description Protocol (SDP) message included in the SIP message body, the number of the port just opened as port number of the sender.

4) send the SIP messages to the next entity and wait for the response.

When the CN receives the request and the user accepts the call request, the CN operates as follows: it opens two UDP ports to be used for RTP and RTCP message exchange with the proxy server, selects the suitable options among those proposed in the received SDP message, constructs a SDP message that includes the ports just opened and the selected options, embodies the SDP message into a SIP OK message, sends this response back to the SIP server and waits for a ACK SIP message that confirm the call has been established.

The SIP server relays the response received from the CN back to the proxy server.

When a proxy receives the response from the next entity, it opens two UDP ports to be used for RTP and RTCP messages exchange with the previous entity, modifies the received SDP message by inserting the ports just opened, sends the response back to the previous entity and waits for a ACK SIP message that confirms the call has been established.

In the particular case in which the two involved entities are the proxy client and proxy server, the procedure is a little bit more complex because the two proxies need to construct an encryption key that will be used to compute and verify the Authentication Tag of each SRTP message that embodies an RTP message. The procedures in based on the ZRTP protocol that implements the Diffie-Elmann algorithm using 10 messages, characterized in the Figure 9 with the labels from 31 to 35 and from 41 to 45.

After the ZRTP procedure has been completed, the two proxies use the encryption key to generate (and verify) the Authentication Tag that identifies the sender and ensures the SRTP messages where not modified in the transmission between the two proxies. In the Figure 9, the messages with label from 46 to 51 represent the initial voice packet of the VoIP call just established.

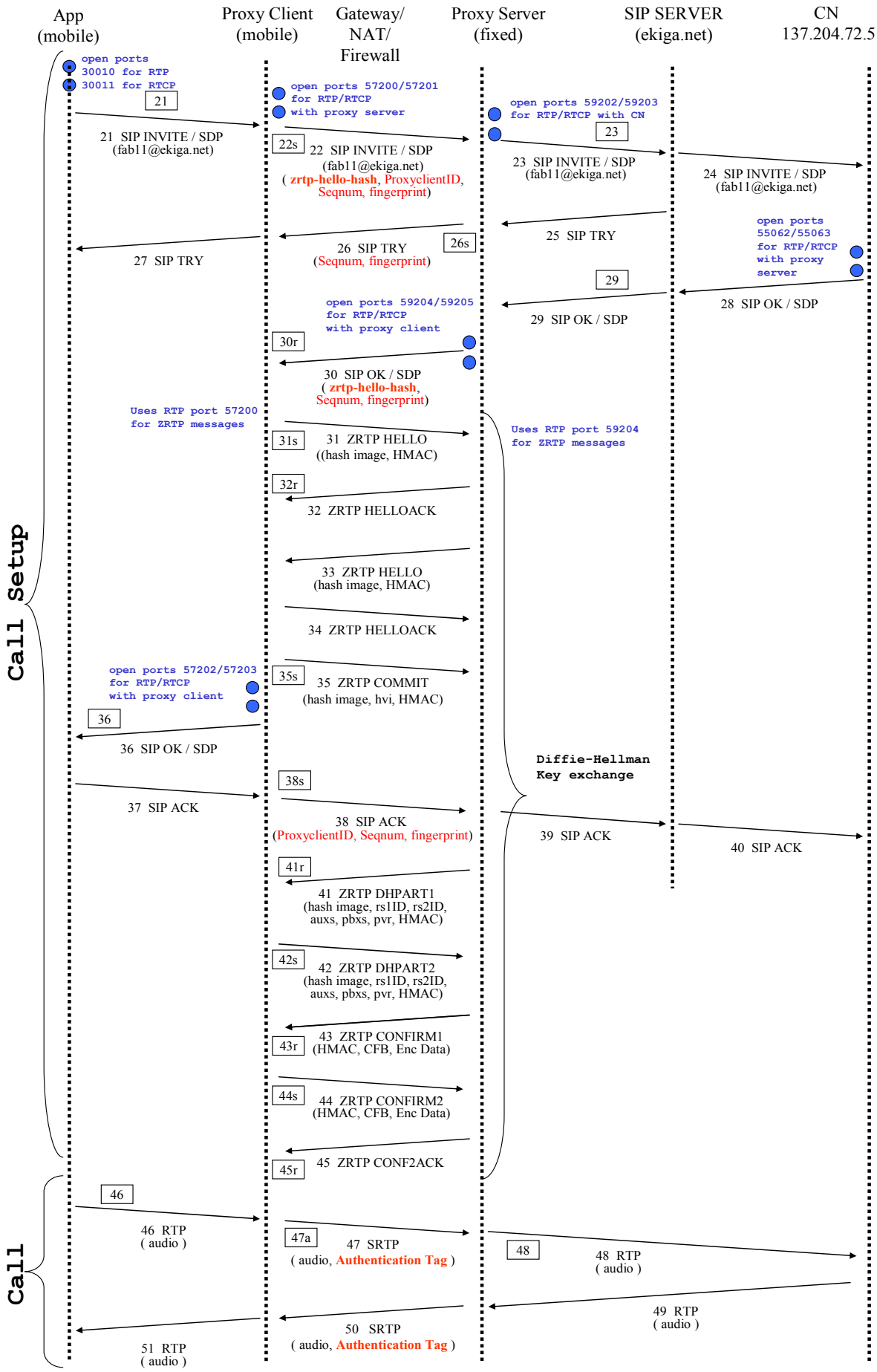

Figure 9: Messages of the Call Setup phase and Call.

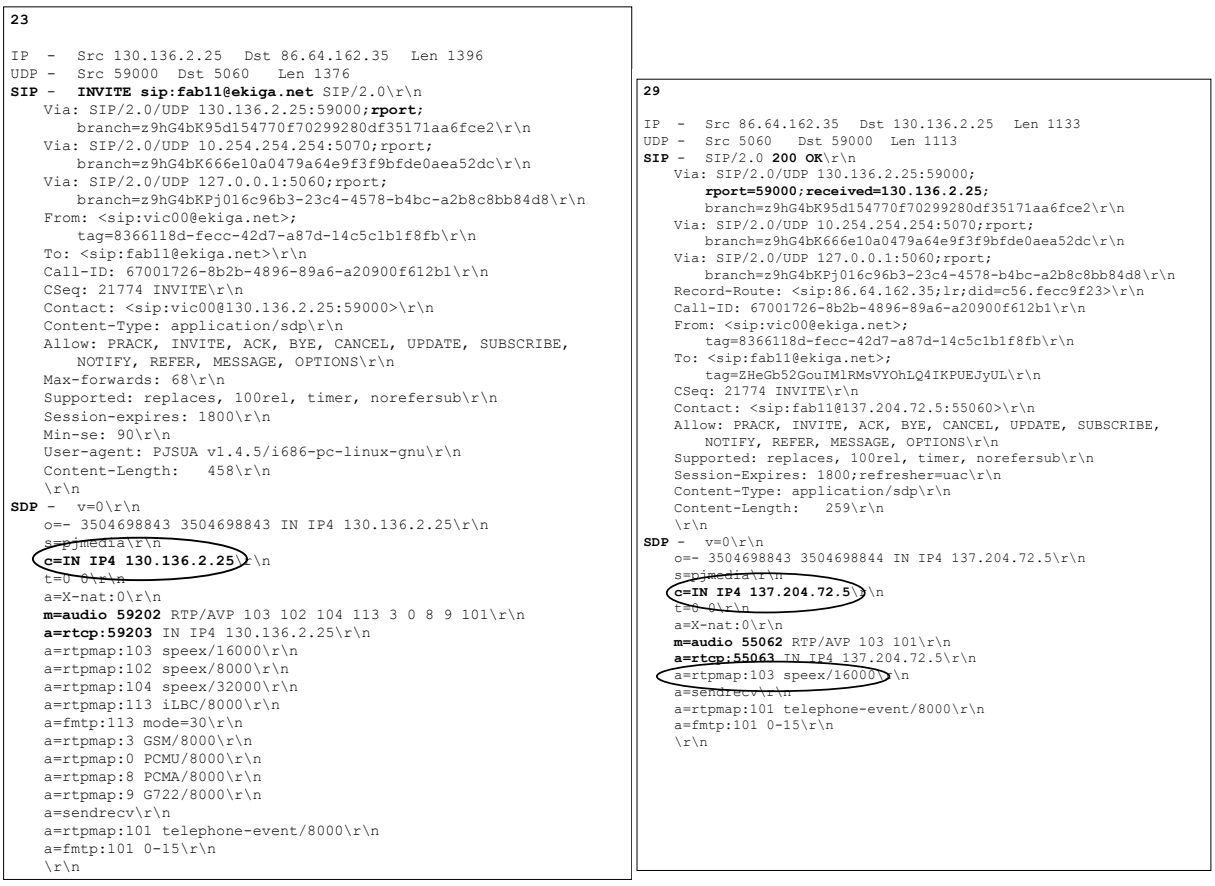

Figure 10: The INVITE request SIP message (left) sent from the proxy server to the SIP server and the OK response (right).

.

V.4 Communication

After the setup phase, RTP data flow between the MN and CN via three consecutive bidirectional paths: i) the path between the MN's application and the ABPS proxy client, where the communication is based on the RTP protocol, ii) the multi-path between the ABPS proxy client and ABPS proxy server (where data may pass through different networks linking the MN with the outside world), in which the ABPS extensions introduced for both the RTP and SIP protocols are deployed, iii) the path between the ABPS proxy server and the CN, based on RTP. We recall that in the scenario we are considering, the critical path in the communication is the one that connects the ABPS client and sever proxies, since wireless networks are used here, whose configuration may change due to the mobile nature of the user.

Due to the fact that multiple networks can be used during the communication, it is possible that messages be received out of order. This problem is solved by the fact that ABPS-SIP and RTP messages include an increasing sequential number, which is used to order (or discard) late messages and discard those that for some reason have been duplicated.

RTP data flow across the multi-path wireless channel between the ABPS client and server proxies, under the control of a software module (i.e. Policies sub-layer, see Figure 4) working at the two nodes. The Policies sub-layers use the ABPS extensions of SIP and RTP to implement QoS mechanisms such as packet-based NIC selection, packet loss detection, early retransmission and load balancing on the wireless channels. Different settings corresponding to different QoS requirements might change the configuration of the multi-path virtual channel. For instance, the concurrent use of mechanisms for determining packet loss detection and early retransmission meets the requirements of interactive applications such as VoIP; instead, a mechanism that privileges a dynamic packet-based NIC selection and load balancing meets the requirements of bandwidth consuming applications, such as VoD.

Our Policies sub-layer implements these QoS-enhanced multi-path virtual channels and exposes each of them as a pair of SIP and RTP ports (on the UDP protocol) to the application on its MN. Thus, each MN's application may select its preferred QoS-enhanced virtual channel simply selecting as SIP outgoing proxy port the corresponding SIP port. This allows each given SIP-based application running on the MN to inter-operate with the ABPS architecture without any modifications.

V.5 Configuration Update

It is worth mentioning that during the communication our approach avoids additional delays occurring in other SIP based solutions when the MN changes its IP configuration, e.g. due to a link-layer handoff, or when, for some reason,

RTP messages must be transmitted through a different NIC. According to these conventional SIP-based solutions, in fact, when one of these two mentioned situations occurs, the transmission of RTP data is paused while the two end-nodes exchange a re-INVITE SIP message to setup the new communication parameters (see Figure 12).

In addition, the ABPS model avoids the use of the re-INVITE messages, since the introduced extension to the RTP protocol includes in the SRTP datagram itself the information necessary to continue the communications after the sender of the datagram has changed its IP address. Hence, if a network failure occurs, the ABPS scheme allows to automatically exploit the other networks (see Figure 12). In particular, it includes the Authentication Tag (a digital signature based on the one-session cryptographic key) that, together with the UDP port on which the SRTP message has been received, identifies univocally the RTF flow between MN and CN, to which the datagram belongs, and avoids counterfeiting. This approach allows the proxy to identify the sender, regardless of its IP address, i.e. regardless of the NIC and the network the sender used to transmit the datagram. Thus, after the sender has been identified, the APBS proxy server detects its IP address reading the 32-bit "source IP address" header field of the IP datagram that transports the received SRTP datagram (see the example of the SRTP message labeled 47a in the figure 11). The same occurs for the sender UDP port number. In this way, the RTP datagram transports both the data and a minimal set of signaling information that allows the MN to switch the datagram through all its available NICs. Currently, we are studying the effectiveness of an extension that aims to including anyone of the complete set of IP addresses of the MN's NICs in the ABPS-RTP datagram.

```
46    RTP MESSAGE

IP  - Src 127.0.0.1       Dst 127.0.0.1       Len 110
UDP - Src 30010           Dst 57202           Len 90
RTP - RTP Version: RFC 1889 Version (2)
      RTP padding: False
      RTP Extension: False
      Contributing source identifiers count: 0
      Marker: False
      Payload type: speex (103)
      Sequence number: 9810
      Extended sequence number: 75346
      Timestamp: 6720
      Synchronization Source identifier: 0x0535d765 (87414629)
      RTP Payload: 70 bytes
         369d1b9a20080168e8e8e8e8e8e8e884
         00b4747474747474742005a3a3a3a3a
         3a3a3a21002d1d1d1d1d1d1d1d1b3b60
         abababab0abababab0abababab
         0abababab7
```

```
47s   SRTP MESSAGE

IP  - Src 192.168.1.100   Dst 130.136.2.25    Len 114
UDP - Src 57200           Dst 59204           Len 94
RTP - RTP Version: RFC 1889 Version (2)
      RTP padding: False
      RTP Extension: False
      Contributing source identifiers count: 0
      Marker: False
      Payload type: speex (103)
      Sequence number: 9810
      Extended sequence number: 75346
      Timestamp: 6720
      Synchronization Source identifier: 0x39303434 (959460404)
      Encripted RTP Payload: 70 bytes
         14be3df5e2aab212a83d6264e2764007
         1e44103c6fa0f222f1c1a6db196dbf6f
         d5dcd5a94f656f0974da2c3ad7b6b29e
         7c7153d3ba9da1c9206b075c58dc2505
         fa8f29f9ace9
      Authentication Tag: 4 bytes
         dd8ae2d2
```

Figure 11: Comparison between the RTP message 46 (left) and the corresponding SRTP message 47a (right). In evidence, the Authentication Tag and the Address of the Sender.

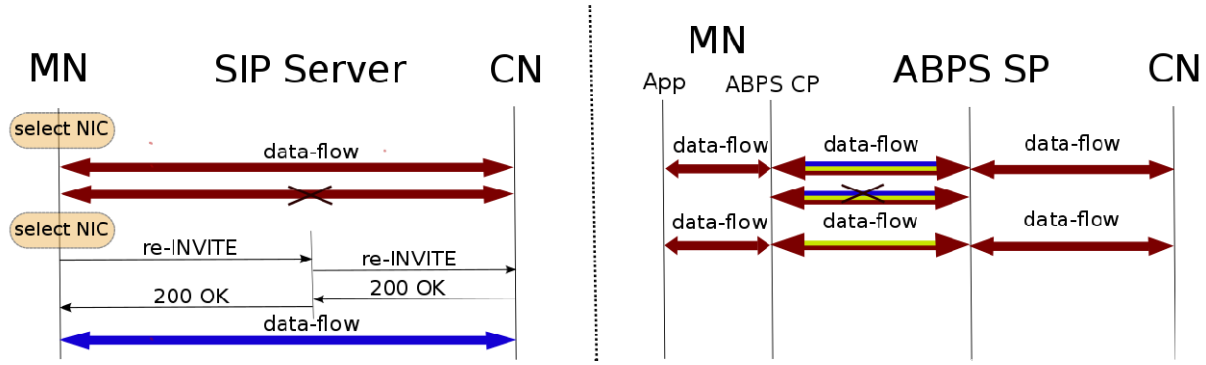

Figure 12: The Communication phase. While in a classic SIP-baed approach, upon an hand-off it is necessary to reconfigure the communication, our approach is able to automatically employ the other available networks

VI. Implementation Notes

The proposed architecture has been developed on a Linux platform. Both the proxies have been implemented by extending the open-source *siproxd* SIP/RTP proxy server version 0.7.1 and using the ZRTP/SRTP libraries version 0.80 of the Zphone Project.

As to the proxy server, a software component, termed "Policies module" has been added. As the name suggests, this module is basically in charge of determining how to transmit data the MN. In particular, during the communication, it determines which is the best MN's network interface to use, i.e. the the proxy client's IP address. To do this, it simply

uses the last received SIP and RTP signed messages to detect the IP address of the proxy client and delivers its audio datagrams to it.

At the MN, instead, a very complex architecture provides the proxy client with several services, in particular: i) dynamic network configuration for all the NICs at the datalink and network OSI layer; ii) packet forwarding support, and more specifically the implementation of techniques for the NIC selection to transmit each given packet; iii) packet loss detection and retransmission mechanisms, required to allow the ABPS Policies module to implement its own QoS policies; iv) simultaneous multiple forwarding of DNS requests through all the available NICs to avoid delay when network reconfiguration happens.

The structure of the ABPS architecture deployed at the MN is composed of 4 components as depicted in figure 13 and described in the following subsections, in particular monitor, transmission error detector, proxy client and proxy DNS.

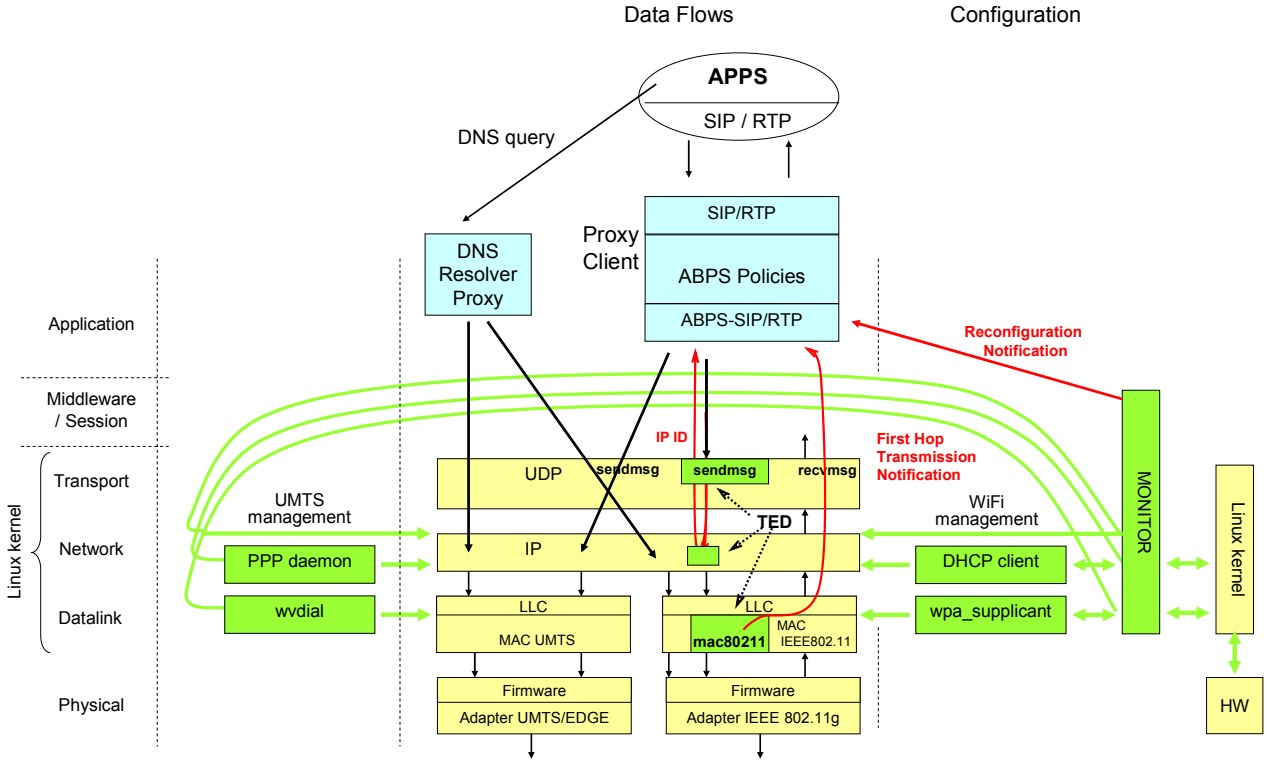

Figure 13: The structure of the ABPS architecture deployed at the MN.

1) The **Monitor** is responsible for monitoring and configuring the wireless interfaces, the routing rules necessary for packet forwarding, and reporting which interfaces are actually active. Once a wireless network interface has been either fully configured or disabled as a consequence of a communication error, a notification to proxy client is sent ("Reconfiguration Notification" message in Figure 13). The Monitor is composed of separate applications, and relies on some features and APIs of the GNU Linux operating system, i.e. those for a proper configuration of the datalink and network layers of the MN. It exploits the Linux Wireless Extensions API for the traffic analysis and the datagram-oriented Netlink socket to manage the dynamic routing tables [4]. Detection, configuration and disconnections of Wi-fi network interface cards have been implemented by reusing (and modifying) the open-source wpa_supplicant application. Instead, configuration of UMTS NICs has been implemented by modifying the wvdial application and customizing the PPP daemon. The Linux kernel can be configured to manage multiple routing tables at the network layer; more specifically, it is possible to perform dynamic routing of IP packets based on both the packet destination address and its source address. For both WiFi and UMTS NICs the monitor configures suitable routing rules using the commands "ip route" and "ip rule". These rules and routing tables impose that the IP datagrams having the same IP source address as a given NIC be routed through it, regardless of their IP destination addresses. As this NIC is configured, it can be used by the session layer. In this way, the IP datagrams of the UDP socket bound to a given NIC by the proxy client are routed and transmitted through that NIC.

2) The **Transmission Error Detector** (TED) operates across the MAC, network and transport layers of the Linux kernel [9]. Its principal responsibility is to monitor each single UDP datagram sent over an active WiFi link in order to detect whether that datagram has been successfully received by the access point or discarded by the MAC layer. TED notifies the proxy client component (see below) of UDP datagram rejections ("First-hop Transmission Notification" in Figure 13). To do so, the TED enables and extends the error message queue of the UDP socket the proxy client uses to transmit its SIP and RTP messages. This error message queue is traditionally used to deliver to the application layer the ICMP messages received from the network. For instance, a router can detect that the destination end system of an IP

datagram is unreachable and notify the sender end system of this event via an ICMP message of type 0 (network unreachable error) or 1 (host unreachable error). Specifically, an application using an UDP socket can properly configure that socket (setting up the IP_RECVERR option through the system call *setsockopt*): if the datagram sent via the socket causes a communication error, the ICMP message, generated for that error and received by the sender end system, is queued in the socket buffer. Thus, the application can receive the error notification by simply reading the incoming message queue for the socket (this is technically carried out with the primitive *recvmsg,* using, as fourth parameter, the flag MSG_ERRQUEUE). In our implementation this error notification mechanism has been extended as follows.

Firstly, we extended the *sendmsg* system call, responsible to transmit a UDP datagram through a UDP socket; the *sendmsg* system call delivers a UDP datagram to a local system buffer of that UDP socket; the routing module decides the NIC to be used for the transmission, and afterwards the *sendmsg* function returns the control to the caller. In addition, our *sendmsg* extension returns into an integer value parameter the value of the ***id*** field of the IP header that precedes the UDP datagram. The *id* is a (temporary) unique IP datagram identifier. The proxy client component uses the *sendmsg* function and maintains the unique *id* of each UDP datagram sent to the destination end system, waiting for a notification of successful or failed transmission from the MAC layer, part of the TED component. This notification contains the *id* of the UDP datagram.

Secondly, we introduced a novel error notification message. Upon UDP datagram transmission, that datagram is encapsulated inside an IP datagram and a datalink frame, and then delivered to the firmware of the wireless NIC. The firmware carries out asynchronously the transmission to the AP and eventually returns the frame transmission outcome to the MAC layer. The TED component receives that outcome from the firmware: if the frame contains a UDP datagram, TED extracts the outcome, finds the socket that has been used to send the datagram, and informs the sending socket by delivering a particular message in the socket message queue. The proxy client has enabled previously the use of that error queue by configuring the socket by invoking the *setsockopt* system call with the IP_RECVERR flag. In our implementation, we have extended the use of the error queue by introducing a new type of message (i.e., the IP_NOTIFY message). This message contains a) the *id* of the IP datagram, b) the outcome of the IP datagram fragment transmission, c) the length of the fragment, d) the more fragment field and, e) the offset field of the IP datagram fragment.

The proxy client uses the extended notification system, enables the use of the socket message queue, sends UDP datagram using the *sendmsg* system call, collects the id field of the IP datagram that encapsulates the UDP datagram and waits for IP_NOTIFY messages. These messages are read using the *recvmsg* system call with the MSG_ERRQUEUE flag in order to understand whether a given UDP datagram has been sent or discarded, and to decide whether that datagram is to be retransmitted.

3) At the session layer the **proxy client** implements the relaying of SIP and RTP/RTCP messages and the necessary digital signatures, as described in the previous Sections IV and V. The proxy client includes a simple ABPS Policies module that provides VoIP applications with the necessary interactivity and low packet losses by means of notifications received from different components, the early packet loss detection TED and the manager of the NICs, the Monitor.

To this end, the proxy client uses a UDP socket for each active wireless network interface of the mobile host (each UDP socket is connected to a wireless interface using the *bind* primitive). The proxy client receives three types of notifications (the non dashed lines in Figure 13): a first type of notification comes from the Monitor component, which informs the proxy client that either a new network interface is available or a given active interface has been disabled. For each new available interface, the proxy client generates a UDP socket and binds that socket with the new interface. In contrast, for each interface that has been disabled, the proxy client closes the corresponding UDP socket and considers as lost the UDP datagrams that have been sent through that socket and for which no "First-hop Transmission Notification" from TED has been received. The second type of notification comes from the ICMP protocol which may notify when the proxy server (or its destination port) is unreachable (through that network interface). The third type of notification is generated by TED to inform the proxy client that a UDP datagram has been either successfully received by the WiFi access point or permanently discarded. If no TED notification arrives at the proxy client before 30 msec after a UDP datagram has been sent, the proxy client assumes that datagram has been lost. Based on these three types of notifications, the proxy client decides if a lost UDP datagram has either to be retransmitted using another wireless network interface among those available on the mobile host, or permanently discarded. In addition, based on both the monitoring notifications of the sent UDP datagrams and the ICMP notifications, the proxy client selects the wireless network interface to be used for sending successive UDP datagrams.

4) The last component is a DNS proxy that receives DNS requests from the local applications and does multiple requests to the DNS servers in parallel through all the available NICs. The first response received from a DNS server is transmitted back to the local application that made the DNS request. The aim of this component is to avoid a DNS request receives no timely response due to a fault of the only NIC through which the request has been sent. The DNS proxy has been implemented extending an open source DNS proxy, the *dnsmasq* application.

VII. Performance Evaluation

In this work we are interested in demonstrating that our ABPS architecture, deployed in a urban context and exploited by a mobile node equipped with both a WiFi and a UMTS wireless network interface, is able to provide the

user on the MN with a seamless VoIP communication. In particular we aim to demonstrate that our architecture reduces the interval time in which the VoIP communications are not available due to both packet losses and unavailability of wireless coverage. To this end, we assume a scenario in which a MN leaves the coverage area of the current selected WiFi access point and, after a given time interval in which that MN exploits the UMTS network, it enters the coverage area of another WiFi access point. In this scenario, we expect that our ABPS architecture avoids additional delays occurring in other SIP based solutions when the MN changes its IP configuration, e.g. due to a link-layer handoff, or when, for some reason, RTP messages must be transmitted through a different NIC. In fact, when a link-layer handoff occurs, or when RTP messages must be transmitted through a different NIC, the classic SIP-based approaches pause RTP transmission while the two end-nodes exchange a re-INVITE SIP message to setup the new communication flows. In the followings, we describe the experimental evaluation we have carried out in the scenario mentioned above in order to show the ability of the ABPS system to effectively react to the changes in the availability of the communication resources.

Real tests were performed within (and in the proximity of) a building where a WiFi network was available; the wireless access point of that network allowed to route the traffic to the Internet via an Italian ADSL Internet Service Provider (ISP). Concurrently, a UMTS cellular network was available in the same area. The proxy server were placed at the Department of Computer Science of the University of Bologna. The CN was placed in a different city, namely Cesena, 80 km from Bologna. The average round trip time times from the mobile client to the CN were 110 and 259 milliseconds when using the WiFi or UMTS network interface cards, respectively. It is important to point out that the average values were measured along the path where the tests were made. The signal strength related to each network varied along this path. Needless to say, the performances of the WiFi network was seriously affected by the distance between the mobile terminal and the access point (and the presence of obstacles between them). In fact, when the signal strength was low, we noticed several packets' retransmissions at the physical level, hence augmenting the final time to transmit each packet at higher levels.

During such tests, a laptop running the VoIP application and the ABPS proxy client covered a route of about 80 meters. During its path, the different network access points available to the user were accessed and used. In particular, the mobile user started its path from a point where both WiFi and UMTS networks were available. In this situation, the configuration of the MN was as described in the previous figure 4. Then, the MN left the building hence causing a disconnection with the WiFi network, while maintaining active its UMTS connection. Finally, the MN re-entered in the WiFi covered area. The motivation behind these tests was to assess: i) if ABPS is able to detect which networks are available during the path; ii) if ABPS is able to automatically decide which network should be exploited when both UMTS and WiFi networks are available (and in case, exploit both networks); iii) the ability of ABPS to switch from a network to another when some disconnection/reconnection is experienced; iv) measure the network performances obtained when multiple networks are utilized to transmit game events. RTP packets transport 30 msec of audio in a payload of 70 bytes.

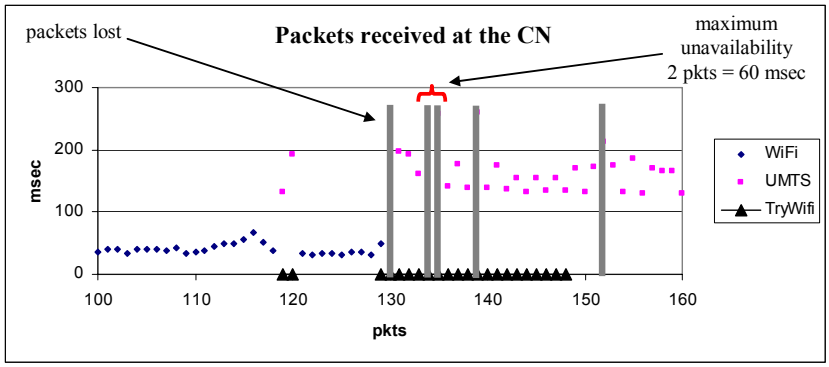

Figure 14: Transition from WiFi to UMTS – Datagram received at the CN side.

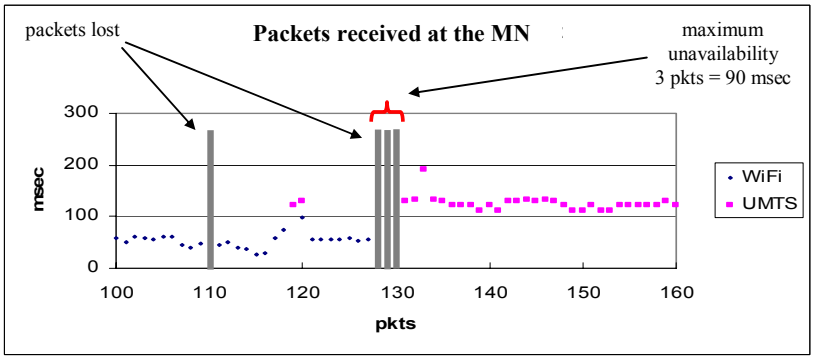

Figure 15: Transition from WiFi to UMTS – Datagram received at the MN side.

Figures 14 and 15 show the delivery times experienced to transmit RTP packets from the client to the CN and vice versa, respectively, in the interval time in which the MN leaves the coverage of the WiFi access point. They refer to a single experiment. We repeated such experiment several times (10 tests), obtaining similar results. Each chart reports, for each message, its delivery time and which network has been exploited. In both Figures 14 and 15, points marked as TryWifi (black points lying on the abscissa line) refer to those packets that have been firstly sent through the WiFi network, when the connection with the WiFi access point is being lost, but the client is still not aware that the WiFi signal strength is deteriorating. In a classic scenario, where no proxies are available (and UDP is exploited at the transport layer), such packets would have been lost. Conversely, using ABPS, the proxy client detects the problem and retransmits the packets through the UMTS network; thus, packets can be received eventually at the CN. This is a demonstration that ABPS prevents communication holes due to the time needed to detect and recover network connections, typical of a "single connection" based approach, e.g. when a single network is exploited, or when an ABC mechanism is employed (which usually performs an automatic network reconfiguration). Finally, the grey vertical bars indicate the packets that have not been received at the destination. The largest interval of consecutive lost packets represents the interval in which the VoIP service results not available to the users.

As expected, we may appreciate different behaviours, depending on the available networks. In fact, when both WiFi and UMTS are available, the communication primarily flows through the WiFi, which provides better communication performance. Conversely, when the WiFi is no longer available the communication flows through the UMTS network. This approach allows to dynamically switch from a preferred network to another without causing interruptions in the communication. A consideration worth of mention is that, even if UDP has been exploited as the transport communication protocol, a negligible amount of RTP messages has been lost, lower than 1%. Neither gaps in the events' transmission have been experienced, due to network (re)configurations.

Such low percentage of lost messages is due to the use of our cross-layer architecture. It is crucial to observe that in our experiment the interval in which the VoIP service results not available for the users is very short; in fact, only two consecutive packets were lost in the direction from MN to CN, for an unavailability of 60 msec, and only three consecutive packets (90 msec) from the CN to the MN.

As to end-to-end latencies experienced using ABPS, these obviously depend on the specific network infrastructure being used to carry voice data between the hosts involved in the communication. If we set the VoIP dependent threshold for network latencies equal to 150 msec, by looking at Figures 14 and 15 we observe that experienced latencies are generally lower than that threshold, but, 35% of the voice messages have higher latencies when UMTS is used. However, worse performances would have been obtained by resorting to a single network. Indeed, using WiFi the communication would have not been possible in the portion of the path without WiFi signal coverage. In contrast, higher average network latencies than those we measured would have been observed when resorting to UMTS.

VIII. Conclusions

In this paper, we have presented a distributed architecture for the provision of seamless and responsive mobile multimedia services. Based on our ABPS operating mode, all the MN's NICs can be simultaneously exploited, in the sense that each datagram is transmitted through the most suitable NIC in use at the time. The architecture is based on the use of two main distributed entities, i.e. a proxy client which is installed on the MN and a fixed proxy server that communicates with the client. The provided experimental results demonstrated the effectiveness of our solution.

Our approach optimized the interaction/communication between a MN and its CN. Scalability of the service can be ensured by adopting a dynamic allocation scheme which starts the execution of a proxy server each time a MN needs it. To this extent, cloud computing architectures might be proficiently employed.

References

[1] Boing wireless, http://www.boingo.com/, 2008.

[2] FONERA, http://www.fon.com/, 2008.

[3] IEEE, "Media Independent Handover", IEEE Draft Standard 802.21, 2008.

[4] Linux wireless extensions api. http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html. 2008.

[5] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[6] Bellavista, P.; Corradi, A.; Foschini, L.; "IMS-Compliant management of vertical handoffs for mobile multimedia session continuity", *Communications Magazine, IEEE*, vol.48, no.4, pp.114-121, April 2010.

[7] G. Camarillo, M.A. Garcia-Martin, M.A. Garcia-Martin, The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds, 2nd Ed. Wiley 2006.

[8] L. Chen, "Recommendation for Key Derivation Using Pseudorandom Functions", NIST Special Publication 800-108, Nov. 2004, available at http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf

[9] V. Ghini, G. Lodi, F. Panzieri, "Always Best Packet Switching: the mobile VoIP case study", Journal of Communications, Academy Publishers Ed., accepted for publication, May, 2009, also available at http://www.cs.unibo.it/~ghini/accepted/Ghini_JCM17.pdf

[10] V. Ghini, S. Ferretti, F. Panzieri, "Mobile Games Through the Nets: a Cross-Layer Architecture for Seamless Playing", in Proceedings of the International Workshop on DIstributed SImulation & Online gaming (DISIO 2010) - ICST Conference on Simulation Tools and Techniques (SIMUTools 2010), Torremolinos (Spain), ICST, March 2010.

[11] S. Gundavalli et al., “Proxy Mobile IPv6”, IETF Internet Draft, draft-ietf-netlmm-proxymip6-01.txt, June 2007.
[12] E. Gustafsson and A. Jonsson, “Always Best Connected", IEEE Comm. Mag., vol. 10, no. 1, Feb. 2003, pp. 49–55.
[13] S. Ferretti, V. Ghini, F. Panzieri, E. Turrini, “Seamless Support of Multimedia Distributed Applications Through a Cloud”, in Proceedings of the 3rd International Conference on Cloud Computing (IEEE Cloud 2010), Miami (USA), IEEE, July 2010.
[14] D. Johnson, C. Perkins, J. Arkko, “Mobility support in IPv6”, RFC 3775, June 2004.
[15] Kalmanek, C.; Murray, J.; Rice, C.; Gessel, B.; Kabre, R.; Moskal, A.; "A network-based architecture for seamless mobility services," Communications Magazine, IEEE , vol.44, no.6, pp.103-109, June 2006.
[16] S. Kashihara, K. Tsukamoto, “Service-oriented mobility management architecture for seamless handover in ubiquitous networks”, IEEE Wireless Communications, pp. 28-34, Apr. 2007.
[17] R. Koodli, “Fast Handover for Mobile IPv6”, IETF RFC 4068, July 2005.
[18] E. Kooler et al., “Datagram Congestion Control Protocol (DCCP)”, IETF RFC 4340, March 2006.
[19] K. Kong et al, “Mobility management for All-IP mobile networks: Mobile IPv6 vs. Proxy Mobile IPv6”, *IEEE Wireless Communications*, April 2008.
[20] T.M. Lim, Chai Kiat Yeo, Francis Bu Sung Lee, Quang Vinh Le, "TMSP: Terminal Mobility Support Protocol," IEEE Transactions on Mobile Computing, vol. 8, no. 6, pp. 849-863, June 2009.
[21] R. Moskowitz, P. Nikander, “Host Identity Protocol (HIP) Architecture,” IETF RFC 4423, May 2006.
[22] M. Riegel, M. Tuexen, “Mobile SCTP,” IETF Internet draft, draft-riegel-tuexen-mobile-sctp-07.txt, Oct. 2006.
[23] J. Rosemberg et al., “SIP: session initiation protocol”, IETF RFC 3261, June 2002.
[24] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, RFC 3489, March 2003.
[25] J. Rosenberg, R. Mahy, C. Huitema, “Traversal Using Relay NAT (TURN)”, Internet-Draft, draft-rosenberg-midcom-turn-08, September 2005.
[26] H. Soliman et al., “Hierachical Mobile IPv6 Mobility Management (HMIPv6)”, IETF RFC 4140, Aug. 2005.
[27] S. Salsano et al., “SIP-based mobility management in next generation networks”, IEEE Wireless Communications, pp. 92-99, April 2008.
[28] H. Schulzrinne, E. Wedlund. 2000. Application-layer mobility using SIP. SIGMOBILE Mob. Comput. Commun. Rev. 4, 3 (July 2000), 47-57.
[29] F. Teraoka, “LIN6: A Solution to Multihoming and Mobility in IPv6”, IETF Internet Draft, draft-teraoka-multi6-lin6-00.txt, 2006.
[30] U. Toseef et al., “Realization of multiple access interface management and flow mobility in IPv6”, ACM Mobilware, Innsbruck (Aut), Feb. 08.
[31] Udugama, A.; Kuladinithi, K.; Gorg, C.; Pittmann, F.; Tionardi, L.; "NetCAPE: Enabling Seamless IMS Service Delivery across Heterogeneous Mobile Networks," Communications Magazine, IEEE , vol.45, no.7, pp.84-91, July 2007.
[32] P. Zimmermann, A. Johnston, J. Callas, "ZRTP: Media Path Key Agreement for Secure RTP", Internet-Draft, draft-zimmermann-avt-zrtp-15, March 2009.

Appendix A. Messages of the Client Registration Phase.

In this Appendix Section, we list the messages exchanged by the entities involved in the client registration phase, based on the APBS operational mode. Figures 16 and 17 report these messages. The numbers placed at the left top margin of each message refer to the numbering of the message exchange described in Figure 7, Section V.2. The source (src) and destination (dst) are specified in the fields of the messages.

5

```
IP  -   Src 192.168.1.100      Dst 130.136.1.110      Len 55
UDP -   Src 56185              Dst 53                 Len 35
DNS -   DNS (query)
        Flags: 0x0100 (Standard query)
        Queries
                ekiga.net: type A, class IN
```

7

```
IP  -   Src 127.0.0.1          Dst 127.0.0.1          Len 485
UDP-    Src 5060               Dst 5070               Len 465
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        Route: <sip:127.0.0.1:5070;lr>\r\n
        Max-Forwards: 70\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        User-Agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Contact: <sip:vic00@127.0.0.1:5060>\r\n
        Expires: 100\r\n
        Content-Length:  0\r\n
        \r\n
```

11

```
IP  -   Src 130.136.2.25       Dst 86.64.162.35       Len 637
UDP -   Src 59000              Dst 5060               Len 617
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 130.136.2.25:59000;rport;
                branch=z9hG4bKe057af0ec22995f436e1c15196bf5719\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
                branch=z9hG4bKc5b2506810b25d2bcae49b3fff76a5f1\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        Contact: <sip:vic00@130.136.2.25:59000>\r\n
        Max-forwards: 68\r\n
        User-agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Expires: 100\r\n
        Content-Length: 0\r\n
        \r\n
```

13r

```
IP  -   Src 130.136.2.26       Dst 192.168.1.100      Len 727
UDP -   Src 59000              Dst 6060               Len 707
SIP -   SIP/2.0 401 Unauthorized\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
                branch=z9hG4bKc5b2506810b25d2bcae49b3fff76a5f1\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        Record-Route: <sip:abps@130.136.2.25:59000;lr>\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>;
                tag=c64e1f832a41ec1c1f4e5673ac5b80f6.a3d0\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        WWW-Authenticate: Digest realm="ekiga.net",
            nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b"\r\n
        Server: Kamailio (1.5.3-notls (i386/linux))\r\n
        Seqnum: 1\r\n
        Fingerprint: ec609d366d43342ddf42083b3eeb3402339a99e3\r\n
        Content-Length: 0\r\n
        \r\n
```

15

```
IP  -   Src 127.0.0.1          Dst 127.0.0.1          Len 668
UDP -   Src 5060               Dst 5070               Len 648
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj6535dde2-817c-4fcc-897d-74bf598106e9\r\n
        Route: <sip:127.0.0.1:5070;lr>\r\n
        Max-Forwards: 70\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52061 REGISTER\r\n
        User-Agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Contact: <sip:vic00@127.0.0.1:5060>\r\n
        Expires: 100\r\n
        Authorization: Digest username="vic00",
                realm="ekiga.net",
                nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b",
                uri="sip:ekiga.net",
                response="2130120d56b1eebcf68572e8473bcce8"\r\n
        Content-Length:  0\r\n
        \r\n
```

6

```
IP  -   Src 130.136.1.110      Dst 192.168.1.100      Len 142
UDP -   Src 53 Dst 56185       Len 122
DSN -   DNS (response)
        Flags: 0x8180 (Standard query response, No error)
        Queries
                ekiga.net: type A, class IN
        Answers
                ekiga.net: type A, class IN, addr 86.64.162.35
        Authoritative nameservers
                ekiga.net: type NS, class IN, ns dns.ovh.net
                ekiga.net: type NS, class IN, ns ns.ovh.net
        Additional records
                ns.ovh.net: type A, class IN, addr 213.251.128.136
                dns.ovh.net: type A, class IN, addr 213.186.33.102
```

8s

```
IP  -   Src 192.168.1.100      Dst 130.136.2.25       Len 647
UDP -   Src 6060               Dst 59000              Len 627
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;
                branch=z9hG4bKc5b2506810b25d2bcae49b3fff76a5f1;rport\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        Contact: <sip:vic00@10.254.254.254:5070>\r\n
        Max-forwards: 69\r\n
        User-agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Expires: 100\r\n
        Proxyclientid: proxyClientBeta1\r\n
        Seqnum: 1\r\n
        Fingerprint: 09379e93894ec6bed8907cdccfabf3ae98c98aaa\r\n
        Content-Length: 0\r\n
        \r\n
```

12

```
IP  -   Src 86.64.162.35       Dst 130.136.2.25       Len 709
UDP     Src 5060               Dst 59000              Len 689
SIP -   SIP/2.0 401 Unauthorized\r\n
        Via: SIP/2.0/UDP 130.136.2.25:59000;rport=59000;
                branch=z9hG4bKe057af0ec22995f436e1c15196bf5719\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
                branch=z9hG4bKc5b2506810b25d2bcae49b3fff76a5f1\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>;
                tag=c64e1f832a41ec1c1f4e5673ac5b80f6.a3d0\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        WWW-Authenticate: Digest realm="ekiga.net",
           nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b"\r\n
        Server: Kamailio (1.5.3-notls (i386/linux))\r\n
        Content-Length: 0\r\n
        \r\n
```

14

```
IP  -   Src 127.0.0.1          Dst 127.0.0.1          Len 617
UDP -   Src 5070               Dst 5060               Len 597
SIP -   SIP/2.0 401 Unauthorized\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        Record-Route: <sip:abps@130.136.2.25:59000;lr>\r\n
        Record-Route: <sip:siproxd@127.0.0.1:5070;lr>\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>;
                tag=c64e1f832a41ec1c1f4e5673ac5b80f6.a3d0\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        WWW-Authenticate: Digest realm="ekiga.net",
           nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b"\r\n
        Server: Kamailio (1.5.3-notls (i386/linux))\r\n
        Content-Length: 0\r\n
        \r\n
```

Figure 16: Messages of the Client Registration Phase - Part 1.

16s

```
IP  -   Src 192.168.1.100     Dst 130.136.2.25       Len 830
UDP -   Src 6060              Dst 59000              Len 810
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
               branch=z9hG4bK75ee3a3cf77ce0144e0abf8acfd58df9\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
               branch=z9hG4bKPj6535dde2-817c-4fcc-897d-74bf598106e9\r\n
        From: <sip:vic00@ekiga.net>;
               tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52061 REGISTER\r\n
        Contact: <sip:vic00@10.254.254.254:5070>\r\n
        Authorization: Digest username="vic00",
               realm="ekiga.net",
              nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b",
               uri="sip:ekiga.net",
               response="2130120d56b1eebcf68572e8473bcce8"\r\n
        Max-forwards: 69\r\n
        User-agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Expires: 100\r\n
        Proxyclientid: proxyClientBeta1\r\n
        Seqnum: 2\r\n
        Fingerprint: a0563466910a05ab7c3e4399c6095e7a7c6ec34f\r\n
        Content-Length: 0\r\n
        \r\n
```

17

```
IP  -   Src 130.136.2.25      Dst 86.64.162.35       Len 820
UDP -   Src 59000             Dst 6060               Len 800
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 130.136.2.25:59000;rport;
               branch=z9hG4bK56095cc8eedc47f59b0b3ad672921cb7\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
               branch=z9hG4bK75ee3a3cf77ce0144e0abf8acfd58df9\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
               branch=z9hG4bKPj6535dde2-817c-4fcc-897d-74bf598106e9\r\n
        From: <sip:vic00@ekiga.net>;
               tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52061 REGISTER\r\n
        Contact: <sip:vic00@130.136.2.25:59000>\r\n
        Authorization: Digest username="vic00",
               realm="ekiga.net",
               nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b",
               uri="sip:ekiga.net",
               response="2130120d56b1eebcf68572e8473bcce8"\r\n
        Max-forwards: 68\r\n
        User-agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Expires: 100\r\n
        Content-Length: 0\r\n
        \r\n
```

19r

```
IP  -   Src 130.136.2.25      Dst 192.168.1.100      Len 667
UDP -   Src 59000             Dst 6060               Len 647
SIP -   SIP/2.0 200 OK\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
               branch=z9hG4bK75ee3a3cf77ce0144e0abf8acfd58df9\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
               branch=z9hG4bKPj6535dde2-817c-4fcc-897d-74bf598106e9\r\n
        Record-Route: <sip:abps@130.136.2.25:59000;lr>\r\n
        From: <sip:vic00@ekiga.net>;
               tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>;
               tag=c64e1f832a41ec1c1f4e5673ac5b80f6.005c\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52061 REGISTER\r\n
        Contact: <sip:vic00@130.136.2.26:6060>;expires=600\r\n
        Server: Kamailio (1.5.3-notls (i386/linux))\r\n
        Seqnum: 2\r\n
        Fingerprint: 188e7911c03fe73600a7283a2ae46ac3fb9f3967\r\n
        Content-Length: 0\r\n
        \r\n
```

20

```
IP  -   Src 127.0.0.1         Dst 127.0.0.1          Len 557
UDP -   Src 5070              Dst 5060               Len 537
SIP -   SIP/2.0 200 OK\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
               branch=z9hG4bKPj6535dde2-817c-4fcc-897d-74bf598106e9\r\n
        Record-Route: <sip:abps@130.136.2.25:59000;lr>\r\n
        Record-Route: <sip:siproxd@127.0.0.1:5070;lr>\r\n
        From: <sip:vic00@ekiga.net>;
               tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>;
               tag=c64e1f832a41ec1c1f4e5673ac5b80f6.005c\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52061 REGISTER\r\n
        Contact: <sip:vic00@130.136.2.26:6060>;expires=600\r\n
        Server: Kamailio (1.5.3-notls (i386/linux))\r\n
        Content-Length: 0\r\n
        \r\n
```

Figure 17: Messages of the Client Registration Phase - Part 2.

Appendix B. Messages of the INVITE Phase.

In this Appendix Section, we list the messages exchanged by the entities involved during the INVITE phase, based on the APBS operational mode (see Figures 18 to 21). The numbers placed at the left top margin of each message refer to the numbering of the message exchange described in Figure 9, Section V.3.

5

```
IP  -   Src 192.168.1.100      Dst 130.136.1.110      Len 55
UDP -   Src 56185              Dst 53                 Len 35
DNS -   DNS (query)
        Flags: 0x0100 (Standard query)
        Queries
                ekiga.net: type A, class IN
```

6

```
IP  -   Src 130.136.1.110      Dst 192.168.1.100      Len 142
UDP -   Src 53 Dst 56185       Len 122
DSN -   DNS (response)
        Flags: 0x8180 (Standard query response, No error)
        Queries
                ekiga.net: type A, class IN
        Answers
                ekiga.net: type A, class IN, addr 86.64.162.35
        Authoritative nameservers
                ekiga.net: type NS, class IN, ns dns.ovh.net
                ekiga.net: type NS, class IN, ns ns.ovh.net
        Additional records
                ns.ovh.net: type A, class IN, addr 213.251.128.136
                dns.ovh.net: type A, class IN, addr 213.186.33.102
```

7

```
IP  -   Src 127.0.0.1          Dst 127.0.0.1          Len 485
UDP-    Src 5060               Dst 5070               Len 465
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        Route: <sip:127.0.0.1:5070;lr>\r\n
        Max-Forwards: 70\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        User-Agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Contact: <sip:vic00@127.0.0.1:5060>\r\n
        Expires: 100\r\n
        Content-Length:  0\r\n
        \r\n
```

8s

```
IP  -   Src 192.168.1.100      Dst 130.136.2.25       Len 647
UDP -   Src 6060               Dst 59000              Len 627
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;
                branch=z9hG4bKc5b2506810b25d2bcae49b3fff76a5f1;rport\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        Contact: <sip:vic00@10.254.254.254:5070>\r\n
        Max-forwards: 69\r\n
        User-agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Expires: 100\r\n
        Proxyclientid: proxyClientBeta1\r\n
        Seqnum: 1\r\n
        Fingerprint: 09379e93894ec6bed8907cdccfabf3ae98c98aaa\r\n
        Content-Length: 0\r\n
        \r\n
```

11

```
IP  -   Src 130.136.2.25       Dst 86.64.162.35       Len 637
UDP -   Src 59000              Dst 5060               Len 617
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 130.136.2.25:59000;rport;
                branch=z9hG4bKe057af0ec22995f436e1c15196bf5719\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
                branch=z9hG4bKc5b2506810b25d2bcae49b3fff76a5f1\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        Contact: <sip:vic00@130.136.2.25:59000>\r\n
        Max-forwards: 68\r\n
        User-agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Expires: 100\r\n
        Content-Length: 0\r\n
        \r\n
```

12

```
IP  -   Src 86.64.162.35       Dst 130.136.2.25       Len 709
UDP     Src 5060               Dst 59000              Len 689
SIP -   SIP/2.0 401 Unauthorized\r\n
        Via: SIP/2.0/UDP 130.136.2.25:59000;rport=59000;
                branch=z9hG4bKe057af0ec22995f436e1c15196bf5719\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
                branch=z9hG4bKc5b2506810b25d2bcae49b3fff76a5f1\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>;
                tag=c64e1f832a41ec1c1f4e5673ac5b80f6.a3d0\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        WWW-Authenticate: Digest realm="ekiga.net",
           nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b"\r\n
        Server: Kamailio (1.5.3-notls (i386/linux))\r\n
        Content-Length: 0\r\n
        \r\n
```

13r

```
IP  -   Src 130.136.2.26       Dst 192.168.1.100      Len 727
UDP -   Src 59000              Dst 6060               Len 707
SIP -   SIP/2.0 401 Unauthorized\r\n
        Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
                branch=z9hG4bKc5b2506810b25d2bcae49b3fff76a5f1\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        Record-Route: <sip:abps@130.136.2.25:59000;lr>\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>;
                tag=c64e1f832a41ec1c1f4e5673ac5b80f6.a3d0\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        WWW-Authenticate: Digest realm="ekiga.net",
            nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b"\r\n
        Server: Kamailio (1.5.3-notls (i386/linux))\r\n
        Seqnum: 1\r\n
        Fingerprint: ec609d366d43342ddf42083b3eeb3402339a99e3\r\n
        Content-Length: 0\r\n
        \r\n
```

14

```
IP  -   Src 127.0.0.1          Dst 127.0.0.1          Len 617
UDP -   Src 5070               Dst 5060               Len 597
SIP -   SIP/2.0 401 Unauthorized\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj3aa20304-2347-4173-8bce-48cd596c7d9e\r\n
        Record-Route: <sip:abps@130.136.2.25:59000;lr>\r\n
        Record-Route: <sip:siproxd@127.0.0.1:5070;lr>\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>;
                tag=c64e1f832a41ec1c1f4e5673ac5b80f6.a3d0\r\n
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52060 REGISTER\r\n
        WWW-Authenticate: Digest realm="ekiga.net",
           nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b"\r\n
        Server: Kamailio (1.5.3-notls (i386/linux))\r\n
        Content-Length: 0\r\n
        \r\n
```

15

```
IP  -   Src 127.0.0.1          Dst 127.0.0.1          Len 668
UDP -   Src 5060               Dst 5070               Len 648
SIP -   REGISTER sip:ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
                branch=z9hG4bKPj6535dde2-817c-4fcc-897d-74bf598106e9\r\n
        Route: <sip:127.0.0.1:5070;lr>\r\n
        Max-Forwards: 70\r\n
        From: <sip:vic00@ekiga.net>;
                tag=7de363d5-50ab-41d3-8bc7-dd4a6929460a\r\n
        To: <sip:vic00@ekiga.net>
        Call-ID: deb6a364-d79e-4c47-add5-f6117ba8b8c6\r\n
        CSeq: 52061 REGISTER\r\n
        User-Agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Contact: <sip:vic00@127.0.0.1:5060>\r\n
        Expires: 100\r\n
        Authorization: Digest username="vic00",
                realm="ekiga.net",
                nonce="4d3af7630001546fea8c3d913132883d30644d3f59fd850b",
                uri="sip:ekiga.net",
                response="2130120d56b1eebcf68572e8473bcce8"\r\n
        Content-Length:  0\r\n
        \r\n
```

```
21

IP  -   Src 127.0.0.1         Dst 127.0.0.1          Len 1130
UDP -   Src 5060              Dst 5070               Len 1110
SIP -   INVITE sip:fab11@ekiga.net SIP/2.0\r\n
        Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
               branch=z9hG4bKPj016c96b3-23c4-4578-b4bc-a2b8c8bb84d8\r\n
        Max-Forwards: 70\r\n
        From: sip:vic00@ekiga.net;
               tag=8366118d-fecc-42d7-a87d-14c5c1b1f8fb\r\n
        To: sip:fab11@ekiga.net\r\n
        Contact: <sip:vic00@127.0.0.1:5060>\r\n
        Call-ID: 67001726-8b2b-4896-89a6-a20900f612b1\r\n
        CSeq: 21774 INVITE\r\n
        Route: <sip:127.0.0.1:5070;lr>\r\n
        Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE,
               SUBSCRIBE, NOTIFY, REFER, MESSAGE, OPTIONS\r\n
        Supported: replaces, 100rel, timer, norefersub\r\n
        Session-Expires: 1800\r\n
        Min-SE: 90\r\n
        User-Agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
        Content-Type: application/sdp\r\n
        Content-Length:   452\r\n
        \r\n
SDP -   v=0\r\n
        o=- 3504698843 3504698843 IN IP4 127.0.0.1\r\n
        s=pjmedia\r\n
        c=IN IP4 127.0.0.1\r\n
        t=0 0\r\n
        a=X-nat:0\r\n
        m=audio 30010 RTP/AVP 103 102 104 113 3 0 8 9 101\r\n
        a=rtcp:30011 IN IP4 127.0.0.1\r\n
        a=rtpmap:103 speex/16000\r\n
        a=rtpmap:102 speex/8000\r\n
        a=rtpmap:104 speex/32000\r\n
        a=rtpmap:113 iLBC/8000\r\n
        a=fmtp:113 mode=30\r\n
        a=rtpmap:3 GSM/8000\r\n
        a=rtpmap:0 PCMU/8000\r\n
        a=rtpmap:8 PCMA/8000\r\n
        a=rtpmap:9 G722/8000\r\n
        a=sendrecv\r\n
        a=rtpmap:101 telephone-event/8000\r\n
        a=fmtp:101 0-15\r\n
        \r\n
```

```
22s

IP  - Src 192.168.1.100    Dst 130.136.2.25     Len 1493
UDP - Src 6060             Dst 59000            Len 1473
SIP - INVITE sip:fab11@ekiga.net SIP/2.0\r\n
      Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
           branch=z9hG4bK666e10a0479a64e9f3f9bfde0aea52dc\r\n
      Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
           branch=z9hG4bKPj016c96b3-23c4-4578-b4bc-a2b8c8bb84d8\r\n
      From: <sip:vic00@ekiga.net>;
           tag=8366118d-fecc-42d7-a87d-14c5c1b1f8fb\r\n
      To: <sip:fab11@ekiga.net>\r\n
      Call-ID: 67001726-8b2b-4896-89a6-a20900f612b1\r\n
      CSeq: 21774 INVITE\r\n
      Contact: <sip:vic00@10.254.254.254:5070>\r\n
      Content-Type: application/sdp\r\n
      Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, SUBSCRIBE,
           NOTIFY, REFER, MESSAGE, OPTIONS\r\n
      Max-forwards: 69\r\n
      Supported: replaces, 100rel, timer, norefersub\r\n
      Session-expires: 1800\r\n
      Min-se: 90\r\n
      User-agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
      Proxyclientid: proxyClientBeta1\r\n
      Seqnum: 3\r\n
      Fingerprint: bb0a776090d2bfccd47a76b430196b428a2976c1\r\n
      Content-Length:   545\r\n
      \r\n
SDP - v=0\r\n
      o=- 3504698843 3504698843 IN IP4 10.254.254.254\r\n
      s=pjmedia\r\n
      c=IN IP4 10.254.254.254\r\n
      t=0 0\r\n
      a=X-nat:0\r\n
      m=audio 57200 RTP/AVP 103 102 104 113 3 0 8 9 101\r\n
      a=rtcp:57201 IN IP4 10.254.254.254\r\n
      a=rtpmap:103 speex/16000\r\n
      a=rtpmap:102 speex/8000\r\n
      a=rtpmap:104 speex/32000\r\n
      a=rtpmap:113 iLBC/8000\r\n
      a=fmtp:113 mode=30\r\n
      a=rtpmap:3 GSM/8000\r\n
      a=rtpmap:0 PCMU/8000\r\n
      a=rtpmap:8 PCMA/8000\r\n
      a=rtpmap:9 G722/8000\r\n
      a=sendrecv\r\n
      a=rtpmap:101 telephone-event/8000\r\n
      a=fmtp:101 0-15\r\n
      a=zrtp-hash:1.00     b63baab73823f664ca4546af9b3006ed
                           ddad2512074fbfd0ae338ed9dba5e039\r\n
      \r\n
```

Figure 18: Messages of the Invite Phase - Part 1/4.

23

```
IP  - Src 130.136.2.25    Dst 86.64.162.35     Len 1396
UDP - Src 59000           Dst 5060             Len 1376
SIP - INVITE sip:fab11@ekiga.net SIP/2.0\r\n
      Via: SIP/2.0/UDP 130.136.2.25:59000;rport;
           branch=z9hG4bK95d154770f70299280df35171aa6fce2\r\n
      Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
           branch=z9hG4bK666e10a0479a64e9f3f9bfde0aea52dc\r\n
      Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
           branch=z9hG4bKPj016c96b3-23c4-4578-b4bc-a2b8c8bb84d8\r\n
      From: <sip:vic00@ekiga.net>;
           tag=8366118d-fecc-42d7-a87d-14c5c1b1f8fb\r\n
      To: <sip:fab11@ekiga.net>\r\n
      Call-ID: 67001726-8b2b-4896-89a6-a20900f612b1\r\n
      CSeq: 21774 INVITE\r\n
      Contact: <sip:vic00@130.136.2.25:59000>\r\n
      Content-Type: application/sdp\r\n
      Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, SUBSCRIBE,
           NOTIFY, REFER, MESSAGE, OPTIONS\r\n
      Max-forwards: 68\r\n
      Supported: replaces, 100rel, timer, norefersub\r\n
      Session-expires: 1800\r\n
      Min-se: 90\r\n
      User-agent: PJSUA v1.4.5/i686-pc-linux-gnu\r\n
      Content-Length:   458\r\n
      \r\n
SDP - v=0\r\n
      o=- 3504698843 3504698843 IN IP4 130.136.2.25\r\n
      s=pjmedia\r\n
      c=IN IP4 130.136.2.25\r\n
      t=0 0\r\n
      a=X-nat:0\r\n
      m=audio 59202 RTP/AVP 103 102 104 113 3 0 8 9 101\r\n
      a=rtcp:59203 IN IP4 130.136.2.25\r\n
      a=rtpmap:103 speex/16000\r\n
      a=rtpmap:102 speex/8000\r\n
      a=rtpmap:104 speex/32000\r\n
      a=rtpmap:113 iLBC/8000\r\n
      a=fmtp:113 mode=30\r\n
      a=rtpmap:3 GSM/8000\r\n
      a=rtpmap:0 PCMU/8000\r\n
      a=rtpmap:8 PCMA/8000\r\n
      a=rtpmap:9 G722/8000\r\n
      a=sendrecv\r\n
      a=rtpmap:101 telephone-event/8000\r\n
      a=fmtp:101 0-15\r\n
      \r\n
```

30r

```
IP  - Src 130.136.2.25    Dst 192.168.1.100    Len 1315
UDP - Src 59000           Dst 6060             Len 1295
SIP - SIP/2.0 200 OK\r\n
      Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
           branch=z9hG4bK666e10a0479a64e9f3f9bfde0aea52dc\r\n
      Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
           branch=z9hG4bKPj016c96b3-23c4-4578-b4bc-a2b8c8bb84d8\r\n
      Record-Route: <sip:86.64.162.35;lr;did=c56.fecc9f23>\r\n
      Record-Route: <sip:abps@130.136.2.25:59000;lr>\r\n
      From: <sip:vic00@ekiga.net>;
           tag=8366118d-fecc-42d7-a87d-14c5c1b1f8fb\r\n
      To: <sip:fab11@ekiga.net>;
           tag=ZHeGb52GouIMlRMsVYOhLQ4IKPUEJyUL\r\n
      Call-ID: 67001726-8b2b-4896-89a6-a20900f612b1\r\n
      CSeq: 21774 INVITE\r\n
      Contact: <sip:fab11@137.204.72.5:55060>\r\n
      Content-Type: application/sdp\r\n
      Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, SUBSCRIBE,
           NOTIFY, REFER, MESSAGE, OPTIONS\r\n
      Supported: replaces, 100rel, timer, norefersub\r\n
      Session-expires: 1800;refresher=uac\r\n
      Seqnum: 4\r\n
      Fingerprint: f0243ddcf3750731c4a65b6bde6e349e1ba312c1\r\n
      Content-Length:   342\r\n
      \r\n
SDP - v=0\r\n
      o=- 3504698843 3504698844 IN IP4 130.136.2.25\r\n
      s=pjmedia\r\n
      c=IN IP4 130.136.2.25\r\n
      t=0 0\r\n
      a=X-nat:0\r\n
      m=audio 59204 RTP/AVP 103 101\r\n
      a=rtcp:59205 IN IP4 137.204.72.5\r\n
      a=rtpmap:103 speex/16000\r\n
      a=sendrecv\r\n
      a=rtpmap:101 telephone-event/8000\r\n
      a=fmtp:101 0-15\r\n
      a=zrtp-hash:1.00
c4db831d0b4b5232afd99d940ca6c5d8675a6b09daa5ba0464a04b7753dcfe4a\r\n
      \r\n
```

32r

```
IP  - Src 130.136.2.25    Dst 192.168.1.100    Len 56
UDP - Src 59204           Dst 57200            Len 36
ZRTP- RTP Version: 0
      RTP padding: False
      RTP Extension: True
      Sequence: 14940
      Magic Cookie: ZRTP
      Source Identifier: 0x66303837
      Message
           Signature: 0x505a
           Length: 3
           Type: HelloACK
      Checksum: 0x415a3475 [correct]
```

26s

```
IP  - Src 130.136.2.25    Dst 130.136.2.26     Len 581
UDP - Src 59000           Dst 6060             Len 561
SIP - SIP/2.0 100 Giving a try\r\n
      Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
           branch=z9hG4bK666e10a0479a64e9f3f9bfde0aea52dc\r\n
      Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
           branch=z9hG4bKPj016c96b3-23c4-4578-b4bc-a2b8c8bb84d8\r\n
      Record-Route: <sip:abps@130.136.2.25:59000;lr>\r\n
      From: <sip:vic00@ekiga.net>;
           tag=8366118d-fecc-42d7-a87d-14c5c1b1f8fb\r\n
      To: <sip:fab11@ekiga.net>\r\n
      Call-ID: 67001726-8b2b-4896-89a6-a20900f612b1\r\n
      CSeq: 21774 INVITE\r\n
      Server: Kamailio (1.5.3-notls (i386/linux))\r\n
      Seqnum: 3\r\n
      Fingerprint: b130f6a0ad524c8f44b260885b2c0ab2a94b4e3a\r\n
      Content-Length: 0\r\n
      \r\n
```

29

```
IP  - Src 86.64.162.35    Dst 130.136.2.25     Len 1133
UDP - Src 5060            Dst 59000            Len 1113
SIP - SIP/2.0 200 OK\r\n
      Via: SIP/2.0/UDP 130.136.2.25:59000;
           rport=59000;received=130.136.2.25;
           branch=z9hG4bK95d154770f70299280df35171aa6fce2\r\n
      Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
           branch=z9hG4bK666e10a0479a64e9f3f9bfde0aea52dc\r\n
      Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
           branch=z9hG4bKPj016c96b3-23c4-4578-b4bc-a2b8c8bb84d8\r\n
      Record-Route: <sip:86.64.162.35;lr;did=c56.fecc9f23>\r\n
      Call-ID: 67001726-8b2b-4896-89a6-a20900f612b1\r\n
      From: <sip:vic00@ekiga.net>;
           tag=8366118d-fecc-42d7-a87d-14c5c1b1f8fb\r\n
      To: <sip:fab11@ekiga.net>;
           tag=ZHeGb52GouIMlRMsVYOhLQ4IKPUEJyUL\r\n
      CSeq: 21774 INVITE\r\n
      Contact: <sip:fab11@137.204.72.5:55060>\r\n
      Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE, SUBSCRIBE,
           NOTIFY, REFER, MESSAGE, OPTIONS\r\n
      Supported: replaces, 100rel, timer, norefersub\r\n
      Session-Expires: 1800;refresher=uac\r\n
      Content-Type: application/sdp\r\n
      Content-Length:   259\r\n
      \r\n
SDP - v=0\r\n
      o=- 3504698843 3504698844 IN IP4 137.204.72.5\r\n
      s=pjmedia\r\n
      c=IN IP4 137.204.72.5\r\n
      t=0 0\r\n
      a=X-nat:0\r\n
      m=audio 55062 RTP/AVP 103 101\r\n
      a=rtcp:55063 IN IP4 137.204.72.5\r\n
      a=rtpmap:103 speex/16000\r\n
      a=sendrecv\r\n
      a=rtpmap:101 telephone-event/8000\r\n
      a=fmtp:101 0-15\r\n
      \r\n
```

31s

```
IP  - Src 192.168.1.100   Dst 130.136.2.25     Len 168
UDP - Src 57200  Dst 59204 Len 148
ZRTP- RTP Version: 0
      RTP padding: False
      RTP Extension: True
      Sequence: 21712
      Magic Cookie: ZRTP
      Source Identifier: 0x39303434
      Message
         Signature: 0x505a
         Length: 31
         Type: Hello
         Data
            ZRTP protocol version: 1.10
            Client Identifier: zrtp_proxyClient
            Hash Image:
DE04448BFB82BE6A42539B7F0D3983EE15438AB2EE247F269CE4C8869ADE3D32
            ZID: 70726F7879436C69656E7442
            MiTM: False
            Passive: False
            Hash type count = 1
                Hash[0]: SHA-256 Hash
            Cipher type count = 2
                Cipher[0]: AES-CM with 256 bit keys
                Cipher[1]: AES-CM with 128 bit keys
            Auth tag count = 1
                Auth tag[0]: HMAC-SHA1 32 bit authentication tag
            Key agreement type count = 3
                Key agreement[0]: DH mode with p=3072 bit prime
                Key agreement[1]: DH mode with p=2048 bit prime
                Key agreement[2]: Multistream mode
            SAS type count = 2
                SAS type[0]: Short authentication string using base 256
                SAS type[1]: Short authentication string using base 32
            HMAC: 8AAD4A20292059B5
      Checksum: 0x482f54a4 [correct]
```

Figure 19: Messages of the Invite Phase - Part 2/4.

35s

```
IP  - Src 192.168.1.100    Dst 130.136.2.25     Len 160
UDP - Src 57200  Dst 59204 Len 140
ZRTP- RTP Version: 0
      RTP padding: False
      RTP Extension: True
      Sequence: 21713
      Magic Cookie: ZRTP
      Source Identifier: 0x39303434
      Message
         Signature: 0x505a
         Length: 29
         Type: Commit
         Data
            Hash Image:
E7E48B3F2D8801564A1507186DB70BD1C68A240CE9C98B4EB8236E89D175FC9E
            ZID: 70726F7879436C69656E7442
            Hash: SHA-256 Hash
            Cipher: AES-CM with 256 bit keys
            Auth tag: HMAC-SHA1 32 bit authentication tag
            Key agreement: DH mode with p=3072 bit prime
            SAS type: Short authentication string using base 256
            hvi:
F0E90BD35F33FBCC2D63867633 8A2EC570374B3A4934F29E7F2A5223AE9969B7
            HMAC: F635B5096EED82F0
      Checksum: 0xb09422c7 [correct]
```

38s

```
IP  - Src 192.168.1.100  Dst 130.136.2.25    Len 677
UDP - Src 6060           Dst 59000           Len 657
SIP - ACK sip:fab11@137.204.72.5:55060 SIP/2.0\r\n
      Via: SIP/2.0/UDP 10.254.254.254:5070;rport;
           branch=z9hG4bKfe8c1b0d35c3c7f8d3133f36f2d8c943\r\n
      Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
           branch=z9hG4bKPj11fcc609-b659-47f7-98bf-980c85842706\r\n
      Route: <sip:abps@130.136.2.25:59000;lr>\r\n
      Route: <sip:86.64.162.35;lr;did=c56.fecc9f23>\r\n
      From: <sip:vic00@ekiga.net>;
           tag=8366118d-fecc-42d7-a87d-14c5c1b1f8fb\r\n
      To: <sip:fab11@ekiga.net>;
           tag=ZHeGb52GouIMlRMsVYOhLQ4IKPUEJyUL\r\n
      Call-ID: 67001726-8b2b-4896-89a6-a20900f612b1\r\n
      CSeq: 21774 ACK\r\n
      Max-forwards: 69\r\n
      Proxyclientid: proxyClientBeta1\r\n
      Seqnum: 4\r\n
      Fingerprint: b82548869c35a81f1aedefebcda081e09c676ecb\r\n
      Content-Length: 0\r\n
      \r\n
```

42s

```
IP  - Src 192.168.1.100  Dst 130.136.2.25    Len 512
UDP - Src 57200          Dst 59204           Len 492
ZRTP- RTP Version: 0
      RTP padding: False
      RTP Extension: True
      Sequence: 21714
      Magic Cookie: ZRTP
      Source Identifier: 0x39303434
      Message
         Signature: 0x505a
         Length: 117
         Type: DHPart2
         Data
            Hash Image:
75980096C0899999E7CFB1990DFA85A93EE529B29045A3A51B026D7D1D5E5126
            rs1ID: 494BBDCACA775570
            rs2ID: F344CF3849670CA8
            auxs: 16E3120FAE1A5BD3
            pbxs: F316354D6C041CB9
            pvi Data:   384 bytes ....
            HMAC: C782114FAB35BF97
      Checksum: 0x6af599ae [correct]
```

44s

```
IP  - Src 192.168.1.100  Dst 130.136.2.25    Len 120
UDP - Src 57200          Dst 59204           Len 100
ZRTP- RTP Version: 0
      RTP padding: False
      RTP Extension: True
      Sequence: 21716
      Magic Cookie: ZRTP
      Source Identifier: 0x39303434
      Message
         Signature: 0x505a
         Length: 19
         Type: Confirm2
         Data
            HMAC: DA636405A033B365
            CFB: D90637AEB95F0EBE0E117F3406695108
            Encrypted Data: 40 bytes ...
      Checksum: 0x39f16c3a [correct]
```

36

```
IP  - Src 127.0.0.1      Dst 127.0.0.1      Len 1116
UDP - Src 5070           Dst 5060           Len 1096
SIP - SIP/2.0 200 OK\r\n
      Via: SIP/2.0/UDP 127.0.0.1:5060;rport;
           branch=z9hG4bKPj016c96b3-23c4-4578-b4bc-a2b8c8bb84d8\r\n
      Record-Route: <sip:86.64.162.35;lr;did=c56.fecc9f23>\r\n
      Record-Route: <sip:abps@130.136.2.25:59000;lr>\r\n
      Record-Route: <sip:siproxd@127.0.0.1:5070;lr>\r\n
      From: <sip:vic00@ekiga.net>;
           tag=8366118d-fecc-42d7-a87d-14c5c1b1f8fb\r\n
      To: <sip:fab11@ekiga.net>;
           tag=ZHeGb52GouIMlRMsVYOhLQ4IKPUEJyUL\r\n
      Call-ID: 67001726-8b2b-4896-89a6-a20900f612b1\r\n
      CSeq: 21774 INVITE\r\n
      Contact: <sip:fab11@137.204.72.5:55060>\r\n
      Content-Type: application/sdp\r\n
      Allow: PRACK, INVITE, ACK, BYE, CANCEL, UPDATE,
           SUBSCRIBE, NOTIFY, REFER, MESSAGE, OPTIONS\r\n
      Supported: replaces, 100rel, timer, norefersub\r\n
      Session-expires: 1800;refresher=uac\r\n
      Content-Length:   253\r\n
      \r\n
SDP - v=0\r\n
      o=- 3504698843 3504698844 IN IP4 127.0.0.1\r\n
      s=pjmedia\r\n
      c=IN IP4 127.0.0.1\r\n
      t=0 0\r\n
      a=X-nat:0\r\n
      m=audio 57202 RTP/AVP 103 101\r\n
      a=rtcp:57203 IN IP4 137.204.72.5\r\n
      a=rtpmap:103 speex/16000\r\n
      a=sendrecv\r\n
      a=rtpmap:101 telephone-event/8000\r\n
      a=fmtp:101 0-15\r\n
      \r\n
```

41r

```
IP  - Src 130.136.2.25    Dst 192.168.1.100    Len 512
UDP - Src 59204           Dst 57200            Len 492
ZRTP- RTP Version: 0
      RTP padding: False
      RTP Extension: True
      Sequence: 14942
      Magic Cookie: ZRTP
      Source Identifier: 0x66303837
      Message
         Signature: 0x505a
         Length: 117
         Type: DHPart1
         Data
            Hash Image:
58B801AD13657A2F580D073FF50D315CEEFA9CBE91C892205AC38F98D9421952
            rs1ID: AC801BE75381B2AA
            rs2ID: 946193F24B096F1E
            auxs: 0028D013A143B1F7
            pbxs: 6FD6624DAE66BBE2
            pvr Data:   384 bytes ....
            HMAC: 854FF719AC5B8124
      Checksum: 0xa61542b0 [correct]
```

43r

```
IP  - Src 130.136.2.25    Dst 192.168.1.100    Len 120
UDP - Src 59204           Dst 57200            Len 100
ZRTP- RTP Version: 0
      RTP padding: False
      RTP Extension: True
      Sequence: 14943
      Magic Cookie: ZRTP
      Source Identifier: 0x66303837
      Message
         Signature: 0x505a
         Length: 19
         Type: Confirm1
         Data
            HMAC: 1CDEB1F113A47AE6
            CFB: EC6DB708DAD93409A0095B4A7EC00FAE
            Encrypted Data: 40 bytes ...
      Checksum: 0xa8a5f108 [correct]
```

45r

```
IP  - Src 130.136.2.25    Dst 192.168.1.100    Len 56
UDP - Src 59204           Dst 57200            Len 36
ZRTP- RTP Version: 0
      RTP padding: False
      RTP Extension: True
      Sequence: 14945
      Magic Cookie: ZRTP
      Source Identifier: 0x66303837
      Message
         Signature: 0x505a
         Length: 3
         Type: Conf2ACK
      Checksum: 0xfc6f3c2b [correct]
```

Figure 20: Messages of the Invite Phase - Part 3/4.

46

```
IP  - Src 127.0.0.1       Dst 127.0.0.1       Len 110
UDP - Src 30010           Dst 57202           Len 90
RTP - RTP Version: RFC 1889 Version (2)
      RTP padding: False
      RTP Extension: False
      Contributing source identifiers count: 0
      Marker: False
      Payload type: speex (103)
      Sequence number: 9810
      Extended sequence number: 75346
      Timestamp: 6720
      Synchronization Source identifier: 0x0535d765 (87414629)
      RTP Payload: 70 bytes
         369d1b9a20080168e8e8e8e8e8e8e884
         00b4747474747474442005a3a3a3a3a
         3a3a3a21002d1d1d1d1d1d1d1b3b60
         abababab0abababab0abababab
         0abababab7
```

47s

```
IP  - Src 192.168.1.100   Dst 130.136.2.25     Len 114
UDP - Src 57200           Dst 59204           Len 94
RTP - RTP Version: RFC 1889 Version (2)
      RTP padding: False
      RTP Extension: False
      Contributing source identifiers count: 0
      Marker: False
      Payload type: speex (103)
      Sequence number: 9810
      Extended sequence number: 75346
      Timestamp: 6720
      Synchronization Source identifier: 0x39303434 (959460404)
      Encripted RTP Payload: 70 bytes
         14be3df5e2aab212a83d6264e2764007
         1e44103c6fa0f222f1c1a6db196dbf6f
         d5dcd5a94f656f0974da2c3ad7b6b29e
         7c7153d3ba9da1c9206b075c58dc2505
         fa8f29f9ace9
      Authentication Tag: 4 bytes
         dd8ae2d2
```

48

```
IP  - Src 130.136.2.25    Dst 137.204.72.5    Len 110
UDP - Src 59202           Dst 55062           Len 90
RTP - RTP Version: RFC 1889 Version (2)
      RTP padding: False
      RTP Extension: False
      Contributing source identifiers count: 0
      Marker: False
      Payload type: speex (103)
      Sequence number: 9810
      Extended sequence number: 75346
      Timestamp: 6720
      Synchronization Source identifier: 0x39303434 (959460404)
      RTP Payload: 70 bytes
         369d1b9a20080168e8e8e8e8e8e8e884
         00b4747474747474442005a3a3a3a3a
         3a3a3a21002d1d1d1d1d1d1d1b3b60
         abababab0abababab0abababab
         0abababab7
```

Figure 21: Messages of the Invite Phase - Part 4/4.