Hindawi Publishing Corporation
Advances in Multimedia
Volume 2011, Article ID 372591, 21 pages
doi:10.1155/2011/372591

Research Article

A Survey of Open Source Products for Building a SIP Communication Platform

Pavel Segec and Tatiana Kovacikova

Department of InfoCom Networks, University of Zilina, Univerzitna 8215/1, 010 26 Zilina, Slovakia

Correspondence should be addressed to Tatiana Kovacikova, tatiana.kovacikova@fri.uniza.sk

Received 29 July 2011; Revised 31 October 2011; Accepted 15 November 2011

Academic Editor: T. Turletti

The Session Initiation Protocol (SIP) is a multimedia signalling protocol that has evolved into a widely adopted communication standard. The integration of SIP into existing IP networks has fostered IP networks becoming a convergence platform for both real-time and non-real-time multimedia communications. This converged platform integrates data, voice, video, presence, messaging, and conference services into a single network that offers new communication experiences for users. The open source community has contributed to SIP adoption through the development of open source software for both SIP clients and servers. In this paper, we provide a survey on open SIP systems that can be built using publically available software. We identify SIP features for service development and programming, services and applications of a SIP-converged platform, and the most important technologies supporting SIP functionalities. We propose an advanced converged IP communication platform that uses SIP for service delivery. The platform supports audio and video calls, along with media services such as audio conferences, voicemail, presence, and instant messaging. Using SIP Application Programming Interfaces (APIs), the platform allows the deployment of advanced integrated services. The platform is implemented with open source software. Architecture components run on standardized hardware with no need for special purpose investments.

1. Introduction

The success of the IP network model with its many attractive services has started a process of network convergence based on IP technology. Thanks to this convergence, the patterns of communication among people have been changing. New IP networking technologies and their evolution have resulted in a communication platform which is able to provide many services and applications. Multimedia technologies have become standardized, and many advanced communication platforms are available, especially for Voice over IP (VoIP). Setting up your own multimedia platform with audio and video telephony, audio conferencing, presence, and messaging has never been easier. In addition, these new multimedia technologies offer new opportunities for creation of innovative and attractive services. Individual services may be integrated with running real-time and non-real-time services, thus providing a converged unified communication environment with an arbitrary combination of services, applications, devices, and networks.

The Session Initiation Protocol (SIP) [1] provides an open multimedia signalling protocol. It is considered as a key protocol for network convergence. SIP facilitates building converged IP communication networks that integrate and cooperate with existing telecommunication networks and equipments. SIP technology is still evolving, but the core specifications have been fully standardized, adopted, and widely implemented. There are many commercial products and communication platforms on the market that are SIP based, including those from Cisco, Microsoft, Avaya, and Radvision. SIP also brings the internet approach to multimedia real-time communication; thus, SIP communication may be seen as just another IP service, running on standard hardware and standard operating systems and middleware. Thanks to this approach, there are many high-performance and high-quality software products available provided under some form of open source or free software licenses. These products together with the appropriate software packages can be used to create a rich multimedia converged network.

In this paper, we identify key protocols, technologies, and services and we propose an advanced converged IP communication platform using SIP for service delivery. This platform is based on SIP and open supplementary communication protocols implemented as open source software. The platform integrates individual open source packages to support audio and video calls along with media services such as audio conference, voicemail, presence, and instant messaging. Using SIP Application Programming Interfaces (APIs), the platform facilitates development and deployment of advanced integrated services such as click2call and web-initiated conferences. A NAT traversal solution is also provided. All architecture components run on standardized hardware with no need for special purpose investments. The proposed platform is not a simple SIP PBX, it is rather a powerful, flexible, and easily extendable open source communication platform that is exploitable in many ways.

This paper does not provide exhaustive comparison of components' features or a complete design guide. It serves for better understanding of required technologies and features which can be incorporated to converged SIP communication platform. The platform facilitates building your own testbeds that can be used for education and research at universities and high schools.

In Section 2, SIP features for service development and programming are highlighted. Services and applications of a SIP-converged platform are introduced in Section 3. In Section 4, the most important technologies supporting SIP functionalities are identified. A proposed SIP-converged platform together with its logical components and appropriate open source products are described in Section 5. Finally, conclusions are provided in Section 6.

2. Creating SIP Services

One of major benefits of SIP is its ability to be used for creation of new communication services. There are a number of approaches for creating SIP services which may extend platform functionalities. An overview of these approaches is provided below.

2.1. SIP Baseline Protocol Mechanism. Unlike classic telephones in PSTN/ISDN, SIP endpoints are not dumb terminals. Therefore, many services which were implemented in telecommunication networks or in a PBX environment can be provided using SIP baseline functions in SIP servers and/or SIP endpoints. A range of services is not specified by the IETF, unlike the traditional telecommunication world. A few Best Current Practice RFCs exist describing the deployment of some services, such as call forwarding, call hold, Music on hold, and so forth call parking, Call pickup.

2.2. New Methods and Headers. SIP has been designed to be easily extensible; thus, new features and services can be implemented by defining new SIP methods or SIP headers. These new methods or headers *do not need* to be supported by all SIP servers. If a SIP proxy receives a request with an unknown method or header, it will proxy the request. The new methods have already been defined (INFO, PRACK, PUBLISH, SUBSCRIBY, NOTIFY, MESSAGE, REFER, UPDATE) as extensions of the baseline SIP specification. These new methods provide services such as presence, instant messaging, and interworking with telecommunication networks.

2.3. Programming SIP Services Using Dedicated Tools. SIP services can be implemented by service logic that is created and implemented in the SIP architecture. In general, any programming language can be used. The logic is used to process a specific SIP signalling message flows to react to special conditions due to special events. These events can be triggered by receiving a specific SIP message, or a SIP header value, or an argument of a specific message [2]. A couple of interfaces have been defined, including Call Processing Language—SIP CPL [3], a Common Gateway Interface—SIP CGI [4], SIP Servlets [5], and Java API for Integrated Networks—JAIN APIs [6]. Using programming APIs for the development of SIP communication services and applications is very similar to IT application development. Programming APIs are usually provided by *SIP application servers*.

3. Advanced SIP Applications and Services

This section surveys services and applications providing an important part of the SIP-converged communication platform.

3.1. SIP Instant Messaging and Presence (SIMPLE). The instant messaging (IM) service together with the presence service has become dominant together with the basic VoIP service. It has been predicated that these services will be as that of "a dial tone" of the twenty-first century [7]. Due to many proprietary IM services, there was an effort in IETF to standardize presence and IM services. Currently, two IETF working groups (WG) are specialized on IM and presence services: Extensible Messaging and Presence Protocol WG (xmpp wg) [8] and SIP for Instant Messaging and Presence Leveraging Extensions (simple wg) [9]. This has led to the definition of two protocols that can be used for IM and presence services: XMPP protocol (a successor of a very popular, but still proprietary jabber protocol) and SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions).

SIMPLE is an umbrella covering more than ten RFCs concerning presence and messaging. SIMPLE covers two services, presence and IM, which are logically coupled together. IM is defined as a fast (close to real-time) exchange of short text messages between participants. There are two IM solutions for SIP page mode and a session mode [10, 11]. Page mode IM, defined in RFC 3428 [12], uses a newly defined SIP method for text communication—MESSAGE. The MESSAGE method is delivered directly to participants, without certain SIP session (no session dialog is established), see Figure 1. The session mode assumes that the IM message exchange is a part of a session which is established by some means (such as rendezvous protocol). For session type of IM implementation, SIP may be the Message Session Relay

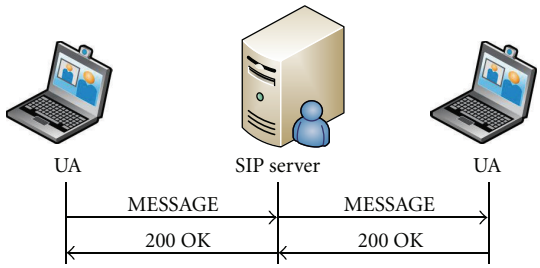

Figure 1: Page mode IM.

Protocol (MSRP) [13, 14]. MSRP is a text-based connection-oriented instant message transport protocol which transports arbitrary (binary) content on a peer-to-peer basis. MSRP is not a standalone protocol, but it uses its own messages (SEND and REPORT) for communication.

SIP presence is a service, where the availability and willingness of an user can be communicated. Initially, only two states were provided—online and offline. Nowadays, presence is a feature-rich service that provides many kinds of information and capabilities. The presence server inside the SIMPLE architecture uses an event notification framework [15]. The event notification framework extends the main SIP specification allowing a SIP user agent to request notification from remote nodes indicating that certain events have occurred. It also uses a general instant messaging and presence (IMP) framework defined in [10, 11]. The event notification extension standardizes two new SIP methods, SUBSCRIBE, NOTIFY, and a couple of new headers [15]. The event notification framework for SIP presence notification specified in RFC 3856 [16] defines the "end-to-end" model of SIP presence. The model was not very powerful; therefore, the presence framework was extended to allow actively push presence information to the network with a new SIP method PUBLISH [17]. This model of SIP presence is called the "agent-based" or centralized model. However, to provide typical IM services such as buddy lists, SIP messengers implementing SIMPLE have adopted the XML Configuration Access Protocol (XCAP) family of protocols. XCAP allows a client to read, write, and modify application configuration data stored in XML format on a server [18]; here, this information is presence information.

To implement the complete SIMPLE standards (see Figure 2) is a complex task, and there are few implementations yet. Many SIP software vendors, due to the complexity of the SIMPLE framework, are implementing the simpler and more feature rich XMPP protocol into their products.

3.2. Media Application and Services. In a SIP network, there is a need for available advanced media processing functionalities. These functionalities enable us to build interesting applications such as unified messaging and unified communication. Media functions include announcement, message recording, service interaction, media mixing, and multi party communication [19]. Media processing is usually provided by media servers. Voicemail, audio conferencing, and interactive voice response (IVR) are the most popular services using media functions. For communication with media servers, as defined in RFC 6230, SIP is used [20]. RFC 6230 describes a framework which is applicable for an architecture with distributed application logic and media processing.

Voicemail is a service which allows sending, storing, and retrieving audio messages, just like an answering machine does. Implementations may differ, but usually each SIP user is associated with a voice mailbox, so that when this user is called and does not answer (because he is busy or offline), then the caller is redirected to a voicemail system and is instructed to leave a message for the callee. When learning that he has voicemail message, for example, by SIP event package, then the callee can contact their voicemail server to retrieve the message. The voicemail service is based on a standard SIP unified resource identifier (URI) addressing scheme [21]. Voicemail system is a very common service in IP PBX solutions.

A conferencing service is another very popular service, which can be integrated into the SIP-converged platform. A conference is a service supporting multisite communications with many participants, policy-based management, management of resources, and notification. Several models of SIP conferencing have been identified [22]. These models include centralized, end-point mixing, full mesh, and multicast SIP natively supports multicast communication; hence, this is a good choice for conferencing. Unfortunately, multicasting is often infeasible due to the lack of a multicast infrastructure. Therefore, the centralized model, based upon a star topology with point-to-point signalling and central media mixing, is frequently used [23]. Additional details of SIP conferencing and high level requirements are provided in RFC 4353 and RFC 4254 [22, 24].

SIP IVR is an application based on SIP that provides automated call interactions with humans based on dual-tone multi-frequency (DTMF) tone commands. IVR provides many possibilities [25] and is often used in an enterprise environment. Recently, RFC 6231 has been published [26]. RFC 6231 defines a media control channel framework for interactive voice response (IVR). This framework also defines dialog management request elements for preparing, starting, and terminating dialog interactions as well as associated responses and notifications to satisfy IVR requirements.

4. Supplementary Network Technologies

In this section, a survey of key supplementary network technologies needed by SIP architecture is provided.

4.1. Domain Name Server (DNS). SIP is tightly coupled with the domain name server (DNS). SIP utilizes DNS in many different ways. SIP addressing is based on SIP entities being identified by SIP uniform resource identifiers (URIs) [27]. SIP URI consists of a user name (or service name) and a host name, separated by "@." There are identified three types of SIP URIs, an address of record (AOR), a fully qualified domain name (FQDN), and a globally routable user agent (UA) URI. SIP client resolves a SIP URI into an IP address, port, and transport protocol of the next hop entity

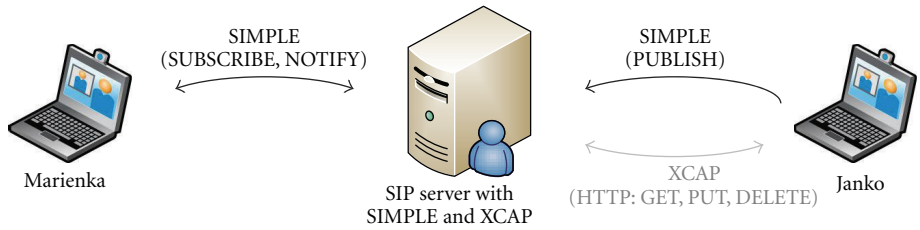

Figure 2: SIMPLE with XCAP architecture.

Selection of the transport protocol is especially interesting, because SIP can run over a variety of transport protocols, including TCP, UDP, SCTP, TLS over TCP, TLS over SCTP, SCTP over IPsec. The DNS resolving process occurs in different SIP entities (i.e., servers and endpoints) during one session. For a simplified topology and some SIP PBX implementations, a standard DNS *A record* may be sufficient. The A record provides a mapping between a name and one or more IP addresses. However, using only A records is not a recommended practice, because it does not support environments with different transport protocols, port numbers, and servers. The recommended DNS deployment for SIP infrastructures uses the DNS service record (DNS SRV) [28] or the naming authority pointer (NAPTR) DNS resource record [29]. The NAPTR RR allows clients to distinguish which protocol should be used to talk to the mapped resource. The SRV RR allows administrators to configure several servers for a single SIP domain and to find out the correct port number.

RFC 3263 [30] provides guidelines of how the SIP endpoints locate the appropriate SIP server. According to guidelines, the SIP endpoint should first use a NAPTR RR to identify which transport protocol should be used for communication with the respective SIP server. The name of a SIP server is identified by a domain name extracted from a SIP URI. The NAPTR RR allows a domain to advertise different servers for the various transport protocols. The NAPTR RR structure specifies the order and preference fields, which allow the SIP domain to define the preferred transport protocol and its backup solutions. Once a resolving process has finished, the SIP endpoint knows which transport protocols are supported and which server on which port provides the service (SRV). In the second step, the endpoint resolves the SRV records to find A/AAAA names. The structure of SRV RR allows to build a service with front end load balancing. Finally, the endpoint queries for an A or AAAA record to learn appropriate IPv4 or IPv6 addresses. Now, the endpoint has the information (see Figure 3) required to connect with the correct SIP server.

DNS may be used not only for learning Registrar/Proxy server IP address(es), but also addresses for a media server, ENUM mapping, or STUN/TURN/ICE server. Companies that are planning to implement SIP into their IP infrastructure should have their own DNS server into which they can configure the appropriate resource records. If the company wishes to interact with others, then it should have its own DNS server with registered domain(s).

4.2. NAT Traversal. Originally, main principles of SIP did not reflect security, an important issue of modern IP networks. However, with the global availability of IP networks, security has started to be more and more considered. Many security devices, either the lightweight ones such as NATs or heavy ones such as firewalls, have appeared on the market. RFC 3235 [31] has been produced to make application protocol design more NAT friendly. Unfortunately, SIP violates most of the recommendations, because being an application layer protocol, it works with L3 IP addresses inside its application messages (SIP and SDP). Therefore, implementation of SIP signalling and media capabilities into a secured network is not a straightforward task. Security devices such as NATs and firewalls in a network invoke some complications for SIP. Two problems have been identified—those related to signalling flows and those related to media flows. The problem with signalling flows has been partially solved with the use of TCP instead of UDP and with the use of "receive" parameters of the "Via" header. For NAT traversal, a number of approaches have been identified. They can be divided into two groups.

The first group relies on SIP endpoints to detect the presence of a NAT and to determine its type. These functions have to be implemented by SIP UAs (clients) and are supported by special network service servers. These servers help UAs with NAT detection task. This group of solutions is called "near-end NAT traversal" and includes solutions such as STUN, TURN, and ICE. STUN (session traversal utilities for NAT) is an IETF standardized solution [32] that allows a SIP UA to detect presence of a NAT and learn the public IP address and port assigned to this SIP UA during the process of NAT-ing. The public IP address and port are then used inside the SIP and SDP headers instead of device local private IP address. Unfortunately, STUN does not work in all situations. For example, STUN does not work over symmetric NAT, and STUN has a problem when a SIP endpoint has more than one IP address. The second solution proposed by the IETF is TURN (traversal using relay around NAT) or STUN relay [33]. TURN is a client-server protocol which allows a SIP UA to learn the public IP address (relayed transport address) of the TURN server and request it to act as a relay entity. This address is used for relaying incoming media packets (see Figure 4). The client may request the TURN server from which communication peers will be relayed and may control how the relaying is done. The client may learn TURN server's IP address through DNS SRV records or by other means. Client may utilize UDP, TCP, or TLS to connect to the TURN server. The utilization of TURN service can be protected using authentication. TURN works well with the UDP protocol; but work is still in progress concerning TCP.

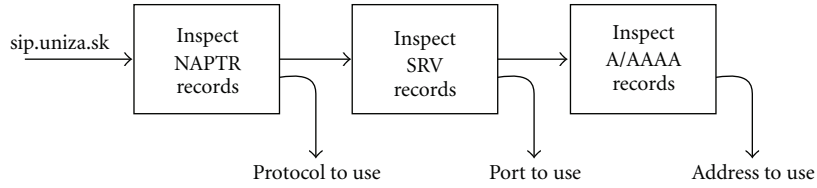

Figure 3: How SIP endpoint locates right servers.

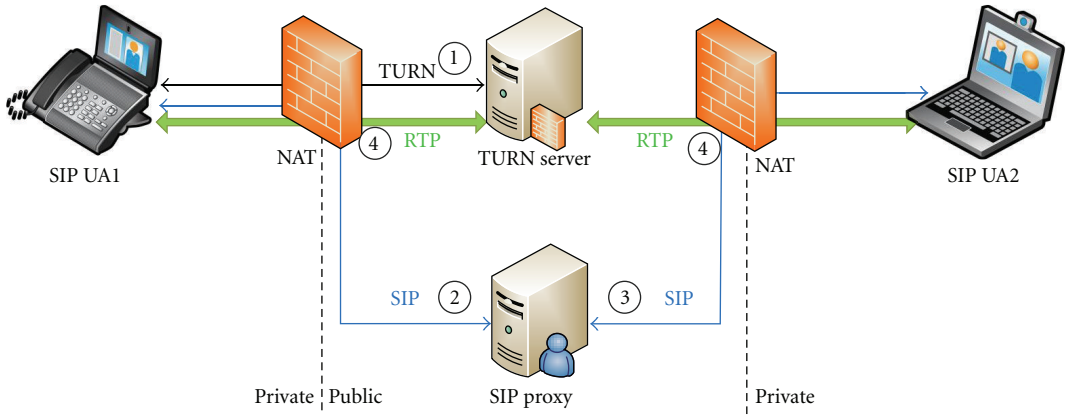

Figure 4: TURN principle.

TURN resolves the NAT traversal problem, but introduces several new "problems." A TURN server is a central point of failure; hence should be deployed reliably. Also, the TURN server must forward all media packets, both incoming and outgoing; therefore, there are some TURN system performance issues that can occur.

Therefore, the third solution, called ICE (interactive connectivity establishment), is recommended. STUN and TURN are parts of the ICE solution. ICE is specified by the IETF in RFC 5245 [34] and RFC 5768 [35]. ICE is quite simple; it allows a SIP UA to collect "candidates of communication" which are offered to the remote party, hence maximizing chance of success. ICE negotiation has several phases. First, a SIP client gathers candidates. During this phase, the SIP client finds out all addresses that can be used for communication. There are three types of candidates: a host candidate who represents clients' IP addresses, a server reflexive candidate for the address that has been resolved from STUN, and a relayed candidate for the address which has been allocated from a TURN relay by the client. The client assigns priorities for particular candidates and sends them to a remote party to start the offer-answer negotiating process. After receiving remote peer candidates, client local candidates are paired with received remote candidates. The client starts to check the connectivity for each of the candidate pairs. Finally, the session is established.

The second group of NAT traversal solutions is called "far-end NAT traversal." This group considers the SIP UA as an entity which does need to be concerned about NATs and firewalls, and it should communicate using local address as usual. This solution requires a NAT or a firewall devices to implement a SIP friendly application layer gateway (ALG), universal plug and play (UPnP), deployment of special devices at the border between the private and public network parts to act as a back to back user agent (B2BUA), or the deployment of special entities which cooperate with a SIP server in a public internet and that serve as media relay servers (RTP Proxy/Media proxy). These solutions are typically proprietary, although UPnP seems to be a promising technology widely supported by many vendors (Microsoft, consumer electronics vendors) and standardized [36]. However, UPnP has many security problems, since UPnP does not implement any security mechanisms and its deployment is considered as risky.

SIP over NAT solutions help to setup conditions required for successful outgoing and incoming signalling and media sessions establishment in situation where a SIP UA is behind a NAT. NAT devices also introduce a problem for the reverse traffic—how to reach SIP endpoints, which are behind NAT for incoming sessions. To solve this issue, RFC 5626 [37] suggests a keep-alive mechanism scheme used to keep NAT translations of communications between SIP endpoint and its serving SIP servers (registrar or proxy) opened. This extension assumes that some NAT translations, as a result of communications initiated by a SIP endpoint with a serving SIP server (e.g., those made during SIP registration) will be kept open; hence, this translation will be reused for routing incoming SIP requests through a NAT to the SIP endpoint. NAT translations are kept open by transmitting keep-alive messages, that is, client-to-server "ping" keep-alive and corresponding server-to-client "pong" messages [37]. There are proposed two keep-alive mechanisms: a CRLF keep-alive and a STUN keep-alive message exchange.

4.3. SIP Security. SIP is an application layer protocol, so it is exposed to all the IP-network security issues on one hand

and to issues directly related to SIP on the other hand. However, being an IP stack protocol, SIP may reuse the whole portfolio of security mechanisms already developed for IP networks. SIP security threats may be classified into three categories [38]. The first category is threats which inherit the classical IP-related threats and vulnerabilities such as replay attacks, denial of service [39], spoofing, sniffing, and man in the middle attacks. The second category includes some SIP-specific vulnerabilities raising from SIP's nature, including attacks such as malicious terminate and register requests. The third category includes vulnerabilities due to the complexity of SIP applications, servers, or other components, such as implementation vulnerabilities due to SQL injection or buffer overflow [38]. Protection of a SIP architecture against all of these security threats is a complex task, as it includes protection of signalling sessions and protection of media streams.

To protect signalling sessions, SIP provides several built-in security mechanisms, for example, authentication, data integrity, and confidentiality. They include a challenge-based authentication mechanism, e-mail-like secure MIME (S/MIME) mechanism, and a secured SIP schema (SIPS) with transport layer security (TLS) [1]. SIP authentication is based on authentication mechanisms of HTTP, where an originator of a request may be challenged to provide assurance of its identity. There are two versions of HTTP authentication: HTTP basic and HTTP digest authentication. HTTP basic authentication is not supported by SIP version 2 since it sends a password as a clear text. HTTP digest authentication is a challenge-response mechanism which uses cryptographic hashing (e.g., MD5) of username/password/realm with a random nonce value. HTTP digest provides some level of replay attack protection with nonce value, which is valid for some period of time only and which contains a "quality of protection (qop)" parameter. Using HTTP digest authentication is a minimal required protection mechanism but is contingent on the use of strong passwords. In a SIP-based network, the HTTP digest mechanism usually takes place between the UA and SIP server (proxy, registrar, UAS) and is used for client authentication and authorization. Using HTTP digest, the user ensures validity of its credentials. However, security of vital SIP headers (as *Contact*, *To*, *From*) is provided. To provide integrity protection and confidentiality for SIP signalling, S/MIME and TLS may be used.

The S/MIME provides a consistent way for sending and receiving secure data among users based on mechanisms such as authentication, message integrity, and data confidentiality (encryption). S/MIME uses certificates which assert that the holder is identified by an end-user address. These certificates are associated with the keys that are used to sign or encrypt bodies of SIP messages. Key exchange mechanism can be either public or private. RFC 3261 specifies the Triple Data Encryption Standard (3DES) cipher suite with SHA1 (Secure Hash Algorithm) signature algorithm as mandatory for S/MIME implementation. RFC 3853 [40] updates the guidance for S/MIME with the Advanced Encryption Standard (AES) as an encryption algorithm, RSA (Rivest, Shamir and Adleman) as a digital signature algorithm, and SHA1 as a digest algorithm [40]. Compared to 3DES, AES is faster mechanism with lower memory requirements, which makes it suitable for mobile devices, for example. AES is also required for using TLS in SIP, as it unifies the cipher suite requirements and simplifies SIP security implementation. SMIME provides end-to-end confidentiality for the SIP message body and, partially, confidentiality for the SIP headers, integrity protection for the body and identity authentication for a sender of the message [41]. Currently S/MIME is not widely deployed in SIP, mainly due to problematic certificate distribution. RFC 6072 [42] addresses this problem as it proposes a way to discover the certificates of other users and mechanisms of certification retrieval and certification management. It uses a "credential service" combined with SIP identity specification (RFC 4474) which is used to manage user's private and public certificates and which allow SIP users to store, update, and retrieve their certificates with a SIP PUBLISH message. Using credential service SIP UAs may subscribe to other SIP UA certificate that is delivered to the subscribing UA using SIP NOTIFY message. Thanks to SIP identity specification, SIP users are allowed to use certificates that are not signed by any well-known certification authority while still strongly binding the user's identity to the certificate.

Another option for secure SIP signalling communication is to use TLS (transport layer security) [43] at the transport layer. TLS has been already used in a SIP secure (SIPS) schema. TLS provides confidentiality, integrity, and replay attack protection services. The TLS protocol is composed of two layers. The upper layer consists from TLS Handshake Protocols and application data protocol entities to whom security services are offered. The TLS Handshake Protocols are the Handshake Protocol used for peer authentication and key exchange, the Alert Protocol used for error condition signaling, and the Change Cipher Protocol used for notification of new cipher and keys usage. The bottom layer, called a TLS Record Protocol, is located on top of the transport protocol (TCP, SCTP). The TLS Record Protocol takes a message from upper layer entities, then it does segmentation, and applies desired processing such as compression, MAC calculation, and encryption. Finally, it encapsulates it and sends over TLS connection to a receiving party, which applies reverse processing over a received message. Inside the SIP architecture, TLS may be used on the hop-by-hop basis (between UA and SIP servers or SIP servers itself), with the credibility of SIP entities provided by a signed certificate. The deployment usually uses one unencrypted channel on the port 5060 and the secure one on the port 5061. TLS runs over connection-oriented protocols such as TCP or SCTP as it relies on their reliable transport features. Currently, there are only few SIP clients which smoothly support TLS. TLS cannot be used with UDP because the lost or reordering of datagrams breaks TLS handshaking and TLS record layer functionalities. When UDP is preferred as a transport protocol, the datagram TLS (DTLS) is used. DTLS, standardized in RFC 4347 [44], enables communication privacy for datagram protocols. DTLS is based on the TLS protocol, and it provides equivalent security guarantees.

Finally, at the network layer, the IPSec may be used to secure SIP communication. IPSec consists from three main

components: a data plane component, policy component, and signalling component. Data plane components enable transport protection. There are two security mechanisms available, authentication header (AH) and encapsulating security payload (ESP). AH provides services such as integrity protection and data origin authentication, with optional replay protection. ESP provides the same services as AH. In addition, it provides confidentiality. The AH and ESP may use various cryptographic algorithms. A policy component defines security associations (SA) as a set of IPSec rules. SA defines sets of parameters on which IPSec neighbours have to agree. As parameters, packet matching criteria, mode of operation, type of security protocols, cryptographic algorithms, integrity algorithms, and keys have to be used with a particular packet flow exchanged among IPSec neighbours. A signalling component provides IPSec neighbours' authentication, SA parameters negotiation, and the key exchange procedures. As a signalling protocol, the Internet Key Exchange (IKE) protocol is used. IPsec may operate in two modes, a transport mode and a tunnel mode. The establishment of IPSec tunnel has two phases. During phase 1 (IKE Phase 1), the IKE protocol is used to negotiate IKE SA parameters used with its own key management exchange procedures. During phase 2 (IKE Phase 2), IPSec SA parameters themselves are negotiated. IPSec is supported by an underlying operating system, while the integration to a SIP application is not usually required. IPSec works for all, UDP-, TCP-, and SCTP based SIP signalling. Since SIP routing entities on the signalling path read, add, or change the information in SIP headers, IPSec for SIP is usually applied on the hop by hop basis. IPSec may also be used to protect the RTP media on the end-to-end basis without the active involvement of SIP UAs.

It is expected that a secured communication architecture secures not only signalling messages but also the media traffic. To protect media streams, with the Real-time transport protocol (RTP) [45] as a dominant bearer, a new RTP media profile (RTP/SAVP) has been proposed—the Secure real-time Transport Protocol (SRTP) [46]. SRTP is an efficient security protocol with low computational cost, memory and bandwidth requirements with good interoperability results. SRTP offers confidentiality, integrity, replay attack protection, and data origin authentication for RTP and RTCP traffic [46]. Confidentiality is achieved by encrypting the RTP payload. RTP defines the AES and NULL encryption methods. The NULL method is used when no encryption is desired. The AES is the default encryption method operable in two modes, segmented integer counter mode and f-8 mode. The authentication algorithm protects the integrity of an entire original RTP packet. Authentication ensures that the packet was neither modified nor inserted along the path. HMAC/SHA1 is used as the mandatory message authentication algorithm. It is recommended to use a 10-byte authentication tag for an RTP stream protection, but, to reduce the overhead of small sizes VoIP packet, a 4-byte authentication tag may be used. If used, authentication operation is performed after encryption to protect the entire RTP packet. SRTP uses two types of keys: session keys and master keys. All session keys used for authentication and encryption can be derived from a single master key. SRTP standard does not define how the master key is exchanged. It is responsible of a key exchange protocol. Currently, several key exchange protocols are available, for example, the SDP Security Descriptions (SDES) protocol [47], Multimedia Internet KEYing (MIKEY) [48], Datagram TLS (DTLS) [49], and ZRTP protocol [50]. SRTP extends the original RTP header with two fields, an optional master key identifier (MKI) and a recommended authentication tag. SRTP MKI identifies which master key was used to derive the current session keys used for encryption and/or message authentication. The authentication tag is used to carry the message authentication data.

The DNS service plays an important role in SIP networks. In general, a DNS service has heavy negative effects on the operation of SIP and IP networks. DNSSEC extensions to DNS are used to resolve DNS security problems. DNSSEC, originally defined in RFC 2535 [51], was designed to verify the authenticity and integrity of query results from a signed zone. It uses a public key (asymmetric) cryptography and a special set of DNS RR to enable a DNSSEC aware client (resolver) to be sure that a validated response is the one intended by the owner of the requested domain name. This helps to avoid a situation when a server tries to redirect the client to a malicious server. However, DNSSEC works perfectly only if the entire DNS hierarchy is signed. This cannot be expected in the near future, because the root zone has been signed recently, in July 2010. Currently, a DNNSEC implementation strategy expects a chain of isolated islands of DNS security interconnected through delegation points.

A secured SIP communication environment is a complex task which can be achieved by implementing various security mechanisms at different levels of a communication stack. Security cannot be guaranteed with a single mechanism or protocol. To improve security of a SIP architecture, the combination of mechanisms mentioned above have to be used (see Figure 5).

4.4. High Availability. SIP technology is used frequently now. SIP communication systems, especially large-scale ones, have to minimize service unavailability due to software or hardware failures. Thus, the high availability (HA) is an important feature of an communication architecture. To avoid a single point of failure, backup components together with suitable failover mechanisms have to be implemented. HA is not a specific technology, but a goal that should be achieved. SIP HA concerns the uninterrupted availability of core SIP network components that provide SIP services despite of link outages, device failures, or attacks.

From a SIP network architecture point of view, there have been several design approaches identified to achieve high availability of SIP network components [52]. The first approach is based on combination of static lists or DNS SRV (service record). The second approach uses hardware-or-software based load balance mechanisms. The third approach utilizes an advanced failover intelligence in redundant devices. All of these approaches provide network resilience through redundant design deployment, where typically

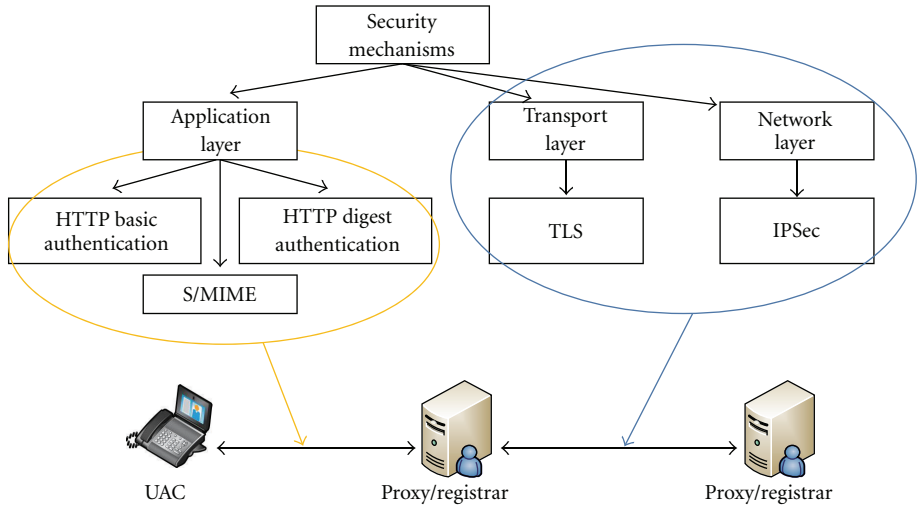

Figure 5: SIP security mechanisms.

a backup system is deployed which is capable of providing the service in case of failure of a master or primary system.

4.4.1. Static Lists. This approach utilizes feature of SIP clients that can be configured with a static list of alternative SIP servers. The client makes attempts to contact a primary server, and if it fails to respond, then the client contacts the backup server. However, this SIP UA feature is not widely supported.

DNS-based HA. DNS-based HA is a quite simple solution for utilizing primary and secondary SIP servers, whereas the HA mechanism DNS service with correctly configured SRV records is used. DNS NAPTR and SRV records allow the operator to adjust the weight and priority of individual records associated with the primary and secondary servers and to control the DNS resolving process. The DNS server handles SIP client requests and provides DNS responses, which provide also the information about the assigned weights and priorities. The SIP client then chooses which SIP server to contact. This solution requires SIP clients that support DNS SRV, support for DNS A records resolving is not sufficient. DNS-based HA is simple; however, there are issues that arise from some DNS service features, such as caching of DNS resolving answers, slower reaction, and adoption to failure changes.

4.4.2. Load Balancing. SIP load balancing is a mechanism for distributing SIP traffic to multiple SIP servers in order to achieve optimal performance scalability, to avoid overload, and to support HA of services. Hardware-based load balancing solutions are implemented in commercial products. Software-based solutions use a special purpose server whose main task is to dispatch the traffic to the next hop SIP server. These traffic dispatchers (load balancers) may use different selection mechanisms, such as round-robin (RR) scheme. Another approach uses weighted or adaptive balancing schemes, in which requests are distributed proportionally, following the assigned weights. Implementing traffic dispatchers should consider some kind of sanity check mechanisms, which control the accessibility of server. If a failure occurs, it should stop dispatching SIP messages to the unavailable server.

The third approach supposes some intelligence at the devices that are protected. The Virtual Router Redundancy Protocol (VRRP), standardized in RFC 2338 [53], provides the election process, where one device is elected to be active and the other to be standby. When using VRRP heartbeat messages, an active server is monitored by a standby server. When the active server fails, a standby server takes the role of the active one. Both servers have the same virtual IP address, but the different real IP addresses. VRRP is a standardized mechanism suitable for a pair of redundant (1 + 1) servers. VRRP has an open source implementation called VRRPd. As an alternative to VRRPd, the open source linux implementation of the CARP (Common Address Redundancy Protocol) (UCARP) can be used. CARP is an open source alternative of the Cisco proprietary HSRP, which is mainly implemented for BSD (Berkeley Software Distribution). Operation mechanisms are very similar to those of VRRP. However, VRRP and CARP have some implementation limitations; the standby backup server must decide correctly when the active server does not work. When the active server is "dead," the failover mechanism must make sure that a "dead" device will not receive any further messages. This is a challenging task, which needs actualization of the different network information, such as LAN switching tables, ARP tables, and DNS caches. Another issue is how to replicate the stateful SIP knowledge from a previously active server to the standby server. This has not been considered in the specification. Thus, VRRP and CARP are usually suitable for layer 3 connectivity checking.

There are other software-based solutions which allow to monitor the availability of each other. The Linux HA project [54] is one of them. The Linux HA project provides a high-availability (clustering) solution for Linux, FreeBSD, OpenBSD, Solaris, and Mac OS X. The project's main software is Heartbeat, a GPL-licensed portable cluster management program for HA clustering, which is actually available

in version 3.0.4. Heartbeat is a daemon that provides communication and membership cluster infrastructure services. Heartbeat provides the information about presence or disappearance of a peer process on other machines of a cluster. On the top of a heartbeat infrastructure, the Pacemaker cluster resource manager has to be deployed. Pacemaker is an open source HA resource manager suitable for both small and large clusters [55]. Pacemaker starts and stops services that a cluster makes HA. There are also alternatively solutions, for example, the Open Telecom Platform (OTP) developed by Ericsson and the open source OpenSAF service HA that can be used. However, there is still an open question on server's synchronization. A Distributed Replicated Block Device (DRBD) can be used for it. DRBD is a software-based, replicated storage mechanism mirroring a content of block devices, such as hard disks, partitions, and logical volumes between hosts in a cluster in real time. DRBD can be seen as a network-based raid system. However, it does not replicate dynamic states, which are usually kept in the RAM. Usually, DBRD is used for synchronizing database servers in a cluster. Therefore, some mechanisms that keep the SIP state information among several SIP servers synchronized are required. This can be achieved with an entity which replicates SIP messages.

Our testing of individual mechanisms simulating hardware failures declares that the average time of unavailability with UCARP was 3,75, with VRRPd 4,12 seconds, for heartbeat 4,49 seconds and for heartbeat with the pacemaker 5,7 seconds. Optimization of configuration parameters allows to decrease the time of failover to 2,25 seconds only for heartbeat. For simulation of software failure with heartbeat and pacemaker, the following results have been obtained: 4,75 seconds for pacemaker with standard settings and 2,73 seconds for optimized configuration.

5. Proposed SIP Communication Platform

In this section, we identify key components of a SIP-based converged platform, which meet the requirements described above. The platform can deliver SIP services such as voice and video over IP, voicemail, conferencing, instant messaging with presence, and it allows designing, implementing, and deploying new integrated services. The platform supports near-end and far-end NAT traversal for signalling and media flows. The communication platform is based on IETF standards and uses open source software [56] components.

5.1. Features and Services Offered by the Platform. The platform provides the following services:

(i) audio and video calls,

(ii) media services such as audio conferencing, voicemail, and IVR,

(iii) IM and presence based on SIMPLE,

(iv) it allows programming new services through standardized APIs of applications servers or developing new modules and services using other means (such as proprietary APIs),

(v) NAT traversal solutions allowing connections for/to users located behind a NAT,

(vi) interconnectivity with other IP multimedia systems (Skinny, H.323) or messaging platforms (XMPP),

(vii) registry and lookup service used to register and locate SIP services,

(viii) multiple SIP devices per SIP account with parallel forking.

The platform may be redundant, since it should reuse the HA mechanisms or underlying layered redundant network design. However, redundancy requires additional investments necessary for redundant servers and network components. It is highly recommended that deployment includes DNS server. The platform enables the interconnection with other types of communications networks (such as GSM or PSTN/ISDN), but additional investments in a specialized hardware is required (e.g., network interfaces cards, routers with telecommunication network interfaces). The platform can be interconnected with other IP communication platforms like H.323 or IMS (e.g., OpenIMS). From the security point of view, the platform uses digest authentication and credentials are stored in a database server. These credentials should be used for different purposes (registration, call admission, IM, presence, routing, etc.). SIP components may provide some protection mechanisms against DoS attacks (blacklist, filters), but security should include network infrastructure (such as firewalls and perimeter routers with whitelists). The platform may provide signalling and media flow protection using TLS and SRTP for clients whose support it. It can be easily extended with load-balancing techniques. The platform provides IPv4 and IPv6 connectivity.

5.2. Components of the Platform. The platform consists of a number of different software components interoperating together to provide the required services and features. These software components are assumed to run on a standard PC or server computer (no specialized hardware). The following components have been identified (see Figure 6).

5.2.1. SIP Endpoints (Applications or Clients). SIP clients enable their users to access platform services. The endpoints should at least provide HTTP digest authentication, audio or video calls, presence, and messaging based on SIMPLE. Advanced SIP endpoints may use TLS and SRTP security features, contact management through XCAP, and direct voicemail access. The client may use STUN NAT.

5.2.2. SIP Server. The SIP server provides the functionalities of a SIP registrar and a SIP proxy server. A SIP server acting as the SIP registrar allows registration and authentication of clients with HTTP digest authentication schema at least. Credentials together with location data (IP addresses and ports) are stored in a database. Different database types should be supported. The SIP server acting as the SIP proxy should allow fast and powerful SIP signalling processing and provide flexible configuration of SIP signalling processing, which helps to integrate other server entities, such as application, media servers, and gateways where SIP communication

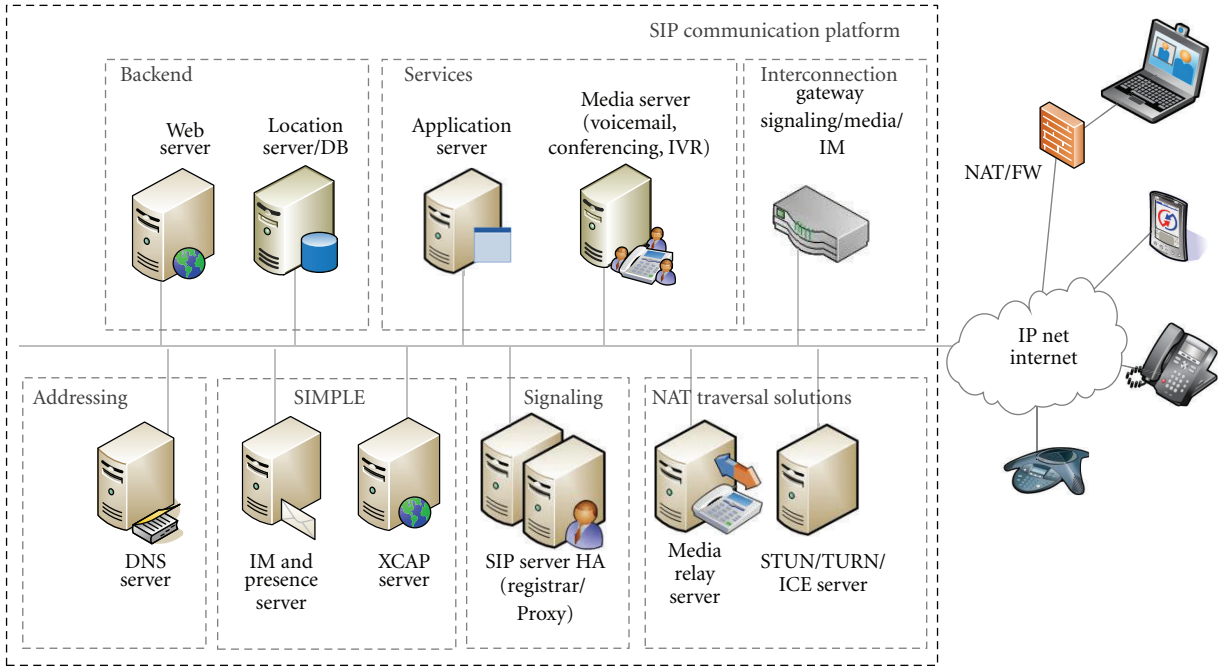

Figure 6: Identified platform components.

interface is used. If more flexible call handling is required, the extendibility of main functionalities of the SIP server should be supported (such as SIP API and programming language). The SIP server should support different network layer protocols (IPv4 and IPv6), different transport protocols (UDP, TCP, SCTP for inter-SIP proxy connectivity), and security protocols (TLS for a hop by hop connectivity). For testing purposes, the self-signed certificates may be used. For production deployments, certificates of the well-know authority have to be used. The SIP server should support DNS lookups as defined in RFC 3263 [30]. The SIP server should support generation of call detail records.

5.2.3. Media Server. A media server provides functions required for media services processing, such as playing announcements, voicemail recording, voicemail storage, voicemail access and play out, media mixing, media transcoding, and audio conferencing. The media server should support self service provisioning and customization. The media server should support different network, transport, and security protocols, which can be used by clients to access to platform services. Provisioning voicemail account should be integrated within the SIP server account management functions.

5.2.4. SIP Presence and IM Server. These servers are part of a SIMPLE architecture and provide centralized solution for handling presence states of users and instant messaging delivery as it has been defined in the SIPMLE framework. For the platform, we use centralized presence solution, rather than a peer-to-peer presence one.

5.2.5. XCAP Server. An XCAP server is part of SIP presence solutions and allows a client to read, write, and modify presence data (such as buddy list) stored in XML format on a server and to synchronize with multiple presence UAs. The XCAP server is also used for handling presence authorization policies. The access to the XCAP server is over HTTP and should be realized through a SIP client interface (have to be implemented) or through a user personal HTTP interface.

5.2.6. Application Server (AS). A SIP AS is used for providing SIP services and applications development and hosting. The SIP AS should provide some of the APIs designed for SIP services development [57]. If required, the SIP AS may act as a fully operating SIP server, but usually the AS is used for developing and provisioning value added services such as click2call, web conferencing, and third party call controller.

5.2.7. Gateway. The purpose of a gateway is to provide interconnection to other platforms. There are different types of gateways depending on required functionality. Our platform uses software packages that allow deploying a signalling gateway for interconnection between SIP and H.323 or SCCP networks and deployment of an IMP gateway for the interconnection of SIMPLE with XMPP. Connection to other types of network with different communication stacks requires special hardware, such as network interface in the gateway.

5.2.8. Components of NAT Traversal Solution. NAT traversal components provide the functions required for correctly processing signalling and media flows for users behind the NAT in the both directions. We propose to implement near-end traversal solutions (client-based) represented by deploying STUN, TURN servers, and clients that incorporated these solutions into a SIP client software. Servers have to be placed in the public part of the network. These servers allow clients to detect presence of a NAT device together with the type of

NAT. Using near-end NAT solution is limited to a few open source solutions that are currently available. Therefore, we propose to implement far-end traversal solution (a server based). Far-end NAT traversal is implemented by media relay servers, which in cooperation with the SIP server allow media streams to bypass the NATs. Using media relay servers does not require SIP clients to be aware of NAT environment. Moreover, it works with any type of NAT. The media relay server should be used as the IPv4 and IPv6 interworking entity that ensures the interconnectivity of clients using different IP protocols. However, a media relay server has to be scaled properly for the total volume of media traffic. Thus, the media server performance for many concurrent media streams can be solved with the use of multiple servers and some kind of load balancing mechanism. We also recommend to use far-end NAT solutions in situations where an IP client with running SIP UA uses several network interfaces with several IP addresses, but without the possibility to setup connectivity preferences.

5.2.9. Database Server. The database server is used for storing client location data, authentication credentials, and personal certificates (if used). The database is used together with other components to realize authentication, authorization, and accounting (AAA) functions. The database in a production service has to be secured following adequate best practices and using IP network infrastructure components (such as firewalling and access lists), underlying operating system means (e.g., IP tables), and the recommended configuration.

5.2.10. DNS Server. The DNS server provides the domain definitions records, and it supports NAPTR and SRV resource records for all of the platform components, enables to define supported transport protocols and to locate address of each of the servers. If the platform is going to be interconnected with the PSTN, then the DNS act as ENUM mapping server. The DNS may provide a front end load balancing service. The DNS server should apply DNNSEC extensions to provide validated DNS responses. However, the DNSSEC deployment depends on the support of DNSSEC on authoritative DNS servers or on the extension of DNSSEC implementation for a particular country.

The list of components provided identifies logical components of the advanced SIP communication platform. An actual real implementation may differ depending on the specific product selected and design goals for a specific realization of the platform.

5.3. A Survey of Available and Relevant Open Source Products

5.3.1. SIP Clients. As a minimal requirement, an SIP client should support core SIP functionalities, including audio and video calls, IM, and presence. Advanced SIP clients should support XCAP, NAT traversal (otherwise far end NAT feature of the platform have to be used implemented), support of some security mechanisms (TLS and SRTP), or multiple transport protocols. There is a number of SIP software clients currently available under open source or free license. The following table (see Table 1) identifies and provides feature overview of nine open source SIP clients available for different operating system (OS) platforms such as Microsoft Windows OS, Android OS, Linux OS, or Mac X OS. We have analyzed SIP clients that have not been released for more than two years. However, it has to be mentioned that there are several SIP clients, which have not been analyzed from several reasons. A client has not a new release for a couple years, thus we suppose it has not been evolving (such as Twinkle, KPhone, miniSIP, and OpenSoftPhone), it is supposed to be a SIP testing tool (QJSimple), or it implements only messaging (Pidgin, Empathy). There are six clients for Microsoft Windows OS, six clients for Linux OS, four clients for Apple Mac X OS, and two new emerging clients for Android OS. The clients provide a messenger-like GUI interface suitable for contacts management, but only one client supports XCAP to store contacts on a XCAP server (Jitsi). The implemented functionalities of the analyzed clients differ; in general, almost all clients support a broad spectrum of audio codecs and SIMPLE, at least SIMPLE page mode instant messaging. The support for video communication is not widely adopted, and a SIMPLE session mode supports only one client (Jitsi). Implementation of security mechanisms has been grown, but still it is not a generic feature of the available SIP clients. Six out of nine clients have implemented TLS and only five out of nine clients implement SRTP with a standardized key exchange mechanism. Only two clients, Jitsi and Linphone, support IPv6.

The support for basic audio/video sessions and page mode messaging is smooth. However, the advanced SIP functionalities such as SIMPLE or TLS provide a different level of standard adoption, and the interoperability of different clients is poor. Based on the required functionalities and our experience, we have evaluated the Jitsi communicator (SIP communicator before) followed by the Blink client as the best and most featured client for Microsoft Windows, Mac X, and Linux OSs. For the Android OS, we consider cSipSimple followed by the SipDroid client as the most appropriate ones, even though a Jitsi android port is now on the way. The Jitsi communicator is a feature-rich audio/video and chat multiprotocol communicator. It was originally developed at the University of Strasbourg, and it is now provided under the LGPL open source license. Finally, we can state that it is very difficult to predict the further development of open source projects. As an example, the Jitsi communicator has not been released as a stable version yet.

5.3.2. SIP Servers. The SIP server should support both SIP registrar and SIP proxy functionalities. The SIP server should be modular and flexible, allowing powerful SIP call handling. It should support security mechanisms as TLS and connectivity with different types of authentications servers. The server should provide a NAT traversal solution. For SIP servers, there exist two widely used solutions: Kamailio and OpenSIPS. Both are successors of the OpenSER project (fork of the SER project). The SER (SIP Express Router) server known as the Sip-Router (since 2008) is a SIP proxy, which has numerous worldwide implementations and is supported by a wide open source community of developers, maintainers,

Table 1: SIP client analysis.

SIP clients	Jitsi	Blink Lite	microSIP	Sfl phone	QuteCom	Ekiga	Linphone	SIPDroid	Csipsimple
www	jitsi.org	Icanblink .com	microsip .org.ua	sflphone.org	qutecom.org	ekiga.org	linphone.org	sipdroid.org	code.google .com/p/ csipsimple
License	LGPLv3	GPLv3	GPL	GPLv3	GPL	GPL	GPLv2	GPLv3	GPL
Release	1.0-beta1-3689	0.2.7	1.8.2	0.9.13	2.2.1	3.2.7	3.4.3	2.3	0.2.3
Date	4.10.2011	25.5.2011	15.10.2011	5.4.2011	9.5.2011	31.5.2010	25.3.2011	24.6.2011	19.6.2011
Operating system	Win, Linux, Mac X	Win, Linux, Mac	Win	Linux	Linux, Mac OS, Win,	Linux, Win	Linux, Win, Mac, Android, WebOS	Android	Android
Multiple accounts	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No
Audio	Yes G722, G728, speex, GSM, iLBC, PCMU, PCMA, DVI4	Yes G722, speex, GSM, iLBC, PCMU, PCMA	Yes Speex, PCMU, PCMA, GSM, iLBC, SILK, G.722	Yes G722, speex, GSM, PCMU, PCMA, celt	Yes iLBC, PCMA, PCMU, G.722, Licensed: AMR, AMR-WB, G.729	Yes G722, G728, G726, G721speex, GSM, PCMU, PCMA, celt	Yes speex, GSM, PCMU, PCMA, G.722	Yes G.722HD, PCMU, PCMA, speex, GSM, BV16	Yes Speex, PCMU, PCMA, GSM, iLBC, SILK, G.722
Video	Yes H264, H263	No	No	No	Yes H.263	Yes H264, H263, H261 theora	Yes H264, H263, H263-1998, MPEG4, theora	Yes	No
DNS SRV/NAPTR	Yes	Yes	Yes	No	Yes	Yes	Yes	—	—
SIMPLE	Yes	No	Yes	No	Yes	Yes	Yes	—	Yes
IM	Page/session mode	—	Page mode	Page mode	Page mode	Page mode	Page mode	—	Page mode
XCAP	Yes	Yes	No	No	No	No	No	—	—
TLS	Yes	Yes	Yes	Yes	No	No	Yes	—	Yes
Media encryption/key manag.	SRTP/ZRTP, MIKEY on progress	SRTP	SRTP	SRTP/ZRTP	SRTP/key exchange over Everbee	No	No	Not officially confirmed	SRTP/ZRTP
NAT	No	ICE	STUN, ICE	STUN	—	STUN	STUN	STUN	STUN, ICE

Table 1: Continued.

SIP clients	Jitsi	Blink Lite	microSIP	Sfl phone	QuteCom	Ekiga	Linphone	SIPDroid	Csipsimple
Communication protocols	SIP, XMPP, AIM/ICQ, MSN, Yahoo!	SIP	SIP	SIP, IAX	SIP, MSN, YIM, AIM, ICQ, XMPP	SIP, H.323	SIP	SIP	SIP
Transport protocols	TCP, TLS, UDP	TCP, TLS, UDP	TCP, TLS, UDP	UDP, TLS/SSL	UDP	UDP	UDP, TCP, TLS	UDP, TCP	UDP, TCO, TLS
IPv4/IPv6	Yes/Yes	Yes/No	Yes/No	Yes/No	Yes/No	Yes/No	Yes/Yes	—	—
Voice mail	Yes	Yes	No	Yes	—	No	No	—	—

Table 2: SIP PROXY server analysis.

SIP server	Kamailio/SIP router	OpenSIPS
www	kamailio.org/sip-router.org	opensips.org
License	GPL	
Last release	3.2.0	1.7.0
Operating system	Linux	Linux
SIP registrar	Yes	
SIP proxy	Yes	
B2BUA	No	Yes
Low-level transaction access	Yes	
Forking	Yes	
Modular architecture	Yes	
NAT traversal	Yes with RTPProxy, Mediaproxy	Yes with RTPProxy, Mediaproxy
Keepalive	Yes	
Authentication and authorization against database	MySQL, PostgreSQL, SQLLite, UnixODBC, BerkeleyDB, Oracle, text files, LDAP, RADIUS, DIAMETER	MySQL, PostgreSQL, UnixODBC, BerkeleyDB, Oracle, text files, LDAP, RADIUS, DIAMETER
Signalling protocols	SIP	
Transport protocols	UDP, TCP, TLS, SCTP	UDP, TCP, TLS, SCTP
IPv4/IPv6	Yes/yes	
Administrative web GUI	Yes SIREMIS	Yes OpenSIPS-CP, SerMyAdmin
DNS	NAPTR, SRV, ENUM	
Multidomain	Yes	
SIMPLE (presence and IM)	Yes	
XCAP server	Yes Embeded	Yes With OpenXCAP server
Gateway	XMPP, MSN, SMS	XMPP, MSN, SMS
Standardized SIP service API	CPL, SIP servlet API (with Seas)	CPL, SIP servlet API
Other service tools	Perl, Lua, Python	Perl
Media services	No	
Carrier class routing	Yes	
Load balancing	Yes	
Attack protection	Yes IP blacklists	

and supporters. SER has been developing since 2001 and released in 2004 under the GPL open source licence. The history behind its development is a bit complicated and has led to establishing several viable forked solutions, such as OpenSER in 2005 (renamed as Kamailio in 2008) and OpenSIPS in 2008. Currently, developers of Kamailio and SIP-router projects are trying to merge these two projects. In a present time, Kamailio and SER can be considered identical (starting from 3.0 releases). The OpenSIPS fork is still going along its own development way.

Kamailio (former OpenSER) and OpenSIPS are both very popular and viable SER-based SIP proxy solutions. Even though the former OpenSER project was separated, both solutions provide very similar core functionalities (see Table 2). Both are released under the GPL license and provide a modular design architecture which is able to handle thousands of call setups per second, supports similar set of protocols (TCP, UDP, SCTP), AAA functionalities (MySQL, PostgreSQL, Oracle, Radius, LDAP), and SNMP monitoring. Both servers support TLS, which may be used to extend a trust concept to a multidomain level, where a mutual agreement can establish trusting relationships. Verification data (certificates and chain of certification authorities) are locally stored. Both servers support ENUM, least cost routing, load balancing, and routing fail-over. Both provide a scripting configuration language which allows flexible SIP routing configuration. They adopted CPL SIP APIs and support additional scripting languages to provide application-level

functionalities. Differences are only in new modules developed after 2008. For example, during 2011 Kamailio IMS module extensions have been introduced, which allow to build an IMS testbed with the latest Kamailio SIP server releases. Both solutions are mature open source implementations of a SIP server which unifies voice, video, IM, and presence services in an efficient way.

The OpenSIPS solution can benefit from a support for an MSRP relay server used in MSRP IM relaying scenarios of session messaging over NAT. Kamailio can benefit from all-in-one integrated SIMPLE solutions. There are also other open source SIP server implementations, but they are still more in an experimental phase, for example, YXA server and OpenJSIP. There are also other well-known solutions such as the Asterisk B2BUA, which provides good media handling capabilities.

There is a few of widely adopted and supported server solutions, which provide advanced media services. They are primarily designed as advanced telephony platforms targeting the PBX market. These platforms are usually designed as standalone solutions which support communication using audio, video, text, or other form of media. However, from the architecture point of view, they are not designed as SIP proxy servers with flexible message routing rules. In our platform, they play a role as *media servers* (voicemail, conferencing, IVR), *gateways*, or *B2BUA entities*. Four solutions have been analyzed: Asterisk, FreeSWITCH, Yate, and SEMS. We propose to use Asterisk, because it has a high-quality documentation and a larger community of supporters. From the experimental point of view, we have to mention solutions, such as FreeSWITCH, and Yate, which both offer interesting features (see Table 3).

The use of gateways depends on the deployment scenario. For interconnection at the signalling level only, Asterisk, FreeSWITCH and Yate may be used. For interconnecting at media level, special dedicated hardware had to be used. Asterisk supports a broad portfolio of network interface cards, so it is most suitable as a solution. A dedicated gateway such as the Cisco ISR router is an alternative solution, which does not require an additional gateway. For the interconnection with XMPP, Kamailio/OpenSIPS with XMPP support is proposed.

5.3.3. SIP Presence and IM Server. SIMPLE allows users to send and receive instant real-time messages and to know the current availability or status of other users. Actually, there are only two open source solutions with an integrated support for SIMPLE presence and IM, Kamailio, and OpenSIPS. The SIMPLE functionalities have been added after the OpenSER project fork; thus, the implementation philosophy is different. The Kamailio server provides all-in-one solution. The SIMPLE functionality is integrated with newly developed presence, messaging, Xcap server and HTTP modules. There is no need for external applications or dependencies, since they are now a part of the Kamailio server. However, together with an *xcap_client* module an external Xcap server may be used. OpenSIPS integrates SIMPLE functionalities which are built in presence modules. Xcap server functionalities are excluded from the main server and it is an OpenSIPS *xcap_client* module which provides connectivity to an external xcap server—the OpenXcap server. The both mentioned solutions may be used as a standalone SIMPLE server with Xcap functionalities that may be used either in IETF SIP architectures, or in IMS testbeds, built on the OpenIMSCore or Kamailio IMS solutions. Currently, there is no other open source SIMPLE-integrated solution. To provide presence only functionalities, the Mobicent Presence Service (MSPS) may be used [58]. MSPS provides presence functionalities to SIP-based networks using standards developed by the Internet Engineering Task Force (IETF), the Open Mobile Alliance (OMA), the 3rd Generation Partnership Project (3GPP), and the European Telecommunications Standards Institute (ETSI).

5.3.4. XCAP Server. Two products have been investigated, the Kamailio SIP server with embedded XCAP/HTTP server functionality or a standalone OpenXCAP server [59].

5.3.5. Application Server (AS). There is a number of open source products available; however, selection depends on which SIP service tools are planned to be used for service development. Create lightweight services defined by the end user himself, the CPL API, may be used. Both the Kamailio and the OpenSIPS servers support it. To develop advanced services with a powerful service logic, more advanced programming API has to be used such as SIP Servlets or JAIN API. The SIP Servlet API provides more flexibility for developing services that integrate SIP and HTTP protocols since a SIP servlet is usually an extension of a HTTP Servlet API. The JAIN API, on the other hand, provides a robust API with many protocols supported, including the telecommunications ones. There are several SIP Servlet or JAIN AS implementations such as:

(i) *SailFin*. Sailfin [60] is a SIP servlet container which extends a widely used GlassFish open source enterprise AS platform developed under the lead of Oracle and provided under GPLv2 license. The SailFin supports a SIP Servlet API technology. The documentation is, however, often obsolete or unavailable because of changing the main project supporter from Sun to Oracle.

(ii) *Mobicents.* Mobicents [58] is an open source VoIP platform certified for JSLEE 1.1 and SIP Servlets1.1 technologies. Mobicents is the SIP AS service platform developed under the lead of RedHat and provided under GPLv2 license. Mobicents subprojects are represented by Mobicents SIP Servlets (GPLv2.1), media server (GPLv2), and SIP presence service (GPLv2).

(iii) *Cipango*. Cipango [61] is an extension of SIP Servlets to the Jetty HTTP Servlet engine. Cipango/Jetty is then a convergent SIP/HTTP application server compliant with both SIP Servlets 1.1 and HTTP Servlets 2.5 standards.

Table 3: Media server analysis.

Media server	Asterisk	FreeSwitch	Yate	SEMS
www	Asterisk.org	Freeswitch.org	Yate.null.ro	www.iptel.org/sems
License	GPL	MPL	GPL	GPLv2+
Last stable rel.	1.8.2.3	1.0.6	3.3.0	1.4.2
Operating system	Linux, Mac OS X, *BSD, Solaris, Windows	Linux, Mac OS X, *BSD, Solaris, Windows	Linux, Mac OS, FreeBSD, Windows	Linux
Wrritten in	C	C/C++	C++	C/C++
Architecture	B2BUA	B2BUA	—	—
Modular	Yes			
NAT traversal	No	STUN	No	—
Authentication, authorization against database	MySQL, PostgreSQL, LDAP, Radius	MySQL, PostgreSQL, LDAP, Radius	MySQL, PostgreSQL, Radius	Diameter
VoIP signalling protocols	SIP, H,323, SCCP, MGCP, IAX, GoogleTalk	SIP, H.323, IAX, SCCP	SIP, H.323, MGCP, IAX, Jingle	SIP
Telephony signalling protocols	ISDN/SS7, FXS/FXO	ISDN/SS7	ISDN/SS7, FXS/FXO, Sigtran	No
Messaging protocol	XMPP	SIMPLE, XMPP	XMPP/Jabber	No
Call encryption	SRTP	SRTP	No	SRTP
Transport protocols	UDP, TCP, SCTP, TLS	UDP, TCP, SCTP, TLS	UDP, TCP, SCTP	UDP
IPv4/IPv6	Yes/Yes	Yes/Yes	Yes/—	Yes/—
Web GUI	Yes	Yes	—	No
SIMPLE	No	Yes	No	No
SIP gateway	Yes	Yes	Yes	No
Audio codecs	ADPCM, PCMU, PCMA, G.722, G.722.1, G.722.1 Annex C, G.723.1, G.726, G.729a, GSM, iLBC, Linear, LPC-10, Speex	CELT, G.722.1, G.722.1C, G.722, PCMU, PCMA, GSM, G.726, AAL2 and RFC 3551, G.723.1, G.729AB, AMR, iLBC, Speex, LPC-10, DVI4, SILK	GSM, speex, iLBC, AMR-NB	PCMU, PCMA, GSM, G.726, iLBCi, speex, adpcm, L16
Video codecs	No	Theora, H.261, H.263, H.263+, H.263++, H.264, MP4	No	No
Transcoding	Yes			
IVR and Announc.	Yes			Ann.
Voice mail	Yes	Yes	—	Yes
Audio conference	Yes			
Call recording	Yes	Yes	—	—
IP/PBX features	Yes			No
CDR	Yes			—
Fax	T.30, T.38	T.30, T.38	—	No
Text to speech	Yes	Yes	—	No
SIP API	No			
Programming languages	With CGI any language, Adhersion	C/C++, Python, Perl, Lua, Java, JavaScript, Erlang, Ruby	Python	C++, Python, DSM

(iv) *WeSIP*. WeSIP [62] is a SIP and HTTP converged application server built on a top of an OpenSER SIP platform, which adds a J2EE layer to OpenSER, so it can benefit from the existing OpenSER modules and features.

There are also solutions which use other means such as external script invocations (Asterisk, Kamailio/OpenSIPS), programming languages such as Perl (Kamailio/OpenSIPS), Lua (Kamailio), Python (Kamailio), Java, or proprietary API such as Adhearsion [63] for the Asterisk, or proprietary scripting language (Astrerisk).

5.3.6. NAT Traversal Components. NAT traversal may include several components. However, only few open source products are available. To deploy STUN, a VOVIDA-based STUN server can be used. This STUN server is available from several Linux distribution repositories (Debian, CentOS, RedHat). Another option is mySTUN package [64]; however, this package has not been evolving for more than two years. Some STUN servers are publicly available. To support NAT traversal for TCP, there is Simple Traversal of UDP through NATs and TCP (STUNT) extension of the Vovida STUN Server available and called STUN-over-TCP. However, it does not work well with symmetric NAT. To support it, TURN or ICE has to be deployed. For TURN implementation, two open source projects have been identified: TurnServer [65] and reStund project [66]. The latest release of the TurnServer is version 0.5 published in June 2011. A free service based on the NUMB server is provided [67]. For using ICE negotiation functionalities, ICE extension for a MediaProxy server can be used. To support the STUN/TURN ICE development, there are several open source libraries, such as PJNATH (PJSIP NAT Helper), ReTURN, and ice4j.

To deploy far-end NAT traversal solution, there are two open source products available, the RTPproxy server [68] and the MediaProxy server [69]. Deployment of an RTPproxy is less time consuming. For communication between a SIP server and an RTPproxy or MediaProxy, a proprietary protocol is used. In contrast, the MediaProxy provides additional functionalities such as TLS support, T.38 fax, RADIUS, and database server logging. The MediaProxy supports ICE negotiation by behaving like a TURN relay candidate, whereas the policy can be controlled from the OpenSIPS configuration. These solutions are closely coupled with Kamailio/OpenSIPS projects that support them by the server modules.

5.3.7. Database Server. The database is used to store users' credentials. It can be also used for storing CDRs (call detail records). In addition, it is used as a location server for a SIP registrar and proxy. There are several open source solutions, such as MySQL, PostrgeSQL, OpenLDAP, and OpenRadius. We recommend to use that one that is supported by several platform components. As the database server, we propose to use the MySQL server, since it has supported connectivity with all other platform components. The MySQL is frequently used database server; moreover, there is a number of information sources and "how-to" implementation guides.

Table 4: Proposed solutions.

SIP server	Kamailio	OpenSIPS
Media server	Asterisk/ FreeSWITCH	Asterisk
SIMPLE and XCAP	Kamailio	OpenSIPS/ OpenXCAP
NAT traversal	RTPproxy	MediaProxy
Application server	Sailfin	Mobicents
Database server	MySQL	MySQL
Gateway	Asterisk	Asterisk
IMP XMPP gateway	Kamailio	Kamailio
DNS server	Bind9	Bind9
DHCP server	dhcp3-server	dhcp3-server

Of course, other solutions can be used, too, but their implementation is more time consuming.

As a *DNS server*, we propose to use a standard BIND9 server, since it provides required functionalities, such as NAPTR/SRV, ENUM, DNSSEC, multidomains, and private trees or public trees. The BIND9 server is available from Linux distribution repositories. As an alternative, the PowerDNS server may be used.

5.4. High Availability and Resiliency. Proposed platform solution can provide high-availability features, which can be implemented in different ways. First, a properly configured DNS HA may be used as a front end HA, as we proposed using a DNS server. The DNS server has to be configured to response with a list of NAPTR and SRV records with a predefined order and preference values (NAPTR), respectively, priorities and weights for SRV records. The second option includes a deployment of load balancers. The Kamailio/OpenSIPS server operating as a stateless SIP proxy together with appropriate modules (dispatcher, carrierroute) can be used. Alternatively, there is a Mobicents SIP load balancer available, which is in fact a simple SIP proxy server. The third option is based on the VRRP protocol [70]. Finally, a server cluster may be deployed which uses a HeartBeat framework [54] with an optional support for a distributed replicated block device (DRBD) [71].

5.5. Proposed Solutions. To popularize SIP technology as a service provider platform, advanced learning, as well as research and experimentation platform, we did not consider commercial products, we analyzed only open source components. The SIP platform provides a living environment that can be used for master degree courses lead by our department. It also allows an easy access to a SIP technology and the advanced services. The platform was aimed to be established with low costs and based on open source software. The SIP platform is open and extendible with new services and communications components.

Having used open source components mentioned above, we built a laboratory prototype environment which has all these features. Even though in our research, we use the Kamailio SIP server (see Table 4, Solution 1), the platform

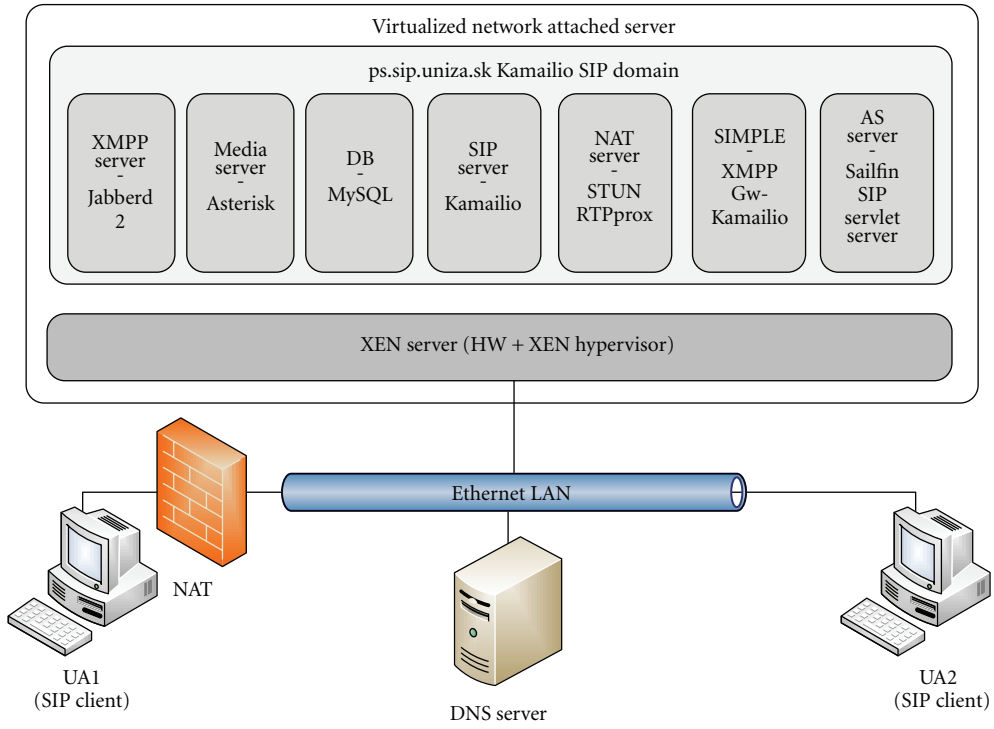

Figure 7: Proposed platform.

can be realized by either of the two fully comparable and alternative solutions and components listed in Table 4.

All the platform components, with a help of Xen server virtualization technology, are running on a single hardware server (see Figure 7). As the OS platform, the 64 bit linux Debian OS release Lenny has been used. Only the XMPP server runs on Ubuntu 10.10 Maverick. All components are based on open source software. As a SIP server, a central part of the Kamailio server is used. The Kamailio SIP server, thanks to flexible routing and configuration functionalities, provides a principal SIP routing logic of the architecture. The Kamailio server acts as a registrar and proxy server of the platform. As a location server and an authentication credential storage, the MySQL server is used (Kamailio *db_mysql* module). Routing logic of the SIP server is configured so that main call routing functionalities are served by the SIP Server itself. If a callee is unavailable (offline, busy, long time ringing), a call is redirected to the media server and a voicemail service of a particular user is invoked through IVR. For this purpose, a real-time integration of the location database with the Asterisk database has been implemented. A voice message is then delivered using an email service to an email box of a callee. A voice message can be also played out using a predefined voicemail SIP address and a user PIN number. In addition, the Asterisk server is used to implement an audio conferencing service. For this, a *MeetMe* application has been used. *MeetMe* supports statically and dynamically created conference rooms, password protected conferences, and conference control mechanisms (e.g., mute). The SIP server is configured to route SIP addresses of the assigned conference service to Asterisk. Messaging and presence services are implemented on a dedicated SIMPLE server with Xcap functionalities (Kamailio with XCAP) using *presence*, *presence_xml*, *xcap_server*, and *xhttp* Kamailio modules. The main routing logic of the SIP server is configured to route corresponding SIMPLE messages to the SIMPLE server. Due to access protection to the Xcap, authentication credentials of the user are used. Hence, the integration of the SIMPLE server with the main location database has been realized. The SIMPLE server also provides the XMPP gateway functionalities (*pua_xmpp*, *xmpp*, and *purple* modules), which enable the integration of two presence and messaging architectures, SIMPLE and XMPP. For the interconnection scenario, we have used an *xmpp* module, the other two have not been finalized yet, or a code bug has been found. The *xmpp* module can act in two modes, a component mode and a server mode. For testing, a dedicated XMPP server has been installed (Jabberd2). Testing of the implementation showed that the development of *xmpp* and *pua_xmpp* Kamailio modules has not been finalized (a server mode has not worked). As far as related to testing services, only messaging has been working without problems. The presence service has not worked properly between XMPP and SIMPLE (in Kamailio version 3.1.5). As an alternative to an XMPP gateway, the OpenSIPS server may be used.

However, during testing OpenSIPS XMPP gateway functionalities (*xmpp* module), an error code inside the *uri_xmpp2sip()* function has been found; hence, it was not possible to utilize correctly module functionalities.

As an example of an integrated SIP/HTTP service, which illustrates SIP service development opportunities, the platform has been extended with the Sailfin AS. As a fully featured alternative, a Mobicent AS can also be used. The Sailfin AS has been used as a SIP servlet development and hosting platform used for a click2call service. Click2call services are services which enable a phone call to be established between two or more involved parties directly from a web page. The click2call service is a good example of a new approach to service creation by SIP technology. The click2call service implements a third party call control entity (3pcc) which is responsible for call initiation.

To address a NAT traversal problem, the STUN server has been implemented to a platform. For special cases, the RTPproxy media proxy has been implemented. The RTPproxy can be alternatively used for IPv4/IPv6 interworking scenarios. Finally, the DNS server has been used with properly defined SIP and XMPP SRV records. The platform is statelessly connected to a cisco call manager (CCM) to route calls through CCM to the Public switched telephone network (PSTN). For calls routed to PSTN, the Remote Party ID header has been inserted. The platform supports TLS with self-signed certificates.

Several testing interconnection scenarios have been performed for implemented services using different SIP clients, simulating intra- and inter-domain connectivity. The signalling message exchange has been captured, analyzed, and evaluated to proof the interconnectivity. For some scenarios, an IMS client (Boghe) which is able to register to a SIP domain and make calls can be used.

In general, the installation of platform components has been straightforward, accessible directly from respective package repositories. Access to special functionalities may require the package compilation. Configuration of platform components, especially routing behavior of Kamailio/OpenSIPS servers requires a deep knowledge of a configuration scripting syntax. We have considered documentation for relevant modules detailed enough. However, "how-to" guides publicly available on projects main portals differ. Our personal view is that the Kamailio project provides more accurate results. On the other side, the two SIP servers are robust and powerful. Documentation for Asterisk is widely available, since Asterisk seems to be the most popular IP PBX with many available extensions and applications. Hence, especially for lighter solutions, other interesting media servers are available, for example, the FreeSWITCH, a scalable open source cross-platform telephony platform.

5.6. Remarks. The two proposed solutions integrate the open source components that could be utilized to create a high performance and feature rich platform. Both proposed solutions are fully comparable. These two proposed solutions consist of a number of independent components which can be implemented in different ways, with different numbers of logically and physically deployed servers depending on a SIP operator strategy and goals. These components are usually deployed as software packages installed in hosting servers running Linux. Additionally, there are powerful open source solutions which integrate many features, for example, SipXecs and Elastix unified communication (UC) solutions. They are provided as a package together with a Linux operating system.

Elastix [72] is an open source project, which has evolved from the Asterisk and which integrates several open source products to provide a unified communications solution available under a GPLv2 license. Elastix includes a number of communication media and features supported by other open source products such as a mail server (Postfix), webmail (Round Cube), CRM (vTigerCRM), IM and presence (OpenFire), fax server (Hylafax), SIP VoIP PBX (Asterisk 1.6) with WebGUI (FreePBX), and a video conference server, all of them running on the CentOS version of Linux.

SipXecs is a highly scalable enterprise-grade unified communication and collaboration solution provided under LGPL license. The development of SipXecs is supported by the open source organization SIPFoundry [73]. SipXecs is SIP-centric software solution that provides PBX telephony services integrated with several open source solutions to deliver an UC package. SipXecs includes a SIP server (SipX), IM and presence (OpenFire), a gateway to other IM, media server for conferencing and voicemail (FreeSWITCH), Automated Contact Distribution (ACD), all of them are manageable by a web-based GUI management interface packaged and shipped running on CentOS linux.

6. Conclusion

The evolution of modern communication technologies offers new communication possibilities. In this paper, we have analyzed key protocols, technologies, and services that can be incorporated into a converged communication platform. Based on a case study, we have analyzed and proposed an open SIP-based communication platform that shows the opportunities for open source products to accomplish this complex task. Open source products we have tested are of good quality and are sufficiently supported with documents, implementation guides and other relevant materials. Their deployment enables to build a flexible, scalable, and powerful multimedia communication solution, which integrates and offers many interesting services, such audio and video calls, conferences, voicemail, presence, and instant messaging. Using open source products in an academic environment enables teachers, researchers, and students to keep in touch with technology innovations.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo et al., SIP: Session Initiation Protocol, RFC 3261, July 2002.

[2] P. Segeč, "Programming SIP services—the SIP APIs," *Acta Electrotechnica et Informatica*, vol. 10, no. 4, pp. 39–45, 2010.

[3] J. Lennox and H. Schulzrinne, Call Processing Language (CPL): A Language for User Control of Internet Telephony Services, RFC 3880, October 2004.

[4] J. Lennox, H. Schulzrinne, and J. Rosenberg, Common Gateway Interface for SIP, RFC 3050, January 2001.

[5] JSR 289 Expert Group, JSR-000289 SIP Servlet 1.1 Final Release, 2008.

[6] JSR-000032, JAIN SIP API Specification, Maintenance Release, 2006, http://jcp.org/aboutJava/communityprocess/mrel/jsr032/index.html.

[7] H. Sinnreich and A. Johnston, *Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol*, John Wiley & Sons, New York, NY, USA, 2nd edition.

[8] http://datatracker.ietf.org/wg/xmpp/charter/.

[9] http://datatracker.ietf.org/wg/simple/charter/.

[10] M. Day, J. Soenberg, and H. Sugano, A Model for Presence and Instant Messaging, RFC 2778, February 2000.

[11] M. Day, S. Aggarwal, G. Mohr, and J. Vincent, Instant Messaging / Presence Protocol Requirements, RFC 2779, February 2000.

[12] B. Cambell, J. Rosenberg, H. Schulzrinne, C. Huitem, and D. Gurle, Session Initiation Protocol (SIP) Extension for Instant Messaging, RFC 3428, December 2002.

[13] B. Cambell, R. Mahy, and C. Jennings, The Message Session Relay Protocol (MSRP), RFC 4975, September 2007.

[14] C. Jennings, R. Mahy, and A. B. Roach, Relay Extensions for the Message Session Relay Protocol (MSRP), RFC 4976, September 2007.

[15] A. B. Roach, Session Initiation Protocol (SIP)-Specific Event Notification, RFC 3265, June 2002.

[16] J. Rosenberg, A Presence Event Package for the Session Initiation Protocol (SIP), RFC 3856, August 2004.

[17] A. Niemi, Session Initiation Protocol (SIP) Extension for Event State Publication, RFC 3903, October 2004.

[18] J. Rosenberg, The Extensible Markup Language (XML) Configuration Access Protocol (XCAP), RFC 4825, May 2007.

[19] E. Burger, J. van Dke, and A. Spitzer, Basic Network Media Services with SIP, RFC 4240, December 2005.

[20] C. Boulton, T. Melanchuk, and S. McGlashan, Media Control Channel Framework, RFC 6230, May 2011.

[21] C. Jennings, F. Audet, and J. Elwell, Session Initiation Protocol (SIP) URIs for Applications such as Voicemail and Interactive Voice Response (IVR), RFC 4458, April 2006.

[22] J. Rosenberg, A Framework for Conferencing with the Session Initiation Protocol (SIP), RFC 4353, February 2006.

[23] M. Barnes, C. Boulton, and O. Leven, A Framework for Centralized Conferencing, RFC 5239, June 2008.

[24] O. Levin and R. Even, High-Level Requirements for Tightly Coupled SIP Conferencing, RFC 4245, November 2005.

[25] J. van Meggelen, J. Smith, and L. Madsen, *Asterisk: The Future of Telephony*, O'Reilly, 2nd edition.

[26] S. McGlashan, T. Melanchuk, and C. Boulton, An Interactive Voice Response (IVR) Control Package for the Media Control Channel Framework, RFC 6231, May 2011.

[27] T. Berners-Lee, R. Fielding, U. C. Irvine, and L. Masinter, Uniform Resource Identifiers (URI): Generic Syntax, RFC2396, August 1998.

[28] A. Gulbrandsen, P. Vixie, and L. Esibov, A DNS RR for specifying the location of services (DNS SRV), RFC 2780, February 2000.

[29] M. Mealling and R. Daniel, The Naming Authority Pointer (NAPTR) DNS Resource Record, RFC2915, September 2000.

[30] J. Rosenberg and H. Schulzrinne, Session Initiation Protocol (SIP): Locating SIP Servers, RFC 3263, June 2002.

[31] D. Senie, Network Address Translator (NAT)-Friendly Application Design Guidelines, RFC3235, January 2002.

[32] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, Session Traversal Utilities for NAT (STUN), RFC 5389, October 2008.

[33] R. Mahy, P. Matthews, and J. Rosenberg, Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN), RFC 5766, April 2010.

[34] J. Rosenberg, Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols, RFC 5245, April 2010.

[35] J. Rosenberg, Indicating Support for Interactive Connectivity Establishment (ICE) in the Session Initiation Protocol (SIP), RFC 5768, April 2010.

[36] The UPnP Forum, http://www.upnp.org/.

[37] C. Jennings, R. Mahy, and F. Audet, Managing Client-Initiated Connections in the Session Initiation Protocol (SIP), RFC 5626, October 2009.

[38] W. Werapun, A. A. E. Kalam, B. Paillassa, and J. Fasson, "Solution analysis for SIP security threats," in *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS '09)*, pp. 174–180, April 2009.

[39] I. Dolnák, "Denial of service attacks in Voice over IP networks," in *Proceedings of the Research in Telecommunication Technology Workshop (RTT '10)*, VŠB-Technical University of Ostrava, Velké Losiny, Czech Republic, September 2010.

[40] J. Peterson, S/MIME Advanced Encryption Standard (AES) Requirement for the Session Initiation Protocol (SIP), RFC 3853, February 2011.

[41] J. Kuthan, J. Floroiu, H. Schulzrinne, S. Sisalem, and U. Aben, *SIP Security*, John Wiley & Sons, New York, NY, USA, 2009.

[42] C. Jennings and J. Fischl, Certificate Management Service for the Session Initiation Protocol (SIP), RFC 6072, February 2011.

[43] T. Dierks and C. Allen, The TLS Protocol, RFC2246, January 1999.

[44] E. Rescorla and N. Modadugu, Datagram Transport Layer Security, RFC4347, April 2006.

[45] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobsion, RTP: A Transport Protocol for Real-Time Applications, RFC 3550, July 2003.

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

[47] F. Andreasen, M. Baugher, and D. Wing, Session Description Protocol (SDP) Security Descriptions for Media Streams, RFC4568, July 2006.

[48] J. Arkko, E. Carrara, F. Lindholm, K. Naslud, and K. Norrman, MIKEY: Multimedia Internet KEYing, RFC 3830, August 2004.

[49] J. Fischl, H. Tschofenig, and E. Rescortla, Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS), RFC 5763, May 2010.

[50] P. Zimmermann, A. Johnston, and J. Calls, ZRTP: Media Path Key Agreement for Unicast Secure RTP, RFC 6189, April 2011.

[51] D. Eastlake, Domain Name System Security Extensions, RFC 2535, March 1999.

[52] Cisco Inc., Overview of High Availability in SIP-based Voice Networks, http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/callc_c/sip_c/sipha_c/hachap1.htm.

[53] S. Knight, D. Weaver, D. Whipple, and R. Hinden, Virtual Router Redundancy Protocol, RFC 2338, April 1998.

[54] The Linux-HA, http://www.linux-ha.org/.

[55] Pacemaker, A scalable High-Availibility cluster resource manager, http://clusterlabs.org/.

[56] The Open Source Initiative (OSI), http://www.opensource.org/.

[57] A. Johnston, R. Sparks, C. Cunningham, S. Donovan, and K. Summers, Session Initiation Protocol Service Examples, RFC 5359, October 2008.
[58] Mobicents project, http://www.mobicents.org/.
[59] OpenXCAP—Free XCAP server for SIP SIMPLE, http://openxcap.org/.
[60] SailFin Project—Project Kenai, http://sailfin.java.net/.
[61] Cipango, http://www.cipango.org/.
[62] WeSIP, http://www.wesip.com/.
[63] Adhearsion: Open-Source Telephony Development Framework, http://adhearsion.com/.
[64] MySTUN server, http://developer.berlios.de/projects/mystun/.
[65] The TurnServer project—open-source TURN server implementation, http://turnserver.sourceforge.net/.
[66] ReStund, http://www.creytiv.com/restund.html.
[67] Numb STUN/TURN Server, http://numb.viagenie.ca.
[68] RTPproxy, http://www.rtpproxy.org.
[69] MediaProxy—Fast and scalable RTP relay, http://mediaproxy.ag-projects.com/.
[70] The Keepalived project, http://www.keepalived.org/.
[71] Distributed replicated block device project, http://www.drbd.org/.
[72] Elastix, The Open Source Unified Communications Server, http://www.elastix.org/.
[73] The sipXecs Enterprise Communications System, http://www.sipfoundry.org/.