International Journal of Disaster Recovery and Business Continuity
Vol. 1, No. 1, February, 2010

High Availability for SIP: Solutions and Real-Time Measurement Performance Evaluation

Georgios Kambourakis[1], Dimitris Geneiatakis[1], Stefanos Gritzalis[1],
Costas Lambrinoudakis[1], Tasos Dagiuklas[2], Sven Ehlert[3], and Jens Fiedler[3]

[1]*Department of Information and Communication Systems Engineering*
University of the Aegean, Karlovassi, GR-83200 Samos, Greece
{gkamb, dgen, sgritz, clam}@aegean.gr

[2]*Dept. of Telecommunication Systems & Networks*
Technological Educational Institute (TEI) of Mesolonghi, Nafpactos 30300, Greece

[3]*Fraunhaufer FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany*
{sven.ehlert, jens.fiedler}@fokus.fraunhaufer.de

Abstract

SIP is rapidly becoming a standard for service integration within a variety of wireless and wireline networks. In this regard high availability, reliability and redundancy are key factors for any SIP based infrastructure. In an adverse environment, especially the Internet and foreseeable 3GPP IMS, high availability solutions are of major importance for SIP network components to smoothly mitigate call increments, device failures, misconfigurations, physical disasters and throttle active attacks. This paper proposes a practical and transparent failover solution for SIP and RTP-Proxy servers. We demonstrate that both methods work properly and increase stability and availability of such systems. Furthermore, high availability solutions are enhanced through the employment of easy to implement load balancing schemes. All the proposed solutions are technically analyzed and evaluated via properly designed test-beds, showing fine performance in terms of service times.

Keywords: *Session Initiation Protocol (SIP); SIP architectures; SIP Redundancy and Failover Architectures; SIP Load Balancing.*

1. Introduction

Session Initiation Protocol (SIP) [33] is an open signaling protocol for establishing any type of real-time communication session. A SIP session can consist of voice, video, or instant messaging, and can be employed in any device that people use for communicating e.g., IP phone, laptop computer, Personal Digital Assistant (PDA), cell phone, or Instant Messaging client. SIP realizes a communication environment where central servers do not only know how to reach an individual's cell phone, work phone etc, but also his instant messaging application, e-mail, and PDA. Moreover, servers are aware of the communication preferences and capabilities of communicating parties as well, and can smartly alert a called party when someone is trying to reach him. Finally, phone calls to a busy person can be intelligently rerouted to another person depending on a number of correlated factors such as time of day, whether the called person is planned to be away on vacation, or whether one or more of his modes of communication is inaccessible. Without doubt, these capabilities make SIP a basic component of foreseeable ubiquitous realms. In a nutshell SIP really provides the intelligence that makes these advanced communications capabilities possible. This is why SIP has been adopted by various standardization organizations as the de-facto protocol for both wireline and wireless world in the Next Generation Networks (NGN) era. For instance, 3GPP's IP Multimedia Subsystem (IMS) [1] employs SIP for call control to support thousands or even millions of users.

But while the standards and products for providing Internet Telephony communications in general and SIP services in particular have reached a mature state, experience in deploying concepts and technologies for securing and ensuring the reliability of VoIP infrastructures and the provision of their services is still in its infancy [17, 39]. It should be also remembered that VoIP technologies are very similar in their nature to Web and email services. This is due to the fact that VoIP services are based on standardized and open technologies like SIP using software servers reachable through the Internet, and often provided over general purpose computing hardware. Therefore, at minimum, such services are exposed to the same security threats as Web services. These include Denial of Service (DoS) attacks and spam on one side, and unavailability of the services due to network, hardware or software failures, planned downtime for maintenance, and catastrophic failure on the other. Nevertheless SIP-based VoIP services are of a nature that requires them to be available at a possible maximum. This is particularly true as today's networks are fast evolving towards IP Convergence (4G). Certainly, VoIP services face a direct comparison to existing Public Switched Telephone Network (PSTN) services, which feature a very high availability; known also as the five nines - 99.999%. Under these circumstances high availability of SIP components becomes a key issue for both today's networks and forthcoming wired and wireless ubiquitous realms.

In this paper, two aspects of high availability are presented. The first regards redundancy meaning that more than several SIP servers are able to provide a specific service. In case of failure of the main system, the backup system takes over the service, ideally seamless and transparent to the service user. The other aspect of high availability focuses on Load Balancing (LB) strategies. Within the context of this paper a novel solution is proposed to provide redundancy, failover and LB functionalities among the different subsystems of a SIP-based VoIP service. The proposed solution is evaluated through experimentation and the results show that it is effective, robust and potentially scalable. On top of that, our scheme is lightweight, practical and easy to implement requiring minimal changes to existing SIP software or hardware components. We should mention that the proposed modules are an integral part of our secure high availability SIP-based infrastructure. A complete description and documentation of the security infrastructure is provided in [15].

The rest of the paper is organized as follows: The next section analyzes and evaluates our redundancy and failover schemes for different subsystems of the VoIP architecture. The proposed SIP load balancing scheme is discussed and evaluated in Section 3. Section 4 addresses previous work in the topic, while the last section concludes the paper and provides pointers to future work.

2. Redundancy and Failover

VoIP services are required to be highly available to minimize occurring service outage times. Research has revealed several reasons for failure in telephony systems, which also affect VoIP services [31]. Compared to classical systems, VoIP still suffers from a higher outage probability [24]. One of the main aspects of high availability is redundancy [36], for creating highly available services. A typical approach to achieve higher reliability is to deploy backup systems, which are capable of providing the service in case of failure of a master system. Figure 1 depicts a typical server architecture for high availability by adding redundant components i.e., a redundant RTP media relay, SIP server and the database system. The SIP server and RTP relay tandems use the technology of IP address takeover to realize service takeover as proposed in [19]. Here, each system is attached with its primary network device to the same network, but only the master "owns" the publicly known (shared) IP and MAC address. In case of failure, the backup server assigns both addresses to its own interface and the existing traffic is redirected to it.

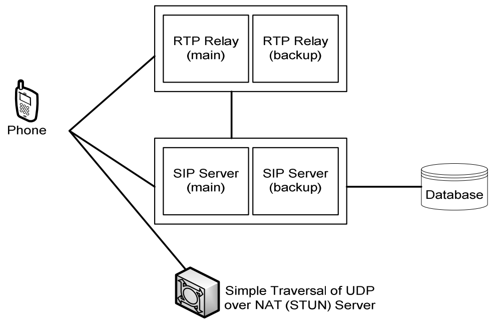

Figure 1. General Architecture for High Availability in SIP infrastructures

2.1 Redundant SIP server

Description of SIP Redundancy Architecture: In order to have a redundant SIP server consisting of two or more different entities, it is necessary that all server instances have the same knowledge about ongoing SIP transactions. To ensure this, it is either possible to directly transfer state changes between each other or to simply replicate the request messages i.e., make all servers get the same messages, but disable all real world communication for the backup server. To achieve this, we propose a High Availability Daemon (HAD), that acts as an additional proxy for each redundant SIP proxy. Each redundant proxy machine is configured with a general purpose SIP proxy and the HAD. The HAD takes care of routing requests to either the primary SIP proxy or the backup proxy. Depending on the state of the system (active or backup) the HAD behaves differently. Specifically, when acting as active instance it will take care of the following: (1) It activates the shared IP address at its primary interface. (2) It receives all SIP requests from the network, replicates them to the backup system, then inserts it own VIA header field and forwards the request to the local SIP Server instance. (3) When receiving a request from the local SIP Server, it routes the request according to the request URI. (4) When receiving any SIP response, it removes its own VIA header field and forwards the message to the next in the VIA stack. (5) It periodically sends heartbeat message to the backup system. (6) It serves as database proxy, forwarding all database requests to the real database cluster. (7) It monitors local SIP Server process(es) if they exist, issuing a "take over" message in case of their failure. It also drops the shared IP address from the primary interface in such a case to avoid the double presence of that address.

On the other hand, when acting as backup instance, it will take care of the following: (1) It deactivates the shared IP address at its primary interface. (2) It deactivates the local proxy interface for database forwarding, thus simulating a down database cluster to the SIP Server. (3) When it receives a replicated request from the active instance, inserts its own VIA header field and forwards the request to the local SIP Server instance. (4) Upon receiving a request from the local SIP Server it drops it. This is not very likely to happen, as the SIP server is meant to answer requests, not to issue them by itself. (5) Upon receiving a SIP response from the local SIP Server it drops it. (6) When it receives a heartbeat it resets the failure timer. If the failure timer hits zero, it changes its role from “backup” to “active”.

The SIP Server e.g., SIP Express Router (SER) [23] is configured to listen only at the local loop interface. It uses the local HAD as outbound proxy to get messages to the network. Figure 2 illustrates a proposed SIP replication architecture based on HAD. Additionally, though not depicted, there exists a kill daemon, which terminates the SIP Server process, in case the HAD process does not exist. This is necessary as the HAD is also a component which is subject to failure. When the HAD fails e.g., due to a bus error, this would result in the heartbeat to fail, which should eventually lead to the backup system becoming active, although the main system is still active. In this case the failure of the SIP Server must be enforced.

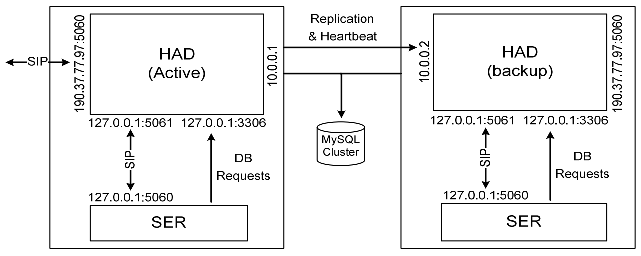

Figure 2. The proposed SIP Replication architecture

For the introduced scenario, it is not necessary to apply any changes to the current version of SER. However, it also may result in a common state establishment taking considerable time if a “naked” server takes over after a failure. To deal with this issue, it would be necessary to add a “burst state table” function to the transaction management of SER.

Testing and Evaluating the SIP Redundancy System: The SIP-servers are configured to act as SIP Registrar and SIP Proxy accordingly. The Registrar saves contact SIP-URIs, while the SIP-Proxy performs contact lookup and forwards INVITE and BYE requests and the equivalent responses. The test setup consists of two Cisco-7905 SIP-Phones and the HA SIP-Server. This setup is a test for functionality evaluation. Basically, the SIP calls (sessions) which are created, need to be present after a service takeover has occurred. This means that e.g., a SIP call, which is at the establishment phase (INVITE sent, waiting for responses) needs to be completed successfully. The corresponding message flow is shown in Figure 3A.

Also, stored contacts need to be present after takeover, so that new calls can still reach the formerly registered users. As an example, a user must be able to initiate a call to another user, who has registered its contact before the crash/takeover. This message flow is depicted in Figure 3B. For the tests, both phones register at the HA SIP-Server. Then, the active SIP-server is halted to enforce a takeover. The outage time, while none of the servers is available, was between five and seven seconds. This time span is configurable in the HAD in terms of heartbeat delay (it has been set to 1500ms) and number of allowed missed heartbeats (it has been set to 4). UACs not receiving responses to their requests during this time re-transmit these requests until the backup server comes into place. The predefined maximum time for

SIP re-transmissions according to RFC-3261 is to be 32s [33]. Therefore, the maximum takeover time must be clearly below this time. Both tests have been performed successfully. The calls were established exactly as shown in Figure 3. In both cases, call termination was also successful.

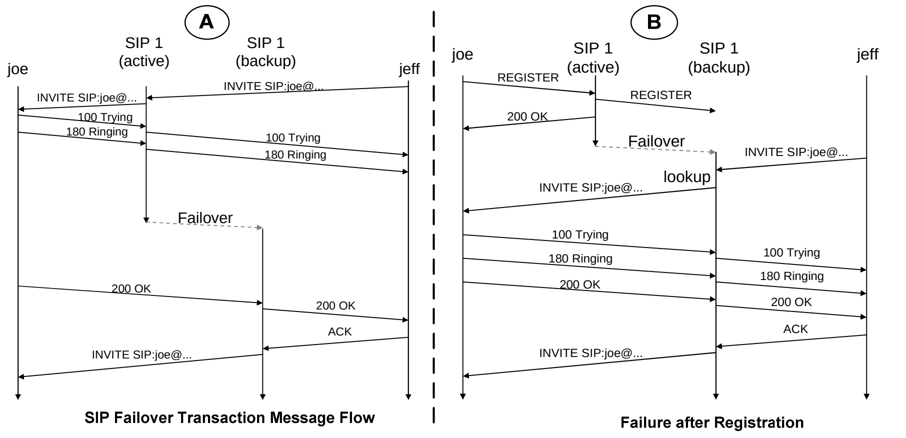

Figure 3. SIP Failover flows

2.2 Redundant RTP Relay

Description of RTP Redundancy Architecture: The requirements for the RTP proxy are similar to that of the SIP proxy. More specifically, the standard RTP proxy assigns two different port numbers for each forwarding relation. Data which is received on one port is forwarded to the other peer through the other port number. The shared knowledge is related to the list of mappings. A single RTP relay has a list of the forwarding relations. An example for a mapping list is depicted in Table 1.

Table 1. A mapping list example

SIP Call ID	Local port #1	Peer 1 IP address:port	Local port #2	Peer 2 IP address:port
47ef8ef6ae0f	12000	130.149.17.5: 40092	12002	81.175.135.177:15023
1f0e5cd783a	12004	81.175.135.177:15023	12006	130.149.17.5: 40092

Upon request by the SIP proxy, the RTP proxy opens the ports and virtually connects them to each other. At that stage it does not know the peer IP addresses and ports, as they are learned when the peers send their first RTP data packet. It is clear that this mapping table must be shared when the RTP relay is going towards redundancy. In case of failure of one relay, the other one must know which peers (i.e., phones) are talking to each other. Thus, the table must be replicated. The previously introduced HAD is also used for replication with the RTP proxy i.e., the RTP media relay. As many functions are common, only an addition for replicating RTP mappings will be introduced. Figure 4 shows the architecture of the redundant RTP relays with the HAD.

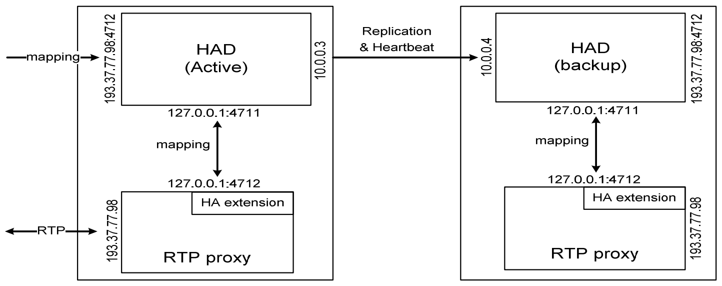

Figure 4. RTP Relay Replication

The HAD receives the mapping requests from the SIP proxy and forwards them to the local RTP proxy. It also intercepts the response from the local RTP proxy and forwards the port it has chosen to the backup system, which is expected to open the same port. The RTP proxy is equipped with a High Availability (HA) extension, which has the following tasks to perform (1) Inform the HAD about mapping completion by RTP reception. As mentioned before, the mappings are completed with peer IP addresses and port numbers when the first RTP packet is received. This information must be replicated by the RTP proxy itself. (2) Provide an option to the requestor for a specific port number. This feature is needed in backup mode to allow port number replication. (3) Inform the HAD about removal of mappings. Mappings are subject to soft state removal in case no explicit removal is performed by the SIP proxy e.g., lost BYE. (3) Perform a re-bind, when the shared IP address is activated. This ensures that the RTP proxy is bound to the correct interface.

As in the SIP proxy scenario, a kill daemon exists with the same task. It terminates the RTP proxy in case of failure of the HAD, causing the heartbeat to fail and the backup system to take over.

Testing and Evaluating the RTP Redundancy System: For a testing scenario, the master emits heartbeat messages with the delay (dh), which is configured by the system administrator. The backup system checks at the same rate, if it has missed heartbeats. The number of heartbeats (nf), which are allowed to be missed without consequences, can also be configured. If the backup system has missed at least one or more heartbeats than allowed, it considers the master to have a failure and initiates the takeover process to become master itself. Thus, in case of a failure, the time of outage can be estimated to be between dh*(nf-1) and dh*(nf+1) seconds, depending on the interleave relation between receiving heartbeats and checking for them. “Early” checking will carry a constant loss of one heartbeat, while “late” checking might oversee a possible missed next heartbeat, which should appear right after the checking.

The testing environment consists of the server side scenario from Figure 5 and a client side employing SIPp [38], which generates calls i.e., acts as SIP UAC and UAS and RTP sender and receiver. The test scenario is as follows: the Call Generator (CG) uses scripts to register users at the CG and to initiate and serve calls. When creating a call, an RTP sender process is spawned to create an 80kbit/s data stream, which corresponds to an 8 KHz PCMU stream. A sniffer is used to register the outage time i.e., the time between the last packet from the HA-RTP-Relay and the first packet, coming from the backup machine in case of failure.

Testing and evaluation has been performed by starting N simultaneous calls. When all calls are established and the RTP proxy serves all N calls, the sniffer is started to record the RTP packets, coming in from the RTP-Proxy. Then the master RTP-Proxy server process gets terminated. The backup server notices the failure, as heartbeats are starting to be missed. It then takes over and starts to forward the RTP-packets, which now hit the backup server, to the call generator. There the sniffer notices the incoming traffic, so that the difference between the last received packet from the master RTP-Proxy and the first received packet from the backup RTP-Proxy give the time for the takeover.

As discussed earlier, this time should be closely related to the configured values for heartbeat delay and the number of allowed misses. For the tests, this has been configured to be 1,500 ms delay and a maximum of 4 missed heartbeats before takeover takes place. This indicates that the time for a full takeover will be between 4,500 ms and 7,500 ms. The test has been performed with N = 20, 50, 100 and 150 calls, to check the influence of the network traffic. Each test has been performed three times. Table 2 illustrates the results from the testing and evaluation session. All observed takeover times are in the margins of the expected range between 4,500 ms and 7,500 ms. It is stressed that reducing the heartbeat time to 500 ms and the number of allowed missed heartbeats to 2 the aforementioned times are expected to span between 500 and 1,500 ms.

The results show that there is no dependency between the number of open calls and the takeover time. This is based on the takeover process itself, as heartbeats do not utilize the same network interface as the audio stream. The backup server just has to create the UDP sockets at takeover time. As there are 4 sockets per call and a standard per process limit of 1024 open file descriptors, the maximum number of calls that the RTP-Proxy can handle is limited to 256. Taking the above results into consideration, it is foreseeable that even 256 calls will be inside the configured time limit.

Table 2. Results from the testing and evaluation session

Calls	First try	Second try	Third try
20	5447.653 ms	6578.933 ms	7124.007 ms
50	5286.816 ms	5141.165 ms	6605.964 ms
100	5589.715 ms	6520.205 ms	5259.155 ms
150	6826.565 ms	5568.898 ms	5795.816 ms

VoIP Database Redundancy: In order to enhance the overall VoIP SIP Based availability it requires also employing a redundant database system and not retaining it as a single point of failure. A variety of solutions for database systems already exist. In the proposed scheme the MySQL-cluster from mysql.com has been employed [29]. It provides a network storage engine which offers the possibility to construct redundancy groups, where the storage engine takes care of any data replication in real time. In our scenario there are two redundancy groups with two storage nodes per group. Thus, two nodes, one in each group, may fail without service interruption. Note however, that an exhaustive analysis of this issue is out of the scope of this paper.

3. SIP load balancing

Any server, regardless of its form factor or application, is limited in the number of concurrent connections and sessions it can handle at any given time. Load balancers can be used to achieve redundancy and improve processing of SIP transactions. The aim here is to increase VoIP service availability, especially when combined with the previously proposed failover techniques. Generally, in LB schemes, new requests are allocated among available servers using a selection algorithm. Large scale corporate VoIP service mandates the deployment of multiple servers in order to serve transactions requested by several VoIP clients concurrently. Multiple installed servers or even clusters of servers aim to smoothly process heavy VoIP traffic so that the service can be sustained unattended without degrading Quality of Service (QoS).

A common selection algorithm targeting on statistical LB is the well known Round-Robin (RR) scheme [8]. Another major category of balancing approaches is weighted or adaptive balancing, which distributes requests proportional to the weight assigned to each available choice or route. Thus, load balancing can be adaptive or not adaptive, depending on whether or not run-time load conditions influence LB decisions. Adaptive LB policies consider real-time system state information based on various metrics e.g., CPU load, available free memory etc, for LB decisions, whereas non-adaptive or static load balancing like RR does not. In any case, to be able to distribute effectively and fairly VoIP traffic to the corresponding redundant servers, the introduction of an appropriate balancing mechanism during the initiation of the call is required. Generally high availability of SIP services is threefold. It concerns signaling, real-time media data, and gateway services. In this section we only consider signaling.

3.1 Load balancing schemes

Certainly the problem of load sharing is not new and goes back many years. As it is discussed in Section 4 a variety of techniques have been considered or applied to cope with the problem. Some of them are ad-hoc and platform specific while some others employ smart schemes to reorder DNS resource records. Two well-known categories of solutions which are mainly utilized for Web server balancing include the following:

Round Robin (RR): Upon a new SIP request, the SIP balancer selects the next IP address record for the specific SIP server alias name as stored in the DNS. This solution is considered non adaptive, due to the fact that it does not require the balancer to maintain and update workload information from the available SIP servers in the domain. For Web balancing this mechanism is built-in into the DNS system by means of DNS SRV [18] and NAPTR [28] mechanisms. One might argue that SIP load balancing can be applied directly to the DNS as it is already supported for other Web services. By doing so however, the DNS will assign one IP address of the SIP server pool to each address resolving request. As a result, the name-to-address mapping will be cached in name servers along the path from the DNS all the way down to the client and consequent address requests reaching the same name servers will be resolved with the same cached addresses. Only after the name-to-address mapping in the cache expires, due to Time-to-Live (TTL) field, it is possible to serve new requests with a new RR decision. This however can be circumvented by setting a low TTL for the A-record bindings. Also one might argue that DNS solutions are good only for scalability in equal weighted servers i.e., it is not possible to dynamically adjust the load on-the-fly. However, the other side of the coin is that DNS may be the preferred way to offer load sharing since it does not mandate servers to reside on the same network. Also replication is a native DNS characteristic; in case a name server goes off-line the system is not affected. Security is also an important parameter here if we assume that DNS transactions are guaranteed by DNS-SEC [14].

Adaptive or Dynamic Weighting: This results in the selection of the server which currently handles the lowest number of SIP transactions or has the minimum overall workload. This category of solutions for Web servers includes Asynchronous Alarms (AAlarm) [12], Ibnamed [34] and TENBIN [35] algorithms for DNS.

In the context of this paper, we employ the following DNS-based balancing approach which has been initially discussed in our previous work [25]: The Load Balancer (LB) module (process or daemon) requests at fixed time intervals the current number of SIP servers (SRV records) available within the DNS server and resolves their Fully Qualified Domain Names (FQDNs). Clearly, this means that for each SIP domain, the DNS server has multiple SRV records corresponding to (redundant) SIP proxies attached to it. In the following example the first four records share a priority of 10, so the weight field's value will be used by clients to determine which server to contact. The big.test.com will be used 60% of the time while next three hosts i.e., medium and small2 will be used for 20% of requests each, with half of the requests that are sent to small2 directed to port 5060 and the remaining half to port 5066. If big.test.com becomes unavailable, these two machines will share the load equally, since they will each be selected 50% of the time. If all four servers with priority 10 are unavailable, backup.example.com having priority 20 will be chosen. The clients, i.e., LB in our case, can use weighted randomization to attain this distribution. After that, the LB assigns the next incoming SIP transaction to the next available SIP proxy according to the aforementioned procedure. Thus, the selection of a certain SIP proxy to serve any initial request is completely transparent to the client.

```
_sip._udp.test.com. 86400 IN SRV 10 60 5060 big.test.com.
_sip._udp.test.com. 86400 IN SRV 10 20 5060 medium.test.com.
_sip._udp.test.com. 86400 IN SRV 10 10 5060 small2.test.com.
_sip._udp.test.com. 86400 IN SRV 10 10 5066 small2.test.com.
_sip._udp.test.com. 86400 IN SRV 20 0 5060 backup.test.com.
```

The testing proxy SER itself does incorporate a dispatcher module that can be used to implement load balancing. However, this method requires all the alternative destination proxies to be included into a text file (loaded at startup), not in DNS as our solution mandates.

3.2 Implementation details

The LB is an add-on entity which is responsible to query DNS and maintain SRV records of all the available SIP proxies in the corresponding domain. For each SIP request issued by a client the LB is responsible for forwarding it to the next SIP proxy available in order to serve it. This is done according to fetched DNS SRV records. SIP clients firstly communicate with the LB entity to discover the next available SIP proxy. If the LB is not responding, the SIP client can communicate directly with the DNS to retrieve all the available SRV records that correspond to SIP servers in the domain. Hereupon, it will select one of them randomly. However until now most SIP clients do not support DNS direct transactions. Thus, another solution for the client is to communicate directly with another available SIP proxy in the same domain. The IP addresses of the LB and the backup SIP proxies can be pre-configured into the SIP client device. As a result, the IP address of the most appropriate SIP proxy is selected by the LB and while the first message (e.g., INVITE) goes through the LB the subsequent messages, for the same session, go directly to the selected by the LB SIP proxy. More specifically, the only SIP message types that need to pass through the LB entity are REGISTER, INVITE, SUBSCRIBE and OPTIONS.

Normally, the LB will be implemented as another standard SIP proxy. Hence, according to the SIP standard, it will usually insert its own VIA header in the incoming SIP message, prior to forwarding it to the corresponding SIP proxy. This procedure has undesired implications as all the subsequent SIP messages referred to the same session will pass again through the LB. To circumvent this problem we propose the following solutions: (1) The first solution enables through proper configuration the home SIP proxies to add a "Record-Route" header field [18]. By doing so, clients would then send follow-up requests within the same session to the home SIP proxy assigned by the LB in the first place. This will then not forwards them to the request-URI (the LB) but process them by itself. In addition, this solution even has the advantage that new calls are load balanced because the route set is valid only for one session. (2) Another alternative is to force the LB not to add a VIA header, hence all subsequent messages go directly to the home SIP proxy in charge. For example, this can be done by utilizing the internal routing engine of the corresponding proxy. In the case of SER, it would be possible to use the SEND command or the Forward function as described in SER's developers' guide [23]. (3) A third alternative is to modify the proxy core source code to force it to ignore the VIA-received header added by the LB. However, this solution is proxy dependent or implementation specific and of course not portable. (3) The final option is to "spoof" the source addresses (IP address and port) of packets (e.g., INVITE messages) which are forwarded by the LB so that proper routing takes place. According to this scenario one can set the IP address and port to the address and port from which the packet arrived to the LB. By employing this solution the LB is more transparent and no changes in the proxy's source code are needed.

Note that for testing and evaluating the LB (see section 3.3) the second one from the previously discussed methods was finally adopted. In our opinion this is the preferred way to cope with the problem since it does not require any modifications to standard SIP proxies, but only to LB. Summarizing, the proposed load balancing solution requires neither modification to existing DNS infrastructure nor to the core of the employed SIP proxy, which acts as a LB. The only actual requirements are: (1) The introduction of the LB independent machine which is being implemented as another typical SER SIP proxy. This server is only required to support DNS-SRV records but this functionality is already mandatory by the SIP standard [33]. Moreover, the employment of a standard SIP server to serve as the LB means that there is no need to develop new software from scratch. Only the "decision-and-forward" engine as a SER module has to be implemented. In case of large corporate SIP networks, including many SIP proxies, we can realize a LB solution consisting of several geographically distributed SIP proxy clusters controlled by equal number of LBs. (2) All SIP home network proxies use either one shared or more (mirrored) databases.

Last but not least, to increase LB availability it is also possible to have one or more backup mirrored LBs to defend against possible DoS or DDoS attacks, physical or human disasters, etc. The most practical solution to this issue enables the backup LB to take over the IP address of the master LB in the case of failure, similarly to failover techniques suggested previously in Section 2.

3.3 Performance evaluation

Our internal test-bed setup and all the corresponding machine and network parameters are illustrated in Figure 5. Alice, who resides in a different sub-network from that of the LB and the corresponding SIP proxies, sends SIP requests (e.g., INVITEs, OPTIONs etc) toward the LB. The LB implements the proposed scheme and balances accordingly the incoming requests to the two available SIP proxies, namely proxy A & B. Requests are always directed to "dummy" users Bob and Mike who respond to the incoming calls. Three machines called "traffic generators" are responsible to generate heavy load SIP traffic in order to stress both the LB and the SIP proxies according to the scenario. The SIP traffic was generated by our own developed attack tool. The maximum, minimum, average and Standard Deviation (SD) ping times between the two sub-networks (actually from user A towards the LB) were 1.140, 0.557, 0.653 and 0.134 ms correspondingly. These times was taken from 70 recorded times with the ping tool.

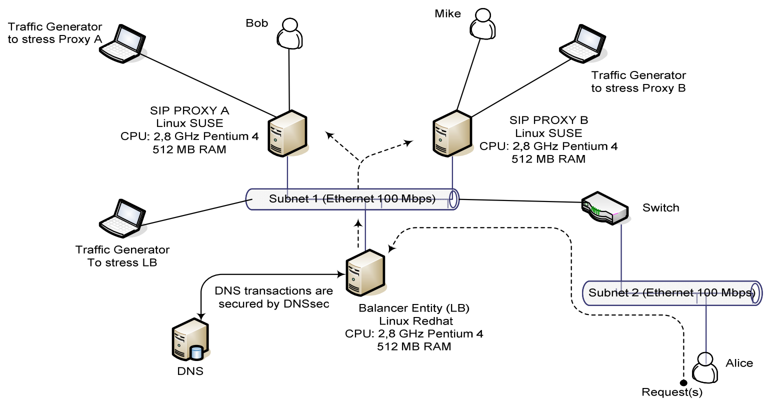
Traffic Generator to stress Proxy A
Bob
Mike
Traffic Generator to stress Proxy B
SIP PROXY A Linux SUSE CPU: 2,8 GHz Pentium 4 512 MB RAM
SIP PROXY B Linux SUSE CPU: 2,8 GHz Pentium 4 512 MB RAM
Subnet 1 (Ethernet 100 Mbps)
Switch
Traffic Generator To stress LB
DNS transactions are secured by DNSsec
Balancer Entity (LB) Linux Redhat CPU: 2,8 GHz Pentium 4 512 MB RAM
Subnet 2 (Ethernet 100 Mbps)
DNS
Alice
Request(s)

Figure 5. Test-bed setup

To evaluate our implementation and determine possible delays introduced by the LB entity two distinct scenarios were implemented:

Scenario I: Every 5 ms the SIP traffic generator responsible to stress LB generates one call. Total duration of this scenario was 40 minutes, thus 480,000 calls were generated in total. All the aforementioned calls pass through the LB. During the 40 minutes time duration Alice generates a new request every 4 sec. All these 600 calls were served by the LB.

Scenario II: During this 40 minutes test both of the SIP proxies were stressed with different background traffic generated by the corresponding traffic generators. More specifically, a new request was sent towards proxy A every 5 ms (a total of 480,000 messages), while for proxy B a new request was sent every 500 ms (a total of 4,800 messages). Simultaneously the LB was accepting a new request by the responsible traffic generator every 5 ms (a total of 480,000 messages). During this time interval Alice generated a new request every 4 sec, i.e., 600 requests in total. Here, 300 calls were served by the LB, while the others went directly to the SIP proxies. This option will show whether or not the LB entity affects significantly the overall roundtrip time to serve a request.

For both scenarios and for all Alice requests we tracked and logged the following metrics: (1) Latency introduced by the LB: namely the maximum, minimum, average and standard deviation (SD) times for the LB to serve an incoming request. This delay time includes the time for the LB to decide (DC_T) to which proxy the message should go (according to RR algorithm), and secondly the actual time needed to forward (FWD_T) the message. (2) Overall serving time: the roundtrip time (RR_T) for a particular request to complete. This is the overall time until the client who generates the request, i.e., user A in our case, receives response from the corresponding proxy (A or B).

The maximum, minimum, average values and standard deviations of the time durations in milliseconds measured for both scenarios are presented in Tables 3 and 4. As we notice, the average LB's total time for dispatching one transaction, that is DC_T + FWD_T, is 0.32 ms for scenario I and about 0.29 ms for scenario II. Consequently, this time duration is negligible, and as expected, almost identical for both scenarios. Also note that this time is totally irrelevant to the number of the domains the request has to traverse. We can only estimate that this time will tend to increase depending on the number of the available proxies. However, considering also the minimal SD value for this metric, the expected overall increment will be in the order of few microseconds, thus still negligible.

On the other hand, the average RR_T time duration is about 3.3205 ms and 3.9405 ms for the two scenarios respectively. Specifically for the second scenario it seems that whether the requests pass through the LB or not, the average RR_T is almost the same (3.9405 vs. 3.9164 ms). Naturally, this observation is confirmed from the insignificant latency that the LB introduces as explained previously. Nevertheless, the more the numbers of network domains the request has to travel, the more the RR_T metric is expected to be. Clearly, the RR_T is affected by the number and the distance of hops existing between the callee and the end-proxy; not by the distance between the LB and the end-proxy, which resides always to the same subnet with the LB. In a nutshell, one can say that all measured times for both scenarios are almost identical. Giving that the latency introduced by the LB is minimal we can argue that the implemented balancing scheme is simple but effective.

Table 3. Results for Scenario I (times in milliseconds)

Number of Transactions →	**480,000**		**600**
Time Description (ms) ↓	DC_T	FWD_T	RR_T
Maximum Time	0.1330	8.9080	51.960
Minimum Time	0.0050	0.2570	0.4150
Average Time	0.0062	0.3138	3.3205
Standard Deviation	0.0018	0.0777	4.5152

Table 4. Results for Scenario II (times in milliseconds)

Number of Transact. →	**480,000**		**300**	**300**
Time Description ↓	DC_T	FWD_T	RR_T (through LB)	RR_T (straight to proxy)
Maximum Time	0.0360	5.0770	38.5940	51.1540
Minimum Time	0.0050	0.2570	1.0520	1.1520
Average Time	0.0061	0.2841	3.9405	3.9164
Standard Deviation	0.0009	0.0300	5.9887	8.5901

4. Related work

Until now various failover and LB methods have been considered and thoroughly tested mainly for Web servers [3,9,11,13,20,31,42,46]. For example, LB via HTTP session based redirection [3,11,46], connection dispatchers [20] and Load Share Network Address Translators (LSNAT) devices [41] are used in Web servers. On the other hand, IP address failover [26], MAC address takeover [11] and TCP connection migration [40] have been considered and investigated for high availability. Another type of client-oriented failover is used by Cisco IP phones [10].

It is to be noted that some of the aforementioned LB and failover schemes can be also profitable for other Internet services like SIP. For instance, the DNS-based load sharing discussed in [18], [27]. However this is not entirely true. For example, TCP and UDP can coexist in SIP proxies, typically call requests and responses do not require extensive bandwidth, caching of responses is needless, NAT fails to address native SIP transactions' characteristics, and so forth [37]. Also, regarding failover, IP anycast will not work in TCP SIP connections and the primary and backup servers need to be synchronized for communication with the backend database [30]. Thus, for real-time communication services like SIP significant research efforts are needed to achieve similar service availability rates we have today for Web services.

Currently many SIP server implementations do not include SIP-specific balancing or failover modules and usually rely on add-on hardware, or peripheral solutions e.g., Web-originated methods. Recently, interests in SIP LB and failover have risen, as some vendors include such modules into their state-of-the-art products. Unfortunately, to the best of our knowledge all but one [44, 45] are proprietary, compound and expensive solutions, targeting to enterprise networks. Moreover, such vendor-dependent LB or failover schemes require special hardware and software modules in order to operate properly. In the following we provide a short review of most important currently offered SIP failover and LB solutions.

Vovida's Load Balancer [45] is the sole open source product. This is an application-layer LB function known as a "stateless load balancing SIP proxy" that can be used in SIP-based VoIP installations in conjunction with multiple identical proxy servers. All users can send their INVITE and REGISTER SIP messages to the same SIP URI and the LB will assign a proxy server dynamically to handle each request. This only works with SIP messages sent over UDP, not TCP. Each request is forwarded to the next available server that appears on a predetermined list of associated servers i.e., according to a "Round Robin" schedule. The LB then receives responses and forwards them back to the requesting party. The Vovida LB adds its own SIP URI address in a "Via" address field in the header of an incoming SIP request packet, before transferring the packet to the assigned server, in order to receive a subsequent response from the server which is then forwarded to the requesting party. However, since the LB does not store data between transactions, it cannot even ensure that requests within a SIP dialog are consistently directed toward the same traffic module. Therefore, all traffic modules must use a shared database for storing the state of any given SIP dialog [32].

All following solutions are proprietary, of high-cost, hardware-oriented and usually combine failover and LB along with other functionalities thus are targeting enterprise networks. The BIG-IP SIP Load Balancing Solution [6] provides high scalability, availability, and reliability to the SIP Proxy, Session Border Controller, media servers and many other SIP devices. This module provides deep packet inspection, so it can distribute and balance SIP and Real-time Transport Protocol (RTP) traffic among multiple SIP devices so that service availability is guaranteed even under high call volumes. In addition, the solution can perform advanced health checks on the SIP devices, routing SIP clients away from unstable or unreliable devices and providing increased reliability to existing SIP solutions.

SIP load balancing of SIP proxy servers in the IBM BladeCenter can be achieved using Nortel Layer 2-7 GbESM switch modules from BLADE Network Technologies [7]. Using this product SIP LB can function with any SIP server that employs shared or clustered databases to share signaling data for Registration and Invites. It performs stateful inspection of SIP messages to scan and hash calls based on a SIP Call-ID header destined for a SIP server. Stateful inspection means that a packet is inspected not only for its source and destination information found in the header, but also packet contents found at the application layer. Once the switch has identified the Call-ID which determines a specific SIP session, it sends future messages from the same Call-ID to the same SIP server.

The Vocalscape Load Balancer began as an open source project which was adopted and improved upon by Vocalscape [43]. It was made compliant with Asterisk Private Branch eXchange (PBX) [4], and the algorithm was revised to distribute calls more evenly. In its previous version, the LB would send calls to a primary server and only when the primary server was overloaded would redirect calls towards additional servers. The updated algorithm balances the load by evenly distributing the calls between the servers. The LB also provides failover capabilities. If a server is not responding, the LB will route all calls to servers that are functional. Some other SIP solutions are provided by IBM [21], A10 Networks [2], and Interactive Intelligence [22].

Literature also lacks significant contribution on SIP failover and load sharing. We are only aware of one recent scientific work [37] that deals with the aforementioned issues. The authors apply existing web server redundancy techniques for high service availability and scalability to the IP telephony context. The paper compares various failover and load sharing methods for registration and call routing servers based on SIP. For failover, the authors choose the DNS-based method, and for load sharing an identifier-based approach. Their schemes are combined in a two-stage reliable and scalable server architecture. Contrarily, we use an IP-takeover based method to provide failover and a DNS-based approach, employing an independent LB machine, to offer load sharing.

5. Conclusions and future work

High-availability solutions for VoIP networks address the need to place and receive calls either under peak-load call rates or during device maintenance or attack incidents or failure. Voice-network downtime results not only in revenue losses for providers but also customer dissatisfaction. This paper has presented redundant and load balancing solutions for critical components of SIP-oriented VoIP infrastructures. It has been proposed that our redundancy solution works properly and efficiently and it is easy to implement in order to increase the overall stability and availability of such systems. It has been demonstrated that even under heavy network load, takeover times follow the configured time frame.

Nevertheless, some issues discussed hereunder are left for future work. Due to the complexity of SIP servers, it would be advisable to integrate the failover function. This would increase efficiency even more, because the SIP server itself could decide, which state information needs to be replicated to ensure a smooth and transparent takeover. Currently, most of the available SIP servers do no incorporate failover functions. In these cases, external state replication is a feasible way to gain high availability. SIP over UDP has a “built-in” feature to overcome packet loss, called “re-transmission”. Thus, as long as a takeover procedure causes just a noticeable loss of packets, clients will behave normally and do not notice a failure. For SIP over TCP this is different, as here the state of the TCP stack needs to be replicated, which would introduce new interfaces to the network core code of the underlying operating system. For the RTP-Proxy part of the failover work, it is advisable to also develop a scalability approach, as the number of simultaneous calls is quite limited. As mapping information is small (about 40 bytes per call), total replication over a scalable set of RTP-Proxies would be a feasible way.

On the other hand, redundant solutions must be seconded by effective load balancing mechanisms, otherwise they are considered only as unilateral. Individual SIP servers have limited scalability. In order to maximize scalability and availability, application servers should be load balanced. Contributing to this subject several implementation issues concerning load sharing were analyzed including architecture, components, interactions etc, showing that the anticipated balancing method is practical and above all easy to implement. Adaptive load balancing is left out for future work. We would also like to expand this study considering clusters of SIP proxies controlled by different LBs.

References

[1] 3rd Generation Partnership Project (3GPP) Consortium, http://www.3gpp.org.

[2] A10 Networks: AX SERIES: SIP Load Balancing, http://www.a10networks.com/products/axseries-sip.php, 2008.

[3] Akamai Technologies, Inc., http://www.akamai.com/html/solutions/index.html.

[4] Asterisk: The Open Source PBX & Telephony Platform, http://www.asterisk.org/.

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

[6] BIG-IP, “SIP Load Balancing Solution”, http://www.f5.com/news-press-events/press/2007/20070212.html.

[7] “SIP Load Balancing in the IBM BladeCenter”, http://www.bladenetwork.net/media/PDFs/WP_VOIP_SIPLoadBalancingIBM.pdf, 2007.

[8] Brisco, T.,”DNS Support for Load Balancing”, RFC 1794, April 1995.

[9] Bryhni, H., Klovning, E., Kure, O., “A comparison of load balancing techniques for scalable web servers”, IEEE Network, Vol. 14, 2000.

[10] Cisco IP phone 7960/7640, Release 2.2, http://www.cisco.com

[11] Cisco Systems, “Failover configuration for LocalDirector”, http://www.cisco.com/warp/public/cc/pd/cxsr/400/tech/index.shtml.

[12] Colajanni, M. Yu, P.S., Dias, D.M., “Scheduling algorithms for distributed Web servers”, in Proc. of the ICDCS '97 17th IEEE International Conference on Distributed Computing Systems, 1997.

[13] Damani, O., Chung, P., Huang, Y., Kintala, C., Wang, Y., “ONE-IP: techniques for hosting a service on a cluster of machines”, Computer Networks, Vol. 29, 1019–1027, 1997.

[14] DNSSEC: DNS Security Extensions Securing the Domain Name System, http://www.dnssec.net/.

[15] Ehlert S., Zhang, G., Geneiatakis, D., Kambourakis, G., Dagiuklas, T, Markl, J., Sisalem, D. "Two Layer Denial of Service Prevention on SIP VoIP Infrastructures", Computer Communications, Vol 31, No. 10, 2008

[16] Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., and Berners-Lee T., Hypertext Transfer Protocol – HTTP/1.1, RFC 2616, June 1999.

[17] Geneiatakis, D., Dagiuklas, T., Kambourakis, G., Ehlert, S., Lambrinoudakis, C., Sisalem, D. and Gritzalis, S., “Survey of Security Vulnerabilities in Session Initiation Protocol”, IEEE Communications Surveys and Tutorials, Vol. 8, No. 3, pp. 68-81, 2006, IEEE Press.

[18] Gulbrandsen, A., Vixie, P. & Esibov, L., “A DNS RR for specifying the location of services (DNS SRV)”, RFC 2782, Feb. 2000.

[19] Hinden, R., “Virtual Router Redundancy Protocol (VRRP)”, RFC 3768, IETF, April 2004.

[20] Hunt, G., Goldszmidt, G., King, R.P., Mukherjee, R., “Network dispatcher: a connection router for scalable Internet services”, Computer Networks 30(1998) 347–357.

[21] IBM Workplace Collaboration Services, version 2.5.1, http://publib.boulder.ibm.com/infocenter/iwphelp/v2r5m1/index.jsp?topic=/com.ibm.wcs.ic.doc_2.5.1/install/i_inst_t_nd_sip_balance.html, 2006.

[22] Interactive Intelligence, “Interaction SIP Proxy”, http://www.inin.com/ ProductSolutions/Documents/SIP-Proxy-Product-Snapshot.pdf

[23] Janak, J., Kuthan, J., Iancu, B., “SIP Express Router v0.9.x”, Developer’s Guide, http://www.iptel.org.

[24] Jiang, W., Schulzrinne, H., “Assessment of VoIP Service Availability in the current Internet”, Passive & Active Measurement Workshop, San Diego, CA, April 2003.

[25] Kambourakis, G., Geneiatakis, D., Dagiouklas, T., Lambrinoudakis, C. and Gritzalis, S., “Towards Effective SIP load balancing”, Proceedings of the 3rd Annual VoIP Security Workshop, June 2006, Berlin, Germany, ACM press.

[26] The High-Availability Linux Project, http://www.linux-ha.org/.

[27] Mealling, M., Daniel, R.W., “The naming authority pointer (NAPTR)”, DNS resource record, RFC 2915, Internet Engineering Task Force 2000.

[28] Mealling, M., Daniel, R.W., “The naming authority pointer (NAPTR) DNS resource record”, RFC 2915, Internet Engineering Task Force, 2000.

[29] MySQL, Open Source SQL server, http://www.mysql.com/.

[30] Ohlmeier, N., “Design and implementation of a high availability SIP server architecture”, Thesis, Computer Science Department, Technical University of Berlin, Berlin, Germany, 2003.

[31] Oppenheimer, D., Ganapathi, A., Patterson, D., “Why do Internet services fail, and what can be done about it?”, Proc. of 4th USENIX Symposium on Internet Technologies and Systems (USITS’03), Seattle, WA, 2003.

[32] Palmeter, M., Danne, A, “A Method and apparatus for Distributing Load on Application Servers”, WO/2006/107249, http://www.wipo.int/pctdb/en/wo.jsp?IA= SE2006000356&DISPLAY=STATUS, 2006.

[33] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E., “SIP: Session Initiation Protocol”, RFC 3261, June 2002.

[34] Schemers, R.J., “Ibnamed: A load balancing name server in Perl”, in LISA ’95 conference, Stanford University, Sept. 1995.

[35] Shimokawa, T., Yoshida, N. & Ushijima, K. “DNS-based Mechanism for Policy-added Server Selection”, http://www.is.kyusanu.ac.jp/~toshi/publications/ssgrr2000.pdf

[36] Singh K., Schulzrinne, H., "Failover and Load Sharing in SIP Telephony”, Technical Report, Dept. of Computer Science, Columbia University, March 2004.

[37] Singh, K. Schulzrinne, H., “Failover, load sharing and server architecture in SIP telephony”, Computer Communications 30 (2007) 927–942, 2007.

[38] SIPp reference documentation, http://sipp.sourceforge.net/doc/reference.html.

[39] Sisalem, D. Kuthan, J. Ehlert, S., “Denial of service attacks targeting a SIP VoIP infrastructure: attack scenarios and prevention mechanisms”, IEEE Network, vol.20, no.5, pp. 26- 31, Sept.-Oct. 2006.

[40] Snoeren, A.C., Andersen, D., Balakrishnan, H., “Fine-grained failover using connection migration”, in: USENIX Symposium on Internet Technologies and Systems, San Francisco, 2001.

[41] Srisuresh, P., Gan, D., “Load sharing using IP network address translation (LSNAT)”, RFC 2391, Internet Engineering Task Force 1998.

[42] Suryanarayanan, K., Christensen, K.J., “Performance evaluation of new methods of automatic redirection for load balancing of apache servers distributed in the Internet”, Proc. of the IEEE Conference on Local Computer Networks, Tampa, Florida, USA, 2000.

[43] Vocalscape Load Balancer, http://www.vocalscape.com/products.htm.

[44] Open Source VOIP Software, Voip-Info.org, http://www.voip-info.org/wiki/view/Vovida.org+load+balancer.

[45] Vovida.org, Load Balancer http://www.vovida.org/applications/downloads/loadbalancer/.

[46] Yang, C.-L., Luo, M.-Y., “Efficient support for content-based routing in web server clusters”, Proc. of the 2nd USENIX Symposium on Internet Technologies and Systems, Boulder, Colorado, USA, 1999.

Authors

Dr. Georgios Kambourakis (www.icsd.aegean.gr/gkamb) received the Diploma in Applied Informatics from the Athens University of Economics and Business (AUEB), and the Ph.D. in information and communication systems engineering from the department of Information and Communications Systems Engineering of the University of Aegean (UoA). He also holds a M.Ed. from the Hellenic Open University. Currently Dr. Kambourakis is a Lecturer at the Department of Information and Communication Systems Engineering of the University of the Aegean, Greece. His research interests are in the fields of Mobile and Wireless networks security, VoIP security, security protocols, Public Key Infrastructure and mLearning and he has more than 55 publications in the above areas. He has been involved in several national and EU funded R&D projects in the areas of Information and Communication Systems Security. He is a reviewer of several IEEE and other international journals and has served as a technical program committee member in numerous conferences. Dr. Kambourakis is a member of the Greek Computer Society.

Dr Dimitris Geneiatakis received a five-year Diploma in Information and Communication Systems Engineering in 2003, and a M.Sc. in Security of Information and Communication Systems in 2005, and a Ph.D. in the field of Information and Communication Systems Security from the Department of Information and Communications Systems Engineering of the University of Aegean, Greece. He has participated in various national and international projects in the area of Information Systems Security. His current research interests are in the areas of security mechanisms in Internet Telephony, Smart Cards, Intrusion Detection Systems and Network Security. He is an author of more that twenty refereed papers in international scientific journals and conference proceedings. Furthermore, he has served as program and organizing committees on several international conferences on Informatics and is a reviewer in various well-known scientific journals. Currently, he is within InCrypto Ltd (www.incrypto.com) as a Security Engineer in Unified Communications. He is a member of the Technical Chamber of Greece since 2004.

Assistant Professor Costas LAMBRINOUDAKIS (B.Sc, M.Sc, Ph.D) was born in Greece in 1963. He holds a B.Sc. (Electrical and Electronic Engineering) degree from the University of Salford (UK), an M.Sc. (Control Systems) and a Ph.D. (Computer Science) degree form the University of London (UK). Currently he is an Assistant Professor at the Department of Digital Systems, University of Piraeus, Greece. From 1998 until 2009 he has held teaching position with the University of the Aegean, Department of Information and Communication Systems Engineering, Greece. He has been involved in several national and EU funded R&D projects in the areas of Information and

Communication Systems Security. These research programs include SERENITY, e-SENSE, SNOCER (FP6 SME-1), e-VOTE (IST), HERMES (Telematics), GESTALT (ACTS), VSAT Network for Telematics and Health Care (SfS NATO), VITAL-Home (ISIS), IRIS (IST), etc. His published scientific work includes six books on Information and Communication Technologies topics, and more than sixty journal and National and International Conference papers. The focus of these publications is on Information and Communication Systems Security and Privacy Enhancing Technologies. He has served on program and organizing committees of National and International conferences on Informatics and is a reviewer for several scientific journals. He is a member of the ACM and the IEEE.

Prof. Stefanos Gritzalis holds a BSc in Physics, an MSc in Electronic Automation, and a PhD in Information and Communications Security from the Dept. of Informatics and Telecommunications, University of Athens, Greece. Currently he is the Deputy Head of the Department of Information and Communication Systems Engineering, University of the Aegean, Greece and the Director of the Laboratory of Information and Communication Systems Security (Info-Sec-Lab). He has been involved in several national and EU funded R&D projects. His published scientific work includes 30 books or book chapters and more than 190 journal and international refereed conference and workshop papers. The focus of these publications is on Information and Communications Security and Privacy. His most highly cited papers have more than 650 citations (h-index=15). He has acted as Guest Editor in 16 journal special issues, and has leaded more than 25 international conferences and workshops as General Chair or Program Commitee Chair. He has served on more than 170 Program Committees of international conferences and workshops. He is an Editor-in-Chief or Editor or Editorial Board member for 12 journals and a Reviewer for more than 35 journals. He has supervised 8 PhD dissertations. He was an elected Member of the Board (Secretary General, Treasurer) of the Greek Computer Society. His professional experience includes senior consulting and researcher positions in a number of private and public institutions. He is a Member of the ACM, the IEEE, and the IEEE Communications Society "Communications and Information Security Technical Committee".

Tasos Dagiuklas was born in Patras, Greece. He received the Engineering Degree from the University of Patras-Greece in 1989, the M.Sc. from the University of Manchester-UK in 1991 and the Ph.D. from the University of Essex-UK in 1995, all in Electrical Engineering. Currently, he is employed as Assistant Professor at the Department. of Telecommunications Systems and Networks, Technological Educational Institute (TEI) of Mesolonghi, Greece. He is also Senior Research Associate within the Wireless Telecommunications Laboratory of the Electrical and Computer Engineering Department at the University of Patras, Greece. Past Positions include teaching Staff at the University of Aegean, Department of Information and Communications Systems Engineering, Greece, senior posts at INTRACOM and OTE, Greece. He has been involved in several EC R&D Research Projects under FP5, FP6 and FP7 research frameworks, in the fields of All-IP network and next generation services. Currently, he is the Technical Manager of the FP7-ICT-PEACE project.

He was the Conference General Chair of the international conference, Mobile Multimedia 2007 (ACM Mobimedia 2007), Technical Co-Chair of MMNS Conference of MANWEEK 2008, IMS Workshop Chair as part of ACM Mobimedia 2008 and Workshop Chair for ACM Mobimedia 2009. He has served as TPC member to more than 15 international conferences. His research interests include All-IP Networks, systems beyond 3G and converged multimedia services over fixed-mobile networks. Dr Dagiuklas ha published more than 80 papers at international journals, conferences and standardisation fora in the above fields. He is a member of IEEE and Technical Chamber of Greece.

Sven Ehlert is the head of the security reserach staff of the "Next Generation Network Integration" divison of the Fraunhofer Insitute FOKUS in Berlin, Germany. He has lead two international research projects in the field of SIP security and has published several refereed scientific papers in the security field. Sven Ehlert received his M.Sc in Computer Science from the Technische Universität Berlin.

Jens Fiedler finished his diploma in computer science in October 2004 at the Technical University of Berlin (TUB). Since May 2005 he works as a researcher at the Fraunhofer institute for open communications systems - FOKUS in the competence center for next generation network infrastructures - NGNI. His expertise includes knowledge in several programming languages, e.g. C/C++, Java. His core competences are VoIP Infrastructures, High Availability, Reliability and Scalability in VoIP Infrastructures, Peer-to-peer technologies, P2P integration and general network protocols. He worked in projects like 6net (EU), SNOCER (EU) and VoIP-Defender (FOKUS). His is currentlyinvolved in developing P2P strategies for IMS in the scope of the FP7-Project VITAL++..