DISSERTATION

Denial-of-Service and Unsolicited Communication Protection in Global Voice-over-IP Infrastructures

ausgeführt am
Institute of Telecommunications
der Technischen Universität Wien

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von
Em.O.Univ.Prof. Dipl.-Ing. Dr.techn. Harmen R. van As
und
PD Assoc.Prof. Dipl.-Ing. Dr. Peter Schartner

ausgeführt durch
Mag.rer.soc.oec. Dipl.-Ing. Michael Hirschbichler

Wien, September 2013

Danksagung

Mein Dank gilt o.Univ.Prof. Dr. Harmen R. van As für die Betreuung der Arbeit und die vielen hilfreichen Diskussionen und die zahllosen Verbesserungen. Ich bedanke mich bei Dr. Joachim Fabini für die jahrelange Unterstützung meiner Forschungstätigkeit und das Korrekturlesen dieser Arbeit. Prof. Dr. Peter Schartner danke ich für das Begutachten der Arbeit.

Ich möchte mich bei der A1 Telekom Austria und bei iptel GmbH (im speziellen Dr. Jiri Kuthan) für die Bereitstellung der Daten bedanken, welche ein wichtiges Kernstück dieser Arbeit sind.

Ebenso möchte ich mich bei allen Arbeits- und Projektkollegen bedanken, ohne die vieles nicht möglich gewesen wäre und ohne die die Arbeit nur halb so viel Freude bereitet hätte. Hier möchte ich besonders Bernhard Seifert, Christoph Egger, Marco Happenhofer, Helmut Hofstetter (A1 Telekom Austria), Erich Hochstöger (A1 Telekom Austria) und Andreas Berger (Forschungszentrum Telekommunikation Wien) hervorheben.

Ich möchte mich bei allen meinen Freunden bedanken, die mir durch Unternehmungen abseits des Arbeits- und Studienalltages halfen, einen Ausgleich und Auflockerung zu finden.

Meiner Familie ein großes Dankeschön für die große Unterstützung im Laufe meines gesamten Studiums.

Der größte Dank gilt meiner Frau Brigitte für die große Geduld und ohne die diese Arbeit nicht möglich gewesen wäre.

Abstract

The switch of future telecommunication infrastructures to packet switched communication using the Internet Protocol (IP) raises new issues and challenges regarding security. Telecom operators will likely face challenges similar to the ones encountered in the last years by operators of established IP services like World Wide Web (WWW) or Email.

The first part of this thesis presents attack scenarios using theoretical discussions and practical examples. These examples demonstrate, how successful DoS attacks against VoIP-infrastructures can be done with little effort.

The second part of this thesis proposes detection mechanisms for service disruptions using active and passive monitoring. State of the art standards by Internet Engineering Task Force (IETF) and 3rd Generation Partnership Project (3GPP) as well as own extensions for precise congestion rating are presented in this section.

The last section of this thesis presents a novel perimeter protection node (Session Border Controller - Advanced (SBC-A)) and validates the effectivity of this infrastructure using reference measurements from operative public IP networks. This infrastructure applies various techniques in fifteen consecutive stages to block malicious messages and to flag problematic ones. The latter are forwarded for further processing to the central VoIP infrastructure. In case of an unavoidable congestion, the markings can be used to drop requests with a high marking earlier than important messages. The effectiveness of this approach is validated using real traffic from A1 Telekom Austria and the VoIP-provider iptel.org.

Zusammenfassung

Die Umstellung zukünftiger Telekommunikationsnetze auf Paketvermittlung unter der Verwendung des Internet Protokolls (IP) wirft viele Fragen und Probleme bezüglich Störungssicherheit auf. So laufen die Betreiber Gefahr, ähnliche Herausforderungen lösen zu müssen, wie in den vergangenen Jahren die Betreiber etablierter IP Dienstleistungen wie World Wide Web (WWW) oder Email.

In dieser Arbeit wird speziell auf Probleme durch unerwünschte Kommunikation, Überlastung und Angriffe eingegangen.

Die vorliegende Arbeit stellt Herausforderungen und Gefährdungen bei Voice-over-IP (VoIP) Systemen vor und behandelt insbesondere Denial-of-Service (DoS) Angriffe und das massive Auftreten von Werbenachrichten ähnlich dem bereits bekannten Email-SPAM.

Im ersten Teil der Arbeit werden Angriffsszenarien anhand von theoretischen Betrachtungen sowie durch konkrete Beispiele vorgestellt. Die angewandten Beispiele zeigen, wie mit geringem Aufwand erfolgreich DoS-Angriffe gegen VoIP-Systeme gefahren werden können.

Aufbauend auf die Erkenntnisse des ersten Teils beschreibt der zweite Teil die Detektion von Störungen unter der Nutzung aktiver und passiver Überwachung. Diese Lösungsansätze der Normierungsorganisationen *Internet Engineering Task Force* (IETF) und *3rd Generation Partnership Project* (3GPP) werden durch eigene Entwicklungen, durch Anpassung bereits etablierter Technologien und Schlussfolgerungen aus Messungen zu einer umfangreichen Schutzinfrastruktur zusammengesetzt.

Im letzten Teil der Arbeit wird Wirksamkeit dieser Schutzinfrastruktur durch Referenzmessungen in realen, operativen Netzwerken validiert und deren Wirksamkeit nachgewiesen. Dieses im Zuge dieser Arbeit entwickelte Analysesystem wendet in fünfzehn Schritten verschiedene Methoden an, um fehlerhafte Nachrichten zu blockieren und potentiell gefährdende Nachrichten zu markieren. Letztere werden in weiterer Folge priorisiert an die zentrale Infrastruktur zur weiteren Verarbeitung weitergeleitet. Im Falle einer unvermeidbaren Überlast sind die überlasteten Systeme anhand der Markierung in der

Lage, problematische oder für den Betreiber weniger wichtige Nachrichten zu verwerfen, bevor sich die Überlast auf die wesentlichen Aufgaben der Systeme auswirken Zur Verifikation dieses Konzeptes werden Verkehrsdaten der A1 Telekom Austria sowie des VoIP-Providers iptel.org durch dieses mehrstufige Analysesystem ausgewertet und die Wirksamkeit des Konzeptes bewiesen.

Contents

Chapter 1

Introduction

Latest with the wide deployment of Long Term Evolution (LTE)-mobile networks and the move from Circuit Switched (CS) technology to Packet Switched (PS) only networks, the signaling of multimedia sessions must be redefined. Selected by the 3GPP the Session Initiation Protocol (SIP) [Rosenberg02] will be the key signaling and control protocol for their IP Multimedia Subsystem (IMS) (rel. 5) [3GPP10b].
In parallel, the introduction of PS technology accelerates the deployment of new services, like machine-to-machine communication, presence services or unified communication. With increasing usage of this technique and due to new ideas and concepts, the SIP traffic will increase considerably.

The introduction of (IETF)-standard Internet protocols like SIP, the underlying *Internet Protocol (IP)* as well as *Transmission Control Protocol (TCP)* and *User Datagram Protocol (UDP)* in combination with interfaces to the Internet will open the former encapsulated telecommunication providers infrastructures to attacks and problems well known from the Internet domain.

Hosting of SIP User Agents (UAs) on top of complex fixed and mobile devices or personal computers (PCs) might be entry gates for malicious attackers. The not-yet solved updating mechanism of mobile devices like Android- or iOS-operated smart phones will result in a high number of outdated and unpatched devices. An example is presented in [Könings] and the according discussion can be found in [InfoSecurity]. As a consequence, as long as the security patching mechanisms in mobile devices are problematic, these devices must be understood as *untrusted*.

The hijacking of huge numbers of mobile devices and the modification of them to parts of botnets might create a new, currently unknown, attack scenario.

In addition, unsolicited communication like well-known from Email com-

munication can be another harassment for the multimedia sessions. The risk of getting voice SPAM ("SPIT") originated by botnets during day and nighttime will probably increase as the distribution will not create any costs for the spammer. Instead, the owner of the hijacked communication devices will confronted with the connection fees. The risk of hidden costs as well as the risk of getting calls the whole day long can be seen as a big acceptance problem for these future deployments. This increasing traffic together with the threat of botnet driven Denial-of-Service (DoS) attacks and SPIT messages asks for a holistic approach for protecting the system-critical telecommunication core networks.

1.1 Motivation

The lack of adequate congestion control mechanisms [Hilt08] in SIP is one of the major risks in future IP-enabled telecommunication solutions. Both, the IETF and the European Telecommunications Standards Institute (ETSI) are working on standards to extend the SIP protocol for overload and congestion control. While the IETF published the informational RFC 6357 [Hilt11] about "Design Considerations for SIP Overload control", the specific SIP extensions are under draft since more than five years. The "SIP Overload Control"-draft [Gurbani13] implements a hop-by-hop overload control. This thesis will demonstrate, that the complexity of large scale IMS deployments cannot be effectively protected using this future standard.

The ETSI TISPAN standard "NGN Congestion and Overload Control" [ETSI10] proposes an extensive standard for overload communication within a dedicated signaling path. The affected core and perimeter nodes are combined together for overload protection communication. This combination is explicitly done by the operator of the infrastructure and creates a close coupling for overload control between the nodes. This benefit is also its main drawback: The complexity reduces its acceptance within vendors, operators and administrative staff. The additional communication path increases load on each node and creates additional risks on congestion.

As the IMS is driven by Internet Standards like SIP and Diameter (RFC 6733 [Fajardo12]), providers are interested to combine SIP enabled IMS components with off-the-shelf Internet VoIP solutions. The latter ones will probably not support the "NGN Congestion and Overload Control" and create a breach inside the operator's overload control concept. As long as there are no common accepted and implemented standards, the risk of congestion will increase with time and with any new deployed complex VoIP solution.

To solve this challenge, this thesis proposes a new component denoted as "SBC-A", which protects SIP-based VoIP systems of various complexity level

already at the border node. It extends the Session Border Controller (SBC) concept presented in RFC 5853 "Requirements from SIP Session Border Controllers" [Hautakorpi10] and adds marking and prioritization of inspected SIP traffic. This system supports vital traffic, which decreases system load/congestion level and reduces traffic which increases system load.

1.2 Main Contribution

The main contribution of this work is divided into four parts: the first part is the development of an implicit load monitoring environment for VoIP infrastructures named *Performance, Availability and Reliability Information System (PARIS)* [Hirschbichler11]. This toolkit considers the system-under-test (SUT) as a black box and creates continuous calls against this infrastructure. The resulting SIP signaling values *"Successful Registration - Session Request Delay (SR SRD)"* and *"Successful Session Setup - Session Request Delay (SSS SRD)"* (defined in RFC 6076 [Malas11]) and the resulting Realtime Transfer Protocol (RTP) Key Performance Indicators (KPIs) *"Jitter"*, *Packet loss"*, *"Latency"* and *"Mean Opinion Score (MOS)"* are taken as input values for eventual alerting and traffic shaping.

The second contribution of this thesis is the introduction and definition of a numeric score to quantify the importance of a SIP request. This header, denoted as `X-dropability`, adds a parameter to estimate the value of a SIP request for the system. The score is low, if this request is of high importance for the provider to operate the infrastructure. The higher the `X-dropability` header score is, the lower is the importance of the monitored request. The resulting value can be used as a decision guidance for congestion control systems in subsequent processing stages. Here, these system can decide which requests to drop and which requests are important to be processed. Compared to uniform dropping, the `X-dropability`-indicator helps to drop the low-priority messages first. This header is fundamental for operating of the fourth contribution.

The third contribution presents an approach to detect SPIT-transmitting nodes by reusing DNS-blacklists originally introduced for SPAM-fighting E-mail-servers. Published in [Hirschbichler09], the results of this approach are additional input values for the fourth contribution.

The fourth contribution introduces the "Session Border Controller - Advanced". This SBC-A combines fifteen steps of DoS- and SPIT-defending mechanisms with SIP syntax checks to block invalid requests, respectively to rate the priority of the request using the `X-dropability`-SIP-header. This overload protection approach has been published in [Hirschbichler12]. The effectiveness of the proposed SBC-A approach is furthermore verified using live

traffic from two distinct VoIP- and IMS-providers.

1.3 Related Work

A definition of a Denial-of-Service attack in general was first published in 1984 in [Gligor84], whereas the increasing problem of Distributed DoS attacks against computer systems and networks is discussed in [Geng00].

Voice-over-IP infrastructures like discussed in this thesis are basing on RFC 3261 [Rosenberg02]. This RFC standard proposes a On-Off congestion control concept. The authors of [Noel09] show the ineffectiveness of this proposed concept by using simulations and presents several other algorithms to control SIP congestion. The informational IETF RFC 5390 [Rosenberg08a] specifies requirements for management of overload in SIP, whereas RFC 6357 [Hilt11] presents models and design considerations for overload and congestion mechanisms in SIP. The IETF draft [Gurbani13] proposes specific SIP header for servers involved in overload control.

The ETSI has defined an architecture for overload communication and control using a dedicated signaling path in [ETSI10]. This standard defines the *Network Overload Control Architecture* and the *Generic Overload Control Application Protocol* for signaling congestion within a Next Generation Network (NGN) infrastructure.

The threat of receiving SPIT as an evolutionary step of the Email-SPAM threat is discussed by Srivastava and Schulzrinne in [Srivastava05]. RFC 5039 [Rosenberg08b] and [Quinten07] summarize the multiple proposals published in the last years and combine them to a guideline for future solutions. The quintessence of these two publications is, that only a combination of several methods together with strong authentication will solve the SPIT thread. [Schlegel06] and [Quittek08b] present a prevention framework to combine several of the proposed methods. These publications propose to forward the resulting SPIT score to the proxy or to the user. The IETF draft [Wing08] introduces a specific header parameter in SIP to transport this proposed score.

Not only the SPIT score, but also the message prioritization basing on RFC 4412 [Schulzrinne06] must be considered when fighting DoS attacks: [Gurbani13] proposes to honor local policies for message prioritization as also the headers introduced in RFC 4412.

1.4 Structure of this Work

This thesis is organized in two main parts: the first part (chapter 3 to 5) presents challenges in complex VoIP infrastructures with respect to congestion and illustrates the difficulties on handling congestion control. Chapter 5

summarizes attack mechanisms suited to decrease live VoIP-systems availability.

The second part presents solution proposals. Chapter 6 outlines implicit overload detection using the self-developed PARIS infrastructure. Chapter 7 discusses overload communication and the standard approaches, while chapter 8 presents anti-SPIT concepts. Chapter 9 introduces a DNS-blacklist approach to detect SPIT-transmitting nodes.

The concepts presented in the second part are combined in chapter 10 to a sophisticated perimeter security component, the SBC-A. The single steps are discussed in this section and are validated using real traffic logs of two SIP and IMS providers. This chapter is the main contribution and demonstrates, that even with limited information about the complex protected system, a sophisticated and effective protection node can be introduced.

Chapter 11 summarizes the results and gives an outlook on future developments.

Chapter 2

Complex VoIP infrastructures

The widespread introduction of IP-based telecommunications in business and residential fields raises a new field for possible attackers, which probably will lead to an increasing amount of assaults against the telecommunication infrastructure. Latest with the complete switch from Circuit Switched (CS)- to Packet Switched (PS)-based communication in mobile and fixed line global telecommunication, DoS attacks will hurt vital infrastructures.
This chapter introduces SIP [Rosenberg02] and points out the possible attack vectors when using this protocol in a large scale communication infrastructure. One of these large scale infrastructures is the future All-IP telecommunication core IMS [3GPP10b], where SIP is the core signaling protocol. The architecture of IMS will be discussed in the following section.

2.1 The Session Initiation Protocol (SIP)

SIP is a protocol to set up and control communication sessions between one or more user agents (UA). It is a text based protocol initially developed by the IETF in Request for Comment (RFC) 2543 [Handley99](1999) and displaced by RFC 3261 [Rosenberg02] in 2002. In contrast to other IP based telecommunication protocols like ITU-T H.323 [ITU-T09] which is tightly coupled to the Integrated Services Digital Network (ISDN)-protocol, SIP is building on the design of the Hypertext Transfer Protocol (HTTP) (RFC 2616 [Fielding99]) and implements a request and response transaction model similar to HTTP.

2.1.1 Organization of SIP

SIP is an application layer protocol designed to create, control and modify media sessions. Additionally, it handles presence state changes, registration, and text-based communication. The protocol does not transport media or

media-information, it is responsible for signaling the call setup and tear down. The SIP protocol requires other protocols like RTP and Session Description Protocol (SDP) [Handley06] in order to create a multimedia session. When initiating a media session, SIP typically transports SDP-messages for media control as payload. SIP uses by default UDP, but is also capable to use TCP and Stream Control Transmission Protocol (SCTP) as underlaying protocol layer.

2.1.2 Building Blocks

Understanding SIP and the underlying concept needs a definition on the components. Figure 2.1 depicts these fundamental elements.

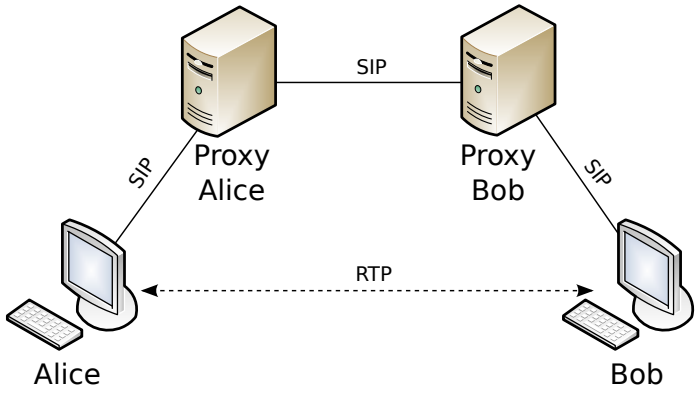


Figure 2.1: SIP trapezoid

- The minimum required components are the *User Agents (UA)*: a UA is a SIP client either implemented as a hard-phone or as an application running on a PC or smartphone, a so called "soft client". To differ between the roles of the caller and the callee, these components are denoted as User Agent Client (UAC) and User Agent Server (UAS).
- The *proxy server* is the routing component. It receives the requests from the UAC and forwards these to the next hop – either another proxy or the UAS. Often collocated with the proxy is the
- *Registrar*: the registrar is acting as a location server. It accepts register-requests and stores the UA's transport address in its location database. Incoming SIP requests targeted for a UAS are then forwarded to the UAS's location stored in the registrars location database.
- *SBC*: the SBC is not initially defined in standard documents but was established by industry to deploy a border node between the public network and the core network. The ex post RFC 5853 [Hautakorpi10] de-

fines the basic requirements a SBC should meet. The important protecting functionality of SBCs is extensively discussed in chapter 10.

- *(Media) Gateway* are signaling as media interface between the SIP infrastructure and other networks. Usually, the Media Gateway (MGW) is used to interact between a PS SIP infrastructure and the CS domain.
- *Application Server (AS)*: The AS is a proxy server with additional functionalities like conferencing systems, video servers or presence servers.

SIP infrastructure is scalable to large and very large systems. A small SIP provider usually operates a SIP proxy collocated with a registrar and a separate SBC. If the operator offers value added services like voice mail or fax-to-mail gateways, additional AS are in use. If the network shall be connected with the PS, the media gateway will also be part of the infrastructure.

2.1.3 Transactions

To reliable deliver SIP messages over connectionless protocols like UDP, SIP uses transactions: Transactions save an internal state in both client and server and assure, that lost messages can be recovered. This challenge of lost packets is solved by retransmission timers: if a SIP request is not replied by a SIP response, various timers are fired, depending on the missing message type.

RFC 3261 defines two categories of SIP transaction, which are handled differently: INVITE and non-INVITE transactions. *INVITE transactions* define the transaction initiating a multimedia call and *non-INVITE transaction* describe all other transactions. The most significant timers in SIP are presented in Table 2.1.

Timer	Timeout	Description
T1	500 ms	the estimated Roundtrip Time (RTT)
T2	4 s	the maximum retransmit interval for non-INVITE requests
Timer A	initially T1	INVITE retransmit interval for UDP only. After each retransmission, the interval between is doubled until the interval-value T2 is reached.
Timer B	64*T1	INVITE transaction retransmit timeout.
Timer E	initially T1	non-INVITE retransmit interval for UDP. Handled similar to Timer A.
Timer F	64*T1	on-INVITE transaction retransmit timeout.

Table 2.1: Important timer values (refer to RFC 3261 Annex A [Rosenberg02])

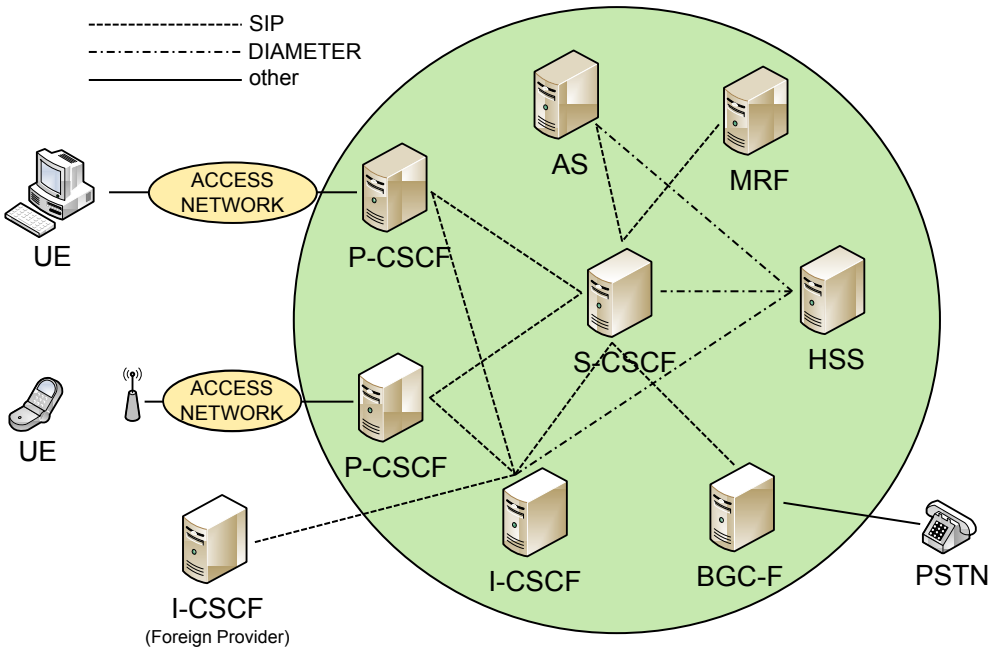


Figure 2.2: IMS architecture (signaling)

The retransmission timers can generate massive problems in congested components as the clients retransmit unanswered requests and therefore cause additional load on potentially already congested components. Although RFC 3261 states in 17.1.1.2, that T1 is just a default value and might be increased in high-latency networks, all observed clients (even in mobile networks) use this timer. A discussion about the high amount of retransmission triggered by this behavior is discussed in chapter 10.

2.2 IP Multimedia Subsystem (IMS)

This section gives an overview on the extensive IMS standard. For further reading, IMS is presented in detail in [Hirschbichler06] and [cam04].

IMS is an architecture and framework for future all-IP networks. Its goal is to offer services and standardized interfaces independent on the underlying access network technology. With the use of media gateways, the concept also supports access to and from other networks like the Public Switched Telephone Networks (PSTN) or ISDN. The main signaling protocol in IMS is SIP, but IMS also uses Diameter [Fajardo12] for authentication, authorization and accounting (AAA), H.248 [ITU-T05] (for media gateways) and XCAP [Rosenberg07] for presence services.

2.2.1 IMS Core Network

3GPP defines in the IMS standard several components to interact with various roles for creating IMS call setups (figure 2.2). The roles are functional and might be collocated on one physical node. Nevertheless, the distributed organization of a IMS network is the reason for dedicated (redundant) hosts per function.

Call Session Control Functions

The Call Session Control Functions (CSCFs) are SIP proxies and the central elements of IMS. They are responsible for routing, initiating and terminating sessions.

Proxy CSCF (P-CSCF)

The Proxy Call Session Control Function (P-CSCF) is the point of contact for a SIP request arriving from a PS UA. Located either in the home or in the visited network, it forwards the requests to either the S-CSCF or the I-CSCF. After initial authentication, the P-CSCF checks the authentication of further UA-initiated dialogs. Usually in live infrastructures, a SBC is located between the P-CSCF and the UA as application layer protecting node.

Interrogating CSCF (I-CSCF)

The Interrogating Call Session Control Function (I-CSCF) is the gateway to other IMS-enabled operators. It is the contact point of all requests incoming to the operators domain. The address of the I-CSCF is stored as a DNS-entry for this operator's domain. Additionally, the I-CSCF selects and assigns an appropriate S-CSCF to home network users during the registration process. Similar to the P-CSCF, it acts as a Topology Hiding Internetwork Gateway (THIG) – a function to hide the internal topology of the operators network.

Serving CSCF (S-CSCF)

The Serving Call Session Control Function (S-CSCF) operates the session control services and is the central component of the signaling plane in the IMS. It both acts as SIP-proxy and SIP-registrar. The user profile (derived from the registration process) is stored and the S-CSCF publishes this registration information by using the location server, which is usually the HSS.

Home Subscriber Server (HSS)

The HSS is the central user database of the provider. It stores all subscriber relevant data and is reliable for authentication and authorization of the user. The S-CSCF and I-CSCF are interacting with the HSS via the Diameter interface to exchange the subscriber-related information. It performs authorization

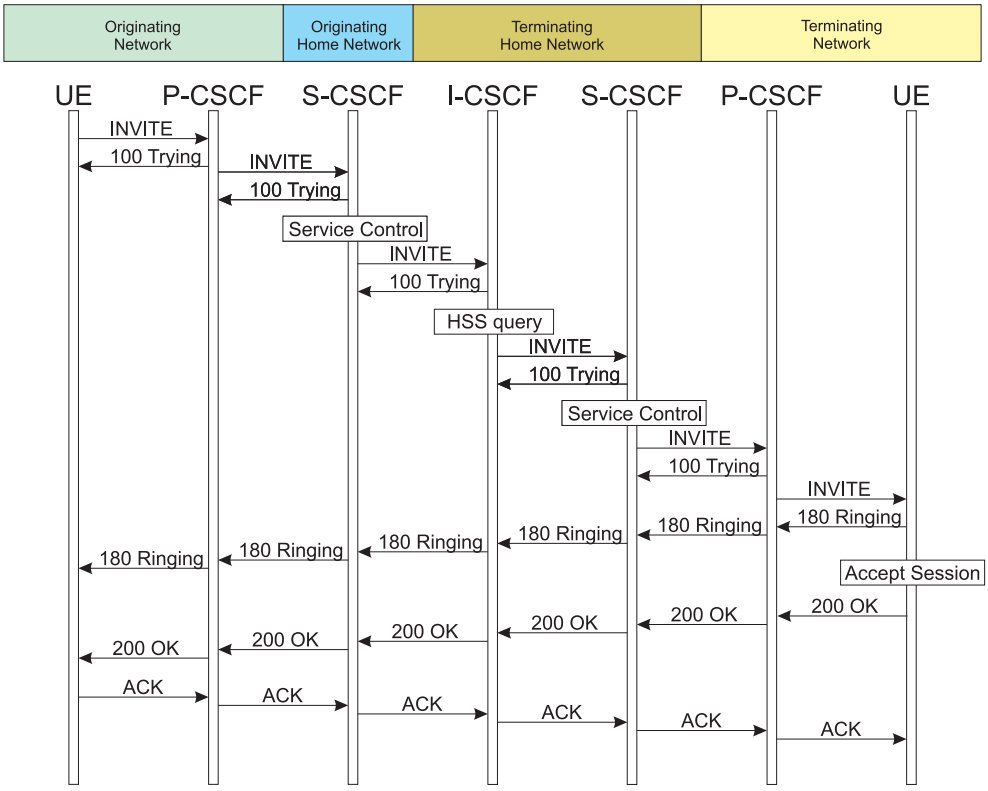


Figure 2.3: IMS connection establishment (signaling)

and authentication of the user and provides information about the subscriber's IP information and location.

2.2.2 IMS call flows

A call setup between two IMS subscribers in two distinct domains results in more than twenty SIP messages passing through the SBC (optional), P-CSCF, I-CSCF and S-CSCF (figure 2.3). The high complexity of these call flows are further extended when the SIP messages are forwarded to Application Servers (AS) or leaving the domain to another provider. The failure or congestion of one IMS core function or even only one node might result in retransmissions of requests or responses, resulting in timeouts or race conditions. It is of huge importance to signal congestion states as early as possible to regulate the allowed request per time interval. Therefore, both the IETF and the ETSI, have started initiatives to standardize congestion control mechanisms, which are presented in chapter 7.

Part I

Congestion in VoIP Networks

Chapter 3

Traffic Distribution and Sources of Congestion

3.1 Congestion in Voice Networks

Finding solutions against overload situations in complex VoIP infrastructures need a differentiation between the sources for overload. Many of the presented issues have the same underlying technical reasons, but the scenarios which lead to these issues vary and need closer discussion.

The first section highlights the changes in paradigm regarding traffic distribution in VoIP infrastructures to understand the changed threat model in VoIP networks compared to classical CS infrastructures.

Section 3.3 uses RFC 5390 [Rosenberg08a] and interviews with engineers of austrian mobile providers to find a list of possible accidental overload situations.

Section 3.4 discusses intended overload situations, like attacks against core- or client-infrastructure.

3.2 Traffic Distribution in VoIP Infrastructures

In CS-networks, the (carried) traffic is defined as an average number of simultaneous calls over a reasonable time interval, usually one hour. The unit used to present this value is "Erlang (E)". The sliding 60-minute period with the highest Erlang-value is called the "busy hour" [itu88], the number of call attempts in the busy hour are the *Busy Hour Call Attempts (BHCA)*. The average duration of a call is called the *Mean Holding Time (MHT)*. With the BHCA (b) in combination with the MHT (t) it is possible to calculate the Erlang (E) using $E = (b * t)/3600$.

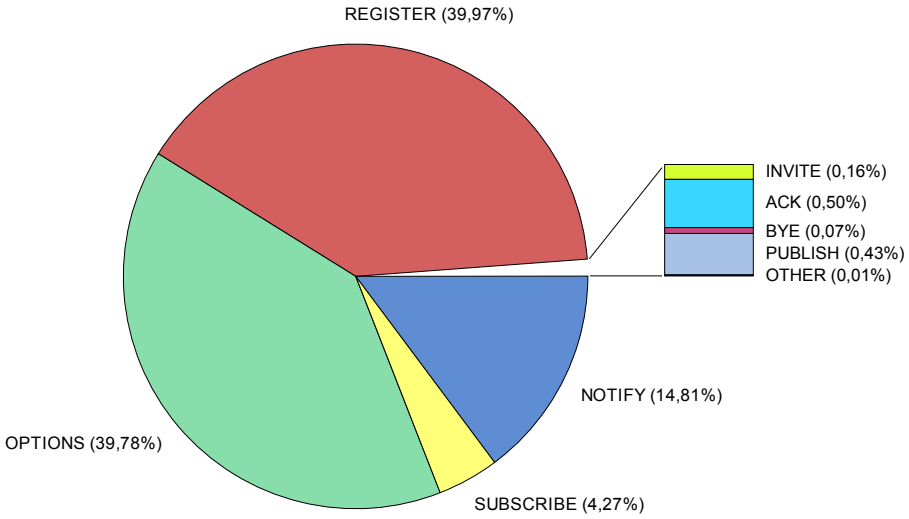


Figure 3.1: Signaling traffic in live-VoIP-systems (here: iptel.org) are mostly non-INVITE requests (sample-size 10.000.000 requests)

The traffic created on VoIP network is divided into the signaling and the media traffic, where ratio between the signaling traffic and the media traffic strongly depends on the usage profile of each user. A user with a high amount of voice (or video) calls creates a high amount of media traffic, whereas a user with a high amount of contacts in its presence list causes a high number of SIP NOTIFY requests.

Hence, finding a definition of "traffic" in VoIP networks needs a differentiation between signaling and media traffic as well as the geographical orientation of the provider and its customers.

3.2.1 Signaling Traffic

SIP as the standard VoIP signaling protocol is on the one hand used for initiating call setups by exchanging media information like supported codecs and supported transport addresses. On the other hand, SIP can also be used to support clients on NAT-traversing by creating background signaling traffic.

Recent analyses of productive VoIP infrastructures (Figure 3.1) illustrate, that the amount of background, non-user-interactive traffic represents over 99% of the total amount of SIP signaling traffic. The observed infrastructure of *iptel.org* handled 10.000.000 requests in 30 hours, resulting in over 100 Re-

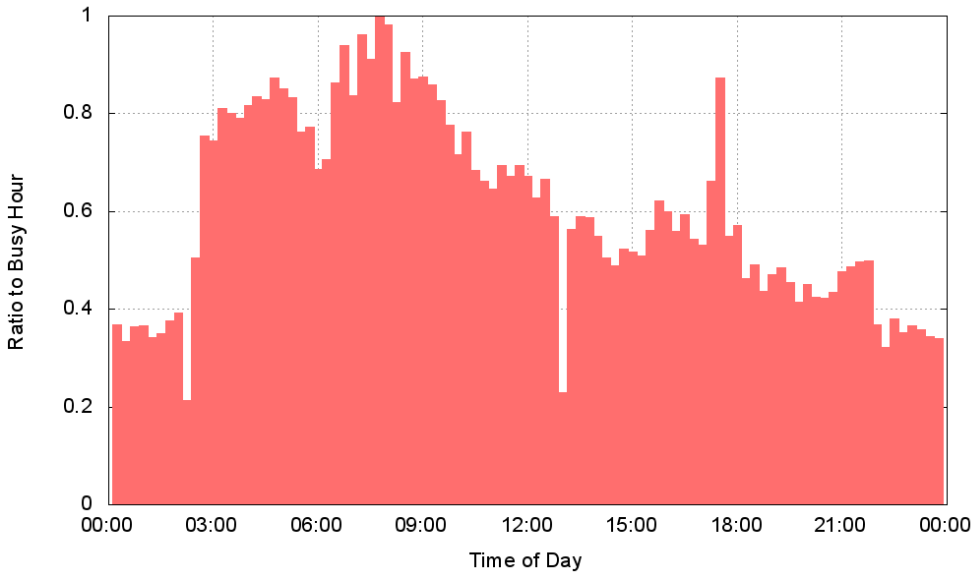


Figure 3.2: iptel.org as a location transparent provider does not show distinct busy hours, but several peaks. The x-axis time is local central European summer time.

quests per Second (rps). Of these 100 rps only 0,1 rps were INVITE-requests. Consequently the amount of call-setup attempts on signaling plane are negligible compared to the massive amount of non-INVITE requests. Section 5 presents examples, where non-INIVTE-requests, in particular the REGISTER-request, will be the reason for overloading a SIP-server.

3.2.2 Media Traffic

Compared to PSTN there is a higher differentiation between high load in signaling and high load in media. From traffic distribution perspective, the media component of Voice over IP (VoIP) best fits to PSTN.

The media component is under risk of congestion when a high amount of *successful* voice- or video-calls are created. Catastrophic incidents, but also New Year's Eve are a driver for problems on media plane. From attacker's perspective, saturating the media component with media traffic is more challenging than using signaling as attack medium: the attacker must create a successful media call and fake SIP header field like e.g., `Authentication:` to pass the authentication mechanisms. In a single DoS-attack, the attacker must have access to a high bandwidth connection to the system-under-attack.

Section 5.4 will present a successful attack against media.

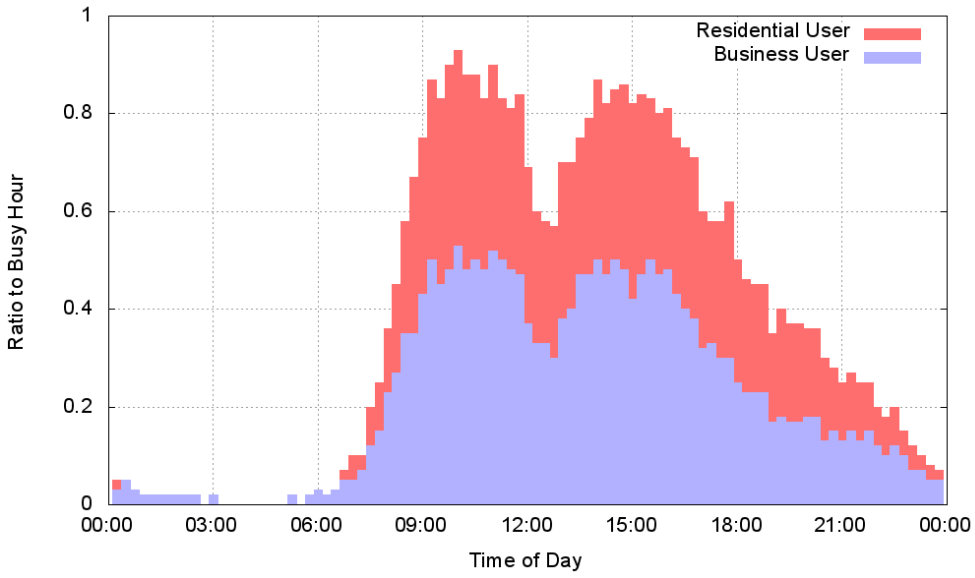


Figure 3.3: Call pattern distribution on a weekday (sample-size: 48 days) compared to the weekly busy hour on friday (Figure 3.4). The diagram distinguishes between residential and business users.

3.2.3 Types of Providers

When discussing the load and traffic pattern in VoIP infrastructures, two types of providers must be discussed differently: Providers, which offer services to customers from a geographical limited area ("*location bound providers*") and providers, whose service is open for worldwide customers (*location transparent providers*).

Location Bound Providers

The first providers serve customers in a local environment and can be understood as equivalent to national CS-providers.

These CS-providers offer VoIP as value added service. "A1overIP"[1] of A1 Telekom Austria and the "VoIP Phone"[2] service of the swiss national provider *swisscom* are examples of such services. Due to this local expansion, a distinct daily usage and traffic profile like presented in image 3.3 can be monitored. Figure 3.3 presents information to identify the common usage of this infrastructure: clear peaks at 10am and 3pm, but low traffic in the

[1] http://www.a1.net/hilfe-support/A1-over-IP (retrieved June 11^{th} 2013)

[2] http://www.swisscom.ch/solutions/de/start/loesungen/telefonie-ucc/enterprise-telephony/hosted-enterprise-telephony.html (retrieved June 11^{th} 2013)

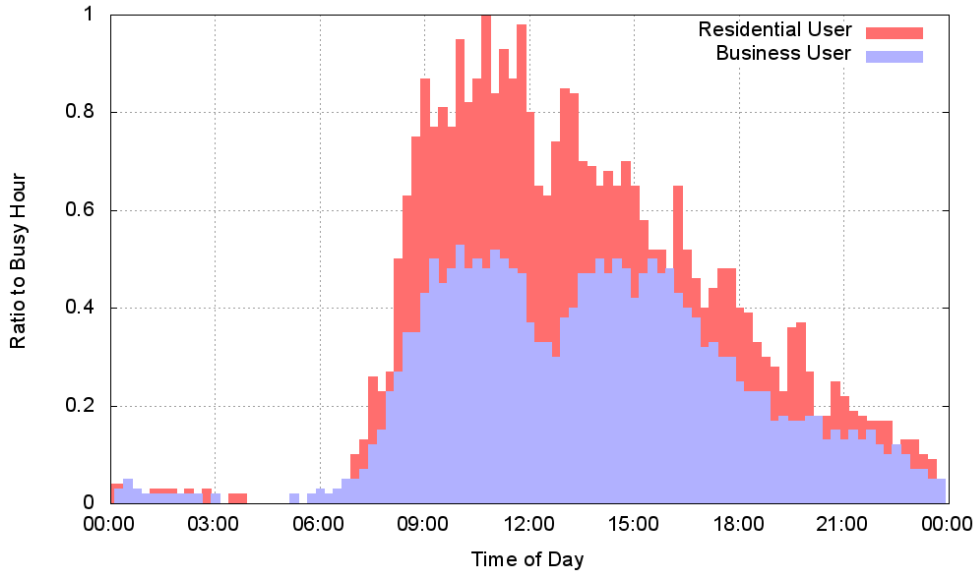


Figure 3.4: The week's most busy hour can be observed on friday morning. In afternoon, the decreasing usage is based on the fact that many employees are leaving on friday noon (sample-size: 12 days)

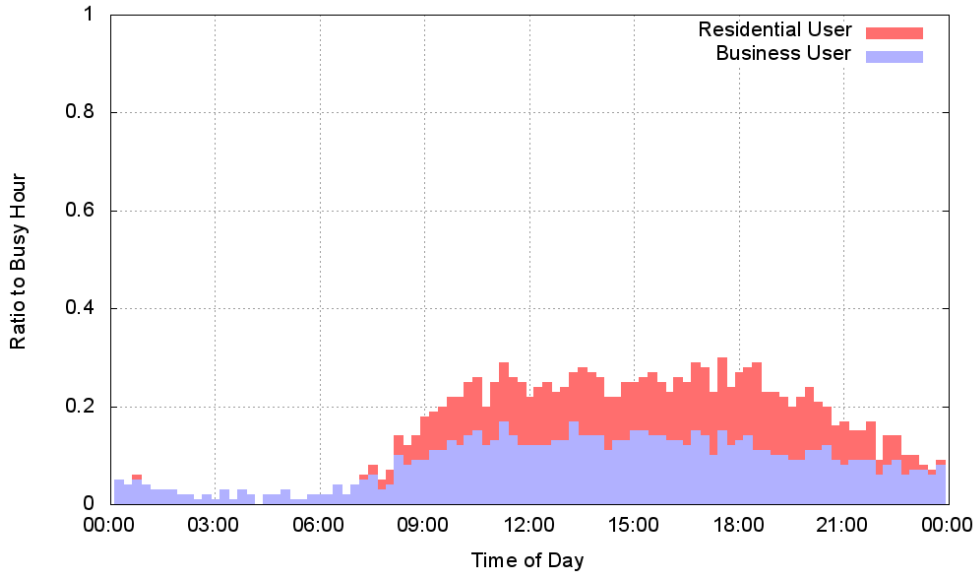


Figure 3.5: There is no clear distinction in usage profiles between saturday and sunday. Therefor the profiles of these two days have been aggregated. The most busy hour is only factor 0.3 of the weekly peak (sample-size: 24 days)

evening define this system as business-user oriented. This interpretation is confirmed by decreasing use on friday afternoon and the common low usage of this system on weekend as depicted in figure 3.4 and 3.5. The input data of these figures are a data warehouse dump (over twelve weeks) of VoIP calls originating or terminating in the A1 Telekom Austria IMS core.

Location Transparent Providers

The other kind of provider is *location transparent* with customers distributed all over the world. The used data from *iptel.org* (ref. figure 3.2), a provider located in Germany, show no clear busy-hour profiles like in figure 3.3, instead, there are local local peaks and drains.

The distribution of the traffic and the busy hour with respect to overload situations is interesting on New Year's Eve: the local bound provider has its peak around midnight local time. Instead, the location transparent provider must consider a risk of congestion all over the 31st of December, as the New Year's Eve-parties differ depending on the time-zone.

3.3 Accidental Congestion

In dialog with the operational team of A1 telekom austria, we aggregate the most common reasons for overload situations in current carrier grade networks. At the time of writing, the critical situations at the A1 telekom austria voice communication core were mostly driven by accidental overload or by broken components.

3.3.1 Televoting

Scenario: Calls uniformly distributed over a provider's service area to an Interactive Voice Response (IVR)
In this case, there is a peak of phone calls against one IVR. This peak is distributed over the entire service area and over a few minutes of time. Currently, televotings with an expected higher number of calls are announced by the televoting person-in-charge to the operators in advance to let the affected providers prepare the infrastructure for the upcoming higher load.

In SIP infrastructures, this overload type will in particular affect the media proxies and -gateways, or – in a fully SIP-enabled infrastructure – the IVRs.

Signaling handling components like SIP-proxies should not be affected due to the low ratio of SIP INVITE requests compared to the total SIP traffic. However, there is a risk of overloading the responsible AS as all calls must be handled by this centralized component.

3.3.2 "Call-Now"–Lotteries

Scenario: Calls uniformly distributed over the provider's service area to an IVR with a distinct, short-time peak-load
Compared to *Televoting*, "Call-Now"–lotteries are mass-calls with a more distinct peak-load: for example, a nationwide radio station starts a lottery, where the first caller at a predefined time wins one million Euro. It must be expected, that there is a more striking peak over a short time interval than in the televoting case. In this case, the media plane is under higher risk. The affected components and the impact are identical to the "Televoting" scenario.

3.3.3 Catastrophic Incidents

Scenario: Geographically limited, multiple subscriber call a few specific (emergency) numbers
This threat is menacing in particular mobile providers as witnesses and victims are calling the emergency services mostly using mobile infrastructure.

The bottleneck in this technique can be traced back to two distinct locations: The first overloaded component is the Base Transceiver Station (BTS) as these calls are originated from the same affected area. The second bottleneck concerns the emergency service operators: In Vienna, the Police emergency hotline is operated by four to eight operators. If the lines to these operators are in use, the caller is put on hold.

These two components will likely be affected also in future SIP driven Voice over LTE (VoLTE)-networks.

From core infrastructure aspect, there will be no specific impact.

3.3.4 New Year's Eve

Scenario: Uniformly distributed subscribers call distinct numbers
New Year's Eve is the peak evening of telecommunication facilities. In no other time-period, more calls are created to domestic and foreign parties. The high amount of calls in the midnight hour is exceeding every infrastructure's capacity and it is not economic to try to handle all of these calls.

From the VoIP perspective, most probably the media-component might be overloaded. As displayed in Figure 3.1, an increase of the user-originated traffic like INVITE- or MESSAGE-requests by factor e.g., ten, is still far below all other, background-originated requests.

Another aspect is be the presence infrastructure, which depends on the type of provided services: if the customers utilize the presence system frequently, an additional impact on the system could be expected: In the iptel.org-measurements, around 20% of all request are presence related. An increase of again factor ten might have strong impact on the core infrastructure.

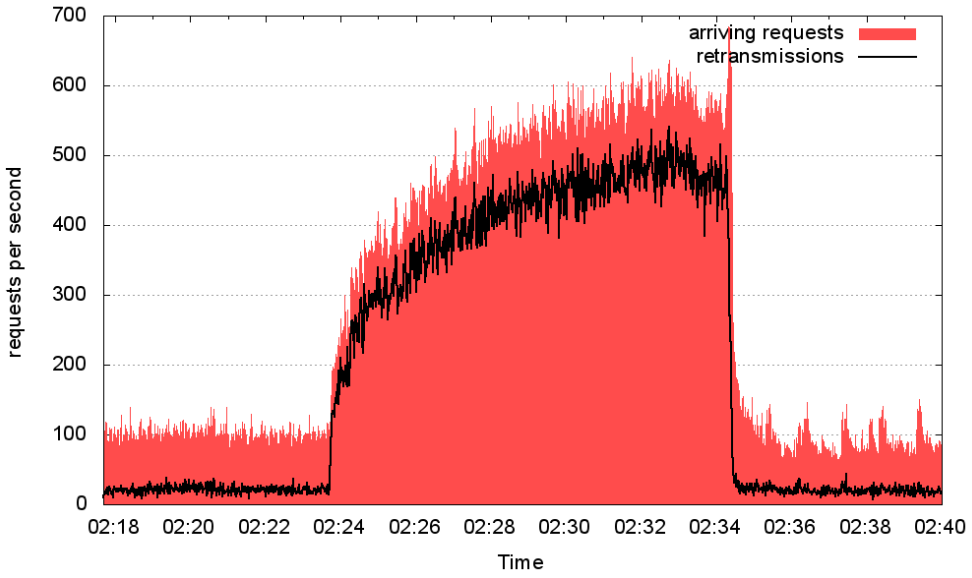


Figure 3.6: The component failure of the iptel.org-proxy between 02:24 and 02:34 resulted in a six-times increased SIP traffic. At the time of recovery, the number of requests increased again by 100 RPS.

Finally, there is a need to distinguish between location bound and location transparent providers. As discussed in section 3.2.1 location transparent providers might have high load traffic all over the 31st of December/1st of January.

3.3.5 Component failure

Scenario: One or more core- or peripheral components fail to provide services
As PS- or CS-components are complex electronic devices, failing of such a component is possible. In normal operation state, redundancy can keep the infrastructure operational, but multiple failures or the failure of non backuped core components may lead to outages of a system or part of the system. Component failure can also be the failure of surrounding and peripheral infrastructures, like failing power supplies of specific areas or regions. Moreover, returning power and subsequent reregistrations of large pools of clients can lead to core overload. By design, the failure of one component in SIP-enabled environments can lead to problematic behavior: The retransmission functionality of UDP-sent SIP-packets is designed to recover from packet loss: When a SIP-client does not receive a SIP response within a predefined time of 500 ms (RFC 3261 [Rosenberg02], 17.1.1.1), the client starts retransmitting the request until the node gets a response or the timer B (for INVITE transaction,

64*T1) or F (for non-INVITE transaction, default also 64*T1) are reached. With an increasing delta between each retransmission, a SIP request will be retransmitted up to 11 times. This behavior creates additional load on the failed infrastructure and possibly prevents a system from recovering.

Figure 3.6 depicts the breakdown of the core SIP proxy of *iptel.org* lasting ten minutes. Due to the non-responding component, the clients start retransmitting the requests. Hence the number of requests per second increase from the usual 100 rps up to 600 rps. The figure illustrates that 80% of this traffic is caused by retransmission of requests. At the time of recovery (02:34), the number of retransmissions decrease immediately (from 500 RPS to 10 RPS), but in parallel, the number of initial requests increase. Finally, the system is loaded with 700 RPS at 02:34. If a system is only designed for e.g., 500 to 600 RPS, this recovery-peak may cause the proxy go and stay in congestion-state [Egger12].

The reason for this steady increasing load is depicted in Figure 3.7: the provider iptel sets as default re-registration interval a timespan of 600 seconds. As this breakdown lasts slightly more than ten minutes, each user agent runs into register expiration and aspires to re-register during the time-of-breakdown. As the clients do not get any response, the retransmission mechanisms are activated at each client resulting in a massive number of retransmitted register-requests. At recovery time, the number of retransmissions decrease abruptly as this proxy is capable to accept all register requests arriving at this time. This capability is caused by the slim design of the *iptel.org* application layer infrastructure where the home subscriber server and the SIP proxy are directly interconnected without any additional SIP proxies.

Complex IMS systems with multiple routing hops in the core infrastructure will need more resources and therefore more time to recover from such a failure.

3.3.6 Avalanche restarting

Scenario: A large number of clients cause overload by simultaneous restarts
One other major risk for for getting into an overload state is the simultaneous re-registration of clients after a major component failure. No matter if in classic PSTN or in SIP, the coincidental sending of registration requests against one central component will create high load. In combination with retransmissions after delayed responses, the load can increase up to an overload state.

For example, a power failure in a facility with many SIP clients generates (after power recovery) a high number of synchronous registration attempts. This behavior is similar to “Component Failure (Section 3.3.5), but here, the source of the load is from a limited geographical area, possibly one or few IP

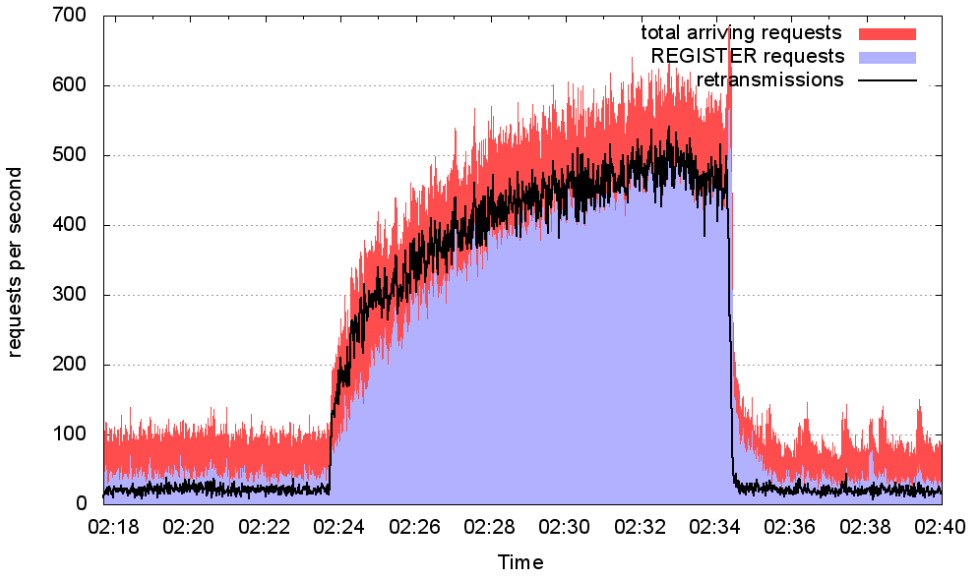


Figure 3.7: The component failure of the core iptel.org-proxy resulted in an increased SIP traffic. 75% of all requests are re-registration attempts (incl. retransmissions).

addresses of gateways and created by REGISTER-requests.

The massive amount of registration request in regular SIP traffic is depicted in Figure 3.1, where about 40% of all regular traffic in the observed infrastructure is contributed by registration requests. Avalanche rebooting of a large number of SIP-clients will increase this high amount additionally.

Section 5.3 will present an DoS-attack successfully emulating an "avalanche restarting" VoIP-infrastructure.

3.3.7 Machine-to-Machine communication

Scenario: Synchronized data transfer of a high number of embedded systems
A new source for congestion is the increasing number of embedded systems using Machine-to-Machine (M2M)-communication for data exchange. M2M is referred as idea, where several networked components are communicating with each other without any human interaction. Examples for this M2M-traffic include smart grid communications for power-grid control and charging as well as health care patient monitoring components or even vending machines.

As already discussed in section 3.2.1 currently more than 99% of all SIP-requests are non-interactive, automatically generated SIP requests. With the increasing number of M2M devices this currently high number will increase even more. Recent Ericsson's studies predict reaching a total number of 50

billion connected devices within 2020 [Alendal10].

The risks in M2M-communications with respect to congestion are manifold: Assuming time synchronized automated reboots of all devices from one vendor after an automatic firmware upgrade will likely be hazardous [Phuyal12], this behavior being similar to avalanche restart after power failures. Additional threats might be system failures, time synchronized periodic status communication to a central component or an undetected botnet composed of countless vending machines.

3.4 Intentional Overload - Denial of Service (DoS) Attacks and SPAM Flooding

The other major reason for inaccessible and nonfunctional telecommunication services are attacks against the infrastructure. The intentional overload can be targeted against two domains: On the one hand against the core infrastructure itself to impede delivering services. On the other hand, the core infrastructure can be misused to guide an attack against a customer's infrastructure.

In this section, we introduce the DoS-threat and its variations as well as the threat of creating Unsolicited Communication (UC).

3.4.1 Denial of Service (DoS)

DoS is a type of attack against a victim's infrastructure to make this resource unavailable or unrespondable to its intended users. This task is solved by the attacker by over saturating the infrastructure with request on Internet-, connection- or application-layer in the TCP/IP reference model.

In complex VoIP infrastructures basing on SIP, these DoS-attacks can be executed by, e.g., massive amounts of SIP-requests originated by one (DoS attack) or a varying number of hijacked clients ("bots" or "zombie agents") organized as "botnet" controlled in an abstract layer system by a central command-and-control server (Distributed Denial-of-Service (DDoS) attack).

Section 5 will present practical examples of a successful DoS-attack whereas chapter 10 will present approaches to protect the core infrastructure against DoS- and DDoS-attacks

3.4.2 "Unsolicited Communication"

UC or also known as SPam over IP Telephony (SPIT) is an evolutionary advancement of the ubiquitous Email-SPAM-threat. In UC, the "SPITter" generates voice- or text-messages and forwards them to the provider's customers. Using the same technique as DoS-attackers, the aim is not to block the core infrastructure, but to relay his advertisements to as many as possible customers.

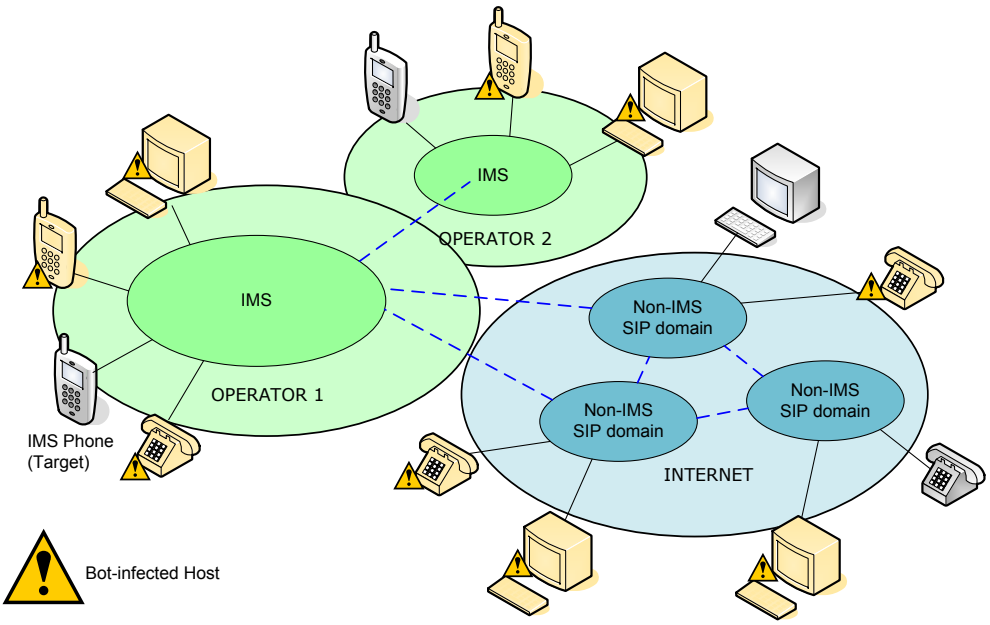


Figure 3.8: Bot-infected hosts as distributed attackers against one client or a core infrastructure

Chapter 4 will discuss the challenges encountered in defending SPIT. Chapter 10 will present with the SBC-A a combined tool to protect a providers infrastructure and its customers.

Chapter 4

Mass-Calls as Congestion Risk

Unsolicited Mass-calls (UC) are a phenomenon which is already well known from the circuit switched domain and legislation already reacted on this threat by putting strong limitations when and under which circumstance mass-calls are allowed [TKG].
The current move to IP-based packet switched signaling in telecommunication networks is increasing this threat by making it easier to deploy UC. Hijacked VoIP accounts together with automatic call-generators like the *SIP Performance Test Tool (SIPp)* are a massive challenge and can result in the same dimension of UC as Email-SPAM during the last years. If these VoIP accounts have access to the classic CS-domain via SIP to Circuit Switched (SIP2CS) media gateways, this threat is not limited to the VoIP-domain anymore.

Additional, the increasing market-share of smart phones with complex Operating Systems (OSs) (like Apple iOS or Google Android) is also resulting in a risk of UC: an infrastructure of centrally controlled hijacked smart phones with one call-generator per phone is also able to create a load of calls within a short timespan.

Both, the hijacked SIP-Accounts or smart phones are a risk to the core network infrastructure, no matter if this core infrastructure is PS or CS controlled.

As a consequence to the harassment of UC and the various meanings and interpretation of UC-related terms, the IETF published the informative RFC "The Session Initiation Protocol (SIP) and SPAM (RFC 5039)" [Rosenberg08b]. This RFC gives an adequate overview over the various types of SPAM in the VoIP environment and some proposals to defend against threat.

This definition is also usable for the IMS infrastructure, whereas the 3GPP Standard 33.937 [3GPP10c] goes more into detail of the specific IMS-related challenges and pitfalls. Some of the proposals to defend against UC are also

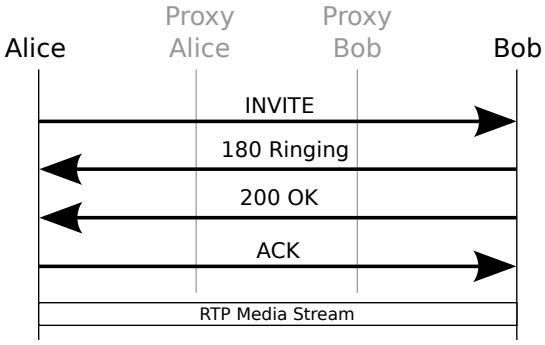


Figure 4.1: Direct end-to-end communication omitting intermediate SIP proxies

integrated in the Overload Control (OC)-controlling infrastructure presented in chapter 10

4.1 Attack Scenarios in plain SIP

The communication between two SIP user agents can be routed directly or via one or more SIP proxies. To understand the different attack scenarios, we have to discuss the distinct ways SIP requests are routed from UAC to UAS.

Direct End-to-End Communication

The simplest method is to send the message directly to the SIP client of the terminating user. If the called user resides on `pc13.biloxi.com` and the SIP UAS is awaiting messages on (default) port 5060, the UAS can be directly addressed by sending SIP requests to `sip:bob@pc13.biloxi.com[:5060]`. In general, all tested user agents ignore the user-part of the SIP-Uniform Ressource Identifier (URI) as long as hostname and port are correct. This use case is rare in Internet Protocol v4 (IPv4) environments as most UAs are behind *symmetric NAT* gateways. These gateways forward requests from external hosts only when these already received packets from the internal host. An overview over the different types of NATs and their vulnerability against direct addressing is presented in Table 4.1.

With the introduction of Internet Protocol v6 (IPv6), the number of UAs with public accessible IP-addresses will increase and this attack scenario using direct end-to-end communication will probably get more interesting. For IPv4, a working scenario is an office Local Area Network (LAN): In this environment an “in house”-attacker can brute force all IP addresses to find a SIP UA.

From the SPAMer’s perspective, direct addressing of SIP clients is the most promising technique as there is no protection infrastructure in between. Most known attacks (e.g., ”Analysis of a VoIP Attack” [Darilion08]) using direct

NAT type	description	vulnerable against direct addressing
Full-cone NAT	All external hosts can send packets to the external socket (*eAddr:ePort*) which are forwarded to the internal hosts address (*iAddr:iPort*)	yes
(Address)-restricted-cone NAT	Only external hosts, which already received a packet from the internal host, (to hAddr:*any*) can send packets via eAddr:ePort to *iAddr:iPort*	no
Port-restricted cone NAT	like *(Address)-restricted-cone*, but the external host (*hAddr:hPort*) can send packet to *iAddr:iPort* via *eAddr:ePort*, only if *iAddr:iPort* sent previously a packet to *hAddr:hPort*	no
Symmetric NAT	Each request from *iAddr* to a specific *hAddr:hPort* is mapped to a unique *eAddr:ePort*. Only external host which received a packet to *hAddr:hPort* can send a packet back to *iAddr:iPort* (via the unique *eAddr:ePort*)	no

Table 4.1: Methods of port translations (RFC 5389 [Rosenberg08c])

end-to-end communication are targeted to find public accessible SIP-to-CS gateways for toll frauding.

Callees proxy/registrar

The registrar is usually co located with a SIP proxy server. This server routes incoming requests to registered SIP UAs. On such a SIP proxy, all SIP-requests dedicated to a specific SIP-URI (e.g., `sip:bob@biloxi.com`) are routed – using the *Domain Name System (DNS)* SRV lookup – to the SIP-proxy of `biloxi.com`. This SIP-proxy is either also a registrar or accesses an external registrar for the current specific location of subscriber `bob`'s UA. It forwards the request to the socket where bobs UA is registered (`sip:bob@pc13.biloxi.com[:5060]`) and also forwards bobs responses to the request-sending node.

From a SPIT-protecting point-of-view, the significant benefit of a SIP proxy is its single-point-of-contact-role: all traffic is routed via this same hop and

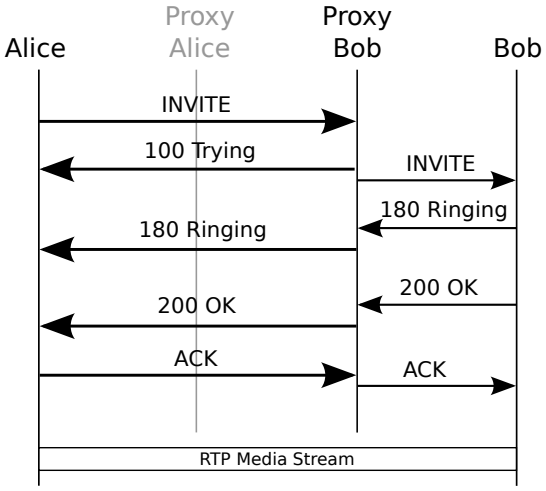


Figure 4.2: SIP communication via the foreign proxy

this hop (the SIP proxy) can implement intrusive and non-intrusive SPIT-prevention mechanisms. However the SIP proxy can not authenticate requests from calling parties arriving from a foreign domain. Here (equivalent to Email), a proxy must rely on the authentication mechanisms of the foreign SIP provider.

In current practice, providers use amongst other techniques white lists[1] of trusted providers for assuring the reliability of incoming requests. The benefits and drawbacks of this approach is discussed later in the SPIT-defending chapter 8. In addition, RFC 3325 [Jennings02] defines a privacy framework which is also applicable for inter provider SIP communication.

Unfortunately, the mechanism proposed in RFC 3325 is not commonly used and therefore the called provider cannot rely on the correct identity of a caller.

Caller's and callee's proxy/registrar

The use of one or more proxies both on the caller's as also on the callee's side is defined as "SIP trapezoid" in RFC 3261 (figure 2.1). The caller transmits all requests to its own outgoing proxy server, which forwards the requests to the proxy-server responsible for the UAS' domain. The latter proxy relays the message to the called user's UA. In this scenario, the sender's proxy has detailed knowledge of the identity of the caller and the receiver's proxy knows about the identity of the callee. For inter-domain authentication, the standard RFC 3261 (26.3.2.2) proposes the use of Transport Layer Security (TLS) connections, but the huge and varying number of proxies and the complexity

[1] e.g., the austrian provider A1TA extensively uses DNS white lists

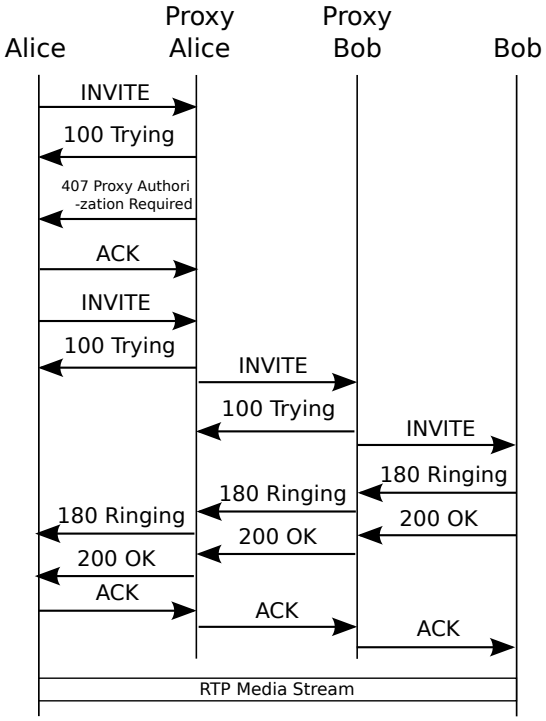


Figure 4.3: SIP communication via the own and the foreign proxy

of certificate handling are the reason, why this technique is not very popular. Without using this mutual authentication, the caller's proxy must rely on correct authentication of the sender on the sender's proxy.

The trapezoid supports creation and maintenance of white lists: The callee's proxy knows, that all request from a specific domain A must originate from the proxy of domain A. All other requests with caller-URIs from domain A can be blocked. The handling of white- and blacklisting and their advantages will be discussed in chapter 8.

In reality, most SPIT attacks are using the second attack variant (*Callees proxy/registrar*). The called domain can therefore not verify the identity of the foreign-domain caller but has to forward the request (with respect to VoIP-communication "openess") to the callee's socket. Main benefit and drawback of the infrastructural approach is the single component where all traffic is aggregated and where all traffic can be inspected with respect to SPIT- and DoS-protecting criteria. If the called proxy is not just one single server but a complex infrastructure like IMS, the SPIT-preventing-component shall be located at the border of this infrastructure to prevent overloading this complex infrastructure.

4.2 RFC 5039: SIP and SPAM Definitions

The aim of RFC 5039 is to define the problem space of UC, resp. SPIT and SIP in an informative way to specify a reference document for further scientific work and more specific standardization activities. For this document, the RFC is used as a red-line and is extended by more current findings and scientific work.

First, RFC 5039 defines the various types of UC:

Call SPAM

As most annoying type of UC, the Call SPAM is defined as a large amount of INVITE session initiation requests for establishing a voice or video call. If the call is successfully established, the initiator of SPAM – the SPAMer – transmits live or pre-recorded media to communicate its message as in classical telemarketing. This technique is usually called SPIT.

IM SPAM

Instant Message (IM) SPAM (*SPam over Instant Messaging (SPIM)*) is the closest comparison to Email-SPAM. It is defined as a bulk of text-messages to a number of recipient mostly by using the MESSAGE-request of RFC3261 [Rosenberg02]. But also other requests presenting text-messages to the SIP client can be used, like INVITE-requests where the IM SPAM is transmitted in the Subject-header. Due to its textual representation, defending IM SPAM can be done by using the same technology as used for defending Email-SPAM.

Presence SPAM

Similar to IM SPAM, *SPam over Presence Protocol (SPPP)* tries to transport the message by using presence requests like the SUBSCRIBE request. This presence SPAM can be used for two objectives: first, to communicate the message by filling the subscribe reason with the SPAM-message and, second, to be added to the target's whitelist. Once on the whitelist, subsequent call- and presence-SPAM messages are reaching the victim at a higher probability as many SPAM defending infrastructures rely on these white lists.

Technically, other SIP requests and responses can also be used for transferring unsolicited messages. These messages are not presented to the customer and are hereby worthless for the SPAMers intention.

4.2.1 Call and Voice SPAM

"Call and Voice SPAM" is the most intrusive type of unsolicited messages in SIP. RFC 5039 assumes that VoIP SPAM will occur in future. In current CS-networks, call SPAM is already appearing through telephone marketing calls.

The volume of incoming telemarketing calls is however limited because of two factors.

First the legal section: in most countries of the European Union (EU) unrequested and unauthorized initial advertisements – so-called "Cold Calls" – are forbidden by law, e.g., the Austrian *Telekommunikationsgesetz* [TKG], 107 states:

> §107. (1) Anrufe - einschließlich das Senden von Fernkopien - zu Werbezwecken ohne vorherige Einwilligung des Teilnehmers sind unzulässig. Der Einwilligung des Teilnehmers steht die Einwilligung einer Person, die vom Teilnehmer zur Benützung seines Anschlusses ermächtigt wurde, gleich. Die erteilte Einwilligung kann jederzeit widerrufen werden; der Widerruf der Einwilligung hat auf ein Vertragsverhältnis mit dem Adressaten der Einwilligung keinen Einfluss.
>
> *§107. (1) Calls – including the sending of telefax – for the reason of advertisement without previous agreement of the customer are illegitimate. Customer agreement means the agreement of a person who is allowed to use the connection. The agreement can be abrogated at anytime; the abrogation has no influence on the contractual relationship.*

If advertisement calls are placed from an international area where jurisdiction and/or law enforcement authorities are not interested in acting against these callers, national legislation becomes ineffective.

Here, the second aspect against a massive appearance of unsolicited messages in CS comes to effect: costs. Depending on (optional) roaming costs, call-rate and call-duration, massive costs will arise for the caller. Because of these costs, massive amounts of unsolicited calls via CS have not yet reported. This might change in future, when first hijacked smart phones or hacked SIP2CS-gateways will appear. If hijacked smart phones are misused as bot clients, these might also create mass-calls without knowledge of the owner or even the operator [Mulliner10]. The same apply to SIP2CS-gateways: such hacked gateways can create calls at the expense of the owner.

When comparing CS against VoIP, the legal barrier is still valid. However, with respect to the second aspect, costs of 0,25$ per call in CS decrease to 0,0000062 $ (6.2 μc, RFC 5039, 2.1.) per VoIP call. Being calculated using costs of broadband access in early 2008, these 6.2 μc are by an order of magnitude of 4 lower than costs of CS-mass-calls. Currently (2013) these costs are further reduced. Moreover, these costs are independent on the called location and no roaming costs apply, such that UC can be originated from countries with a low 'legal pressure' against SPAMers without any additional costs.

As stated in RFC 5039, there is a high probability that SPAMers do not even have to pay this amount, as they might reuse existing Email-SPAM infrastructure: hijacked PCs or mobile terminals as part of a remotely controlled botnet. By extending the bots of an Email-botnet with VoIP capability, already established botnets could be used with limited effort. Calculating the total costs for VoIP SPAM must also consider the collection of URIs: RFC 5039 argues that in future the extension of Email-addresses to VoIP-addresses will be one way to get SPIT-victims: trying to create a VoIP-call addressing already collected Email addresses might have a high probability of success.

Another way of getting SIP-compliant addresses is try sequentially the finite number of telephone-numbers by directly calling them via a hijacked SIP2CS-gateway or by sequentially digging the public ENUM-databases for entries as presented in an example in the practical part 5.

Chapter 5

Specific Attack-Examples

5.1 Introduction

This section presents two specific examples of DoS-attacks against specific components and the preparatory work. The first section defines techniques to find valid identities on the attacked system, whereas the following section presents a method to create a DoS-attack using REGISTER-flooding, as well as a way to oversaturate a media gateway.

5.2 Harvesting SIP URIs

For successfully addressing users to send SPIT messages, the SIP-URIs of the target users must be available. This section presents three different techniques to acquire this information.

5.2.1 Public ENUM-enumeration

For the convergence of classic PSTN to VoIP it is essential to dial telephone numbers the way customers have learned over the last decades. The *E.164 NUmber Mapping (ENUM)*-service defined in RFC 6616 [Lear12] is a DNS-extension to map telephone numbers into SIP URIs using special DNS record types. When a caller creates a call using a telephone number, the client (or a intermediate proxy) transparently creates a DNS ENUM query to resolve the telephone number into a SIP URI. This alphanumeric URI is then used for establishing the call or sending the message.

Creating an ENUM-Query

The basic steps of ENUM-translation using the example-phone number *+44-20-7946-0148* [Lear12] are as following:

1. Remove all spaces and other non digits. *+44-20-7946-0148* converts to *442079460148*
2. Reverse the order of the number. *442079460148* is modified to *84106 4970244*.
3. Add dots between the digits: *841064970244* is converted to *8.4.1.0. 6.4.9.7.0.2.4.4*.
4. Add the trailing string ".e164.arpa." resulting in *8.4.1.0.6.4.9.7.0.2.4.4. e164.arpa.*

The resulting string *8.4.1.0.6.4.9.7.0.2.4.4.e164.arpa.* is now interpretable as domain name.

Fetching SIP-URIs

The free service `e164.org` is a public accessible ENUM server which can be used for SIP-URI-fetching: With the following steps all Vienna's public phone numbers can be checked, if one of them is mappable to a SIP-URI. As these phone numbers all start with the country-prefix 43 and the local prefix 1 (e.g., +43-1-123 45 67), the last section of the ENUM-"domain" looks as following: .1.3.4.e164.arpa. The usual number space of Vienna's phone numbers is a seven digit number, resulting in about 10 million phone numbers (0-9.999.999). After removing all digits smaller than 1.000.000, 9 million numbers are remaining. When sending 10 enum requests per second (to avoid an unknowingly DoS-attack against the ENUM-server), all these numbers are checked within 900.000 seconds (250 hours or ten days). Rural areas with small numbers of subscribers per area code can be harvested within a fraction of these 900.000 s. A five-digit area can be harvested with 10 enum requests per second within 167 minutes.

A Perl-script testing a subset of the Viennese numbers is presented in Listing 5.1. The resulting SIP-URIs are stored in an array named @results.

```
#!/usr/bin/perl
use strict;

my @results;
my $l=0;
while($l < 10){
  my $m=0;
  while($m < 10){
    my $n=0;
    while($n < 10){
      my $o=0;
      while($o < 10){
```

```
        my $sipuri=`dig $o.$n.$m.$l.4.8.4.1.3.4.e164.org
                    \@e164.org -t NAPTR | grep E2U\+SIP
                    | cut -d ! -f3`;
        if (length($sipuri)>0){
          push(@results,$sipuri);
        }
        $o++
      }
      $n++;
    }
    $m++;
  }
  $l++;
}
```

Listing 5.1: Perl script to fetch all SIP-URIs from the e164.org-server for phone numbers starting with +43 1 484

After 10 days of (background) testing using a simple Commercial Off the Shelf (COTS)-computer, a list of all SIP-URIs of Vienna fixed line customers is available. The resulting numbers can be sold for regional specific SPIT-communication.

5.2.2 REGISTRATION-request feedback

This technique of user guessing relies on the deployment of IMS infrastructure as "early-IMS"-VoIP services. "Early-IMS" [3GPP07] is a common name for IMS infrastructures, which are not restricted to the telecom core network accessible only with dedicated IMS hard- or mobile phones, but also using the public Internet access. Examples for such installations are "A1 over IP" and "A1 Network Unified Voice Services" by A1 Telekom Austria, "Vip over IP" by the Croatian vipnet-mobile phone provider or the "VoIP phone"-service by the swisscom, the national Swiss telecom provider. All these providers offer the IMS-service in a reduced form as a value added service to their customers.

A potential security weakness is the IMS authentication procedure: it differs from best-practices in Internet security: 3GPP TS 33.203 rel. 12 [3GPP10a] states, that a valid subscriber shall be challenged with a SIP "401 Unauthorized" response (TS 33.203, 6.1.1, depicted in Figure 5.1). In contrast, invalid and non-existing subscriber shall be blocked with a final "40x"-error response without any authentication-challenge header (TS 33.203, 6.1.2.2, depicted in Figure 5.2). This differentiation is crucial, as unauthorized attackers can send multiple SIP registration requests with varying subscriber names within a small amount of time to find valid identities.

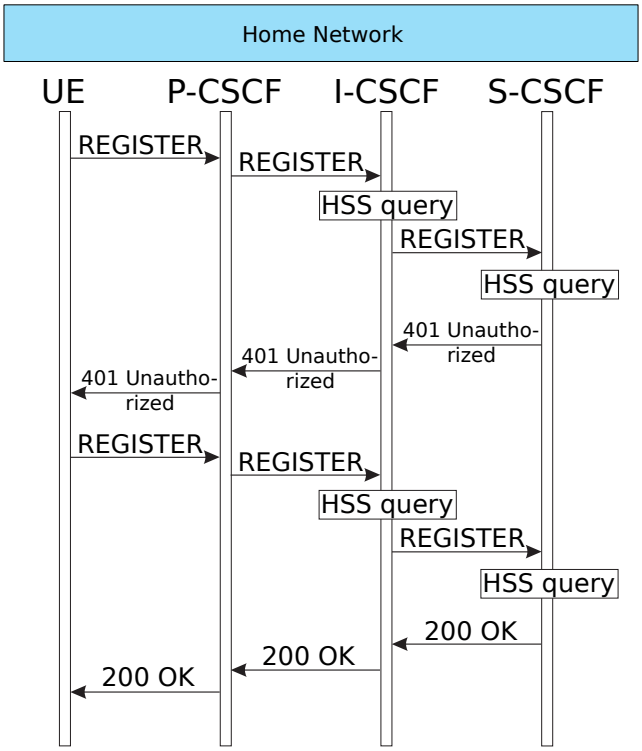


Figure 5.1: A successful client registration as defined in 3GPP 33.203. The "401 Unauthorized" response contains a SIP-authentication header.

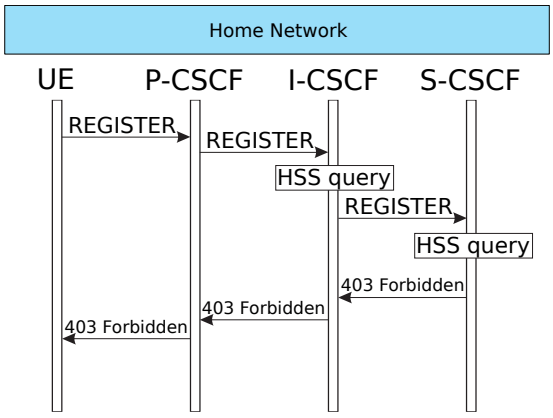


Figure 5.2: A client registration-attempt of a unknown identity as defined in 3GPP 33.203. The "403 Forbidden" response contains no SIP-authentication header and is hereby different than the response to requests containing a correct user-identity.

Realization

This attack scenario is implemented with SIPp, a flexible XML-controlled SIP message generator. The XML-infile defines a simple register request and handles the incoming responses as illustrated in Figure 5.2.

```
<scenario>
  <send retrans="500">
    <![CDATA[
      REGISTER sip:a1.net SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];
        rport;branch=[branch]
      From: [field0] <sip:[field0]@a1.net>;tag=[pid]
        [field0][call_number]
      To: [field0] <sip:[field0]@a1.net>
      Call-ID: [call_id]
      CSeq: 0 REGISTER
      Contact: sip:[field0]@[local_ip]:[local_port];rport
      Expires: 3600
      Content-Length: 0
    ]]>
  </send>

  <recv response="403" optional="true"  next="1">
    <log message="User [field0] does not exist"/>
  </recv>

  <recv response="401">
    <log message="User [field0] exists"/>
  </recv>

  <label id="1"/>
</scenario>
```

Listing 5.2: REGISTER request XML-listing as SIPp control-file

The fields enclosed by brackets are filled during the test run and can be divided into two types: the first ones are filled automatically with system-specific values like IP-addresses or ports. The other placeholders named [field0] (line 6 and 7 in Listing 5.2) are filled with values of a comma separated values-input file.

This input file contains a predefined list of possible user names. During execution of SIPp, the application steps sequentially through this list (the first 5 lines are presented in Listing 5.3) and changes the value with each transmitted SIP-register request. Hence, each request sent by SIPp varies by

the user-part of the registering URI. Depending on the user-part, the attacked system responses either with a “401 Unauthorized” if the user is valid or a “403 Forbidden” on invalid subscribers.

```
SEQUENTIAL
adalbert
albert
aleksandra
alexander
```

Listing 5.3: The first line of a 65537 lines long injection file

The attack proved to be successful as it returned eight times “401 Unauthorized” indicating that this user exists. The resulting log file was analyzed and the eight existing identities were extracted for further (mis)use. This approach is a security risk as this concept makes it easy for attackers to harvest through the user database of the attacked infrastructure. It demasks existing identities which can then be used for further SPAM or identity-theft attacks.

5.2.3 Side Channel Attack on Response Time Distribution

If the existence of a subscriber is not detectable using enum-queries and the SIP proxy is responding with the same message, no matter, whether a user exists or not, the side channel attack might solve the challenge on finding existing users.

Side channel attacks are cryptoanalytical methods which analyze and misuse the (Hard- or Software-) implementation of an attacked system. Introduced by Kocher in [Kocher96], this method observes, e.g., the power consumption or the response time changes of an attacked system.

The side channel attack on the Response Time Distribution (RTD) is basing on the assumption, that REGISTER-requests belonging to an existing subscriber are processed differently inside the core. E.g., the user-database queries can differ when storing usage information and user statistics. Additionally storing logging information and the consumed time may be different between existing and non-existing subscribers.

These various latencies before receiving a “401 response” to a REGISTER request may be an indicator whether a user exists or not.

Practical verification

The RTD of sipgate.at is tested as a reference. The operators of the sipgate.at SIP-server configured the system to automatically generate user identities with numerical user parts, e.g., 6600198@sipgate.at or 6600227@sipgate.at. These two identities exist and are used as a positive reference.

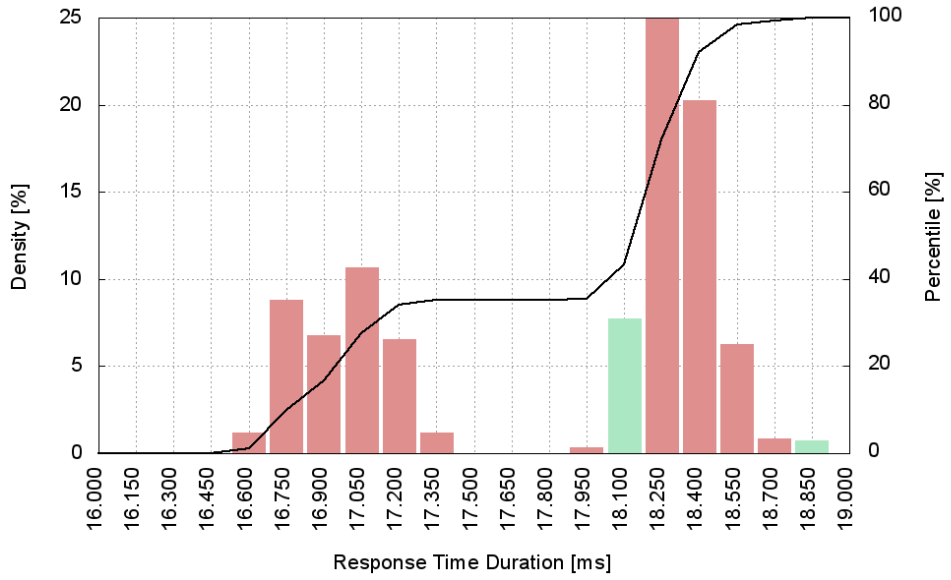


Figure 5.3: Two distinct normal distributed curves present the response-time distribution of REGISTER requests against sipgate.at. The RTD of both known identities are in bins within the right curve (green colored)

For running the test, the attacking node creates three times 1000 registration requests with various identities using the SIPp XML-script as presented in Listing 5.2 (here, the "a1.net"-string is replaced by "sipgate.at). The two known identities are also in this list as references.

After these three test runs, the results are analyzed, computing the average RTD of these 1000 identities. All responses exceeding an average RTD of 500 ms are removed (1.4%) as they are retransmitted requests. The other RTD are compared and presented in Figure 5.3. Here, two distinct, normal distributed curves are illustrated in the histogram. As both known identities are within the right curve, an attacker can suppose, that there is a possible RTD distinction between existing and non-existing subscribers.

Other reasons might have the same effect for this RTD-curves. Redundant systems with various round-trip times, distributed databases on several server are only two of the possible alternate reasons.

These results can therefore only be used as indicator and starting point for further attacks.

5.2.4 Summary SIP URI Harvesting

Several approaches support the attacker in fetching valid SIP-URIs for SPITting or DoS-attacks. One method uses the ENUM-approach, where a brute

force enumeration method helps to find valid identities. The second presented method is the less sophisticated: using a mixture between standards, which were defined for a closed, protected network together with the exposition of this system against the public Internet.

The third method is the least reliable technique: the response latency might differ between REGISTER-approaches when trying to register an existing and a non-existing subscriber. The same effect can also be caused by various other reasons and these effects and the conclusion are very vague.

5.3 Multisourceport DoS

One of the initial steps in protecting a SIP infrastructure is to block multiple requests from one socket within a specific time interval. With this action, most trivial flooding attacks can be blocked. On the other hand, a protecting system must be able to forward requests arriving from a large office LAN located behind a Network Address Translation (NAT)-gateway without blocking the connecting IP address. As presented and discussed in section 3.3.6, a recovering SIP infrastructure behind a NAT must not be blocked by a border protecting infrastructure.

The question is: how does a protecting system distinguish between a flooding attacker and a recovering, natted office LAN? The first should be blocked, whereas the second must not be blocked.

This challenge is solved by inspecting not just the sending IP-address, but also the source port of the UDP packet. RFC 3489 “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)” [Rosenberg03] presents four variations of NAT. The most common type of NAT is the “symmetric NAT”, where all requests from the same internal socket to a specific foreign IP address and port combination are mapped to the same external socket. If the same internal host sends a packet from the same socket, but to another address, another external socket is used. Only an external host, which already received a packet from an internal host can send a UDP packet back to the internal host (RFC 3489 [Rosenberg03]).

To detect if a register-request is sent from a host behind a NAT, the protecting proxy can apply three criteria:

1. On transport layer: do the requests originate from the same IP but from different ports?
2. On application layer:
   (a) do the IP addresses inside the SIP-request (Via- and Contact-header as well as the addresses inside the SDP payload) differ from the sending IP address?

(b) Do identities inside the SIP-requests vary?

These three criteria are the information a protecting node can use to decide, whether a request arrives from a natted LAN or a probable flooding attacker. If all these criteria apply, the system must assume a request from a natted LAN.

To start a successful DoS-attack against the infrastructure, these three criteria must be emulated by the attacker.

Successful DoS attack

Like in subsection 5.2.2, the REGISTER-message is the best fitting request for a DoS-attack: each new REGISTER-request initiates a cascade of SIP messages inside of complex IMS or SIP cores. Figure 5.1 presents the messages caused by an arriving SIP REGISTER request: The request of the first transaction initiates six (with a SBC eight) SIP requests and two Diameter [Fajardo12]-requests and passes four core-nodes: the P-CSCF, the I-CSCF, the S-CSCF and finally the Home Subscriber Server (HSS). On each of these nodes, the REGISTER requests are parsed, modified, cached and forwarded to the next hop.

Compared to the REGISTER requests, SIP INVITE requests are blocked at the SBC (if available) or at the P-CSCF when the calling party is not registered yet. INVITE requests from foreign domains are in some configurations blocked or their maximum number of requests per second is reduced to a non-critical level. REGISTER-requests are categorized as requests from unregistered *customers* and are hence accepted at a higher rate. When considering REGISTER requests as message for a DoS-attack, the next step is to continuously modify the sending port. The tool SIPp offers a parameter (`-t un`), where each new transaction is sent from another port.

The successful attack uses the XML-infile from Listing 5.2. SIPp is called as presented in Listing 5.4.

```
sipp -i 128.131.88.45 -sf register.xml -l 1000 -inf
   infile.csv -max_retrans 1 -r 400 -t un -m 10000 biloxi
   .com
```

Listing 5.4: SIPp command line parameters to start a register DoS attack

The relevant parameter are

- `-l 1000`: creates 1000 parallel calls
- `-r 400`: a rate of 400 requests per second
- `-t un`: with varying sending ports
- `biloxi.com`: to the target `biloxi.com`

```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
 400.0(0 ms)/1.000s             5.64 s         2256  80.75.55.xxx:5060(UDP)

  251 new calls during 0.626 s period    0 ms scheduler resolution
  602 calls (limit 1000)                 Peak was 603 calls, after 2 s
  0 Running, 2257 Paused, 744 Woken up
  9 dead call msg (discarded)            0 out-of-call msg (discarded)
  605 open sockets

                                Messages  Retrans   Timeout   Unexpected-Msg
    REGISTER ---------->        2256      1946      1358
         100 <----------        0         0         0         0
         500 <----------        276       0         0         0
         400 <----------        17        0         0         0
         403 <----------        0         0         0         0
         503 <----------        0         0         0         0
         401 <----------        3         0         0         0
------------------------------ Test Terminated --------------------------------
```

Figure 5.4: This Register-flooding DoS-attack with 2256 requests within 5,56 seconds results in 60% timeouts (the number of retransmissions was set to '1'), 12% `SIP/2.0 500 Cx Unable To Comply`- and 0,7% `SIP/2.0 400 Invalid From`-responses. The latter were correctly generated by the SBC due to invalid datasets in the injection table. There were no `403 Forbidden` responses at all.

Starting the attack results in a breakdown of the tested system within a few seconds. After 2256 sent REGISTER-request, the attacked system was not able to create one single `403 Forbidden` response. Instead, the overloaded system responded immediately with multiple `SIP/2.0 500 Cx Unable To Comply` responses. A screen shot of the executed SIPp-script with the results is depicted in Figure 5.4.

With the source-port variation DoS-attack, a single COTS computer is able to break a carrier-grade infrastructure. The error-response "500 Cx unable to comply" is an indicator, that the overloaded node is not the SBC. Instead, the Cx-interface is the link between the I- or S-CSCF and the HSS which is an indicator, that one of these components are congested. This in turn is an indication, that the SBC was not able perform its tasks (as defined in [Hautakorpi10]) to defend the DoS attack.

Chapter 7 will present standardization approaches to communicate such overload states to adapt the blocking rate of the protecting SBC. In addition chapter 10 will present approaches to build a SBC which is able to block parts of these DoS-attacks without additional, core internal communication.

5.4 Provider Specific DoS-attacks

Successful attacks against a victim's infrastructure are supported by detailed knowledge of the attacked architecture. Information can be gained using port scans in combination with OS-detecting on Internet- and Transport-layer, but

also on application layer by inspecting the header of the SIP-response. As an example, the error-responses `SIP/2.0 500 Cx Unable To Comply` in combination with an Internet recherché indicates, that this system is operated by an IMS core.

Other points of attacks can be found using the public available technical description of the attacked service. The example presented in this section is an attack against a mobile phone provider. The VoIP-service of this provider (provider 'B') is sold as a value added service like offered by A1 Telekom Austria, SwissCom or Vipnet.hr.

In specific, the following services are available in the attacked system of provider 'B':

- 'B'-customer can call other 'B'-customers using SIP or via a media-gateway to the CS-domain, SIP to SIP (SIP2SIP) or SIP2CS.
- 'B'-customer (using their SIP-device) call customers of other providers using the media-gateway too (SIP2CS)
- SIP-enabled 'B'-customers can be reached by other callers either via their mobile-phone or using their SIP-phone (Circuit Switched to SIP (CS2SIP))
- VoIP-users can customers of provider 'B' only when their SIP-device is registered. Foreign users cannot initiate calls using the media-gateway to the mobile phone of a 'B'-customer.
- If a 'B'-customer is not registered, all incoming SIP-calls are forwarded to the voice mail service.

The last item in this list is a potential attack vector: Calls from foreign (and therefore unauthorized) domains are automatically forwarded to the voice-mail service of provider 'B', which is located in the CS-domain. Hence, even unauthorized and unknown SIP caller can create sessions over the SIP2CS media gateway. As this gateway is an essential component for the operator, a possible oversaturation of this SIP2CS gateway by unrecognized callers will be a major security issue.

A SIP2CS media gateway is also a limited infrastructure with a maximum number of parallel calls. Additionally, the voice mail service of one customer is also limited to a maximum number of minutes or messages. Here, two possible attack options are possible: "Media Gateway saturation and "Voice Mail/Mailbox saturation.

The approach is similar on both attacks, the only difference is the focusing on one called party with a limited number of calls when attacking the voice mail of a chosen victim.

5.4.1 Media-Gateway Saturation attack

Calls to existing and non-existing subscriber are handled differently on provider 'B': if a subscriber is unknown, the system rejects a call attempt with a `SIP/2.0 404 Called User Unknown`-response. Calls to an existing but unregistered subscriber are forwarded to the voice mail as presented in the introducing section. As the attacker wants to saturate a media-proxy, a media-stream is needed. Hence, call attempts to non-existing subscriber have no influence on the attack target.

A precondition to this saturation attack is a list of existing subscribers. Section 5.2 presents techniques, on how to build such a list. The count of existing users which are required to execute the attack depends on the policy, how many calls can be placed in parallel to each of these users over the media gateway.

```
sipp -i 128.131.88.45 -sf invite.xml -l 1000 -inf infile.
   csv -max_retrans 1 -r 40 -t un -m 10000 B
```

Listing 5.5: Command line parameters to start an attack against the media proxy are similar to Listing 5.4

The `infile.csv`-file showed in Listing 5.5 is filled with known existing subscribers. The file `invite.xml` contains the call flow of the SIP-dialog used for the attack.

Listing 5.6 presents the first INVITE-request of the attack, where the main aspect is the distinct domain of the calling URI. If this domain is similar to the attacked identity, this call is interpreted as in-domain call and the SIP-proxy attempts to authenticate the caller. Using a different caller domain, this call is a foreign-domain call which cannot be authenticated (as already discussed in the 'attack vectors'-section 4.1).

```
INVITE sip:[field2]@B SIP/2.0
Via: SIP/2.0/[transport] [local_ip];branch=[branch]
CSeq: 1 INVITE
To: [field2] <sip:[field2]@B>
Content-Type: application/sdp
From: sip:[field0][call_number]@iptel.org;user=phone;tag=
  [call_number]
Call-ID: [call_id]
Max-Forwards: 70
Subject: sip:[field0][call_number]
Content-Length: [len]
User-Agent: kphone/4.2
Contact: sip:[field0][call_number]@[local_ip]:[local_port
```

```
  ];transport=[transport]

(sdp removed)
```

Listing 5.6: Initial SIP INVITE request for creating media-gateway calls in domain 'B' without authentication. The caller pretends to be customer of "iptel.org"

This attack is executed similar to the REGISTER-DoS-attack but with different result: fewer requests are needed to oversaturate the media proxy. The number of parallel calls depends on the length of each call. The longer the voice-message is, the fewer calls are needed to create the maximum number of parallel calls required to oversaturate the media proxy.

5.5 Conclusions

The presented examples show how to acquire a list of potential SPIT victim subscribers and how to misuse common services like REGISTER-attempts or provider-specific services like the CS-hosted voice Mailbox in example 5.4.

All attacks illustrate the weakness of SIP against DoS-attacks. The NAT problem of the first example is a problem-by-design as the operator has no real options to handle these requests other than presented. Alternative options to prevent these kinds of attacks will be discussed in the solutions-part.

Part II

Solution Space

Chapter 6

Anomaly Detection

6.1 Introduction

The high complexity of telecommunication solutions in future PS networks are a challenge for detecting and communicating anomalies. Beside explicit overload detection by monitoring system- and infrastructure-KPIs, the implicit congestion monitoring is essential for operating complex IMS infrastructures: the distributed deployment, the mix of multiple vendors and the geo-redundancy of components increase the challenge to aggregate these information for congestion analysis.

This section introduces the "PARIS" system, a solution for implicit detection of anomalies in complex black boxes of modern VoIP solutions. The use of PARIS in the productive A1 Telekom Austria (A1TA) infrastructure demonstrates the success of this approach.

The focus of this section is on measurement techniques, analyses and communication methods of this system, as well as representative results on long term monitoring and alerting.

6.2 Response Delay Monitoring using "Performance, Availability and Reliability Information System (PARIS)"

The continuous monitoring of VoIP performance parameters over an extended time interval is an extension to short termed evaluations and monitoring. It illustrates the behavior of the core infrastructure correlated over the time and highlights peaks like busy-hours or the relation between weekday-load and weekend-load. With observations collected by PARIS it is possible to detect infrastructure anomalies when comparing current measurement results with long-term results. Together with the long-term observations, PARIS

implements near-real-time alerting on unexpected monitoring results. The toolkit compares current results against monitored results of the last test cycle and initiates alerts using the Icinga[1] alerting toolkit

PARIS implements longterm monitoring from an end-user perspective by creating automated test calls at predefined time intervals. This time span varies between 10 and 30 minutes and creates SIP2SIP- and SIP2CS-voice calls. Additionally, PARIS is capable to monitor the KPIs of server based presence.

Test concept

The End-to-End monitoring uses one single Linux PC running two independent instances of SIPp to simulate both UAC and UAS as illustrated in Figure 6.1. Each of these two SIPp instances and the tools to analyze the received signaling and media are wrapped by a script.

6.2.1 Architecture

Each test is initiated by a time triggered controller. Its task is to concurrently start the `imstest.sh` scripts for the UAC and optionally for the UAS against one *System Under Test (SUT)*. The structure is depicted in Figure 6.1.

After call generation, the script aggregates and analyses the results, creates alerts and stores statistics for trend monitoring. The results are presented for visual analysis using diagrams (as shown exemplarily in Figure 6.2) for signaling delay and media statistics.

The alerting focus is set on Icinga, a powerful and popular IT-infrastructure monitoring tool. Icinga is executed either locally on the PARIS node, or – when using multiple computers for PARIS-monitoring – by using the *Nagios Remote Plugin Executor (nrpe)*-plugin. Due the the modular architecture of Icinga as alerting application, it is possible to extend and to add more plugins with minor effort.

Implementation description

Each UA instance is controlled by a time triggered shell-script (`imstest.sh`). Both instances (UAC and UAS) follow an identical sequence (depicted in Figure 6.1), the UAC-instance being extended by additional analytical features.

Creating the Calls

As first step `imstest.sh` initializes a semaphore. This semaphore is the connects UAC and UAS and controls the synchronization of the UA-instances.

[1] Icinga – Open Source Monitoring, `http://www.icinga.org`

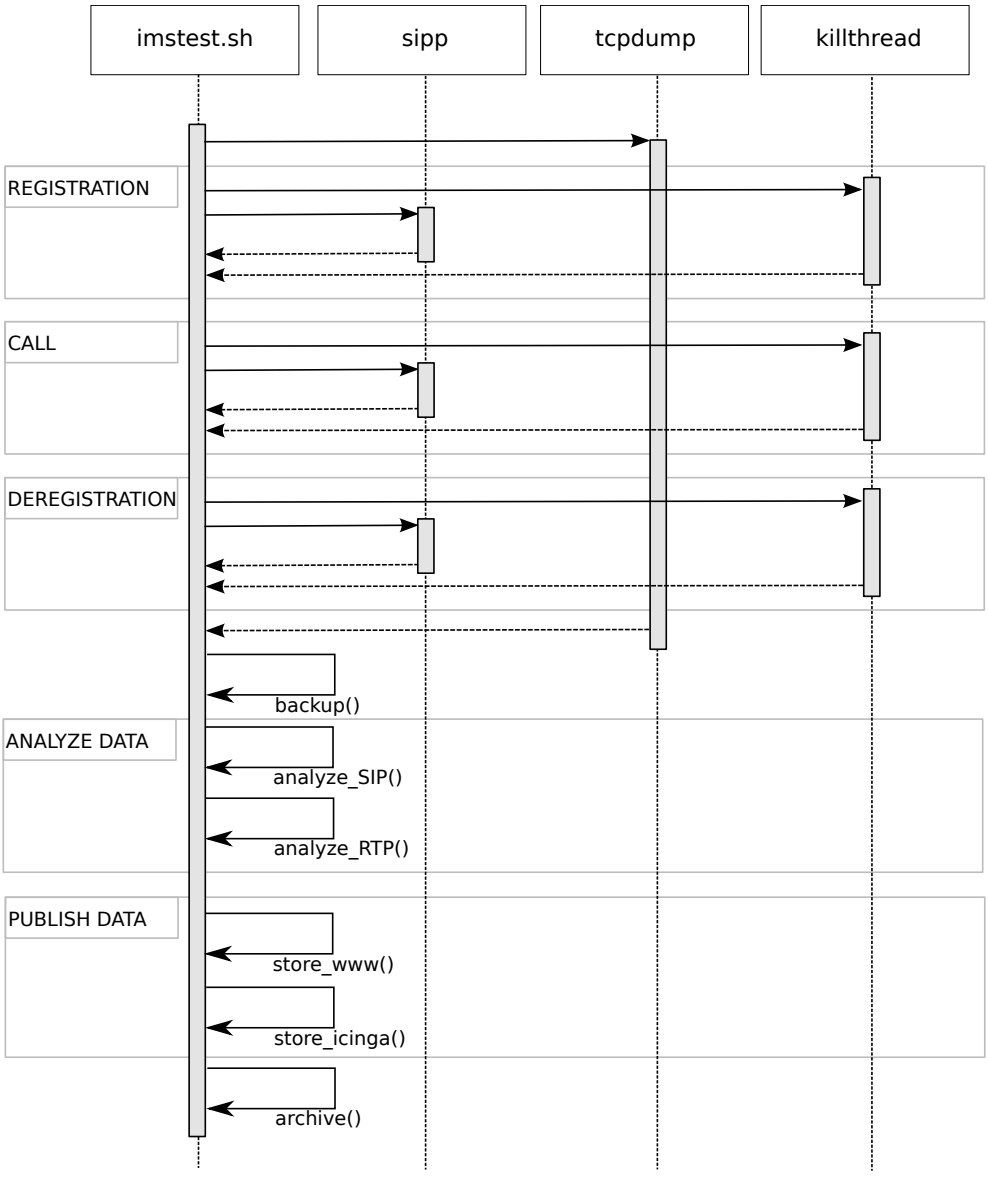


Figure 6.1: After registration and call generation, the generated results are analyzed and forwarded to further processing and alerting tools

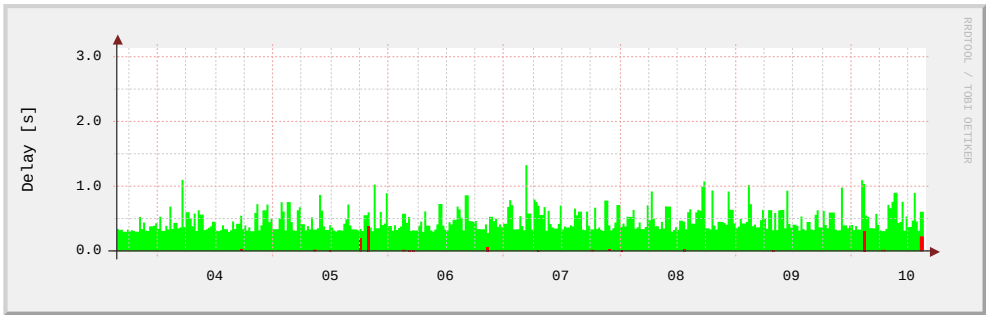


Figure 6.2: Sample diagram for Session Setup Delay in seconds from UAC-perspective during November 4^{th} to 10^{th} 2012. Red bars mark failed call-attempts

The next step is to calculate a random number, to copy the sipp-binary three times to a temporary directory and renaming them `sipp_`*`role`*`_`*`random`* (where `role` is either `reg` for registration, `uax` for either UAC or UAS role or `unreg` for de-registration). It is important to create unique names for all three binaries to terminate each of them by executing the `killall`-command. This `killall`-command is useful for terminating non responding SIPp instances to avoid zombie-processes. When acting in UAC-mode, the script is starting the network packet recording application `tcpdump`. This application is active during the entire test-run to record both signaling and media.

Afterwards, the integer value of the semaphore-file is increased twice by the script `countup.sh`, which is executed in `imstest.sh`. This semaphore is later decreased once after the registration of the UAC and once after the registration of the UAS. This semaphore ensures, that both UA processes are synchronized after registration and before creating the SIP call.

More accurate, the UAS-process must already be running before the UAC begins to send the first INVITE request to avoid retransmissions.

Registering the client

The next step starts a Perl script in another thread. This script waits as background process for a predefined amount of seconds (default: 60s) and then terminates all running SIPp-registering instances. This approach is the fall back mechanism to terminate crashed or locked SIPp instances. Typically the registration process finishes within less than one second and SIPp crashes are observed only in rare cases. After registration, both UAC and UAS are lined up and synchronized.

Creating the Call

Similar to the registration process, a background thread to kill an potentially non-responding SIPp process is started.

Afterwards, the two SIPp instances simulating UAC and UAS are started – first the UAS and after a delay of one second, the UAC. The behavior of the UAs can be controlled by changing the SIPp-XML scenario-filename in the configuration files. When modifying the XML-scenario-file, the UA can also act as a MESSAGE or presence sending client.

After finishing the call, the client is unregistered similar to the registration process and the analyzing steps start.

Backing up the log-files

SIPp as SIP client creates several log-files. In the presented configuration, the system creates four files, which are copied to a temporary directory for further analysis and subsequent backup. The first file contains the SIP-messages as tab-separated list with one request or response per line, containing the essential information like time of transmission (in μs) and the request/response types. The next file is the extended version of the first log file: all SIP-messages with all their headers are stored in this file. The third file contains errors in transmissions, like failed connections due to port failure. The last file contains the screen-output of SIPp.

Each of the three steps (registration, call, unregistration) creates this collection of files.

Analyzing the Results

After copying the log-files to the temporary working directory, a tool set of Perl- and Bash-scripts are executed to analyze the signaling and the media data.

Signaling

RFC 6076 [Malas11] defines the SIP end-to-end performance metrics. The PARIS tool calculates the KPIs *Successful Session Setup - Session Request Delay (SSS SRD)* (RFC 6076, 4.3.1) respectively *Session Establishment Effectiveness Ratio (SEER)* (RFC 6076, 4.7) for registration, call-setup and deregistration.

Media

After the SIP signaling, the recorded media is analyzed to compute KPIs as, e.g., the *Perceptual Evaluation of Speech Quality (PESQ)*, the jitter, packet-loss and end-to-end latency of the received media stream.

The PESQ value is ranging from 0 to maximum 4.5, where 4.5 is perfect voice quality and 0 is failed. As the audio stream in this test-tool is transcoded

to G.711 and this transcoding is decreasing the overall voice quality, the maximum theoretically possible PESQ-value of the test-sample is 4.251.

Populating the results to several sensors

The next step is to forward the measured data to

- the web interface containing graphs over various time spans as also sound samples and signaling logs
- Icinga status files, which will be read both by a local Icinga installation and a remote Icinga accessed by cooperation partners
- the local logging archive.

Web interface

The PARIS web interface is implemented as a HTML/ PHP website running on an Apache web server on a Secure Socket Layer (SSL) secured website `https://paris.ibk.tuwien.ac.at`. Dynamically generated images on the web site are generated using the RRDtool[2].

The database for the RRDtool is a multilevel round-robin-database, which stores data for a predefined timespan (in this case two weeks) in a round-robin manner. If the age of a data point in the table exceeds the predefined timespan, the data point will be removed. Before this data point is dropped, the *rrdtool* calculates an average value with data points of similar age. This average value is stored in another table with a wider granularity for three more months. The third table with data for one year is filled in the same way, using the values of the three-months table. The results of this lossy data compression are presented in Figure 6.3.

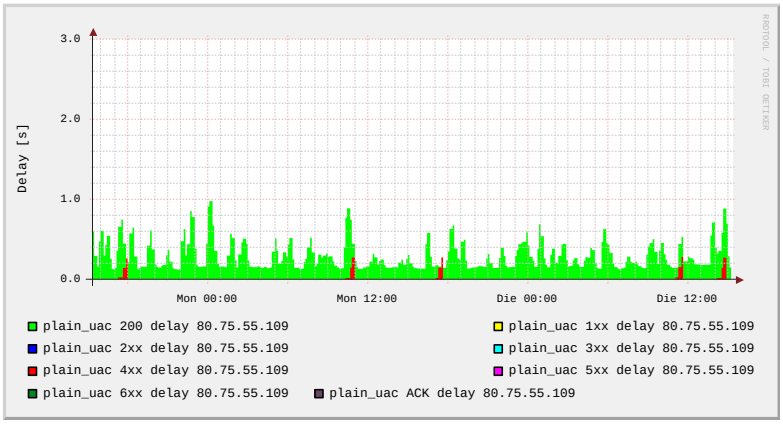
Additionally, the SIPp log files are linked to the website. If a call succeeded, only short, tab-delimited logs are stored, whereas if a call failed, the extensive log files are saved for more detailed fault diagnostics.

Icinga Monitoring

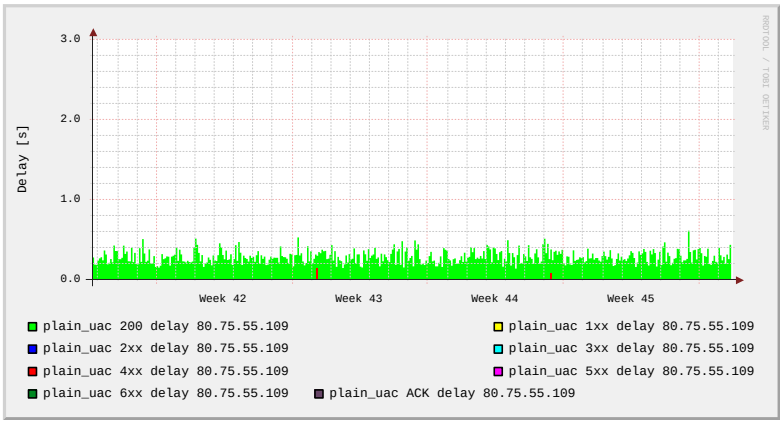
The Icinga Monitoring Toolkit is a Open Source computer and network monitoring tool. It plugin-based architecture is expandable with further modules and plugins. This modular design supports integration of PARIS monitoring alerts.

In the central PARIS-script, success and KPIs of each test are stored to distinct state files. The self-developed Icinga plug in reads the content of status-files and reports the state (either *OK*, *CRITICAL* or *WARNING*) to the Icinga core component or the "Nagios remote execution plug in". Depending

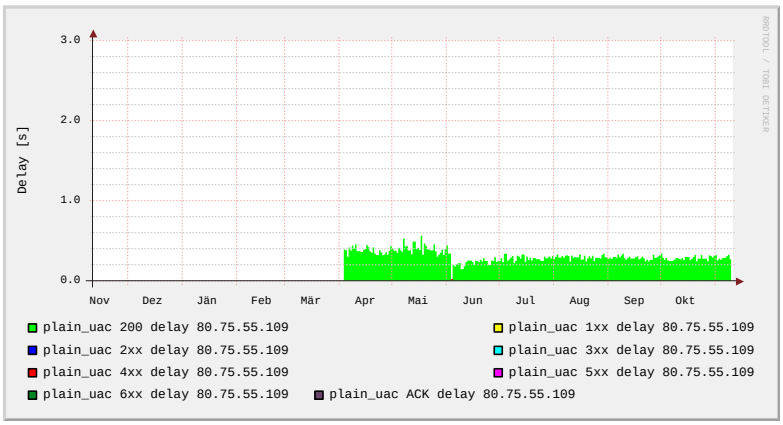
[2]RRDTools: http://oss.oetiker.ch/rrdtool/



(a) 48 Hours Statistics



(b) Four Weeks Statistics



(c) One Year Statistics

Figure 6.3: A comparison between representations of the session setup delay created with the rrdtool

on these results, Icinga creates alert messages either by Email or by *Simple Network Management Protocol (SNMP)* [Case02] traps.

Filing the results

As last step of a test run, the log files are all moved to an archive data directory and the remaining temporary files are removed.

6.2.2 Findings and learnings as input for congestion control mechanisms

The presented black-box monitoring "PARIS" proved to be an appropriate tool to supervise availability and Quality-of-Service (QoS) of the tested infrastructures. The following providers and services are monitored.

- *SIP2SIP call*, *SIP2CS* and *presence* for A1TA productive and testing infrastructure
- *SIP2SIP call* for iptel.org
- *SIP2SIP call* for sipgate.at[3]
- *SIP2SIP call* for sipcall.at[4]
- *SIP2SIP call* for the IMS testbed of Forschungszentrum Telekommunikation Wien (FTW) [5]
- *SIP2SIP call* for the IMS testbed of the Institute of Telecommunications (ITC)

The tests against iptel, sipgate and sipcall are executed as reference sources to isolate/detect local and remote network or component failures.

Availability

The periodic test calls are started at a time-interval between ten minutes for important services, and twenty minutes for comparative tests. The alerting infrastructure *Icinga* is configured to record at least three failed tests before raising an alert. The inter-test interval should not be decreased as each test lasts two minutes. In these two minutes, starting other tests must be avoided to circumvent mutual interference.

The log files can be used to quantify the SIP performance metrics as defined in RFC 6076. This standardized view on delays and availabilities help to compare services with contractual guaranteed service levels and request delays.

[3]sipgate GmbH (V.i.S.d.P), Gladbacher Strae 74, D 40219 Düsseldorf
[4]Backbone Solutions AG, Chaltenbodenstrasse 4b, CH 8834 Schindellegi
[5]Forschungszentrum Telekommunikation Wien, Donau-City-Str. 1, A 1220 Wien

The SSS SRD (RFC 6076, 4.3.1) can be used for calculating the call setup delay. It is defined as

$$\text{SSS SRD [ms]} = t_{\downarrow 180}[ms] - t_{\uparrow \text{INVITE}}[ms] \tag{6.1}$$

where $t_{\downarrow}$ is the arriving and $t_{\uparrow}$ the sending time of the message. Another important metric is the SEER calculating the availability of the SUT with e.g., n_{200} is the number of received "200 OK" messages.

$$\text{SEER} = \frac{n_{(200,\ 480,\ 486,\ 600)}}{n_{(\text{INVITE})} - n_{(3\text{XX},\ 401,\ 402,\ 407)}} \tag{6.2}$$

Starting with early year 2009, the SSS SRD and the SEER for the VoIP providers are recorded and the results are presented in Table 6.1.The test-results show, that only one provider is able to offer a level of availability of more than 99%. These values, even the result of the best performing SipCall, are below from carrier-grade availabilities of 99.99% or better.

One of the reasons for this result is the complexity of the infrastructure of some providers: A1TA, FTW and ITC are operating IMS-cores, where several components are combined. The high number of components decrease the availability A according to the equation $A = \prod_{i=1}^{n} A_i$.

As the IMS infrastructures are still under development or extension, the second reason for the presented low availability are downtimes or reboots of components. These are affecting the SSS SRD as long as the perimeter SIP infrastructure is reachable and responding to SIP requests.

Provider	$\tilde{x}_{\textbf{SSS SRD}}$ [s]	$\sigma_{\textbf{SSS SRD}}$ [s]	**SEER** [%]	$n_{\textbf{tests}}$
A1TA prod. core	0.336	0.368	96.64	35263
iptel.org	0.028	0.013	98.12	10289
sipcall.at	0.306	0.602	99.08	14678
sipgate.at	0.051	0.071	95.22	11989
A1TA ref. core	0.433	0.351	93.46	37715
ibk	0.011	0.015	8.44	1100
ftw	0.145	0.382	93.96	18313

Table 6.1: Comparison of successful setup delay and availability for selected VoIP providers. The last three SUTs are test-systems

SSS SRD - Distribution

The SSS SRD is an indicator for reactiveness and stability of the tested system. Consistent SRD values over the time-of-day represent a system, which is capable to handle requests independent to the offered load ("busy hour").

Converting the SRD values into a graphical histogram, the differences in response time distribution are significant as illustrated by Figure 6.4, Figure 6.5 and Figure 6.6.

The SIP providers IPtel depicted in Figure 6.4[6], a) and SipGate depicted in Figure 6.4, b) offer short SSS SRDs caused by their reduced number of components. For instance, SipGate is powered by the Open Source software OpenSER[7] as SIP proxy and MySQL[8] as user-database. IPtel is operated by OpenSER and uses probably either MySQL or PostgreSQL[9].

SipCall's results show strong varying response times as depicted in Figure 6.4, c). The `User-Agent` header of the SIP responses show OpenSER acting as SBC and Sippy [10] as SIP- and media-proxy.

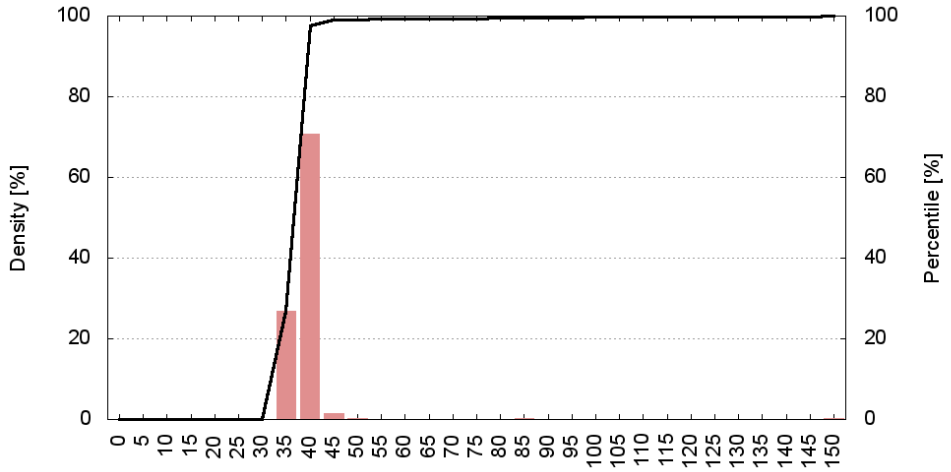
Comparing the results of the other four systems – all IMS – presents different behaviors: The largest tested IMS infrastructure and the only one, which is used at productive level is the A1TA productive core (Figure 6.5, a): The graphic presents a standard distribution of the SSS SRD-values close to 300 ms (60% of all measurements) and almost 30% with a SSS SRD of larger than 500 ms. These significantly higher values are results of one or more SIP retransmissions. Results of the A1TA reference system black box tests (depicted in Figure 6.5, b)) have a structure, which is similar to the one of productive systems results. Like in the productive system, the complexity of the IMS infrastructure originates the same distribution profile, but shifted to the right. Regarding the same location and the same connectivity of the test-client, these shifted results are a potential indicator of a lower performing system. 25% of the response-times of both tested A1TA IMS infrastructures are above 500 ms (RFC 3261 T1 retransmission timer) and indicate retransmitted requests.

The FTW IMS core as a research infrastructure was modified often during the observation interval and this also affected the SSS SRD. This results in a flat response time distribution histogram shown in figure 6.6 b).

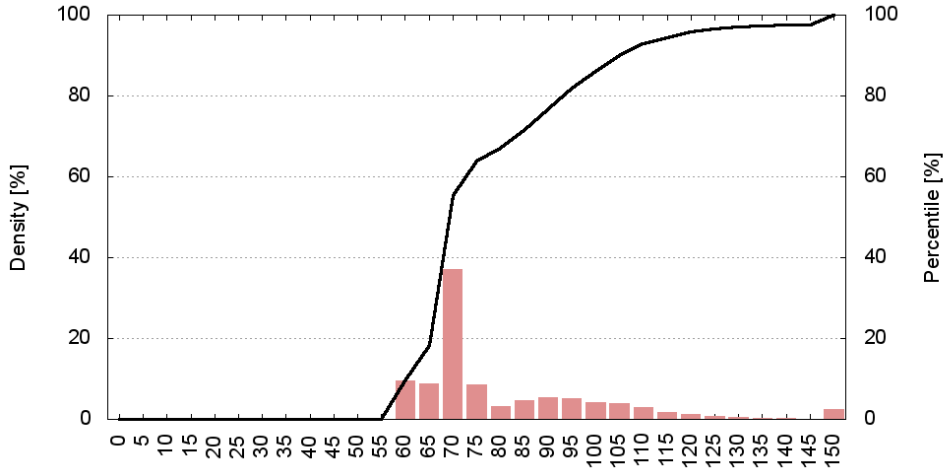
The last inspected system, the IMS core of the ITC mostly acted under low load and low usage. This fact, and the direct geographical neighborhood to the monitoring infrastructure results in low response times with a median of 11 ms as depicted in figure fig:refdist a).

It is important to note, that the SSS SRD of a carrier-grade IMS infrastructure is by one order of magnitude slower than a plain VoIP infrastructure.

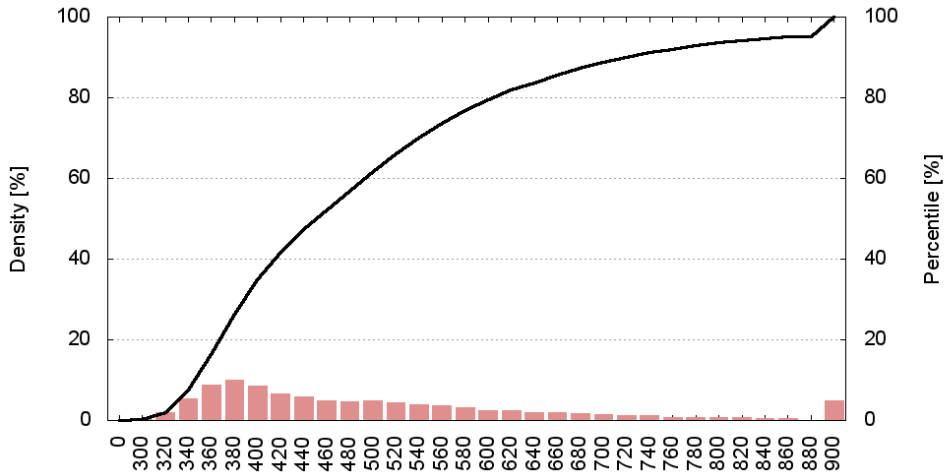
[6] FONTS
[7] Kamailio: http://www.kamailio.org/
[8] MySQL: http://www.mysql.de/
[9] postgresql: http://www.postgresql.org/
[10] sippy RTPProxy http://www.rtpproxy.org/



(a) IPtel Successful Session Setup Session Request Delay (SSS SRD) [ms], n=12011

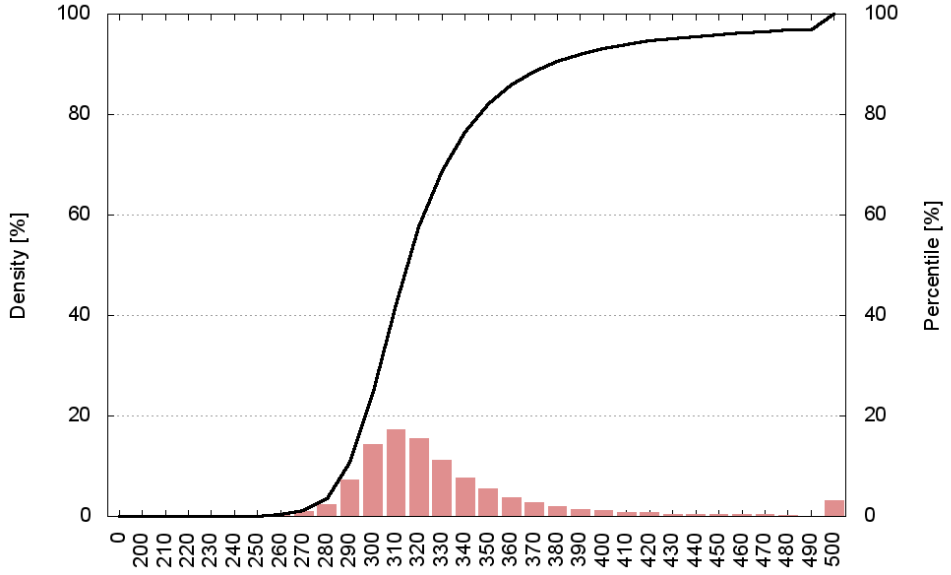


(b) SipGate SSS SRD [ms], n=9904

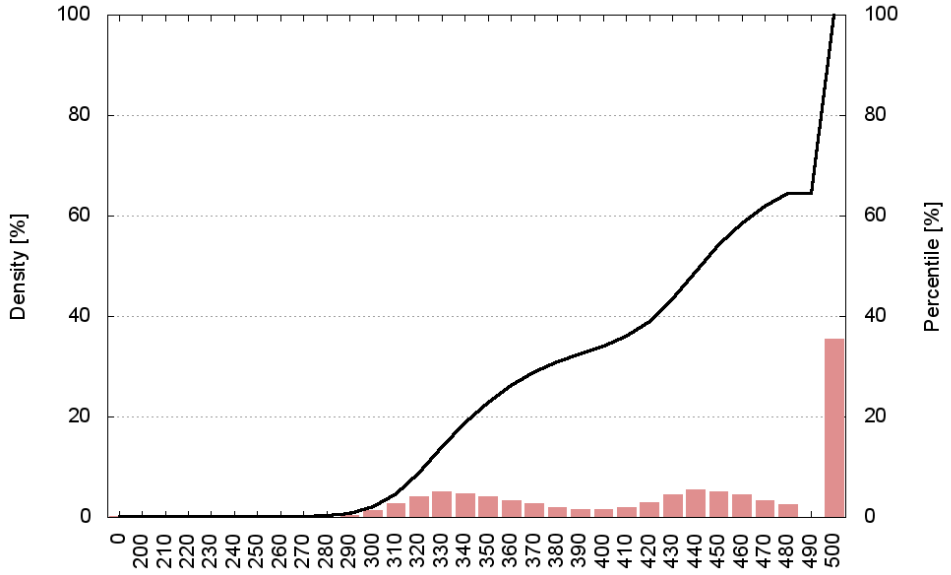


(c) SipCall SSS SRD [ms], n=10109

Figure 6.4: SSS SRD distribution of the three SIP provider

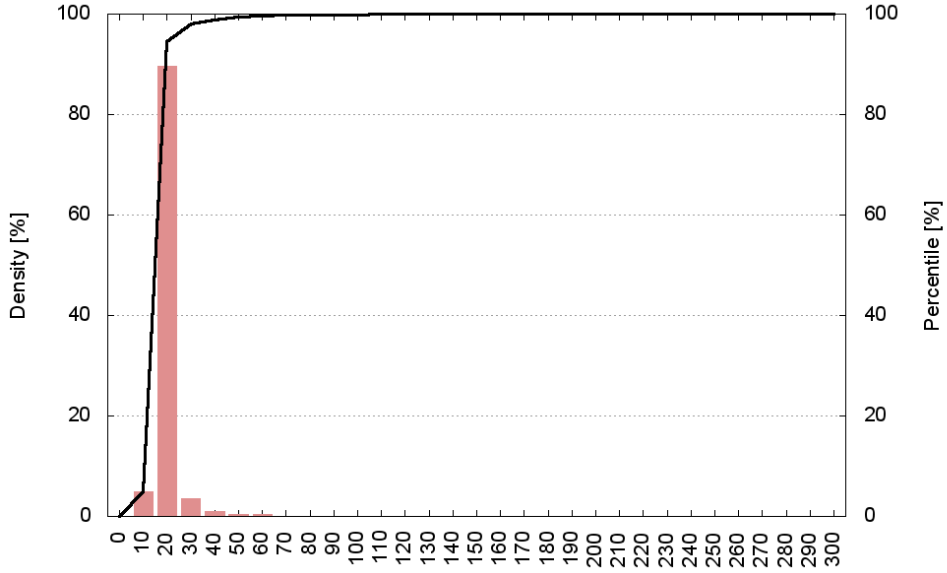


(a) A1TA productive IMS core SSS SRD [ms], n=49060

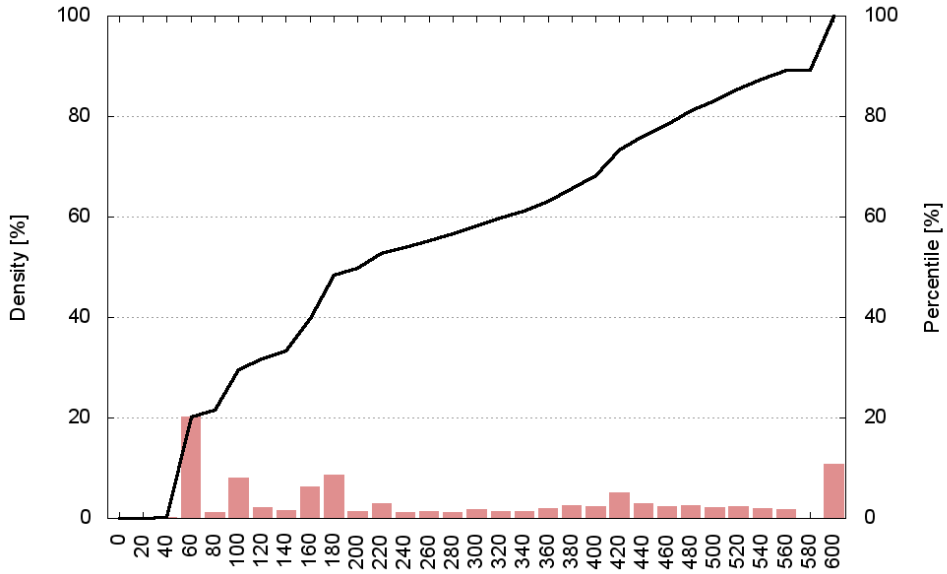


(b) A1TA reference IMS core SSS SRD [ms], n=47417

Figure 6.5: The A1TA productive and reference IMS cores. The reference IMS core has higher response times than the productive system



(a) ITC IMS core SSS SRD [ms], n=2095



(b) FTW IMS core SSS SRD [ms], n=7951

Figure 6.6: The two OpenIMS research IMS cores of ITC and ftw

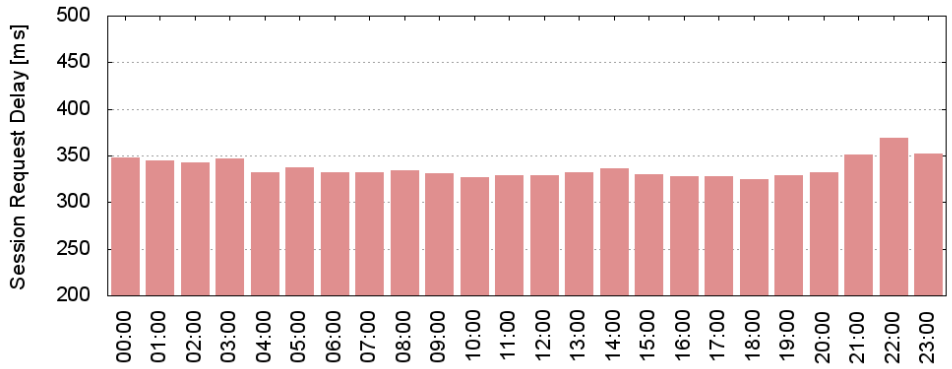


Figure 6.7: 24 hours median SSS SRD of the A1TA productive IMS infrastructure

6.2.3 Using the Results for Congestion Alerts

Test results which differ from the expected longterm values are an indication that the observed system is potentially overloaded. Aggregated with explicit monitoring data from inside the observed infrastructure,this data can be used as input parameters for congestion-control mechanisms.

Signaling plane

After completing a test-call, measurement results are compared against the expected result for this day-of-week and hour. If the SSS SRD is differing by more than a predefined factor, this is interpreted as an alert case.

For the following diagrams and analyses, the SRDs of all weekdays (Monday to Thursday) are combined for each of the tests.

Figure 6.7 depicts for the A1TA IMS core, that the SSS SRD is constant over 24 hours with minimal variations between 330 and 380 ms. This is the default behavior of systems with sufficient performance and adequate connectivity. Compared to productive A1TA, the plain SIP provider SipCall behaves different. Figure 6.8 illustrates a strong increase at 22:30 by factor 1.5 lasting until 4:30 in the morning. The lowest median setup value at 19:00 and the highest such value at 23:00 are both acceptable values, but the periodic daily change of SSS SRD is an indicator for a problem. The reason of this high latency during nighttime is unclear and can be caused by performance problems due to e.g., backup-jobs, M2M synchronization or maintenance-jobs. The hours with the highest latencies in the SIPCall environment are during nighttime and not during central European busy hours. As these systems are all black boxes for the monitoring infrastructure, no certain conclusions can be drawn from this behavior.

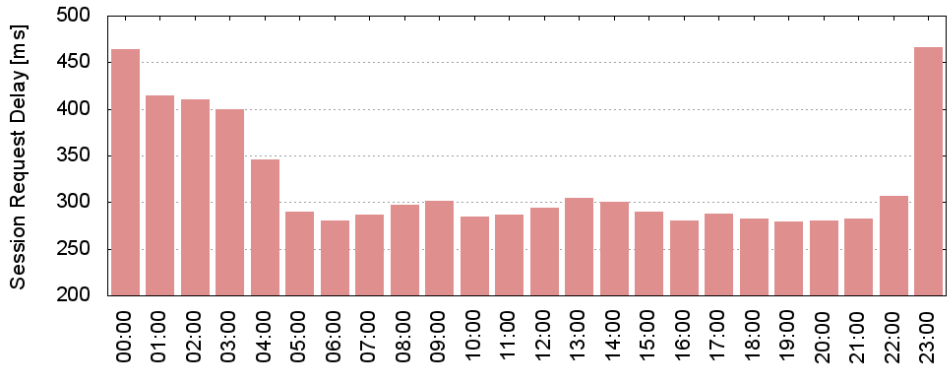


Figure 6.8: The 24 h median SSS SRD of SipCall's VoIP infrastructure

Media plane

In basic SIP scenarios as introduced in Figure 2.1, RTP-data is sent directly end-to-end, from UAC to UAS and vice versa without any intermediate media proxies. In the PARIS-solution, both UAC and UAS are located on the same host and hence media is sent over the local loop back device. The resulting jitter and latency are introduced by local processing delays and are not significant for the black box tests.

Two of the monitored systems, A1TA productive- and A1TA reference-system, use media proxies in their infrastructure. In this case, RTP data is transmitted from the UAC to the media proxy and back to the UAS. The timespan between sending and receiving the RTP packets results in latency and jitter information which depict the current system and connection load and can therefore be used for PARIS/Icinga alerting. 2000 RTP packets per call and direction are sent to the UAs so each direction creates 2000 latency bins. The latency values can be used for jitter and packet loss calculation (RFC 3550, 6.4.1 [Schulzrinne03]). The high number of bins per call and the high dependability of latency values with respect to tested system load recommends latency and jitter as reliable sensors for congestion monitoring.

Figure 6.9 (A1TA productive core) shows that jitter results for busy hours and nighttime differ by a factor of 4.5.

The results provide a significant reference profile of the RTP infrastructure's main busy hours. Congestion control and alert systems can compare current measured values against the acquired reference profiles.

6.2.4 Summary

The long term monitoring toolkit PARIS is a consequent extension to explicit and short-time alerting. Due to the extendability of the central script, this

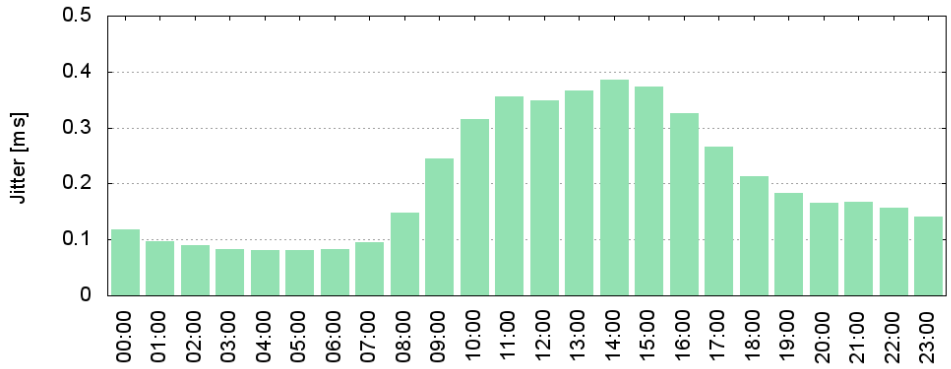


Figure 6.9: The median jitter over 24 h on weekdays on A1TA productive core.

system is able to alert on system failures on the one hand and present the long term performance trends on the other hand.

With the ability to monitor both SIP signaling and RTP media, the Session Setup- and Registration Delays and their success rate can be measured as well as RTP packet loss and jitter as an indicator for decreased voice quality.

Knowledge of the monitored infrastructure's default behavior, this know-how is an input for anomaly detection and alerting using the congestion control infrastructures.

Chapter 7

Overload Communication and Congestion Control

Science, standardization and industry are aware of that reliable operation of future VoIP enabled telecommunication networks will be endangered by massive overload situations. As a consequence, both the IETF and the ETSI have started counteracting standardization activities. The IETF defines SIP extensions to signal overload situations in-line the standard SIP communication, whereas the ETSI creates a new explicit signaling path in IMS architecture for (live-)protocol independent overload communication.
This chapter presents the two architectures and the various types of overload communication.

7.1 Basic components

Both approaches define the basic components for organizing the overload control mechanisms.

- **SIP processor** (IETF), *Communication Application* (ETSI): the component processing the specific protocol messages (mostly SIP, but ETSI defines the Communication Application more generally). This component is the essential part in the communication infrastructure which needs to be protected from overload.
- **Monitor** (IETF), *Control Adaptor (CA)* (ETSI): this component measures the current system state and announces this state to the Control Function.
- **Control Function (CF)** (IETF), *CA* and *Control Distribution (CD)* (ETSI): implements the overload control algorithm. It analyses the load

samples and decides, whether an overload state is occurring or not. Depending on this result, it communicates the throttling decision to the *Actuator*. In the ETSI standards, this function is shared between CA and CD.

- **Actuator** (IETF), *Restrictor Manager (RM)* (ETSI): this function implements the methods which are needed for limiting the arriving load. It reads the information transmitted by the CF and throttles the traffic accordingly.

The following sections describe the two standardization approaches and extend the IETF-concept by own proposals to extend its dedicated functionality.

7.2 IETF Activities on Overload Control

The IETF as one of the main standardization contributors is interested in solving the overload problem and initiated working groups led to a number of RFCs and RFC-drafts. The current main driving IETF working group is (since July 2009) the *SIP Overload Control - Working Group (SOC)*. As one result of these activities, the group finalized in August 2011 a standard entitled "RFC 6537 – Design Considerations for Session Initiation Protocol (SIP) Overload Control" [Hilt11]. Other active drafts are a proposal for signaling overload control ("Session Initiation Protocol (SIP) Overload Control" [Gurbani12]) and an initiative to define the various dropping mechanisms ("Session Initiation Protocol (SIP) Rate Control" [Noel12]).

These three documents are presenting on one hand the organizational structure of overload control, and on the other hand a proposed in-bound SIP signaling-mechanism combined with overload control mechanisms.

7.2.1 "RFC 6357 – Design Considerations for Session Initiation Protocol (SIP) Overload Control"

This standard deals with the topic of designing, locating and grouping SIP components for effective overload communication. The authors start with a comparison of implicit versus explicit communication:

- **Explicit overload communication**: The explicit overload communication technique bases on the concept of a SIP signaling extension for a SIP server B to explicitly alert its upstream server A about the fact that B is now reaching its design limitations. Using this signaling, the upstream server A can regulate the downstream traffic to protect the endangered server B.

The SIP Overload Control Draft [Gurbani12] presented in the following section focuses on this challenge.

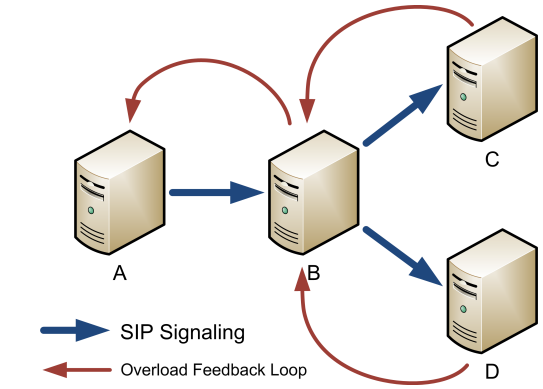
- **Implicit overload communication**: is a solution without any additional signaling. The overload state detection of a downstream SIP instance is solved by monitoring the response time compared to the expected, e.g., average or median, response time or by monitoring packet loss. When observing such an unusual behavior, the upstream neighbor reduces the forwarding load to this potentially overloaded system. An approach to measure these indicators over a longer period of time for consideration as input values is discussed in chapter 6.2.

  As an example, Happenhofer et.al. present in [Happenhofer11] an approach to detect congestion by measuring the RTD *and the number of pending transactions* which are not finalized at the time of measurement. A high number of open transactions is an indicator of an overload state within the monitored infrastructure.

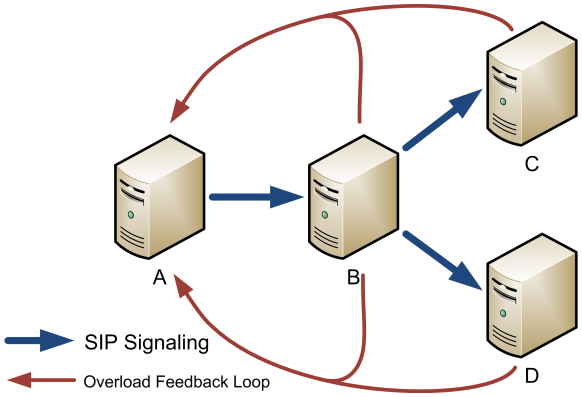
Inter Server Cooperation

In current complex installations like IMS or other large VoIP-deployments, more than one SIP-server is acting as processing node for SIP-requests. For successful overload communication and handling, it is of importance to focus on the location of the overload *Monitor* and *Actuator*, and on the communication between these functions. In RFC 6357 the authors present three degrees of cooperation which are depicted in Figure 7.1.

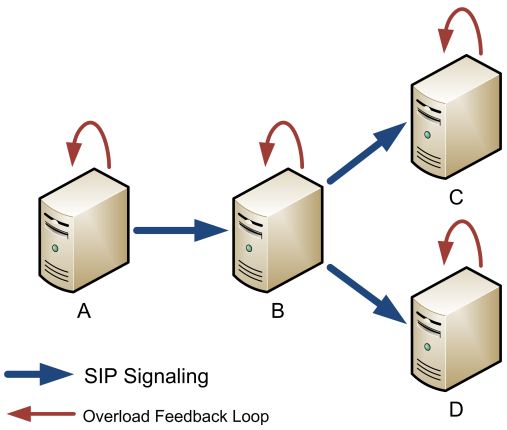
1. **Hop-by-hop** (Figure 7.1, a): In this architecture, the monitor and the actuator are direct neighbors: the monitor transmits the overload state to its direct upstream neighbor where the actuator resides. These direct control loops between upstream and downstream neighbors are completely independent on other control loops in the infrastructure. Overload signaling received from a downstream neighbor is not transmitted to the next upstream neighbor but used as input for the local actuator to – for instance – reject SIP requests for reducing the load on the overloaded downstream neighbor. So, in this cascading hop-by-hop overload control scenario, overload situations are always solved by the direct upstream neighbor. This technique is simple and is scaling well also with many SIP entities. Furthermore, this technique does not need a special aggregation mechanism to collect overload information of many hosts.
2. **End-to-end** (Figure 7.1, b): The monitor is located on all nodes on the signaling path. The actuator instead is acting on the first possible upstream host. This should be the edge node of the core network, but



(a) Hop by Hop



(b) End 2 End



(c) Local Loop

Figure 7.1: Degree of Cooperation between Servers

also the UA might be integrated into this infrastructure. The throttling SIP-server (or UA) must be aware of the traffic flow: it should only throttle requests which are known to influence the load on the overloaded server. All other requests should not be throttled or rejected.

Illustrated in Figure 7.1, (b), server A must possess global or local topology information to decide (or to predict) which message is arriving at host C. As this is complex and often varying, e.g., when using load-balancing algorithms, these end-to-end-mechanisms are problematic in extended VoIP-infrastructures.

A useful end-to-end overload control architecture is possible, when several closed coupled servers can be combined to a control loop.

3. **Local** (Figure 7.1, c): Overload control sending and receiving entities are located on the same SIP-server and *Monitor* and *Actuator* are operating on the same SIP-processor. Hence, the control loop is acting as an isolated application, where in contrast *Hop-by-hop-* and *End-to-end-*technique are external overload control mechanisms.

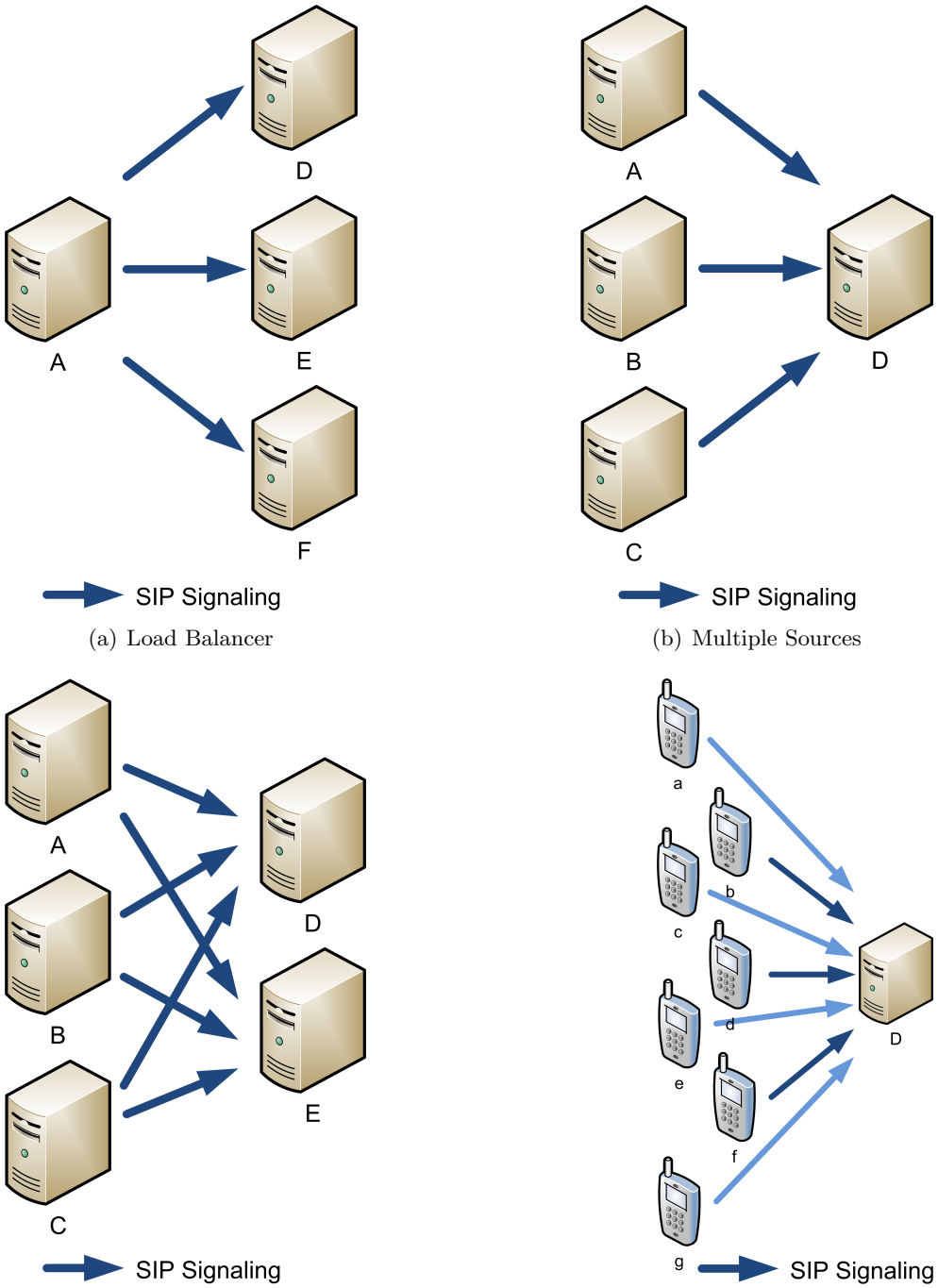
   The fundamental assumption of local overload control is the idea, that dropping a request in advance is less load-consuming than processing this requests. Rejecting a request with a final error response is also stopping the SIP retransmissions, which are creating load as well [Egger10].

   Local overload control can be used in combination with external overload mechanisms and can be understood as the last possible method to stop overload. The benefit of this kind of overload control is, that the entire functionality can be implemented in the SIP server and needs no interaction with other SIP instances. With this isolated solution, compatibility problems can be avoided and an overload control enabled host can be introduced without further adaption.

Topologies

The decision, which type of overload communication to choose depends strongly on the SIP server topology. In Figure 7.2, a-d, the distinct types of inter server cooperations for overload communication are presented. A complex SIP/IMS-architecture is a combination of two or more of these cooperation types, best described in Figure 7.2 c, but with additional backward requests signaling from (D—E) to (A—B—C). To implement the concepts presented in this section, the architectural components should be sub-grouped accordingly.

- In Figure 7.2 a, server A acts as an upstream neighbor of three redundant/load balanced servers D, E and F. If an overload state occurs on



(a) Load Balancer

(b) Multiple Sources

(c) Meshed Architecture

(d) Edge Server

Figure 7.2: Cooperation topology between SIP Nodes

one of these three servers, it communicates its state to A. Server A then decides to reroute the traffic to the two remaining servers. If A knows by monitoring or by predefined configuration, that the three servers are 1) the only downstream servers and 2) all of them overloaded, it can communicate a throttling request to its upstream neighbors acting as representative of these three overloaded hosts.

- Figure 7.2 b, illustrates the reversed architecture. SIP-server D receives a various amount of traffic from multiple sources. If D is now getting overloaded, it decides, which of the upstream neighbors should reduce its traffic by an appropriate amount. The *fairness*-challenge to split accepted traffic between several upstream servers can (according to RFC 6357) be divided into two categories:

  1. **Basic fairness**: in this category, the overloaded server treats all neighbors (and users) equally, and each request has the same chance of succeeding or failing. This basic fairness is interesting with respect to the "Call-Now"-overload-scenarios discussed in section 3.3.2.
  2. **Customized fairness**: the overloaded system reserves ressources according to various priorities. An example is the favoring of specific Service Level Agreements (SLAs).

- A *mesh*-architecture (Figure 7.2c) combines both, the load balancer- and the multiple sources-scenario. Multiple sources (in IMS e.g., multiple P-CSCFs) create load on load balanced core systems (like the S-CSCF). From the viewpoint of server D and E, the scenario is equivalent to the multiple sources-scenario and can be handled in the same way. The decision of the server A to C how to handle the signaled overload can be done in the style of the load balancer-scenario.
- The fourth topology is the *edge proxy* (Figure 7.2d). When an overloaded server (D) interacts with a large amount of upstream clients (like UAs) where each of them is sending requests with a low frequency, the throttling can be done by rejecting a percentage of the incoming requests with a *503 Service Unavailable* response.

Performance Metrics

The performance of an overload protection infrastructure can be qualified using several metrics. The most important KPI is the *goodput*, the amount of requests per second (rps) successfully handled during a capacity-limit state. For example, if a SIP server has a capacity of 140 rps, it should handle these

140 rps even when (in an overload state) the amount of incoming requests is far beyond this design-capacity.

A further KPI is the *signaling delay*: this delay should not increase significantly during periods of high or overload. Increasing this delay will result in an increasing number of SIP retransmissions again resulting in additional high load.

RFC 6357 highlights *Reactiveness* and *Stability* as further important system properties. The first one defines the delay when an overloaded server needs to react and communicate its state to the actuators even during sudden peaks. If the overload state disappears, it is important to remove all throttles in a fast and reliable way. The second property, stability, is needed to avoid significant oscillations of the server load.

Explicit Overload Control Feedback

As introduced in the beginning of this chapter, there is a difference between explicit and implicit overload control feedback. *Implicit feedback* is the method of gaining information about the overload state of a downstream node by analyzing, e.g., changing response delays. *Explicit feedback* is defined when a server actively communicates how much traffic it wants to receive or it wants to be removed.

In the following, categories of explicit overload control feedback will be presented in more detail:

- **Rate-based OC** controls the incoming load by limiting all upstream neighbors. The high-loaded system communicates a maximum rate it is willing to receive. Each of the upstream servers can be assigned a distinct rate cap. In rate based control a server sends a rate cap to each of the upstream neighbors. The server needs to keep a list of all upstream servers and their according rate caps and must be aware of newly added servers. If an upstream server is added, the protected server needs to modify the rate cap of all other hosts accordingly. This must be done to assure, that the sum of allowed rate does not exceed the maximum acceptable request rate.
- **Lost-based OC**: the overloaded SIP-server defines a loss-rate in percent by which each upstream server should reduce its current amount of requests to. The overloaded system does not need to know the current amount of requests per second, nor does it need to know the number and the load ability of the upstream servers. It just communicates a desired reduction in percent. In lost-based OC the server needs to adapt the percentage quickly to handle peak loads. RFC 6357 states the following example: *"if a SIP server sets a throttle value of 10% at time* t_1 *and the*

number of requests increases by 20% between time t_1 and t_2 ($t_1 < t_2$), then the server will see an increase in traffic by 10% between time t_1 and t_2 " [Hilt11]. This example demonstrates the importance of reacting quickly when using lost based OC. It is also demonstrated, that short overload peaks may still arrive at the overloaded system.

- **Window-based OC** allows a server to send a predefined number of messages before a confirmation is needed. Each server maintains an overload window defining a number of messages to be in transit without confirmation. When the counter is reached, the sender stops transmitting.
- **Overload Signal-based OC**. The idea is to throttle the load sent to the overloaded system by using the RFC 3261 [Rosenberg02] *503 Service Unavailable* responses. Assuming, that this response transmitted by the overloaded system is *not* containing a *Retry-After*-header, the upstream proxy can reduce the load to avoid future *503*-responses. If this proxy receives still *503*-responses, it keeps throttling the load until no more *503*-responses are received. By slowly increasing the traffic, the upstream proxy can try to find out if the overload state is resolved. *"Signal-based overload control for SIP servers"* [Abdelal10] characterizes this method and demonstrates, that this technique can enhance the overall performance of an overloaded infrastructure.

  A received *Retry-After*-header in the "503 Service Unavailable"-response would thwart this concept, as the protecting proxy then has to stop the traffic entirely for the *Retry-After*-interval [Rosenberg02] instead of throttling the load stepwise.
- **On-/Off OC**. This control is implemented using *503 (Service Unavailable)* response with an included *Retry-After*-header and enables the congested SIP-server to turn the transmission of requests on or off per upstream host. It is trivial to introduce and already standardized in RFC 3261 [Rosenberg02], but it is not as fine grained as the other presented techniques.

These explicit overload control mechanisms all need a standardized signaling procedure, which must be supported both by the overloaded component as also by the protecting nodes. The "Session Initiation Protocol (SIP) Overload Control"-draft as presented in the next section shows, that the standardization process is long lasting with an open result and an unknown grade of deployment.

Instead, the implicit overload control can be implemented without explicit, standardized communication:

Implicit Overload Control

RFC 6357 discusses also the implicit overload control: Implicit OC assures, that the sending host detects the overload state of a downstream server without explicit communication. It slows down the transmission if it gets indications, that the server is overloaded.

Implicit indicators might be increased response times or even missing responses. Implicit OC should help avoiding additional overload in cases when the downstream server is that much overloaded it is actually unable to send even OC-control responses.

Message Prioritization

Finally, RFC 6357 presents the message prioritization concept: For OC-reasons, a server can prioritize various types of messages and can select explicitly messages to be rejected or redirected. This is mainly a matter of local policies and settings. Generally, messages of high-priority, like emergency service requests and messages of ongoing dialogs should be kept alive even in times of overload. The *Resource-Priority*-field (RFC 4412 [Schulzrinne06]) enables the proxy to identify emergency requests and handle them with adequate high priority.

7.2.2 Draft "SIP Overload Control"

In parallel to the activities in [Hilt11], the "Session Initiation Protocol (SIP) Overload Control"-Draft (version 12, February 2013) [Gurbani12] proposes specific techniques to signal overload states from an overloaded server to upstream nearby servers. This draft focuses only on inter-server communication and is applicable not for server-UA overload control.

For dispatching the OC-messages, the SIP-Via-Header is extended by four specific parameters presented later in this section.

With respect to the design considerations presented in section 7.2.1, this overload control is focusing on "hop-by-hop"-OC (Figure 7.1, (a)). On this purpose the draft recommends using the SIP-Via-header and extends this header by five parameters.

OC-parameters

The following four parameters enable a SIP server to signal its ability to communicate and understand overload-control mechanisms both downstream and upstream in the SIP signaling flow.

1. `oc:` the functionality of the oc-parameter is twofold: an upstream server adding this parameter signals to the downstream neighbor its ability to interpret overload control related messages. The downstream server

must always add a value to the oc-parameter (e.g., `oc=0`) in the responses traveling upstream. With this value, it communicates on the one hand (during the initial handshake), that it supports overload control and on the other hand (when overload control is active) the control to be applied.

Upstream neighbors receiving a response with a value greater zero should reduce the traffic by the amount signaled in oc and the algorithm signaled in oc-algo.

2. `oc-algo:` this parameter is inserted by the upstream server as initial proposal for the oc-handshake and updated by the downstream server. The default algorithm "loss" must always be added (and supported). Upstream server also may insert additional control algorithms such as rate- (introduced in section 7.2.3), x-dropability-loss or x-dropability-rate (section 7.3.3) in a comma-separated list. The server may order the algorithms according to its preferences, but the downstream server does not need to follow this preferred order. The downstream server selects one of the proposed mechanisms and adds it into the oc-algo-field of the response to achieve mutual agreement. Once both servers have agreed on a common algorithm, the algorithm must remain in effect until the downstream server explicitly selects another algorithm. To support this future selection, the upstream server must include all supported algorithms in further requests and the downstream node must communicate the specific selection with each response. Nevertheless, the downstream server must not change the selection within a short time interval to stabilize the overload control mechanism.
3. `oc-validity:` defines the time interval in milliseconds, the loss-rate should be active. If the upstream node receives a response with an oc-value greater 0, but with no oc-validity value, it must use the default value of 500 ms. If the upstream system receives an oc-validity-value of 0, the protected downstream system signals to finish overload control.
4. `oc-seq:` defines an increasing, unique sequence number associated with the *oc*-parameter to protect from response reordering. This parameter is inserted by the downstream node. The draft proposes to use a Unix-timestamp

In *Backus-Naur Form (BNF)*, the parameters are defined as presented in Listing 7.1.

```
oc = "oc" [EQUAL oc-num]
oc-num = 1*DIGIT
oc-validity = "oc_validity" [EQUAL delta-ms]
```

```
  oc-seq = "oc-seq" EQUAL 1*12DIGIT "." 1*5DIGIT
  oc-algo = "oc-algo" EQUAL DQUOTE algo-list *(COMMA algo
   -list) DQUOTE
  algo-list = "loss" / *(other-algo)
  other-algo = %x41-5A / %x61-7A / %x30-39
  delta-ms = 1*DIGIT
```

Listing 7.1: The BNF to define the overload control extensions

Example

As example, the draft presents these snippets of an INVITE dialog between two OC enabled proxies A and B, where B is the overloaded host (Listing 7.2).

```
INVITE sips:user@example.com SIP/2.0
Via: SIP/2.0/TLS a.example.net;
  branch=z9hG4bK2d4790.1;oc;oc-algo="loss"
...

SIP/2.0 100 Trying
Via: SIP/2.0/TLS a.example.net;
  branch=z9hG4bK2d4790.1;received=192.0.2.111;
  oc=0;oc-algo="loss";oc-validity=0
...
```

Listing 7.2: Example call setup with initial exchange of overload control information

A adds to the `INVITE`-request the `oc`-parameter to show B, that it understands and handles the OC-parameters as defined in this draft. B (overloaded) responds to this request with a `100 Trying` adding three OC- parameters `oc`, `oc-algo` and `oc-seq`. With this initial dialog, the two proxies mutually agree on a (currently inactive) loss-based algorithm.

On overload state, the overloaded proxy B responds on an INIVTE-request from the protecting upstream A with a 180 Ringing, containing overload control information (presented in Listing 7.3).

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TLS a.example.net;
  branch=z9hG4bK2d4790.3;received=192.0.2.111;
  oc=20;oc-algo="loss";oc-validity=500;
  oc-seq=1282321615.782
...
```

Listing 7.3: A 180 Ringing response with activated overload control

B confirms again the selected loss-based-algorithm at a desired drop rate of 20 percent. These restrictions shall be active for 500 ms.

Review

Summarizing, the draft is after six years still under high activity and it is currently not clear, which parameters will find their way to the final release. The decision to remove the algorithms (except loss-based) from this draft is correct, as it shortens the discussions to bring the draft to the RFC-state. This draft offers a viable signaling framework for overload-control and -handling, but the drawbacks of this standard proposal should also be considered:

- only Hop-by-Hop signaling
- only Loss-based control feedback
- bound to SIP only, therefore Diameter-connected HSSs are not integrable.
- only server to server overload communication

Nevertheless, this draft will be the first RFC defining methods to control the upcoming threat of overload and DoS in future VoIP networks (see current attack examples in chapter 5).

7.2.3 "SIP Rate Control"

The SIP overload control document already presents a loss based algorithm for overload handling. The complementing "Session Initiation Protocol (SIP) Rate Control" [Noel12]-draft concentrates on rate-based methods.

Using rating mechanisms, the system is robust against heavy *varying* incoming load and defines a strict upper bound of allowed requests. Drawbacks are the need to calculate these upper bounds for each of the upstream neighbors.

Leaky Bucket Algorithm

The default algorithm proposed in this draft is the Leaky Bucket algorithm based on the ITU-T recommendation I.371 [itu04] (Appendix A.2). This algorithm can be understood as a bucket, where the capacity of this bucket is acting as a burst buffer and the allowed rate is draining throughout a hole. The drain-rate is a continuous rate of one unit per time unit and the content of the bucket increases by the number T for each arriving request. According to the SIP overload control draft, T is computed as $T = 1/oc$. If the oc value is zero and the oc-validity is non-zero, the server should reject all incoming requests.

The parameter τ defines a buffering tolerance factor and increases the size of the bucket (b) to $s_b = T + \tau$. A large value of τ and the resulting queuing delay increase the response times. Hence, the draft proposes to configure a large τ for systems with a large number of clients with small arrival rates and a small τ for systems with a small number of clients each with high arrival rates.

Implementing message priority

The SIP overload control draft defines the upstream node as responsible for message prioritization. The proposed leaky bucket implementation solves this task by defining *two* tolerance factors, τ_1 and τ_2.

The algorithm to handle two thresholds, one for all request and one for prioritized is the following:

- all requests are accepted as long as the bucket fill is below or equal to τ_1
- only prioritized requests are accepted when fill is between τ_1 and τ_2
- all requests are rejected when bucket fill is above τ_2

When both τ_1 and τ_2 are set to an equal value, this system acts as if there is no priority handling.

Example request

Similar to the SIP overload control-draft [Gurbani12] the request from the upstream neighbor communicates its rate-capabilities.In addition to the the loss-based algorithm, the server adds the rate-based technique (listing 7.4).

```
INVITE sips:user@example.com SIP/2.0
Via: SIP/2.0/TLS a.example.net;
  branch=z9hG4bK2d4790.1;received=192.0.2.111;
  oc;oc-algo="loss,rate"
...

SIP/2.0 100 Trying
Via: SIP/2.0/TLS a.example.net;
branch=z9hG4bK2d4790.1;received=192.0.2.111;
  oc-algo="rate";oc-validity=0;
  oc-seq=1282321615.781
...
```

Listing 7.4: "INVITE" initiates and "100 Trying" completes the capabilities-exchange dialog

With the 100 Trying response, the downstream (and potential overloaded) server signals, that it is currently in normal state and that it will use the rate-based algorithm in case of overload control.

In overload state, the downstream server will signal the expected rate as presented in Listing 7.5.

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TLS a.example.net;
  branch=z9hG4bK2d4790.1;received=192.0.2.111;
  oc=150;oc-algo="rate";oc-validity=1000;
  oc-seq=1282321615.782
  ...
```

Listing 7.5: "180 Ringing" activating the rate-limitation

In this listing, oc-algo parameter requests, that the protecting server must use rate-limitation for overload control and the oc- parameter defines a rate of 150 requests per second over a time of 1000 ms (`oc-validity=1000`).

7.3 Proposed Standard Extensions for SIP Overload Control

During research work and simulations, we observed potential extensions of the RFC 3261 resp. of the "Session Initiation Protocol (SIP) Overload Control" [Gurbani13].

7.3.1 Modifying T1-timer

The first extension concerns potential SIP retransmission problems. Figure 3.7 of the introducing section 3.3 depicts a high amount of retransmissions up to 25% of all SIP traffic. These retransmissions are mainly caused by clients connected using a high latency connection like GPRS or EDGE [Fabini13]. The round-trip times of these high-latency connections are exceeding (even under optimum conditions) the default T1 timer and the clients are therefore creating retransmissions on all of their requests. The T1 timer value of 500 ms is a default value and the standard RFC 3261 recommends to adapt the timer to a suitable value ([Rosenberg02], section 17.1.1.2). In particular mobile clients with varying RTTs should be motivated to change their T1 timer according to their current connection. An adapted T1 timer will reduce the number of traffic dramatically.

The IETF primarily focuses on inter-server overload control and there is currently no way to reduce load on the edges of the core infrastructure as presented in Section 7.2.1. The only proposed solution is to send the strict *503 Service Unavailable* response. During high-load situations in core infrastructure or high-latency access-nets, clients using UDP are retransmitting their

requests if the server is not reacting within a predefined timespan. The timer T1, which is introduced to retransmit lost or delayed SIP requests therefore creates additional load on a high-loaded SIP-server. This additional load can cause a so called congestion collapse where solely the amount of retransmissions congest the infrastructure [Egger12]. To avoid retransmissions due to the discussed high-latency connections or overloaded core systems, the presented extension proposes a dynamical increase of retransmission timer T1 to reduce the load caused by retransmitted SIP requests. To prove this approach, we simulated the behavior when increasing T1 on an already high-, but not overloaded infrastructure. The findings of these simulations are presented by Egger, Hirschbichler, Reichl in "Enhancing SIP Performance by Dynamic Manipulation of Retransmission Timers" [Egger10]. Simulation results in figures 7.3a-d illustrate, that the success-rate can recover even when the amount of new requests stays constantly high. An interesting point is the advance of the overload state. If the overload situation lasts already for a longer time, the increase of timer T1 must be significantly larger.

If T1 in increased at an early stage, soon after congestion detection (Figure 7.3 b), a timer T1 value of 900 ms can prevent congestion collapse. However, a T1 value of 1200 ms is needed for maintaining an operational system after 100 seconds of congestions (Figure 7.3 d), a T1 value of 900 ms is ending up in a congestion collapse (Figure 7.3 c).

Signaling a modified T1 timer

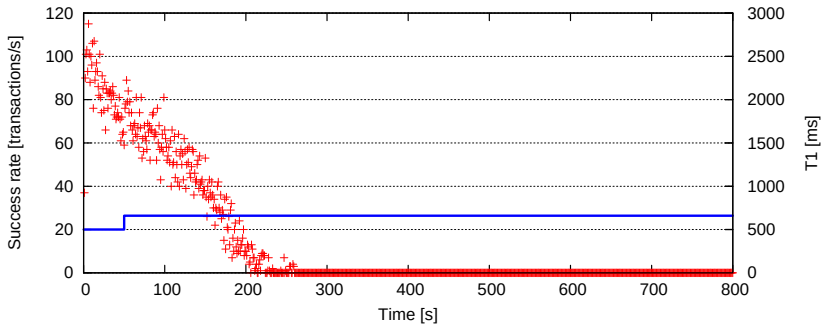
To signal this retransmission timer's modification to SIP clients, it has been decided to extend the RFC draft "Session Initiation Protocol (SIP) Overload Control" [Gurbani13] by two more parameters (Listing 7.6).

```
oc-t1 = "oc_t1" [EQUAL 500++]
oc-t1-validity = "oc_t1_validity" [EQUAL delta-ms]
```

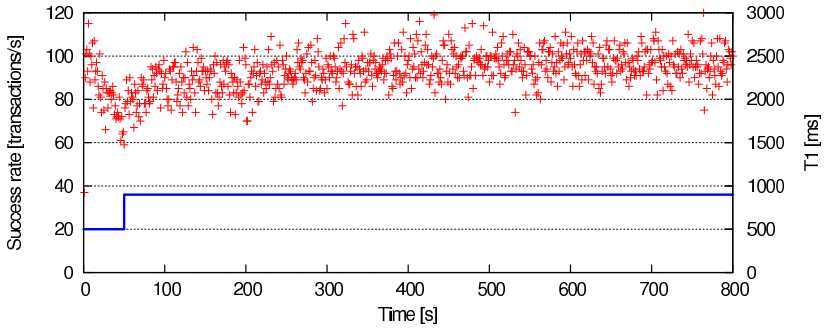
Listing 7.6: BNF definition to activate the T1 timer modification

The parameter `oc_t1` stores the proposed T1-timer value (in ms) and signals the upstream neighbor to change the retransmission settings to this value. The second mandatory parameter, `oc_t1_validity` defines the timespan over which this retransmission-timer modification is valid. After this timespan, the default T1-timer of the affected host will automatically be set back to the RFC3261 default 500 ms.

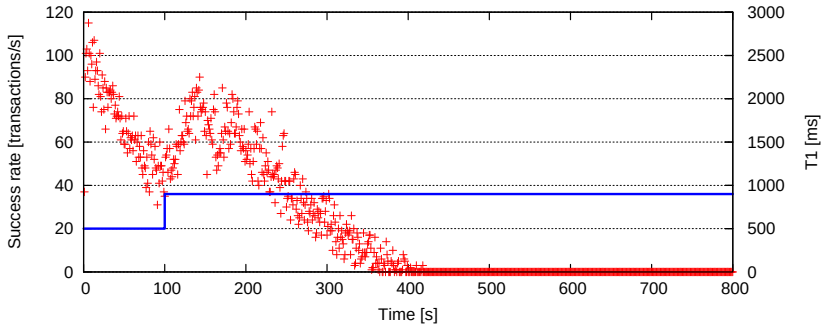
We propose to signal this T1-timer modification in responses to periodic arriving requests, like the REGISTER-request. These requests, which are also updating location-information and re-registration timers, are suitable for



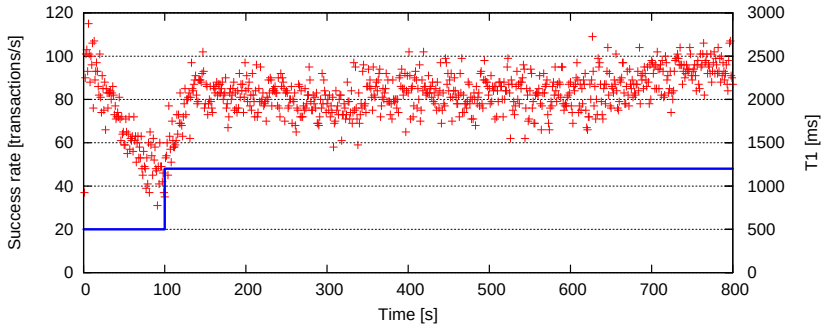
(a) Changing to T1 = 660 ms after 50 s



(b) Changing to T1 = 900 ms after 50 s



(c) Changing to T1 = 900 ms after 100 s



(d) Changing to T1 = 1200 ms after 100 s

Figure 7.3: Depending of SIP session success rates (red points) when modifying the default 500 ms value of retransmission timer T1 (blue line)

retransmission-timer modifications.

As an instructive example, a `200 OK`-response of the high-loaded border node `proxy.atlanta.com` responding to a REGISTER request of a UA which uses the proposed extensions will be presented as follows:

```
#proxy.atlanta.com -> alicepc.atlanta.com

 SIP/2.0 200 OK
 Via: SIP/2.0/UDP alicespc.atlanta.com:5060;
  branch=z9hG4bKnashds7;received=192.0.2.4;
  oc-t1=1000;oc-t1-validity=180000
 To: Alice <sip:alice@atlanta.com>;tag=249334
 From: Alice <sip:alice@atlanta.com>;tag=45a4e33
 Call-ID: 8438176376842308998sdasdh09
 CSeq: 1826 REGISTER
 Contact: <sip:alice@192.0.2.4>
 Expires: 180
```

Listing 7.7: A 200 OK response with overload-control parameters

This response instructs the user agent running on `alicespc.atlanta.com` to change the T1 retransmission timer to 1 sec for a duration of 180 seconds. This time period is equal to the expiration timer (`Expires`-Header), so – when the UA is re-registering after almost 180 sec – the modified retransmission timer will be renewed or will time out and change to the default timer. The simulation results in Figure 7.3 show, that modifying the T1 timer on the fly might be a viable enhancement of the "SIP Overload Control" [Gurbani13]-draft.

7.3.2 "X-Dropability"-header

On complete congestion, rejecting or dropping requests is the last option to protect the overloaded downstream host.

The SBC-A introduced in chapter 10 presents a novel approach to reduce the amount of messages (and the resulting load) already at the infrastructure border. This is done by rejecting obviously useless requests on the one hand and on the other hand by *rating useful requests by their value for the communication and/or the operator* and forward them to the next downstream hop. The header-parameter which is presented in the following will transport the calculated request value.

Syntax

The new `X-Dropability` header denotes the "value" of a request to the system. The higher the value of the request, the lower is the `X-Dropability` header.

The BNF form which extends the RFC 3261 definitions is presented in Listing 7.8:

```
X-Dropability: (1*DIGIT) / (1*DIGIT "." 1*DIGIT)
```

Listing 7.8: The BNF defining the X-dropability header

This header is added by the first supporting host. Initially, this value is set to "0". SBC-As (Or other nodes, which support the `X-Dropability`-concept) might increase or decrease this value. A node must not add a second `X-Dropability` header. If a server is unable to interpret the `X-dropability`-header, it must (according to RFC 3261 8.3 [Rosenberg02]) leave this header untouched and forward it to the next hop.

7.3.3 Extending the `oc`-functionality when using `X-dropability`

Rating the value of each request using the `X-dropability` header supports the overload communication concept: By default, the loss-based algorithms are standard for all oc-compliant implementations. Rate-based algorithms are specified in the "SIP Rate Control" [Noel12]-draft. Common to both techniques is the fact that they do not differentiate by default between economical "valuable"-requests (like INVITE- or MESSAGE-requests) or call-flow-relevant requests like terminating BYE-requests. This topic will be discussed in detail in section 10). Instead, the requests are dropped with identical probability both with loss- or rate-based algorithms.

The `X-dropability`-header adds this per-request rating functionality. To benefit from this functionality, two new concepts for "SIP Rate Control" are implemented:

Loss-based-limitation using Dropability Header

The first loss-based approach is to limit the forwarded requests down to requests, which have a high value for the operator and have therefore been assigned a dropability value below a certain threshold.

To solve this task, the additional oc-algorithm "`X-dropability`" is introduced: Together with the `oc`-value, the overloaded system signals to the protecting upstream server to drop all requests with `X-dropability` values larger than the `oc`-value.

Example request

In Listing 7.9, the INVITE-requests communicates support of three different algorithms and the *100 Trying*-response signals the "`X-dropability`"-technique as chosen algorithm for further communication.

```
INVITE sips:user@example.com SIP/2.0
Via: SIP/2.0/TLS p1.example.net;
  branch=z9hG4bK2d4790.1;received=192.0.2.111;
 oc;oc-algo="loss,rate,x-dropability"
 ...

SIP/2.0 100 Trying
Via: SIP/2.0/TLS p1.example.net;
branch=z9hG4bK2d4790.1;received=192.0.2.111;
  oc-algo="x-dropability";oc-validity=0;
  oc-seq=1282321615.781
  ...
```

Listing 7.9: INVITE-request after SBC-A signaling three supported oc-algorithms: Loss- Rate- and `X-dropability`

In case of overload, the overloaded component transmits the selected algorithm (`oc-algo="x-dropability"`) similar to the original draft and the selected maximum `x-dropability` value (`oc=10.3`, Listing 7.10)

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TLS p1.example.net;
  branch=z9hG4bK2d4790.1;received=192.0.2.111;
  oc=10.3;oc-algo="x-dropablity";oc-validity=1000;
  oc-seq=1282321615.782
  ...
```

Listing 7.10: "180 Ringing" sent by an overloaded system

This response triggers the protecting server to drop all requests having a `X-dropability`-value of greater than or equal to *10.3* over a duration of one second. After one second, the upstream server stops dropping and returns to normal state. This technique fits to the presented SBC-A architecture (chapter 10) and offers an opportunity to drop invaluable requests first. Thereby, the evaluation in chapter 10 shows, that this solution is more effective and offers a fair weighted dropping with limited additional computational costs compared to uniform dropping of requests.

Rate-limitation using Dropability header

The second approach adapts the leaky-bucket driven rate limitation from the "SIP Rate Control" [Noel12]-draft which has been discussed earlier in section 7.2.3. In this draft, the authors present the handling of prioritized packets by defining two bucket-limits. With the precalculated `X-dropability`-value, this concept can be extended to a more effective rate-limitation approach.

Parameter T is defined as drain-rate and τ_1 as the pre-defined soft limit of the bucket. Additionally, a second pre-defined value is τ_2 as hard limit of the bucket ($\tau_1 \leq \tau_2$). In overload state, when the overloaded downstream node signals a rate-limit to the upstream neighbor, the following algorithm will be activated. Before the system places an arriving request with a specific dropability-value d into the bucket, an algorithm calculates a probability p value, which – depending on the fill-grade (δ) of the bucket – decides whether to insert the request into this bucket or to reject the request (using a 503 response). The probability is p is calculated as follows:

$$p = \begin{cases} 1 & \delta \leq \tau_1 \\ \frac{d}{d_{max}} & \tau_1 < \delta < \tau_2 \\ 0 & \delta \geq \tau_2 \end{cases} . \tag{7.1}$$

Using this approach, the bucket is always filled (disregarding the dropability-value) until level τ_1 is reached. When this lower level is exceeded, the dropability value (denoted as d) is brought into calculation: the higher the dropability, the lower is the probability, that a request is inserted into the bucket. d_{max} is the maximum observed d during a specific time span. If the fill-grade δ reaches its hard limiting value τ_2, all further requests will be blocked using the 503-mechanism.

Compared to "SIP Overload Rate Control", this mechanism implements a more sophisticated distinction and prioritization of requests in the fill-rate-area between τ_1 and τ_2. The operator (pre-)defines these thresholds.To signal the capability of this rate-technique, a new `oc-algo`-type named `x-dropability-rate` is introduced. Otherwise, communication is equivalent to 7.2.3, as illustrated by the example in Listing 7.11.

```
INVITE sips:user@example.com SIP/2.0
Via: SIP/2.0/TLS p1.example.net;
  branch=z9hG4bK2d4790.1;received=192.0.2.111;
  oc;oc-algo="loss,x-dropability-rate"
            ...

SIP/2.0 100 Trying
Via: SIP/2.0/TLS p1.example.net;
  branch=z9hG4bK2d4790.1;received=192.0.2.111;
  oc-algo="x-dropability-rate";oc-validity=0;
  oc-seq=1282321615.781
  ...
```

Listing 7.11: INVITE-request after SBC-A signaling the `x-dropability-rate`-support

The 100 Trying response signals its support of the `x-dropability-rate`-rate limitation In state of congestion, the downstream server activates the upstream-server's `x-dropability-rate` support and requests a maximum drainrate of 150 requests over a timespan of 1000 ms.

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TLS p1.example.net;
  branch=z9hG4bK2d4790.1;received=192.0.2.111;
  oc=150;oc-algo="x-dropability-rate";oc-validity=1000;
  oc-seq=1282321615.782
              ...
```

Listing 7.12: "180 Ringing" activating the dropability-controlled rate-limitation

The presented solution extends the proposed rate-limitation to a request- and dialog-sensitive one. In state of overload low-value requests will be dropped first on the upstream SIP-server to avoid further congestion.

Chapter 10 will evaluate the effects of this approach and the increase of total effectiveness.

7.3.4 Summary of IETF-based overload handling

The IETF is running an active working group to enhance the currently limited overload control-capabilities of RFC 3261, moving away from the simple *503 Service Unavailable*-response to more sophisticated types of overload signaling and controlling.

RFC 6357, *Design Considerations for SIP Overload Control* [Hilt11] is the basic document for designing overload infrastructures. It presents different architectures and communication paths for overload handling and signaling, without going into details of implementation or protocol specific tasks.

SIP Overload Control [Gurbani13] adapts the findings of RFC 6357 to propose specific extensions of the SIP-protocol. It focuses on hop-by-hop-overload signaling and does not consider complex server-to-server relationships as implemented e.g., in IMS. This draft is the most active paper of the working group, currently in *Working Group Last Call (WGLC)*-stage and at the end of 2013 a RFC of the protocol extensions can be expected, providing a stable and useful overload control standard.

Our proposed extension published in [Egger10] extends [Gurbani13] and provides a more comfortable way to include IMS and SIP terminals and user agents into the overload handling infrastructure than defined in RFC3261. With introduction of the X-dropability-concept, subsequent overload control mechanisms can be implemented to refine request dropping effectiveness.

7.4 Generic Overload Control: The ETSI Approach

Whitehead discusses in [Whitehead05], that it is essential and feasible to design a generic control of overload processing for next generation networks (NGN). NGNs combine multiple application protocols (like SIP, HTTP, Diameter, Radius, DNS, SNMP) over multiple server instances. Although some effort has been made to manage bandwidth and router congestion (like RFC 3124 'The Congestion Manager' [Balakrishnan01]) almost none of the approaches addresses host and server processing overload.

Some of the presented protocols provide simple overload handling by the definition of appropriate error response codes (like SIP' "503 Service Unavailable"), but few currently provide specific overload control mechanisms.

Whitehead states, that the traditional way providing overload control is to introduce one mechanism for each protocol. As an example, the current "SIP Overload Control"-draft presented in section 7.2.2 is discussing a limited SIP-protocol mechanism only with reduced capabilities. A quicker and cheaper method of solving overload handling is to introduce a dedicated overload protocol independent of the application protocols. The basic design principle of such a generic protocol is a distinction between the components to monitor (M), a function to update the restriction levels (U) and one to restrict the traffic (R).

The benefits of the *Generic Overload Control Application Protocol (GOCAP)* are according to Whitehead:

- Easy integration of Internet-based protocols to the service-level-enabled assured domain of telecom suppliers and operators
- Only one overload control mechanism to support and configure
- Easy integration into standardization bodies and
- flexibility of implementation, as there is no need to implement GOCAP if not required.

The scope of GOCAP is limited to the implementation of server/host overload control caused by processing applications like SIP or LDAP. Out of scope is the overload of transmission capacity.

7.4.1 ETSI ES 283 039

Using the GOCAP concept introduced by Whitehead in [Whitehead05], the ETSI Technical Committee "*Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN)*" has developed a five-part ETSI Standard (ES) defining the core features for a protocol independent network overload control architecture (NOCA).

1. "Overview" – withdrawn (*ES 283 039-1*)

2. “Core GOCAP and NOCA Entity Behaviors” [ETSI10]
3. “Overload and Congestion Control for H.248 MG/MGC” – ETSI Standard (ES 283 039-3) [ETSI07a] since 2007-06
4. “Adaptive Control for the MGC” – ETSI Standard (ES 283 039-4) [ETSI07b] since 2007-04
5. “ISDN overload control at the Access Gateway” – Draft

Aim of the withdrawn ES 283 039-1 was to present conceptual discussions as presented in IETF RFC 6357 [Hilt11]. Until releasing ES 283 039-1 [Whitehead05] and [Hilt11] can be taken as background information for reading ETSI *ES 283 039-2*.

This section focuses on the document *ES 283 039-2* which covers the IMS core functionalities and the ways how to interact and communicate the overload state in the core.

7.4.2 Design Concepts

Whitehead and Williams propose in [Whitehead02] design principles which are adapted for ES 283 039.

Internal overload control

All components, which can run into processing congestion must be able to detect processor overload. Additionally, they need an admission control to reject incoming requests in case of overload. When rejecting, the control shall reject the minimum amount of incoming requests to successfully handle as much traffic as possible.

External overload control

Rejecting incoming requests consumes processing effort, so it is necessary that a congested component can control external upstream components to reduce the number of incoming requests.

Discrimination

Both internal and external overload control must discriminate between initial and subsequent requests, so that subsequent requests should not be lost. Emergency requests must also be preferred compared to other requests.

Closed loop feedback control

Figure 7.4 depicts a closed feedback loop controlling the application's incoming workload. The components are the *Application* which represents the protected functionality/proxy. This application communicates its load or remaining

capacity-information to the *Control Adaptor (CA)*. The CA calculates (based on the capacity information) and continuously recalculates a global leak rate which is passed to the *Control Distribution (CD)*. The combination of these components is also called *GOCAP Master*. The CD applies simple load poli-

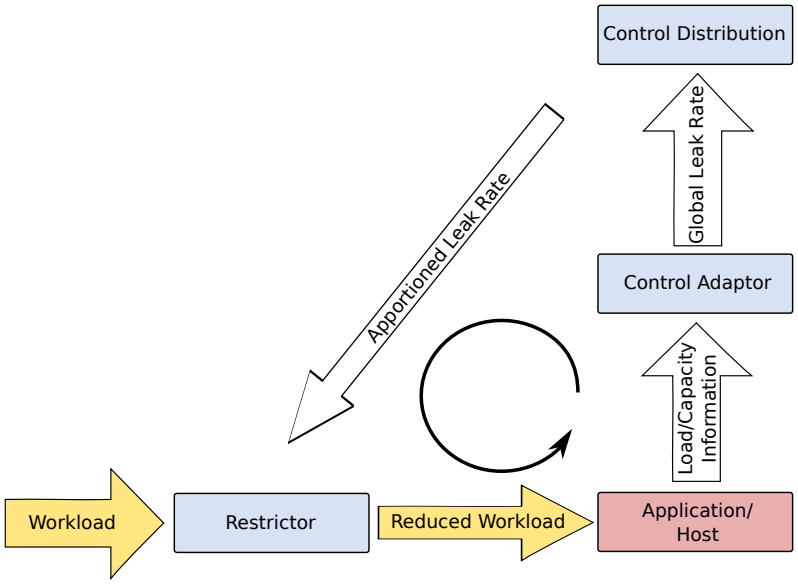


Figure 7.4: Control components implementing a NOCA feedback control path, ETSI ES 283 039-2, figure 1

cies and transmits the leak rates to the *Restrictors* (also *GOCAP Slaves*) on local or remote hosts using the *Generic Overload Control Application Protocol (GOCAP)*.

Finally, the *Restrictor* applies a request priority aware leaky bucket rate limiting algorithm. This algorithm is controlled by the leak rate received from the CD component and reduces the workload transmitted to the *Application*. A restrictor can act as GOCAP slave for multiple masters.

7.4.3 The GOCAP Implementation

ETSI ES 283 039-2 specifies Diameter (RFC 6733 [Fajardo12]) and SIP (using the SUBSCRIBE/NOTIFY mechanism) as the two protocols, which are used to transfer the GOCAP relevant messages as payload. According to ES 283 039, other protocols are also allowed as long as the behaviors of the hosts and the overload control will be the same. The authors propose to use the protocol, which is already supported by the entities of the feedback control path for other purposes (e.g. SIP Application Servers or SIP Proxies). In SIP, the GOCAP

master implements the role of a SIP *notifier* and the GOCAP slave acts as a SIP *subscriber* (RFC 3265 [Roach02]). On activation of overload control, the slave subscribes to the *"congestion_control"*-event on each protected master in its list. On restriction event, the GOCAP master sends a NOTIFY request to all GOCAP slaves which have subscribed for the *"congestion_control"*-event. The GOCAP slave must then parse the attached *Extensible Markup Language (XML)* payload of the NOTIFY request and process the GOCAP directives.

7.4.4 Benefits and Conclusions of GOCAP

The demand for an extensive congestion control in a multi-protocol environment is the reason for the introduction of a protocol independent overload control layer. The proposed GOCAP approach tries to meet these requirements. The high complexity in configuration and development is compensated by the benefit of one single protocol for all different kind of servers.

Although ETSI ES 283 039 [ETSI10] was originally introduced 2007 and the operators are aware of this problem, there is currently no known implementation of GOCAP.

7.5 Comparison of ETSI and IETF Solutions

The main problem in complex VoIP infrastructure like IMS is the mixture of protocols which do not specify overload control behavior in detail.

The ETSI concept defining NOCA and the protocol GOCAP follows a generic and holistic approach: an explicit overload control protocol is defined independently of the application-protocols of the protected infrastructure. This overlaying overload control protects hosts and servers no matter which protocol they use. This concept simplifies the integration of Internet-based protocols into the service-level assured environments of telecommunication operators. The high complexity of this strategy and the currently missing demand for congestion control in IP-based telecommunication are probably the main reason, why there are currently no known implementation of the NOCA-approach.

The main signaling protocol in future complex VoIP infrastructures will probably be the Session Initiation Protocol (SIP). Together with Diameter for authentication, authorization and accounting it will create the main load in core infrastructures. The IETF decided (in contrast to ETSI) to develop congestion control extensions on a protocol base. The presented informational RFC 6357 [Hilt11] discusses the problem space of SIP overload concerning the types of inter server cooperation, the SIP node topology and explicit- versus implicit overload control. More specific, the current SIP Overload Control Draft [Gurbani12] defines a SIP-based in-line overload control communica-

tion on a hop-by-hop base. Compared to the ETSI concept, this approach lacks the possibility to create end-to-end overload control to keep the traffic at the perimeter of the protected core infrastructure. Furthermore, this protocol-extension is limited to SIP and is not capable to protect e.g., congested Diameter nodes.

Despite its reduced functionality, the IETF based concept might have good chances to be realized as this extension is easy to implement and the most common and Open Source SIP proxy *Kamailio* offers a developer-friendly interface for extending SIP headers.

Chapter 8

Anti-SPIT Solution proposals

Besides denial of service attacks, unsolicited communications are threatening VoIP infrastructure deployments. Section 4 introduces the voice SPAM-problem compared to email SPAM. This chapter presents proposals to defend unsolicited communication. These proposals base on the informational RFC 5039 [Rosenberg08b] and are extended by current scientific work. The concepts discussed in this section will be used in the reference architecture of the SBC-A in chapter 10. For better differentiation, the SPIT prevention mechanisms are classified into non-intrusive and intrusive anti-SPIT mechanisms.

8.1 Non-intrusive Anti-SPIT Mechanisms

The non-intrusive anti-SPIT approach is transparent to caller and callee and is the preferred mechanism as the involved parties are not disturbed by unexpected interaction. However, these mechanisms are not sufficiently accurate in differentiating between SPIT originators and simple users, such that support by intrusive anti-SPIT mechanisms (section 8.2) might be required for more accurate results.

The following sections will describe the most relevant non-intrusive anti SPIT mechanisms.

8.1.1 Content Filtering

Content Filtering is the most common technique in email SPAM-filtering: Receiving or forwarding Mail Transfer Agents (MTAs) parse email headers, content and attachments for finding SPAM-suspicious content. Usually, these are nouns like "Rolex", "Viagra", etc. Modern Email-parser like SpamAssassin[1] are also able to detect even blended strings like "R0lex" or "V!agra". This

[1] SpamAssassin, http://www.spamassassin.org

technique is not working for blocking VoIP SPAM due to two reasons (RFC 5039):

1. VoIP content is delivered in realtime and no precedent analyzing is possible. If the called party's client rings and the callee answers the call, the disturbance already happened.
2. Analyzing of VoIP media streams for specific content is a complex task: audio recognition might be consciously disturbed by the SPIT sender by adding background noise, poor grammar or varying accents.

On the other hand, this approach is useful against IM- and Presence-SPAM. The advertising message is transported in the request itself and therefore can be parsed with the same techniques as already deployed in Email-SPAM-prevention. This concept and a practical realization is presented in chapter 9.

The small delay while parsing the messages is negligible as the requests do not have to be transmitted in realtime. The parser must in particular check the `Subject`- and the `From`-header as well as payload of `SUBSCRIBE`-, `NOTIFY`- and `MESSAGE`-requests as these information are directly forwarded to the User Interface (UI) of the terminating UA.

Summing up, content filtering is not usable for the most significant and intrusive SPIT-threat, the call SPAM, but it is effective for defending against *message-based* VoIP SPAM.

8.1.2 Black Lists

Blacklisting is a technique of blocking email addresses or complete domains. This approach was useful when email SPAM first arrived, but at the time of writing (2013) professional SPAM originators are able to deploy the messages using an vast number of existing Email-addresses without any big effort. Additionally, as there is usually no authentication or From-address verification when sending Emails, Email-addresses in the Simple Mail-Transfer Protocol (SMTP)-From-Header can be spoofed. Techniques like the "sender policy framework" [Wong06] result in additional challenges rather than solving the SPAM threat as this technique is breaking the automatic mail-forwarding. The fake-header problem also exists in the SIP-Domain: The From-Header in SIP can be modified when SIP communication is sent directly to the B-Party or the proxy of the B-party (case 1 and 2 in section 4.1).

One approach using the black-lists in a useful way is to focus on the sender's IP-address: Ramachandran and Feamster illustrated in [Ramachandran06], that 80% of all SPAM messages are sent from hosts which are already black-listed in a central antiSPAM-blacklist, and 50% of all Email-SPAM in two

or even more blacklists. They showed, that only a small fraction of SPAM is received from an "unknown" host. Inferring from these results onto VoIP, Voice-over-IP SPAM might also be sent from a limited number of infected hosts which are already registered in SPAM blacklists.

Compared to Email-SPAM, VoIP-SPAM defending has one big asset: for creating a SIP call, the SPAM originator must transmit a valid IP address. Otherwise, all responses are lost. Especially the 200 OK-responses must be highlighted as they are containing the essential SDP-content describing the target's RTP stream endpoint. Without this description, the SPAM originator is not able to create a voice dialog and the SPAM attempt fails

VoIP-SPAM Filtering using IP-Blacklists

Section 9 presents the concept of using the results of DNS-based Blackhole List (DNSBL)-queries to rate the SPIT-probability of a SIP-request. For querying a DNSBL server, IP addresses are extracted from the SIP-request, inverted and added as a 'virtual' hostname to the base-address of the DNSBL-server. This concept is similar to ENUM-queries presented in section 5.2.1. When the requested IP-address is part of the database, the DNSBL server returns a "virtual"-IP address (usually from 127.0.0.0/24 address space). If the IP-address is not blacklisted, the DNS server replies with a `Name Error`-message.

8.1.3 Whitelisting

RFC 5039 discusses whitelisting as an alternative approach to blacklisting. The protecting system stores the user identity of callers who are allowed to initiate calls. Compared to blacklisting, the address spoofing problem is reduced to the risk, that an address already on the white list is misused. With strong authentication, this problem can be reduced significantly.

Whitelisting is successfully in use for instant messaging and comfortable when communicating with a limited number of contacts. The main challenge is the *introduction problem* on adding a contact to the whitelist.

The introduction problem can be solved by several more or less invasive ways, like using the *Consent based communication* as presented in the following paragraph.

8.1.4 Consent-Based Communication

This technique proposed by RFC 5039 supports the introduction problem for white- or blacklists. In the first step of a connection establishment, the call-attempt is rejected by the SIP proxy. The called party is hereby contacted and gets informed, that a contact request happened. If the called party accepts

the contact, this contact is added to a white list and future calls will be routed to the callee without additional delay.

In *SIP presence*, this technique is already in use. The method on being added to a presence-contact list is defined using the watcher information event package defined in RFC 3856 [Rosenberg04a] and RFC 3857 [Rosenberg04b].

As discussed earlier in section 4.2, the call SPAM will be reduced, but defending the SPPP needs additional actions: If a contact-list request contains the From-header `"please-buy-my- product-on-this-website@spam.com"`, the SPAM-message is sent to the victim. This weakness can be solved with the content-filtering system as presented earlier in section 8.1.1.

The feasibility of consent based filtering is demonstrated by the prototype implementation of *SPITassassin*. This tool (presented in the following chapter 9) is a connector-implementation for the popular "Kamailio SIP Proxy"[2] to use SPAMAssassin, the Open Source email SPAM-filtering tool[3], for parsing SIP messages.

8.1.5 Reputation Systems

RFC 5039 proposes the concept of *Reputation Systems* as centralized architectures which manage a reputation score: this score supports the contacted person to decide whether a caller should be added to the contact- or white-list or not.

RFC 5039 proposes as example a button in the user interface of a messaging client to interact with the central reputation system whether a user is abusive or not. Hereby, the input of one single user is insufficient, but adequate negative feedback of multiple users would give the abusive user a negative reputation score. Reputation systems are already successfully used in centralized social communication or trading-platforms like Ebay[4]. Similar to blacklisting, one of the main challenges of reputation lists is the simplicity of creating new SIP accounts: If accounts are easy to acquire, identities with negative reputation will simply generate a new account. Additionally, users can be blackmailed by groups of other users threating these users to give them bad reputation scores.

Reputation systems basing on positive feedback are facing similar problems than the ones basing on negative reputation. Groups of SPIT originators may create a lot of accounts for producing positive feedback to each other – creating *artificial positive reputation*. The advantage of positive reputation is its similarity to whitelisting: newly created accounts are not yet marked

[2] http://www.kamailio.org
[3] http://www.SPAMassassin.org
[4] http://www.ebay.com

with positive reputation and first-of-all they need some positive feedback to be perceived as positive part of this network. According to RFC 5039, reputation systems basing on positive feedback are superior to negative-reputation based system – although, they are far away from being perfect.

One option of creating and maintaining positive reputation lists is the combination with the "buddy-list" concept in presence-systems. If someone is part of the presence list of his buddy, this person is automatically on his whitelist, too.Buddies on this buddy-list have buddies too, hence a multilevel-reputation tree (or a "web of trust") as used in social networks like Facebook[5] is introduced. The level of trust implicitly decreases with the number of nodes between the caller and the callee.

An additional threat is local security: when the computer is hacked and transformed into a remote controlled host (*zombie-host*) which is SPAMing all others on the contact-list, the reputation will decrease quickly. Regaining a positive reputation is then difficult and restricts the user of getting onto the positive reputation list to communicate again with his buddies.

8.1.6 Address Obfuscation

A widely used technique for Email-SPAM originators to gather target-addresses is to crawl various Internet-services like websites and news-server (Network News Transfer Protocol (NNTP) [Barber00]). Being aware of this risk, the email addresses in some NNTP-implementations are automatically hidden and often obfuscated on website. These techniques can also be adapted to fetch SIP-URIs as these are also presented on public web sites.

Another way (RFC 5039, 3.5) to collect SIP-URIs is the (mis-)use of ENUM [Faltstrom04]: ENUM offers SPAM originators a chance to collect SIP-URIs by crawling through the tree of a public accessible ENUM-server like `164.arpa`. A proof-of-concept–script for using this attack-vector is presented in section 5.2.1.

8.1.7 Limited-Use Addresses

RFC 5039 proposes another address-obfuscation related technique named *limited-use addresses*. Like in Email, the creation of time-limited addresses for web-forms and company-contact-requests can help to reduce SPAM and SPIT.

But not just time-limited, also group limited addresses are possible. So, the user could create unique addresses each for company contacts, friends or email lists. When one of these addresses is affected by SPAM, it can be deactivated (or filtered) without affecting the other addresses. As this deactivation also

[5] http://www.facebook.com

affects all other users using this address, the usefulness of this proposal is reduced.

8.1.8 Computational Puzzle

SIPp[6] is a SIP testing tool capable to create hundreds of calls per second. The *Computational puzzles* approach is a concept to reduce such high-rate attacks.

Similar to the intrusive Turing test presented in the next section, the "Computational Puzzle" (RFC 5039, 3.9) introduces a technique, where the UAS or the incoming proxy prompts the UAC to solve a puzzle. Performing a time- and processing-power consuming calculation where the calling computer has to spend a few hundred milliseconds can act as a technique for reducing mass-calls. This short processing-delay makes mass-calling Central Processing Unit (CPU)-expensive, whereas one single legitimate call setup be slowed down, but this delay is not perceptible.

Although this technique seems to be a method to reduce mass-calls, it must be taken into account that mobile terminals have a massive reduced processing capacity compared to high-end multi-core PCs. Solving one computational puzzle on a mobile device might need more time and processing cycles than solving one hundred computational puzzles in parallel on one high end SPIT-generating machine.

The computational puzzle approach will fail when the attacker use distributed botnet hosts. The high amount of nodes in combination with the CPU-power makes it impossible to find a puzzle-algorithm which is sufficiently complex against the SPAM originator but which will not block normal communication.

8.2 Intrusive Anti-SPIT Mechanisms

The presented non-intrusive SPIT-protection mechanisms have weaknesses and drawbacks which prevent them to stop SPIT reliably. Extending these mechanisms by interactive techniques improves detection reliability, but decreases (in some cases massively) the usability of the service.

Nevertheless, when observing massive SPIT attacks, some of the in RFC 5039 presented interactive methods might be unavoidable. A proposal for integrating interactive methods is presented in chapter 10.

8.2.1 Turing Tests

Alan Turing introduced in his article "Computing Machinery and Intelligence" [Turing50] a method to tell apart humans from computers. In WWW-

[6]SIP performance testing toolkit, developed by HP, http://sipp.sourceforge.net

forms (guestbooks, contact forms, etc.) this approach is used to differ between automatic SPAM originators and real humans. For accessing specific information a puzzle must be solve, which is feasible for humans but complex to impossible for automated algorithms. For web-forms these tests – also known as *Captcha* – are mostly implemented by presenting a string depicted in an image with a lot of background noise. Such puzzles can be solved by humans, but with the graphical background-noise it is hard to be automatically recognized by pattern-detection algorithms.

Defending voice SPIT is more sophisticated. In this case, the voice based Turing test is used when a caller is currently unknown and not already on a whitelist. The call is automatically transfered to an *Interactive Voice Response (IVR)* where the user is challenged by e.g., telling a few random ciphers, which this caller has to enter on the keypad. To increase the failure-safety background-music or -sounds might be played (similar to the background noise in the captcha-images), which makes it more challenging for automatic voice recognizing SPIT-agents. If the user enters the ciphers correctly, he is automatically added on a whitelist for the caller or the called domain. This guarantees, that each caller-identity is only challenged once.

One important topic is that handicapped or disabled users as also persons with foreign languages must be able to understand these challenges. The developer must make sure, that e.g., a older person is also able to place a call, or that an English speaking person can successfully call an emergency number in Austria. The latter problem could be solved by using and analyzing the SIP-`Accept-Language`-header defined in RFC3261 [Rosenberg02].

Successful attacks against Turing tests are already recorded in the SPAM-field. On the one side, RFC 5039 mentions paying cheap workers from poor countries who can solve multiple Turing tests per minute on cheap rates (RFC 5039, 3.8).

On th other side, another entry solving Turing tests is *social engineering*: The image-captcha is embedded in a porn or warez site animating interested users to solve the (embedded) captcha for getting "more detailed" pictures of free pirate copies from this website [Ordoñez07]. If and how this approach can be used for solving Turing tests in voice networks is currently an open task.

Summarizing, RFC 5039 states, that Turing tests (alone) may never completely solve the SPIT-problem, but instead, the concept reduces the user's perception of the offered service.

8.2.2 Payments at Risk

The next presented approach is also similar to Email-SPAM: the caller has to deposit a small amount of money onto an account of the recipient. If the

called user qualifies the message as non-SPAM, the money is returned to the sender – if the message is SPAM, the called user can keep the money.

This approach can also be used for defending against SPIT and is again solely used for the phase of first contact. If the caller is on the whitelist once, this transaction is not needed anymore. In addition to the complex design with two money-transactions over a trusted third party, which probably will charge a small amount of this micro-payment (RFC 5039 states around 15%), this micro-payment will create more problems than it solves. This technique needs the same understandings of "small amount of money" between the caller and the callee: a poor person (e.g., in the Third World) will keep the money whether the call was SPAM or not. On the other hand, the poor person is not willing to create a call fearing that it will lose the invested money.

If the amount of money is lowered to a level, where also massively poor people can handle this problem, professional SPAM originators might also afford to create these calls

8.2.3 Legal Action

In western countries there are two legal approaches for preventing unsolicited communication; the "Opt-In"-system like for customers in the European Community [ec202] [TKG] and "do-not-call" ("Opt-Out" or "Robinson-Lists") as used in the United States of America [Tauzin03].

In Opt-In-systems, a telemarketer is allowed to place calls to potential customers, when the customer agreed *in advance* to get this kind of communication. As additional feature for classic CS-calls, the caller is not allowed to suppress the calling line identity – this helps the customer (and providers) to operate blacklists.

In other legal-systems, customers can be registered to "do-not-call"-lists. Marketers must take care not to contact people on these lists.

RFC 5039 states, that a similar "do-not-call"-list will be less successful in Email- or VoIP-communication, as this SPAM is likely to be sent from international sources, where the senders are not subject to national law and national law enforcement authorities. One proposed solution is (under the precondition of strong identities) that the system determines whether the caller is in a jurisdiction with strong anti-SPAM laws or not. If the SPAM originator is not from such a country, the communication can be rejected.

8.2.4 Circles of Trust

This technique proposes a cooperation of a group of providers which interconnect SIP-calls and initiate safe inter-domain authentication. With this

authentication and the close coupling between the companies or providers, actions against customers sending SPIT can be provided without huge effort.

IMS which uses SIP as core signaling protocol can be seen as such a closed coupled group of operators. Each of them know their customers (for Authentication and Accounting) and can act to stop them from sending UC.

Another approach is to create isolated VoIP-networks which are not connected to an external packet switched network. Instead all outgoing calls are routed through SIP2CS gateways. In this model, the customers are well-known by the operator and they can be blocked on demand. The authors of RFC 5039 are skeptical about increasing sizes of these close-connected providers, as with the increasing size of these Circles-of-Trust networks: the benefit of sending SPAM will get higher than the risk of being closed out or being fined by the provider. This skepticism is demonstrated by the fact, that also in classical CS-networks (with the same preconditions as Circles-of-Trust) SPAM appears. The amount of SPAM is in fact by orders of magnitude lower than in Email-infrastructures, where these circles-of-trust are not available.

8.2.5 Centralized SIP Providers

Extending the approach of section 8.2.4, the standard proposes a central SIP-routing instance. The authors of RFC 5039 adapt the concept of the PSTN inter-exchange carriers and reuse it for the VoIP-domain. Every SIP provider connects solely to a central inter-domain SIP providers and other SIP operators only accept foreign domain requests arriving from these inter-exchange providers. The inter-domain providers charge local providers for each request.

The IP eXchange (IPX) [ipx07] by the GSM Association (GSMA) implements this concept for IP based packet exchange. For IMS-roaming the concept has several advantages: primarily designed to guarantee a high QoS, the IPX also solves the problem of mutual trust. As a result, each request will probably be charged against the central instance (which can be forwarded to the customer or not), but there is no need to charge micro-payment from the customer (like presented in 8.2.2).

8.3 Authentication-based SPIT Defense

Many of the problems discussed in section 4 can be solved with strong authentication of callers within one or across several domains.

8.3.1 Authentication in SIP

The initial meanwhile obsolete RFC 2543 [Handley99] and RFC 3261 offer a basic authentication scheme. Later standardization approaches extend this basic approach.

RFC 3261 authentication

IETF RFC 3261 [Rosenberg02] proposes a mechanism using the "HTTP digest authentication" developed in RFC 2617 [Franks99] to authenticate a client during registration process using a "401 Unauthorized" challenging response. The same technique is used on creating a SIP call setup by challenging the INIVTE request using the "SIP/2.0 407 Proxy Authorization Required"-response. This challenge-response approach can be used for all other non-ACK SIP requests.

The authentication of a SIP client during registration process is presented in Listings 8.1 and 8.2, where a SIP UA of the longterm-monitoring-project (introduced in section 6.2) registers successfully at the A1TA VoIP-service's registrar.

```
REGISTER sip:a1.net;transport=UDP SIP/2.0
...
Contact: <sip:lttuser1@128.131.88.50:15060;rport>
Expires: 6000

SIP/2.0 401 Unauthorized
...
WWW-Authenticate: Digest realm="a1.net",domain=
  "sip:ttcnserver@a1.net",nonce="5d16f8a9dc9e1e",
  stale=false,qop="auth",algorithm=MD5
```

Listing 8.1: Initial registration attempt rejected with a "401 Unauthorized"-response

```
REGISTER sip:a1.net;transport=UDP SIP/2.0
...
Contact: <sip:lttuser1@128.131.88.50:15060;rport>
Authorization: Digest username="lttuser1",realm="a1.net",
   cnonce="6b8b4567",nc=00000001,qop=auth,uri=
   "sip:80.75.55.109:5060",nonce="5d16f8a9dc9e1e",
   response="91aa56ed0eb7d957585b6c88c8c78181",algorithm=
   MD5
Expires: 6000
...

SIP/2.0 200 OK
...
Contact: <sip:lttuser1@128.131.88.50:15060;rport>;expires
   =60;q=1
Authentication-Info: nextnonce="5d16f8a9dc9e1e",qop=auth,
   rspauth="f7901919dd4ad1",cnonce= "6b8b4567",nc
   =00000001
```

Listing 8.2: Second registration attempt with authentication header

The presented three-way-handshake is initiated by the challenging server: In the “401 Unauthorized” response, the SIP server transmits the “WWW-Authenticate”-header. In this header, the nonce-parameter is used as a random seed for the subsequent authentication procedure. The challenged client uses this nonce together with the user-credentials to generate a response. This response is hashed using the MD5-algorithm and is put into the “Authorization”-header in the next REGISTER-request. Extending the same procedure with the stored user credentials, the authenticating server is able to compare the calculated hash with the transmitted hash to validate the authenticity of the registering user.

The same steps are required when challenging a call: the first INVITE-request is challenged with a 407-response together with an HTTP digest-nonce. The client calculates the MD5-hash and transmits this hash similar to the REGISTER-example in the ”Proxy-Authorization“-header.

This authentication is only working against the home registrar (or proxy), as only this registrar knows the credentials and is able to authenticate the caller.

RFC 3893 authentication

A further approach for providing the identity of a sender is the ”SIP Authenticated Identity Body (AIB) Format“ defined in RFC 3893 [Peterson04]. This standard extends the concept of RFC 3261 for signing the SIP-request: In RFC 3261, section 23.4.2, a copy of the entire SIP-request is attached as payload to this request (Content-Type: multipart/signed) and this payload is signed using a certificate.

RFC 3893 reuses and extends this concept: in contrast to RFC 3261 which proposes to hash the entire SIP-request, this standard proposes a reduction of the number of hashed headers so that the integrity can be assured further-on on the one side but the payload will not increase to much on the other side. The introduced Content-Type is ”message/sipfrag”.

```
INVITE sip:bob@example.net SIP/2.0
Via: SIP/2.0/UDP pc33.example.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@example.net>
From: Alice <sip:alice@example.com>;tag=1928301774
Date: Thu, 21 Feb 2002 13:02:03 GMT
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.example.com>
Content-Type: multipart/mixed; boundary=unique-boundary-1
--unique-boundary-1
  Content-Type: application/sdp
```

```
  Content-Length: 147

  v=0
  o=UserA 2890844526 2890844526 IN IP4 example.com
  s=Session SDP
  c=IN IP4 pc33.example.com
  m=audio 49172 RTP/AVP 0
--unique-boundary-1
  Content-Type: multipart/signed;protocol="application/
   pkcs7-signature";micalg=sha1; boundary=boundary42
  Content-Length: 608
  --boundary42
    Content-Type: message/sipfrag
    Content-Disposition: aib; handling=optional
    From: Alice <sip:alice@example.com>
    To: Bob <sip:bob@example.net>
    Contact: <sip:alice@pc33.example.com>
    Date: Thu, 21 Feb 2002 13:02:03 GMT
    Call-ID: a84b4c76e66710
    CSeq: 314159 INVITE
  --boundary42
    Content-Type:application/pkcs7-signature;name=smime
    Content-Transfer-Encoding: base64
    Content-Disposition: attachment; filename=smime.p7s;
    handling=required
     hash value
  --boundary42
--unique-boundary-1
```

Listing 8.3: signed subset of SIP headers according to RFC 3839

Listing 8.3 presents two nested multiparts. The first one – delimited by the keyword "unique-boundary-1" – contains the SDP part in the first section and in the second section the signed header subset.

This second segment describes the nested part in the "Content-Type"-header (line 23f) and the name of the nested boundary (`boundary-42`). In the first subsegment, the content is described as "message/sipfrag" and then, the most relevant headers are repeated (line 30 to 35). The second subsegment (beginning with line 38) contains the signature of the first subsegment.

With this methodology, it is possible to prove and guarantee the real identity of the caller in-line in the SIP-signaling. This solution can be used for SPIT-defending techniques. From privacy perspective, the SDP-payload is indeed unprotected and there is a risk of manipulating the RTP sockets for man-in-the-middle attacks.

Two additional drawbacks are worth mentioning: the first (and most important) one is the problem when using *Session Border Controllers (SBCs)* and *Back-to-Back (B2B)-user-agents*. These are designed to manipulate or to completely recreate the SIP request. These changes modify the parameters which have been used for calculating the signature, rendering the signature invalid. The second problem is the low number of devices or applications supporting this RFC. An increasing number of attacks might be an incentive to increase the number of supporting implementations and devices.

RFC 4474 authentication

Section 4.1 presents three approaches to send initial requests to a SIP UA. Via own- and foreign-proxies, solely via foreign proxy and direct addressing. The most problematic approach is directly addressing the proxy of the called domain to forward the request to the callee. From attackers point of view, the benefit of this technique is that this proxy is not able to challenge the caller as the called proxy does not have any information about its identity.

Here, the "SIP identity"(introduced in RFC 4474 [Peterson06]) offers a solution: it introduces two new headers, "Identity" and "Identity-Info". The "identity" header is generated by the callers authenticating proxy after a caller has successfully authenticated itself using the HTTP-digest. As the proxy knows about the identity of the caller (denoted as Alice), it generates this header by combining seven significant headers of the INVITE-request and creates a base64 signature string.

```
sip:alice@atlanta.example.com|sip:bob@biloxi.example.org|
   a84b4c76e66710|314159 INVITE|Thu, 21 Feb 2002 13:02:03
    GMT| sip:alice@pc33.atlanta.example.com|v=0
o=UserA 2890844 2890846 IN IP4 pc33.atlanta.example.com
c=IN IP4 pc33.atlanta.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Listing 8.4: The signed SIP and SDP parameter for RFC 4474 authentication

The headers (listing 8.4) define the caller, the callee, the call-ID and the most relevant headers of the SDP payload to create an effective description of the call. The resulting hash is added as "Identity"-header (listing 8.5). In addition, the "Identity-Info"-header stores the HTTPS-URL where the certificate of the proxy is available.

```
INVITE sip:bob@biloxi.example.org SIP/2.0
Via: SIP/2.0/TLS pc33.atlanta.example.com;branch=z9hG4b
To: Bob <sip:bob@biloxi.example.org>
From: Alice <sip:alice@atlanta.example.com>;tag=1928301
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.example.com>
Identity: hash value
Identity-Info: <https://atlanta.example.com/atlanta.cer>;
   alg=rsa-sha1
Content-Type: application/sdp
Content-Length: 147
 [SDP payload removed]
```

Listing 8.5: The Identity header contains the hash for the RFC 4474 authentication

This extension is a way to "secure" the important parameters of the call (identities and media socket) and offer a viable technique for ensuring the presented identity.

The drawbacks of this technique are in general similar to RFC 3893: problems with B2B-UAs, SBCs and the lack of implementations. To keep a system conform with most of the current clients, a fall-back-mechanism must be offered for RFC3261 compliance.

8.3.2 Interdomain Authentication

RFC 3893 and RFC 4474 offer techniques to transmit trusted identities, but with a limited number of implementations. In contrast RFC 3325 [Jennings02] proposes the `P-Asserted-Identity-header` which can be used to communicate the identity of a caller over a connection between two mutually trusted providers. This approach does not execute any authentication by itself, but relies on the correct authentication done by the partner provider. As an example, this header is used on inter-IMS communication in NGNs.

In the current situation, many providers are working with domain-whitelists implemented in their DNS-service where they store trusted foreign SIP proxies. Only foreign calls to these proxies or requests from these proxies are allowed. As a consequence, this concept reduces the global approach of the Internet-based SIP down to limited sized networks of trusts.

Other providers, like *silverserver.at* or *chello.at* even reduce their SIP-network down to their own domain and allow domain-outgoing or -incoming calls only via their SIP2CS-gateway. This isolated solution does not stop the risk of SPIT, but it reduces the consequences: the provider has one single

point of control (the gateway), where all (but only authenticated) calls will pass through. Local monitoring can detect and alert potential attacks or botnets fast and quite reliable.

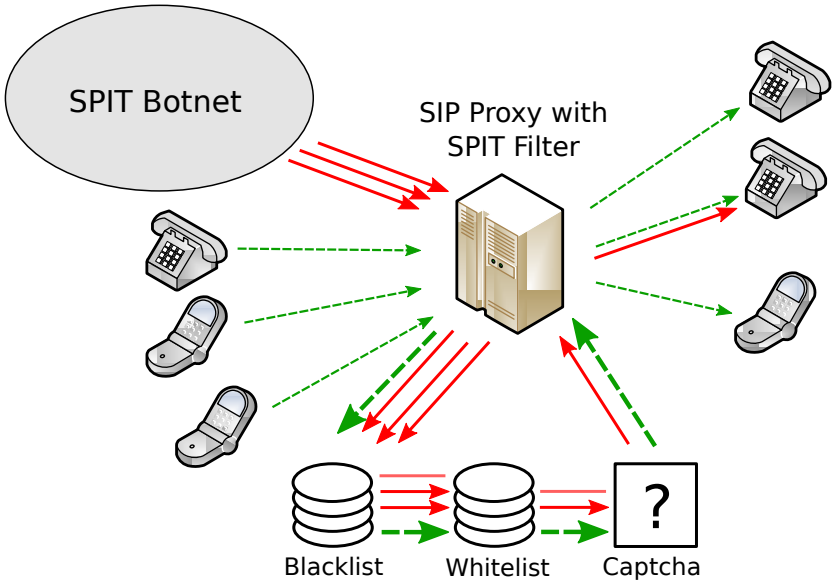


Figure 8.1: Multiple steps analyzing a SIP-request

8.4 Framework for Anti-SPIT Mechanisms

All of the presented techniques have their benefits, but none of them is capable to solve all the SPIT- and mass-call problems in SIP. The combination of several of these techniques plus interaction of the calling partners can improve the systems anti-SPIT performance significantly. Figure 8.1 illustrates how defending solutions can be aggregated to an overall architecture.

Chapter 10 will present such a concept and demonstrate the positive effect of these techniques using traffic samples from two VoIP-providers, A1 Telekom Austria and iptel.org.

Finally, RFC 5039 "SIP SPAM" proposes four recommendations to solve the SPAM-problem.

- **Strong identity**: nearly all of the discussed solutions are relying on strong authentication and reliable identities.
- **White Lists**: when strong identities are implemented, the implementation of whitelists makes sense. Adding to these whitelists can be done by adding the contact manually (like in "friend-lists") or automatically by solving one or more of the introduction mechanisms.

- **Do Not Wait Until It Is Too Late**: RFC 5039 proposes not to wait too long. Providers must find and implement solution as soon as possible. At the moment the risk become real, it needs a time to detect that current problem, to design and to implement solutions. It is unknown how the customer reacts to this threat.
- **Solve the Introduction Problem**: How can white lists get filled with strong identities? RFC 5039 proposes to use several steps for rating new contacts as SPAM originators or not. Figure 8.1 illustrate a number of steps a new request must take to be qualified as "welcome" and to be added to the whitelist

8.5 Summary

Although RFC 5039 is written in 2007 (and published in 2008), the threat of VoIP-SPAM is not wide-spread. There are some minor happenings and attacks recorded in science and media [Darilion08], but the big wave did not arrive until now. It is important not to forget about this threat, the benefits of transmitting SPAM automatically via voice telecommunication at almost no costs are a huge incentive for attackers.

Chapter 9

Use of Email Anti SPAM techniques to rate VoIP Call SPIT-Probability

9.1 Basics

Future VoIP-SPIT calls will likely be distributed by the same organizations and infrastructures as today's Email SPAM. The presented defense solution bases on the same concept: re-use of existing techniques which are successfully in use for defending Email SPAM.

Section 8.1 discusses a number of non-intrusive anti-SPIT solutions which mostly rely on the concept of white- and blacklisting. One technique for acquiring Email SPAM sender addresses is the "honeySPAM" approach, which is presented in the paper *HoneySPAM: Honeypots Fighting SPAM at the Source* [Mauro05]. Here the authors present virtualized open SMTP relays for SPAM harvesting.

Another technique is the use of special Email-addresses placed by providers on websites. These Email-addresses are solely deployed to be found by automatic SPAM address-harvesters. For example, an Email-address in white color on white ground on a website can be discovered by harvesters, but usual users cannot see and accidentally use these addresses. Consequently when Emails arrive at these accounts, these Emails can be qualified as SPAM and therefore the sending hosts classified as SPAM sending node. This host is then stored in the blacklist and future Emails from this node will be blocked (for a specific time).

Sections of this chapter have been previously published as "Using E-Mail SPAM DNS Blacklists for Qualifying the SPAM-over-Internet-Telephony Probability of a SIP Call" [Hirschbichler09].

The presented techniques are two examples to fill blacklists with the IP-addresses of SPAM sending nodes.

9.2 Technical Background of SPAM Blacklists

Blacklisting is implemented either by large providers or by nonprofit organizations, which store IP addresses or entire IP address subnets. The size of these lists and the duration how long an IP address is blacklisted varies strongly depending on the providers. Additionally the rules how an address is added (or how addresses are removed) is not standardized.

Similar to all of these providers is the technique how blacklists can be polled: The blacklists are acting as DNS servers, which can be queried in nearly real-time. The Realtime-DNSBL (RT-DNSBL) has been initially designed to store the IP address of open relay SMTP server. As the SPAM source moved away from open relay SMTP server to distributed networks of hijacked computers (botnets), these DNSBL servers are now used to store IP addresses of hijacked hosts.

9.2.1 Querying DNSBL Servers

Defending Email-SPAM using SMTP is facing the same challenges as voice-SPIT and SIP: It is possible to fake most of the headers to hide the real originator of the SPAM message. For SPAM-blacklisting, the most reliable SMTP-header is the `Received`-Header, which is added at each relaying MTA. This header contains the hostname or IP address of the sending and the receiving host.

Listing 9.1 depicts two hops inside the Vienna University of Technologies (VUT) Email infrastructure.

```
Received: from mail1.zserv.tuwien.ac.at (mail1.zserv.
   tuwien.ac.at [128.130.35.37]) by mri3.kom.tuwien.ac.at
   (8.14.2/8.14.2) with ESMTP id r27Dm6UF015379; Thu, 7
   Mar 2013 14:48:06 +0100
Received: from [128.131.88.xxx] (xxx.ibk.tuwien.ac.at
   [128.131.88.xxx]) (authenticated bits=0)  by mail1.
   zserv.tuwien.ac.at (8.13.8/8.13.8) with ESMTP id
   r27Dm6fZ008900 (version=TLSv1/SSLv3 cipher=DHE-RSA-
   AES256-SHA bits=256 verify=NO);
```

Listing 9.1: Two SMTP-Received header

For querying a DNSBL server, the IP address of the sending host is extracted, reversed and added as a 'virtual' hostname to the base-address of the DNSBL server. If the sending IP is `a.b.c.d` and the DNSBL-server is `blacklist.net`,

the querying Email-Server does a DNS lookup for the `A` record of the name `d.c.b.a.blacklist.net`. This concept is similar to ENUM-queries presented in section 5.2.1.

9.2.2 Results of DNSBL Queries

The DNSBL-server returns a "virtual"-IP address (e.g., `127.0.0.1`) when the IP address `a.b.c.d` is already blacklisted. If it is not, the DNSBL-server returns as response code a `Name Error` reply.

9.2.3 Parsing SIP Messages

To query the e-mail blacklists for IP addresses and hostnames extracted from SIP and SDP (RFC 4566 [Handley06]) messages, relevant and reliable IP-addresses are needed. This information can be found in the following portions of a SIP message:

- **`Via` headers:** at least the topmost `Via` header on the incoming request must be valid to create a SIP Session. If the request is passing an incoming proxy, this proxy also adds an additional Via-Header. Here, the system must be aware not to add the local proxy to the blacklist. This can be avoided by creating an initial short whitelist containing the trusted SIP proxies.
- **`Contact` header:** the contact header must be present and contains the URI for subsequent requests. This contact header might be replaced by masquerading SIP proxies offering a B2BUA-functionality. Like for Via-headers, the blacklisting functionality must be configured to avoid this misbehavior.
- **SDP-message:** IP addresses in the SDP header are located in the `c=` field where the connection information for establishing the audio-stream is stored. This IP address is important for SPIT-types where the calling party is human and tries to establish a dialog with the caller. If the SPIT call is just a prerecorded audio-file, this IP address could also be faked. Faking the SDP-IPs is indeed unlikely, as most called customers are behind NAT-gateways. To transfer RTP-media to a host behind a NATed network, RTP packets must be sent in both directions to create an entry in the NAT table (*Symmetric RTP*, RFC 4961 [Wing07]).
- **The source IP:** this IP address is valuable for checking against blacklists. Similar to the Via-header analysis, the monitoring component must be aware of the provider's IP-addresses: if the SPITter uses the customer's proxy, the source IP is always the IP address of this proxy and must not be blacklisted.

Compared to e-mail, where only the `Received` header is useful to be compared with DNSBLs, the SIP/SDP message offers several enclosed and mandatory valid IP addresses and hostnames. Using this sender information, the effectiveness of a blacklist might be better for rating SPIT than for rating SPAM.

If the gained information is not an IP-address but a hostname, additional DNS-lookups are required to resolve the hostname.

9.3 SpitAssassin – a modular SPIT-Filter

The Apache-licensed SpamAssassin[1], introduced in 1997, is a modular Perl application used by Email server providers to analyze and mark incoming Emails. It uses several tests like

- static content filters using regular expression for typical words used in SPAM
- creating hash-checksums of the content and querying remote databases for this hash.
- using Bayesian filters to compare incoming Email-content with already analyzed old Emails.
- using DNSBLs for querying IP addresses displayed in the `Received:`-Header.

To increase the performance of SpamAssassin, it works as a daemon listening by default on port 783 for incoming connections. When receiving incoming Emails, the Email-system forwards these Emails to the SpamAssassin-client `spamc` which forwards the message to the SpamAssassin daemon. After parsing and analyzing the Email, `spamc` receives the e-mail with an optionally added header and the SPAM-probability-score of this e-mail as response.

9.3.1 Extending SpitAssassin to cooperate with OpenSIPS

Most of the tests executed by SpamAssassin are also usable for defending SPIT. A prototype has been implemented, which extends (and renames) SpamAssassin to interface with OpenSIPS and parse and modify SIP headers instead of SMTP headers. The first step was to implement a module for OpenSIPS (`spit.so`) to act as connector to `spamc` and to correctly add the returned SPIT-marking header to the SIP message. The next step was extending SpamAssassin and its modules to correctly break up the header and modify the header inspecting modules to correctly handle SIP headers.

[1] SpamAssassin, `http://SpamAssassin.apache.org`

9.3.2 Coupling SpitAssassin with OpenSIPS

To combine SpitAssassin with OpenSIPS, the `spit.so` module is activated in `OpenSIPS.cfg`. To use the module, the configuration is extended to filter specific arriving requests for their SIP method (listing 9.2 inspects INVITE and MESSAGE requests).

```
loadmodule "spit.so"
if ((method=="INVITE")||
    (method=="MESSAGE")){
        sl_send_reply("100", "Trying");
        spit_check("127.0.0.1");
}
```

Listing 9.2: Filter SIP requests for INVITE and MESSAGE methods and apply the spit_check mechanism to these messages

On arriving of an INVITE- or MESSAGE-request, a `100 Trying` is sent as provisional response to stop or delay retransmissions (see Figure 9.1). This is needed because SpitAssassin increases the call setup delays due to its complexity. The provisional response does not stop retransmissions of non-INVITE requests but it slows down retransmissions to timer T2 (4s, RFC 3261, 17). The use of this response to non-INVITE requests is not recommended by the RFC 3261, but it helps to reduce the retransmission traffic. The method `spit_check` forwards the incoming request to the `spamc` component. The only parameter of the spit_check method is the IP-address of the SpitAssassin-server. `spamc` opens a connection to the SpitAssassin host, where the request is analyzed and the SPIT probability is calculated. The `spit_check` method adds then an additional header `XSPAM: xxx`, where `xxx` is the calculated score.

The higher this score, the higher is the probability, that this SIP-request contains SPIT or is part of a SPIT-call. When the value exceeds a preconfigured threshold, the SIP-proxy may alter the `Subject` or the `From` header to alert the called party by displaying the SPIT-warning on the users phone.

9.4 Performance Measurements and Results

To gain information about the additional delays SpitAssassin in connection with DNSBLs creates, an OpenSIPS SIP router is configured with several varying configurations. The resulting Call Setup Delays (CSDs) are then compared on the caller's host.

Target is to prove, that SpitAssassin and DNSBL-lookups add an additional, but acceptable delay to the call setup delay.

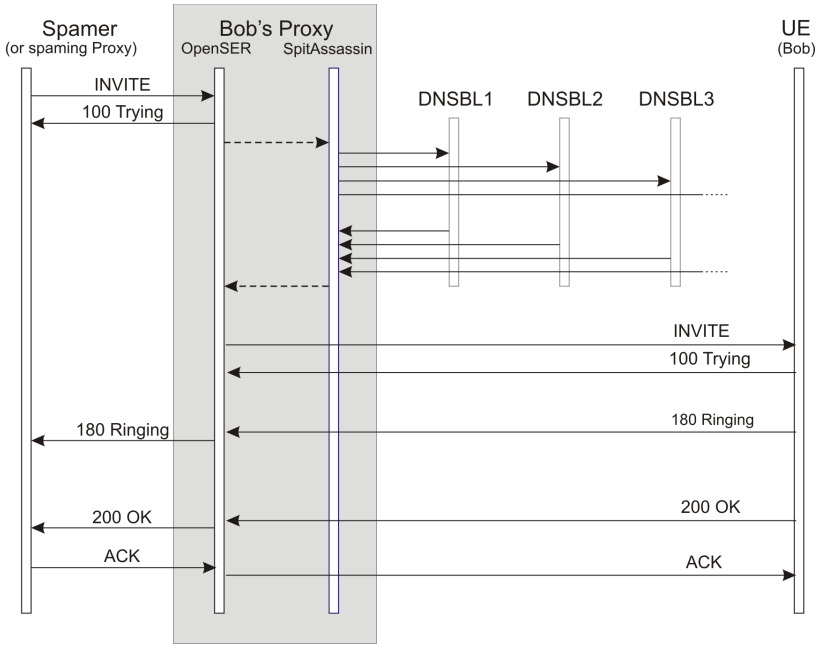


Figure 9.1: SpitAssassin querying several DNSBLs

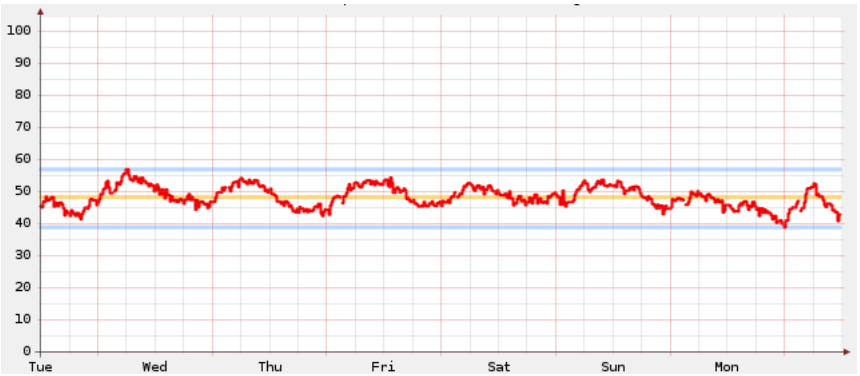


Figure 9.2: Hit-rate of the NiXSPAM Email DNSBL Server used for our Tests

9.4.1 Used Blacklist Server

The German publisher Heise introduced a blacklist and a content-hash database[2] in 2003. The most current 40 000 entries of this blacklist are available for download as text-file and will be used for the performance tests as input data for the Via-header. For each test-call several IP addresses out of this list are taken to avoid caching of DNS queries.

The duration IP addresses are stored in the NixSpam blacklist is three days and there are about 400 000 IPs stored. Figure 9.2 (provided by heise)

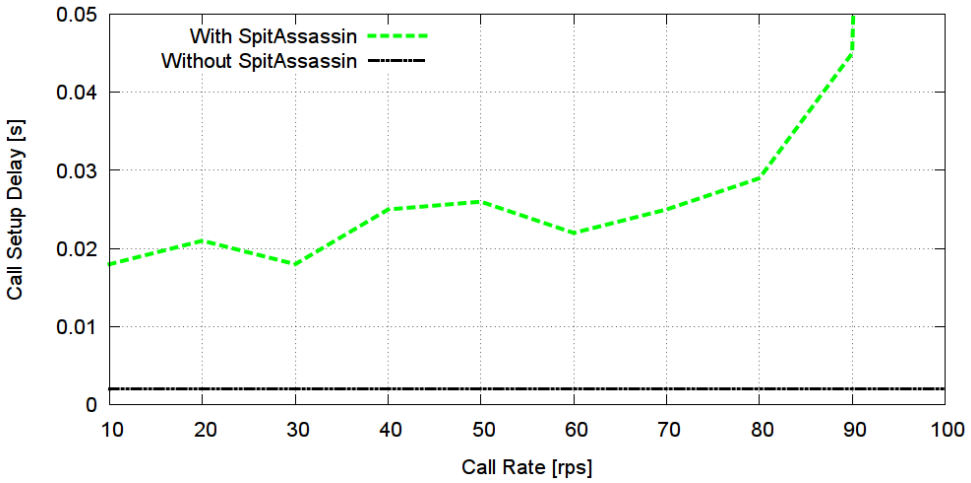


Figure 9.3: Comparing call-setup-delays of OpenSIPS with and without SpitAssassin (no DNSBL queries)

presents the success-rate of DNSBL queries against the NiXSpam project in 2008. In median, every second IP-address queried using the DNSBL results in a successful response.

9.4.2 Multiple Test Runs

To evaluate our approach, we implemented a set of test-runs.

1. The first test-run uses OpenSIPS with default settings and acts as a performance reference. No SpitAsssin specific changes are made to the configuration.
2. The second test-run activates SpitAssassin by loading the `spit.so` module.

[2]DNSBL NiX SPAM, `http://www.heise.de/ix/nixSPAM/`

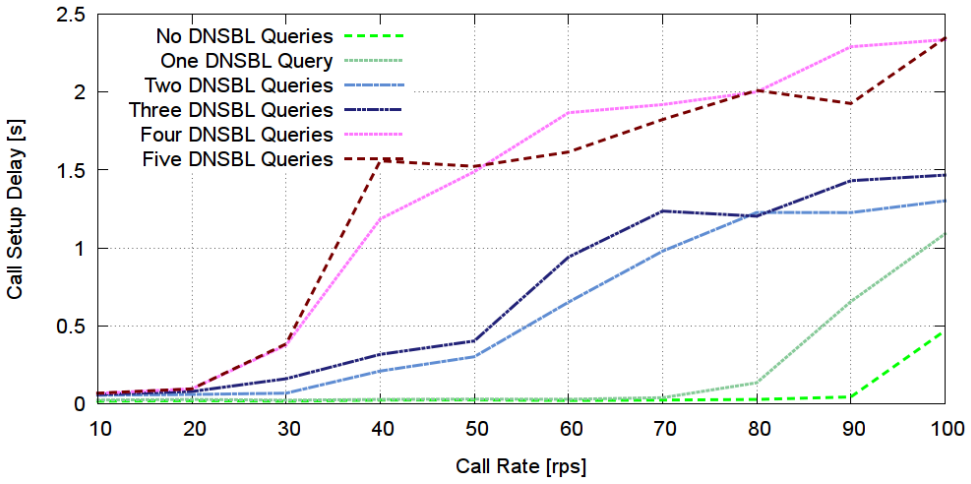


Figure 9.4: Comparison of call setup delays depending on number of DNSBL lookups and call rates

3. The third test-run is a collection of tests and executed against OpenSIPS with activated SpitAssassin *and* DNSBL lookups.

   As we are expecting increasing latencies with each additional Via-header in the analyzed SIP-request, the third test-run is divided into five sub tests. The first test analyses the DNSBL-lookup latency when the request contains only a Via-header with the IP-address of the sending client (`own_ip`). The other sub tests evaluate the latency with an increasing number of faked Via-headers per received request.

   To execute this test, the call-generator *sipp* adds Via-headers with addresses from the downloaded list of 40 000 blacklisted IP-addresses (denoted in Listing 9.3 as `bl_ip`).

   For example, the third test-run has the Via-header layout as described in Listing 9.3:

```
Via: SIP/2.0/UDP own_ip:5060;
Via: SIP/2.0/UDP bl_ip1:5060;
Via: SIP/2.0/UDP bl_ip2:5060;
```

Listing 9.3: Example Via-header layout

   In fact exactly n-1 headers are hits, where n is the amount of `Via` headers of a message.

All of these test runs had a duration of five minutes and created between 10 and 100 INVITE requests per second (rps). When considering the test with four additional `Via` headers, the SpitAssassin module created up to 500 DNSBL queries per second or 150 000 DNS queries over the time span of five minutes.

9.4.3 Test results

This section presents the performance numbers and success rates of the tested implementation.

Delays caused by SpitAssassin

Figure 9.3 depicts median call setup delay for the original OpenSIPS and the modified OpenSIPS with activated SPIT module (combined with SPIT-Assassin). In the second configuration, SpitAssassin is set up without any filtering or analyzing rules. The median of unfiltered call setup delay is steadily 2 ms and not influenced by the increasing load. Call setup delay with SpitAssassin increases by a factor of 10 to 15 for medium load up to 80 handled rps. Increasing the load even more causes CPU performance limitations, which result in exponential increase of call setup delay.

Delays caused by DNSBL Queries

The following test-case compares delay of one to five DNS queries per call. SpitAssassin handles several subsequent DNS lookups of IP addresses in an effective way. At a call rate of 10 cps there are no remarkable differences

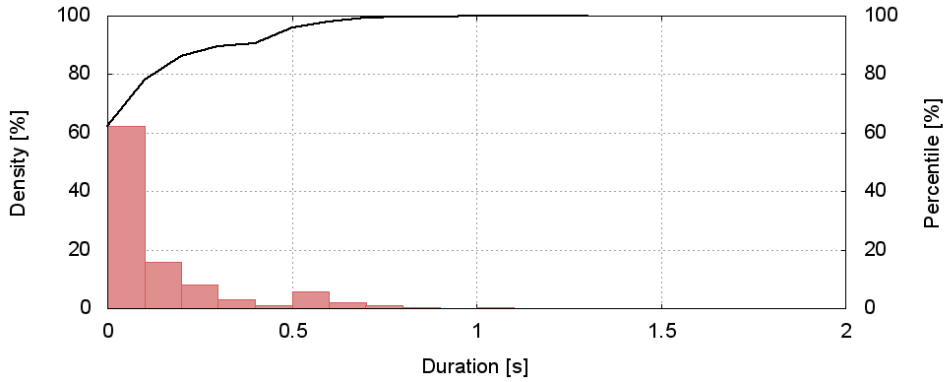


Figure 9.5: Call-setup-delay distribution of a 30 cps test run with two DNSBL-lookups per call

between the amount of DNSBL lookups per call. The delay of all five test

runs (one to five DNSBL-lookups) is in median between 40 and 50 ms. This changes when increasing the call rate to 20 cps. Two lookups per call require in median 40 ms additional delay and three lookups per call in median 60 ms. The higher number of DNSBL lookups per call increase the call setup delay to 80 ms, both for four and five DNSBL lookups.

A remarkable split between two and three lookups on the one hand, and four and five lookups on the other hand can be seen at call rates higher than 20 cps. The split is possibly caused by the high load on the dual core CPU of the SpitAssassin server. Focusing on an individual test-case, Figure 9.5 depicts a

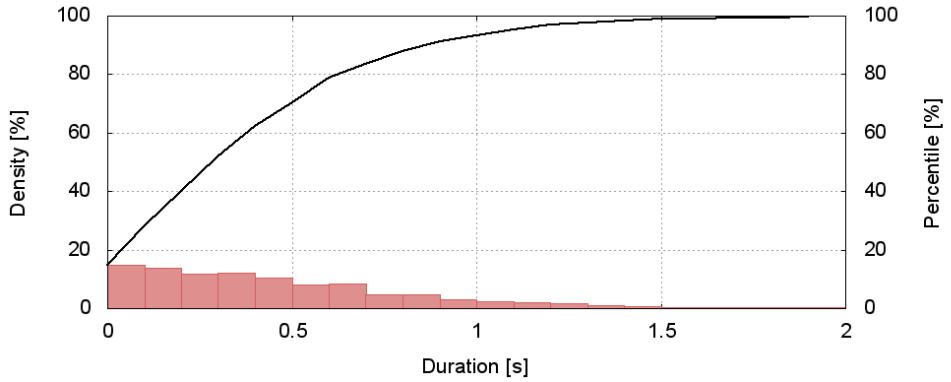


Figure 9.6: Call-setup-delay distribution of a 30 cps test run with five DNSBL-lookups per call

five-minute-test-run at 30 rps with two DNSBL lookups per call. Nearly 80% of all calls are successfully set up within 0.1 s. All call attempts succeed within 1.2 seconds.

Figure 9.6 presents the same test scenario, but with five DNSBL lookups per call. In this case, the 80% setup rate is reached after 0.6 s, all calls are successfully set up after 2 s.

Concluding, the processor capacity is the most limiting factor. Increasing the CPU capacity or the number of CPUs will decrease the call setup delay.

Finally, the median request delay at 100 cps is still lower than 2.4 s even with five DNSBL queries, which is in fact an acceptable delay and below the values recommended by ITU E.721 [itu99](for ISDN-calls): ITU recommends an average delay of not more than 5 s for non-local calls.

Success Rate of Call Setups with DNSBL Queries

Success rate of the various test runs is proportional to the call setup delay (see Figure 9.7). The tests with two, three, four and five DNSBL lookups again

are grouped together, which might be related to the dual CPU system.

Successful call setups are defined in this analysis as call setups, which were successfully answered with a `200 OK` message within 20 seconds. Figure 9.7 illustrates, that the system is beginning to drop calls at a call rate higher than 70 cps at one DNSBL lookup per call or at call rates higher than 20 cps when issuing four or five DNSBL lookups per call.

Concluding the success rate aspect, when observing and checking a trend to more `Via` headers or parsing other IP addresses in the SIP request, the system's hardware must be extended or a maximum amount of analyzed IP addresses per message must be defined. Considering a typical incoming SIP

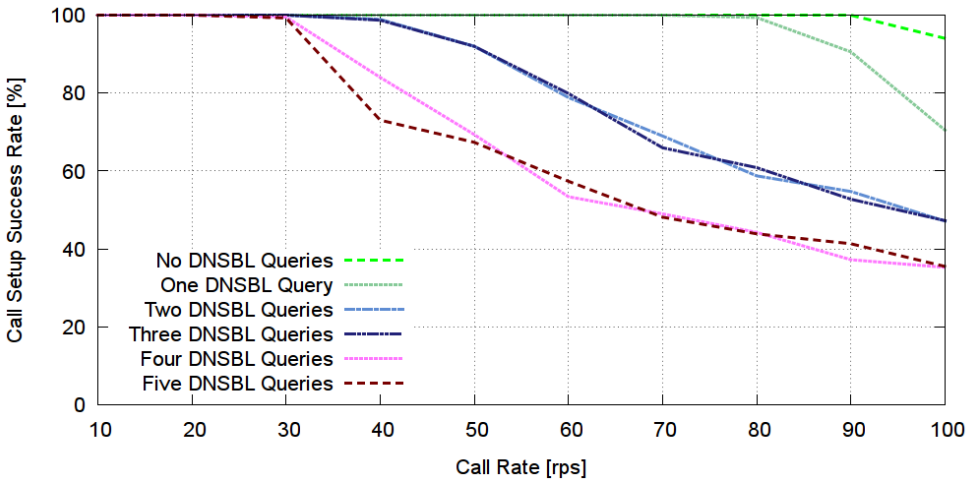


Figure 9.7: Comparing call setup success rates with different numbers of DNSBL lookups.

request, five IP addresses are relevant: the first and the second Via-header, the contact header, the SDP `c=` header and the source IP.

9.5 Conclusion

The presented concepts reuse already established e-mail SPAM DNSBLs for the detection of VoIP SPIT. Assuming that professional SPITter will use their Email SPAM-infrastructure also for Voice-SPIT, the popular SPAM filter system *SpamAssassin* has been modified to parse SIP signaling. The measurements presented in this section depict that this filter is able to perform SIP analysis in an effective way. The results also illustrate, that the use of DNSBLs for one or more checked IP addresses per call does not increase the call setup delay to an unacceptable value as long as the CPU power is adequate.

Chapter 10

Session Border Controller Advanced (SBC-A)

This section introduces an enhanced *Session Border Controller (SBC)* which is able to detect high-rate SIP DoS attacks and to mark all forwarded requests with a value indicating the "quality" of the request. This score, denoted as "dropability" and introduced in section 7.3.2, reflects the effort the system has already invested for this request and the expected value for both the operator and the customer. This dropability value depends amongst other presented factors on the SPAM-probability and the economic- or QoS-effect of the respective request.
This dropability score supports overloaded core-components to decide with a minimum of processing effort, which requests should be dropped first and which requests have severe effects on the customer's perception or the operator's economic income.

10.1 Introduction

The term SBC is not explicitly specified but already used by many VoIP providers to denote a perimeter security function. At the edge nodes of the providers core infrastructure, network policies are typically enforced using this SBC. The informational RFC 5853 [Hautakorpi10] summarizes the common functionalities and highlights three SBC-specific areas:

1. perimeter defense (...)
2. functionality not available in the endpoints (...)
3. traffic management (...)

This chapter is extending the publication "Stop the Flood – PerimeterSecurity- and Overload-Pre-Evaluation in Carrier Grade VoIP Infrastructures" [Hirschbichler12].

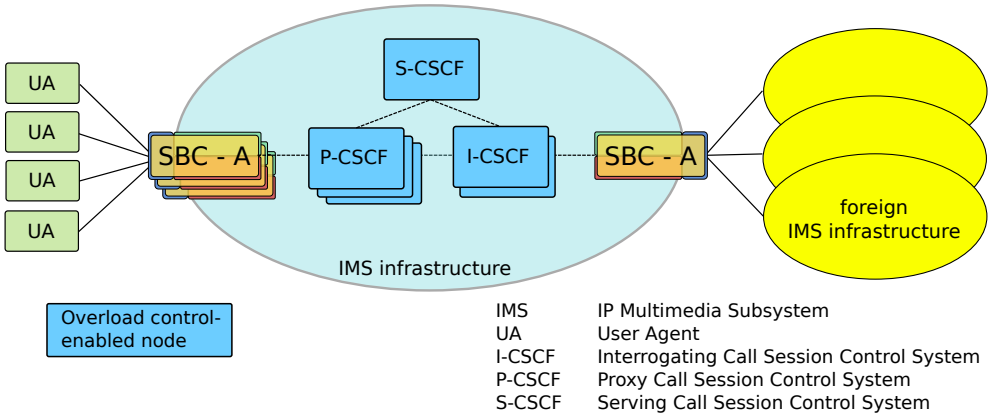


Figure 10.1: Multiple SBC-A' protecting an overload-controlled enabled IMS infrastructure both from access net and from peering partners

The presented concept extends the areas "1)" and "3)" by adding overload control pre-evaluation and -marking. To differentiate between the "typical" SBC and the presented extended SBC, the later one is denoted as *Session Border Controller - Advanced (SBC-A)*.

Figure 10.2 depicts the architecture of the proposed SBC-A, which consists of multiple nested stages. With each stage, the intensity of traffic inspection increases at the cost of additional CPU load and increasing latency.

First, two categories **Unknown Relationship (UR)** (blue frame in figure 10.2) and **Known Relationship (KR)** (green and red frames in figure 10.2) are defined where the KR is split into *Trusted Relationship* (green frame) and *Untrusted Relationship* (red frame). An incoming request passes the two categories (*UR* and *KR*) and the following three nested analyzing groups:

1. Denial of Service (DoS)-detection and -protection
2. Unsolicited Communication (UC)-detection
3. Overload Specific Request Rating

Depending on their classification all inspected requests are either rejected, dropped or forwarded with a `X-dropability`-header as introduced in subsection 7.3.2. The next sections present these three groups and show the results of using the overload-control marking-headers.

10.2 Reference Traffic from A1 Telekom Austria and *iptel.org*

To prove the usability of the presented concept, traces from A1TA's *A1 Network Unified Voice Service (A1NUVS)* and *A1overIP* service are compared against traces of *iptel.org*'s VoIP-service.

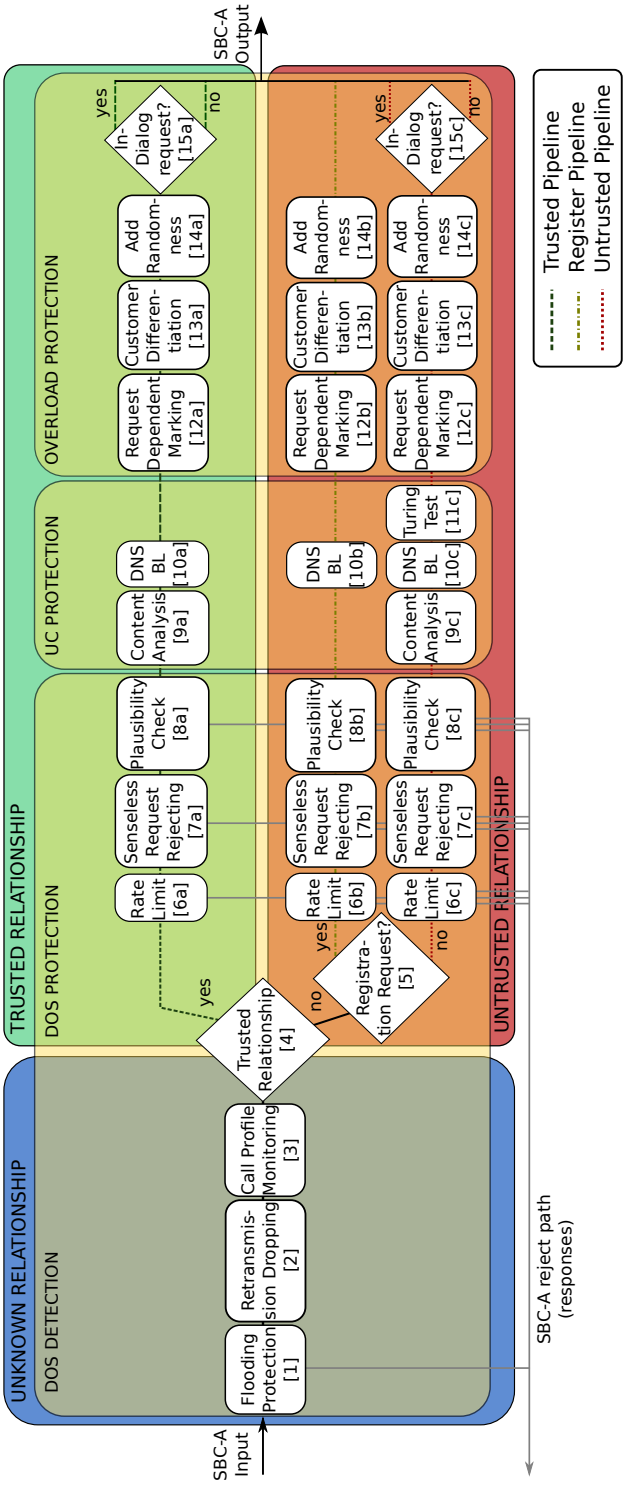


Figure 10.2: Detailed step-by-step processing of SBC-A

Section 3.2 presents the traffic profiles of the *A1 TA*- and the iptel-services and discusses the mutualities and differences. These traces have been recorded on Q2/2012 (iptel), resp., on Q4/2012 (*A1 TA*). For comparability, both traces start at 11:35 and end at 13:55. All *iptel.org* requests are analyzed, whereas some of *A1 TA*'s log records have been removed. The reason for this pre-selection is that *A1 TA*'s A1NUVS and A1overIP offer multiple calling paths:

- *SIP-outgoing calls* from the SIP to the CS-domain (SIP2CS),
- *SIP-incoming calls* from the CS to the SIP domain (CS2SIP)
- *SIP-internal calls* (SIP2SIP)

As the SBC is dedicated to protect the core against packet networks, only SIP requests incoming from external interfaces are inspected and the calling path CS2SIP is ignored.

10.2.1 Differences between *iptel.org* and *A1 TA* traffic

iptel.org:

As discussed in section 3.2, ***iptel.org*** offers a location transparent SIP registrar service open to a global market. With this approach, the traffic does not show distinct peak busy hours as visible in figure 3.2. Its customers are using a wide range of clients and exhibit possibly distinct usage behaviors. Additionally, a high ratio of mobile SIP-clients are used which create a high amount of retransmissions probably caused due to high latencies in mobile 2G/3G networks and other factors: 22.34% of all requests transmitted by mobile clients are retransmissions. With fixed clients, the retransmission ratio is lower at 7.08% in the *iptel.org*-network [Fabini13].

A1 TA's

solution is a value added service for Telekom Austria's customers. With this service, customers get a free SIP soft-client ("Counterpath Bria"[1]), a price-reduced hardphone (Snom[2]) or an ADSL router with an embedded SIP Private Branch Exchange (PBX)(AVM FritzBox[3]). However, all other RFC 3261-compliant SIP terminals are accepted, too.

A closer inspection of the SIP request distribution between *iptel.org* and *A1 TA* illustrates differences, which can be explained by the different service profiles between *A1 TA* and iptel. First, in both infrastructures the INVITE-transactions are negligible. Only 0.19% (in iptel-traffic, figure 10.4) and 0.53% (in *A1 TA*-traffic, figure 10.3) are INVITE requests. Instead, the main part

[1] http://www.counterpath.com/bria.html
[2] http://www.snom.com
[3] http://www.avm.de

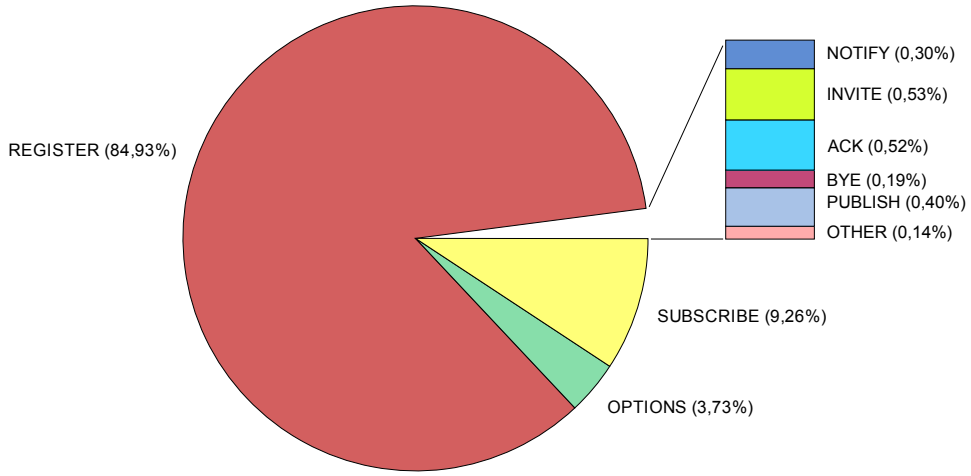


Figure 10.3: *A1 TA* request-distribution during the analyzed time interval (sample size 1 149 447 requests)

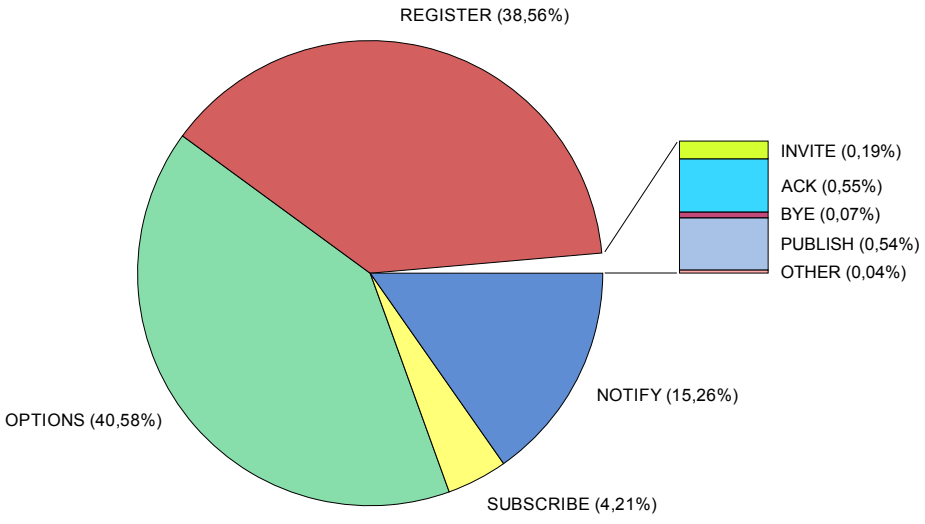


Figure 10.4: *iptel.org* request-distribution during the analyzed time interval (sample size 833 257 requests)

of all requests is shared between REGISTER- and OPTIONS-requests in *iptel.org*-traffic, resp. is contributed by REGISTER-requests in *A1 TA*-traffic.

The high amount of REGISTER requests in *A1 TA* traffic compared to *iptel.org* is caused by the NAT-helping mechanism. *A1 TA* reduces the default re-REGISTER timer to 60 s (compared to the usual 3600 s as proposed in RFC 3261, 10.2). The high frequency of REGISTER requests results in a continuous refresh of entries in the source-NAT tables of the Customer Premises Eqipments (CPEs). With this technique, the clients behind a NAT are able to receive SIP requests without configuring helper protocols like *Session Traversal Utilities for NAT (STUN)* [Rosenberg03] or *Traversal Using Relays around NAT (TURN)* [Mahy10].

iptel.org does not force this "short re-registration" approach from operators side. The high amount of REGISTER- and OPTIONS-requests is an indicator, that the customers (resp. their SIP-clients) are using the continuous sending of REGISTER- or OPTION-requests to achieve the same objective.

From operators point-of-view, the OPTIONS-requests are not generating any positive effects, but create additional traffic and processing load. Solving the NAT-problem by switching to IPv6 might reduce the ratio of these two supporting requests.

Summing up, the usage profile of the two compared services differ. *A1 TA* offers a geographically bound service focused on central Europe compared to a global service offered by *iptel.org*. Similar on both traces is the small amount of user generated traffic (like voice calls) compared to machine created traffic. The following sections introduce the SBC-A and illustrate the influence of each of the steps of the advanced SBC on the traffic of *A1 TA* and *iptel.org*.

10.3 DoS-Detection and -Protection

Blocking of high-rate attacks does not generally stop the threat of processing congestion in complex VoIP infrastructures. Instead, there are multiple factors, which can together lead to a high- and overloaded system.

RFC 5390 [Rosenberg08a] discusses six reasons for overload, but only one of them can be blocked by simple "high-rate attack prevention" mechanisms. These six reasons which can be grouped into operator originated overload like

- "poor capacity planning"
- "dependency failures"
- "component failures"

and external initiated overload like

- "avalanche restart of clients"

- "flash crowd of multiple users simultaneously creating a call"
- "DoS from one or multiple sources" [Rosenberg08a]

Only *DoS from one single source* can be blocked by source-port-blocking, all other reasons for overload stem from multiple sources or internal failures. A combination of the algorithms proposed in this chapter will support the system in marking (and potentially blocking) suspicious and/or unimportant requests.

10.4 DoS-Detection

The analyzing steps for DoS-detection are split into two categories *Unknown Relationship (UR)* and *Known Relationships (KR)* as shown in figure 10.2 and presented in section 10.1. The first DoS-protecting steps are executed in the UR-category. The analyzing application does not differ, whether requests are arriving from a known or from an unknown host.

10.4.1 Stage 1: Flooding Protection

The flooding protection (Figure 10.5, stage 1) defends the system against attacks by one malicious client. In this case, the attacker tries to oversaturate the system by sending a high number of requests in a short time interval. Using a sliding window algorithm (proposed in [Liec]), the SBC-A records the amount of arriving SIP requests from one single socket within a fixed period. When the number of requests exceed a predefined threshold, the sending socket is blacklisted and all subsequent requests are blocked with a specific SIP response, e.g., *403 Forbidden*. The blacklisting-duration of the affected sockets is defined using the configuration parameter *bts* described in Table 10.1.

All other requests which are not blocked but forwarded to the next analyzing step are marked with the `X-dropability`-header and the initial score "0".

Known Drawbacks

Filtering by sockets is under most circumstances an effective approach, but SIP-traffic generators (namely SIPp[4]) can modify the sending port on a per-request-base. If a perimeter security node receives a high number of requests from the same IP, but from varying ports, it is unclear, whether the sending node is a NAT-gateway (with e.g., an avalanche rebooting SIP network in behind) or a malicious attacker.

There is no definite solution to this challenge. Hence a proposal is to define a low upper limit of requests per IP address (disregarding the source port) and

[4] http://sipp.sourceforge.net

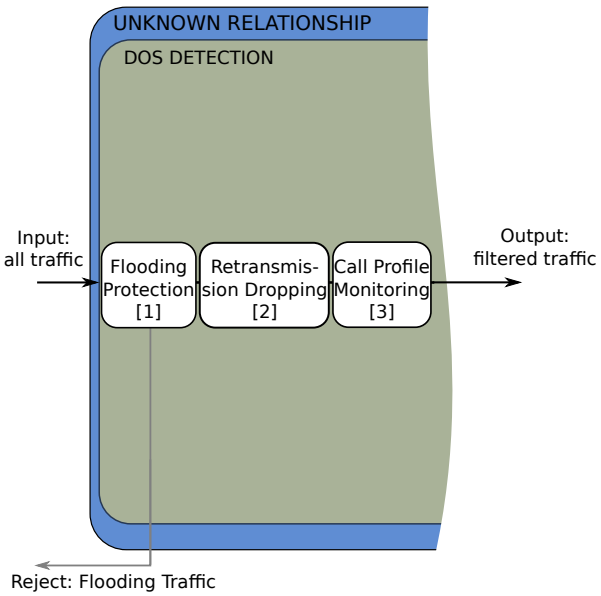


Figure 10.5: Denial of service-protecting section of the SBC-A. The first steps do not differ between trusted and untrusted relationships

maintain a whitelist with IP addresses allowed to transmit more requests per time interval. In this whitelist well known customers and their IP-addresses are stored. These customers are usually larger companies or companies with above-average call amounts like call-centers or poll institutes, all with fixed IP-addresses on their NAT-gateway.

Configuration Parameters

Three parameters presented in Table 10.1, support accurate operation of DoS filtering. These values are configured based on operator's policies and learnings to find appropriately balanced values.

Results

With the traces of A1 Telekom Austria and *iptel.org* (introduced in section 3.2), it is possible to analyze the effect of the presented mechanism. The limit of 20 rps is applied both to *iptel.org* and *A1 TA* with the option of adding contacts to a whitelist.

parameter	description	example value
mps	Allowed messages per second	20 rps
mpsw	Allowed messages per second of white listed clients	50 rps
bts	Blocking Time [s]	1 min
Λ	List of white listed IP addresses	undef

Table 10.1: Configuration parameters for flooding protection (Stage 1)

All request types	*A1 TA* - Traffic	*iptel.org* - Traffic
Initial Traffic	151.26 rps	106.83 rps
Traffic Reduction	3.24%	0.29%
Remaining Traffic after Stage 1	146.36 rps	106.52 rps

Table 10.2: Stage 1: Number of reduced requests after flooding protection

Only *INVITE* requests	*A1 TA* - Traffic	*iptel.org* - Traffic
Initial Traffic	2.26 rps	0.198 rps
Traffic Reduction	1.97%	0.13%
Remaining Traffic after Stage 1	2.21 rps	0.197 rps

Table 10.3: Stage 1: Ratio of reduced INVITE requests after flooding protection

A1 TA

A high INVITE-load of 20 rps (in a five minutes average) from one single socket is observed when analyzing the *A1 TA* trace. Due to the high amount of created traffic, all requests from this host (probably a polling/marketing institute) were blocked until the introduction of the rps-whitelist including this IP address.

This high number of legitimate requests from one socket confirms the assumption, that there is a strong need for specific, administrator-revised whitelists. In this specific case, an extension of the maximum accepted amount of 10 rps up to 40 rps for this IP address shows significant improvements of the acceptance rate.

While the average traffic per socket is below the accepted 20 rps, some clients create peak-numbers of requests which result in a total reject rate of 3.24% as presented in Table 10.2.

iptel.org

For the *iptel.org*-traffic, the same limiting parameter of 20 requests per socket proves to be more effective, as only 0.29% of all requests are rejected (Table 10.2).

Considering only INVITE-requests, the block-rate about 0.13% (or 2 of 1540 INVITE requests, figure 10.2).

10.4.2 Stage 2: Retransmission Dropping

Section 7.3.1 and [Egger10] discuss the problem of retransmissions introduced by the T1 retransmission timer in RFC 3261. The request analysis in section 10.2.1 revealed that a high ratio of SIP request retransmissions occur in live networks. In particular the *iptel.org*-traffic contains a huge number of retransmissions, close to 25%.

In this stage of SBC-A-processing (Figure 10.5, stage 2), the *Message-Digest Algorithm 5 (MD5)*-hash [Rivest92] of each incoming request is stored in a table. Retransmissions are then detected by comparing the MD5-hash of an incoming request with MD5-hashes of already received requests. If a retransmission is detected, it is silently discarded.

Results

Dropped retransmissions account for the largest portion of reduced traffic during the entire SBC-A processing in non-congested state. The high amount of retransmissions in the *iptel.org*-trace (compared with *A1 TA*) is caused by the high ratio of mobile clients. The round-trip time of mobile clients registered in 2G/3G networks (in combination with the *iptel.org* core network delay of about 80 ms for REGISTER-requests and the worldwide distribution of the clients) exceeds the default T1-timer of 500 ms (RFC 3261, Annex A), resulting in retransmissions [Fabini13]. Extending the timer to a higher value (e.g., 1 s) could reduce the traffic massively without further negative impact. A proposal on modifying this timer has been presented in subsection 7.3.1.

In contrast, the *A1 TA* trace contains only 8% retransmissions. The reason for these retransmissions is unclear, possible reasons including, but not being limited to losses, a mixture of mobile-clients and high latency fixed line connections. Compared to *iptel.org*, the *A1 TA* core-side processing delay (measured using the PARIS-tool, section 6.2) is due to the complexity of the IMS architecture 20% higher for registration requests and by 100% higher for call setups.

	***A1 TA* - Traffic**	***iptel.org* - Traffic**
Traffic before Stage 2	146.36 rps	106.52 rps
Traffic Reduction	8.21%	23.70%
Remaining Traffic after Stage 2	134.33 rps	81.27 rps

Table 10.4: Stage 2: Reduced request-rate after retransmission blocking

10.4.3 Stage 3: Call Profile Monitoring

For networks which service mainly customers of limited geographical area, the number of received SIP requests shows recurring patterns with periods of 24 hours and one week. In particular user-originated requests like INVITE, MESSAGE or NOTIFY are emphasizing the busy hour profile: during working time, the amount of originated INVITE requests are higher than during nighttime. This problem is discussed in section 3.2.

Additionally, repeating peak hours occur in automatically generated requests: the amount of REGISTER requests might be varying by the number of active clients: hard clients are typically always on, but soft clients on office computers might be off-line during weekends or nighttime, resulting in a reduced number of REGISTER and re-REGISTER requests. In future M2M-environments, peaks are possible when i.e., many vending systems are sending UPDATE requests to the supplier. Such profiles differ from provider to provider and from time to time and must be adapted by a self learning system on the fly. This self-learning-process can be executed by monitoring the SIP traffic in-line, by counting the processed requests or – if there is no such interface accessible – implicitly by monitoring the call-setup and request-delay like proposed in the PARIS-monitoring tool (section 6.2).

In the presented concept, the SBC-A should use a self learning per-request-type system (Figure 10.5, stage 3). If the number of requests arriving within a specific sliding window interval exceeds the expected and "learned" amount of requests, the `X-dropability`-score of all requests of the affected type will be set to the exceeding factor.

When using the PARIS-system, the Session Request Delay (SRD) (defined in [Malas11]) presented by PARIS is a useful indicator and basing on this SRD-value, the `X-dropability`-increase can be calculated.

Configuration Parameters

For operating this step, the presented multiplication-factors (table 10.5) are needed.

parameter	description	example value
factor	the multiplicator between observed exceeding factor and the increased `X-dropability`-value	0.3

Table 10.5: Stage 3: Call profile monitoring parameter

Results

This step cannot be executed using the supported off-line traces as their duration is too short.The traffic of at least one week must be observed for representative results.

10.5 Pipelines

Sorting arriving requests into different pipelines helps prioritizing requests and supports on balancing the priority of each of these requests.

On SBC-A, this splitting differentiates between 1) Trusted and untrusted sockets and 2) register-requests and non register-requests:

10.5.1 Stage 4: Trusted / Untrusted Relationship

This stage sorts requests incoming from SBC-A stage 3 into two message pipelines, one high priority queue for requests from trusted nodes and one low-priority queue from for requests from untrusted or still unknown nodes. A "trusted node" is defined as an already successfully registered socket. This distinction is a precondition for request handling in subsequent analyzing steps. In addition, processing priorities are implementable in the SBC-A. Requests from untrusted nodes are only accepted, if there are open processing capabilities available.

A precondition for integrating a SBC-A into an existing infrastructure is the ease of configuration and the autonomy of this system. As a consequence, there is a need to avoid additional lookups from SBC-A to the user- or registration-database.

If a register-request is replied with a successful SIP 200 OK response, the system stores the request-sending socket for a predefined duration in a whitelist. This duration should be identical with the register-expiration time. All subsequent requests from this socket (within the predefined time interval) are handled as requests from a *Trusted Relationship* and will then be processed in the **trusted pipeline**, all other requests will be put in the lower priority *Untrusted Relationship*-pipeline (**untrusted pipeline**).

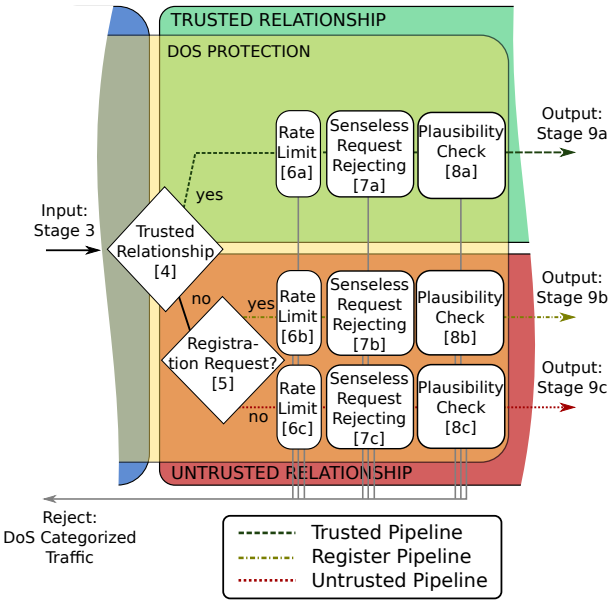


Figure 10.6: DoS-protecting section of the SBC-A. The second section differentiates between the trusted, untrusted and registration pipeline

parameter	description	example value
duration	timespan, during which a socket is stored in the whitelist	3600 s

Table 10.6: Stage 4: Configuration settings

Configuration Parameters

Operating this functionality needs a definition for how long a sending socket is stored in the whitelist (table 10.6)

10.5.2 Stage 5: Registration Request Prioritization

The REGISTER requests play a special role when handling untrusted requests: These registration attempts from untrusted hosts might be initial registration requests from well-known customers and will probably finish in a successful registration and in an entry in the whitelist presented in 10.5.1. Consequently, a second processing pipeline reserved for Register requests (**register pipeline**) is defined as depicted in figure 10.6.

The risk of Register flooding is tremendous, so the priority of the *Trusted*

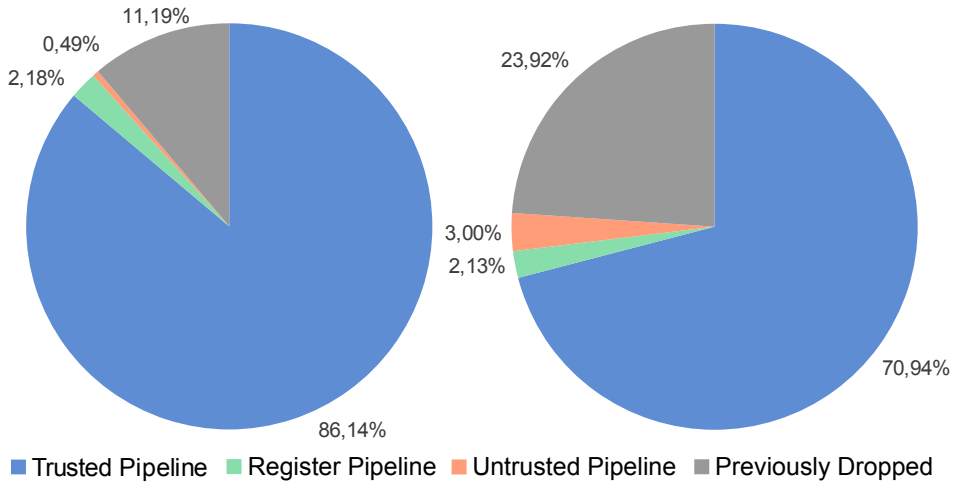
Relationships-pipeline must be privileged compared to the registration pipeline. Additionally, the registration pipeline as potential entry gate for well-known customers must be prioritized compared to the pipeline with untrusted, non-register requests.

There are no configuration parameters needed for this functionality.

Results of Stage 4 and Stage 5

Figure 10.7(a) (*A1 TA*) and 10.7(b) (*iptel.org*) depict the results of the pipelining processes. Both observed systems show a high ratio of requests from already successful registered hosts (*trusted pipeline*).

iptel.org's traces contain a high amount of non-REGISTER requests arriving from untrusted hosts. The high amount of OPTIONS-requests indicates, that the *iptel.org* infrastructure might be used as reference or test system with the OPTIONS-requests as application layer-"pings".



(a) *A1 TA* (b) *iptel.org*

Figure 10.7: Stage 4 and 5: the predominant part of all requests is assigned to the *trusted pipeline*

10.6 DoS Protection

The second part of the DoS-detection and -protection infrastructure receives an already reduced amount of requests (after flood-protection and retransmission dropping) categorized by trusted-, REGISTER- and untrusted-requests. With this categorization, analyzing tools in the next steps are able to apply several tests to each of these three pipelines.

	***A1 TA* - Traffic**	***iptel.org* - Traffic**
Traffic before Stage 4 and 5	134.33 rps	81.27 rps
Traffic to Trusted Pipeline	130.30 rps	75.79 rps
Traffic to Register Pipeline	3.29 rps	2.27 rps
Traffic to Untrusted Pipeline	0.74 rps	3.21 rps

Table 10.7: Stage 4 and 5: Sorting the requests into three pipelines

10.6.1 Stage 6: Rate Limitation

Rate limitation is (in contrast to flooding protection in subsection 10.4.1) not defined on a per-socket base. Instead, this stage defines and enforces the maximum allowed request rate per pipeline in a hierarchical manner. If this maximum allowed rate is exceeded, incoming requests are uniformly distributed dropped using a leaky bucket approach.

The maximum load capacity (in rps) of the deployed SBC-A must be evaluated (denoted as λ) first. Afterwards, the maximum allowed traffic for the trusted pipeline (λ_t), the register pipeline (λ_r) and the untrusted pipeline (λ_u) are defined. These λ-values define also the capacity of the leaky buckets in the rate limitation algorithm.

This pipeline-based load is defined hierarchical, the sum of λ_t, λ_r and λ_u should be equal to or larger than λ. As a practical example, λ for a specific system is set to 140 rps. The pipeline-dependent λ are set to λ_t=125, λ_r=6 and λ_u=8. If the current load in the trusted pipeline reaches 125, the leaky bucket algorithm of the trusted bucket starts rejecting requests. In parallel, the remaining loading capacity ratio (ρ) of the SBC-A

$$\rho = \lambda - \lambda_t \tag{10.1}$$

is split up between the register and the untrusted pipeline proportional to their λ-value:

$$\rho_r = \rho * \frac{\lambda_r}{\lambda_r + \lambda_u} \tag{10.2}$$

$$\rho_u = \rho * \frac{\lambda_u}{\lambda_r + \lambda_u} \tag{10.3}$$

In the specific example after λ_t reached its upper limit, the maximum load of the register pipeline (ρ_r) is 6.4 rps and ρ_u for untrusted requests is 8.6 rps. To avoid amplifying, the minimum allowed time interval for bucket size recalculation (t_{min}) should not be less than 1 sec.

parameter	description	example value
λ	Maximum loading capacity	140
λ_t	Maximum loading capacity for requests in the trusted pipeline	125
λ_r	Maximum loading capacity for requests in the registration pipeline	6
λ_u	Maximum loading capacity for requests in the untrusted pipeline	8
t_{min}	time interval of bucket size recalculation	1

Table 10.8: Stage 6: Rate limitation parameters

parameter	*A1 TA*-value	*iptel.org* value
λ_t	205	125
λ_r	6	6
λ_u	8	8

Table 10.9: Stage 6: Appropriate empirical λ-values for *A1 TA* and *iptel.org*

Configuration Parameters

The loading capacities of the three pipelines are defined separately to specify the different priorities (table 10.8).

Results

The traces of both *A1 TA* and *iptel.org* do not contain explicit high-load situations. Hence, the goal is to find appropriate λ-values for both infrastructures. The values chosen for λ_t, λ_r and λ_u (presented in Table 10.9) yielded acceptable results.

10.6.2 Stage 7: Dropping of Meaningless Requests

The SIP requests defined in RFC 3261 [Rosenberg02] initiate various actions in a SIP router with different processing costs. Depending on the pipeline and the type of request, the SBC-A might know in advance, which requests will never be positively processed. For these findings, the SBC-A does not even need to know the filter- and processing-rules applied in the core infrastructure. Hence it is sufficient to define a list of accepted requests per pipeline.

The advantage of a SBC-A compared to the NGN Overload Control Architecture (NOCA) concept (in section 7.4) is the absence of predefined routing

Untrusted Pipeline	*A1 TA* - Traffic	*iptel.org* - Traffic
Traffic before Stage 7	0.74 rps	3.21 rps
Traffic Reduction	96.64%	98.16%
Remaining Traffic after Stage 7	0.02 rps	0.06 rps

Table 10.10: Stage 7: Most requests arriving in the untrusted pipeline are handled as *meaningless*

parameter	description	example value
θ_t	White- or blacklist-flag for trusted pipeline	'b'
Θ_t	Request list for trusted pipeline	–
θ_r	White- or blacklist-flag for register pipeline	'w'
Θ_r	Request list for register pipeline	REGISTER
θ_u	White- or blacklist-flag for untrusted pipeline	'b'
Θ_u	Request list for untrusted pipeline	SUBSCRIBE, OPTIONS, NOTIFY

Table 10.11: Stage 7: The white-, resp. blacklists of the three pipelines

information. A list of (accepted or blocked) SIP requests per processing-pipeline is sufficient to block un-serviceable requests.

Consequently the trusted pipeline will accept most request-types, the registration pipeline will only accept register-requests and the untrusted pipeline will only accept a limited number of requests. Hence, a blacklist (for trusted pipelines) and two whitelists (for register- and untrusted pipeline) will be the best choice.

Configuration Parameters

For a live operating system, an *empty blacklist* for the trusted pipeline, a whitelist for accepting *register-requests* in the registration pipeline and either a *white-* or a *blacklist* for the untrusted pipeline is proposed. The configuration parameters for this step are three lists (Θ_t, Θ_r and Θ_u) and for each of them a white-/blacklist-flag (θ_t, θ_r and θ_u). An example is depicted in Table 10.11.

Results

The implementation of the *meaningless request dropping* is set up with the parameters presented in Table 10.11 and results confirm the expected behavior:

all requests in the trusted- and register-pipeline are accepted (resulting in a droprate of 0%) and a high number of requests are dropped in the untrusted pipeline.

As depicted in Table 10.10, 96.6% of all requests in the untrusted pipeline of *A1 TA* are rejected. Consequently only 3.36% of all requests in the untrusted pipeline are *not* SUBSCRIBE-, OPTIONS- or NOTIFY-requests. Analyzing the iptel-trace, the high amount of OPTIONS requests (figure 10.4) results in an even higher reject-rate of 98.2%.

As the ratio of requests in the untrusted pipeline is low compared to the total traffic, the effectivity of the request-based white- and blacklisting reduces the overall traffic only by 0.53% (*A1 TA*), resp. by 3.80% (iptel) .

10.6.3 Stage 8: Malformed SIP and Invalid Header Detection

Section 25 of RFC 3261 includes over 280 *BNF* rules for defining the SIP message and header syntax. To verify the validity of a SIP request, these rules are converted into regular expressions and applied to the SIP message. Seo et. al. discuss in [Seo08] this approach and the challenges of syntax-checking in detail. These regular expressions do only check the syntactical correctness, but not whether the header and header-values are making sense in this context or not.

As a consequence, the authors of [Seo08] propose to add a distinction between *mandatory*, *optional* and *forbidden headers* on a per-request base. For example, the paper discusses the ACK-request and lists up the three header categorizations (table 10.12). The SIP headers of this example are defined in SIP RFC 3261 and this example is not considering all other SIP RFCs. Obviously, the handling and configuring of such a *Non-allowed*-blacklist is impractical. Alternatively two whitelists for *mandatory*- and *optional*-headers (per request) are defined. All remaining (in this context *unknown headers*) should consequently increase the `X-dropability`-header.

Summing up, three processing steps are defined: First, the requests which fail the regular expression check will be rejected using a *403 Forbidden* response. Second, the parser checks whether the mandatory headers are included: if they are missing, the system rejects the requests with another *403 Forbidden*. Finally, each of the remaining, not-white-listed headers will increase the X-dropability header score by a small value (defined as χ in Table 10.13).

After these operations, *all syntactical correct requests* (no matter, if they are containing unknown headers or not) will then be forwarded (with the increased `X-dropability`-header) to the next SBC-A processing step.

Types	Header fields
Mandatory	Call-ID, CSeq, From, Max-Forwards, To, Via
Optional	Authorization, Contact, Content-Disposition, Content-Encoding, Content-Language, Content-Length, Content-Type, Date, MIME-Version, Record-Route, Route, Timestamp, User-Agent
Non-allowed	Accept, Accept-Encoding, Accept-Language, Alert-Info, Allow, Authentication-Info, Call-Info, Error-Info, Expires, In-Reply-To, Min-Expires, Organization, Priority, Proxy-Authenticate, Proxy-Authorization, Proxy-Require, Reply-To, Require, Retry-After, Server, Subject, Supported, Unsupported, Warning, WWW-Authenticate

Table 10.12: Stage 8: Example mandatory, optional and non-allowed header for ACK-requests as proposed by Seo et. al. in [Seo08]

parameter	description	example value
$H_\mu(INVITE)$	Mandatory header for INVITE-requests	Call-ID, From, To, ...
$H_o(INVITE)$	Optional header for INVITE-requests	Route, Allow, ...
χ	Increasing value of `X-dropability`	'0.1'

Table 10.13: Stage 8: depending of the request type, the list of allowed and optional headers differ.

Configuration Parameters

For setting up this analyzing step, a list of mandatory and a list of optional headers are predefined for each accepted request (table 10.13).

Practical realization and results

The practical realization focuses on the *syntactical correctness* of the analyzed offline traces.

For checking the correctness of the arriving SIP requests, two regular expressions are defined: the first (Listing 10.1) parses the SIP request line to validate it. The second regular expression (Listing 10.2) is applied to all other SIP headers.

With these two regular expressions, all analyzed requests are validated and the invalid requests are marked and rejected.

```
(
  ([a-z]*)\ +sip\:(([a-z\*0-9\.\-\_\+\%]+)@)*
  ([a-z0-9\.\-]+ )(\:[0-9]*)?
  (;[a-z\-\_]*(\=[a-z0-9\-\.\_]*) ?)*
  (\ )+sip\/2\.0
)
```

Listing 10.1: Regular Expression for parsing a request-line of a SIP message

```
(
  ^(
    ([\w+\-]*\:(\ )*)|(\w\:)
  ){1}
  (".*")*
  (
    (
      [a-z:0-9\%\.\-\+\|\!\_\(\)\*@\<\>\ \?\&
        \/\#=,\[\]\~]*
      (".*")*
    );?
  )*
)
```

Listing 10.2: Regular Expression for parsing all remaining headers of a SIP message

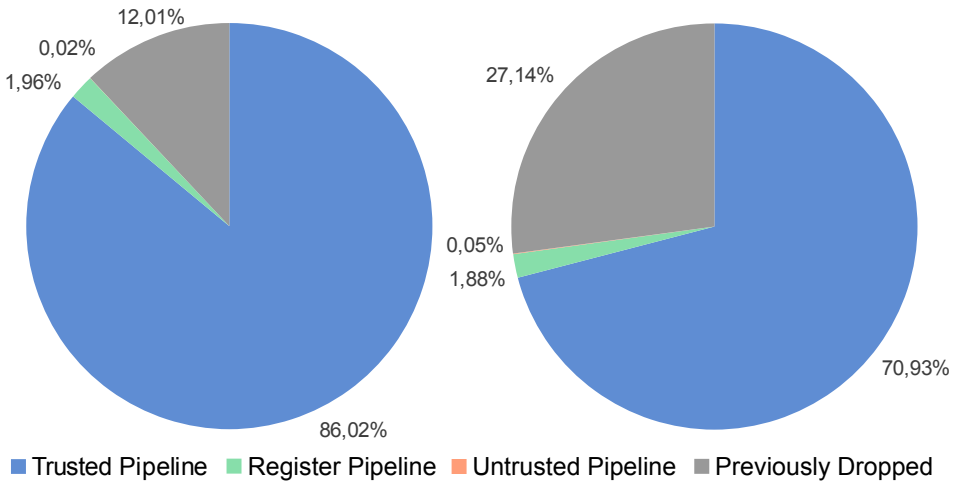
The results of stage 8 are presented in Table 10.14. Most of the syntactically faulty requests are of REGISTER-type. In particular the system detected and rejected mis-configured "Authentication"-headers.

	***A1 TA* - Traffic**	***iptel.org* - Traffic**
Reduced Traffic Trusted Pipeline	0.00%	0.01%
Traffic reduction Register Pipeline	2.22%	2.57%
Traffic reduction Untrusted Pipeline	0.00%	0.09%

Table 10.14: Stage 8: Most of the syntactical incorrect requests are REGISTER-requests for both *iptel.org* and *A1 TA*

10.6.4 Summary of Anti DoS mechanisms

The task of the Anti DoS-mechanisms is to reduce the traffic by rejecting and blocking requests. The blocked requests might be from flooding nodes (Step 1), might be retransmissions (Step 2) or the dropping of meaningless or invalid requests (step 7 and 8) to relieve the core. The forwarded requests are categorized and hence grouped into distinct priorities. The effectiveness of these



(a) *A1 TA* (b) *iptel.org*

Figure 10.8: After Stage 8: the stages 1-8 removed 12.01% (*A1 TA*), resp. 27.14% (*iptel.org*) of all requests

mechanisms (applied to the two discussed non-overloaded traces of *A1 TA* and *iptel.org*) illustrate a high potential in reducing traffic: 12.01% of all *A1 TA* traffic and 27.14% of the incoming *iptel.org* requests can be removed without any influence on functionality or user perception. The remaining requests are split into a trusted, a register and a untrusted pipeline as presented in Table 10.15 and figures 10.8(a) and 10.8(b).

As presented in the introductive section, only the Anti-DoS stages reject or drop requests. The following UC- and Overload-protecting stages forward the requests to the core SIP infrastructure, but mark the requests by their *value* using the `X-dropability`-header.

	***A1 TA* - Traffic**	***iptel.org* - Traffic**
Trusted Pipeline	130.11 rps	75.78 rps
Register Pipeline	2.96 rps	2.00 rps
Untrusted Pipeline	0.02 rps	0.06 rps
Dropped Requests	12.01%	27.14%

Table 10.15: After Stage 8: the majority of the requests forwarded to the UC protection are assigned to the trusted pipeline

10.7 Unsolicited Communication Protection

The novel approach of the SBC-A is to combine the results of overload protection with the results of UC-detection. As discussed in detail in section 4, it is generally not clear, whether a call setup attempt is resulting in UC or not. Hence the probably high ratio of "false positives" will therefore be intolerable for commercial VoIP-providers.

A second aspect to consider is the legal point of view. Similar to email SPAM, some providers might not block incoming calls even if they know, that these calls are unsolicited commercial phone calls. The final decision on blocking a call must be initiated by the customer. The presented approach

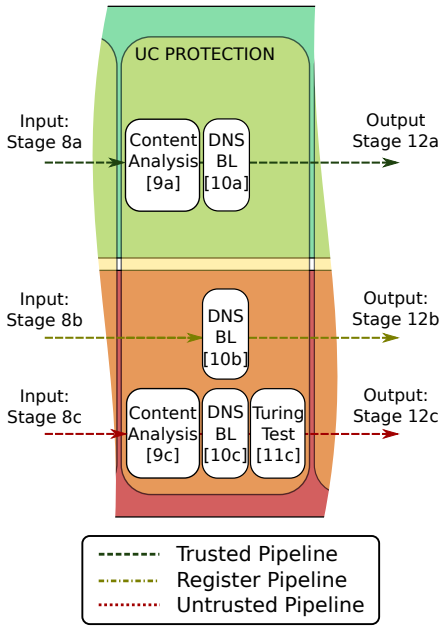


Figure 10.9: Unsolicited communication-protecting section of the SBC-A

takes this concept into account and handles UC-suspicious calls similar to the other threats: the SBC-A increases the `X-dropability`-score according to the probability the specific request might result in SPAM.

With this concept, all risk-factors are **combined into one single rating**, which will be analyzed by the core infrastructure (in case of congestion) or finally at the customer's user agent to block unwanted requests.

10.7.1 Stage 9: Content Analysis

Section 4.2 presents the three types of UC: the "voice based" *Call-SPAM* (SPIT), the text-based *Instant Messaging-SPAM* (SPIM) and *Presence-SPAM*

(SPPP).

"Voice-Based"-SIP SPAM is hard to detect in advance, as the SPAM-message is sent as voice- or video-stream *after* setting up the call. The text-based types are easier to detect as these messages are transmitted in specific SIP-headers or as SIP payload. The relevant headers are particularly the headers, which are displayed on the display of the SIP client.

The SBC-A as a SIP proxy component implicitly focuses on text-based SPAM, as this kind of UC is transported within the SIP headers and -payloads. The messages embedded in INVITE-, MESSAGE- or presence-requests can be understood as the equivalent to Email. When passing a request through the SBC-A, the content analyzing module parses the request and compares all bare-words strings with a blacklist of forbidden words.

A reference implementation applying the techniques of Email-SPAM prevention on top of text-based SIP messages is presented in [Hirschbichler09].

Figure 10.10: From- and To-header SPIT displayed in hardphone

Examples

Listing 10.3 presents an INVITE-request with UC-containing manipulated SIP headers. The resulting screen-presentation is depicted in figure 10.10, a screen-shot of a Grandstream SIP hardphone.

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Buy_viagra_at_viagra.com <sip:bob@biloxi.com>
From: Buy_viagra_at_viagra.com <sip:alice@atlanta.com>;
   tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
(sdp payload removed)
```

Listing 10.3: textual SPIT containing INVITE request advertising medicine

The second example (listing 10.4) presents a MESSAGE request, where the SPIT containing part is embedded in the request-payload (defined in RFC 3428 [Campbell02]).

```
MESSAGE sip:user2@domain.com SIP/2.0
From: sip:user1@domain.com;tag=49583
To: sip:user2@domain.com
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 18
Buy Viagra at Viagra.com
```

Listing 10.4: A MESSAGE request with an added SPIT-text in the text/plain payload

The content analysis is able to detect such messages and increases the `X-dropability` header accordingly

Configuration Parameters

Two parameter are needed for the content analysis: the list of blocked words and the `X-dropability`-increasing parameter (table 10.16).

Results

For the SBC-A reference tests, Λ_ϕ is filled with a list of 85 SPAM-words retrieved from the website `http://codex.wordpress.org/SPAM_Words` (downloaded 6.11.2012). This list is originally designed to block SPAM-messages in comment-areas of websites, but can also be used for the SBC-A approach.

parameter	description	example value
Λ_ϕ	List of suspicious words	Viagra, Cialis, C1alis , ...
χ	Increasing value of `X-dropability` per found suspicious word	'1'

Table 10.16: Stage 9: the list of SPAM-phrases

	***A1 TA* - Traffic**	***iptel.org* - Traffic**
SPAM Trusted Pipeline	0.00%	0.00%
SPAM Register Pipeline	0.00%	0.00%
SPAM Untrusted Pipeline	0.00%	0.00%

Table 10.17: Stage 9: the (empty) results of the SPAM-analyzing stage

To avoid false-positives when parsing SIP tag- and branch-fields, only SPAM-words with a length of five or more characters are accepted. Otherwise, the random characters in these fields may result in false positives.

The results on the live system shown in Table 10.17 illustrate one more time that there is currently no real SPAM-threat in VoIP communications.

10.7.2 Stage 10: DNS Blacklist

The next stage of the SBC-A is the application of a DNSBL server. Ramachandran et al. demonstrated in [Ramachandran06] that a high amount of Email-SPAM is arriving from the same, already blacklisted hosts: 80% of all incoming SPAM is created by already well-known and blacklisted computers. Consequently, if an Email SPAM system checks against a DNSBL and blocks all messages from these hosts, 80% of all Email SPAM can be successfully removed (or detected) by considering these DNSBL-blacklists.

Chapter 9 introduces a novel anti-SPIT blacklist-mechanism using a DNS-blacklist (DNSBL) server dedicated for SPAM-protecting Email-server. This stage tests the presented implementation and checks the relevant IP addresses in the SIP request against a DNSBL. Using this DNSBL-result, the SBC-A increments the `X-dropability` header accordingly.

Configuration Parameters

The blacklist-module of the SBC-A needs four configuration parameters, the DNSBL-server, the DNSBL-lookup timeout, the buffering duration and the `X-dropability` header incrementation (Table 10.18)

parameter	description	example value
Λ_δ	List of DNS-server	ix.dnsbl.manitu.net
θ	DNS query timeout in ms	100 ms
χ	`X-dropability` incrementation on a DNSBL-hit	'8'
β	The caching duration of the DNS-response	'14400'

Table 10.18: Stage 10: Configuration settings for DNSBL-lookups

	***A1 TA* - Traffic**	***iptel.org* - Traffic**
DNSBL Entries Trusted Pipeline	135 (0.013%)	426 (0.072%)
DNSBL Entries Register Pipeline	4 (0.017%)	103 (0.659%)
DNSBL Entries Untrusted Pipeline	0 (0.000%)	1 (0.223%)

Table 10.19: Stage 10: Result of DNSBL-lookups for the off-line-traces from *iptel.org* and *A1 TA* (recorded November 2012)

Results

The DNSBL-server list (Λ_δ) is set to `ix.dnsbl.manitu.net`, an Email SPAM DNSBL-Server operated by the german publisher Heise. Each source IP-address is checked against this DNSBL-server and the result value is stored for a predefined duration β (14400 seconds) in a local cache. During this interval, no further lookups are done to avoid high additional signaling latency. The value χ for the `X-dropability` increment is set to "8" for all three pipelines.

A small but significant number of requests from blocked hosts have been reported, both for *iptel.org* and *A1 TA* log files (Table 10.19).

10.7.3 Stage 11: Turing Test

The Turing test is the last proposed step in the unsolicited communication part. Section 8.2 presents the ideas and concept of this test, which is used to differentiate between human and machines. The probability of SPIT-content is likely higher when a call is detected as being generated by a calling generator compared to a call generated by a human.

Drawback and Usability

The *intrusive* approach of the Turing test in VoIP systems is problematic. Callers might be surprised by the fact an automate is answering (or manipu-

parameter	description	example value
τ_{min}	lower random number limit	0
τ_{max}	upper random number limit	99
χ	Increasing value of `X-dropability` when failing Turing test	'5'

Table 10.20: Stage 11: The Turing-test configuration

lating) the call and asks for caller interaction in some way.

The less intrusive Turing tests like *greylisting* are unusable in the SBC-A design, as this architecture is designed *not* to maintain explicit call states. Instead an intrusive Turing test is introduced for the implementation of a SBC-A: This applied test is an automated audio *Completely Automated Public Turing test to tell Computers and Humans Apart (captcha)* (like proposed by Quittek et.al. in [Quittek08a]). This approach creates a response with a voice-challenge like "Dial 5-2 to continue" and waits for the correct digits. If the user selected the correct number, the user passed the Turing test and its socket is stored in a whitelist to avoid this test in further calls.

Due to the needed interaction and the subsequent user annoyance, this test is activated only in the untrusted stream. If a user (or a robot) fails the test, the `X-dropability`-header will be increased by a high score.

Configuration Parameters

For the Turing test, the only configuration is the range of the random number and the `X-dropability` increment when failing the test (table 10.20)

Results

As this technique is interactive, the Turing test cannot be applied to the off-line analyzes of the *iptel.org* and *A1 TA* traces. Instead, a fixed quotient of 20% of failed Turing tests is assumed and applied to all INVITE-requests in the *untrusted* pipeline. The high quotient is chosen, as this novel technique will probably surprise callers and will result in failed tests. This approach results in the number presented in Table 10.21.

10.7.4 Summary of SPIT-marking mechanisms

This SBC-A draft proposes three analyzing steps to mark requests by their SPIT probability. The common concept of the SBC-A recommends avoiding statefulness and suggests no user interaction at all.

	***A1 TA* - Traffic**	***iptel.org* - Traffic**
Failed Test Trusted Pipeline	–	–
Failed Test Register Pipeline	–	–
Failed Test Untrusted Pipeline	13 (20%)	45 (20%)

Table 10.21: Stage 11: Results of the emulated Turing Test. Only requests in the untrusted pipeline have been challenged

There is only one exception to these requirements: in the rare case of incoming INVITE requests in the untrusted pipeline, the interactive Turing test is unpleasant for the caller, but a reliable method to block auto generated calls. Due to the high risk of false positives and the low acceptance of failed calls in professional telecommunication infrastructures, the UC protecting part of the SBC-A does not block or reject any traffic. The SPAM protection does only increase the score of `X-dropability`-header.

Due to the modularity of the SBC-A approach, these three steps are expandable by other tests suggested in RFC 5039 [Rosenberg08b]. By adding further SPIT-defending jigsaw-pieces, the reliability of the SPIT-marking infrastructure will increase.

Currently, practical analyzes demonstrate the limited amount of SPIT-relevant requests. The content analyzes did not result in a hit, but the DNS-blacklist results surprisingly in 0.09% of Email-SPAM blacklisted IP addresses in *iptel.org* and 0.013% in the *A1 TA* traffic. The sending hosts listed on the DNSBLs are probably botnet infected hosts.

10.8 Overload Protection

After two steps of DoS-detection and -protection as well as the UC-protection, the final SBC-A steps focus on request-rating for overload-handling.

In this section, the SBC-A differentiates between prioritized customers, inspects the type of request, adds randomness for scattering the `X-dropability`-scores and finally, checks whether the requests are part of a dialog or not. These stages complete the proposed SBC-A solution.

10.8.1 Stage 12: Request Dependent Marking

Section 10.6.2 presents the white- and blacklisting concept for SIP requests. This approach is extended in this stage by marking the requests by their "value", where value stands for the commercial *or* the operational value.

The operational value addresses requests, which are vital for the call- or the signaling-flow. Highlights are the BYE or ACK requests, where the first finishes a call and releases the infrastructure for further call-processing, the

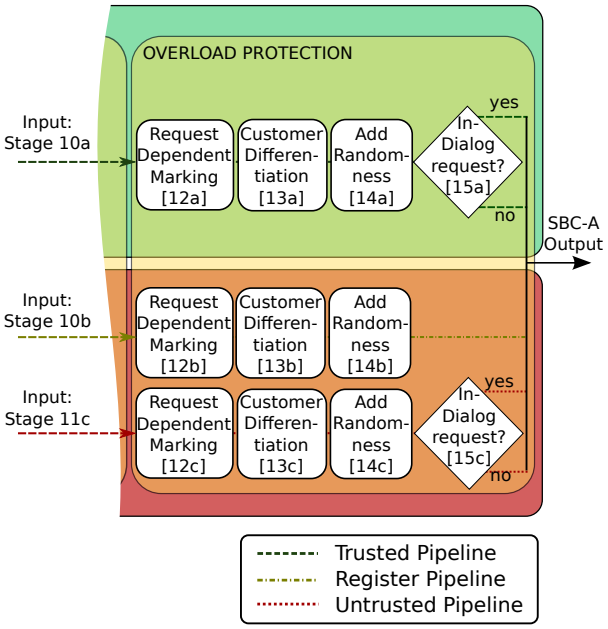


Figure 10.11: Overload Control-section of the SBC-A

latter finalizes a call setup. If the ACK-request are not preferred, the resource consuming call-setup – with all its signaling and processing – will be worthless at all, as the ACK-request is needed for a successful call-setup. A missing ACK-request also forces the callee to retransmit the "200 OK" response.

The commercial value is driven by profit: Which type of request supports the operator to earn revenue and which requests are "worthless"? The revenue-generating requests are usually INVITE- and MESSAGE-requests. Both are customer driven and generate profit for the operator. Additionally, dropping of these requests (in particular the INVITE-request) is observed by the customer and reduces the customer's perceived satisfaction.

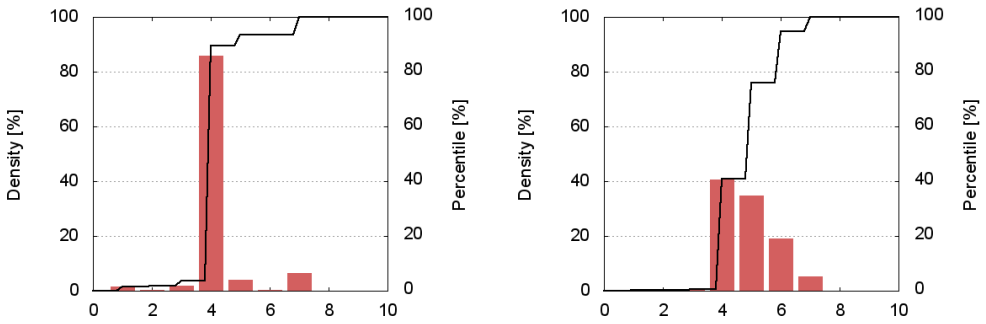
Finally, the REGISTER-request needs closer discussion: this request is the precondition for receiving or transmitting SIP requests. Hence blocking REGISTER requests results in an incapability to create revenue-generating requests and is notified by the customer reducing the perceived QoS.

The next step for creating a list of requests and their weight is the differentiation between the *trusted* and the *untrusted* pipeline: requests in the trusted pipeline are always related to the operator's customers and should be preferred. Requests inside the untrusted pipeline might be unwanted traffic, SPIT, unsuccessful call setup attempts, etc. These requests should be marked with a higher `X-dropability`-value.

Request in *Trusted* Pipeline		Request in *Untrusted* Pipeline	
Request	**Rank**	**Request**	**Rank**
ACK	1	REGISTER	8
BYE	2	ACK	5
INVITE	3	BYE	5
CANCEL	3	CANCEL	5
REGISTER	4	INVITE	7
MESSAGE	5	MESSAGE	11
UPDATE	6	UPDATE	12
NOTIFY	6	NOTIFY	12
SUBSCRIBE	7	SUBSCRIBE	12
other	5	other	14

Table 10.22: Stage 12: The `X-dropability`-header is increased by the value of the column "Rank". The proposed values are used for the reference implementation of the SBC-A

The definition of the `X-dropability`-header states, that the higher the score, the lower the priority and the value of this requests is. Hence, requests with a high signaling or economical value need to get a **low** additional `X-dropability`-score and requests with a low economical or technical value a high additional `X-dropability`-score.



(a) *A1 TA* (b) *iptel.org*

Figure 10.12: Stage 12: The density of the `X-Dropability`-header after the request-dependent marking.

Configuration Parameters

For activating this functionality, a table of requests and their specific values is needed. A specific example of proposed values is presented in Table 10.22.

Results

iptel.org request distribution is split into REGISTER-, OPTIONS- and NOTIFY-requests. The various dropability-scores of these requests (as defined in Table 10.22) result in a wide distribution of `X-dropability`-scores as illustrated in histogram 10.12(b).

In contrast, most of *A1 TA*'s traffic consists of 84.93% REGISTER-requests. Therefore, most of the request-dependent `X-dropability` scores are "4" (histogram 10.12(a)), as this is the rank of the register-requests as defined in table 10.22.

10.8.2 Stage 13: Customer Differentiation

In addition to the request-type prioritization or the prioritization on the trusted/untrusted pipeline, a prioritization based on the customer's type of contract is proposed. Customers of most common telecommunication providers can choose – depending on their needs – between different types of contracts or service ranges. Depending on this decision (and the monthly fee), customers with a gold-class contract will be offered higher bandwidth, guaranteed QoS, etc.

To map this differentiation to the proposed SBC-A infrastructure, a small extra `X-dropability` score is added to requests of bronze customers to prioritize the customers who are willing to pay more.

Drawbacks and Challenges

The SBC-A is designed as a solution with limited configuration- and communication-overhead. Therefore resource consuming lookups of gold- or bronze class customers on each incoming request should be avoided.

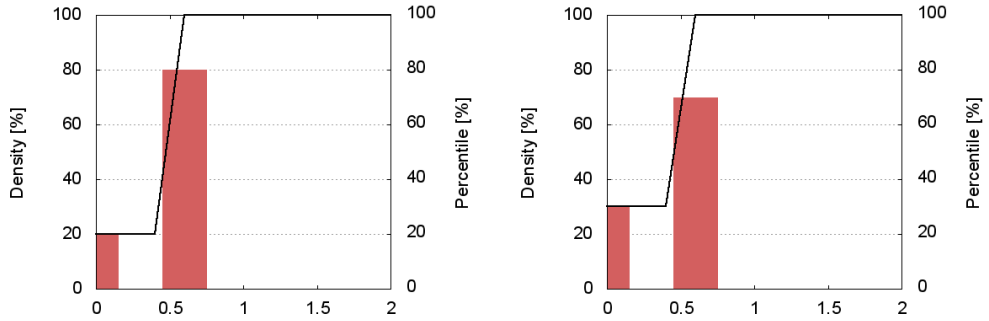
Hence, this design proposes to keep an internal list of gold-class customers. This list is filled by reading an additional SIP-parameter which is added by the core infrastructure during registration process and committed to the SBC-A using the "To"-header of the 200 OK-response to the REGISTER-request. Such a 200 OK response is presented in Listing 10.5.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP sbca.biloxi.com;branch=z9hG4bK
Via: SIP/2.0/UDP bpc.biloxi.com:5060;branch=z9hG4
To: sip:bob@biloxi.com;customertype=gold;tag=459
From: sip:bob@biloxi.com;tag=2493k59kd
Call-ID: 8438176376842300998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:bob@131.130.88.111:5060>
Expires: 7200
```

```
Content-Length: 0
```

Listing 10.5: A 200 OK response from the VoIP core infrastructure containing the "customertype"-header extension

When the SBC-A receives this header, it parses the To-header, extracts the `customertype`-field and stores this value together with the socket from the Contact-header and the value of the Expiration-header in a local database. Each subsequent request can then be compared with the entry in this local database and the request can then be rated accordingly.



(a) *A1 TA* (b) *iptel.org*

Figure 10.13: Stage 13: Hypothetical 79% (*A1 TA*), resp. 71% (*iptel.org*) of all received requests are caused by bronze-class customers. The `X-dropability`-score of these requests is increased by 0.5

parameter	description	example value
π	To-header parameter	`customertype`
$\Delta_{parameter}$	List of allowed parameters and their `X-dropability`-increase	((gold, 0), (bronze, 0.5), ...)

Table 10.23: Stage 13: Configuration of customer differentiation

Configuration Parameters

This functionality needs two parameters. First, the name of the header field and second a list of possible customer-types and their corresponding `X-dropability` incrementation (table 10.23). When defining the `X-dropability`-increasing value, the administrator has to keep in mind a **weighted balance** between the different stages: the `X-dropability`-increase of an bronze-user

should be considerably smaller than e.g., the increment when failing a Turing-test.

Results

The analyzed off-line traces do not contain any information whether a user is a gold-class- or an bronze-class-user. For this reason, the distribution between the different customer-types cannot be determined in lack of external sources like provisioning databases.

For the reference off-line trace analyses, a gold quotient (the quotient, whether a customer is a gold-class-user or not) of 30% of all registered users is supposed. In practice, the 200 OK-responses to register requests are analyzed and the registered socket is stored with a probability of 0.3 as gold-class-customer. All requests from this sockets are then interpreted as requests from a gold-class customer.

This approach results in the `X-dropability` distribution presented in figure 10.13(a) (*A1 TA*) and in figure 10.13(b) (*iptel.org*).

10.8.3 Stage 14: Add Randomness

Section 7.3.3 presents the handling of `X-dropability`-marked SIP requests inside the SIP core infrastructure when activating overload control mechanisms.

Due to 15 processing stages, the probability of identical `X-dropability` score is high. In contrast, the algorithm presented in 7.3.3 is able to handle fine grained floating point values. Consequently, a scattering of the SBC-A resulting `X-dropability`-header is requested. A proposal to solve this task is

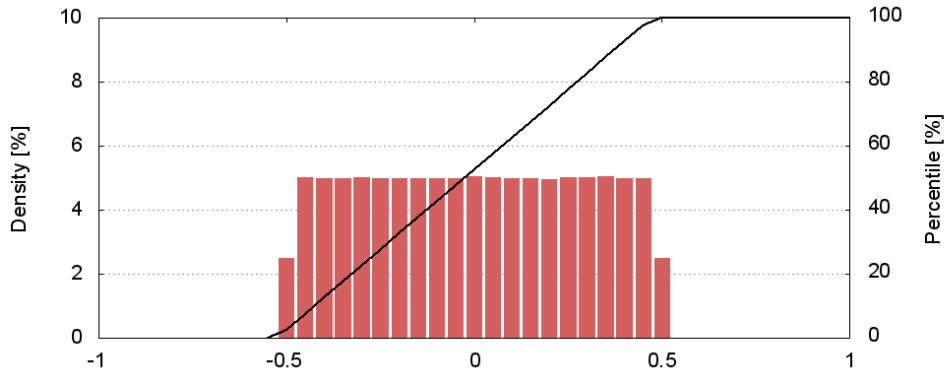


Figure 10.14: Stage 14, *A1 TA* (and also *iptel.org*): The histogram over the distribution of the random values. These values (between -0.5 and 0.5) are added to the current `X-dropability`-value

parameter	description	example value
ξ_u	Upper limit of random value	0.5
ξ_l	Lower limit of random value	-0.5

Table 10.24: Stage 14: The lower and the upper bound of the randomization value

to add a random score ρ (e.g., between -0.5 and 0.5) to the `X-dropability`-header (X_{old}) to get a new `X-dropability`-header (X_{new}). The maximum score is denoted as ξ_u, the minimum as ξ_l:

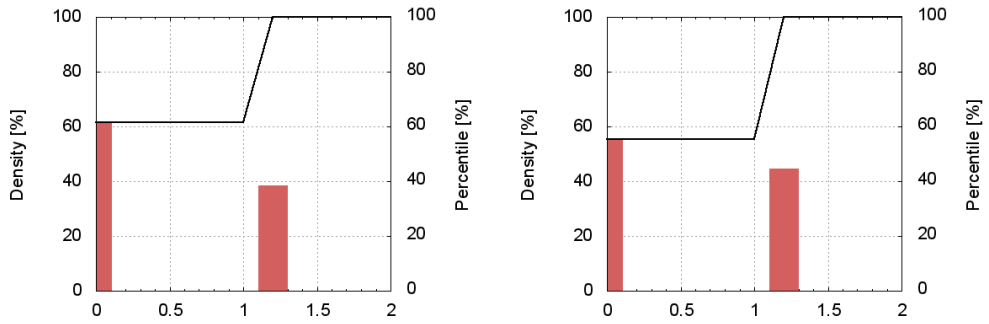
$$X_{new} = X_{old} + \rho(\xi_l, \xi_u) \tag{10.4}$$

Configuration Parameters

This SBC-A needs only two parameters as presented in Table 10.24, the upper and the lower limit of the random score. Setting both values to "0", deactivates the randomization.

10.8.4 Stage 15: In Dialog Request

RFC 3261 [Rosenberg02], section 6 defines a dialog as a timely limited peer to peer relation between two UAs. A dialog is identified by a call-id, a local tag and a remote tag. This dialog is established by a SIP final response.



(a) *A1 TA* (b) *iptel.org*

Figure 10.15: Stage 15: In-dialog-requests are the majority (*A1 TA*: 61%, *iptel.org*: 55%) of all requests. The `X-dropability`-header value of all other requests is increased by *1.2 points*.

This stage prefers requests, which contain the call-id, the to- and from-tag of an already successfully established dialog. These requests are trustworthy, as the B-party (or the proxy) already accepted a request from this contact.

This last stage is the only one which keeps a state and has to store information of previous transactions. In practice, the stage stores the call-id, two tags and a time-stamp within a database. If a request arrives, the SBC-A compares these parameters with the database to retrieve the information, whether this request is arriving within an established dialog. After a predefined time the entry is removed from the whitelist. If a request's identifier is not found within the database, the `X-dropability`-header will be increased slightly.

This functionality is added, as it offers a reliable information on the credibility of a request. As many requests are already blocked before this stage, this method is placed right at the end of all stages to analyze only a reduced number of requests. This ordering should keep the load low.

parameter	description	example value
ι	time-to-live of database entry (in s)	120
χ	Increasing value of `X-dropability` when incoming request is out of dialog	1.2

Table 10.25: Stage 15: Configuration options for In-Dialog checks

Configuration Parameters

This final analyzing stage defines two parameters: the first is the duration an entry is kept in the database (denoted as ι). The second defines the `X-dropability` (denoted as χ) increment (Table 10.25).

Results

Although the two analyzed traces differ in the traffic- and request-profile, the ratio of in-dialog-requests is similar. The majority of requests is part of an established dialog (figure 10.15(a) and figure 10.15(b)). A closer inspection of the requests in both traces illustrate, that the majority of all in-dialog-requests are reregistration requests.

Preferring the reregistrations using the in-dialog-check makes sense, as these requests are part of already authorized and authenticated dialogs.

10.9 SBC-A Results

The extensive off-line traces provided by *A1 TA* and *iptel.org* illustrate the high potential of reducing and marking traffic using the SBC-A.

In *A1 TA*, the traffic was reduced by 12.01% as shown in figure 10.16 and all remaining requests are sorted into three pipelines and marked with the `X-dropability`-header. Figure 10.17 illustrates a wide distribution of

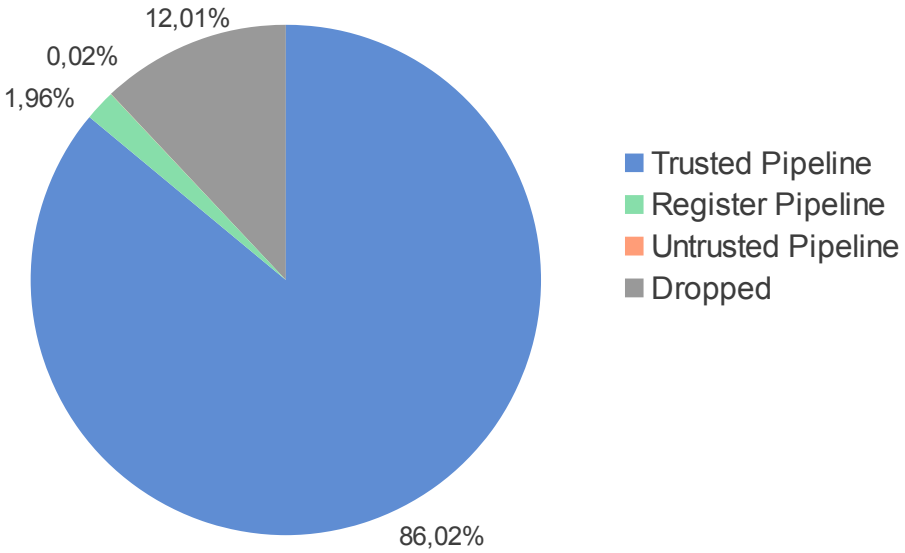


Figure 10.16: *A1 TA*: The results after the dropping-, rejecting- and categorizing activities: 12% of the initially received requests are removed and the remaining requests are sorted into the three pipelines and forwarded to the dropability rating-steps

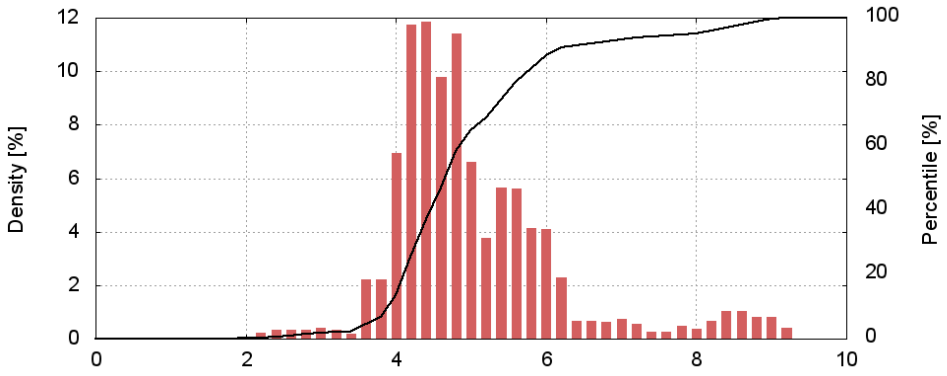


Figure 10.17: *A1 TA*: the remaining requests are rated with the presented techniques according to their value. The result is a broad number of `X-dropability`-scores, which can be used by overload controlling nodes as input for potential request-dropping

the X-dropability-scores for precise request dropping on the OC-protecting infrastructure.

Similar, *iptel.org*'s traffic was reduced by 27.17% as depicted in figure 10.18. Forwarded requests are assigned to the three pipelines and are marked with the X-dropability-header. The distribution of these header values is presented in figure 10.19.

10.10 Further processing steps

With the introduction of the three pipelines, the modular architecture and the central request rating parameter "X-dropability", further research can extend this concept with additional analyzing and processing stages.

When extending the SBC-A, it is important to maintain the main SBC-A principles of

- low interaction with other components
- per request analyzes
- stateless operations.

The result of new introduced processing stages must either be mappable to the X-dropability header, or must result in negative final responses (like "403" or "503"), respectively.

10.11 Core and UAS processing

After passing through the three analyzing and marking sections of the SBC-A, the initially received INVITE request appears is extended by the X-dropability header as presented in Listing 10.6:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP sbc-a.biloxi.com;branch=z9hG4bKd84ks2
Via: SIP/2.0/UDP pc3.atlanta.com;branch=z9hG4bKnashds8
X-dropability:  14.331
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
```

Listing 10.6: INVITE-request after SBC-A analysis

This request is then forwarded to the next SIP-downstream hop of the core infrastructure for further SIP processing. In case of congestion, the X-dropability header is referenced as a decision guidance for ordered request dropping. When requests with a set X-dropability header reach the UAS, the X-dropability header can be useful for final processing at the user agent:

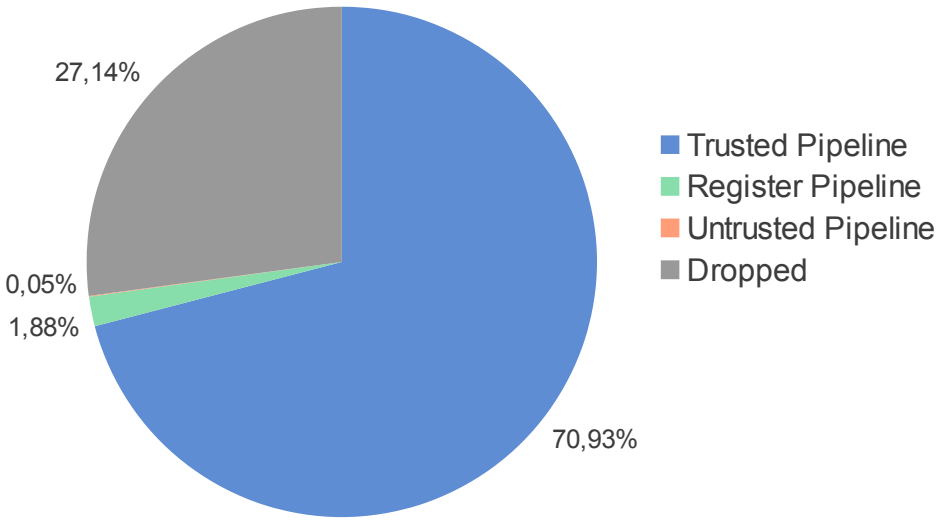


Figure 10.18: *iptel.org*: The presented dropping and rejecting techniques reduce the amount of incoming requests by 27%. The remaining requests are rated and forwarded to the next SBC-A analyzing steps

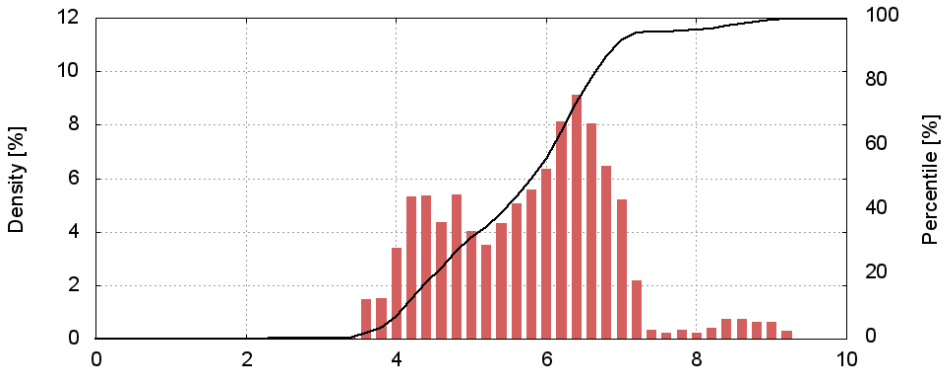


Figure 10.19: *iptel.org*: The analyzing steps of the SBC-A create a widely distributed number of `X-dropability` scores for further processing and eventual dropping/rejecting in overload state.

Giving the UA the possibility to inspect the `X-dropability` header helps the UA to decide whether to accept a request or not. When implementing a configuration-setting in the UA to deny calls with a high `X-dropability`-score, the callee can decide, whether he wants to face the risk of false positives or not.

10.12 Overload Control in the SBC-A

As the SBC-A itself is also part of the OC-infrastructure, it must be able to handle overload control messages too.

Two situations are possible: acting as protecting device for overloaded SIP core components or protecting itself when facing a local overload situation.

10.12.1 Handling Overload Control messages

When the SBC-A receives responses from the VoIP core infrastructure containing active overload control, the loss- or rate-based algorithms must be located **after** the SBC-A-analyzing steps. Using the analyzing results, the SIP processing core infrastructure is capable to drop the invaluable requests first as presented in section 7.3.3.

10.12.2 Handling own overload state

When the SBC-A itself is in an overloaded state, a multi-step algorithm can be applied:

1. reject all requests from the untrusted pipeline, then
2. reject all requests from the register pipeline, then
3. start rejecting all requests from the trusted pipeline until a normal state
4. if the system is congesting, the system will not be able to walk through all the SBC-A-steps. The final way is to introduce a step "0", where random requests (using the mechanism from RFC 3261 [Rosenberg02]) are rejected until a "normal" state is reached and step 1-3 can be applied.

10.13 Evaluation

A REGISTER flooding attack with 100 rps is simulated to evaluate the effectiveness of the proposed technique. The simulated traffic is merged with the already recorded *A1 TA* trace and the resulting trace is then read by the SBC-A reference implementation.

This section presents the results of the *essential* protection stages and shows the percentage of successfully blocked malicious traffic.

parameter	description	value
rand_rate	request rate of the attack	100 rps
rand_fixed_ip	source IP – in this test, a single source DoS is assumed	192.168.1.12
rand_fixed_port	source port – this test varies the sending ports, hence this value is empty	""
rand_fixed_request	request type	REGISTER

Table 10.26: The simulated attack scenario "against" the *A1 TA* trace

REGISTER requests	*A1 TA* - Traffic	
Trusted Pipeline	143.34 rps	96.9%
Register Pipeline	4.59 rps	3.1%

Table 10.27: Most (conventional) Register requests are arriving from already registered (*trusted*) hosts and are handled in the trusted pipeline.

10.13.1 Attack scenario

The REGISTER flooding attack with varying source ports is one of the most effective attacks against a SIP driven infrastructure. The configuration and characteristics of such attacks has been discussed earlier in section 5.3.

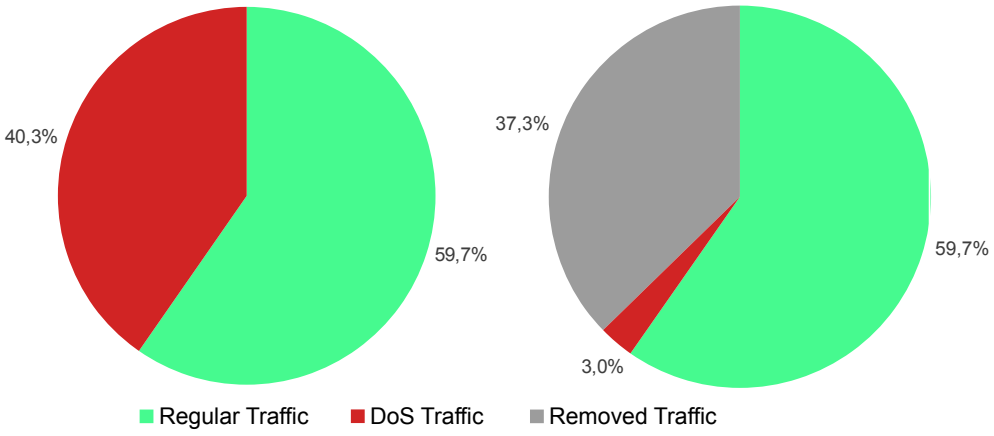
These REGISTER-flooding attacks are easy to implement and executable with on single COTS PC. The applied attack to the *A1 TA*-trace is set up with the parameters presented in Table 10.26. The duration of this attack is 5124 s. During this time, 512.400 REGISTER flooding messages and 758.000 conventional requests are generated. Hence, 40 % of all 1.270.400 requests are generated by this DoS-attack. As already discussed, a "varying source port" flooding attack is hard to detect and cannot be blocked by 100% without a massive risk of false positives.

The goal is to considerably reduce the DoS-traffic without loosing regular traffic. The question is: how large is the ratio of DoS-requests forwarded to the next hop compared to the incoming ratio and which steps are protecting the system effectively?

10.13.2 Results

In the original traffic, 97% of the inspected REGISTER-requests are categorized into the trusted pipeline as they are already in a known relationship with the protected infrastructure (table 10.27). In contrast, the REGISTER requests from the flooding-traffic are not authenticated and hence categorized into the (untrusted) REGISTER pipeline.

During the simulated REGISTER-flooding attack over 40% of all incoming requests are part of the DoS-attack (as shown in figure 10.20(a)). After processing the SBC-A steps (in particular the pipeline dependent request reduction), the amount of DoS-requests is reduced to 3% of all messages (figure 10.20(b)) as the REGISTER pipeline rate limitation (Stage 6) truncates the traffic down to the maximum allowed request rate. This mechanism is (like discussed in section 10.6.1) not able to distinguish between the "good" and flooding-requests in this pipeline, but reduces the traffic uniformly distributed. The high portion of flooding-requests and the low portion of "good" initial REGISTER-requests inside the register pipeline result in the effective reduction of flooding requests and the little reduction of good requests.



(a) Before SBC-A processing (b) After SBC-A processing

Figure 10.20: The results of a REGISTER flooding attack after applying the SBC-A protection mechanism

Compared to the initial *A1 TA* trace data, the SBC-A outgoing traffic is **increased slightly by only 3%**. This result illustrates the positive effect of the presented SBC-A approach in the complex attack-form.

Summarizing, the number of malicious messages is successfully reduced and the protected systems can maintain their state. Traffic handled via the trusted pipeline is still forwarded to the next step and only initial REGISTER-requests are limited during the time of attack.

10.14 SBC-A Summary

In case of overload in a core element, there is no alternative to blocking and dropping requests to reduce the load to an acceptable amount. The preevalu-

ation of the SBC-A supports the decision process, which requests are dispensable and can be dropped, and which requests are needed to keep existing calls or registrations alive, or which are needed to fulfill the customers need of high QoS in the best possible way.

The presented approach is not completely aware of false-positive decisions, but compared to uniform dropping, the preevaluation keeps the number of lost (important) requests low. Requests marked with a high `X-dropability`-score are dropped earlier, high value requests (with a low dropability-score) are instead "protected" from being dropped.

The SBC-A architecture limits flooding DoS-attacks and marks requests for further OC protection as well as rates SIP requests by their (economic- and QoS-) quality. It relocates time- and CPU-consuming activities for overload control according to the SOC-draft [Gurbani12] off from the (high-loaded) core components to a single, dedicated perimeter-security component, the SBC-A.

Using traffic samples of *iptel.org* and the A1 Telekom Austria core, the positive effects of nonuniform request-marking and -dropping are presented in detail. Compared to uniform dropping, the requests are marked and prioritized in a detailed manner. The `X-dropability`-score is nearly equally distributed in both analyzed samples and can therefore be used in further overload control processing.

Summing up, the SBC-A is an important and powerful component at the perimeter of complex VoIP infrastructures to act as a protecting node. It combines overload control and SPIT protection in one node and offers all other components an indicator to rate importance of the various arriving SIP requests.

Chapter 11

Summary and Conclusion

The transition from circuit switched to packet switched telecommunication systems offer new thread scenarios against the carrier's infrastructure. The danger of *Denial of Service (DoS)* attacks, massive amounts of unsolicited voice and text messages (*Spam over Internet Telephony (SPIT)*) as well as possible accidental congestion increase with the deployment of SIP operated devices. This thesis presented the reasons for processing congestion of the core infrastructure in a theoretical way and illustrated the thread with specific practical examples. These examples documented the few ressources an attacker needs to block the availability of a carrier grade telecommunication infrastructure.
This thesis discussed the IETF and ETSI concepts to reduce SPIT and to effectively communicate congestion control information to upstream nodes to reduce arriving load. The compared concepts of IETF and ETSI showed to have strengths but also weaknesses and limitations. The IETF approach is limited to SIP and hop-by-hop communication and the ETSI concept suffers from its high complexity and therefor low acceptance. Next to load generated by new requests, the problem of load initiated by SIP retransmissions is discussed in an additional section. This thesis proposed an extension of the SIP retransmission timer-concept to either static or dynamical timer-reconfiguration to limit traffic which is generated by retransmissions introduced in high latency mobile 2G access networks.

As a consequence to the limited congestion protection and as main contribution, the thesis introduced an extended perimeter protection node named *Session Border Controller - Advanced (SBC-A)*. The concept of the SBC-A combined DoS-protection by rejecting unwanted traffic together with an algorithm to mark the "value" (denoted as the dimensionless "dropability" value) of each accepted and forwarded request. Since the detection of voice-SPIT

is not reliable without the risk of false-positives, all SPIT-suspected requests were not rejected but marked with an increased "dropability"-value. In case of congestion of one (or more) of the protected nodes, the calculated dropability value of the SIP requests can be reused to decide, which requests to drop and which requests are important to be forwarded.

The reliability of the SBC-A concept was tested by applying this concept to *A1 Telekom Austria (A1 TA)* and *iptel.org* VoIP traffic. For this test, recorded real-live traffic of both providers was analyzed by a reference implementation of the SBC-A. The results showed, that the amount of forwarded requests was reduced by 12% (A1 TA), respectively by 27% of the *iptel.org* traffic. All other forwarded requests were marked with significant dropability values. A simulated DoS attack similar to the practical examples indicated, that the SBC-A concept was able to remove 92% of the malicious traffic.

Future research in the field of telecommunication security should first and foremost focus on the the problem of DoS attacks against IMS networks to avoid upcoming massive outages in critical communication environments. The switch to PS in voice telecommunication at latest with the introduction of the PS-only LTE is facing a continuous risk of congestion and therefor a working congestion control communication is a must to reduce the impact of DoS-attack generated traffic.

Annex

List of Abbreviations

3GPP	3rd Generation Partnership Project
A1NUVS	A1 Network Unified Voice Service
A1TA	A1 Telekom Austria
AAA	authentication, authorization and accounting
AIB	Authenticated Identity Body
AS	Application Server
B2B	Back-to-Back
BHCA	Busy Hour Call Attempts
BNF	Backus-Naur Form
BTS	Base Transceiver Station
CA	Control Adaptor
CD	Control Distribution
CDP	CDProcess
CDR	CDRestriction
CF	Control Function
COTS	Commercial Off the Shelf
CPU	Central Processing Unit
CS	Circuit Switched
CS2SIP	Circuit Switched to SIP
DDoS	Distributed Denial-of-Service
DNS	Domain Name System
DNSBL	DNS-based Blackhole List
DoS	Denial-of-Service
E1	E-carrier level 1
ENUM	E.164 NUmber Mapping

ES	ETSI Standard
ETSI	European Telecommunications Standards Institute
EU	European Union
FTW	Forschungszentrum Telekommunikation Wien
G-MSC	Gateway-Mobile Switching Center
GOCAP	Generic Overload Control Application Protocol
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GSMA	GSM Association
HLR	Home Location Register
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IM	Instant Message
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPv4	Internet Protocol v4
IPv6	Internet Protocol v6
IPX	IP eXchange
ISDN	Integrated Services Digital Network
ITC	Institute of Telecommunications
IVR	Interactive Voice Response
KPI	Key Performance Indicator
KR	Known Relationship
LAN	Local Area Network
LTE	Long Term Evolution
M2M	Machine-to-Machine
MD5	Message-Digest Algorithm 5
MGW	Media Gateway
MHT	Mean Holding Time
MOS	Mean Opinion Score
MSC	Mobile Switching Center
MTA	Mail Transfer Agent
NAT	Network Address Translation
NGN	Next Generation Network
NNTP	Network News Transfer Protocol
NOCA	NGN Overload Control Architecture
nrpe	Nagios Remote Plugin Executor
NSS	Network Subsystem
OC	Overload Control

OS	Operating System
PARIS	Performance, Availability and Reliability Information System
PBX	Private Branch Exchange
PESQ	Perceptual Evaluation of Speech Quality
PS	Packet Switched
PSTN	Public Switched Telephone Networks
QoS	Quality-of-Service
RFC	Request for Comment
RM	Restrictor Manager
RMP	Restrictor Manager Process
RTD	Response Time Distribution
RTP	Realtime Transfer Protocol
RTT	Roundtrip Time
SBC	Session Border Controller
SBC-A	Session Border Controller - Advanced
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SEER	Session Establishment Effectiveness Ratio
SIP	Session Initiation Protocol
SIP2CS	SIP to Circuit Switched
SIPp	SIP Performance Test Tool
SLA	Service Level Agreement
SMTP	Simple Mail-Transfer Protocol
SNMP	Simple Network Management Protocol
SOC	SIP Overload Control - Working Group
SPIM	SPam over Instant Messaging
SPIT	SPam over IP Telephony
SPPP	SPam over Presence Protocol
SRD	Session Request Delay
SRD	Session Request Delay
SSL	Secure Socket Layer
STUN	Session Traversal Utilities for NAT
SUT	system-under-test
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
THIG	Topology Hiding Internetwork Gateway
TISPAN	Telecommunications and Internet converged Services and Protocols for Advanced Networking
TLS	Transport Layer Security

TURN	Traversal Using Relays around NAT
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UC	Unsolicited Communication
UDP	User Datagram Protocol
UI	User Interface
UMTS	Universal Mobile Telecommunications System
UR	Unknown Relationship
URI	Uniform Ressource Identifier
VoIP	Voice over IP
VoLTE	Voice over LTE
WGLC	Working Group Last Call
WWW	World Wide Web
XML	Extensible Markup Language

List of Figures

List of Tables

Listings

Bibliography

[3GPP07] 3GPP. Security aspects of early IP Multimedia Subsystem (IMS). TR 33.978, 3rd Generation Partnership Project (3GPP), June 2007.

[3GPP10a] 3GPP. 3G security; Access security for IP-based services. TS 33.203, 3rd Generation Partnership Project (3GPP), October 2010.

[3GPP10b] 3GPP. IP Multimedia Subsystem (IMS); Stage 2. TS 23.228, 3rd Generation Partnership Project (3GPP), September 2010.

[3GPP10c] 3GPP. Study of mechanisms for Protection against Unsolicited Communication for IMS (PUCI). TR 33.937, 3rd Generation Partnership Project (3GPP), June 2010.

[Abdelal10] Ahmed Abdelal and Wassim Matragi. Signal-based overload control for sip servers. Technical report, Piscataway, NJ, USA, 2010.

[Alendal10] M. Alendal. Operators need an ecosystem to support 50 billion connections. In *Ericsson Business Review*, pages 40–43, 2010.

[Balakrishnan01] H. Balakrishnan and S. Seshan. The Congestion Manager. RFC 3124, Internet Engineering Task Force, June 2001.

[Barber00] S. Barber. Common NNTP Extensions. RFC 2980, Internet Engineering Task Force, October 2000.

[cam04] *The 3G IP Multimedia Subsystem.* John Wiley and Sons, Ltd, 2004.

[Campbell02] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428, Internet Engineering Task Force, December 2002.

[Case02] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and Applicability Statements for Internet-Standard Management Framework. RFC 3410, Internet Engineering Task Force, December 2002.

[Darilion08] K. Darilion. Analysis of a VoIP Attack. Technical report, IPCom Gesellschaft fr internetbasierte Kommunikationsdienste mbH, 10 2008.

[ec202] Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications), , July 2002.

[Egger10] C. Egger, M. Hirschbichler, and P. Reichl. Enhancing SIP Performance by Dynamic Manipulation of Retransmission Timers. Technical report, 2010. 29th IEEE International Performance Computing and Communications Conference, December 9th to 11th.

[Egger12] C. Egger. Improvement Strategies for the Signaling Performance of the Session Initiation Protocol. *Dissertation*, , 05 2012.

[ETSI07a] ETSI. NGN Congestion and Overload Control; Part 3: Overload and Congestion Control for H.248 MG/MGC. ES 283 039-3, Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN), June 2007.

[ETSI07b] ETSI. NGN Congestion and Overload Control; Part 4: Overload and Congestion Control for H.248 MG/MGC. ES 283 039-4, Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN), April 2007.

[ETSI10] ETSI. NGN Congestion and Overload Control; Part 2: Core GOCAP and NOCA Entitiy Behaviours. ES 283 039-2, Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN), January 2010.

[Fabini13] J. Fabini, M. Hirschbichler, J. Kuthan, and W. Wiedermann. Mobile sip: An empirical study on sip retransmission timers in hspa 3g networks. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013.

[Fajardo12] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn. Diameter Base Protocol. RFC 6733, Internet Engineering Task Force, October 2012.

[Faltstrom04] P. Faltstrom and M. Mealling. The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM). RFC 3761, Internet Engineering Task Force, April 2004.

[Fielding99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.

[Franks99] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617, Internet Engineering Task Force, June 1999.

[Geng00] Xianjun Geng and A.B. Whinston. Defeating distributed denial of service attacks. *IT Professional*, vol. 2, no. 4, pp. 36–42, 2000.

[Gligor84] V.D. Gligor. A note on denial-of-service in operating systems. *Software Engineering, IEEE Transactions on*, vol. SE-10, no. 3, pp. 320–324, 1984.

[Gurbani12] V. Gurbani, V. Hilt, and H. Schulzrinne. Session Initiation Protocol (SIP) Overload Control. Internet-Draft draft-ietf-soc-overload-control-09, Internet Engineering Task Force, July 2012. Work in progress.

[Gurbani13] V. Gurbani, V. Hilt, and H. Schulzrinne. Session Initiation Protocol (SIP) Overload Control. Internet-Draft draft-ietf-soc-overload-control-13, Internet Engineering Task Force, May 2013. Work in progress.

[Handley99] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. RFC 2543, Internet Engineering Task Force, March 1999.

[Handley06] M. Handley, V. Jacobson, and C. Perkins. SDP: Session Description Protocol. RFC 4566, Internet Engineering Task Force, July 2006.

[Happenhofer11] M. Happenhofer and C. Egger. Implicit SIP proxy overload detection mechanism based on response behavior. In N. Rozic, editor, *Proc. 19th International Conference on Software, Telecommunications and Computer Networks - (SoftCOM 2011), Split, Croatia*. IEEE, November 2011.

[Hautakorpi10] J. Hautakorpi, G. Camarillo, R. Penfield, A. Hawrylyshen, and M. Bhatia. Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments. RFC 5853, Internet Engineering Task Force, April 2010.

[Hilt08] V. Hilt and I. Widjaja. Controlling overload in networks of sip servers. In *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, pages 83–93, 2008.

[Hilt11] V. Hilt, E. Noel, C. Shen, and A. Abdelal. Design Considerations for Session Initiation Protocol (SIP) Overload Control. RFC 6357, Internet Engineering Task Force, August 2011.

[Hirschbichler06] M. Hirschbichler. A Traffic Generator and Performance Analysis Toolkit for the IMS. *Diploma Thesis*, , 11 2006.

[Hirschbichler09] M. Hirschbichler, C. Egger, O. Pasteka, and A. Berger. Using E-Mail SPAM DNS Blacklists for Qualifying the SPAM-over-Internet-Telephony Probability of a SIP

Call. In *Digital Society, 2009. ICDS '09. Third International Conference on*, pages 254 –259, feb. 2009.

[Hirschbichler11] M. Hirschbichler. VoIP monitoring - reasons and solutions for automated voIP blackbox and long-term monitoring at the telekom austria group. In *SIP Forum SIPNOC*, Washington, USA, April 2011.

[Hirschbichler12] M. Hirschbichler, J. Fabini, B. Seifert, and C. Egger. Stop the flood perimeter security- and overload- preevaluation in carrier grade voip infrastructures. In Sergey Andreev, Sergey Balandin, and Yevgeni Koucheryavy, editors, *Internet of Things, Smart Spaces, and Next Generation Networking*, volume 7469 of *Lecture Notes in Computer Science*, pages 359–370. Springer Berlin Heidelberg, 2012.

[InfoSecurity] MWR InfoSecurity. The Google Android Update Dilemma.

[ipx07] IPX White Paper, Version 1.2. Technical report, GSM Association, March 2007.

[ITU-T05] ITU-T. H.248.1 : Gateway control protocol: Version 3. Recommendation H.248.1, International Telecommunication Union, Geneva, September 2005.

[ITU-T09] ITU-T. H.323 : Packet-based multimedia communications systems. Recommendation H.323, International Telecommunication Union, Geneva, December 2009.

[itu88] E.600: Terms and Definitions of Traffic Engineering. ITU-T, International Telecommunication Union, 1988.

[itu99] E.721: Network grade of service parameters and target values for circuit-switched services in the evolving ISDN. ITU-T, International Telecommunication Union, May 1999.

[itu04] I.371: Traffic control and congestion control in B-ISDN . ITU-T, International Telecommunication Union, 2004.

[Jennings02] C. Jennings, J. Peterson, and M. Watson. Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks. RFC 3325, Internet Engineering Task Force, November 2002.

[Kocher96] P. C. Kocher. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, volume 1109, pages 104–113. Springer, 1996.

[Könings] Bastian Könings, Jens Nickels, and Florian Schaub. Catching AuthTokens in the Wild – The Insecurity of Google's ClientLogin Protocol.

[Lear12] E. Lear, H. Tschofenig, H. Mauldin, and S. Josefsson. A Simple Authentication and Security Layer (SASL) and Generic Security Service Application Program Interface (GSS-API) Mechanism for OpenID. RFC 6616, Internet Engineering Task Force, May 2012.

[Liec] Wenhai Li, Wei Guo, Xiaolei Luo, and Xiang Li. On sliding window based change point detection for hybrid sip dos attack. In *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, pages 425–432, Dec.

[Mahy10] R. Mahy, P. Matthews, and J. Rosenberg. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766, Internet Engineering Task Force, April 2010.

[Malas11] D. Malas and A. Morton. Basic Telephony SIP End-to-End Performance Metrics. RFC 6076, Internet Engineering Task Force, January 2011.

[Mauro05] Andreolini Mauro, Bulgarelli Alessandro, Colajanni Michele, and Mazzoni Francesca. Honeyspam: Honeypots fighting spam at the source, pp. 77–83, 2005.

[Mulliner10] C. Mulliner and J.-P. Seifert. Rise of the ibots: Owning a telco network. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 71–80, 2010.

[Noel09] E. Noel and C.R. Johnson. Novel overload controls for sip networks. In *Teletraffic Congress, 2009. ITC 21 2009. 21st International*, pages 1–8, 2009.

[Noel12] E. Noel and P. PhilipWilliams. Session Initiation Protocol (SIP) Rate Control. Internet-Draft draft-ietf-soc-overload-rate-control-02, Internet Engineering Task Force, June 2012. Work in progress.

[Ordoñez07] Roderick Ordoñez. Captcha wish your girlfriend was hot like me? Technical report, TREND MICRO Deutschland GmbH, 10 2007.

[Peterson04] J. Peterson. Session Initiation Protocol (SIP) Authenticated Identity Body (AIB) Format. RFC 3893, Internet Engineering Task Force, September 2004.

[Peterson06] J. Peterson and C. Jennings. Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP). RFC 4474, Internet Engineering Task Force, August 2006.

[Phuyal12] U. Phuyal, A.T. Koc, Mo-Han Fong, and R. Vannithamby. Controlling access overload and signaling congestion in m2m networks. In *Signals, Systems and Computers (ASILOMAR), 2012 Conference Record of the Forty Sixth Asilomar Conference on*, pages 591–595, 2012.

[Quinten07] Vincent M Quinten, Remco Van De Meent, and Aiko Pras. Analysis of Techniques for Protection Against Spam over Internet Telephony. *Techniques*, pp. 70–77, 2007.

[Quittek08a] J. Quittek, S. Niccolini, S. Tartarelli, and R. Schlegel. On spam over internet telephony (spit) prevention. *Communications Magazine, IEEE*, vol. 46, no. 8, pp. 80 –86, august 2008.

[Quittek08b] Juergen Quittek, Saverio Niccolini, Sandra Tartarelli, and Roman Schlegel. On Spam over Internet Telephony (SPIT) Prevention. *IEEE Communications Magazine*, no. August, pp. 80–86, 2008.

[Ramachandran06] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06, pages 291–302, New York, NY, USA, 2006. ACM.

[Rivest92] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, Internet Engineering Task Force, April 1992.

[Roach02] A. B. Roach. Session Initiation Protocol (SIP)-Specific Event Notification. RFC 3265, Internet Engineering Task Force, June 2002.

[Rosenberg02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002.

[Rosenberg03] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489, Internet Engineering Task Force, March 2003.

[Rosenberg04a] J. Rosenberg. A Presence Event Package for the Session Initiation Protocol (SIP). RFC 3856, Internet Engineering Task Force, August 2004.

[Rosenberg04b] J. Rosenberg. A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP). RFC 3857, Internet Engineering Task Force, August 2004.

[Rosenberg07] J. Rosenberg. The Extensible Markup Language (XML) Configuration Access Protocol (XCAP). RFC 4825, Internet Engineering Task Force, May 2007.

[Rosenberg08a] J. Rosenberg. Requirements for Management of Overload in the Session Initiation Protocol. RFC 5390, Internet Engineering Task Force, December 2008.

[Rosenberg08b] J. Rosenberg and C. Jennings. The Session Initiation Protocol (SIP) and Spam. RFC 5039, Internet Engineering Task Force, January 2008.

[Rosenberg08c] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389, Internet Engineering Task Force, October 2008.

[Schlegel06] Roman Schlegel, Saverio Niccolini, Sandra Tartarelli, and Marcus Brunner. ISE03-2: SPam over Internet Telephony (SPIT) Prevention Framework. *IEEE Globecom 2006*, pp. 1–6, November 2006.

[Schulzrinne03] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, Internet Engineering Task Force, July 2003.

[Schulzrinne06] H. Schulzrinne and J. Polk. Communications Resource Priority for the Session Initiation Protocol (SIP). RFC 4412, Internet Engineering Task Force, February 2006.

[Seo08] Dongwon Seo, Heejo Lee, and Ejovi Nuwere. Detecting more sip attacks on voip services by combining rule matching and state transition models. volume 278 of *IFIP International Federation for Information Processing*, pages 397–411. Springer Boston, 2008.

[Srivastava05] Kumar Srivastava and Henning Schulzrinne. Preventing spam for sip-based instant messages and sessions. In *Department of Computer Science, Columbia University, Technical Report*, oct 2005.

[Tauzin03] Sponsor: Rep. William Billy Tauzin. H.R. 395 (108th): Do-Not-Call Implementation Act, , January 2003.

[TKG] Telekommunikationsgesetz, . 9.11.2011 (BGBl. I Nr. 70/2003).

[Turing50] Alan Mathison Turing. Computing machinery and intelligence. In *Mind*, pages 433–460, 1950.

[Whitehead02] Martin Whitehead and Philip Williams. Adaptive network overload controls. *BT Technology Journal*, vol. 20, no. 3, pp. 31–54, 2002.

[Whitehead05] Martin Whitehead. Gocap one standardised overload control for next generation networks. *BT Technology*

Journal, vol. 23, pp. 147–153, 2005. 10.1007/s10550-005-0115-1.

[Wing07] D. Wing. Symmetric RTP / RTP Control Protocol (RTCP). RFC 4961, Internet Engineering Task Force, July 2007.

[Wing08] D. Wing, S. Niccolini, M. Stiemerling, and H. Tschofenig. Spam Score for SIP. Internet-Draft draft-wing-sipping-spam-score-02, Internet Engineering Task Force, February 2008. Work in progress.

[Wong06] M. Wong and W. Schlitt. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408, Internet Engineering Task Force, April 2006.