# Characteristics of Real Open SIP-Server Traffic

Jan Stanek[1], Lukas Kencl[1], and Jiri Kuthan[2]

[1] Czech Technical University in Prague
Technicka 2, 166 27 Prague 6, Czech Republic
{jan.stanek,lukas.kencl}@fel.cvut.cz
[2] Tekelec
Am Borsigturm 11, 13507 Berlin, Germany
jiri.kuthan@tekelec.com

**Abstract.** Voice-over-IP (VoIP) is currently one of the most commonly used communication options and Session Initiation Protocol (SIP) is most often used for VoIP deployment. However, there is not a lot of general knowledge about typical SIP traffic and research in this area largely works with various assumptions. To address this deficiency, we present a thorough study of traffic of a real, free and publicly open SIP server. The findings reveal, among others, a surprisingly high overhead of SIP due to connection maintenance through Network Address Translation (NAT) nodes, differences from typical Web server Zipf's-law patterns and various unexpected creative uses of SIP servers.

## 1 Introduction

With proliferation of Voice-over-IP (VoIP) communication, its infrastructure – the signaling protocol (Session Initiation Protocol (SIP) [1, 2]) and the control nodes (SIP servers) – attracts much interest from the perspective of practical functioning within the Internet infrastructure. SIP traffic characteristics form a necessary basis for considerations of deployment in the light of emerging trends. SIP is lightweight, has easily understandable and human-readable structure and as a signaling protocol, it does not generate much traffic by itself. A lot of research is dedicated to VoIP improvement in reliability, stability, quality of service, security and other areas. However, publicly available analyses of SIP traffic are rare and thus not a lot of knowledge exists in the networking community about typical behavior of SIP servers (as opposed to, e.g. HTTP servers). Typically, a lot of assumptions are made about SIP traffic, but are they realistic?

The goal of this paper is to study properties of real SIP-server traffic. We analyze a freely accessible and open SIP server with a worldwide user base. Anyone can register and use this SIP server, free of charge and with no restrictions on the client device. A vibrant and multi-faceted community of users from all around the world form the user base, using various SIP devices, including proprietary ones. These often do not behave exactly as they should. This creates very interesting and sometimes non-standard SIP traffic that we also examine in this paper.

M. Roughan and R. Chang (Eds.) PAM 2013, LNCS 7799, pp. 187–197, 2013.

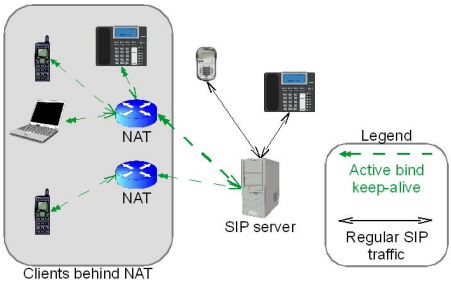



**Fig. 1.** Server-based connection maintenance through NAT

In more detail we study the problem of Network Address Translation (NAT) traversal, still not well-solved in SIP. The server and clients typically need to execute other mechanisms (STUN [3], TURN [4], ICE [5] etc.) to facilitate NAT traversal. The monitored server, too, uses its own mechanism to keep NAT connections alive (see Fig. 1).

The main contributions of this work are as follows:

- a thorough traffic analysis of three days of an operational SIP server, including geographic spread and time-series analysis;
- identification, measurements and demonstration of the large overhead in SIP traffic caused by NAT and keep-alive mechanisms;
- discussion of the findings and suggestion of possible remedies;
- analysis of geographic traffic distribution.

## 2 Related work

Sparks [6] and Prasad and Kumar [7] describe SIP basics and some typical extensions used. The problem of NAT traversal in SIP and its various solutions have been described in numerous papers e.g by Yeryomin et al. [8] and Song et al. [9]. Despite the many proposals for solving the NAT traversal issue, some of which are very popular (STUN, ICE), none of these was adopted generally as they impose strict extra requirements on the SIP clients and servers. Much work has also focused on identifying SIP anomalies. Heo et al. [10] studied the use of statistical distances for respective SIP message types, Cortes et al. [11] used performance metrics to identify SIP processing times and Kang et al. [12] used known profilers on various real SIP datasets. Ehlert et al. [13] used decision modules and Nassar et al. [14] vector machines to detect anomalous SIP traffic.

Significant amount of papers was dedicated to SIP security. Hentehzadeh et al. [15] used statistical approach to filter out potentially hazardous SIP traffic. Akbar et al. [16] proposed using evolutionary algorithms, able to adapt to the changing nature of SIP traffic. Sisalem, Kuthan and Ehlert [17] proposed some best practices for secure SIP deployment. Despite this body of works, many SIP deployments remain unprotected even against the simplest of attacks.

Considering cloud environment, deploying only SIP server itself in cloud is not economical due to its typical stable and continuous resource consumption. However, next to SIP there is a large body of related media traffic (typically in the form of RTP streams) and routing it through the best available paths is an important open problem. One solution, using anycast, was proposed by Andel et al. [18]. Considering the distributed nature of the cloud and its location-awareness, using it to deploy media relays might be beneficial.

## 3 SIP Server Setup and Dataset Description

We have analyzed SIP traffic from a publicly, globally and freely available SIP server a long time in operation. It is configured so that it expects user devices not to be able to do any advanced tasks and tries to solve any issues (such as maintaining the bindings for clients behind NAT) on the server side. Users are encouraged to use it for testing of their experimental set-ups and devices. Despite these differences from a typical commercial SIP server, analysis of its traffic is very useful as it exhibits many interesting and unexpected aspects of SIP traffic. Observations based upon this traffic are generally applicable since a lot of the "unexpectances" comes from client behavior or misconfiguration.

The respective iptel.org SIP server runs on a single host, located in Berlin, Germany. It is an instance of SIP Express Router (SER) [19], running on top of Linux on a Dell Poweredge 1850 server. The only services running on the server are SER, STUN (mystun) and a UDP relay for NAT traversal. The processing capacity is sufficient to process typical SIP-server traffic. About 3400 users are typically registered generating call rate between 2 and 50 concurrent calls.

We captured all network traffic on the server continuously for 67 hours using tcpdump [20]. The capture started on March 16 2012 at 13:00 GMT and ended on March 19 2012 at 8:00 GMT, its size is 40GB and it contains about 85 million packets. Of these, about 39GB are SIP and 1GB non-SIP packets, yet about 74 million are SIP and 11 million non-SIP packets. The difference in size and packet counts is caused by the non-SIP packets being mostly very small – TCP related, or UDP keep-alives.

Note that we focus solely on the signaling traffic. We do not analyze the media traffic as it is does not influence the signaling processing.

## 4 Data Analysis Methodology

We split the larger traffic dump so we could process it faster in parallel. We had 3 servers available so we split the dump accordingly – the first two parts corresponding to the first two days and the third to the remaining 19 hours. We used programs from the Wireshark [21] program suite for trace splitting and the various analyses. We divide the analysis into several phases:

*Phase 1 - Load over time.* We focused on the evolution of load over time. Using Wireshark we extracted the timestamp in unix time format from every packet.

We then imported these timestamps to Matlab and created a timeline for each part. As the timestamps in plaintext format consumed about 2GB of disk space, we aggregated them to blocks of 1 second for faster processing.

*Phase 2 - SIP request types.* We analyzed types of SIP requests and responses contained in the dump. We used tshark (a command-line version of wireshark) to produce SIP statistics (*tshark -z sip,stat*) i.e. a list of SIP message types with numbers of occurrences per each type observed. As the three blocks are still too large to be efficiently processed by tshark, we split them again to parts of 250000 SIP packets in size. The resulted reports were concatenated and an awk script was used to extract the necessary information. Final aggregation was executed in Matlab, producing a sorted lists of SIP requests and responses.

*Phase 3 - Traffic sources.* We analyzed the traffic according to its sources. We reused the parts from Phase 2 and analyzed them using tshark, now with *-z conv, udp/tcp* options. These options aggregate traffic between source and target locations (identified by IP address and port pair) and produce summary containing numbers of packets (bytes, frames) exchanged between every pair of locations that communicated with each other. These data were again aggregated and cleaned using awk and imported to Matlab for results generation.

*Phase 4 - Geographic distribution.* We analyzed the geographic spread of users. We used a userloc – text database of registered users containing all the user information the SIP server stores (including IP addresses). We extracted the IP addresses using awk and mapped them to their geographic locations using the WebNET77 Multiple IP address lookup tool [22].

*Phase 5 - Geographic locations.* Finally, we wanted to obtain geographic location for the caller and callee per every call observed. We did this by filtering the INVITE and REGISTER messages and extracting information about mapping of IP addresses to individual users. We also created a list of calls by separating unique INVITEs (e.g. filtering out re-INVITES). Then we loaded the IP addresses into the webnet tool [22]. We saved the resulting IP to location mapping and replaced IP addresses with locations. Some calls were still unmappable since there were also reserved and local IP addresses present, but for the rest we obtained the list of calls in pairwise format (from country - to country).

## 5 SIP Server Traffic Analysis

### 5.1 General Properties

SIP traffic on the studied server is generated by about 3400 active SIP User Agents, mostly in Europe (41%), North America (25%) and Asia (22%). Other continents are less involved (South America 6%, Africa 3% and Oceania 3%).

Some of the clients are represented by bulk services, such as ipkall or virtualpbx. The User Agents are represented by about 280 distinct SIP client implementations. The distribution of client types can be seen in Fig. 2. The most frequently used SIP clients are AVM Fritzbox and Draytek (about 10% each).

Various Linksys devices represent a significant fraction (12%) as well. Another interesting aspect is the presence of mobile clients, representing about 12% of users. The most used mobile SIP client implementations are Comrex (7.7% of all clients), Acrobits(2.2%) and sipdroid(1.4% ).

For this signaling-traffic analysis, we prefer to focus on the count of signaling messages rather than its (tiny) byte volume, as it is rather the former which influences SIP-server performance.

The split of traffic among TCP and UDP in numbers of packets is about 9:91 (less than 10% of traffic consists of TCP packets). Interestingly, only 150 out of the 3308 registered users were registered as using TCP (150/3308 = 4.5%), showing that the TCP clients generate more traffic than the UDP ones.

A graph of a typical one-day traffic can be seen in Fig. 3(a). About 8600 calls are made per day and there are about 3400 registered users (slight fluctuations, but no big changes if we do not count server outages that were observed in the traffic dumps - reason unknown, probably network failure). Calls express very strong 24-hour stability with just a small increases during prime-time. This is due to the high amount of options/keep-alives in the traffic (see Fig. 3(b)), caused by the deployed NAT traversal solution (notice that number of clients behind NAT does not fluctuate, so the generated keep-alive traffic is stable).

The division of load among the request and response message types can be seen in Fig. 4(a) and 4(b).

### 5.2 NAT Traversal

NAT traversal is still generally an unsolved problem of SIP. STUN is not reliable, ICE is often not implemented and usage of media-relays invokes notable overhead. Many clients and servers thus implement proprietary solutions – the most common one being periodic sending of "keep-alives" – messages with predefined content (OPTIONS and REGISTER being used most often). These messages are supposed to keep the NAT binding alive, thus solving the problem.

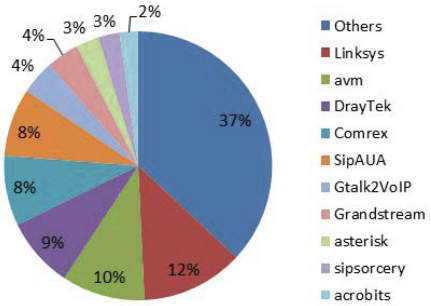
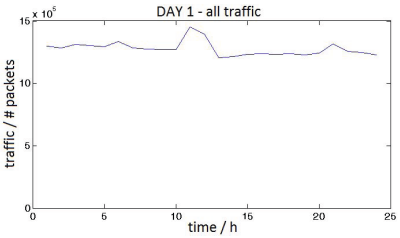
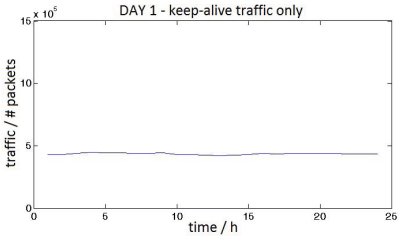



**Fig. 2.** Distribution of user devices




(a) SIP traffic, total.

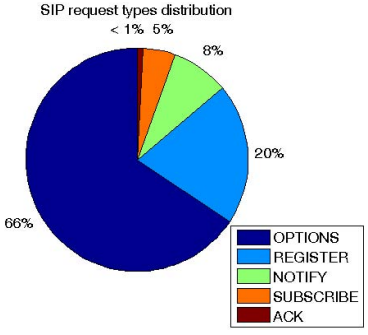



(b) SIP traffic, OPTIONS only.

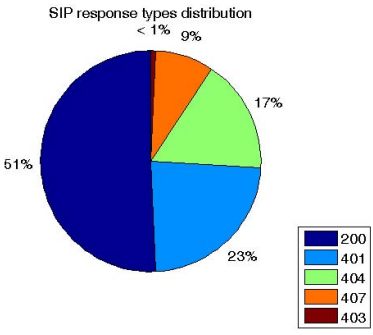
**Fig. 3.** Measurements of one day traffic

About 1500 out of the 3400 total users (44%) were detected to be behind NAT (from the view of the SIP server, actual number might be higher as some clients solve it on their own e.g. using ALGs). As the SIP server is open and supposed to support even proprietary SIP clients, it must use its own keep-alive messages to prevent eventual loss of NAT bindings. The clients, however, are not aware of this proactive server solution and run their own proprietary keep-alives. If the keep-alive messages were not sent in short intervals, NAT bindings would expire and clients would become unavailable for incoming calls. This results in an enormous part of the observed traffic (over 50%, see below for more detail) being just an exchange of proprietary keep-alive messages.

Considering the server-side NAT keep-alives only, every 15s there is one OPTIONS message sent per every client detected as behind the NAT, resulting in roughly 8.64 million keep-alive requests per day. Not counting the responses to these requests, this corresponds to about 33% of the total server traffic (there are about 16.2 million requests and 9 million responses per day in total). Only about 10% of the clients were able to respond to them, meaning that about 90% of this excessive traffic was effectively useless.




(a) SIP request types




(b) SIP response types

**Fig. 4.** Overall distribution of SIP message types

The REGISTER requests represent about 20% of total traffic. This is mainly due to every client re-registering after a predefined amount of time (default 600s) and some clients using the REGISTER requests as their NAT keep-alive mechanism. Note that a successful registration in SIP requires two REGISTER messages (first response indicates that authorization is necessary and provides necessary information) and a REGISTER request with the Expires attribute set to 0 is used for deregistration. Concretely, we have about 3400 clients, so counting two REGISTER requests per a successful registration this amounts to about 979200 REGISTER requests per day. In fact we observe about 3.3 million REGISTER messages per day, so probably the rest are client NAT keep-alives. The INVITE and BYE messages together represent only about 0.19% of all requests per day (30564 out of 16154468).

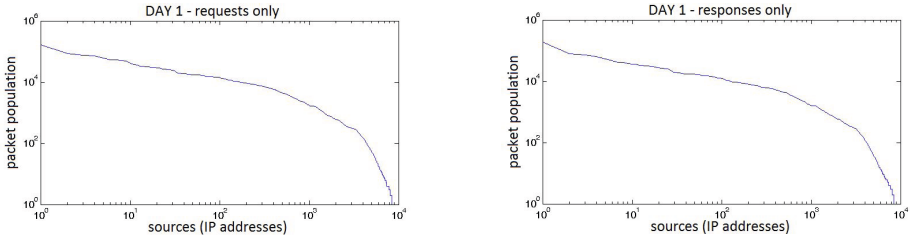
In summary, NAT traversal from the server causes overhead of 33%. The keep-alives from clients we can only estimate, but thanks to the high numbers of OPTIONS and REGISTERs we can conjecture that keep-alive messages may amount to over 50% of the total traffic.

### 5.3 HTTP Server / SIP Server Workload Comparison

It is a well known observation that typical network traffic follows Zipf's law. We analyzed this aspect in SIP traffic as well. For comparison, we have used data of HTTP traffic from a freely available source at [23]. These data show a very clear Zipf's law observable by a straight line in a graph on a log-log scale. We have created log-log graphs of SIP traffic of the first day of our capture, isolating the requests and responses, see Fig 5(a) and Fig.5(b).

Obviously, this traffic does not exhibit Zipf's law characteristics. The difference is caused by absence of sources with very low packet count ("mice"). This can be expected due to the periodic registration property of SIP that leads to periodically generated traffic and so one can hardly find a user sending only a few SIP packets (at least 2 packets are sent during registration and re-registration, occurring every 10 minutes).

SIP traffic thus does not generally follow Zipf's law and assumptions valid for network traffic should be handled with care when applied to SIP.




(a) SIP requests (b) SIP responses

**Fig. 5.** Log-log scale graphs of captured SIP traffic, Day 1

### 5.4 Geographic Traffic Distribution

We have analyzed the division of traffic among countries and continents. Due to limited space in the paper, we present only the continental-level traffic division in Tables 1– 3. Clearly, the traffic is typically intra-continental (the non-zero numbers are on the diagonal in the tables). There is one large deviation though and that is traffic from North America to Asia. We scrutinized this unexpected phenomenon and found out a service called Virtualphoneline that offers American phoneline numbers and routes them via SIP anywhere the customers need. This way, for example a Chinese merchant can have an American helpline for his American customers automatically rerouted to his offices in China. Very useful and very unexpected to be run on a public, test-oriented SIP server. With Virtualphoneline traffic subtracted, however, the calls are basically carried-out intra-continentally, thus exhibiting geographic locality.

### 5.5 Registration Storms

Our analysis has shown that there is a big difference between normal traffic and traffic during an unexpected event such as a registration storm. Users are enforced to renew their registrations periodically (once each 600s). Considering a theoretical situation where every client re-registers exactly after 600s, then having 3400 clients one would expect traffic of approximately 12 REGISTER requests per second. Many users are however re-registering more often. Our data show that over 35% of the users have their re-registration interval set under 540s. When we measured the real registration load over a 600s period we obtained load of approx 65 requests per second (rps), much higher than the theoretical expectation. We have also observed a few server outages. When measuring load after an outage longer than 600s, it reached up to 900 rps after service renewal!

For a SIP service to be reliable, the server must be able to process the traffic even in the worst-case scenario of a registration storm. Filtering would not help as all the REGISTER requests are valid and must be processed. Overprovisioning is sure to help, but would be quite expensive. A highly distributed cluster might work, splitting outages to small parts served from different locations.

**Table 1.** Day 1

| | AF | AS | EU | NA | OC | SA |
|---|---|---|---|---|---|---|
| AF | 1 | 0 | 2 | 4 | 0 | 0 |
| AS | 1 | 169 | 4 | 26 | 5 | 0 |
| EU | 20 | 19 | 213 | 0 | 0 | 0 |
| NA | 19 | 355 | 18 | 114 | 1 | 1 |
| OC | 1 | 8 | 0 | 0 | 0 | 0 |
| SA | 0 | 0 | 0 | 0 | 0 | 11 |

**Table 2.** Day 2

| | AF | AS | EU | NA | OC | SA |
|---|---|---|---|---|---|---|
| AF | 1 | 0 | 4 | 4 | 0 | 0 |
| AS | 0 | 71 | 1 | 3 | 23 | 0 |
| EU | 10 | 0 | 109 | 2 | 12 | 0 |
| NA | 16 | 589 | 26 | 74 | 0 | 18 |
| OC | 0 | 7 | 11 | 0 | 0 | 0 |
| SA | 0 | 0 | 1 | 4 | 0 | 3 |

**Table 3.** Day 3

| | AF | AS | EU | NA | OC | SA |
|---|---|---|---|---|---|---|
| AF | 2 | 0 | 0 | 8 | 0 | 0 |
| AS | 2 | 289 | 0 | 1 | 5 | 0 |
| EU | 8 | 0 | 135 | 4 | 2 | 1 |
| NA | 3 | 202 | 5 | 148 | 1 | 1 |
| OC | 1 | 1 | 2 | 0 | 1 | 0 |
| SA | 0 | 0 | 2 | 0 | 0 | 0 |

Continental-level division of SIP traffic observed. Abbreviations stand for continents. AF Africa, AS Asia, EU Europe, NA N. America, OC Oceania, SA S. America.

## 6 Conclusion

The biggest issue we encountered, a large amount of "waste" traffic, stems from misconfiguration and poor implementation of user devices and improper use of NAT traversal mechanisms. Requests are often invalid, user devices unable to process valid but specific SIP messages. A large portion of the traffic is thus "wasted", as the requests are not properly handled (about 90% of servers' keep-alive traffic was effectively useless.). These results show that it is imperative to filter SIP traffic, whereby overall traffic could potentially be reduced by 60-70%. Note that in the case of the non-signaling, media traffic (e.g. VoIP calls), the overhead analysis might end-up differently, however, this paper focuses solely on issues of signaling.

Proper anomaly detection could help and it would also address another vital issue – security and protection of SIP infrastructure. Anomaly detection is generally hard – a lot of wasteful traffic and proprietary behavior complicates the analysis. A public SIP service must be very liberal in what traffic it receives, it must be prepared to deal with clients' imperfections and compensate for these, but at the same time it must not be prone to DoS attacks. Therefore anomaly detection should be carried out separately. These requirements should be always considered when deploying a SIP server as improper anomaly detection leads to unexpected problems. Some of the common security solutions, based on traffic limiting, might conflict with the "unexpected but harmless" behavior of misconfigured clients, thus blocking the service for legitimate clients. Service outage then exacerbates the problem as legitimate clients generate valid REGISTER requests which might create a false positive flood attack.

Our analysis showed that it not necessarily advantageous for SIP servers to be migrated into cloud datacenters, as SIP traffic is very stable and does not exhibit excessive spikes in terms of resources consumption. However, SIP traffic and, more importantly, media traffic, also exhibits strong geographic locality. This could be exploited using a cloud-based architecture since cloud providers usually allow the choice of a geographic location for running the services. Therefore we propose using distributed media relays in cloud. These could then be positioned according to the needs of the individual domains/zones and one could avoid the triangular routing introduced by UDP relays.

As future work we intend to analyze the registration storms and possibilities of minimizing their impact. We also plan to create a SIP-trace anonymizer so that our SIP traffic data can be made publicly available. Due to the nature of SIP, it is not possible to release the data simply IP-anonymized because it still contains interconnected personal information (usernames, domains, locations etc.) spread across different parts of multiple SIP messages. In spite of our every intention for public release of the data, proper anonymization that would maintain the analytical value appears to be a research problem on its own and will take some time to solve. Until the anonymizer is finished (we plan for Q2 2013 the latest) we can only share the data analyzed in this paper under NDA (please do not hesitate to contact us, if interested).

**Acknowledgment.** We thank iptel.org for generously providing access to its SIP-server traffic records. We also thank P. Kasparek and V. Kubart for help with data analysis.

## References

1. Handley et. al. Sip: Session initiation protocol (rfc 2543), `http://www.ietf.org/rfc/rfc2543.txt`
2. Rosenberg et. al. Sip: Session initiation protocol (rfc 3261), `http://www.ietf.org/rfc/rfc3261.txt`
3. Rosenberg, J., et al.: Session traversal utilities for nat (stun), `http://tools.ietf.org/html/rfc5389`
4. Mahy, R., et al.: Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun), `http://tools.ietf.org/html/rfc5766`
5. Rosenberg, J.: Ice: A protocol for network address translator traversal for offer/answer protocols, `http://tools.ietf.org/html/rfc5245`
6. Sparks, R.: Sip: Basics and beyond. Queue 5(2), 22–33 (2007)
7. Prasad, J.K., Kumar, B.A.: Analysis of sip and realization of advanced ip-pbx features. In: ICECT 2011, vol. 6, pp. 218–222 (April 2011)
8. Yeryomin, Y., Evers, F., Seitz, J.: Solving the firewall and nat traversal issues for sip-based voip. In: ICT 2008, pp. 1–6 (June 2008)
9. Song, M., Chi, J., Pi, R., Song, J.: Implementing an express sip nat traversal server. In: ICPCA 2007, pp. 527–529 (July 2007)
10. Heo, J., Chen, E.Y., Kusumoto, T., Itoh, M.: Statistical sip traffic modeling and analysis system. In: ISCIT, pp. 1223 –1228 (2010)
11. Cortes, M., Ensor, J.R., Esteban, J.O.: On sip performance. Bell Labs Technical Journal 9(3), 155–172 (2004)
12. Kang, H.J., Zhang, Z.-L., Ranjan, S., Nucci, A.: Sip-based voip traffic behavior profiling and its applications. In: MineNet 2007, pp. 39–44 (2007)
13. Ehlert, S., Wang, C., Magedanz, T., Sisalem, D.: Specification-based denial-of-service detection for sip voice-over-ip networks. In: Internet Monitoring and Protection, ICIMP 2008, June 29 -July 5, pp. 59–66 (2008)
14. Nassar, M., State, R., Festor, O.: Monitoring SIP Traffic Using Support Vector Machines. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) RAID 2008. LNCS, vol. 5230, pp. 311–330. Springer, Heidelberg (2008)
15. Hentehzadeh, N., et al.: Statistical analysis of self-similar session initiation protocol (sip) messages for anomaly detection. In: 2011 4th IFIP Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5 (February 2011)
16. Ali Akbar, M., Farooq, M.: Application of evolutionary algorithms in detection of sip based flooding attacks. In: GECCO 2009 (2009)
17. Sisalem, D., Kuthan, J., Ehlert, S.: Denial of service attacks targeting a sip voip infrastructure: attack scenarios and prevention mechanisms. IEEE Network 20(5), 26–31 (2006)
18. Andel, L., Kuthan, J., Sisalem, D.: Distributed media server architecture for sip using ip anycast. In: IPTComm 2009, pp. 5:1–5:11 (2009)
19. Community of developers. The sip router project (developed from openser), `http://sip-router.org/`

20. Van Jacobson, Leres, C., McCanne, S., many later contributors: Tcpdump: Commandline packet analyzer, http://www.tcpdump.org/
21. Combs, G., contributors: Wireshark - network protocol analyzer, http://www.wireshark.org
22. WEBNet77. Ip to country multi-lookup tool, http://software77.net/geo-ip/multi-lookup/
23. ACM SIGCOMM partners. The internet traffic archive, http://ita.ee.lbl.gov/html/traces.html