Towards effective SIP load Balancing

Georgios Kambourakis | Dimitris Geneiatakis | Tasos Dagiuklas | Costas Lambrinoudakis | Stefanos Gritzalis

Laboratory of Information and Communication Systems Security
Department of Information and Communication Systems Engineering
University of the Aegean
Karlovassi, GR-83200 Samos, Greece
Tel: +30-22730-82247

Email:{gkamb, dgen, ntan, clam, sgritz}@aegean.gr

ABSTRACT

Session Initiation Protocol (SIP) high availability, reliability and redundancy are determined by the ability of the core SIP network components to offer high quality SIP services in the event(s) of high call transactions, link outages, device failures, misconfigurations and security attacks. In this context, load balancers can be used to achieve redundancy and active load balancing of SIP transactions. In load balancing schemes, new requests are allocated across available servers using a selection algorithm. Although considerable work has been already done for Web traffic balancing, little research effort is primarily aiming to SIP load balancing. This paper proposes a SIP dedicated load balancing solution, which is currently under development within the EC funded project SNOCER. We describe in detail our balancing scheme, its associated architecture elements and provide implementation details showing that it is simple to realize, effective, flexible, robust and secure.

Categories and Subject Descriptors

C2.1 [**Computer Communication Networks**]: Network Architecture and Design, C.2.2 Network Protocols.

General Terms

Design, Reliability, Security.

Keywords

Session Initiation Protocol; Load balancing; SIP architectures; Redundancy.

1. INTRODUCTION

Large scale corporate Voice over IP (VoIP) service may necessitate the deployment of multiple servers in order to serve transactions requested by several VoIP clients concurrently. Multiple installed servers or even clusters of servers aim at "absorbing" smoothly heavy VoIP traffic so that VoIP services can be delivered unattended without delays contributing to the high quality of service (QoS).

Generally, in load-balancing schemes, new requests are assigned to existing servers following a pre-determined algorithm. A common selection algorithm targeting in statistical load balancing is the well known round-robin scheme [1], which has to be better considered as a load distribution option rather than a "pure" load balancing mechanism. Another eminent category of balancing approaches is weighted or adaptive balancing, which distributes requests proportional to the weight assigned to each available choice or route.

More specifically, load balancing can be adaptive or not adaptive, depending on whether or not run-time load conditions influence load balancing decisions. Adaptive load balancing policies use real-time system state information based on various metrics (e.g. CPU consumption, free available memory etc), to take load balancing decisions, whereas non-adaptive or static load balancing do not. In any case, to be able to distribute effectively and fairly VoIP traffic in the corresponding redundant servers, the introduction of the appropriate balancing mechanism during the initiation of the call is considered as the most crucial factor. In a VoIP environment the call establishment is accomplished through the utilization of standard signalling protocols like H.323, SIP, MGCP, etc. However, Session Initiation Protocol (SIP) seems to overwhelm all the rest, mainly due to the fact that it has been adopted by various standardisation organisations (i.e. IETF, ETSI, 3GPP) as the protocol to establish multimedia sessions at both wireline and wireless world in the Next Generation Networks (NGN) era. There are different ways to calculate the SIP servers' workload and different schemes to deal with it. However, in practice, the effectiveness of adaptive load balancing depends on the load metrics chosen and on other load run-time parameters needed.

While many load balancing strategies and various techniques (see Section 2.1) have been considered and thoroughly tested mainly for Web servers, significant research must be targeted towards loading-balancing schemes for real-time services. Current SIP servers' implementations do not include native SIP balancing modules and usually rely on add-on or peripheral balancing methods. To the best of our knowledge, the only SIP-oriented balancing scheme is the Vovida's one (www.vovida.org) which, at least until know, is far from being complete [2]. This paper focuses on load SIP balancing, proposing a novel but simple approach that can be implemented in any SIP realm. We address

VSW'06, June, 2006, Berlin, Germany.

architecture requirements and describe in detail how the proposed solution can be applied, discussing alternative scenarios. As a final point, we also touch on some failover issues and security parameters that stem from the presentation and are considered important for the proposed scheme to work.

The rest of this paper is structured as follows: Section 2 provides a short introductory to balancing techniques focusing on SIP and presents our solution in general. Section 3 analyses the anticipated balancing scheme in detail presenting its interactions with the other architecture components, discussing its advantages and showing that is robust and simple enough to realize. Last Section concludes the paper and gives pointers to future work.

2. SIP BALANCING PRELIMINARIES

2.1 Load Balancing Schemes

Some well-known solutions in the literature mainly about Web servers balancing include the following (here we have made some adaptations for the SIP case):

Round Robin (RR): Upon to a new SIP request, the SIP balancer selects the next IP address record for the specific SIP server alias name as stored in the Domain Name System (DNS). This solution is considered as non adaptive, due to the fact that it does not require from the balancer to maintain and update workload information from the available SIP servers in the domain. For Web balancing this mechanism is built-in into the DNS. One will argue that SIP load balancing can be applied directly to the DNS as it is already supported for other Web services. Using this principle, there is no real need for the balancer independent machine. However, the DNS will assign one of IP addresses of SIP servers to each address resolving request. As a result, the name-to-address mapping will be cached in name servers along the path from the DNS all way down to the client and consequent address requests reaching the same name servers will be resolved with cached addresses. Only after the name-to-address mapping in the cache expires, due to Time-to-Live (TTL) field, is possible to serve new requests with a fresh RR decision. Additionally, RR is proved ineffective when all servers do not have equal processing performance and resources. Even the improved Two-Tier-Round-Robin (RR2) algorithm [3], which takes into account client domain information, does not really improve balancing for SIP applications.

Adaptive versus Dynamic Weighting: This means that it is selected the server which currently handles the least number of SIP transactions or it has the minimum overall workload. This category of solutions for Web servers includes *Asynchronous Alarms* (AAlarm) [4], *Ibnamed* [5] and *TENBIN* [6] algorithms for DNS. For the SNOCER architecture, we propose the following general balancing schema:

- The Load Balancer (LB) module (process or daemon) at fixed time intervals (e.g. 30 minutes) requests the current number of SIP servers (SRV records) [7] available within the DNS server and resolves their Fully Qualified Domain Names (FQDNs). Clearly, this means that for each SIP domain, the DNS server has multiple SRV records corresponding to (redundant) SIP proxies attached to it.

- After that, the LB at predefined or dynamically assigned intervals (say 120 seconds) queries all the available SIP home network proxies for information like CPU usage percentage and memory consumption. Generally, the following examples of "availability" metrics can be taken into account: Performance of each server, load average of CPU in each server, network load, network distance (hops) between a SIP client and a SIP server, vote from users, using predefined availability as add-on to dynamic load balancing. Based on the returned data, the LB generates a SIP proxy preference list from the least to the most utilized. In addition, every SIP Home proxy can send at regular thick-time intervals heartbeats to the LB machine saying "I am alive". Therefore, in case a SIP proxy goes down or suffers a sudden attack, the LB will be indirectly informed.

In the proposed load balancing method, when a SIP client requests to initiate a SIP call, the proposed entity called "SIP Load Balancer" (LB) will perform a selection of the most appropriate SIP server, based: (a) on the DNS SRV records of the available SIP servers and (b) on various workload metrics collected from the existing SIP proxy servers, such as locations of the client and SIP server, SIP server load in terms of processing transactions, overall workload in each SIP server, etc. Clearly, the selection of a certain proxy to serve any initial request is completely transparent to the client. One can argue that there is no real need for the LB to query each time the SIP proxies consuming network bandwidth, rather simply keep locally statistics about how many transactions have been assigned to each of them. However, this solution is rather static as: (a) does not provide real time load metrics because as one or more SIP jobs finish in the proxies there is no way to inform back the LB and (b) in case a SIP proxy goes down e.g. due to hardware failure, the LB will continue to send it new transactions to dispatch. For the above reasons, we consider the frequent communication between the LB and SIP proxies as mandatory.

2.2 Load Balancing within SNOCER Architecture

This section describes the use of the load balancing scheme within the SNOCER architecture. The LB network element will reside inside the internal High Availability Network (HAN) as described in Figure 1. Key components of the high availability SIP architecture within the SNOCER architecture, as proposed in [8], are the following:

- Bastion Host: This host acts as a gatekeeper into the internal VoIP network of the operator. Its assignment is to detect basic attacks on the VoIP systems and deny access to unsolicited traffic into the network through a firewall.
- Enhanced SIP Proxy: SIP proxies are enhanced in two ways. An integrated IDS system will be able to detect more sophisticated attacks on the SIP proxy, which the bastion host might have missed. Furthermore, the proxy's performance is optimised through the addition of a specialized DNS module to enhance throughput capabilities and repel DNS related attacks.
- The High Availability Network: Key components of the VoIP network will be secured trough an internal high availability network providing failover capabilities to these components.
- Operator's console: At a centralized point the status of the enhanced VoIP infrastructure can be observed and controlled.

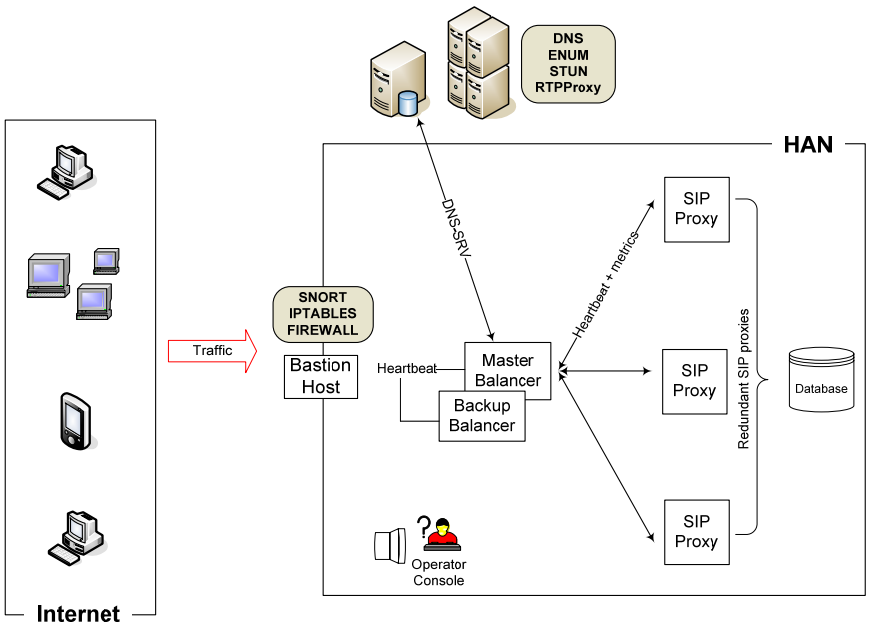

Figure 1. General high availability architecture for SIP domains

3. IMPLEMENTATION DETAILS

The LB is an add-on entity which is responsible to query DNS and maintain SRV records of all the available SIP proxies in the corresponding domain. For each SIP client's request the LB is responsible to forward the request to the most appropriate SIP proxy, in terms of workload, to serve it. In other words, as described in Figures 2 and 3, SIP clients firstly communicate with the LB entity to find out the best SIP proxy available. If the LB is not responding the SIP client can communicate directly with the DNS to retrieve all the available SRV records corresponding to SIP servers in the domain and select one.

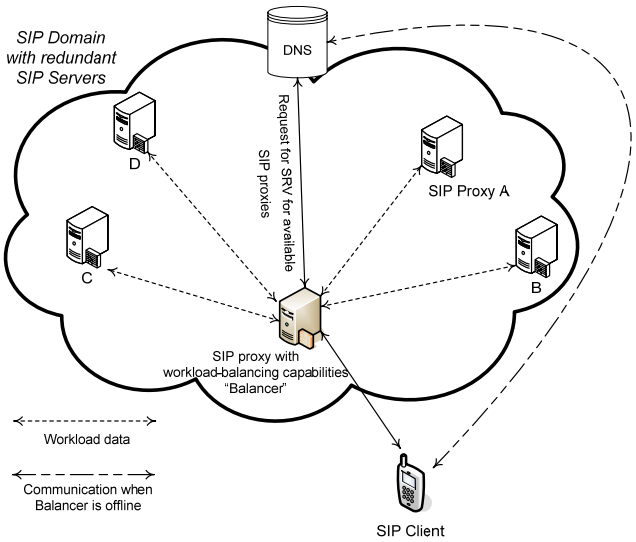

Figure 2. General SIP load balancing architecture

However, having in mind that until now most SIP clients do not support DNS direct transactions, another solution for them is to communicate directly with another available SIP proxy in the same domain. The IP addresses of the LB and the backup SIP proxies can be pre-configured in the SIP client device. As a result, the IP address of the most appropriate SIP proxy is selected by the LB and while the initial message (e.g. INVITE) goes through the LB the subsequent messages for the same session go directly to the selected by the LB SIP proxy.

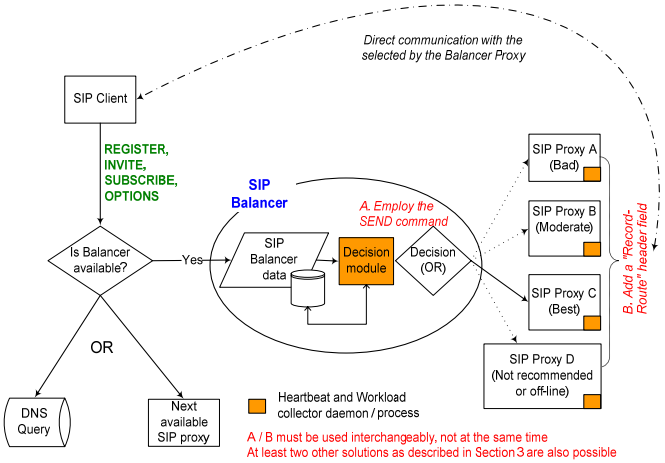

Figure 3. LB Decision Flowchart

To be more precise, the only SIP message types that need to pass through the LB entity are: REGISTER, INVITE, SUBSCRIBE and OPTIONS. Naturally, in case the LB is implemented as another SIP proxy e.g. the open source Express Router (SER) (www.iptel.org/ser/), it will normally insert its own VIA header in the incoming SIP message prior to forwarding it to the corresponding SIP proxy. This standard (for SIP proxies) VIA-adding procedure do have undesired implications as all the corresponding responses referred to the same session will pass through the balancer entity.

To resolve this problem have proposed the following solutions, while the exchanged messages between all the involved entities are depicted in Figure 4:

- One possible solution, is to manage to somehow have the LB's VIA header not inserted at all, inducing this way all the consequent messages to go directly to the home SIP proxy in

charge. For example, this can be done by utilizing the routing engine of the corresponding proxy. In the case of SER, it would be possible to use the SEND command as described in the SER's developers' guide [10].

- A second alternative is to modify the proxy core source code to force it to ignore the VIA-received header added by the LB. However, this solution is proxy dependent or implementation specific and of course not portable.
- The final option is to spoof the source addresses (IP and port) of packets (e.g. INVITE messages) which are forwarded by the LB so that proper routing takes place. According to this scenario we set the IP address and port to the address and port from which the packet arrived to the LB. By employing this solution the LB is more transparent and we don't need to do any changes in the proxy's source code.

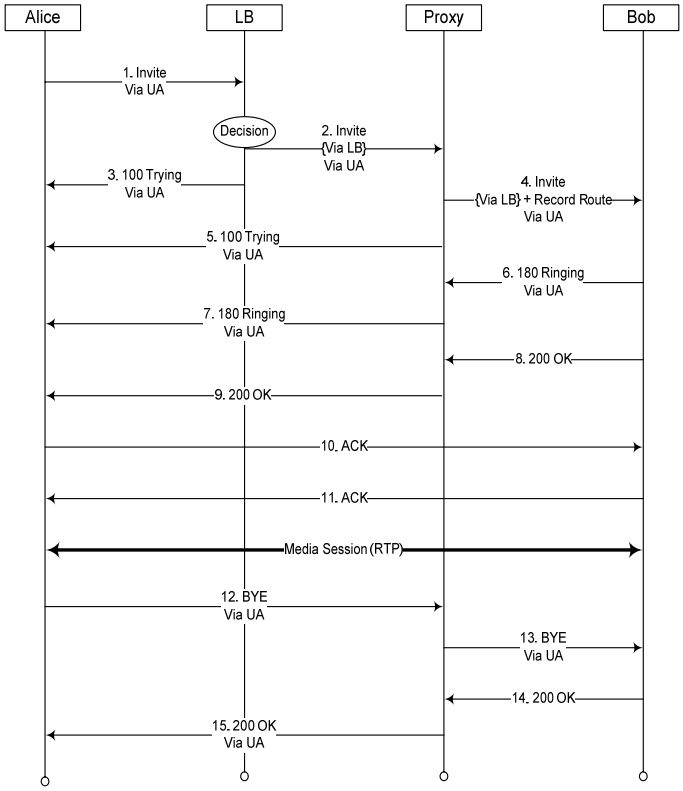

Figure 4. Call flow between all the involved entities (The Via in brackets means optionally (depends on the solution selected, see below)

One other transparency problem that emerged with the utilization of the LB is that the subsequent requests belonging to the same dialog must not pass through the LB. For this reason it is suggested the SIP proxies in the home network, to introduce the Record-Route header field [9]. By doing this, the clients would then send follow-up requests regarding that session to the home SIP proxy assigned by the balancer in the first place. This should then not forward them to the request-URI (the LB) but process them by itself. In addition, this solution even has the advantage that new calls are load balanced because the route set is valid only for one session.

As already mentioned earlier, an orthogonal to the proposed balancing method, as in AAlarm algorithm, is to enable the SIP home proxies to dynamically inform at any time the LB entity when some workload parameters (e.g. concurrent number of SIP transactions) exceed a specified threshold. For example, consider the case where suddenly one or more SIP proxies are under a DoS/DDoS attack. In fact, this capability requires the installation of an appropriate daemon or module in each SIP proxy to monitor the corresponding workload metrics. In case that all available SIP servers are overloaded, it is selected the one with the least current overload. Therefore, it is expected that the SIP home network proxies are selected based to the rate of availability as following:

To be more precise, assume that there exist n operational SIP servers in the domain, with availabilities A_1, A_2, A_3,…A_n, where ($0 \le A_i \le A_{max}$) and Amax is the upper bound of the "Availability" parameter. Under this situation, the i-th SIP server is chosen by the probability $\frac{A_i}{\sum_{j=1}^{n} A_j}$. This situation is depicted in Figure 5. Considering also the worst and of course rare case when all SIP proxy servers are not desirable or down, the LB can choose one SIP proxy randomly and simultaneously display a message to the operator's console).

Summarizing the above paragraphs, the SNOCER solution is DNS independent, so it does not require any modification or extension to existing DNS records or mechanisms nor to the core source code of the employed SIP proxy to act as LB. The only actual requisitions are:

(a) The addition of the LB independent machine implemented as an existing SIP proxy (e.g. SER). The SIP server is only required to support DNS-SRV records but this functionality is already mandatory by the SIP standard (RFC 3261) [9]. Moreover, the employment of a standard SIP server to serve as the LB means that there is no need to develop from the scratch a lot of new software. Only the decision-and-forward engine and the communication modules with the proxies have to be implemented. Note, that in case of a large SIP network, including many SIP proxies, we can realize a LB solution consisting of several geographically distributed SIP proxy clusters controlled by equal number of LBs.

(b) A daemon or process running in each SIP home network proxy sending heartbeats, collecting workload metrics and handling the incoming queries from the LB and optionally,

(c) Inform in an asynchronous fashion the LB when a certain threshold is violated.

(d) All SIP home network proxies use either one shared or more (mirrored) databases.

In our opinion the most compatible with the RFC 3261 of heardbeating functionality is the utilization of *SIP-Specific Event Notification* [11] due to the fact that, it has been designed as a framework by which SIP components can request notification from remote nodes indicating that certain events have been occurred. In this case the LB forks and send a SUBSCRIBE request to all the available SIP proxies. Consequently proxies responds to SUBSCRIBE request with NOTIFY message. LB needs to refresh subscriptions on a periodic basis using a new SUBSCRIBE message on the same dialog as defined in RFC 3261. When a certain SIP proxy does not send NOTIFY messages during a pre-set time period then it is considered unavailable. The body of NOTIFY messages can carry all the necessary information like CPU load, memory consumption or even introduce new headers like WORKLOAD_INFO. Such a header can have the following general structure: WORKLOAD_INFO = {CPU_USAGE, MEMORY_CONSUMPTION, etc}. However, such information must be considered vulnerable to malicious modifications it is suggested to utilize S/MIME to protect either the header or the body from malicious modifications. Moreover, the exchanged messages (see Figure 6) are lightweight having

minimum impact on home SIP proxy servers' performance and network bandwidth. In case the SIP proxy initiates the communication only the last two arrows are present.

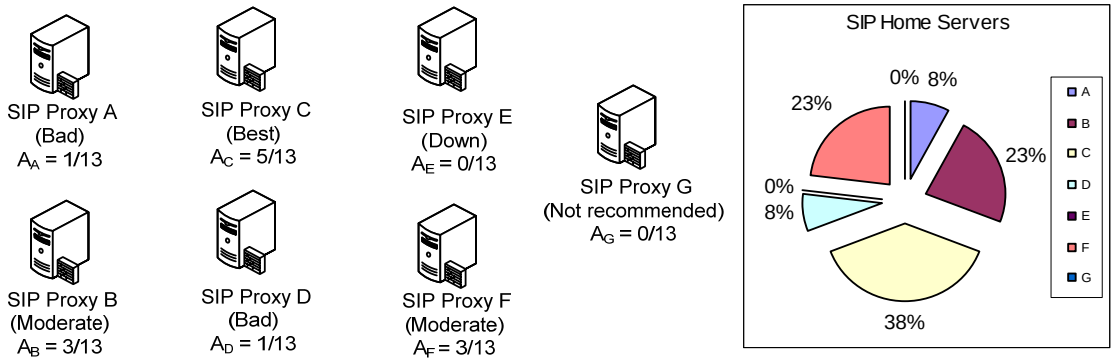

Figure 5. Example of selection rate

In addition, for security reasons, the exchanged messages between the LB and the SIP proxies can be signed using symmetric or asymmetric key technology to protect against replay attacks, tampering and man-in-the-middle attacks. For instance, messages can be protected using the de-facto TLS protocol [12]. Last but not least, to increase LB availability it is also possible to have one or more backup mirrored balancers to defend against possible DoS or DDoS attacks, physical or human disasters, etc. The most practical solution to this issue enables the backup LB to take over the IP address of main LB in the case of main LB's failure. Additionally, we must utilize some type of heart beating between the main and the backup(s) LB to ensure the latter is alive. This can be easily ensured by using standard HA built into e.g. the Linux operating system (http://linux-ha.org).

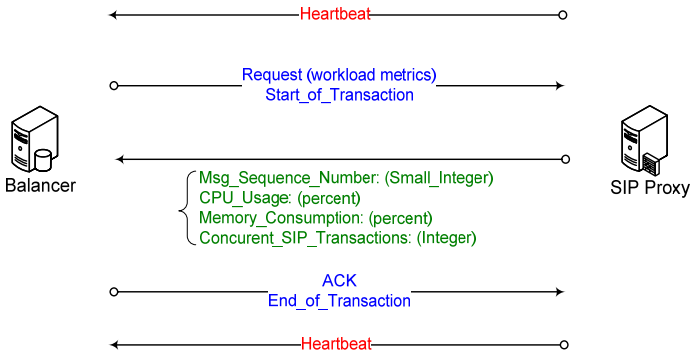

Figure 6. Indicative communication protocol between the Balancer and SIP proxies

We should also mention that user location database is another important issue that is related with the balancing scheme. There are two alternatives. First, to use a shared by all proxies database and secondly to have more databases that are replicated real-time. As the first solution introduces a sensitive single point of failure we intent to select the second one. However, this issue remains out of the scope of this paper.

4. CONCLUSIONS

As VoIP deployments continue to increase and become adopted by more and more commercial organisations, reliability and availability issues turn out to be increasingly important. In this context, balancing the load of SIP transactions raises as a major factor in terms of high availability, redundancy and QoS. Despite the different balancing approaches that have been proposed and developed for Web applications, until now, no SIP-oriented complete balancing solution has emerged.

In this paper we describe a complete, lightweight SIP load balancing scheme which is currently under development in the context of EC SNOCER project. Several implementation issues were analysed including architecture, components, interactions, etc, showing that the anticipated balancing method is practical and above all easy to implement. We also touch upon some complementary questions like failover and security that we would like to continue investigating. Currently, we are planning to thoroughly evaluate the projected solution, which is almost finished, in terms of robustness and performance. As future work, we would like to expand this study considering clusters of SIP proxies controlled by different LBs. Furthermore, considering that the LB is a sensitive and tempting for the attackers' network entity, it is of our interest to systematically probe the security parameters e.g. threats that may be compromise its smooth operation.

5. ACKNOWLEDGMENTS

This work was conducted with the support of the EC under the 2005 project COOP-005892 - SNOCER

The authors would like to thank the FhG FOKUS (DE) and the Nextsoft (CZ) SNOCER partners for their contribution to this work

6. REFERENCES

[1] T. Brisco, *DNS Support for Load Balancing*, RFC 1794, April 1995.

[2] Vovida.org, *Load Balancer Proxy Readme*, Version 1.0.0, July 2002 (www.vovida.org).

[3] M. Colajanni, P. S. Yu & D. M. Dias, Analysis of task assignment policies in scalable distributed Web-server systems, *IEEE Transactions on Parallel and Distributed Systems*, 9(6), June 1998.

[4] M. Colajanni, P.S. Yu, D.M. Dias, Scheduling algorithms for distributed Web servers, *17th IEEE International Conference on Distributed Computing Systems (ICDCS '97)*, 1997.

[5] R. J. Schemers III, Ibnamed: A load balancing name server in Perl, *in LISA '95 conference*, Stanford University, Sept. 1995.

[6] T. Shimokawa, N. Yoshida & K. Ushijima, *DNS-based Mechanism for Policy-added Server Selection*, http://www.tenbin.org.

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

[8] T. Dagiuklas, D. Geneiatakis, G. Kambourakis, D. Sisalem, S. Ehlert, J. Fiedler, J. Markl, M. Rokos, O. Botron, J. Rodriguez & J. Liu, *General Reliability and Security Framework for VoIP Infrastructures*, http://www.snocer.org, Aug. 2005.

[9] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Spark, M. Handley & E. Schooler, *Session Initiation Protocol*, RFC 3261, June 2002.

[10] Janak, J., Kuthan, J., Iancu, B., *SIP Express Router v0.8.8*, Developer's Guide, http://www.iptel.org.

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

[12] T. Dierks & C. Allen, *The TLS Protocol Version 1.0*, RFC 2246, Jan. 19