國立交通大學

網路工程研究所

碩 士 論 文

高可用性 SIP 網路中無單點失效的
整合代理/註冊伺服器之設計與實現

Integrated Proxy/Registrar Servers without SPOF
for High Availability SIP Networks

研 究 生：徐偉原

指導教授：王國禎　教授

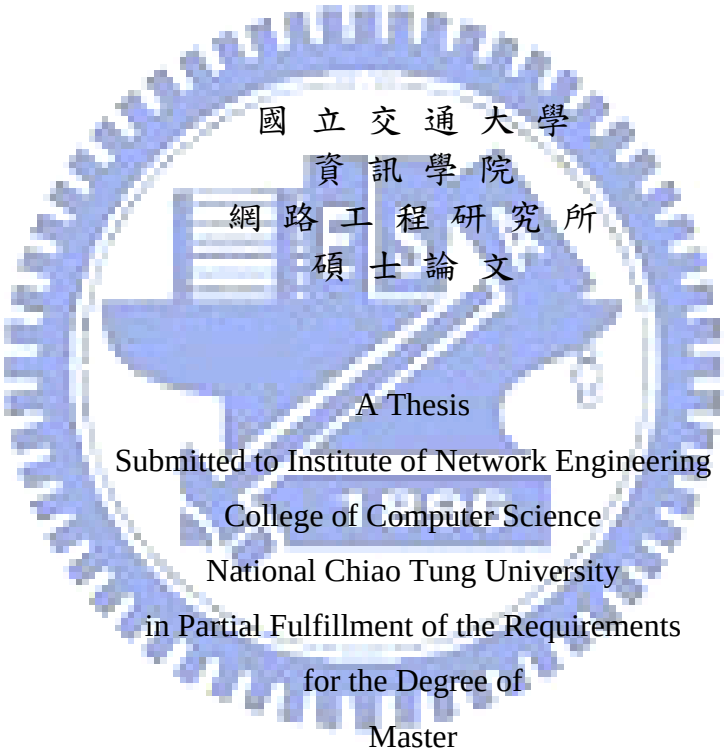
中 華 民 國　九 十 七　年 六 月

高可用性 SIP 網路中無單點失效的整合代理/註冊伺服器之設計與實現

Integrated Proxy/Registrar Servers without SPOF for High Availability SIP Networks

研 究 生：徐偉原　　Student：Wei-Yuan Hsu

指導教授：王國禎　　Advisor：Kuochen Wang



國 立 交 通 大 學

資 訊 學 院

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

高可用性 SIP 網路中無單點失效的整合代理/註冊伺服器之設計與實現

學生：徐偉原　　指導教授：王國禎 博士

國立交通大學 資訊學院 網路工程研究所

摘 要

SIP 目前被應用在許多網路服務，如網路電話或 IP 多媒體子系統服務。在一個高可用性的 SIP 網路中，SIP 代理伺服器和 SIP 註冊器都是相當重要的元件。在這篇論文中，我們設計並且實作了一個沒有單點失效的整合 SIP 代理/註冊伺服器(IPRS)，以建構一個高可用性的 SIP 網路。我們將使用者資訊存在每台 SIP 代理/註冊伺服器的記憶體中，而不是存在一個獨立的資料庫裡。IPRS 提出的 ”$n + m$” 整合 SIP 代理/註冊伺服器是由 n 個使用中的 SIP 代理/註冊伺服器和 m 個備援的 SIP 代理/註冊伺服器所組成。每個使用中的 SIP 代理/註冊伺服器可以經由 OpenAIS 的服務換手到備援的 SIP

代理/註冊伺服器。實作測量的結果顯示，在呼叫頻率為每秒四百個連線時，IPRS 因為使用中的 SIP 代理/註冊伺服器失效而產生的失效呼叫數目比 PPDR 減少了百分之五十六。

關鍵詞：呼叫建立時間，高可用性，OpenAIS，代理伺服器，單點失效，SIP。

Integrated Proxy/Registrar Servers without SPOF for High Availability SIP Networks

Student：Wei-Yuan Hsu **Advisor：Dr. Kuochen Wang**

Department of Computer Science
National Chiao Tung University

Abstract

SIP (session initiation protocol) has been applied to many services, such as VoIP and IP multimedia subsystem (IMS). The SIP proxy server and SIP registrar play an important role in high availability SIP networks. In this thesis, we propose and implement a high availability SIP network architecture that deploys integrated proxy/registrar servers (IPRSs) with no single point of failure problem (SPOF). It stores each user agent's (UA's) information in the memory of each active proxy/registrar server rather in a separate database. The "*n* + *m*" architecture consists of *n* active SIP proxy/registrar servers and *m* standby SIP proxy/registrar servers to process SIP messages from UAs. Each active proxy/registrar server may failover to a standby proxy/registrar using the services of OpenAIS. Experimental results show that when an active SIP proxy/registrar server fails, the number of failed calls in the proposed IPRS is 56% smaller than that of PPDR, a related work with a database involved, under a call rate of 400 calls/per second.

Index Terms — call setup time, high availability, OpenAIS, proxy server, single point of failure, SIP.

Acknowledgements

Many people have helped me with this paper. I deeply appreciate my thesis advisor, Dr. Kuochen Wang, for his intensive advice and instruction. I would like to thank all the classmates in the *Mobile Computing and Broadband Networking Laboratory (MBL)* for their invaluable assistance and suggestions. This work was supported by the National Science Council under Grants NSC94-2219-E-009-023 and NSC96-2219-E-009-012.

Finally, I thank my father and my mother for their endless love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Basic concept of SIP

SIP, an application layer protocol, is defined in RFC3261 [1] by the Internet Engineering Task Force (IETF). SIP is a control protocol to assist in providing telephony services on Internet. It was developed to establish, modify, and terminate a call session that involves multimedia messages, such as video, voice, and instant messages. REGISTER, INVITE, ACK, BYE, CANCEL, and OPTION are six main methods of SIP request messages. REGISTER is for users to register their actual locations or other information to a registrar server. INVITE is for inviting a user to join a call session. ACK is for confirming that a final response corresponding to an INVITE message has been received. BYE is for terminating a call. CANCEL is for canceling a pending request. OPTION is for querying the capability of a server. An SIP response message contains a status code of the corresponding request. Six different classes of status codes are 1XX, 2XX, 3XX, 4XX, 5XX, and 6XX. 1XX represents a provisional response. 2XX represents a success. 3XX represents a redirection response. 4XX represents a request failure. 5XX represents a server failure. 6XX represents a global failure. SIP itself is just a signaling protocol without any ability of handling a multimedia session. Handling the multimedia session is the responsibility of other protocols, such as real-time transport (RTP) protocol or real-time control protocol (RTCP) [6]. We can see that SIP plays an important role in Fig. 1 [2].

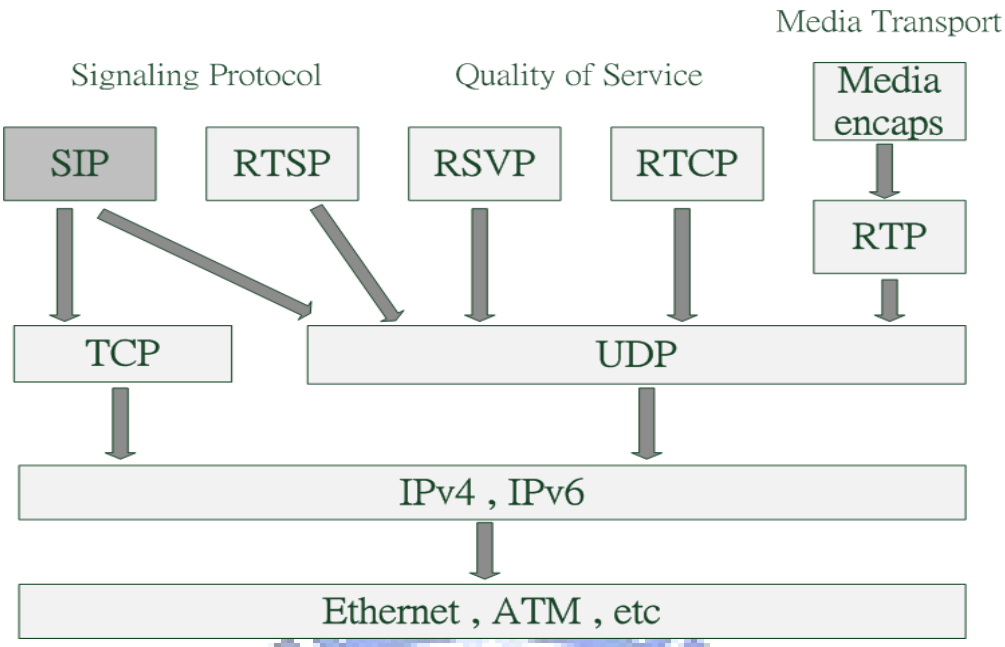


Fig. 1. SIP related protocols [2].

1.2 SIP components

There are four major components in a SIP network [8]:

•**User Agent (UA)** — UAs include user agent clients (UACs) and user agent servers (UASs). UACs are responsible for generating SIP requests to UASs. UASs are responsible for receiving requests sending from UACs and then generating responses to UACs.

•**Proxy Server** — It receives SIP messages from UAs and provides the routing operation of SIP messages. It may simply forward the messages to the appropriate destination. It may also forward the messages after some processing of the messages based on the routing information.

•**Redirect Server** — It receives INVITE messages from UAs and returns another SIP URI or destination address back to UAs. The UAs can use such information to complete a SIP request by sending the request to the appropriate destination.

•**Registrar or Location Server** — It receives SIP REGISTER messages from a UA and stores the UA's information such as the IP address or port. The UA's information can be stored in memory or in a local or remote database depending on the design of system architecture.

1.3 SIP and high availability

Capacity and *redundancy* are two important elements involved when discussing availability of a system [10] [11]. Capacity is the maximal processing ability that the system can achieve. In the aspect of VoIP networks, the capacity is commonly measured in calls per second that the system can handle. Redundancy is the extra capacity that the system can provide only when the event of a component failure occurs. The notation "*m* + *n*" describes the deployed capacity (*m*) and redundancy (*n*) of nodes in the system. For example, if we deploy two active SIP proxy servers and a standby SIP proxy server in a system, we use the notation "2 + 1" to describe the deployed redundancy model. The term "five nines" is another important metric when discussing carrier-grade availability [23]. If we say that the availability of a system achieves "five nines," it means the service availability is 99.999%. The number of nine reflects the degree of availability in a system.

OpenAIS (open application interface specification) [19] is a middleware that implements the Service Availability Forum (SAF) Application Interface Specification (AIS) [20]. OpenAIS AMF (application management framework) service can provide a high availability of an application by clustering redundant servers. AMF can monitor the states and handle the failover events between all servers in its management. OpenAIS CKPT (checkpoint) service provides the information backup operation between active and standby servers.

Chapter 2

Related Work

There are several researches considering the availability of SIP-based VoIP networks. In [5] [15], the main point of the papers is to discuss different methods to failover when some components fail in SIP-based VoIP networks. They also discussed some load balancing methods. The backup server should know when the active server fails. There are two ways to achieve the goal. (1) Active polling: the backup server polls the active server periodically, so the backup server can know the active server fails when the active server did not send the response for the polling from the backup server. (2) Passive heartbeat: the active server sends messages regularly to the backup server, so the backup server can know the active server fails when it does not receive the heartbeat messages from the active server. When the backup server knows the active server fails, there are two ways for the backup server to take over the service. (1) *DNS take over*: this kind of method can be done by prioritized DNS records and DNS server updating. Prioritized DNS records assign different priorities to the symbolic names of the active and backup servers. The client can access the server that owns higher priority and access the other server owns lower priority if there are errors occurred while accessing the one that owns higher priority. The following shows an example of DNS SRV record [16]:

somedomain.com *_sip._udp* *SRV* *0* *0* *active.com*

SRV *1* *0* *backup.com*

In the method of DNS server updating, only the active server is associated with the service, the backup server updates the DNS association when the active server fails. (2) *Local IP address takes over*: active and backup servers are assigned one shared IP address. Normally,

the active server owns the shared IP address by associating the shared IP address with its MAC address. The backup server will reassociate the shared IP address with its MAC address when the active server fails. The authors [5] [15] also mentioned two load balancing schemes. (1) *DNS load balancing*: DNS provides a way to assign a workload to each server with the same priority. The following is an example:

somedomain.com _sip._udp SRV 0 40 sip1.com
SRV 0 30 sip2.com
SRV 0 30 sip3.com

Sip1, sip2, and sip3 servers would receive about 40%, 30%, and 30% of total workload from UACs, respectively. (2) *Load director*: the method uses a centralized load balancer to dispatch SIP messages to the appropriate SIP servers behind the load balancer by some rules. The rules can be round robin (RR) or hash on *call-id* of the messages. But the load balancer becomes the bottleneck of the system.

In [10] [11], Cisco System, Inc. presented the intelligence at the redundant hop. Fig. 2 shows its basic idea. The virtual router redundancy protocol (VRRP) [3] is a standard mechanism for a pair of redundant servers (1+1) in a LAN to negotiate who owns a virtual IP address. When the system starts, one machine is associated with a virtual IP address to be activated by VRRP, and the other one is standby. The standby SIP proxy server will take over the service when the active SIP proxy server fails. But the disadvantage of VRRP is that it only works for 1 + 1 redundancy. Since VRRP itself does not provide any additional function for backup data between the active SIP proxy server and standby SIP proxy server, the UAs' information must be stored at a database in this designed architecture although it is not mentioned in the original paper. However, the architecture faces the single point of failure problem at the database server.

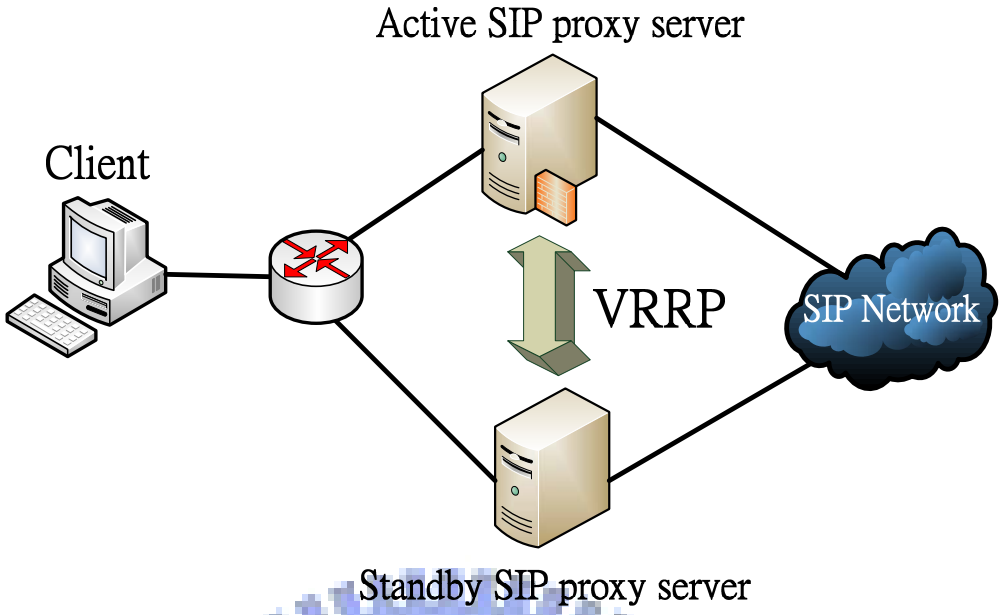


Fig. 2. Architecture that applies VRRP to a pair of SIP proxy servers.

Fig. 3 illustrates the architecture proposed in [7]. The author deployed *n* + *k* SIP dispatchers and *m* + *0* SIP proxy servers. The SIP dispatcher cluster is responsible for dispatching SIP messages to the SIP proxy server cluster. There is one active SIP dispatcher handle a SAP at a time. When an active SIP dispatcher fails, one standby SIP dispatcher will be elected by OpenAIS to take over the SAP. The SIP dispatcher cluster can know the state of the SIP proxy server cluster via the OpenAIS checkpoint service. The SIP dispatcher cluster will stop dispatching the workload to a SIP proxy server when it detects that the SIP proxy server has failed. Although the authors [7] did not mention that there is any backup operation for UAs' information between SIP proxy servers, there must be a database for storing and looking up UAs' information. This architecture suffers the single point of failure problem at the database.

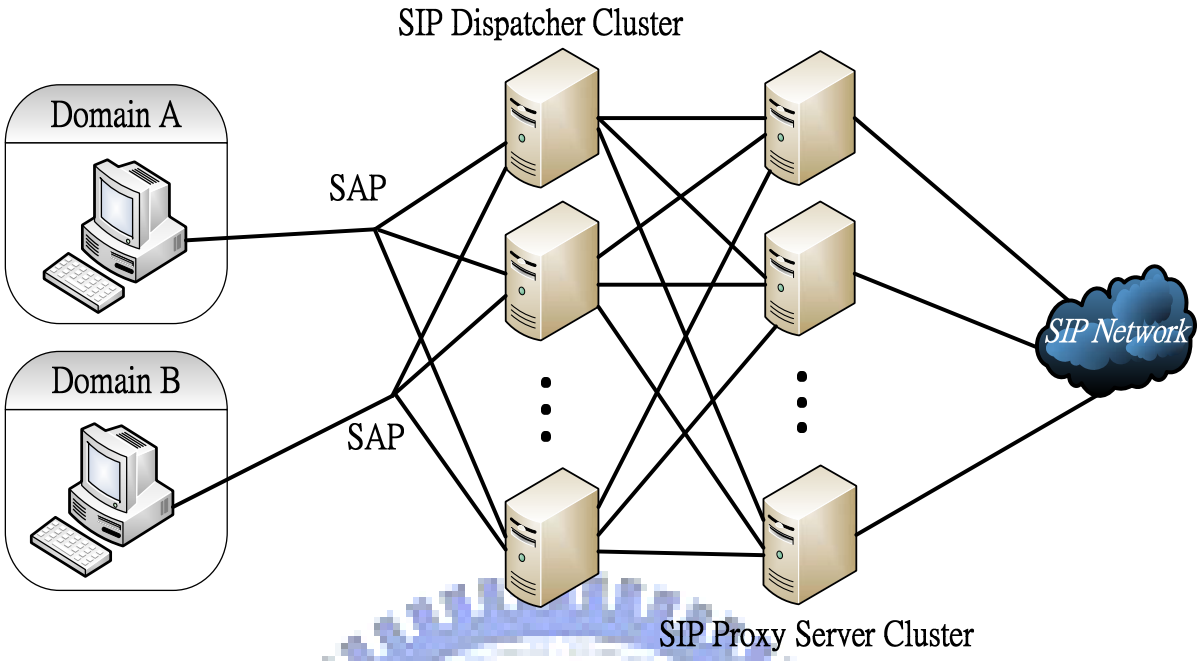


Fig. 3. A high availability SIP network with multiple SIP dispatchers and SIP proxy servers.

In [13], the authors proposed a PPDR architecture which is illustrated in Fig. 4. The Linux virtual server (LVS) [18] is a virtual server in aspect of clients, and there are some real servers inside the virtual server providing load balance and high availability. The architecture deploys two SIP load directors to dispatch the workload to three SIP proxy servers. The MySQL database is for the SIP proxy server cluster to store and lookup the UAs' information. The load balancing scheme is realized via dispatching by Call-ID in the SIP header field. The failover scheme between the active SIP load director and the standby SIP load director is provided by VRRP and Keepalived [4]. The Keepalived on the standby SIP load director can know the state of the active SIP load director by communicating with the Keepalived on the active SIP load director with the VRRP protocol. The standby SIP load director will take over the IP address by using VRRP when the active SIP load director fails. But there is the single point of failure problem at the MySQL database.

In [2], the authors shared the experience using OpenAIS to build a high availability SIP registrar server. The paper focuses on the lessons the authors learned from implementing their work. They deployed 1 + 1 SIP registrars by using the service provided by OpenAIS to allow the standby registrar to take over the service when the active registrar fails. Table. 1 summarizes the above discussion that qualitatively compares these existing approaches with the proposed IPRS approach.

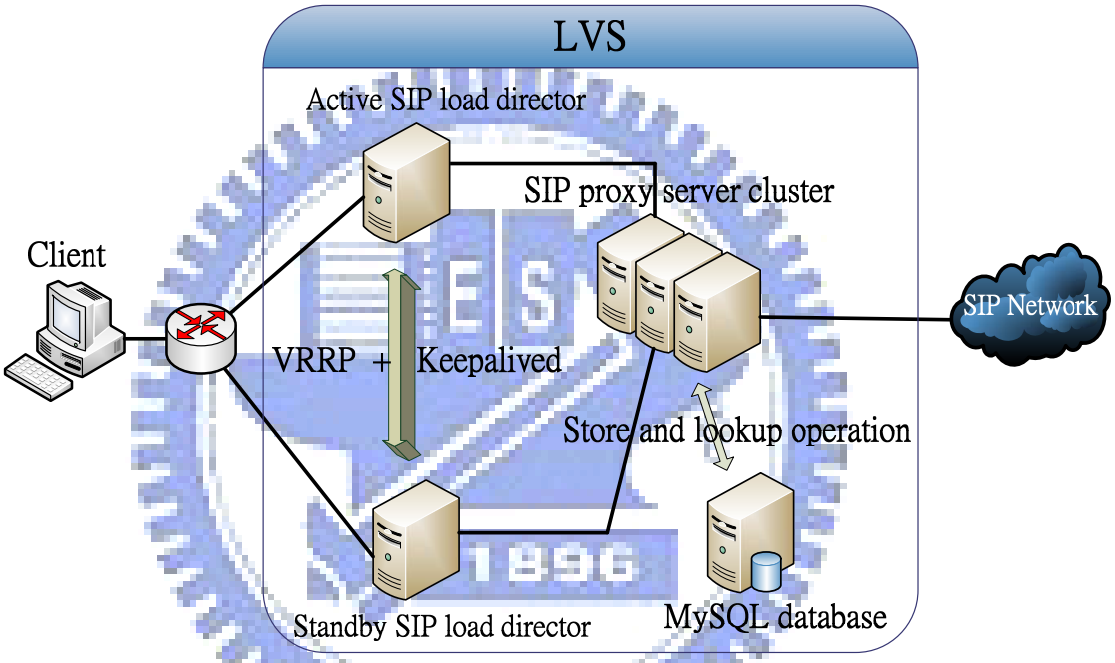


Fig. 4. A high availability SIP network using LVS, Keepalived, and VRRP.

Scheme	Ciscos'[10] [11]	FFF [7]	PPDR [13]	IPRS (proposed)
Redundancy model	Proxy: *1 + 1*	Dispatcher: *m + k* Proxy: *n + 0*	Director: *1 + 1* Proxy: *n + 0*	Proxy: *n + k*
Failover mechanism	VRRP	OpenAIS	LVS + VRRP + Keepalived	OpenAIS
Load balancing support	No	Yes	Yes	Yes
Storage location for UAs' information	A database	A database	A database	Proxy/registrar server
Accessing a database during a call setup procedure	Yes	Yes	Yes	No
Scalability (with DNS SRV)	No	Yes	No	Yes
Failed calls when failover	Medium	Low	Medium to High	Low
Call setup time	Medium	Medium	Medium	Medium to Low

Table. 1. Qualitative comparison of related work for SIP high availability networks.

Chapter 3

Design Approach

3.1 An integrated proxy/registrar server (IPRS)

In this section, we design an architecture that does not suffer the single point of failure problem for SIP high availability networks. Fig. 5 illustrates the IPRS architecture. We deploy $n + m$ SIP proxy/registrar servers (n active SIP proxy/registrar servers and m standby SIP proxy/registrar servers). A SIP proxy/registrar server combines the functions of the SIP proxy server and the SIP registrar. It is responsible for receiving SIP REGISTER messages from UAs and storing the information of UAs in its own *memory* instead of a separate *database*. In addition, a SIP proxy/registrar server also has the function of a SIP proxy server. There are three advantages of storing the information of UAs in the memory instead of a database. First, we avoid the single point of failure problem caused by a database that is used for storing UAs' information. Second, we may cost down by eliminating the need for constructing a database or even achieving high availability of the database. There are many issues to involve considering the availability of a database, but it needs extra cost and effort [9]. Although SIP proxy/registrar servers in IPRS store UAs' information in the memory, not in a database, the SIP proxy/registrar servers in IPRS can still cooperate with a database to provide extra services, which as accounting etc. This also means when the database fails, the extra services based on the database will fail too, but the services provided by IPRS can still work correctly because the UAs' information needed by SIP proxy/registrar servers still exist in the memory. It also means that users can make calls when the database fails. Another merit is that we achieve the goal of high availability for SIP proxy/registrar servers with low cost. Third, when a SIP proxy/registrar server receives a SIP message, it will look up the real location of the

destination user in the request line of the SIP message from its own memory or a database after checking the SIP message is for itself. The SIP proxy/registrar then forwards the SIP messages to the appropriate destination. Because the time required to look up the user's location from memory is shorter than from a local or a remote database, we can have shorter processing time at the SIP proxy/registrar server and thus shorter call setup time compared to that of deploying a proxy/registrar server with a database.

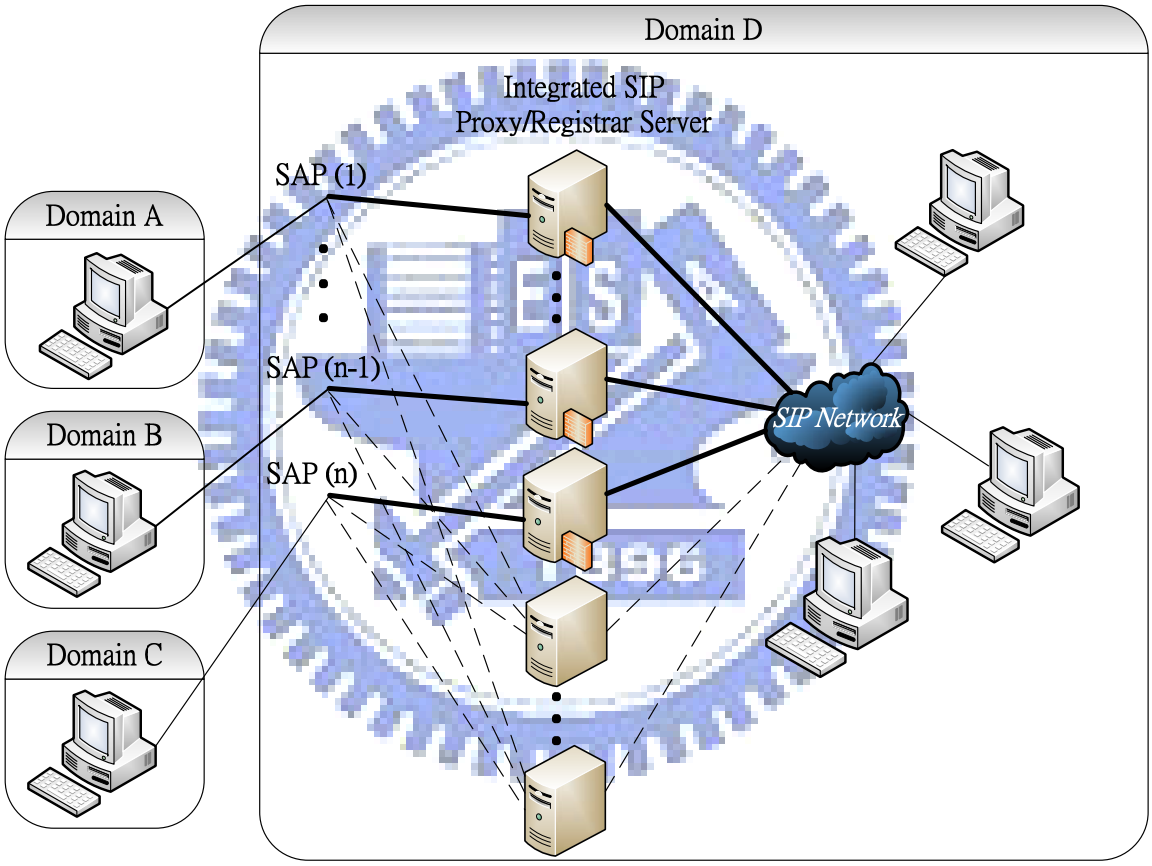


Fig. 5. The proposed IPRS architecture for high availability.

Fig. 5 shows the proposed IPRS architecture for high availability SIP networks. An integrated SIP proxy/registrar server can be constructed using OpenSER [17], an open source for building a SIP dispatcher, SIP proxy server, registrar server, and redirect server as well. A SAP (Service Access Point) is a virtual IP address that represents one active SIP

proxy/registrar server. In Fig. 5, a thick line represents a connection to an active SIP proxy/registrar server. A dotted line represents a connection to a standby SIP proxy/registrar server. Assume domain D deploys our designed architecture. Each UA inside domain D first registers with all active SIP proxy/registrar via SAPs when it joins the SIP network. UAs are allowed to register with multiple SIP registrar servers. UAs outside domain D can access the SAPs through DNS SRV as we mentioned in Section 2. The following shows an example DNS SRV record:

domainD.com _sip._udp SRV 0 30 SAP1

SRV 0 40 SAP2

SRV 0 30 SAP3

In this way, SAP1, SAP2, and SAP3 will receive about 30%, 40%, and 30% of total SIP requests from UAs, respectively. We can provide basic load balancing by this way.

3.2 Failover between active and standby SIP proxy/registrar servers

OpenAIS AMF service running on each active and standby proxy/registrar servers periodically sends control messages as heartbeats to each other with a fixed frequency (100 *ms, default value*), and listens to the heartbeats from others. Therefore, each standby SIP proxy/registrar server can detect whether any active SIP proxy/registrar server fails or not.

When an active proxy/registrar server receives a SIP REGISTER message sending from a UA, it stores the UA's information in memory. The active proxy/registrar server also stores the contents of received SIP REGISTER packets with the corresponding socket information in the checkpoint created by OpenAIS CKPT service. OpenAIS CKPT service is responsible for synchronizing the checkpoint created on the active SIP proxy/registrar server and the checkpoint created on the standby proxy/registrar server. This means the OpenAIS on the

active SIP proxy/registrar server needs to update the checkpoint and notifies the OpenAIS on the standby SIP proxy/registrar server if the checkpoint is updated. The OpenAIS on the standby SIP proxy/registrar server will detect that the checkpoint has been updated by the active SIP proxy/registrar. The standby proxy/registrar server then reads the updated data in the checkpoint that contains the newest data.

When an active proxy/registrar server fails, one of the standby proxy/registrar servers will be chosen by OpenAIS AMF service to take over the original service. The chosen proxy/registrar server will then be notified by OpenAIS that it should claim the virtual IP (SAP) of the failed active proxy/registrar server. The chosen proxy/registrar server will send an ARP message to update the ARP tables of other nodes in the same LAN. Then it replaces the failed active proxy/registrar server and becomes the active proxy/registrar server. The service can be resumed by the chosen standby SIP proxy/registrar server in a short period.

Chapter 4

Evaluation and Discussion

First, we verify the correctness of our proposed architecture. Fig. 6 shows the evaluation environment. X-Lite [12] is a popular open source for VoIP softphone. We installed it on PC1, PC2, and PC3 as our VoIP phones for verification. Table. 2 lists the environment parameter settings. In Fig. 6, PC1, PC2, and PC3 registered with all active SIP proxy/registrar servers when they wanted to join the SIP network. PC1 invited PC2, and PC2 answered that call. PC1 then hanged up. PC1 invited PC3, and PC3 answered that call. PC1 then hanged up again. PC2 and PC3 did the same procedure that called the other two PCs. They could establish a call session to each other successfully. We then let one active SIP proxy/registrar server (PC4) fail. The UA information of PC1, PC2, and PC3 should be available in the standby SIP proxy/registrar server (PC5), and PC5 should change its state as active. PC1, PC2, and PC3 did the same procedure mentioned above again. They could establish a call session to each other successfully. This verified the correctness of the function of the proposed integrated SIP proxy/registrar server using IPRS.

SIPp [14] is an open source traffic generator for SIP protocols. It includes a few basic scenarios for UAC and UAS and it establishes multiple calls with INVITE and BYE methods. We used a default UAC and UAS scenario provided by SIPp for the following test. We installed SIPp on PC1 for evaluation. Fig. 7 and Fig. 8 illustrate the INVITE scenario and REGISTER scenario in SIPp, respectively.

Environment parameter	**value**
CPU (PC1)	Intel Celeron 2.6GHz
Memory (PC1)	1 GB
CPU (PC2)	Intel Pentium 2.0 GHz
Memory (PC2)	512 MB
CPU (PC3)	Intel Pentium Dual 2.0 GHz
Memory (PC3)	1 GB
CPU (PC4, PC6)	Intel Pentium M 1.8GHz
Memory (PC4, PC6)	512MB
CPU (PC5)	Intel Pentium M 1.2GHz
Memory (PC5)	768MB
Call rate in SIPp (for evaluating processing time)	50, 100, 150, 200 calls/sec
Call rate in SIPp (for evaluating number of failed calls)	100, 200, 300, 400 calls/sec
Call limit	10000 calls

Table. 2. Evaluation environment parameter settings.

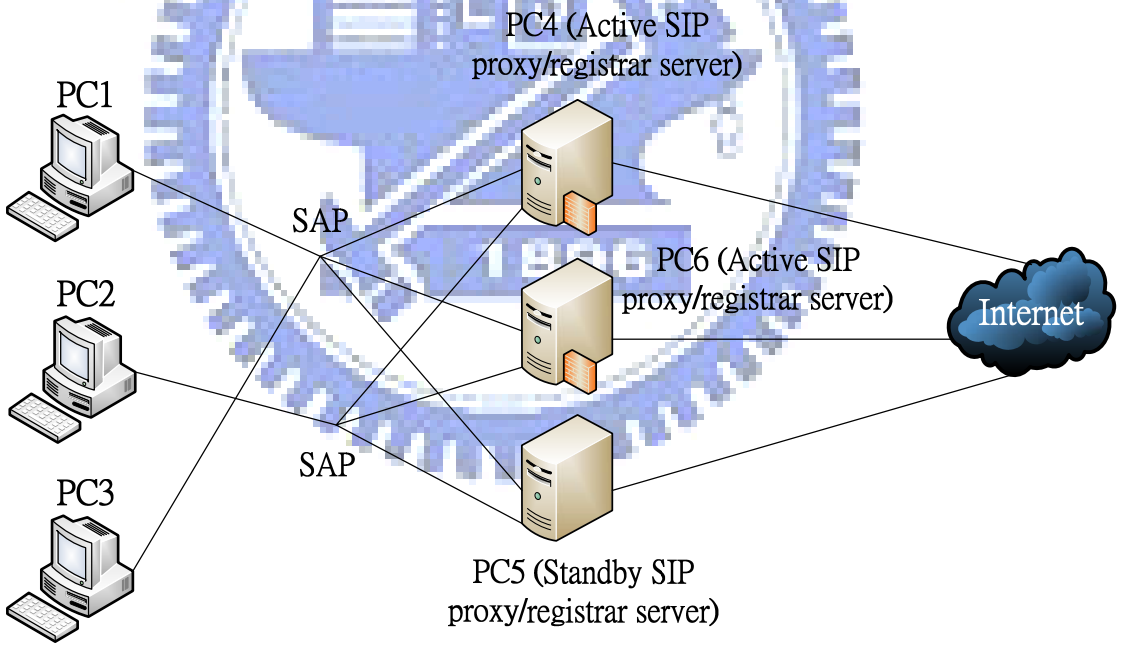


Fig. 6. Evaluation environment.

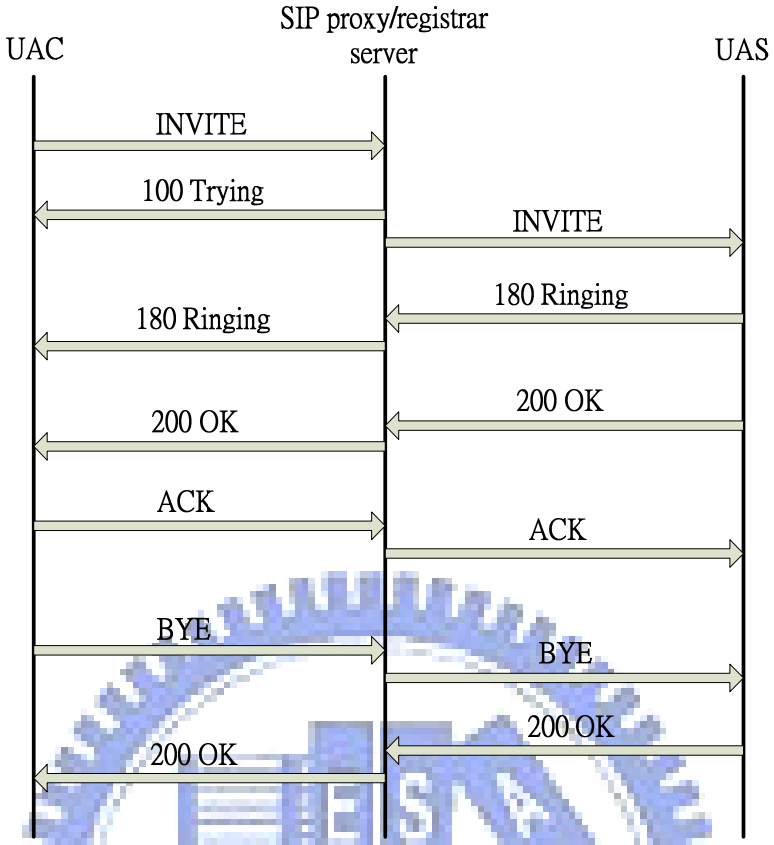


Fig. 7. INVITE scenario in SIPp.

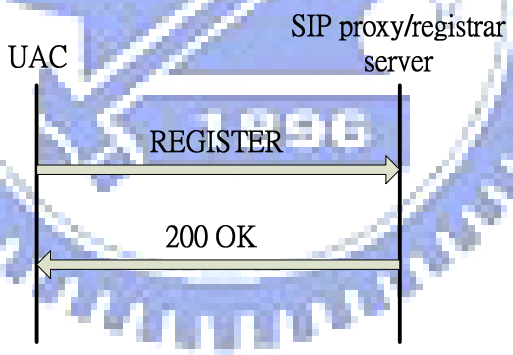


Fig. 8. REGISTER scenario in SIPp.

We turned off the retransmission in SIPp when we tested the failover calls caused by a SIP proxy/registrar server failure so that we could see the failed calls in the results generated by SIPp. We used the INVITE scenario showed in Fig. 7 for observing failed calls, and let PC1 as UAC, and PC3 as UAS. We used SIPp UAS to send SIP REGISTER messages to the active SIP proxy/registrar servers (PC4 and PC6) based on the scenario as shown in Fig. 8.

The user name was generated randomly and the length of each user name is three to four characters. We then used SIPp UAC and UAS to perform the scenario shown in Fig. 7. When the UAC established a call with UAS, we let the active SIP proxy/registrar server (PC4) fail and computed the number of failed calls. We evaluated the number of failed calls caused by the SIP proxy/registrar server failure under various call rates. In Fig. 9, the number of failed calls in IPRS is smaller than that in PPDR [13]. When an active SIP proxy/registrar server failed, the number of failed calls using IPRS is reduced by 29%, 31%, 54%, and 56% compared to that in PPDR under the call rates of 100, 200, 300, and 400 (calls/per second), respectively. The number of failed calls increases both in IPRS and PPDR when the call rate increases. However, the increased degree in IPRS is slower than that in PPDR.

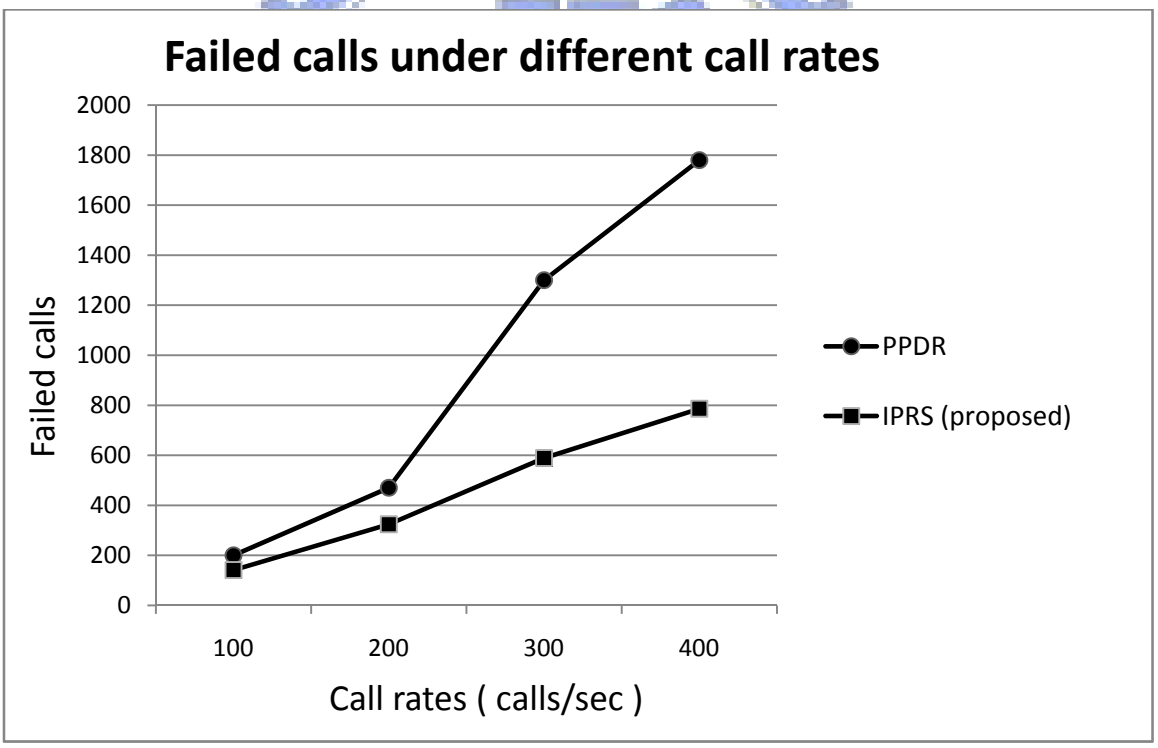


Fig. 9. Number of failed calls caused by a SIP proxy/registrar server failure.

In PPDR, there is a database constructed by MySQL for storing UAs' information. The proposed IPRS stores the UAs' information in the memory of the SIP proxy/registrar server. Next, we want to know the effect on the processing time of a SIP proxy/registrar server if a database is involved in a call setup procedure. We define the processing time of a SIP

proxy/registrar server for a SIP INVITE message as the elapsed time from the SIP proxy/registrar server receiving an INVITE message, looking up the user's location from a memory or a database, and then forwarding it to the appropriate destination. We repeated the above procedure for testing the number of failed calls and computed the processing time of the active SIP proxy/registrar server (PC4). The only difference in the evaluation environment between this test and the last test is that PC6 here did not act as an active SIP proxy/registrar but as a database server constructed by MySQL. The processing time of a SIP proxy/registrar server under call rates of 50, 100, 150, and 200 (calls/second) is shown in Fig. 10. The results show that the processing time of a SIP proxy/registrar server without a database involved is shorter than that with a database involved. The results also show that the processing time of the active SIP proxy/registrar server in IPRS is less affected by the increased call rate than that in PPDR.

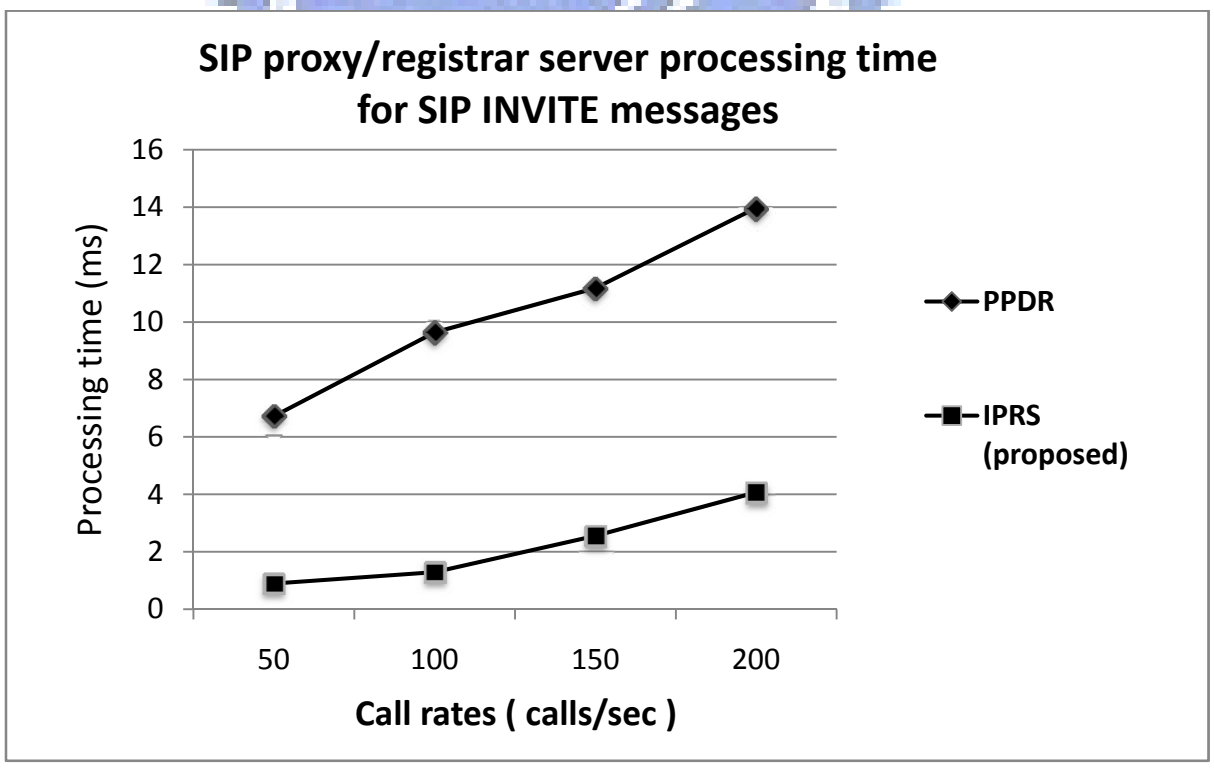


Fig. 10. SIP proxy/registrar server processing time comparison for SIP INVITE messages under various call rates.

We also compare the call setup time between IPRS (without a database involved) and PPDS (with a database involved). Fig. 11 shows that in terms of the call setup time.

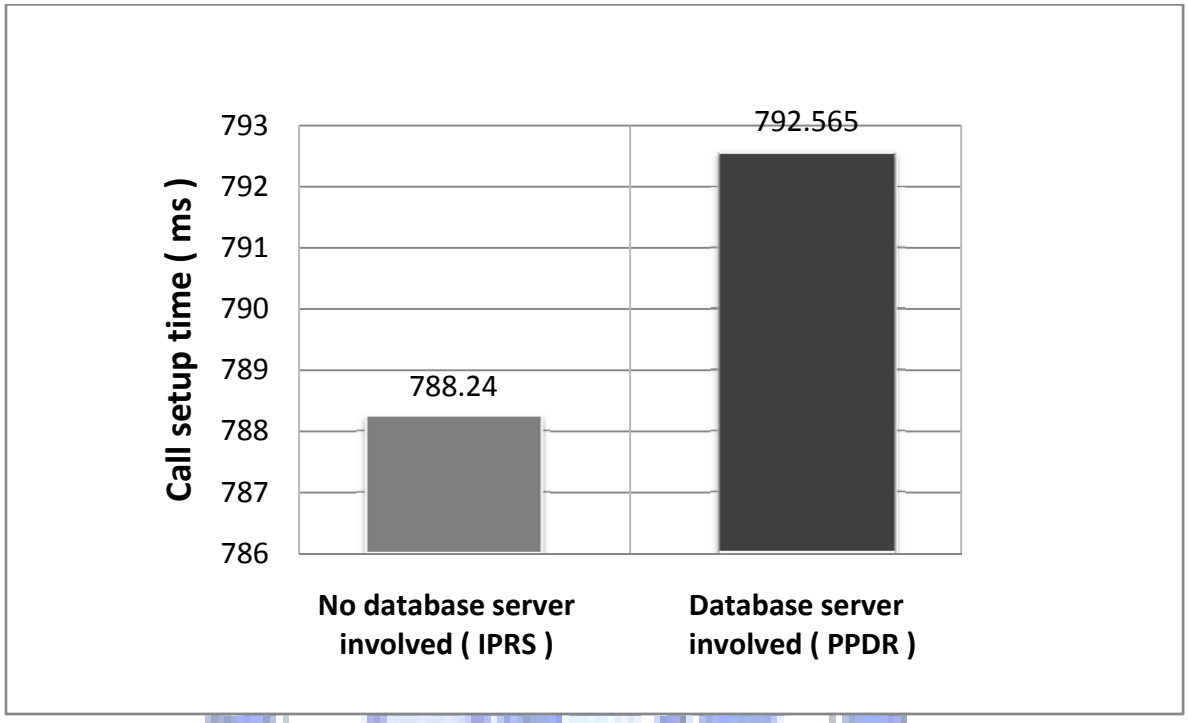


Fig. 11. SIP call setup time comparison.

Finally, we analyze the service availability of IPRS in terms of the number of nines. The availability of a SIP proxy/registar server ($A_{proxy/registrar}$) can be calculated by the mean time to failure of a SIP proxy/registrar server ($MTTF_{proxy/registrar}$) and the mean time to recovery of a SIP proxy/registrar server ($MTTR_{proxy/registrar}$). The availibility of an integrated proxy/registrar server ($A_{integrated\ proxy/registrar}$) can be calculated by the availability of a SIP proxy/registrar server ($A_{proxy/registrar}$) and the number of SIP proxy/registrar servers (N). The availability of the integrated proxy/registrar server is the overall service availability ($A_{service}$). The following are the equations we used [7].

$$A_{proxy/registrar} = MTTF_{proxy/registrar} / (MTTF_{proxy/registrar} + MTTR_{proxy/registrar}) \quad (1)$$

$$A_{integrated\ proxy/registrar} = 1 - (1 - A_{proxy/registrar})^N \quad (2)$$

$$A_{service} = A_{integrated\ proxy/registrar}$$

(3)

Fig. 12. shows experimental results. The service availability is calculated based on various MTTFs (hours) and $MTTR_{proxy/registrar}$ = 72 hours [7][22]. Experimental results show that if we want to have higher availability when the MTTF is lower, we need to deploy more SIP proxy/registrar servers in the IPRS architecture. The result also shows that the smallest number of SIP proxy/registrar servers needed if we want to achieve "five nines" when the MTTF is fixed. For example, when we want to achieve "five nines" under MTTF = 1500 hours, the smallest number of SIP proxy/registrar servers required is four. We can analyze the results in another aspect. We can find out a tolerable MTTF when we deploy a fixed number of SIP proxy/registrar servers in IPRS and want to achieve "five nines." For example, when we want to achieve the "five nines" if three SIP proxy/registrar servers are deployed, the tolerable MTTF is 6000 hours.

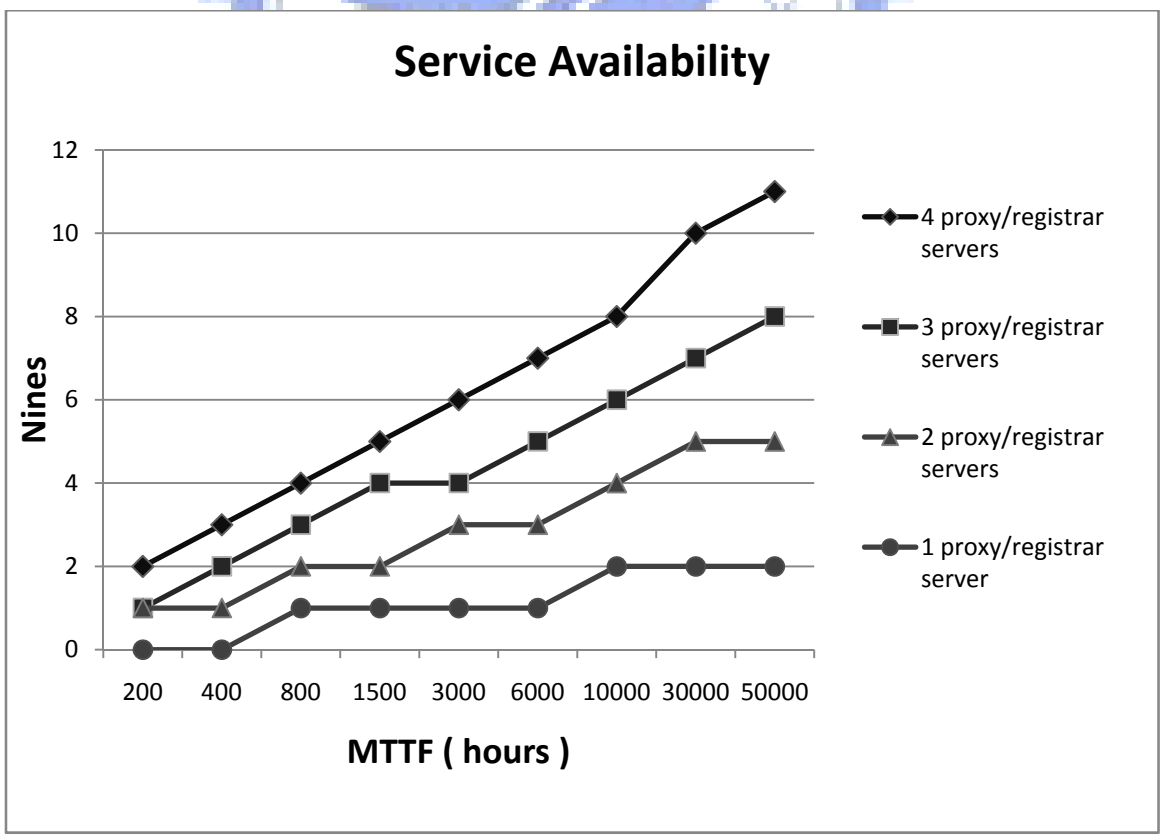


Fig. 12. Service availability under various MTTFs.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The proposed IPRS architecture for high availability SIP networks uses a cluster of integrated SIP proxy/registrar servers without a database involved. It does not suffer the problem of single point of failure in SIP-based VoIP networks. OpenAIS is the middleware that provides the functions of failure detection and failover among SIP proxy/registrar servers in the architecture. In our approach, no modification of SIP protocols is needed for realizing our architecture. Experimental results have shown that when an active SIP proxy/registrar server fails, the number of failed calls in IPRS can be reduced by 29%, 31%, 54%, and 56% compared to that in PPDR under the call rates of 100, 200, 300, and 400 (calls/per second), respectively. In addition, we have also resolved the single point of failure problem by reallocating UAs' registration information from a database to a server's memory.

5.2 Future work

The load balancing scheme in our architecture is DNS-based. There may be some other ways to integrate a load balancing scheme into our architecture to make the whole architecture independent of DNS.

Bibliography

[1] J. Rosenberg, et al., SIP: Session Initiation Protocol, *RFC 3261*, Internet Engineering Task Force, June 2002.

[2] A. Kamalvanshi and T. Jokiaho, "Using OpenAIS for Building Highly Available Session Initiation Protocol (SIP) Registrar", Nokia Corporation, SA Forum, in *Proc. of the Third Annual International Service Availability Symposium*, May 2006, pp. 217-228.

[3] R. Hinden, et al., VRRP: Virtual Router Redundancy Protocol, *RFC 3768*, Internet Engineering Task Force, April 2004.

[4] "Keepalived," Available: http://www.keepalived.org/.

[5] K. Singh and H. Schulzrinne, "Failover, Load Sharing and Server Architecture in SIP Telephony," *Computer Communication*, Volume 30, Issue 5, pp. 927-942, March 2007.

[6] H. Schulzrinne, et al., RTP: A Transport Protocol for Real-Time Applications, *RFC 3550*, Internet Engineering Task Force, July 2003.

[7] W.-M. Wu, K. Wang, R.-H. Jan, and C.-Y. Huang, "A Fast Failure Detection and Failover Scheme for SIP High Availability Networks", in *Proc. of the 13th Pacific Rim International Symposium on Dependable Computing*, Dec. 2007.

[8] "SIP Server Technical Overview," Available: http://www.radvision.com/NR/rdonlyres/0AFA30DF-DAD6-461D-943C-ED33F3E7ABD8/0/SIPServerTechnicalOverviewWhitepaper.pdf.

[9] "A Guide to Database High Availability," Available: http://www.greatlinux.com.cn/userfiles/MySQL_WhitePaper_Guide_to_DB_HighAvailability.pdf.

[10] Cisco Inc., "Overview of High Availability in SIP-Based Voice Networks,"

Available: http://www.cisco.com/en/US/docs/ios/12_3/vvf_c/cisco_ios_sip_high_availability_application_guide/hachap1.pdf.

[11] Cisco Inc., "High-Availability Solutions for SIP Enabled Voice-over-IP Networks," Available: http://www.sipcenter.com/sip.nsf/html/WEBB5YP4SU/$FILE/cisco_high_availability.pdf.

[12] Counterpath Corporation, "X-Lite", Available: http://www.counterpath.com/.

[13] V. Matic, I. Franicevic, and D. Sekalec, "Parallel SIP Proxy Servers Using Direct Routing Approach," *Software in Telecommunications and Computer Networks*, Sept. 2006, pp. 218-222.

[14] "SIPp - A Free Open Source Test Tool / Traffic Generator for the SIP Protocol," Available: http://sipp.sourceforge.net/.

[15] "High Availability and Scalability of VoIP Infrastructures," Available: http://www.fokus.fraunhofer.de/ngni/topics/paper/Whitepaper_SIP_Scalability.pdf.

[16] A. Gulbrandsen et al., "A DNS RR for Specifying the Location of Services (DNS SRV)," RFC 2782, Internet Engineering Task Force, February 2000.

[17] "OpenSER," Available: http://www.openser.org/.

[18] Linux Virtual Server (LVS), Available: http://www.linuxvirtualserver.org/.

[19] "OpenAIS," Available: http://openais.org/.

[20] Service Availability Forum, Available: http://www.saf.ud.it/.

[21] "MySQL - The World's Most Popular Open Source Database," Available: http://www.mysql.com/.

[22] K. Uhlemann et al., "JOSHUA: Symmetric Active/active Replication for Highly Available HPC Job and Resource Management," in *Proc. IEEE International Conference on Cluster Computing*, Sept. 2006, pp. 1-10.

[23] C. R. Johnson et al., “VoIP Reliability: A Service Provider’s Perspective,” *IEEE Communications Magazine*, July, 2004, pp. 48-54.