國立交通大學

網路工程研究所

碩 士 論 文

實現 SIP 高可用性網路之快速錯誤偵測與復原機制

A Fast Failure Detection and Failover Scheme for SIP High Availability Networks

研 究 生：吳偉銘

指導教授：王國禎　教授

中 華 民 國 九 十 六 年 六 月

實現 SIP 高可用性網路之快速錯誤偵測與復原機制
A Fast Failure Detection and Failover Scheme for SIP High Availability Networks

研 究 生：吳偉銘　　Student：Wei-Ming Wu

指導教授：王國禎　　Advisor：Kuochen Wang

國 立 交 通 大 學
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 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

實現 SIP 高可用性網路之快速錯誤偵測與復原機制

學生：吳偉銘　　　指導教授：王國禎 博士

國立交通大學網路工程研究所

摘 要

SIP 代理伺服器在 SIP 網路中是一個重要建構單元，它可以組成一個叢集來避免軟體或硬體錯誤所造成的服務中斷。此外搭配一個分派器的 SIP 代理伺服器叢集設計仍然會遭遇到單點失敗的問題。現有的研究曾提出使用雙 SIP 分派器的架構，並且在 IP 接手機制中使用 VRRP (Virtual Router Redundancy Protocol)。在這篇論文中，我們設計且實作了一個可靠的 SIP 網路，它包含了 $n + k$ 個 SIP 分派器 (n 個使用中的分派器 $+ k$ 個備援的分派器) 去控制並監測提供建立 VoIP 與視訊會議連線的 m 個 SIP 代理伺服器所組成的叢集。我們提出一個快速錯誤偵測與復原(FFF)機制。FFF 使用 OpenAIS 作為一個高可用性中繼軟體去實現分派器與代理伺服器的健康偵測與錯誤復原。實作量測結果顯示，FFF 機制比 VRRP 機制減少了百分之

八十之的錯誤復原時間，以及比 OpenSER 機制減少了百分之八十五的錯誤復原時間。

關鍵詞：錯誤復原, 高可用性, 中繼軟體, OpenAIS, OpenSER, SIP,

A Fast Failure Detection and Failover Scheme for SIP High Availability Networks

Student：Wei-Ming Wu **Advisor：Dr. Kuochen Wang**

Department of Computer Science
National Chiao Tung University

Abstract

The SIP proxy server, which is an important building block of a SIP network, can be in a form of a cluster to prevent service unavailability caused by software or hardware failures. The design of a SIP proxy server cluster with a single dispatcher still faces the single-point-of-failure problem. The use of dual SIP dispatchers architecture was proposed and VRRP (Virtual Router Redundancy Protocol) was used as an IP failover mechanism. In this thesis, we have designed and implemented a dependable SIP network that includes n + k SIP dispatchers (n active dispatchers + k backup dispatchers) to control and monitor a cluster of m SIP proxy servers for VoIP and Video conferencing applications. A fast failure detection and failover (FFF) scheme has also been proposed. FFF uses OpenAIS as a high availability middleware to perform health check and failover of dispatchers and proxy servers. Experimental results have shown that the FFF scheme reduces the dispatcher failover time by 80% compared to the VRRP mechanism and also shorten the proxy failover time by 85% compared to the mechanism provided by OpenSER itself.

Index Terms — Failover, high availability, middleware, OpenAIS, OpenSER, SIP.

Acknowledgements

Many people have helped me with this thesis. 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* 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 Mother for their endless love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Session Initiation Protocol (SIP)

VoIP and video conferencing applications are getting popular in resent years. The main internet protocol which was designed for VoIP and video (multimedia) conferencing services is the Session Initiation Protocol (SIP). SIP is a signaling, presence and instant messaging protocol developed to handle multimedia sessions [1]. SIP allows two or more end points to establish, modify, and tear down the multimedia session with each other. The Real-time Transport Protocol (RTP) [2] is often used as the media transport protocol in the multimedia conferencing, and SIP is the related signaling protocol which does not involved the media streaming issue.

A SIP proxy server is a call-control device that provides many services such as routing of SIP messages between SIP user agents [3]. In the SIP session establishment, a SIP user agent, a SIP client, generally sends a request to a SIP proxy server. The SIP proxy server gets information from the database maintained by the SIP registrar first. Then it decides to process the request by itself or routes the request to a SIP user agent or another SIP proxy server. Since it is directly accessed by SIP user agents and provides the above services, the SIP proxy server is considered as an important device in the SIP networks.

1.2 SIP and high availability

The basic idea to prevent service interruption caused by server maintenance or server failures is to setup multiple backup servers that have the same capabilities as the original one.

The backup servers should monitor the health condition of the active one, and handle requests when the active one is unavailable. The notation "n + k" is commonly used to describe the capacity (n) and redundancy (k) of nodes in an availability solution [3]. Capacity means how many nodes are set to serve at the startup and redundancy means how many nodes are standby and prepared to handle requests when required.

In traditional PSTN (public switch telecommunications networks) world, carrier class reliability is associated with 99.999 percent availability, with a term of "five nines" availability. The goal of VoIP is to achieve the same level of network reliability as provided by the PSTN infrastructure [4]. The traditional design of SIP high availability uses limited (one or two) dispatchers and multiple SIP proxy servers. Increasing the number of SIP proxy servers will increase the total availability of entire service, but still faces the bottleneck of less SIP dispatchers. In this thesis, we designed and inplemented a dependable SIP network that includes $n + k$ SIP dispatchers (n active dispatchers + k backup dispatcher) to control and monitor a cluster of m SIP proxy servers. With this system architecture, we could provide carrier class reliability as well as higher availability. The details will be discussed in chapter 3.

1.3 High availability middleware

The demand for high availability services is based on different business requirements. High availability middleware is suggested to provide business continuity, especially the companies that need to fulfill 99.99% or even five nines availability of services.

OpenAIS (open application interface specification) [5] is an implementation of Service Availability Forums [7] API Specification. The main feature of OpenAIS is cluster management in order to perform high availability. The most important component of OpenAIS is Availability Management Framework (AMF) which is in charge of the health check and redundancy model.

Using OpenAIS to design SIP high availability networks has been proposed in [7]. Instead of using redundant SIP proxy servers, OpenAIS was used as cluster middleware of SIP registrars to prevent service errors caused by SIP registrar failure. It used OpenAIS to monitor the health of two registrars and handle the failover between registrars. This design is 1 + 1 SIP registrars architecture because there is only one active server and one backup server.

Chapter 2

Related work

The basic idea to achieve high availability of SIP networks is adding backup nodes for service nodes like SIP proxy servers. Depending on where the failure is detected and who does the failover, there are various design choices of system architecture: client-based failover, DNS-based failover, database failover, and failover using IP address takeover [8]. In the following, we classify different design choices from system architecture point of view.

2.1 Client or DNS based failover

Failover between SIP proxy servers can be done in the client side and without extra support from the server side. SIP user agents can set a single or multiple backup SIP proxy server entries to deliver the SIP request if the primary SIP proxy server fails to response. In this design, all SIP proxy servers can work independently and no need to communicate with each other. This is easy for clients to setup if there is only one backup server but not convenient for them to input many IP addresses or domain names if there are many backup servers.

By using DNS SRV records [9], clients no longer need to record multiple IP addresses or domain names. A DNS SRV record is commonly configured to point to SIP proxy servers and associate the domain name of each service with an IP address [10]. This design can provide failover between multiple SIP proxy servers. The service provider can set the associated weight and priority of each server in the server list of the DNS SRV record. By adjusting the weight and priority settings, the service provider can change the capacity and redundancy behavior of the SIP proxy servers based on their needs.

2.2 Single controller for proxy server cluster

F5's BIG-IP [11] system is an application traffic management solution and it provides high availability and reliability of SIP networks. Fig. 1 illustrates the high availability SIP network architecture using F5's BIG-IP solution. The BIG-IP not only performs as a controller of the SIP proxy server cluster, it also controls SIP media servers and other service nodes like web servers.

The BIG-IP does the advanced health check by sending SIP OPTIONS requests to SIP media and proxy servers. With these health checks, the BIG-IP can route messages away from unstable and unreliable servers and provide a more proactive approach to high availability [11].

The BIG-IP product plays an important part of the SIP network architecture because it acts as a SIP server failover controller and a service entry point of SIP user agents. It prevents the system become unavailable when a single SIP media or proxy server fails. However, the system will still become unavailable when BIG-IP itself failed. That is, the single-point-of-failure problem still exists in this architecture.

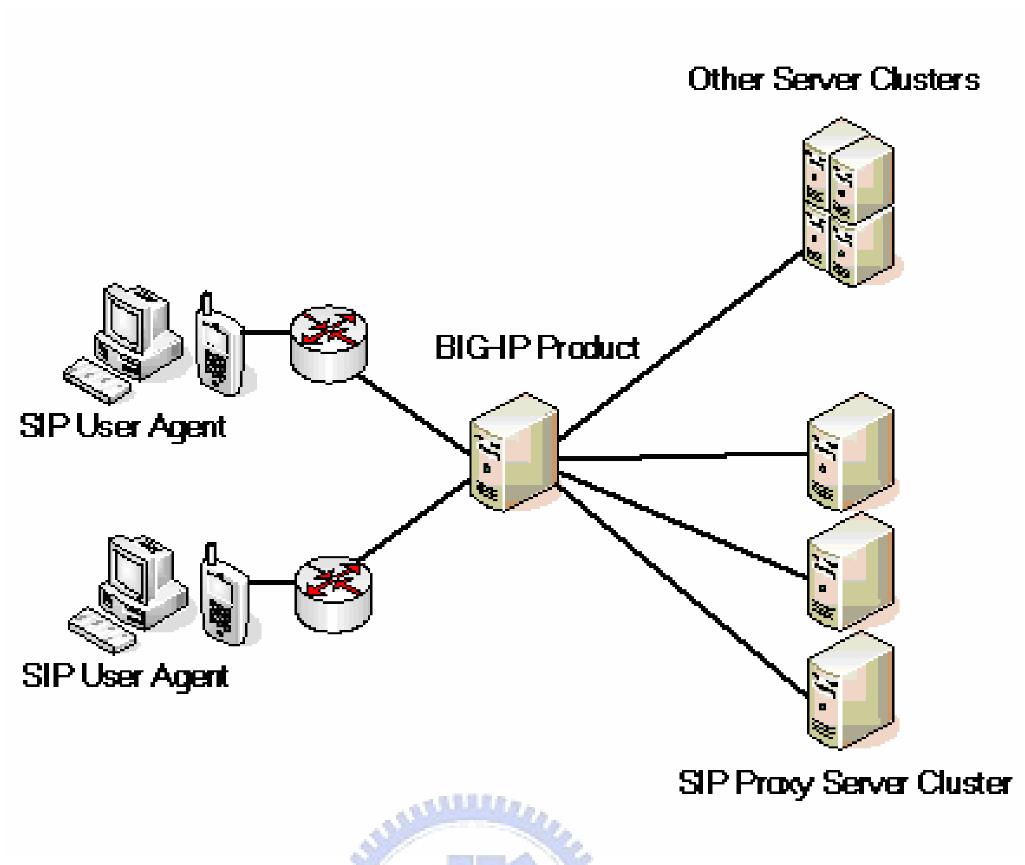

Fig. 1. F5's BIG-IP infrastructure.

OpenSER [12] is an open source software. It performs not only the function of SIP proxy servers and also includes the functions of SIP registrars and SIP redirect servers. OpenSER also has modules to support more functions, such as SNMP and high availability functions. The DISPATCHER module is designed to dispatch user agents' requests to SIP proxy servers. That is, it can connect to SIP proxy servers that form as a cluster, and dispatch the requests coming from SIP user agents to selected SIP proxy servers.

The OpenSER with DISPATCHER module design is like the architecture of BIG-IP. SIP dispatchers which are built by OpenSER take the responsibility of monitoring the health condition of SIP proxy servers. If any of the SIP proxy servers in the cluster fails, the dispatcher would find it out and stop sending SIP requests and responses to it.

2.3 Intelligence in the redundant hop

Cisco IOS SIP High Availability Application [3] provides two different types of redundancy to insure the high availability over SIP proxy servers. In SIP networks, the design of using intelligence at the server side is using redundant hop on SIP servers which can be the SIP proxy servers or SIP load balancers (or called dispatchers). The Virtual Router Redundancy Protocol (VRRP) [13] was used for the redundant control on the both sides (active and standby) of redundant nodes.

VRRP defines a standard procedure that enables multiple redundant servers on a LAN to negotiate ownership of a single virtual IP address. This design is to solve the single-point-of-failure problem of the node on the static route path. VRRP first associates the virtual IP to a server called Master server. When the Master server becomes unavailable, VRRP selects the highest priority server of the other servers to take over the virtual IP and continue serving the incoming request. The clients only have to record the virtual IP maintained by VRRP as a service access point and can be served without interruption when the Master server failed.

In this design, each hop of SIP servers in the SIP networks used VRRP to set as 1 + 1 redundancy. Fig. 2 shows the architecture of 1 + 1 redundant SIP proxy servers. Fig. 3 shows the architecture of 1 + 1 redundant SIP load balancers. If SIP proxy servers adopt $n + k$ instead of 1 + 1 redundancy model, using 1 + 1 redundant SIP load balancers not only enables the load balancing of SIP proxy servers and also can avoids the single-point-of-failure problem when the SIP load balancer fails. The redundant control of SIP load balancers is based on VRRP.

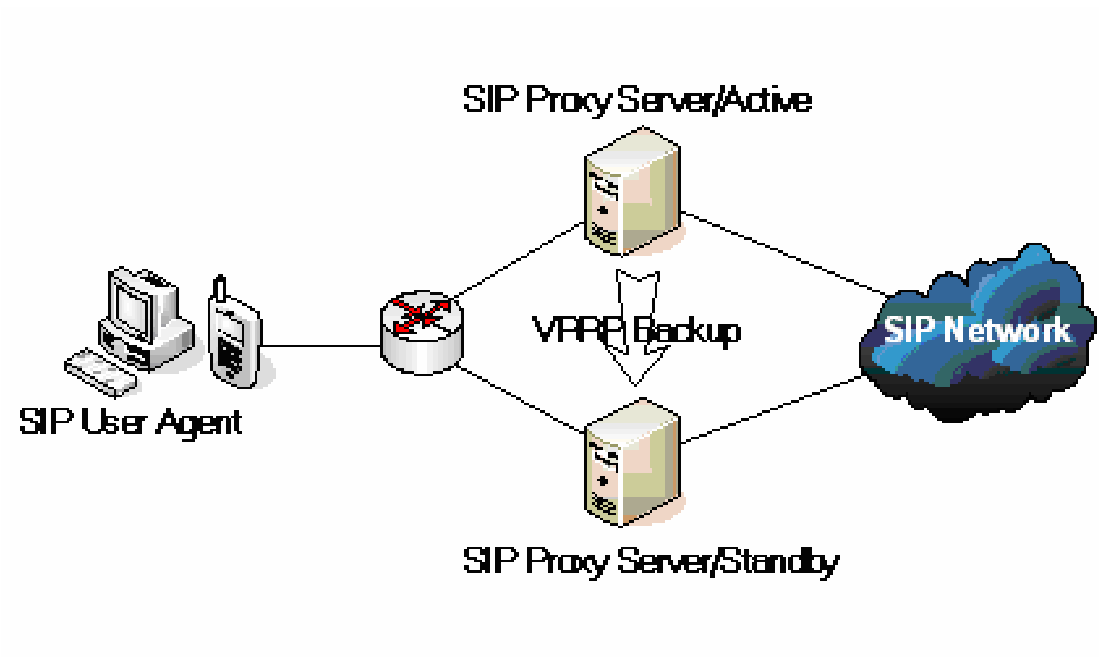

Fig. 2. High availability SIP network with intelligence placed in SIP proxy servers.

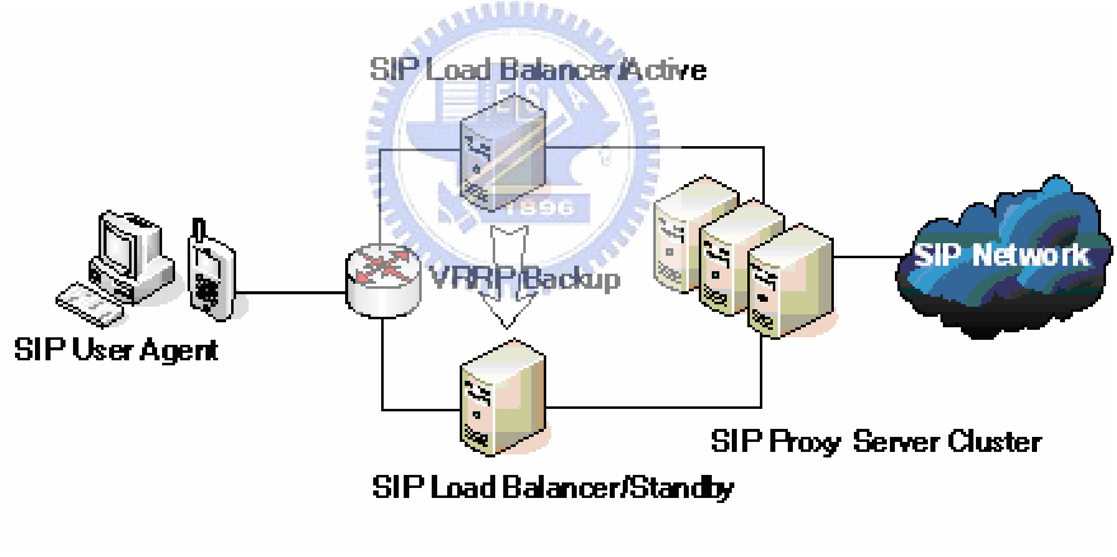

Fig. 3. High availability SIP network with intelligence placed in SIP load balancer.

2.4 Redundant controller for proxy servers cluster

This method not only uses multiple SIP proxy servers but also uses multiple SIP proxy server controllers, which are SIP load balancers, to setup a SIP high availability network. The idea is the same as the design of setting SIP dispatchers as a redundant hop that was mentioned in the last subsection.

The SIP high availability design in [14] used two SIP load balancers and three SIP proxy servers to implement a SIP high availability network. The main design approach is based on the Linux virtual system (LVS) [15] and it uses the direct routing approach as the routing mechanism. SIP dispatchers and SIP proxy servers are considered as a single server since they all use the same virtual IP as the service entry point and outgoing IP. The SIP requests sent by the SIP user agents will be processed by the SIP dispatchers first and the SIP dispatcher would select a SIP proxy server from the maintained list to forward. The response messages of SIP requests generated by the SIP proxy server will use the virtual IP to directly deliver to the SIP user agents without going back to the dispatcher.

Keepalived [16] is a demon designed for providing high availability in LVS environments. It implements a framework to monitor health condition of a LVS server pool and also implements an independent VRRP stack to handle failover of LVS directors (load balancers). The master load balancer in this design uses Keepalived to communicate with the Keepalived on the slave load balancer and handles the failover issue by the VRRP protocol implemented in Keepalived. Fig. 4 shows the system design architecture.

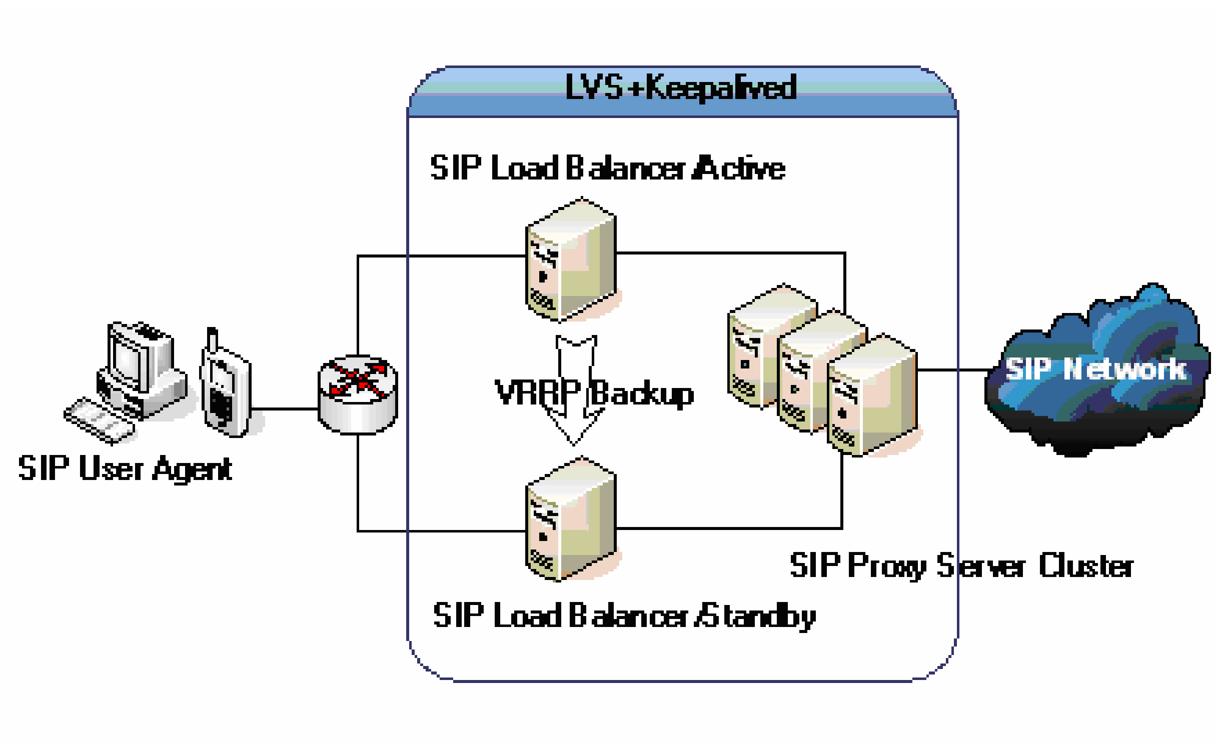

Fig. 4. High availability SIP network using LVS + Keepalived.

2.5 Qualitative comparison

We compare the design of DNS SRV [3], redundant hop of proxy server design [3], BIG-IP [11], redundant hop of load balancer design [3], parallel SIP proxy servers using direct routing approach [14], and the proposed FFF qualitatively, as shown in Table 1. The following eight metrics are considered: Configuration (in the terms of "dispatchers * proxy servers"), failover mechanism, load balancing support, redundancy model (*n* (active) + *k* (standby), client and/or DNS involved, failover mechanism (failover time), single-point-of-failure problem, scalability (with DNS SRV). We have the following observations:

Firstly, the configuration of each design is listed, only the proposed FFF uses n * m. Secondly, we found that the designs of redundancy at the server side, such the designs in[3][11][14], all adopt 1+1 redundant SIP servers (dispatchers or proxy servers) by using VRRP. Thirdly, those using the redundant SIP proxy server model, such as the designs in [3] do not have the load balancing support. Fourthly, the redundancy model of each approach is listed and some of them are not involve with the SIP dispatcher, such as the designs in [3] [11]. Fifthly, only DNS SRV is the client side solution. Sixthly, based on the evaluation results, the proposed FFF has the fastest failover. Seventhly, only BIG-IP would face the single-point-of-failure problem. Eighthly, The FFF and redundant hop of SIP load balancer design can make the SIP dispatchers become all active by using DNS SRV. However, the parallel SIP proxy servers design can not take this advantage due to the limitation of LVS design.

Approach	DNS SRV [3]	Redundant hop of proxy server [3]	BIG-IP [11]	Redundant hop of load balancer [3]	Parallel SIP proxy servers [14]	FFF (proposed)
Configuration (dispatchers * proxy servers)	*0*m*	*0*2*	*1*m*	2*m*	2*m*	*n*m*
Failover mechanism	DNS SRV	VRRP	use SIP Option request as heartbeat	VRRP	VRRP + LVS + Keepalived	OpenAIS
Load balancing support	Yes (DNS support)	No	Yes	Yes	Yes	Yes
Redundancy model (*n*+*k*)	Proxy: m_1+0	Proxy: $1+1$	Proxy: m_1+0	Dispatcher: $1+1$ Proxy: m_1+k	Dispatcher: $1+1$ Proxy: m_1+0	Dispatcher: n_1+k Proxy: m_1+0
Client and/or DNS involved	Yes	No	No	No	No	No
Failover mechanism (failover time)	dependent on DNS updating period	VRRPD (3.163 s)	Not mentioned	VRRPD (3.163 s)	VRRPD (3.163 s)	OpenAIS (0.651 s)
Single point of failure problem	No	No	Yes	No	No	No
Scalability (with DNS SRV)	No	No	No	Yes	No	Yes

Table 1. Comparison of different SIP high availability networks

Chapter 3

System design approach

In order to achieve “five nines” availability in SIP high availability networks, a fast failure detection and failover (FFF) scheme is proposed in this sector. The cluster of SIP proxy servers is controlled by centralized job controllers, which are SIP dispatchers. We use an *n* + *k* redundancy model for SIP dispatchers. The reason of using a clustered dispatchers design is not only protecting the system from crashing caused by the single-point-of-failure problem but also improving the overall system availability. With the proposed FFF, the carrier class reliability of SIP networks can be achieved.

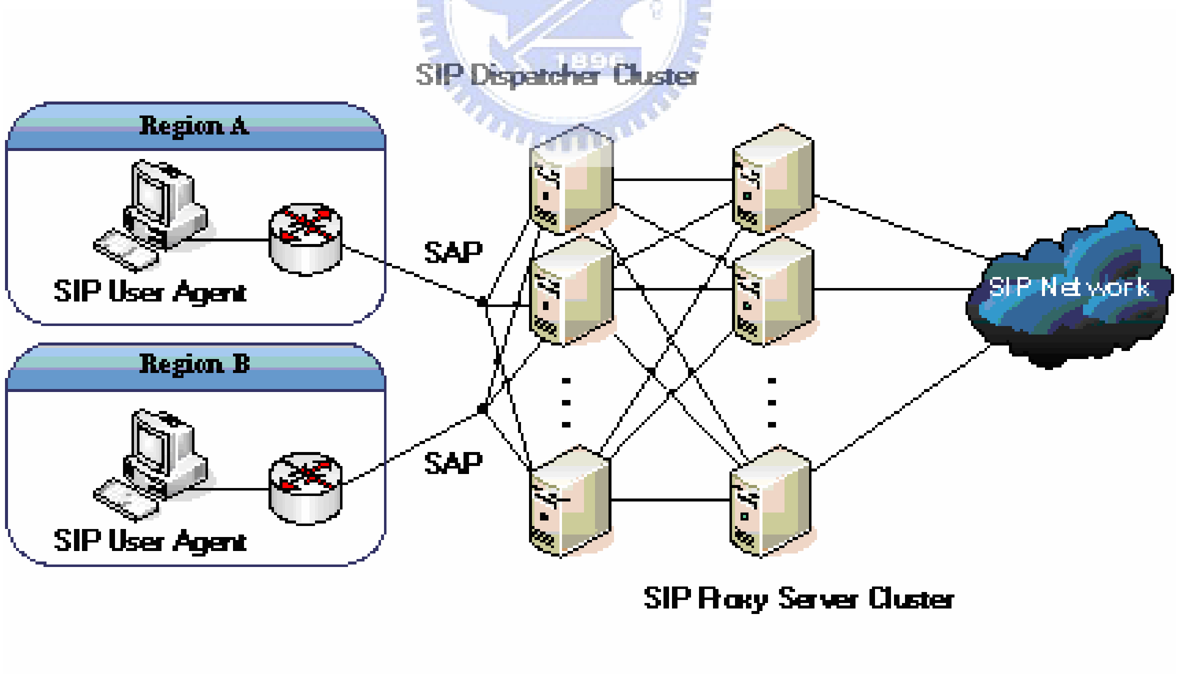

Fig. 5 shows the designed architecture of FFF for SIP high availability networks. As shown in Fig. 5, an SAP (service access point) is a virtual IP that is associated with a SIP dispatcher, which is active and is responsible for handling incoming SIP messages. With an SAP, SIP user clients can access the service correctly without knowing the high availability design at the server side. To have high availability, when an active SIP dispatcher fails, the SAP will be reassociated with the one of the backup SIP dispatchers. The newly associated SIP dispatcher will process SIP requests that are sent to the SAP to avoid service interruption. Clients at different regions (for example, regions A and B) can access respective SAP to enhance scalability.

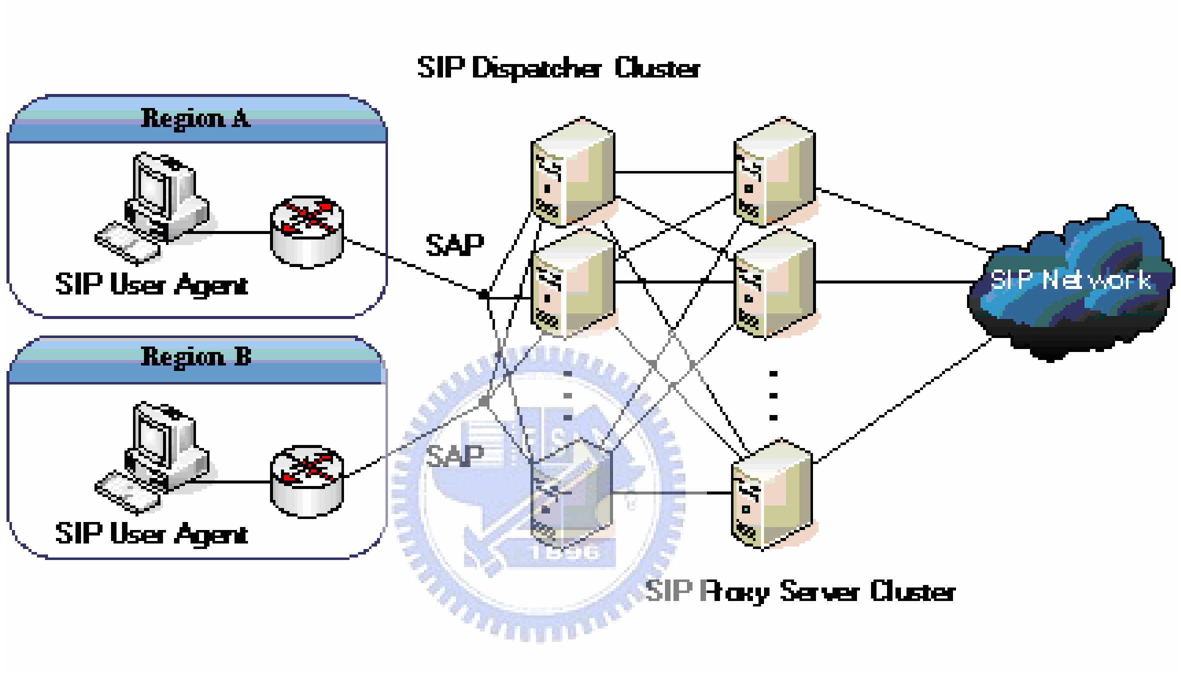

Fig. 5 . Design architecture of the proposed FFF scheme for SIP high availability networks.

The mechanism of how a backup SIP dispatcher takes over the SAP is maintained by OpenAIS AMF service. Each SIP dispatcher has its own real IP address. As to which SIP dispatcher is given the virtual IP (SAP) to become active is assigned by OpenAIS AMF service. After owing the virtual IP address, the “active” SIP dispatcher would make an announcement and map the virtual IP address to its own MAC address of the network interface. According to the group and redundant settings set in OpenAIS, AMF services running on each SIP dispatcher would keep sending control messages as heartbeats to the group members of AMF setting with a high frequency (100 *ms*), and also listen to the heartbeats of others to detect failures. When the “active” SIP dispatcher fails, the AMF would choose one of the backup (standby) SIP dispatchers to be the “active” SIP dispatcher and inform it to claim that it owns the virtual IP (SAP). In this operation, a gratuitous ARP message is sent by the new active SIP dispatcher in order to refresh the ARP table of the other network devices in the same LAN. This is an important step during the failover because all network devices on the same LAN will send SIP requests and responses to the new serving node after this operation.

In clustered SIP proxy servers, when one of the SIP proxy servers becomes unavailable, the AMF would inform all the SIP dispatchers, and they would stop forwarding SIP messages to the failed SIP proxy server. In order to monitor multiple SIP proxy servers and to achieve load balancing, the OpenAIS checkpoint service was used. Each SIP proxy server uses OpenAIS checkpoint service to keep sending a counting number as a heartbeat to SIP dispatchers. The SIP dispatchers would check the number every 100 *ms*. Note that this interval of 100 *ms* is the as to the AMF’s heartbeat interval for the health check of SIP dispatchers. If the number did not increase as expected, the SIP dispatchers would consider the associate SIP proxy server is failed.

Fig. 6 shows the system state diagram, which includes failure detection, failover and recovery operations. Server failures caused by hardware failures in this design can be detected. The failover operations will then be initiated automatically. The changing of system state and corresponding operation are listed.

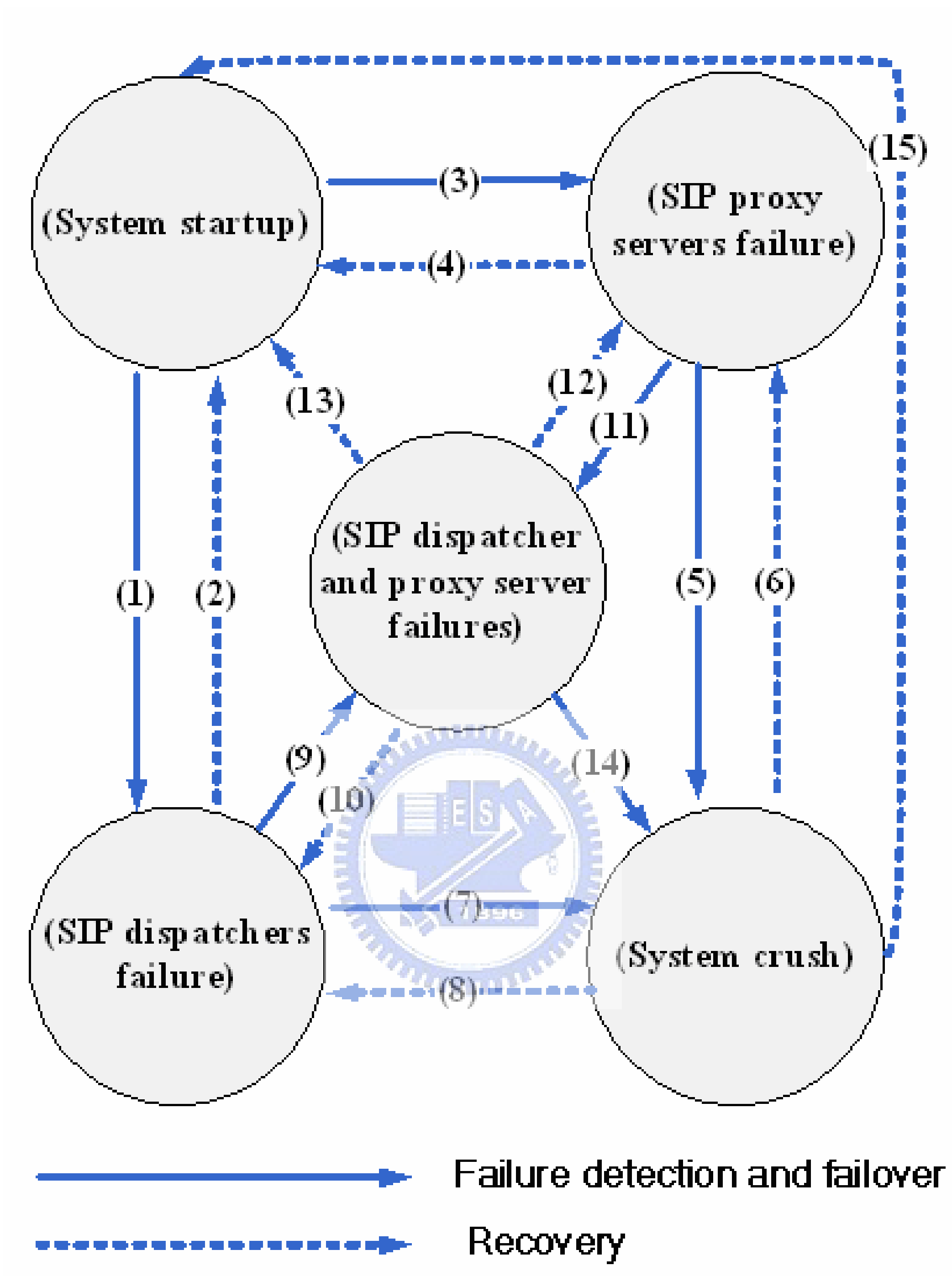
(System startup)
(SIP proxy servers failure)
(SIP dispatcher and proxy server failures)
(SIP dispatchers failure)
(System crush)
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
(13)
(14)
(15)
Failure detection and failover
Recovery

Fig. 6 . System state diagram with failure detection, failover and recovery operations.

(1) and (11): After active SIP dispatcher failure is detected, a standby SIP dispatcher claims the SAP to become active, and start to serve incoming messages. (3) and (9): After one of the SIP proxy servers failed, all active SIP dispatchers will stop delivering SIP messages to it. (5), (7) and (14): When all of the SIP dispatchers or SIP proxy servers fail, the system is considered as unavailable. (2) and (12): If one of the failed SIP dispatchers becomes available,

the system will consider it as a standby. (4), (6) and (8): If one of the failed SIP proxy server becomes available, all SIP dispatchers will add it back to the proxy server list.

The use of OpenAIS AMF service for failover is a fast failover scheme because it was designed to check the health condition of a node in a very short period of time. If the “active” SIP dispatcher fails, the AMF can detect the health condition quickly and informs one of the backups take over its job immediately. In the proposed architecture, the health condition of each SIP proxy server is monitored by all active SIP dispatchers via the checkpoint service of OpenAIS. It also has fast failover time between SIP proxy servers by .using a short heartbeat interval.

The “token” parameter in the AMF was adjusted to reduce the failover time in our design. It does not cause apparently increasing of control messages because it is a timer to decide whether the monitored server is dead if no heartbeat has been received within the “token” period. Adjusting the token parameter value will not change the generating frequency of heartbeats.

The proposed redundancy model of SIP dispatchers is *n* +*k* where *n* is the number of active SIP dispatchers (or SAPs), since each active SIP dispatcher has one virtual IP address (SAP). SIP proxy servers is an all-active redundancy model because they all can be accessed by any SIP dispatcher. If SIP user agents know all the IP addresses of SIP dispatchers or have the DNS SRV support, the design of the SIP dispatchers cluster can be enhanced as an all-active redundancy model. This is because all SIP dispatchers can serve incoming messages with its real IP address.

Chapter 4

Evaluation and discussion

SIPp [17] was used as a SIP traffic generator and analyzer in the performance evaluation. The usages of SIPp include generating SIP requests and responses, checking the operation with implemented SIPStone [18] scenarios, and reporting testing results. The default UAC (user agent client) and UAS (user agent server) scenarios which have already been implemented in SIPp were used to test the proposed SIP high availability network architecture. Fig. 7 shows the call flow of SIPp UAC and UAS scenario. Each SIP request or response is sent immediately when the corresponding message arrived.

Fig. 8 shows the evaluation environment for FFF. It contains three SIP dispatchers with 2 + 1 redundancy model and three SIP proxy servers with 3 + 0 (all active) redundancy model. SIPp UAC and UAS were placed at different sides of the proposed SIP network to test the functions and failover operations of dispatchers and proxy servers. All SIP servers (dispatchers and proxy servers) were implemented by OpenSER on Linux operating systems.

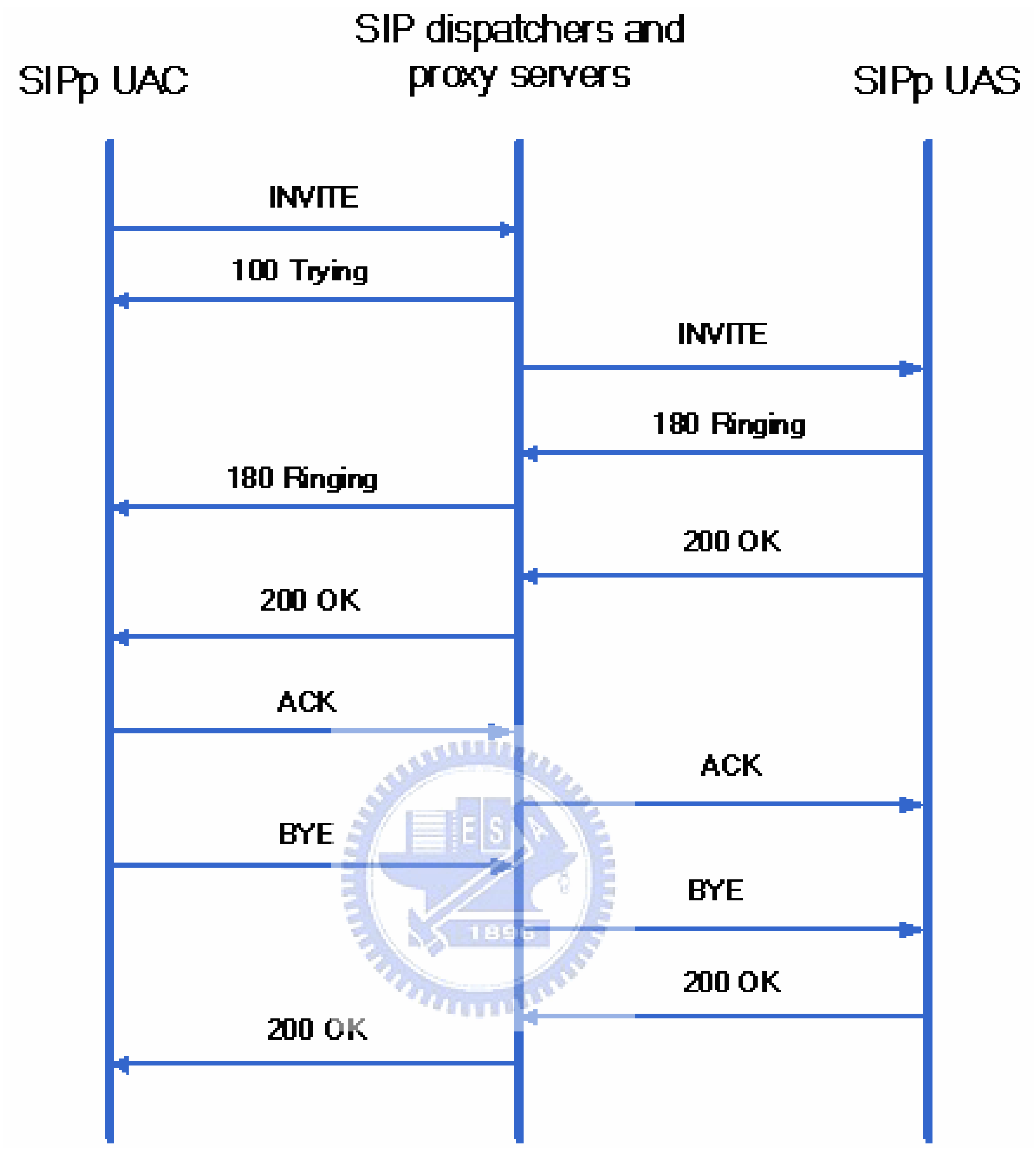

Fig. 7. SIPp UAC and UAS scenario.

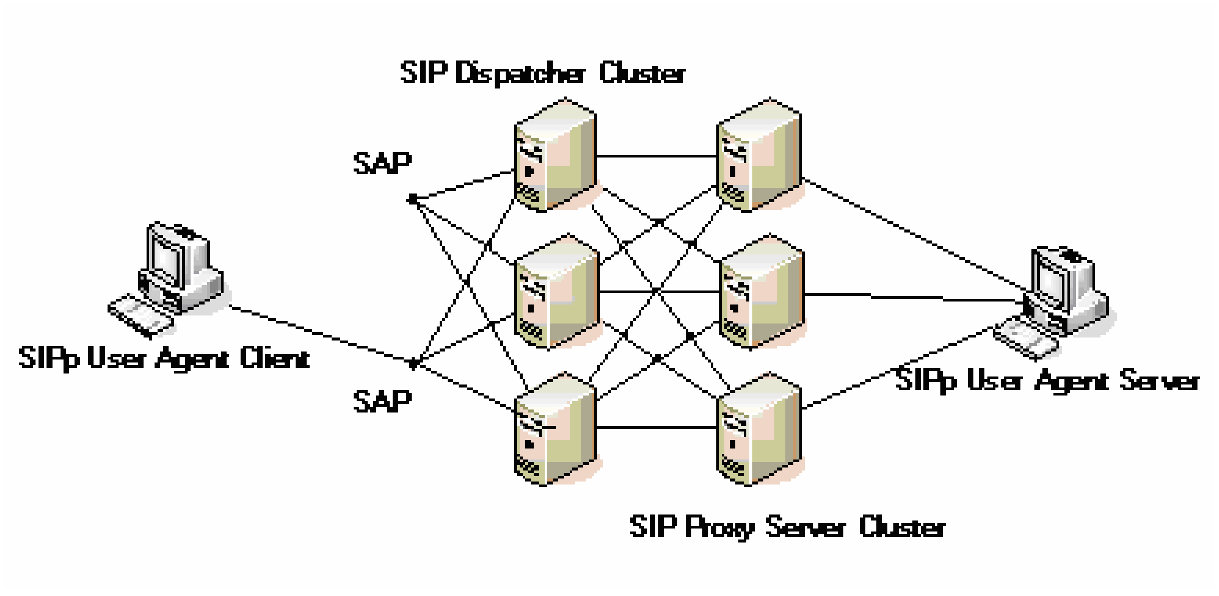

Fig. 8 . Evaluation environment for FFF.

All SIP servers are running on the virtual machines of one PC. The SIPp user agent client and user agent server were running on the other machine. The testing environment parameters settings are shown in Table 2.

Environment parameter	**value**
CPU (for virtual machines)	Pentium 4 3.0G
Memory (for virtual machines)	3 Gigabytes
CPU (for SIPp)	Pentium 4 m 1.8G
Memory (for SIPp)	512 Megabytes
Call rate (test for failover time) in SIPp	10 calls/sec (default in SIPp)
Call rate (test for number of failed calls) in SIPp	50, 100, 150, 200 calls/sec
Call limit	10000 calls
Token (OpenAIS AMF)	300, 1000 (default in AMF) *ms*

Table 2. Evaluation environment parameters Settings

The call generating speed of SIPp was set as default, 10 packets per second. The default setting of OpenAIS AMF "token" value (1000 *ms*) and a smaller value (300 *ms*) were used to test if this parameter would affect the failover time or not. 300 ms is the smallest value allowed in the OpenAIS configuration.

The failover time between the SIP dispatchers is defined as the elapsed time that the standby SIP dispatcher detects the failure of the active SIP dispatcher, takes over IP and serves a new incoming SIP request correctly. In the test of failover time, we induced a service failure by power down the active SIP dispatcher. We compared the SIP dispatcher failover time of ours with that of VRRP, because it is a common used failover mechanism by IP takeover. The VRRP was setup by using an open source demon VRRPd [19] with default settings.

The effect of CPU loading on failure time is first evaluated. Low CPU loading is defined as there is no extra loading of other traffic. Running a heavy CPU program in the background is to simulate the high CPU loading condition that all SIP servers have 100% CPU loading. Fig. 9 shows the effect of CPU loading on failover time. Firstly, the FFF scheme had much shorter failover time than VRRPd. Secondly, the smaller value (300 *ms*) of the token parameter could reduce the failover time to almost half compared to the default (1000 *ms*). Thirdly, under the high CPU loading condition, all schemes would have longer failover time, but the FFF still has shorter failover time than the other two. The failover time of the proposed FFF was not affected too much by the increase of CPU loading (0.651 vs. 0.769). The results also show that the FFF scheme reduces the dispatcher failover time by 80% (0.651 vs. 3.163) compared to VRRP.

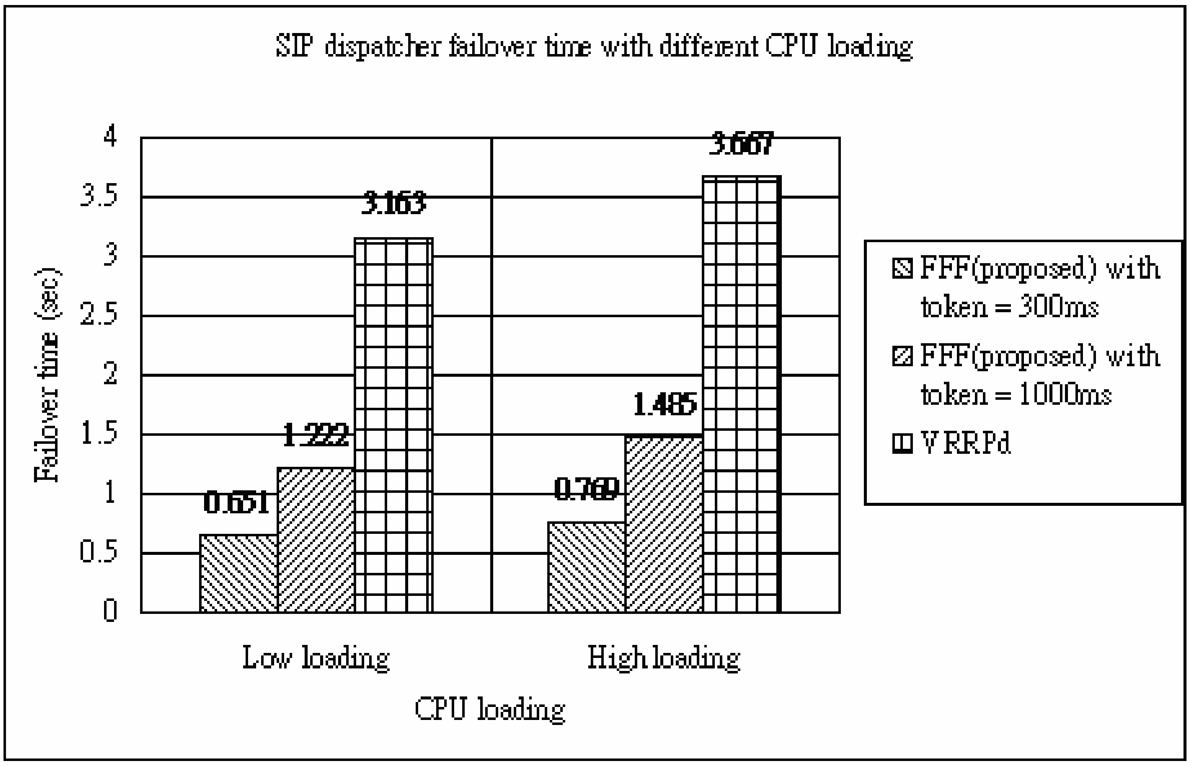

Fig. 9. Effect of CPU loading on failover time.

Normally, SIPp will not show the number of failed calls during the test of failover time because it would retransmit the SIP requests when no response is received and the overall retransmission time is much longer than the failover time. Therefore, we turned off SIPp's retransmission mechanism and computed the number of failed calls caused by the SIP dispatcher failover under different call rates. The retransmission mechanism is only turned off at the UAC side. Fig. shows failed calls caused by a dispatcher failure under different call rates. Using VRRP, it resulted in a much higher number of failed calls because its failover time was much longer than that of FFF. In addition, adjusting the token value would also reduce the number of failed calls. This result shows that the FFF scheme would have fewer failed calls compared to the VRRPd scheme when a SIP dispatcher failed.

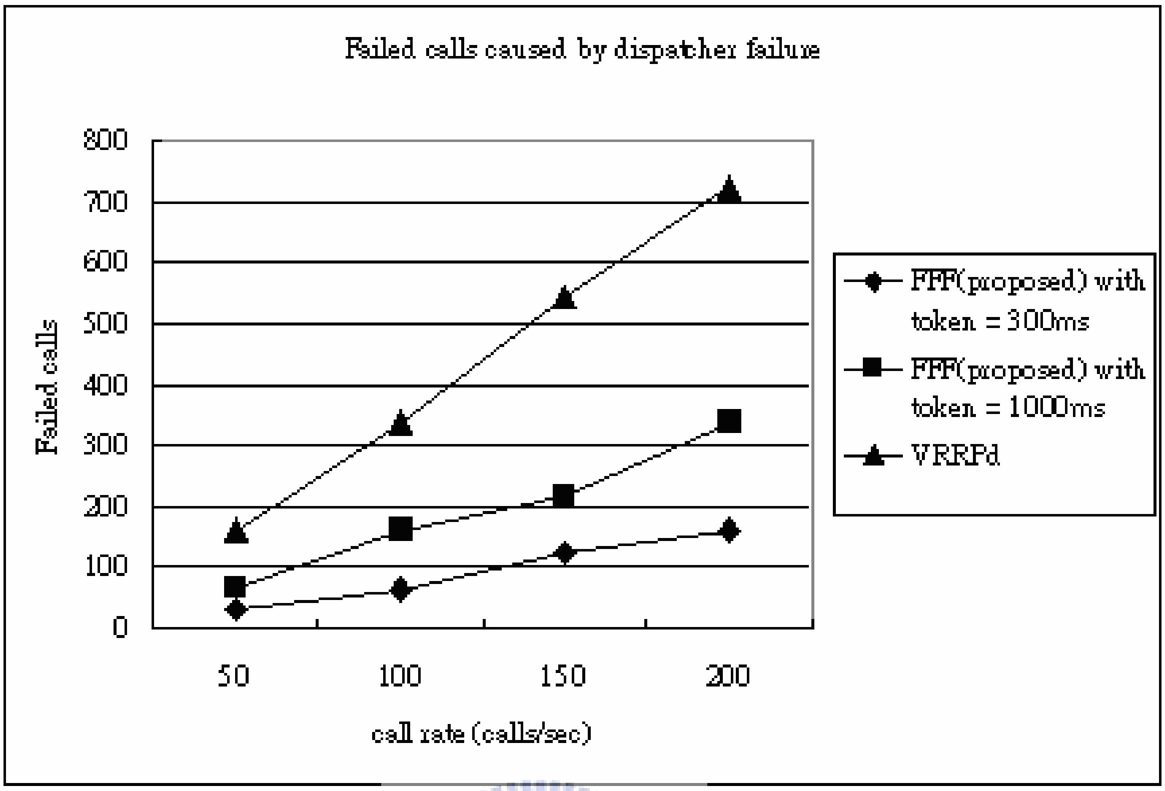

Fig. 10. Failed calls caused by dispatcher failure under different call rates.

The SIP proxy server failover time is defined as how long the SIP dispatchers can detect a proxy server failure and stops forwarding the SIP messages to it when the SIP proxy server failed. The OpenSER with default settings was used to compare with the FFF scheme because the OpenSER itself has a SIP proxy failover scheme. Fig. 11 shows the SIP proxy server failover time comparison. The result shows the proposed FFF scheme can detect a failure and react much faster than OpenSER when a SIP proxy server failed. If SIP requests were forwarded to the failed SIP proxy server, the packets would be lost and the call setup procedure would halt until the SIP dispatcher or SIP user agent retransmits the packets to the serving SIP proxy server. That is, using the FFF scheme, the SIP client will encounter less probability of service errors during the SIP proxy server failover time. The results show that the FFF scheme shortens the proxy failover time by 85% (3.89 vs. 24.79) compared to the mechanism provided by OpenSER itself.

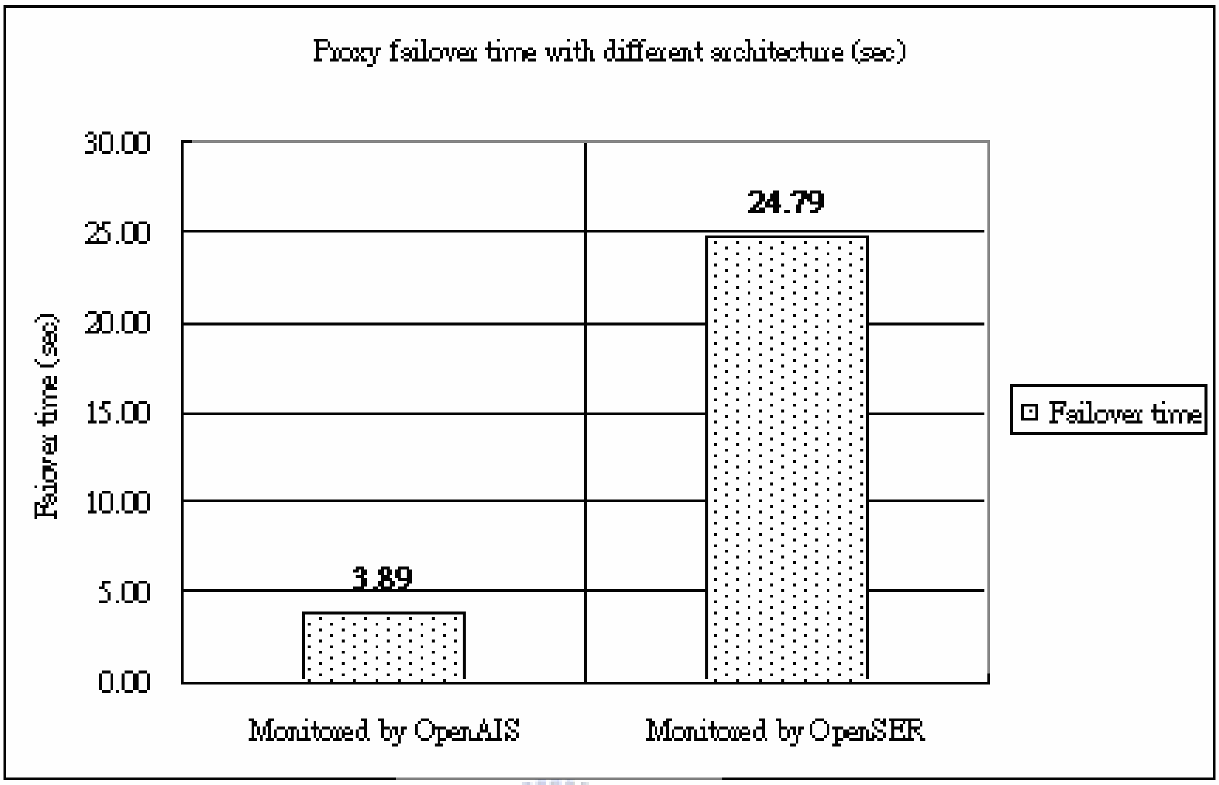

Fig. 11. SIP proxy server failover time comparison.

Chapter 5

Analysis and discussion

In this section, based on our proposed high availability SIP network architecture, we analyze theoretical service availability of using multiple SIP dispatchers and multiple SIP proxy servers. The availability of a SIP dispatcher ($A_{dispatcher}$) can be calculated by the mean time to failure of a SIP dispatcher ($MTTF_{dispatcher}$) and the mean time to restore of a SIP dispatcher ($MTTR_{dispatcher}$). The availability of a SIP proxy server (A_{proxy}) can be calculated similarly. The overall service availability ($A_{service}$) of the network architecture can be calculated by the availability of clustered SIP dispatchers ($A_{dispatcher\_cluster}$) and the availability of clustered SIP proxy servers ($A_{proxy\_cluster}$). The equations are expressed as follows:

$$A_{dispatcher} = \frac{MTTF_{dispatcher}}{MTTF_{dispatcher} + MTTR_{dispatcher}} \tag{1}$$

$$A_{proxy} = \frac{MTTF_{proxy}}{MTTF_{proxy} + MTTR_{proxy}} \tag{2}$$

$$A_{dispatcher\_cluster} = 1 - \left(1 - A_{dispatcher}\right)^{n} \tag{3}$$

$$A_{proxy\_cluster} = 1 - \left(1 - A_{proxy}\right)^{m} \tag{4}$$

$$A_{service} = A_{dispatcher\_cluster} \times A_{proxy\_cluster} \qquad (5)$$

We analyzed the overall service availability based on $MTTF_{dispatcher}$ = $MTTF_{proxy}$ = 5000 hours and $MTTF_{dispatcher}$ = $MTTR_{proxy}$ = 72 [20]. By this setting, the service availability of a single dispatcher $A_{dispatcher}$ or proxy server A_{proxy}, is about 98.58%. This value is close to the average service availability of a server implemented for real services (98.46%),[21].

Fig. 12 shows the overall service availability in terms of how many nines with respect to various number of dispatchers and proxy servers. The analysis results show the overall service availability will reach a limit if we only increase the number of SIP proxy servers. To increase the overall service availability, we have to increase the number of dispatchers as well as the number of proxy servers. From Fig. 12, we found that to achieve a carrier class reliability of five nines service availability with a minimal cost, we need to install three SIP dispatchers and three SIP proxy servers. This figure also shows that to enhance service availability with a minimal cost, the number of dispatcher should be equal to the number of proxy servers.

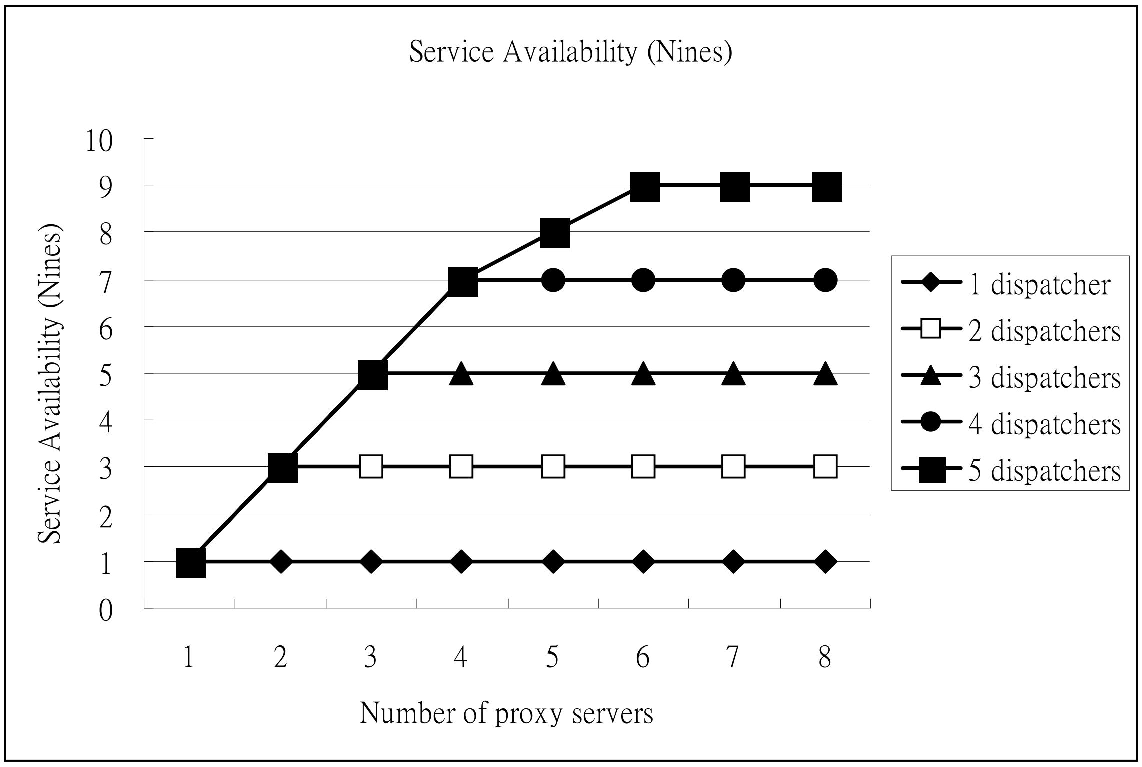

Fig. 12 Analysis of service availability under different numbers of dispatchers and proxy servers.

The service availability under different MTTFs was also analyzed. We used the same numbers of SIP dispatchers and SIP proxy servers. That is, it is a *n* * *n* SIP network. Fig. 13 shows that when the availability (MTTF) of a single server is low (high), it is necessary to use more servers to build a SIP high availability network.

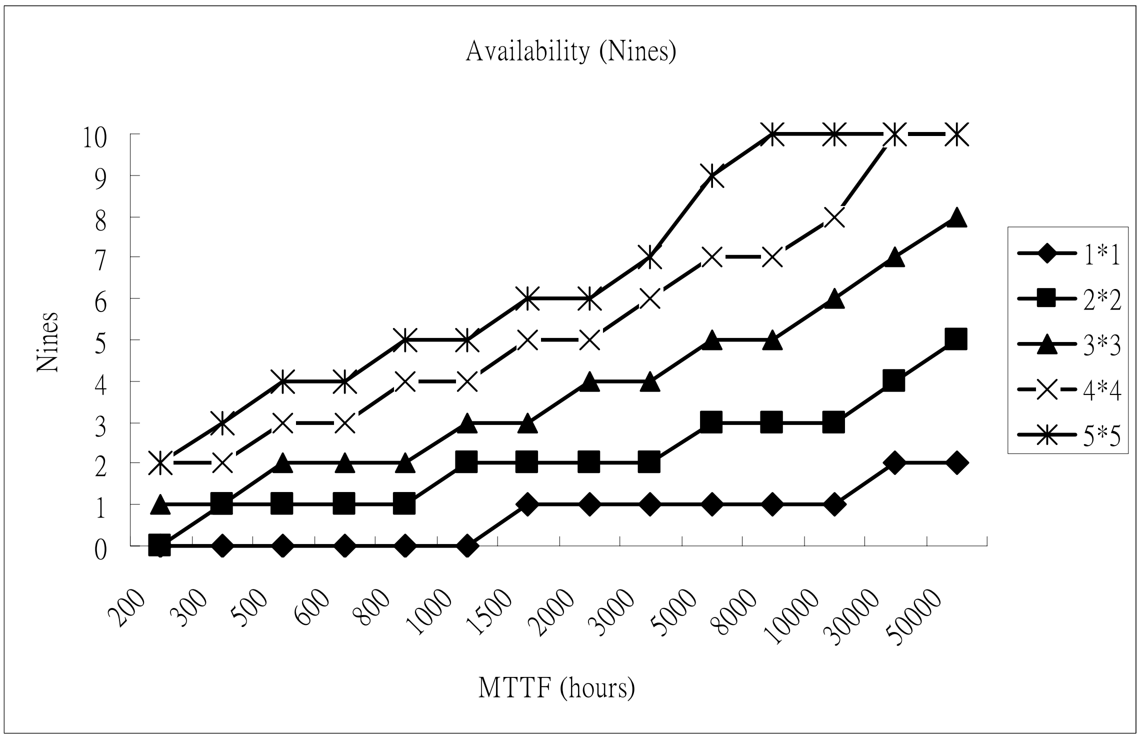

Fig. 13 Analysis of service availability under different MTTFs.

Chapter 6

Conclusion and future work

6.1 Conclusion

The traditional designs of SIP high availability networks have a potential problem of limited overall service availability since they used limited (one or two) dispatchers and multiple SIP proxy servers. This was verified by our analysis results that has the following conclusion. Increasing the number of SIP proxy servers will increase the service availability initially; however, the enhancement of the service availability will encounters a bottleneck if SIP dispatchers were not increased proportionally as well. The proposed fast failure detection and failover (FFF) scheme uses multiple SIP dispatchers and multiple SIP proxy servers to reach carrier class reliability. In this thesis, the OpenAIS was used as a high availability middleware to handle the failure detection and failover among SIP dispatchers and SIP proxy servers. Experimental results have shown that the FFF scheme reduces the SIP dispatcher failover time by 80% compared to the VRRP solution and also shortens the SIP proxy server failover time by 85% compared to the mechanism provided by OpenSER itself. Analysis also shows that using multiple SIP dispatchers is key to raise the service availability in SIP high availability networks.

6.2 Future work

In this thesis, we only consider the hardware failure of servers, either dispatchers or proxy servers. Failures that occurred on the SIP server software (for example, failed OpenSER processes) will not initiate the failover operation in our design. If an OpenSER process can be registered as an OpenAIS component, then this problem can be solved as well. In addition, a high availability middleware (for example, OpenAIS) can perform faster failover in the condition of software failures by directly monitoring the server demons. These points deserve further study to verify.

Bibliography

[1] A.B. Johnston, SIP: *Understanding the Session Initiation Protocol*, 2nd Edition, Artech House, 2004.

[2] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: a transport protocol for real time applications *RFC 1889*, Internet Engineering Force, January 1996.

[3] Cisco Inc., "Overview of High Availability in SIP-Based Voice Networks," [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/callc_c/sip_c/sipha_c/hachap1.htm

[4] CR. Johnson, Y. Kogan, Y. Levy, F. Saheban, P. Tarapore, "VoIP Reliability: A Service Provider's Perspective," *IEEE Communications Magazine*, July, 2004, pp. 48-54.

[5] OpenAIS at Open Source Development Labs, Beaverton, OR, USA. [Online]. Available: http://developer.osdl.org/dev/openais/.

[6] Service Availability Forum. [Online]. Available: http://www.saforum.org.

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

[8] K. Singh, H. Schulzrinne, "Failover, load sharing and server architecture in SIP telephony," *Computer Communications*, Volume 30, Issue 5, March, 2007, pp. 927-942.

[9] M. Mealling, R. W. Daniel, "The Naming Authority Pointer (NAPTR) DNS Resource Record," *RFC 2915*, Internet Engineering Task Force, Sept. 2000.

[10] J. Rosenberg and H. Schulzrinne, “Session initiation protocol (SIP): locating SIP servers,” *RFC 3263*, Internet Engineering Task Force, June 2002.

[11] F5 Networks Inc., “A New Paradigm for SIP High Availability and Reliability,” [Online]. Available: http://www.f5.com/solutions/technology/pdfs/sip_wp.pdf.

[12] “OpenSER,” [Online]. Available: http://www.openser.org/.

[13] S. Knight, D. Weaver, D. Whipple, R. Hinden, D. Mitzel, P. Hunt, P. Higginson, M. Shand, and A. Lindem. *RFC 2338: Virtual Router Redundancy Protocol*, April 1998.

[14] V. Matic, I. Franicevic, D. Sekalec, “Parallel SIP Proxy Servers Using Direct Routing Approach,“ in *Proc. International Conference on Software in Telecommunications and Computer Networks*, Sept. 2006, pp. 218-222.

[15] “Linux Virtual Server,” [Online]. Available: http://www.linuxvirtualserver.org/.

[16] “Keepalived,” [Online]. Available: http://www.keepalived.org/.

[17] “SIPp - Test Tool/Traffic Generator for the SIP Protocol,” Mar 2006, [Online]. Available: http://sipp.sourceforge.net.

[18] H. Schulzrinne, S. Narayanan, J. Lennox, and M. Doyle, “SIPstone - Benchmarking SIP Server Performance,” Apr 2002, [Online]. Available: http://www.sipstone.com/.

[19] “VRRPD,” [Online]. Available: http://off.net/~jme/vrrpd/.

[20] K. Uhlemann, C. Engelmann, and S. L. Scott, “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.

[21] J. Neises, “Benefit Evaluation of High-Availability Middleware,” Service Availability, in *Proc. First International Service Availability Symposium*, May, 2004, pp. 73-85.