國　立　交　通　大　學

電信工程研究所

碩　士　論　文

應用於VoIP系統之以軟體實現的H.323閘道器與代理轉碼器之嶄新互動架構

A Novel Software-Based H.323 Gateway with Proxy-TC for VoIP Systems

研 究 生：殷偉盛

指導教授：陳伯寧 博士

中華民國九十年五月

應用於VoIP系統之以軟體實現的H.323閘道器與代理轉碼器之嶄新互動架構

A Novel Software-Based H.323 Gateway with Proxy-TC for VoIP Systems

研 究 生：殷偉盛　　Student：Wei-Sheng Yin

指導教授：陳伯寧 博士　　Advisor：Dr. Po-Ning Chen

國 立 交 通 大 學
電 信 工 程 研 究 所
碩 士 論 文

A Thesis

Submitted to Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electrical Engineering

May 2001

Hsinchu, Taiwan, Republic of China

中華民國九十年五月

應用於VoIP系統之以軟體實現的H.323閘道器與代理轉碼器之嶄新互動架構

學生：殷偉盛　　　　　　　　　　指導教授：陳伯寧 博士

國立交通大學電信工程研究所

中文摘要

在這一篇碩士論文當中，我們針對運作於個人電腦平台上軟體實現的 H.323閘道器提出了一個嶄新架構。這個動機是基於兩個觀察。首先，我們觀察到一個軟體實現的H.323閘道器的計算量負載，主要是來自於不同媒體格式的轉換動作，例如轉換語音的編碼格式。我們也觀察到一個網際網路電話終端機(在我們的系統中，網際網路電話終端機是建構在個人電腦上)或是一個軟體實現的H.323閘道器所需使用的CPU計算資源，有時會被作業系統切換到其他的工作；因此，一個正在進行中的網際網路電話可能會有瞬時的不順暢或中斷。因此我們提出一個代理轉碼器(proxy-TC) 架構來平衡網際網路電話所需的計算量負載，同時平緩所給定計算資源的浮動。根據我們的模擬顯示，在我們所提的架構之下，系統的容量和新進電話的阻斷率(blocking probability) 會有相當大的改善。除此之外，我們的模擬結果也顯示，加入代理轉碼器之間的移轉機制(TC-handoff mechanism)，僅僅微量的提高新進電話阻斷率，卻可使電話中斷率(dropping probability) 大幅的降低。

A Novel Software-Based H.323 Gateway with Proxy-TC for VoIP Systems

Student：Wei-Sheng Yin Advisor：Dr. Po-Ning Chen

Institute of Communications Engineering
National Chiao Tung University

ABSTRACT

In this paper, we present a novel architecture for *software-based* H.323 gateways that operate over PC platforms. It is motivated by two observations. First, we found that the computation load of a software-based H.323 gateway is mainly contributed by media transformation activities, such as the transcoding of voice streams. We also observed that the CPU computation power of an IP call terminal (which is a PC in our system setting) or a software-based H.323 gateway may be unevenly switched to another task by the operation system; hence, an active IP call may experience occasional interrupts. We therefore propose a proxy–transcoder (proxy-TC) architecture to balance the computation load of, as well as to smooth the fluctuation of dedicated computation powers to, the IP call activities. Our simulation showed that both the system capacity and the new call blocking probability can be significantly improved under our proposed architecture. Besides, by adding an additional TC-handoff mechanism, the active call dropping probability can be drastically decreased at the expense of a little increase in the new call blocking probability.

Table of Contents

List of Figures

Chapter 1

Introduction

In recent years, the Internet has been widely deployed, and the number of Internet hosts and users has been dramatically increased. Due to the constant growth of transmission bandwidth, real-time applications, such as the Internet Telephony or Voice Over Internet Protocol (VoIP), gradually become feasible on the Internet.

VoIP can bring us advantages like toll-by-pass, coexistence of voice and data services, and easy deployment of telephony network between two branch offices. In techniques, VoIP carries voice traffic as data packets over a packet-switched network, instead of as a synchronous stream of binary sampled data over a circuit-switched, time-division multiplexed (TDM) network.

To cope with this trend, the International Telecommunication Union (ITU) has initiated various standardization activities (e.g. [1-3]) specifically on packet voice transmission, where the most renowned one is perhaps the H.323. The H.323 defines the components for real-time communication over packet networks, including Gateways, Gatekeepers and Terminals. According to the standard, the Gateway is responsible for real-time, two-way communication between the H.323 Terminals on the LAN side and other Terminals on the Switched Circuit Network (SCN) side (e.g., PSTN), and it shall provide the appropriate translation for media formats and communication procedures.

Through some experimental implementation on H.323 gateways, we found that the computation load of a pure software-based H.323 gateway is mainly contributed by media transformation activities, such as the transcoding of voice streams. We also observed that the CPU computation power of an IP call terminal (which is a PC in our system setting) or a software-based H.323 gateway may be unevenly switched to another task by the operation system; hence, an active IP call may experience occasional interrupts. We therefore propose a proxy–transcoder (proxy-TC) architecture to balance the computation load of, as well as to smooth the fluctuation of dedicated computation powers to, the IP call activities. Our simulation showed that both the system capacity and the new call blocking probability are significantly improved under our proposed architecture. Besides, by adding an additional TC-handoff mechanism, the active call dropping probability can be drastically decreased at the expense of a little increase in the new call blocking probability.

Chapter 2

Proxy-TC Architecture

2.1 Motivations

The system architectures of a PC-based H.323 gateway on the market can be roughly divided into two categories. The first category incorporates separate powerful modules, such as DSP-based add-on cards for medium transcoding, to achieve high performance and stable communication quality, while the second category implements the whole gateway functionality entirely on PC platform in order to provide a low price solution to small businesses. One can then choose between these two types of gateways to suit its need.

Nowadays, the Internet connection and the deployment of corporate network almost become prerequisites for businesses. By installing the software-based gateway onto an available PC on the existing network, a small office (e.g., with several people work together but are separately located) can easily set up their inter-branch telephony network without extra hardware cost. Although cost-effective, the telephony links maintained by a software-based gateway at times suffer short interrupts due to the possible uneven switching of PC platforms among several computation-bound tasks. Moreover, the number of active calls is limited by the computation capability of the gateway PC.

On the other hand, we observe that the computation powers of the desktop PCs in use for general employees are sometimes not in full utilization. This leads us to the idea of gathering these excessive computation powers to share the load of the software-based gateway, and hence, reduce the number of computation-bound tasks on it. In such case, one may further increase the gateway capacity without introducing additional hardware cost.

In order to comply with H.323, we propose to only transfer the load of the transcoding operation to these office desktops with excessive computation powers, which are named the *proxy-transcoders* (proxy-TC). By installing an agent, the proxy-TC can perform the media format transformation, such as the encoding and decoding of G.723.1, in the background under the monitor of its associated gateway. The coded medium will then be sent to its call destination directly. Since the excessive computation power of an employee's desktop may vary in a burst fashion with time, a handoff-like mechanism to switch the job of a medium transcoding from a proxy-TC to another is also proposed to cope with a sudden shortage of computation power in a proxy-TC. Detailed command-exchange procedure that conforms to H.323 will be introduced in subsequent subsections.

The main advantage of our proposed H.323 compliant architecture is its scalability, where the performance of a gateway can be easily enhanced by simply adding one or more computers as its proxy-TCs. With these proxy-TCs, a powerful software-based gateway can be established.

2.2 System Architecture

The complete internetworking scenario is depicted in Fig. 1.

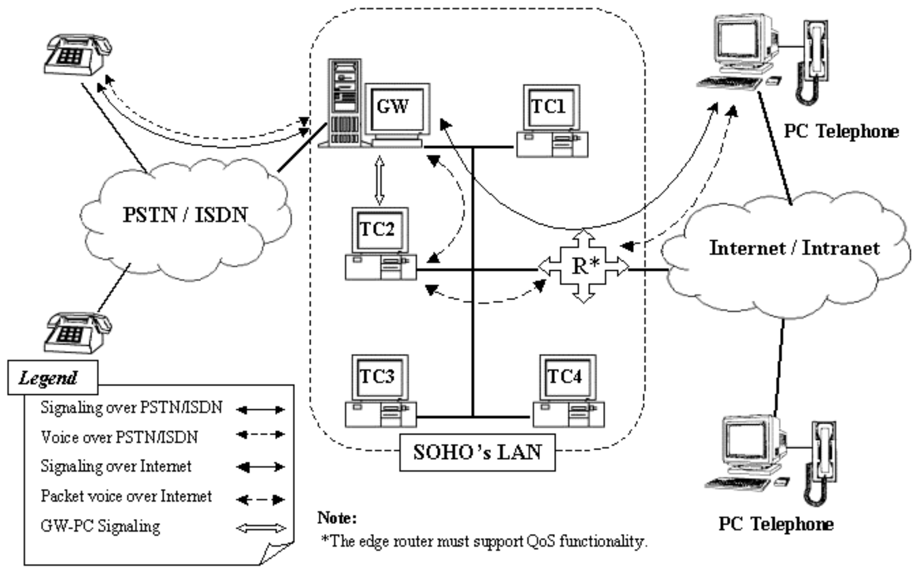


Fig. 1 The software-based gateway with proxy-TC architecture.

In our system setting, the software-based gateway, as shown in Fig. 1, composes of a PC equipped with PSTN (or ISDN) and LAN cards and the Windows NT operating system. The supported telephony protocols are PSTN telephony on the PSTN side and H.323 on the LAN side.

On the Control Plane, the gateway performs the signaling protocol conversion. In the scenario mentioned above, the control channels on the LAN side, which carry H.225 call signaling and H.245 control signaling, are established between the gateway and the H.323 endpoints involved in a call, and the control signaling messages are directly exchanged between the gateway and the H.323 endpoints on the control channels without being forwarded by the proxy-TC.

On the User Plane, the voice streams from the PSTN side may need to be encoded using the standard codec algorithms (e.g. G.723.1 [4] or G.729 [5]), which are pre-negotiated through H.245 control signaling, and then be packetized for transmission on the LAN side. In our proposed architecture, when the gateway is busy upon a call arrival, the voice encoding will be performed by one of the proxy-TCs on the LAN side. The proposed procedure is as follows.

First, the gateway finds an available proxy-TC that has enough CPU resource to cooperate with through handshaking. And then the gateway reads the voice stream from the telephony module in G.711 format, and paketizes and transmits them to the proxy-TC for voice encoding. After the proxy-TC performs voice encoding, the encoded voice stream are re-packetized and directly forwarded to the destined H.323 endpoint. In the converse direction, the voice stream, produced by the H.323 endpoint and encoded in G.723 format, should be sent to the same proxy-TC for decoding into G.711 format and then forwarded to the gateway after decoding. An alternative is to let the H.323 endpoint directly send their voice stream to the gateway in G.711 format. In such case, the bandwidth consumption of the external network, as shown in Fig. 1, may be greatly increased. We justify the feasibility of using the 64Kbps G.711 voice stream exchanged between proxy-TC and gateway as the bandwidth of the Ethernet at LAN environment is now up to 100Mbps, and hence, the traffic load of the G.711 voice streams is endurable within the LAN.

In order to complete this scenario, a protocol between gateway and proxy-TC is needed, which is defined as GW-TC signaling in this thesis and will be introduced later.

2.3 Proxy-Transcoder (Proxy-TC)

The structure of the proxy-TC is depicted in Fig. 2. The proxy-TC can be viewed as a software component installed on a computer. Without ambiguity, we also refer the computer equipped with this software component as a proxy-TC (since it is meaningless to install two proxy-TC components in a PC). In our design, a computer that is willing to donate its excessive computation power to gateway transcoding activity should perform a web-registration process. Then a proxy-TC demon will be downloaded onto the computer, and will be executed in background. The demon consists of three parts: an Observer, a Controller and a Transcoder.

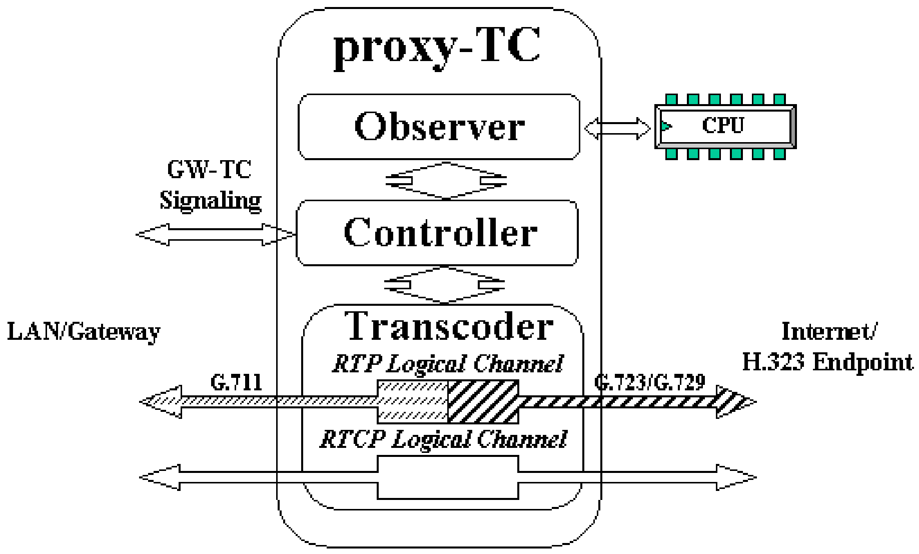


Fig. 2 The structure of a proxy-TC.

The Observer monitors local resource condition on the PC, such as available CPU power, and periodically updates the information for use by Controller.

The Controller maintains the communication between the proxy-TC and its associated gateway by using GW-TC signaling procedures. These control messages include (1) informing the gateway that the proxy-TC is still active upon the request of the proxy-TC's associated gateway, (2) receiving the request of the transcoding service form the gateway, and (3) updating the information of CPU utilization.

The Transcoder is responsible for voice stream transcoding between different speech coding algorithms, such as G.711, G.723.1 and G.729.

Another important task for the proxy-TC is to handle the RTP/RTCP protocols [6-7] for the traffic sent between the proxy-TC and the destined H.323 endpoint. In order to comply with the H.323 standardization, the proxy-TC only deals with the RTP packets, but forwards the original RTCP packets between the H.323 endpoint and the gateway.

2.4 The GW-TC signaling

The major goal of this GW-TC signaling is to enable the cooperation between the gateway and the proxy-TCs. There are seven commands in the GW-TC signaling, which are respectively described below:

- **Setup:** This command is used by the gateway to request one of the proxy-TCs to perform the transcoding service upon the occurrence of a new call.
- **Release:** The gateway uses this command to release the transcoding service either (1) upon the end of a call or (2) upon the receipt of the **Emergency** command from the proxy-TC.
- **Register:** The proxy-TCs send this command manually to the gateway (through web-browsing activity) to register to the gateway so that the gateway can create corresponding proxy-TC record.
- **Update:** The proxy-TCs use this command to update their latest statuses kept on the gateway.

- **Emergency:** This command is sent by the proxy-TCs to notify the gateway of their (sudden) shortage of transcoding resources.
- **Query:** If the update timer expires the gateway will send this command to request the proxy-TC to update its status.
- **Acknowledge (ACK):** This command may be sent by both the gateway and the proxy-TCs to inform the other party that the previous command has been successfully received. This command may also contain additional information, such as, when a proxy-TC receives the **Setup** command from a gateway, the proxy-TC should allocate two port numbers respectively for RTP and RTCP, and these port numbers can be returned to the gateway in a piggyback fashion with the **Acknowledge** command.

2.5 GW-TC signaling procedure

Fig. 3 illustrates the procedure for a normal call setup that originates by the PSTN side, and that is destined to an IP phone. In addition to the call procedure commands defined in H.323, the gateway sends the **Setup** command to proxy-TC1 before sending the H.245 commands. Proxy-TC1 then responses with an **ACK** command, consisting of a pair of port numbers for RTP/RTCP that is used by proxy-TC1 for this medium stream. Upon receipt of the H.245 **OpenLogicalChannel** command from the H.323 endpoint, the gateway also obtained two port numbers respectively for RTP/RTCP channels that are used on the H.323 endpoint; these pair of port numbers will then be sent to proxy-TC1, again, using **ACK** command. One may view the second **ACK** command as an acknowledgement to the previous **ACK** command from proxy-TC1.

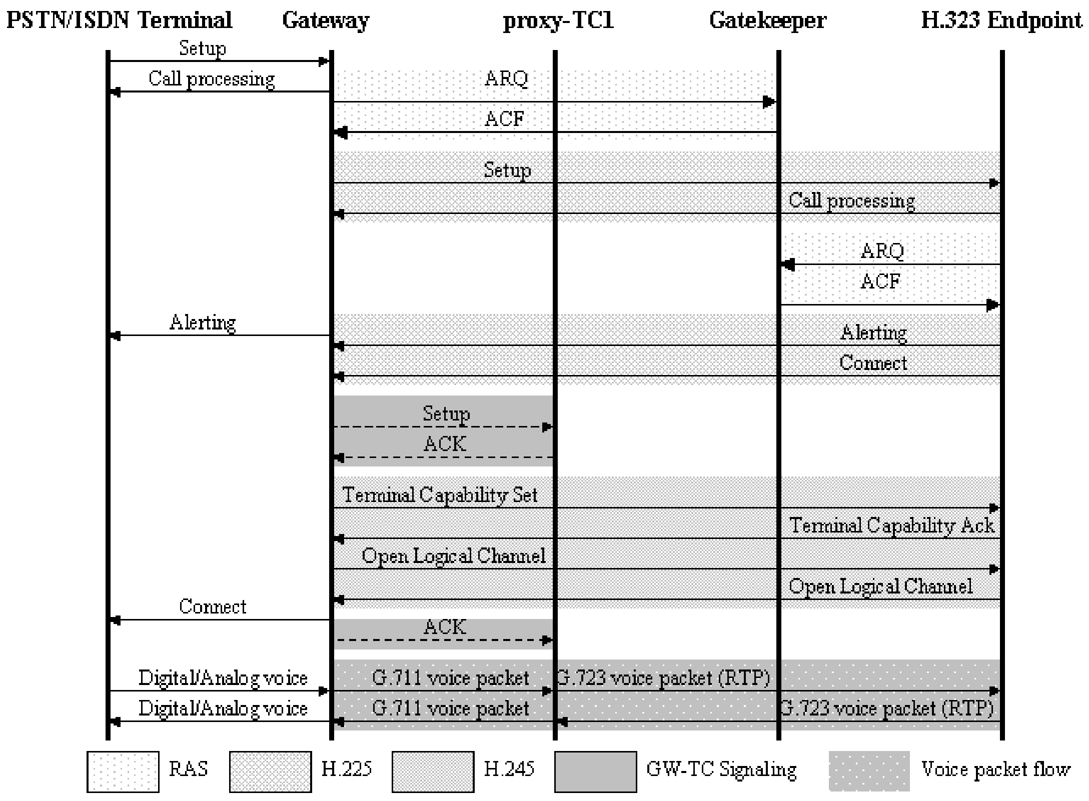


Fig. 3 A normal call setup procedure.

The call clearing procedure, initiated by the PSTN side, is shown in Fig. 4. The **Release** command is sent to the proxy-TC1 from the gateway after the gateway completes the H.225 call clearing signaling. An ACK will be replied to indicate the completeness of the resource releasing of the proxy-TC. In the end, the gateway will complete the RAS call clearing procedure defined in H.323.

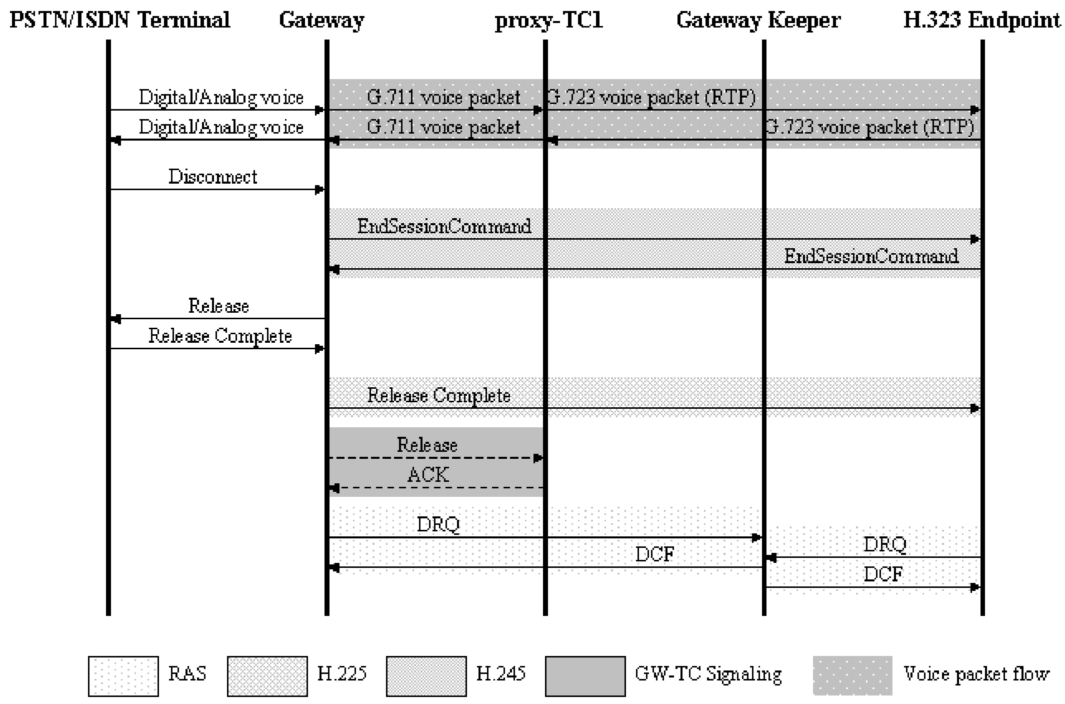


Fig. 4 A normal call clearing procedure.

2.6 The handoff mechanism

As mentioned previously, the CPU usage in reality occasionally appears burstily. So when the Observer on the proxy-TCs detects that the CPU resource is too low to perform voice encoding, it will inform the gateway using the **Emergence** command, and the gateway should find another proxy-TC to take over the call at once.

Fig. 5 shows the handoff procedure. The design of the handoff procedure is based on the concept of the so-called *soft handoff*. When the gateway receives the **Emergency** command from proxy-TC1, it handoffs the call to proxy-TC2. Specifically, the gateway requests proxy-TC2 for transcoding service before it releases the transcoding service of proxy-TC1. After proxy-TC2 successfully establishes new logical channels, the transcoding service on proxy-TC1 is then released.

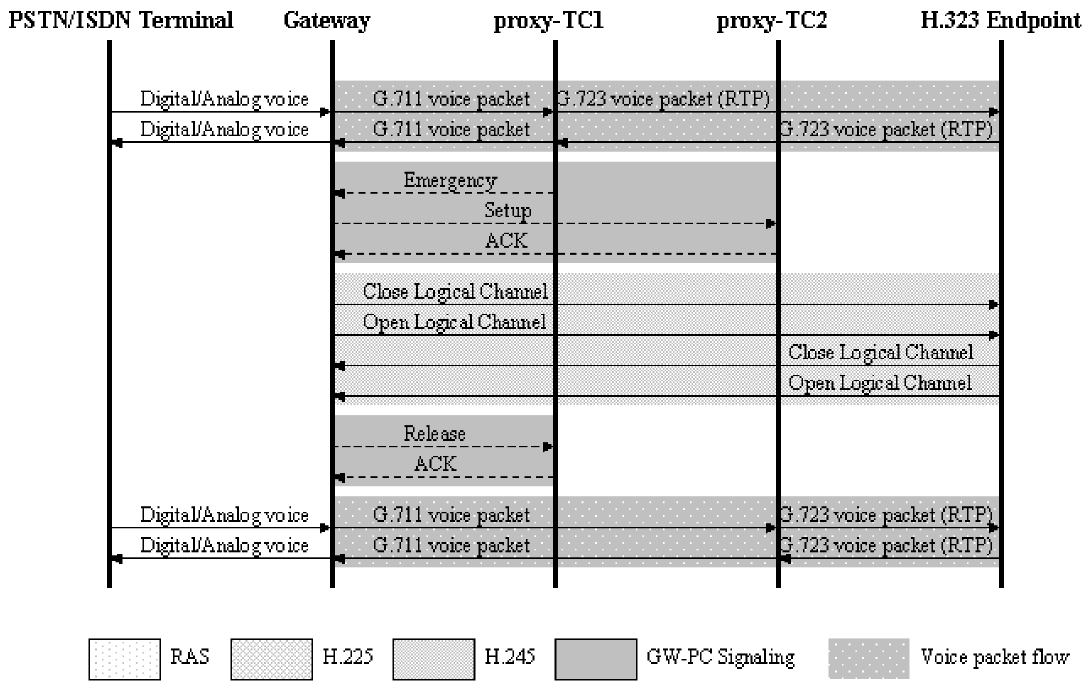


Fig. 5 The handoff procedure.

2.7 The error recovery procedure

To ensure the stability of the system, a proper design of an error recovery procedure is necessary. Basically, our design incorporates the time-out mechanism to cope with possible errors encountered in any normal procedure.

Fig. 6 shows the error recovery procedure during call setup. As shown in the figure, the gateway tries to request proxy-TC1 for transcoding service by sending the **Setup** command, but the **ACK** command is not received before the expiration of the setup timer. As a result, the gateway turns to request proxy-TC2 for transcoding service, and the setup procedure succeeds upon the receipt of the **ACK** command from the proxy-TC2.

The emergency error recovery procedure is shown in Fig. 7. The upper portion of Fig. 7 illustrates a normal emergency procedure while the lower portion of this figure illustrates how the proxy-TC reacts when error occurs in this procedure. As shown in the lower portion of the figure, proxy-TC1 sends the **Emergency** command to the gateway, but does not receive the **ACK** command before the expiration of the emergency timer. Because this procedure is time-sensitive, the emergency time-out time should be kept short, for which we suggest to be less than 50ms or lower. When not receiving the **ACK** command, proxy-TC1 repeats the sending of the **Emergency** command until its successfully receiving the **ACK** command. At last, if the time during which proxy-TC1 does not receive the **ACK** command while keep sending the **Emergency** command exceeds a threshold, possibly 500ms or 1s, this procedure, as well as the call, will be dropped.

As mentioned previously, the proxy-TCs should periodically update their latest statuses to the gateway to keep the corresponding proxy-TC records up-to-date. Fig. 8

shows some occasional situations of the update procedure. In our design, a proxy-TC should constantly use the **Update** command to update its status, and the gateway should trace this procedure by the update timer. At the moment of status updating, the gateway resets the update timer (i.e., (1) and (3) of Fig. 8), if the timer expires the gateway will send **Query** command to request the proxy-TC to update their status (i.e., (2) and (4) of Fig. 8). If the number of times of sending **Query** command without response exceeds a certain threshold, the gateway will recognize this proxy-TC as being shutdown, and remove it from the TC records. This update procedure is very critical. It can let the gateway be aware whether the proxy-TC is active or not.

Fig. 9 shows the error recovery procedure for call release. This procedure is to ensure that the proxy-TC has released the resources of transcoding services. The gateway will keep sending the **Release** command until it receives the **ACK** command.

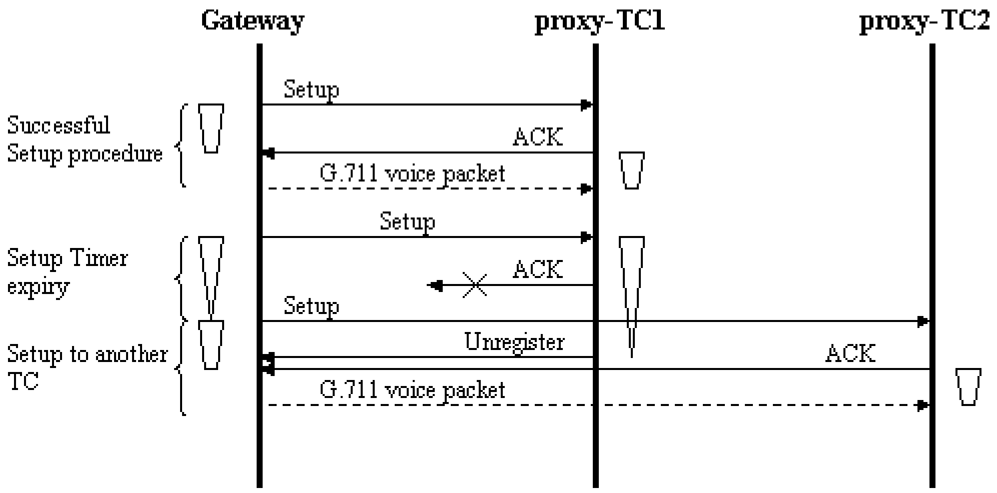


Fig. 6 The error recovery procedure for call setup.

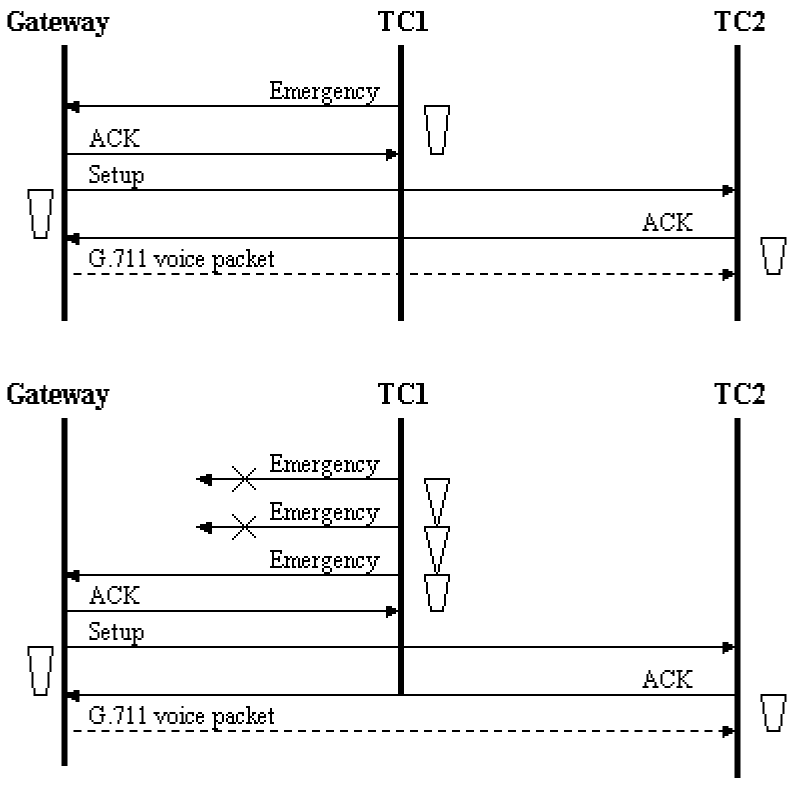


Fig.7 The error recovery procedure for call emergency

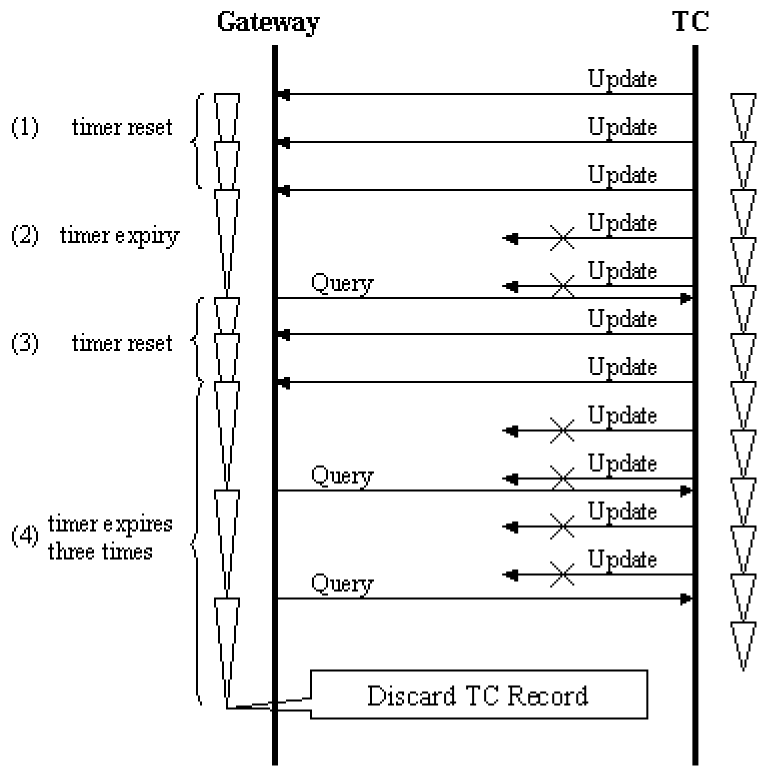


Fig. 8 The error recovery procedure for update procedure

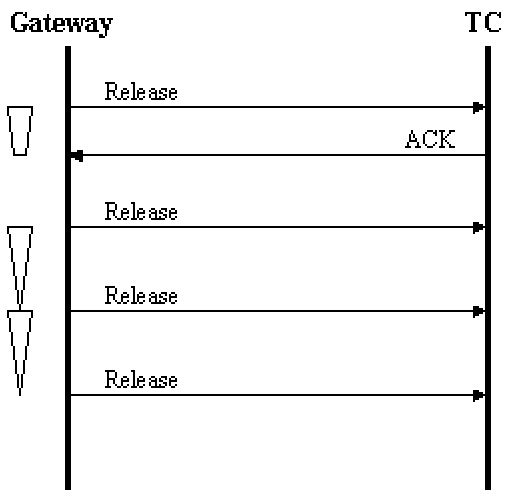


Fig. 9 The error recovery procedure for release procedure

2.8 Data structure.

To fulfill the procedures mentioned above, the gateway needs to maintain two tables: the first one records the periodic update information of CPU utilization of each proxy-TC, and the second one keeps the call information in the system. Also recorded in the first table is the CPU maximum computation power that is measured by the downloaded proxy-TC demon at the first time it is installed.

Chapter 3

Simulation Model and Results

3.1 The simulation model

Our simulations involve up to three proxy-TCs as shown in Fig. 10. The First-Fit-Round-Robin (FFRR) proxy-TC selection scheme is adopted.

We first measure the CPU utilization on four Pentium II 233 (PII 233) PCs, which are respectively shown in Fig. 11. Based on these measurements, we find no appropriate statistical models to fit their statistics. Hence, we will use these collected data directly as our simulation inputs. We assume that both the call inter-arrival time and the call duration are exponentially distributed, and the mean duration of each call is three minutes.

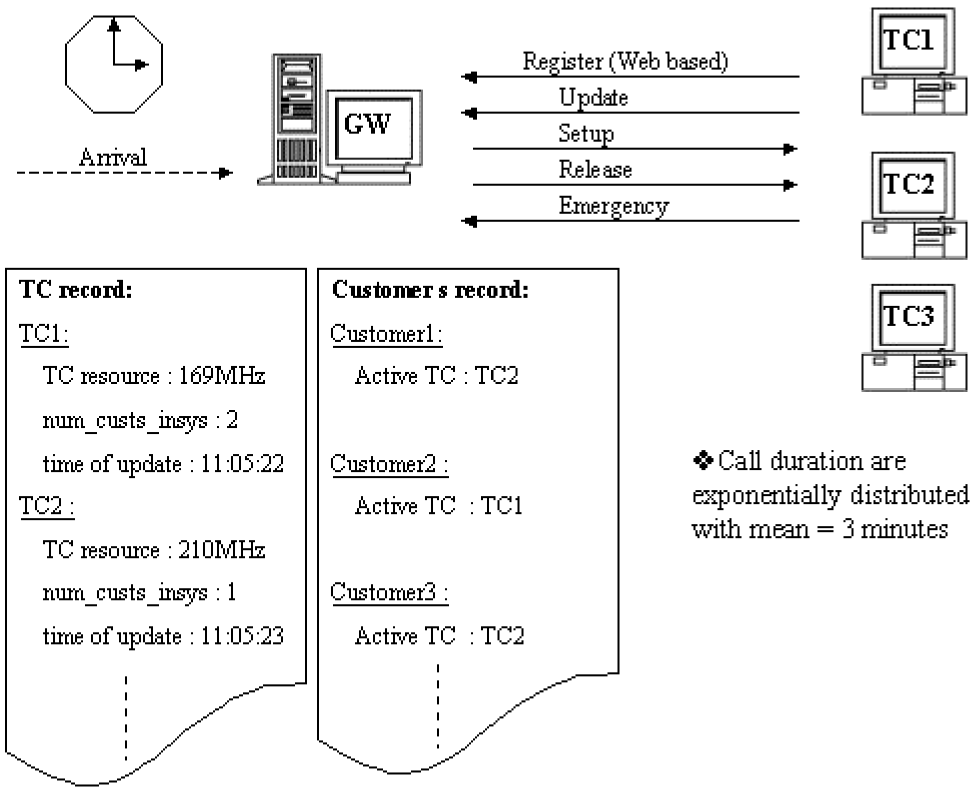
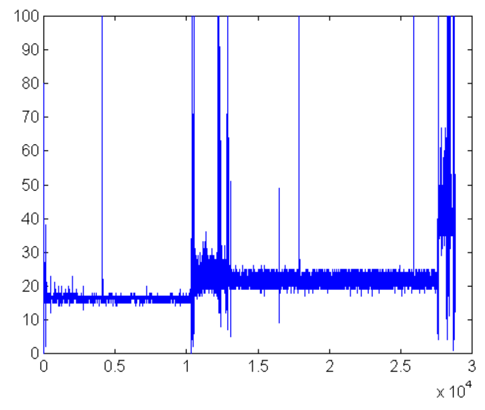
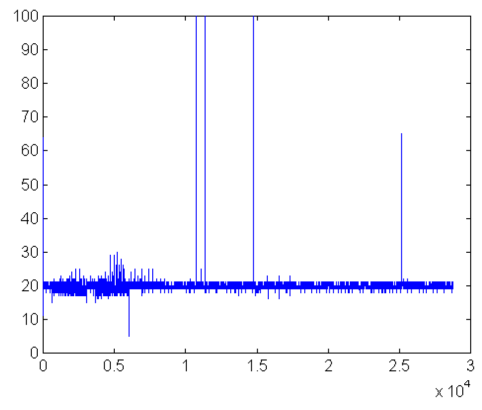
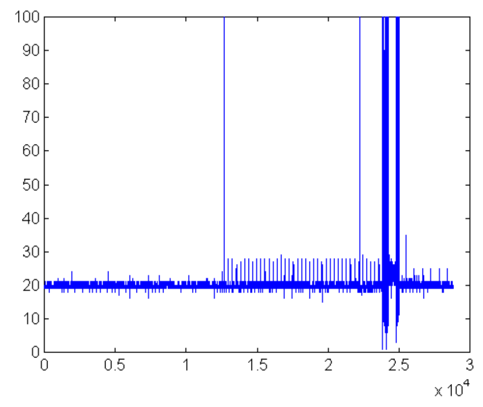


Fig. 10 Simulation Model


(a)


(b)


(c)

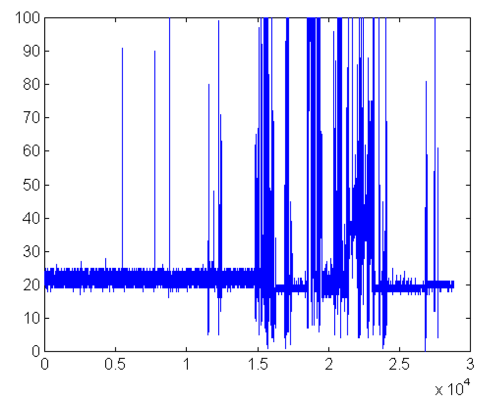

(d)

Fig. 11 CPU utilization of (a) gateway (b) proxy-TC1 (c) proxy-TC2 (d) proxy-TC3.

3.2 Performance indices

The definitions of blocked calls, dropped calls and handoff frequency are given below. We recognize a call as a blocked call, when the individual CPU resources of the gateway and all registered proxy-TCs are less than 25MIPS upon receipt of a new call. A dropped call occurs when the serving proxy-TC is occasionally running out of resources for at least one second and no other proxy-TC can take over this call. Therefore, the new call blocking probability and the active call dropping probability are:

$$P_b = \Pr[blocking] = \frac{\text{Number of blocked calls}}{\text{Total arrival calls}}$$

$$P_d = \Pr[droping] = \frac{\text{Number of dropped calls}}{\text{Total served calls}}.$$

The handoff frequency is defined as:

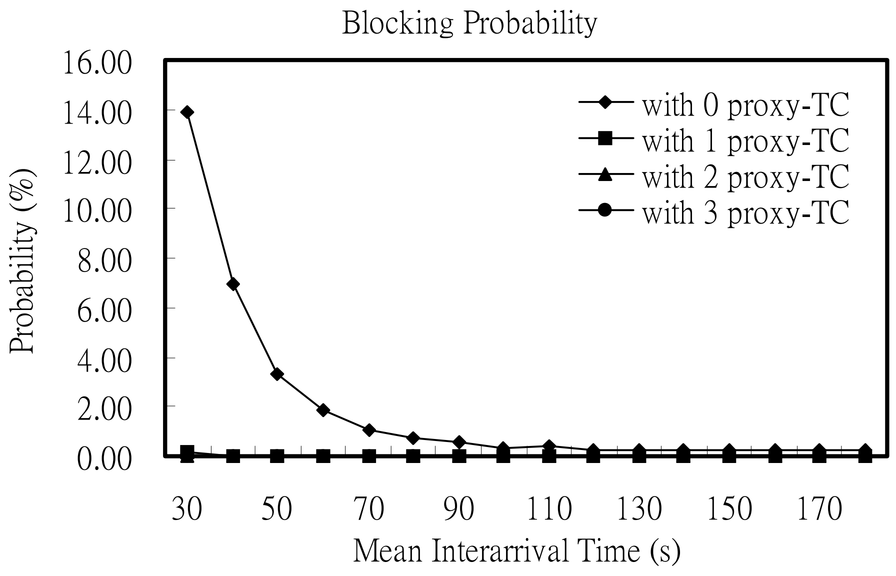
$$P_h = \Pr[Handoff] = \frac{\text{Number of handoff calls}}{\text{Total served calls}}.$$

3.3 Simulation results

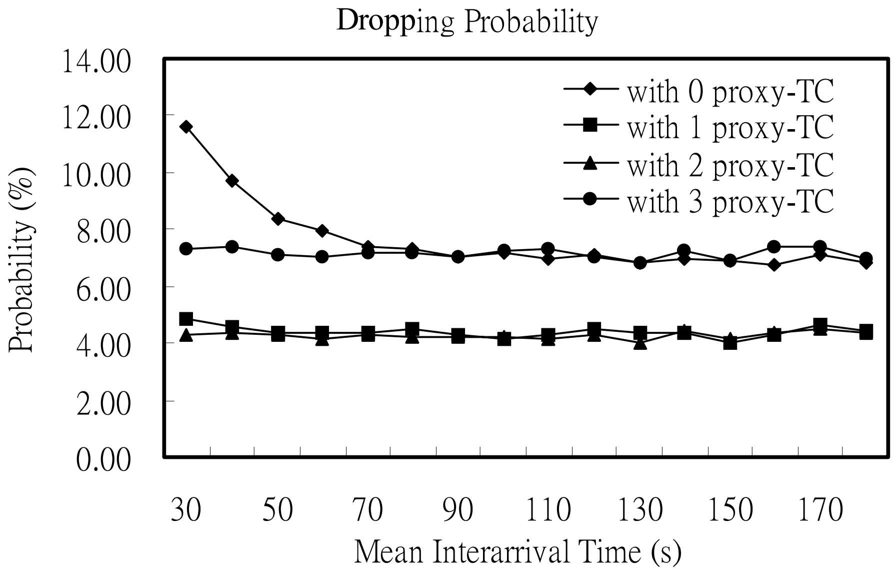
Fig. 12 shows the relation between the number of registered proxy-TCs and the probabilities of call blocking and call dropping, when no handoff mechanism is implemented. The first graph of Fig. 12 indicates that the new call blocking probability drastically decrease as the number of proxy-TCs increases under the same inter-arrival time, and as the number of the registered proxy-TCs reaches three, almost no calls are blocked. On the other hand, P_b decreases exponentially with respect to the inter-arrival time while the number of the registered proxy-TCs remains constant, which can be easily justified from queuing analysis. As to the active call dropping probability (P_d) plotted in (b) of Fig. 12, we observe that when the number of the registered proxy-TCs is within one and two, P_d is approximately half of its original quantity without proxy-TCs. However, when adding the third proxy-TC, we note that P_d surprisingly increases instead of decreases. This is in fact due to that the CPU utilization of the third proxy-TC is very much bursty in nature as shown in (d) of Fig. 11, and thereby a call being assigned to this proxy-TC suffers a high active call dropping probability, which in turns increases the overall active call dropping probability. We conclude that without handoff mechanism, P_d is highly dependent on the statistical nature of the CPU utilization on individual registered proxy-TC.

The previous simulation hints the necessity of introducing a handoff mechanism for maintaining a low P_d, which is substantiated by our second simulations shown in Fig. 13.

In the second simulation, we presume that the number of registered proxy-TCs is only one. We found that the handoff mechanism only slightly increases P_b when it is compared to the situation without handoff mechanism. This result is anticipated since the handoff mechanism keeps most of the dropped calls in the previous simulation alive, and hence, the available resource on the proxy-TC slightly decreases. The goodness of introducing the handoff mechanism is that P_d is drastically decreased. In short, the handoff mechanism can decrease a great amount of P_d at the expense of a little increase in P_b.

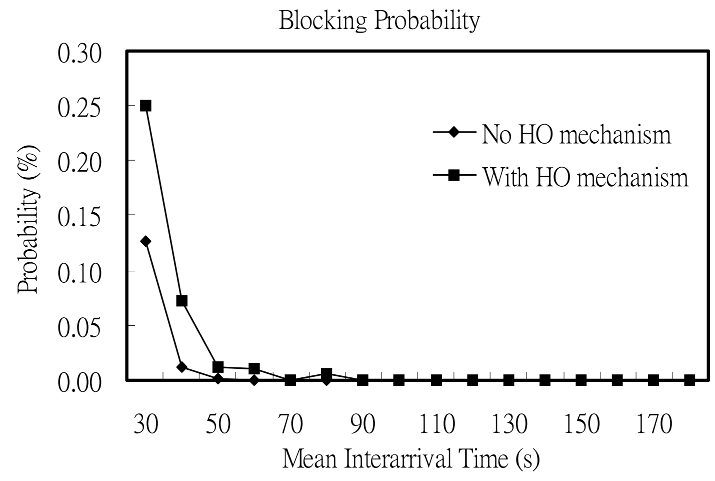


(a)

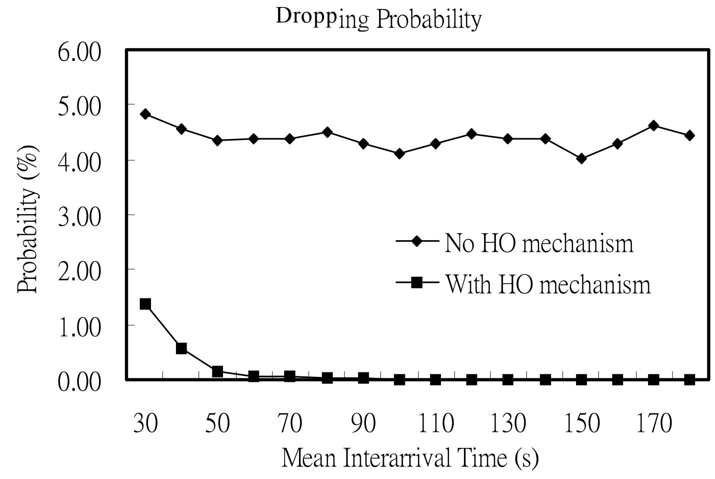


(b)

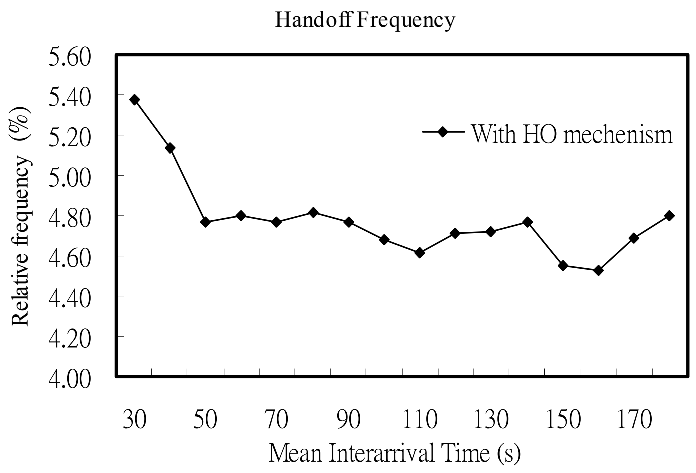
Fig. 12(a) The new call blocking probability and (b) the active call dropping probability without handoff mechanism.



(a)



(b)



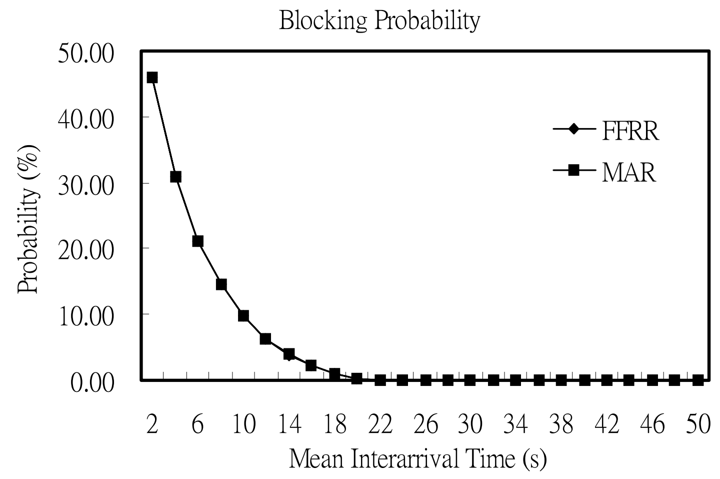
(c)

Fig. 13 (a) The new call blocking probability, (b) the active call dropping probability, and (c) the handoff frequency with handoff mechanism.

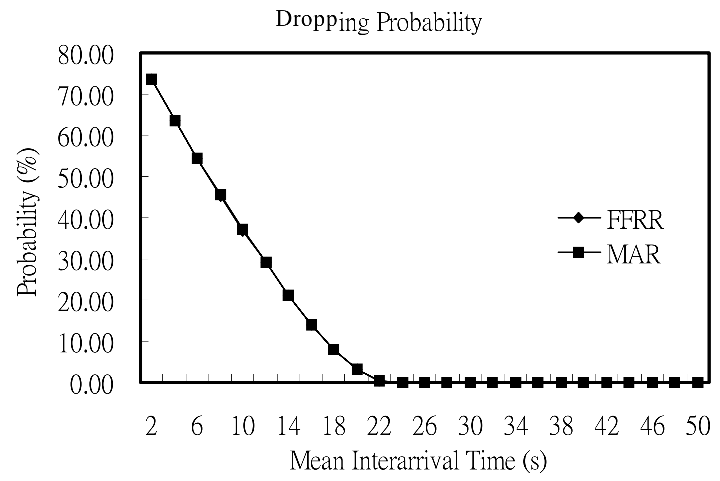
In the third simulation, we try to find another proxy-TC selection scheme to reduce the handoff frequency while the FFRR is adopted in the first two simulations. We propose the second scheme, maximum available resource (MAR), using this scheme the gateway looks for the proxy-TC, which has the maximum available resource among all proxy-TCs, to cooperate with. For evident distinction of result between the two proxy-TC selection schemes, ten proxy-TCs are involved in this simulation and the mean service time is extended to thirty minutes.

Fig. 14 shows the performances of the two proxy-TC selection schemes. From the observation of (a) and (b) of Fig. 14, we find that P_b and P_d are independent of the proxy-TC selection scheme when the handoff mechanism is adopted. The main difference of performance is the handoff frequency as shown in (c) of Fig. 14. We can easily recognize that the FFRR scheme performs better than the MAR scheme does due to the lower handoff frequency. We investigate and find the reason, described below. When the traffic load is higher and the MAR scheme is adopted, it is possible that each proxy-TC will receive certain amounts of transcoding service loads of calls. If there is a very busy proxy-TC, namely the CPU utilization of this proxy-TC is very much bursty in nature. And thus this proxy-TC will handoffs all calls served by it to other proxy-TCs more frequently. After finishing some job, the CPU utilization of this busy proxy-TC will calm down in a short period of time and will be the one with maximum available resource at some moment. When a new call arrivals, this call will be assigned to this busy proxy-TC more likely and suffers higher handoff frequency. But using FFRR, a new call will be assigned to every proxy-TC equally in probability so that the handoff frequency will be lower.

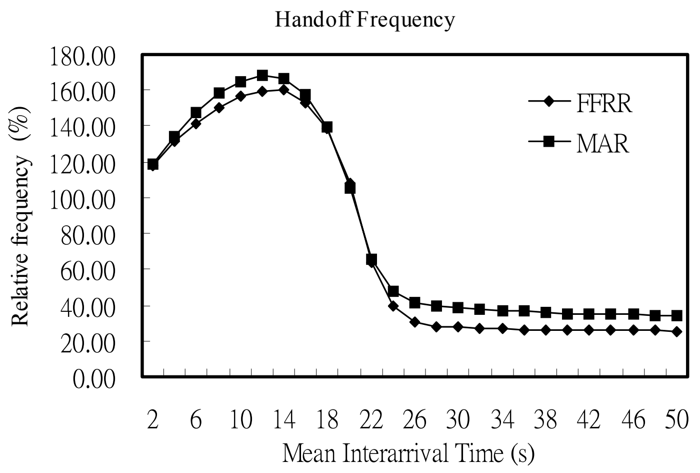
This simulation tells us that a good method to predict the CPU utilization will result in low handoff frequency and reduce the signaling traffic.



(a)



(b)



(c)

Fig. 14 (a) The new call blocking probability, (b) the active call dropping probability, and (c) the handoff frequency with handoff mechanism.

Chapter 4

Conclusions and Future Work

In this thesis, we propose a novel architecture of a software-based gateway with proxy-TCs operating on PC platforms. Simulation results show that it may provide a cost-effective alternative for the gateway implementation. We summarize that the proposed architecture achieves higher capacity and lower blocking probability with respect to the arrival calls, and the employment of the handoff mechanism leads to a drastic decrease in the active call dropping probability at the expense of a little increase in the new call blocking probability. Along this research direction, an interesting future work will be to find a method to predict the CPU utilization, as well as a good call placement approach to reduce the handoff frequency.

References

[1] ITU-T, Recommendation H.323, "Packet-based Multimedia Communication," Nov. 2000.

[2] ITU-T, Recommendation H.245 "Control Protocol for Multimedia Communication," Nov. 2000.

[3] ITU-T, Recommendation H.225.0, "Media Stream Packetization and Synchronization on Non-Guaranteed Quality of Service LANs," Mar. 2001.

[4] ITU-T, Recommendation G.723.1, "Dual Rate Speech Codec for Multimedia Telecommunications Transmitting at 6.4 and 5.3 kbit/s," Mar. 1996.

[5] ITU-T, Recommendation G.729, "Speech Codec for Multimedia Telecommunications Transmitting at 8/13 kbit/s," Mar. 1996.

[6] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: A Transport Protocol for Real-time Application," *IETF RFC 1889*, 1996.

[7] A. M. Grilo, P. M. Carvalho, L. M. Medeiros and M. S. Nunes, "VTOA/VoIP/ISDN Telephony Gateway," in proceedings of *ICATM '99*, pp. 230-235, 1999.

[8] D. Rizzetto and C. Catania, "A Voice over IP Service Architecture for Integrated Communication," *IEEE Network*, vol. 13, pp. 34-40, May-June 1999.

[9] M. Hamdi, O. Verscheure, J.-P. Hubaux, I. Dalgic and P. Wang, "Voice Service Interworking for PSTN and IP Networks," *IEEE Communications Magazine*, vol. 37, pp. 104-111, May 1999.