An H.323 Gatekeeper Prototype: Design, Implementation, and Performance Analysis

Cheng-Yue Chang, Ming-Syan Chen, *Fellow, IEEE*, and Pai-Han Huang

***Abstract*—In recent years, Internet telephony or voice-over IP is attracting an increasing amount of attention for the reason that it has the potential to significantly reduce the cost of long-distance voice communication. To provide the interoperability between different manufacturers and vendors, ITU-T is actively drawing up the standard for the Internet telephony as ITU-T Recommendation H.323. While H.323 is continuously revised, there are, however, still many important issues needed to be further discussed. In particular, the bandwidth management, address translation, and intelligent call signaling/routing, which are the major functions of an H.323 gatekeeper, have not yet been fully addressed. In view of this, we develop in this paper an H.323 gatekeeper prototype which fully complies with the H.323 recommendation, and explicitly illustrate the usefulness of this prototype by devising two improved call-routing methods to deal with the load-balancing problem among the gatekeepers. As a matter of fact, the existing call-routing method used in the current version of the H.323 is very primitive and will cause the loads of the gatekeepers unbalanced. Using the gatekeeper prototype devised, two improved call-routing methods are proposed based on *Multiple-Registration with Multicasted Location Request (MRML)* and *Multiple-Registration with Hierarchical Database (MRHD)*. To evaluate the performance improvement by these two load-balancing methods, we design an event-driven simulator and conduct some experiments on it. We focus on two performance metrics, *call-blocking rate* and *channel utilization*, which are both sensitive to the load distribution. The experimental results show that the *call-blocking ratio* and *channel utilization* are optimized with the proposed load-balancing methods. It is also observed from our empirical results that the *call-blocking ratio* is more sensitive to the traffic load whereas the *channel utilization* is more sensitive to the duration of a call.**

***Index Terms*—Communication system signaling, ITU, Protocol, teleconferencing, traffic control (communication).**

I. Introduction

In RECENT years, Internet telephony or voice-over IP is attracting an increasing amount of attention for the reason that it has the potential to significantly reduce the cost of long-distance voice communication [10], [24]. Architectures for real-time audiovisual communication are proposed in [2], [8], [11], [17], [19], [25], [30], [31], and [34] to provide telephony services over packet-based networks. Among others, ITU-T Recommendation H.323 [17], [22], [32] is the primary standard for the Internet telephony. The H.323, first approved in 1996 by ITU Study Group 15, specifies the technical requirements for audio, video, and data communication across packet-based networks. In essence, the H.323 is an umbrella recommendation which includes the specifications of the system components [i.e., terminals, gateways, gatekeepers and multipoint control units (MCUs)], audio/video codecs [4], [12], [15], [16], call-signaling/control protocols [14], [18], and network interface [13]. By complying with the H.323, Internet telephony products and applications from multiple vendors can interoperate with one another, allowing users of communicating with each other without concern for compatibility.

While the H.323 is continuously revised, there are, however, still many important issues needed to be further discussed [20], [23]. In particular, the bandwidth management, address translation, and intelligent call signaling/routing, which are the major functions of an H.323 gatekeeper, have not yet been fully addressed. In view of this, it is necessary to build a prototype of an H.323 gatekeeper for further experiments and developments. In this study, we explore several methodologies for the implementation of an H.323 gatekeeper prototype, and based on these methodologies, devise the corresponding architecture and implement an H.323 gatekeeper prototype. As will be explained in details later, the gatekeeper prototype built consists of three software modules, i.e., the registration, admission, and status (RAS) module, the call-routing module and the registration database. Using an H.323 protocol stack [1], we have successfully implemented the H.323 gatekeeper prototype and validated the system built by having fluent two-way communication according to the call-routing scenarios specified in the H.323.

It is worth mentioning that the H.323 gatekeeper prototype we devised is very versatile and can be easily to incorporate various features for further experiments. Specifically, we extend our gatekeeper prototype to deal with the load-balancing problem among the gatekeepers by incorporating two improved call-routing methods into it. As will be explained later, the existing call-routing method used in the current version of the H.323 is very primitive and will cause the loads of the gatekeepers unbalanced (See Section III-A). Using the gatekeeper prototype devised, two improved call-routing methods are proposed based on *Multiple-Registration with Multicasted Location Request (MRML)* and *Multiple-Registration with Hierarchical Database (MRHD)*. We will describe the design principles and discuss the effects of these two methods upon the system performance later. In addition, to evaluate the performance improvement by our load-balancing methods, we design an event-driven simulator and conduct some experiments on it. The system model of the simulation is designed to reflect the load-balancing problem among the gatekeepers. We focus on

Manuscript received April 11, 2001; revised August 26, 2002. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Chung-Sheng Li.

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. (e-mail: cychang@arbor.ee.ntu.edu.tw; mschen@cc.ee.ntu.edu.tw; bhhuang@arbor.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TMM.2004.834857

two performance metrics, *call-blocking ratio* and *channel utilization*, which are both sensitive to the distribution of the load. The experimental results show that the *call-blocking ratio* and *channel utilization* can be significantly improved by the proposed load-balancing methods by 34.2% and 13.1%, respectively. It is also observed from our empirical results that the *call-blocking ratio* is more sensitive to the traffic load whereas the *channel utilization* is more sensitive to the duration of a call.

The contributions of this paper are twofold. We not only devise an H.323 gatekeeper prototype for further developments and experiments, but also, using the gatekeeper prototype devised, propose two improved call-routing methods to tackle the load-balancing problem among gatekeepers. To the best of our knowledge, despite of their importance, the design, implementation and performance analysis of an H.323 gatekeeper were little explored in the literature thus far. The fast growth of IP-based applications justifies the timeliness and importance of this study. Note that another competing architecture proposed by IETF also plays a decisive role and has becomes increasingly important in recent years [30]. IETF's architecture addresses more on the signaling protocol, SIP [9]. Accompanying this signaling protocol, several auxiliary protocols (e.g., RTP [28], RTSP [27], and TRIP [26]) are adopted as the building blocks to provide the infrastructure for the Internet telephony. We refer interested readers to [7], and [29] for the comparison of the H.323 and IETF's architecture and to [6], [21], [33] for the extensions to SIP protocol.

The rest of this paper is organized as follows. The methodologies for building an H.323 gatekeeper prototype is introduced in Section II. In Section III, we explain the design principles of the two improved call-routing methods to deal with the load-balancing problem among the gatekeepers. The performance of the extended gatekeeper prototype is evaluated and analyzed in Section IV. We conclude this paper with Section V.

II. Building an H.323 Gatekeeper Prototype

In this section, we devise methodologies for building an H.323 gatekeeper prototype. The prototype devised consists of three software modules. The RAS module is designed in Section II-A. The call-routing module is explained in Section II-B. The registration database is described in Section II-C. Section II-D depicts the corresponding architecture for the H.323 gatekeeper prototype.

A. RAS Module Design

RAS messages are used in the H.323 to exchange information between H.323 gatekeepers and the other endpoints (i.e., terminals, gateways, and MCUs). The details of the RAS messages are defined in the ITU-T Recommendation H.225.0 [13]. An H.323 gatekeeper utilizes RAS messages to provide mandatory functions (i.e., admission control, address translation, bandwidth control and zone management) as specified in the H.323. To facilitate our later discussion in Section III, we focus on the designs of the endpoint registration and the call-admission processes in this subsection.

1) Endpoint Registration: The H.323 gatekeeper only provides services for the endpoints who have registered with it. In other words, an endpoint has to register with a gatekeeper before it is able to take advantages of the services provided. To determine which gatekeeper to register with, an H.323 endpoint shall carry out the gatekeeper discovery process first. As specified in the H.323, gatekeeper discovery may be done manually or automatically. Since our gatekeeper prototype is devised for empirical studies, all the test endpoints are manually configured in such a way that their mapping with gatekeepers is known *a priori*.

After the gatekeeper discovery process, the endpoint knows which gatekeeper to be associated with, and then sends a registration request (RRQ) message to that gatekeeper. Upon receiving the RRQ messages from the endpoints, our gatekeeper prototype authorizes the registration based on the following rules:

1) the endpoint is our own test program;
2) the endpoint resides in the same subnet as our gatekeeper prototype;
3) the number of the registered endpoints is smaller than the threshold of the gatekeeper's capacity.

If the above rules are satisfied, our gatekeeper prototype confirms the registration by sending an registration confirm (RCF) message back to the endpoint, and keeps the registration information in the local database (which will be explained later in Section II-C). Note that the authorization rules of the registration can be customized by the gatekeeper administrator to facilitate the management of the registered endpoints. For example, we may limit the vendors or manufacturers of the endpoints to the trusted ones.

2) Call Admission: As specified in the H.323, the endpoints shall get the admission from their registered gatekeepers before they initiate or answer a call. In our design of the gatekeeper prototype, the endpoints will have the qualifications for call admission if they have registered at our gatekeeper and the load of the gatekeeper is smaller than the administrator-defined threshold. However, the request for the call admission would still fail if the address translation process did not succeed. Note that the H.323 allows three kinds of addressing methods (i.e., the E.164 [3] number, the H.323 ID and the transport address) for the calling party to indicate the called party. To establish calls with E.164 numbers or H.323 IDs, the H.323 gatekeeper has to first translate these alias addresses to the transport addresses. If the called party has registered at the same gatekeeper as the calling party, the gatekeeper can finish the address translation by looking up the local registration database. If not, the gatekeeper would have to multicast LRQ messages to all other gatekeepers and wait for the result of the address resolution from them. If no gatekeeper replies, the address translation will fail, and the request for the call admission will be rejected. In this case, our gatekeeper will send an admission reject (ARJ) message back to the calling party. The algorithm for our gatekeeper prototype to authorize an admission request is listed in the Appendix for interested readers.

The admission control is very critical for the gatekeeper to manage the resource (including the network bandwidth, CPU

time, memory, etc.) owned by it. As noted in the previous subsection, the qualifications for call admission could be also customized by the gatekeeper administrator to facilitate the management of the resource owned by the gatekeeper. For example, we can extend our gatekeeper built to work with other mechanisms (such as bandwidth management or call-management modules).

B. Call-Routing Module Design

An H.323 call can be established directly between the endpoints or routed via the H.323 gatekeepers. To deal with all kinds of scenarios effectively, we unify them into one *call state machine*, and design a specific data structure to keep track of the state and the routing information for each call. The *call state machine* is developed based on the methodology described in [5]. The data structure designed contains the following information: 1) the state of the call (i.e., idle, call offering, call connected, or call disconnected); 2) the duration of the call; 3) the transport address of the call routed from; 4) the transport address of the call routed to; 5) the call signaling address of the calling party; and 6) the call-signaling address of the called party. Note that the transport address of the call routed from may be different from the call-signaling address of the calling party. The transport address of the call routed from is the transport address of the last intermediate node along the routing path. It can be either the transport address of the calling party or the transport address of some gatekeeper along the routing path. If the transport address of the call routed from is the same as the call-signaling address of the calling party, it means that no other gatekeeper is between the calling party and this gatekeeper. For the same reason, the transport address of the call routed to may be different from the call-signaling address of the called party.

The primitive method in the current version of the H.323 is by sending the LRQ message to a specific *Gatekeeper's RAS Channel TSAP Identifier*, or by multicasting the LRQ message to the gatekeeper's well-known discovery multicast address. The gatekeeper which the requested endpoint is registered with shall respond the LCF message containing the contact information of the endpoint or the endpoint's gatekeeper. It is noted that such a routing method suffers from several drawbacks. First, it may cause the loads of the gatekeepers unbalanced. In addition, multicasting LRQ messages will cause considerable bandwidth consumption. To eliminate these problems, we design two improved call-routing methods and incorporate them into the gatekeeper prototype built.

C. Registration Database

As mentioned in Section II-A, the H.323 gatekeeper shall keep the registration information to provide address translation, call admission, and other mandatory functions defined in the H.323. To this end, the schema of the registration database is designed as shown in Fig. 1.

The attributes of *call-signaling address* and *call-signaling port* are used to keep the transport address of the endpoint. Since every endpoint is associated with only one transport address, we set the combination of these two attributes as the primary key to identify an endpoint. The attribute of the *RAS Port* is where

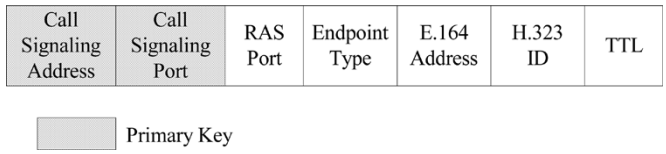


Fig. 1. Schema of the relational database.

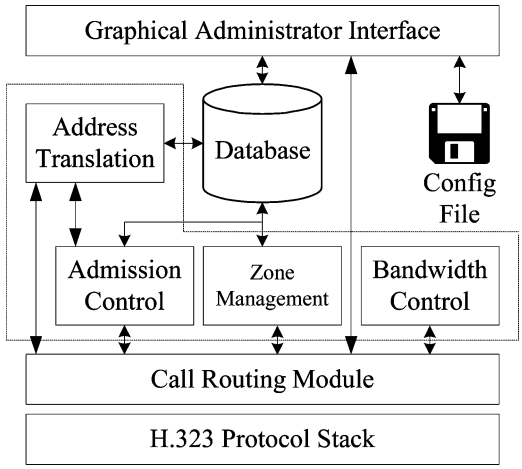


Fig. 2. Architecture of the H.323 gatekeeper prototype.

the endpoint keeps listening for incoming RAS messages. *Endpoint Type* indicates the type of the endpoints (which may be a terminal, gateway, or MCU). The attributes of *E.164* and *H.323 ID* are the E.164 number and the H.323 ID of the endpoint, respectively. *Time to live* (TTL) records the remaining time of the valid registration. With this information, our prototype is able to provide the mandatory functions of an H.323 gatekeeper, including address translation, zone management, and admission control.

D. Architecture of the H.323 Gatekeeper Prototype

We then construct the corresponding architecture for our H.323 gatekeeper prototype. As shown in Fig. 2, the architecture is built on top of the H.323 protocol stack and composed of five components, i.e.,: 1) the call-routing module; 2) the RAS module; 3) the database of the registration information; 4) the graphical administrator interface; and 5) the configuration file. The arrows in Fig. 2 stand for the data flows among these components. It is noted that all incoming messages (i.e., RAS messages, Q.931 call-signaling messages and H.245 call-control messages) are first processed by the call-routing module. For the call-related messages, the call-routing module triggers the call state machine (described in Section II-B) and updates the call state and the call-routing information stored in the data structure of the corresponding call. For the other messages, the call-routing module passes them directly to the RAS module for further processing. The RAS module is composed of four submodules which implement the four mandatory functions of an H.323 gatekeeper, i.e., zone management, admission control, bandwidth control, and address translation. The graphical administrator interface is used by the system administrator to monitor the status of each ongoing call, manipulate the database and change the system configuration. The configuration file

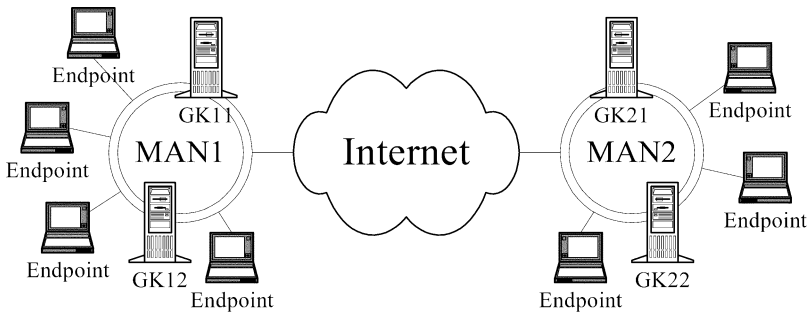


Fig. 3. Illustrative scenario of the load-balancing problem.

stores the system parameters such as system capacity, existing subnet, and so on.

Our H.323 gatekeeper prototype and the H.323 terminal program (i.e., which is used to test our gatekeeper) are developed on a PC with Pentium III 450 CPU and 128-MB RAM. The operating system we adopt is Windows NT version 4.0 with service pack 6a. We utilize an H.323 protocol stack to implement our gatekeeper. The network interface to the Internet is 100-base-T Ethernet card. We use Microsoft Visual C++ 6.0 as the developing tool.

III. Improved H.323 Call Routing Methods

Based on the gatekeeper prototype built in Section II, we devise two improved H.323 call-routing methods in this section to deal with the load-balancing problem among the gatekeepers. These two methods are *MRML* and *MRHD*. We introduce the load-balancing problem in Section III-A. The proposed *MRML* and *MRHD* methods are explained in Sections III-B and III-C, respectively. The effects of these two methods upon the system performance is discussed in Section III-D.

A. Load-Balancing Problem

As pointed out in Section II-B, the primitive call-routing method of the H.323 will cause the loads of the gatekeepers unbalanced. One reason is that the H.323 only allows the endpoint of registering with one gatekeeper but does not provide any mechanism for load balancing. Hence, if the endpoints are not equally distributed among the gatekeepers, the loads of the gatekeepers will be unbalanced. Consider the scenario shown in Fig. 3. Assume that two gatekeepers, GK11 and GK12, are available for the endpoints on MAN1, and two gatekeepers, GK21 and GK22, are available for the endpoints on MAN2 after the gatekeeper discovery process. Suppose that all the endpoints on MAN1 choose GK11 to register with. In this case, every call from/to the endpoints on MAN1 will be routed via GK11. As a result, GK11 will be heavily loaded whereas GK12 will be idle all the time. The situation is similar if all the endpoints on MAN2 choose GK22 to register with.

Second, the load-balancing problem will occur even though the endpoints are equally distributed among the gatekeepers. For example, assume that both GK11 and GK12 have two registered endpoints on MAN1. However, if the registered endpoints of GK11 make more calls than the registered endpoints of GK12, the load of GK11 will be heavier than that of GK12. Thus, even though the endpoints are equally distributed among the gatekeepers, the load-balancing problem will still occur. Furthermore, the gatekeepers are likely to be different from one to another in computing power, disk storage and bandwidth connectivity. If the endpoints are not distributed in accordance with the gatekeepers' capabilities, the load-balancing problem also exists. In view of this, we shall devise improved call-routing methods to deal with the load-balancing problem among the gatekeepers.

B. MRML

The first improved call-routing method devised is based on *MRML*. Conventionally, each terminal always chooses "one" gatekeeper to register with after the gatekeeper discovery process in the H.323. In our method, we design our terminal program to allow of registering with more than one gatekeeper. Explicitly, after the endpoint multicasts gatekeeper request (GRQ) messages to the gatekeeper's well-known discovery multicast address, the endpoint will probably receive more than one gatekeeper confirm (GCF) message replied from the available gatekeeper. Instead of choosing one gatekeeper to register with, we allow our terminal program of registering with all the gatekeepers who have responded with GCF messages. The motivation of such *multiple registration* is that we would like the endpoint to have the opportunities of selecting the gatekeeper with the lightest load for routing its call.

Next, recall that the endpoints shall get the admission from their registered gatekeepers before they initiate or answer a call. In our method, having registered with multiple gatekeepers, an endpoint will send the ARQ messages to all the registered gatekeepers before initiating or answering a call. In the case that the endpoint wants to initiate a call, the registered gatekeepers of the caller will authorize the admission request as described in Section II-A.2 with the following modifications. First, to learn about the load conditions of the called gatekeepers (i.e., the registered gatekeepers of the callee), the calling gatekeepers (i.e., the registered gatekeepers of the caller) will multicast LRQ messages to all other gatekeepers no matter which type is the indicated callee's address in the ARQ messages. After the called gatekeepers receives the LRQ messages, they will respond LCF messages with their load conditions piggybacked. As such, the calling gatekeepers will know the load condition of each called gatekeepers and thus be able to choose a called gatekeeper with the lightest load to route their calls later.

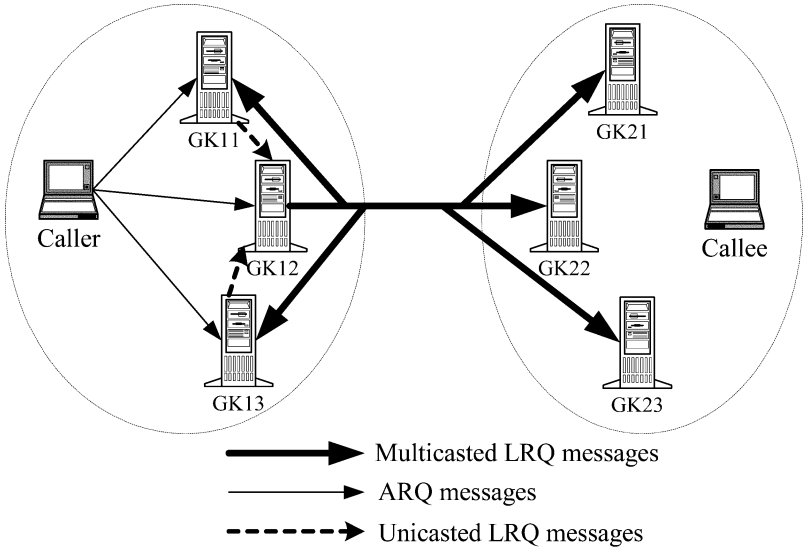


Fig. 4. Scenarios of admission request and LRQ processes in the *MRML* method.

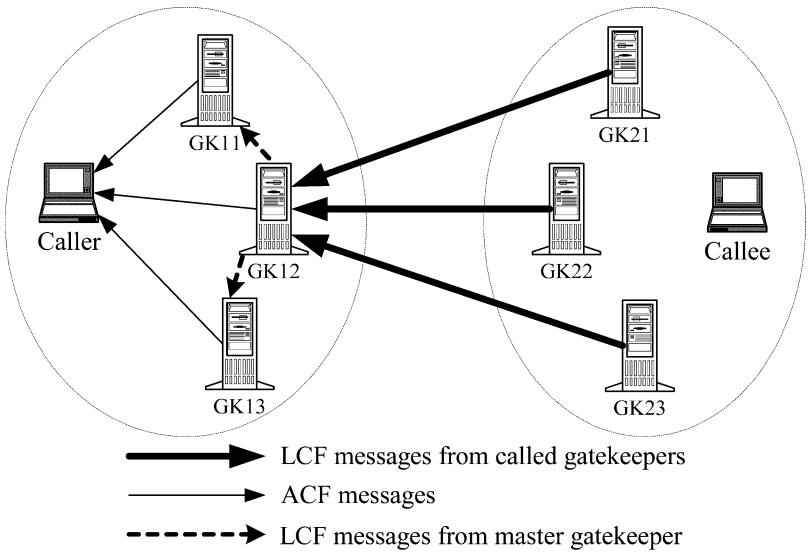


Fig. 5. Scenarios of LCF and admission confirm processes in the *MRML* method.

Note, however, that such an approach would consume much bandwidth if every calling gatekeeper multicasts the LRQ messages once to all other gatekeepers and every called gatekeeper sends back the LCF messages to all the calling gatekeepers. To remedy this drawback, our terminal program is designed to possess the capability of specifying one calling gatekeeper as the master and leaving the other calling gatekeepers as the slaves in the ARQ messages. If the calling gatekeeper is specified as a master gatekeeper, it shall multicast the LRQ messages to all other gatekeepers to inquire the callee's transport address and the load conditions of the called gatekeepers. If the calling gatekeeper is specified as a slave gatekeeper, it shall send the LRQ message to the master gatekeeper and waits for the reply.

Fig. 4 illustrates the scenarios of the admission request and location request (LRQ) processes in the *MRML* method. Assume that GK11, GK12, and GK13 are the caller's gatekeepers, whereas GK21, GK22 and GK23 are the callee's gatekeepers. To initiate a call, the caller sends the ARQ messages to all the registered gatekeepers (i.e., GK11, GK12, and GK13) and specifies GK12 as the master gatekeeper and GK11 and GK13 as the slave gatekeepers for example. After receiving the ARQ message from the caller, GK12 will then multicast the LRQ messages to all other gatekeepers to inquire callee's transport address and the load conditions of the called gatekeepers. Since GK11 and GK13 are the slaves, they will simply send the LRQ messages to GK12 and wait for the replies. Note that when GK12 receives the LRQ messages from GK11 and GK13, it will first examine the call IDs associated with those messages to determine whether the LRQ messages are sent from the slave gatekeepers or they are "real" LRQs from other gatekeepers. If the LRQ messages are sent from the slave gatekeepers, GK12 will not respond to them until it obtains the replies to the LRQ messages multicasted to all other gatekeepers. Otherwise, GK12 will respond to them accordingly.

Fig. 5 shows the scenarios of the LCF and admission confirm processes in the *MRML* method. Following the above ex-

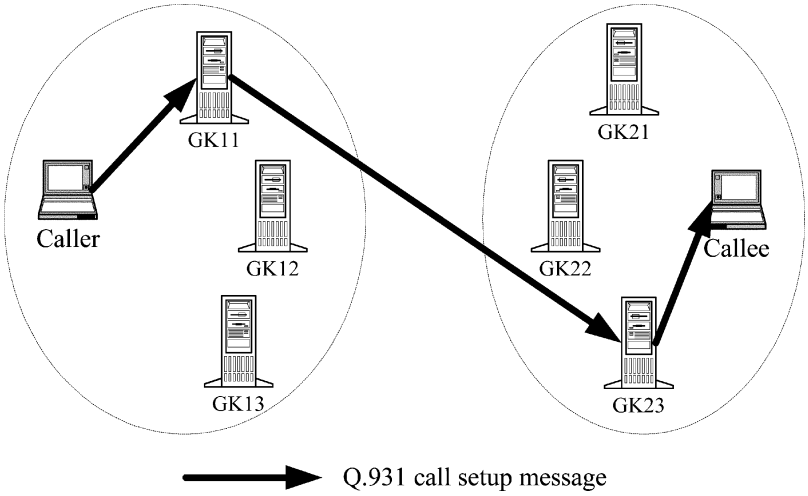


Fig. 6. Routing path of the Q.931 call-setup message.

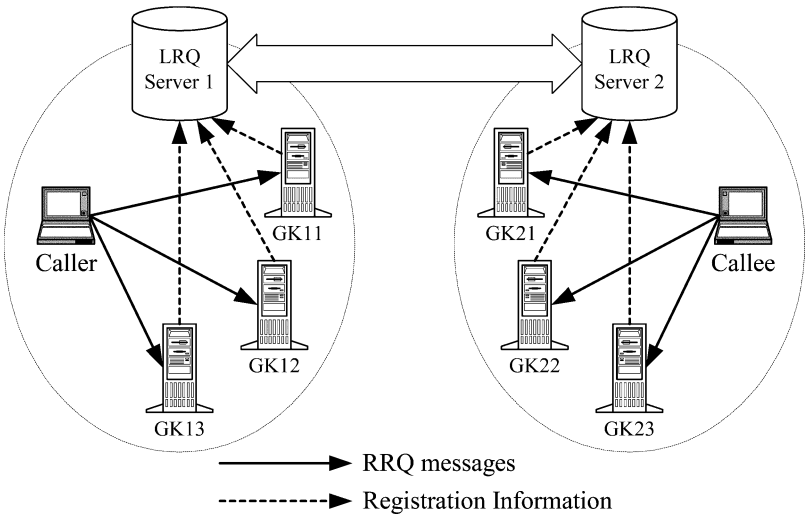


Fig. 7. Proposed architecture of the MRHD method.

ample, since GK21, GK22, and GK23 are the callee's gatekeepers, they will return LCF messages indicating the transport address of the callee and piggyback the load conditions of themselves to GK12. Thereafter, GK12 will know the callee's transport address and the lightest gatekeeper in the called party side. Assume that the lightest gatekeeper in the called party side is GK23. GK12 will then return LCF messages containing the transport address of the callee and the information that GK23 is the lightest gatekeeper in the called party side to GK11 and GK13. Finally, GK11, GK12, and GK13 will return the ACF messages with their load conditions piggybacked to the caller, so that the caller will be able to determine which gatekeeper in the calling party side is of the lightest load and then set up the call via the lightest calling gatekeeper. Suppose that GK11 is the lightest gatekeeper in the calling party side. The Q.931 call-setup message will be sent from the caller, routed through GK11 and GK23, and then forwarded to the callee, as shown in Fig. 6.

It is noted that, in this *MRML* method, the dissemination of the load conditions is totally based on the piggybacking technique. These piggybacked load conditions are stuffed in the non-standard parameter reserved by the H.323 protocol for testing purpose. Hence, the improved *MRML* routing method not only achieves the goal of load balancing, but also remains compatible with the conventional routing method of the H.323.

C. MRHD

The second call-routing method devised for load balancing is based on *MRHD*. The concept of *multiple registration* in *MRHD* follows the one in *MRML*. However, we notice that multicasting LRQ messages for address translation are costly in *MRML*, especially when many gatekeepers are loosely distributed on the Internet. In addition, multicast channel is an unreliable channel which does not guarantee that the LRQ messages be sent to all gatekeepers. In view of this, we employ the concept of a *hierarchical database* and propose an extended architecture for the purposes of address translation and load balancing as well.

Fig. 7 shows the extended architecture we propose. A two-level hierarchical database is designed for the address translation and load balancing. The first level of the hierarchical database is the local registration database in each gatekeeper, and the second level of the hierarchical database is the regional *LRQ*

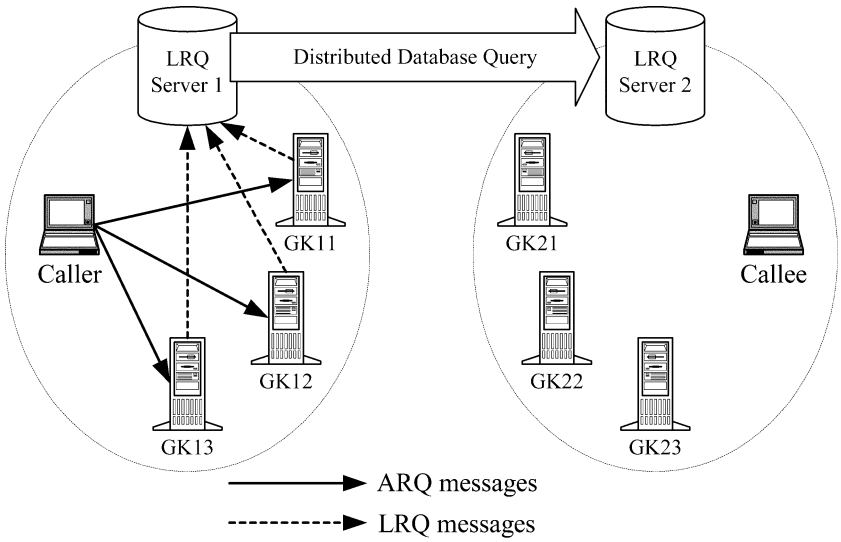


Fig. 8. Scenarios of the admission request and LRQ in the *MRHD* method.

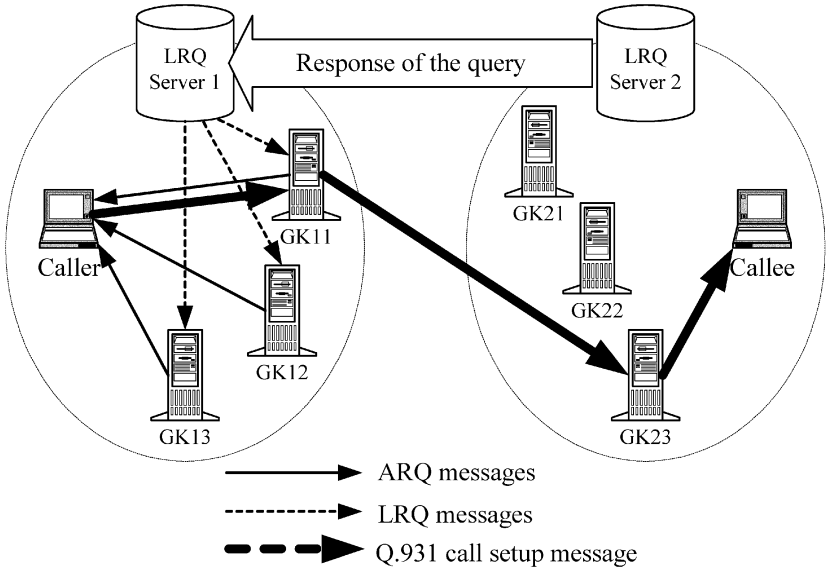


Fig. 9. Scenarios of the LCF and admission confirm in the *MRHD* method.

server which covers all registration information of the gatekeepers residing in that region. When one endpoint attempts to register with the gatekeeper, the gatekeeper will keep a version of the registration information in its local database and forward a version to the *LRQ server* for record. Similarly, when the endpoint attempts to un-register with the gatekeeper, the gatekeeper will not only delete the registration information in its local database but also send a request to the *LRQ server* for deleting the corresponding record. Furthermore, to provide the mechanism of load balancing, the *LRQ server* also keeps track of the load condition of each gatekeeper. Whenever a message is sent to the *LRQ server*, the load condition of the source gatekeeper will be updated with the piggybacked information.

Based on the proposed architecture, the *MRHD* method works as the scenarios shown in Fig. 8 and Fig. 9. When initiating a call, the caller will send the ARQ messages to all the registered gatekeepers (i.e., GK11, GK12, and GK13). To acquire the callee's transport address and the load conditions of the called gatekeepers, the calling gatekeepers GK11, GK12, and GK13 will send the LRQ messages to the LRQ Server 1, no matter which type is the indicated callee's address in the ARQ messages. Since the called gatekeepers are not within the resident region of the LRQ Server 1, the LRQ Server 1 will send a distributed database query message to retrieve the callee's transport address and the load conditions of the called gatekeepers. After the LRQ Server 1 receives the response from the LRQ Server 2, it will return the LCF messages with the load conditions of the called gatekeepers piggybacked to GK11, GK12, and GK13. Therefore, the calling gatekeepers will know which gatekeeper in the called party side is of the lightest load and will route the call to it later. Finally, GK11, GK12, and GK13 will return the ACF messages with their load conditions piggybacked to the caller, so that the caller will be able to determine which gatekeeper in the calling party side is of the lightest load and then set up the call via the lightest calling gatekeeper. Assume that GK11 and GK23 are the lightest gatekeepers in the calling and

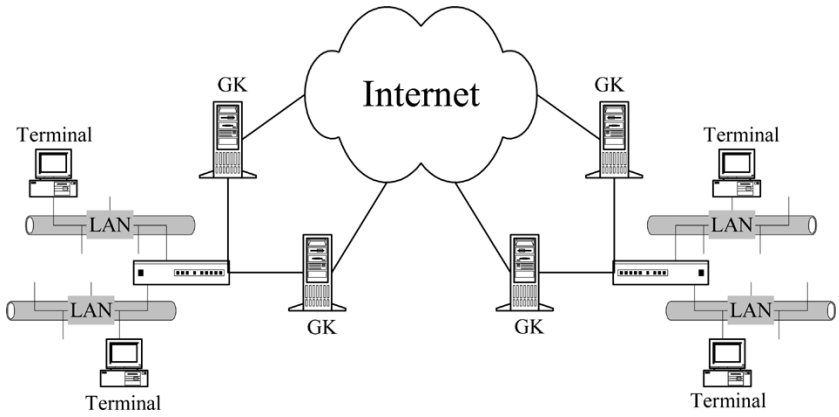


Fig. 10. System model of the simulation.

called party sides, respectively. As a result, the Q.931 call-setup message will be sent from the caller, routed through GK11 and GK23, and then forwarded to the callee, as shown in Fig. 9.

D. Discussion

With the help of the two improved call-routing methods (i.e., *MRML* and *MRHD*), we can obtain better performance than the circumstance without using them. First, the technique of *multiple registration* allows an endpoint of registering with more than one gatekeeper. The potential benefit of this feature is that a terminal will be served by the gatekeeper with the lightest load and thus gain better quality of service. The extra cost for this technique is only on slightly increased storage capacity for more registration entries. Such cost is deemed negligible in view of the huge capacity of current hard drives.

Second, we piggyback the load condition of each gatekeeper in the ACF and LCF messages. This technique allows the caller of choosing a calling gatekeeper with the lightest load to set up a call and allows the calling gatekeeper of choosing a called gatekeeper with the lightest load to route the call. Clearly, this feature is able to achieve the goal of load balancing among the gatekeepers and in turn accept more users to initiate video-conferences (as will be shown in Section IV). Note that the extra data piggybacked in the ACF or LCF message takes only 4 bytes, resulting in very small overhead.

In the *MRHD* method, we propose a two-level hierarchical database to reduce the network traffic incurred by multicasting LRQ messages. This method not only solves the load-balancing problem among the gatekeepers but also improve the address translation efficiency, especially when all gatekeepers are loosely distributed on the Internet. With the H.323 gatekeeper prototype built, we can devise customized functions to achieve the goal of a high quality of service and low transmission cost.

IV. Performance Analysis

We experimentally compare the performance of an H.323 gatekeeper under the circumstances with and without the load-balancing methods. The system model is introduced in Section IV-A. The experimental results are discussed and analyzed in Section IV-B.

A. System Model

In order to evaluate the performance of the improved call-routing methods described in Section III, we develop a system model as depicted in Fig. 10. This model is designed to reflect the load-balancing problem among the gatekeepers. Without loss of generality, we assume that the number of available gatekeepers in the calling and called party side are both m. That is, there are total $2m$ gatekeepers in our model. Without load-balancing methods, the calling party randomly selects a certain gatekeeper in the calling party side to set up the Internet telephony call, and the selected gatekeeper then also randomly selects a certain gatekeeper in the called party side to route this call. On the other hand, with the help of our load-balancing methods, the gatekeepers with the lightest loads will be always selected in the calling and called party sides to initiate and route the call.

In addition, we assume that each gatekeeper can admit at most c full-duplex channels, i.e., allows at most c Internet telephony calls to be established simultaneously. The interarrival time of two consecutive call-setup attempts follows an exponential distribution with the mean value of t. The mean duration of each call is assumed to be d. Table I provides a summary of the parameters and the default values used in the simulation.

B. Experimental Results

Based on the system model designed, we implement a simulator using Microsoft Visual C++ package on an IBM-compatible PC with Pentium III 450 CPU and 128-MB RAM. We focus on two performance metrics, *call-blocking ratio* and *channel utilization*, which are both sensitive to the distribution of the load. The main objective of our simulation is to show the impact of our load-balancing methods on the *call-blocking ratio* with the aim to minimize it, and also the impact on the *channel utilization* with the aim to maximize it. Each value is the result of 10 000 inputs of the call-setup attempts.

In the first experiment, we compare the *call-blocking ratios* under the circumstances with and without the load-balancing methods. First, we fix the mean duration of each call to 100 s, and examine the *call-blocking ratios* when the value of the mean interarrival time of two consecutive call-setup attempts

TABLE I
PARAMETERS OF THE SYSTEM MODEL

System Parameters	*Value (default)*
Total number of call setup attempts	10000
Number of GKs ($2m$)	2 in each side
Number of channels each GK admits (c)	10
Interarrival time of call setup attempts (t)	Exp. ($\mu = 5$ sec.)
Duration of each call (d)	100 sec.

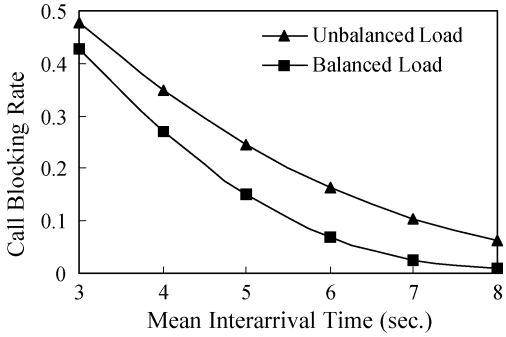


Fig. 11. Mean call-blocking rates under various mean interarrival time of call-setup attempts.

increases. The *call-blocking ratio* in the experiment is calculated as

$$\text{call blocking ratio} = \frac{\text{number of call setup rejects}}{\text{number of call setup attempts}}.$$

Fig. 11 shows the *call-blocking ratio* as the value of the mean interarrival time of two consecutive call-setup attempts varies. We observe that the *call-blocking ratio* under the circumstance with load-balancing methods is in general smaller for different values of the mean interarrival time of two consecutive call-setup attempts. Specifically, the mean improvement of the *call-blocking ratio* is 34.2%. The reason is that without load-balancing methods, the probability of which the overall load will be rivetted on some gatekeepers raises as the overall traffic increases. It will, in turn, increase the *call-blocking ratio*. However, with our load-balancing methods, the overall load will be equally distributed to each gatekeeper while the overall traffic increases. Thus, the *call-blocking ratio* can be decreased. Note that as the traffic load increases to some level, the *call-blocking ratios* are high in both cases.

Next, we fix the mean interarrival time of two consecutive call-setup attempts to 5 s, and increase the mean duration of each call to examine the variation of the *call-blocking ratio*. Fig. 12 shows the *call-blocking ratio* against the mean duration of each call. We note that the *call-blocking ratio* is minimized with proposed load-balancing methods when the mean duration of each call increases. Specifically, the mean improvement of the *call-blocking ratio* is 25.5%. This is also due to the fact that the overall load is fairly distributed among all gatekeepers. Note that the increasing slopes of the unbalanced and balanced load are similar to each other. It is because the *call-blocking ratio* is more sensitive to the traffic load (as shown in Fig. 11) rather than to the mean duration of each call (as shown in Fig. 12).

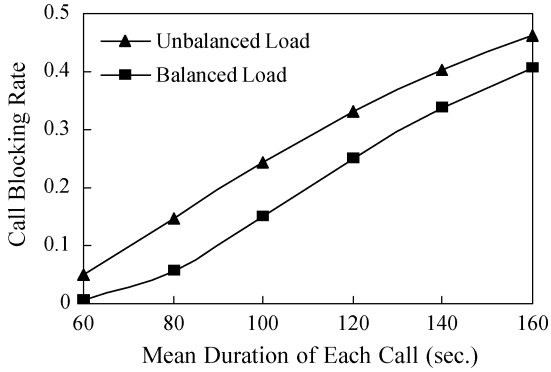


Fig. 12. Mean call-blocking rates under various mean duration of each call.

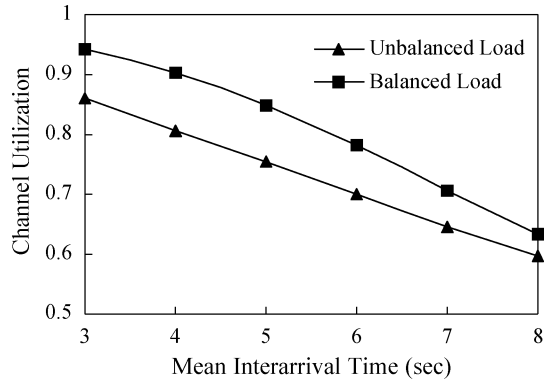


Fig. 13. Channel utilization under various mean interarrival time.

In the second experiment, we evaluate the effect on the *channel utilization* under the circumstances with load-balancing methods. Similar to the first experiment, we first fix the mean duration of each call to 100 s, and examine the *channel utilization* when the value of the mean interarrival time of two consecutive call-setup attempts increases. The *channel utilization* in the experiment is calculated as

$$\text{channel utilization} = \frac{\text{mean number of channels occupied}}{\text{total number of available channels}}.$$

Fig. 13 shows the *channel utilization* against the mean interarrival time of two consecutive call-setup attempts. It can be seen that *channel utilization* performs better with our load-balancing methods especially when the traffic load is high (i.e., the mean interarrival time of two consecutive call-setup attempts is short). Specifically, the mean improvement of the *channel utilization* is 13.1%. The reason is that the *call-blocking ratio* is minimized by our load-balancing methods, so the *channel utilization* is, in turn, increased.

Then, we fix the mean interarrival time of two consecutive call-setup attempts to 5 s, and increase the mean duration of each call to examine the variation of the *channel utilization*. Fig. 14 shows the *channel utilization* against the mean duration of each call. The result also reveals that the *channel utilization* performs better under the effect of our load-balancing methods on minimizing the *call-blocking ratio*. Specifically, the mean improvement of the *channel utilization* is 11.7%. It is noted that the *channel utilization* raises more sharply under the circumstance with load-balancing methods when the mean duration of each

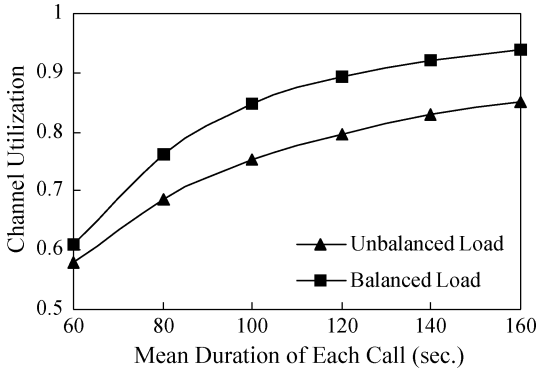


Fig. 14. Channel utilization under various mean duration of each call.

call increases. The reason is that the *channel utilization* is more sensitive to the mean duration of each call than the traffic load.

V. Conclusions

In this paper, we designed and implemented an H.323 gatekeeper prototype for video conferencing on packet-based networks. The gatekeeper prototype is designed to follow the ITU-T H.323 Recommendation, and is hence interoperable with other H.323 compatible endpoints on the Internet. To illustrate the usefulness of this prototype, we devised two improved call-routing methods to deal with the load-balancing problem among the gatekeepers. Using the gatekeeper prototype built, one method for load balancing is based on *MRML* and the other is based on *MRHD*. To evaluate the performance improvement by these two methods, we designed an event-driven simulator and conducted some experiments on it. We focused on two performance metrics, *call-blocking ratio* and *channel utilization*, which are both sensitive to the distribution of the load. The experimental results showed that the *call-blocking ratio* and *channel utilization* can be significantly improved by the proposed load-balancing methods since the overall load will be equally distributed to every participating gatekeeper. It is also observed from our empirical results that the *call-blocking ratio* is more sensitive to the traffic load, whereas the *channel utilization* is more sensitive to the duration of a call.

Appendix
Algorithm for Admission Control

```
Algorithm Admission Control()
if (the caller has registered at the GK){
  if (the type of the callee's address !=
  Transport Address){
    if (the callee has registered at the GK){
      Search the local registration database
  for address translation;
      send ACF to the callee;
      return true;
    } else {
      Multicast LRQ to other GKs;
      Wait for LCF (timeout = 2 second);
      if (LCF is received){
        send ACF to the callee;
        return true;
      } else {
        send ARJ to the callee;
        return false;
      }
    }
  } else {
    Send ACF to the caller;
    return true;
  }
} else
  Send ARJ to the caller;
  return false;
}
```

References

[1] RADVISION, Ltd. [Online] Available http://www.radvision.com

[2] N. Anerousis, R. Gopalakrishnan, C. R. Kalmanek, A. E. Kaplan, W. T. Marshall, P. P. Mishra, P. Z. Onufryk, K. K. Ramakrishanan, and C. J. Sreenan, "TOPS: an architecture for telephony over packet networks," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 91–108, Jan. 1999.

[3] *Recommendation E.164: Telephone Network and ISDN Operation, Numbering, Routing and Mobile Service*, CCITT/ITU-T.

[4] *Recommendation G.711: Pulse Code Modulation of Voice Frequencies*, CCITT/ITU-T.

[5] C.-Y. Chang and M.-S. Chen, "On building an internet gateway for internet telephony," in *Proc. IEEE Int. Conf. Multimedia and Expo*, New York, Aug. 2000.

[6] I. Dalgic, M. Borella, R. Dean, J. Grabiec, J. Mahler, G. Schuster, and I. Sidhu, "True number portability and advanced call screening in a SIP-based IP telephony system," *IEEE Commun. Mag.*, vol. 37, pp. 96–101, July 1999.

[7] T. Eyers and H. Schulzrinne, "Predicting internet telephony call setup delay," in *Proc. 1st IP Telephony Workshop (IPTel 2000)*, Berlin, Germany, 2000.

[8] P. Goyal, A. Greenberg, C. R. Kalmanek, W. T. Marshall, P. Mishra, D. Nortz, and K. K. Ramakrishnan, "Integration of call signaling and resource management for IP telephony," *IEEE Network*, vol. 13, pp. 24–32, May/June 1999.

[9] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, RFC 2543: SIP: Session Initiation Protocol.

[10] M. Hassan, A. Nayandoro, and M. Atiquzzaman, "Internet telephony: Services, technical challenges, and products," *IEEE Commun. Mag.*, vol. 38, pp. 96–103, Apr. 2000.

[11] C. Huitema, J. Cameron, P. Mouchtaris, and D. Smyk, "An architecture for residential internet telephony service," *IEEE Network*, vol. 13, pp. 50–56, May–June 1999.

[12] *Recommendation G.723: Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 Kbps Using Low-delay Code Excited Linear Prediction*, ITU-T.

[13] *Recommendation H.225: Media Stream Packetization and Synchronization on Non-Guaranteed Quality of Service LANs*, ITU-T.

[14] *Recommendation H.245: Control Protocol for Multimedia Conferencing*, ITU-T.

[15] *Recommendation H.261: Video Codec for Audiovisual Services at $p \times 64$ Kbps*, ITU-T.

[16] *Recommendation H.263: Video Coding for Low Bit Rate Communication*, ITU-T.

[17] *Recommendation H.323: Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-Guaranteed Quality of Services*, ITU-T.

[18] *Recommendation Q.931: ISDN User-Network Interface Layer 3 Specification for Basic Call Control*, ITU-T.

[19] C. R. Kalmanek, W. T. Marshall, P. P. Mishra, D. M. Nartz, and K. K. Ramakrishnan, "DOSA: an architecture for providing a robust IP telephony service," in *Proc. 19th Annu. Joint Conf. IEEE Computer and Communications Societies (INFOCOM)*, Tel-Aviv, Israel, 2000.

[20] W.-J. Liao, "Mobile internet telephony: Mobile extensions to H.323," in *Proc. 18th Annu. Joint Conf. IEEE Computer and Communications Societies (INFOCOM)*, New York, Mar. 1999.

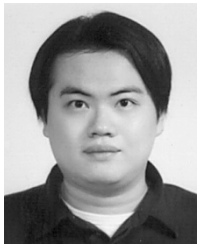
[21] M. Moh, G. Berquin, and Y. Chen, "Mobile IP telephony: Mobility support of SIP," in *Proc. IEEE ICCN*, San Juan, PR, 1999.

[22] S. Okubo, S. Dunstan, G. Morrison, M. Nilsson, H. Radha, D. L. Skran, and G. Thom, "ITU standardization of audiovisual communication systems in ATM and LAN environment," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 965–982, Aug. 1997.
[23] B. Pagurek, J. Tang, T. White, and R. Glitho, "Management of advanced services in H.323 internet protocol telephony," in *Proc. 19th Annu. Joint Conf. IEEE Computer and Communications Societies (INFOCOM)*, Tel-Aviv, Israel, 2000.
[24] C. A. Polyzois, K. H. Purdy, P. F. Yang, D. Shrader, H. Sinnreich, F. Mard, and H. Schulzrinne, "From POTS to PANS—a commentary on the evolution to internet telephony," *IEEE Network*, vol. 3, May/June 1999.
[25] D. Rizzetto and C. Catania, "A voice over IP service architecture for integrated communications," *IEEE Internet Comput.*, vol. 3, pp. 53–62, May/June 1999.
[26] J. Rosenberg and H. Schulrinne, RFC 2871: A Framework for Telephony Routing Over IP (TRIP).
[27] H. Schulrinne, A. Rao, and R. Lanphier, RFC 2326: Real Time Streaming Protocol (RTSP).
[28] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, RFC 1889: RTP—A Transport Protocol for Real-Time Applications.
[29] H. Schulzrinne and J. Rosenberg, "A comparison of SIP and H.323 for internet telephony," in *Proc. Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, U.K., 1998.
[30] ——, "Internet telephony: architecture and protocols—an IETF perspective," *Comput. Networks ISDN Syst.*, vol. 31, Feb. 1999.
[31] A. Srinivasan, K. G. RamaKrishnan, and K. Kurnaran, "Optimal design of signaling networks for internet telephony," in *Proc. 19th Annu. Joint Conf. IEEE Computer and Communications Societies (INFOCOM)*, Israel, 2000.
[32] G. A. Thom, "H.323: the multimedia communications standard for local area networks," *IEEE Commun. Mag.*, pp. 52–56, Dec. 1996.
[33] H. J. Wang, A. D. Joseph, and R. H. Katz, "A signaling system using lightweight call sessions," in *Proc. 19th Annu. Joint Conf. IEEE Computer and Communications Societies (INFOCOM)*, Israel, 2000.
[34] M. H. Willebeek-LeMair and Z.-Y. Shae, "Videoconferencing over packet-based networks," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 1101–1114, June 1997.

Ming-Syan Chen (M'88–SM'93–F'04) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., and the M.S. and Ph.D. degrees in computer, information, and control engineering from The University of Michigan, Ann Arbor, in 1985 and 1988, respectively. He is currently a Professor in the Electrical Engineering Department, National Taiwan University. He was a Research Staff Member at IBM Thomas J. Watson Research Center, Yorktown Heights, NY, from 1988 to 1996. His research interests include database systems, data mining, mobile computing systems, and multimedia networking,. He has published more than 140 papers in his areas of research. He holds, or has applied for, 18 U.S. patents and seven R.O.C. patents in the areas of data mining, Web applications, interactive video playout, video server design, and concurrency and coherency control protocols.

Dr. Chen has served as an Associate Editor (1997–2000) and Guest Co-Editor (1996) for the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING and as an Editor of the *Journal of Information Science and Engineering*. He served as Program Vice-Chair of Very Large Data Bases (VLDB-2002), as General Chair of the Real-Time Multimedia System Workshop (2001), as Program Chair for the IEEE ICDCS Workshop on Knowledge Discovery and Data Mining in the World Wide Web (2000) and the Pacific Area Knowledge Discovery and Data Mining Conference (PAKDD-02), and as Program Co-Chair of the International Conference on Mobile Data Management (2003), the International Computer Symposium (ICS) on Computer Networks, Internet and Multimedia (1998 and 2000), and ICS on Databases and Software Engineering (2002). He was a keynote speaker on Web data mining for the International Computer Congress in Hong Kong (1999), a tutorial speaker on Web data mining for DASFAA-1999 and on parallel databases for the Eleventh IEEE International Conference on Data Engineering in 1995, and a Distinguished Visitor of the IEEE Computer Society for the Asia-Pacific Region (1998–2000). He has coauthored (with his student) a paper which received the ACM SIGMOD Research Stduent Award. He received the Outstanding Innovation Award from IBM Corporation in 1994 for his contribution to parallel transaction design and implementation for a major database product, and numerous awards for his research, teaching, inventions, and patent applications.

Dr. Chen is a member of ACM.



Cheng-Yue Chang received the M.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1998 and 2003, respectively.

He is currently a Senior Researcher with Philips Research East Asia—Taipei. His research interests include World Wide Web systems, multimedia networks, and home networks.

Pai-Han Huang received the B.S. degree in civil engineering in 1998 and the Master's degree in electrical engineering in 2000, both from National Taiwan University, Taipei, Taiwan, R.O.C.

His research interests include multimedia networking and Web computing.