http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 5 (6): 1088-1092, 2006
ISSN 1812-5638

Applying Application Level Gateways for Real-Time Service

Zhao Dongping, Zhang Deyun and He Lin
Department of Computer Science and Technology,
Xi'an Jiaotong University, P.O. Box 710049, Xi'an China

Abstract: Network Address Translation (NAT) has become a powerful function of routers, which effectively conceals the size and topology of the private network from the outside. NAT works at the IP and TCP/UDP levels by translating the address and port numbers in the IP header and TCP/UDP header, respectively. But real-time applications such as voice over IP use IP addresses and port numbers inside their data payloads. In this paper, the author presents a new method, Application Level Gateways (ALGs), to allow an application on a host in one address realm to connect to its counterpart running on a host in different realm transparently. An ALG may interact with NAT to set up state, use NAT state information, modify application specific payload and perform whatever else is necessary to get the application running across disparate address realms. The concept of sub-flow classifier for H.323 packets is also proposed to shorten the call set-up time. The experiments demonstrate it very efficient and feasible.

Key words: Network address translation, application level gateways, H.323, voice over IP

INTRODUCTION

Network Address Translation (NAT) (Srisuresh and Egevang, 2001; Senie, 2002) is widely used in routers, firewalls, DSL and cable modems. A device running NAT can connect an entire department or small office to the Internet using only a single global IP address. Address mapping and port translation effectively conceal the size and topology of the private network, providing a basic level of security. NAT can also save network administration costs by reducing the number of public IP addresses that must be purchased or leased from a service provider.

Multimedia real-time services like audio, video communications, especially VoIP, have been made rapid progress in recent years (Boucouvalas, 2002). As a basic application of router, NAT should support those services. However, there are many problems to achieve NAT for VoIP, because the IP address and port number used during call setup is inside their data payloads, NAT works only at the IP and TCP/UDP levels. We should take some other measures to resolve this problem.

DESIGN AND IMPLEMENTATION

Introduction of H.323: Recommendation H.323 (ITU, 2003a) is a set of protocols for voice, video and data conferencing over packet-based networks. The current recommendation, known as version 5, was ratified by the

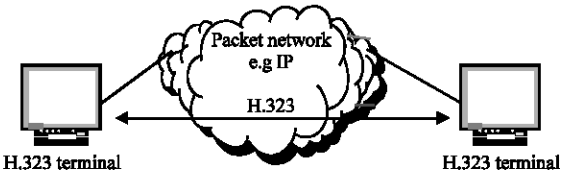

Fig. 1: H.323 protocol

Telecommunications Sector of the International Telecommunication Union (ITU-T). The H.323 protocol stack is designed to operate above the transport layer of the underlying network. As such, H.323 can be used on top of any packet-based network transport like Ethernet, TCP/UDP/IP, ATM and Frame Relay to provide real-time multimedia communication.

The H.323 protocol uses the concept of channels to structure the information exchange between communication entities. A channel is a transport layer connection, which can be either unidirectional or bi-directional (Fig. 1). H.323 for regular call involves multiple messages from its underlying protocols like H.225 (ITU, 2003b) and H.245 (ITU, 2003c). H.225 call signaling is used to set up connections between H.323 endpoints (terminals and gateways), over which the real-time data can be transported. Call signaling involves the exchange of H.225 protocol messages over a reliable call signaling channel. For example, H.225 protocol messages are carried over TCP in an IP-based H.323 network. H.245 Control signaling consists of the exchange

Corresponding Author: Zhao Dongping, Department of Computer Science and Technology, Xi'an Jiaotong University, P.O. Box 710049, Xi'an China Tel: 86-29-82668096

of end-to-end H.245 messages between communicating H.323 endpoints. The H.245 control messages are carried over H.245 control channels. The H.245 control channel is the logical channel and is permanently open, unlike the media channels. The messages carried include messages to exchange capabilities of terminals and to open and close logical channels.

Application Level Gateways (ALGs): NAT works only at the IP and TCP/UDP levels by translating the address and port numbers in the IP header and TCP/UDP header, respectively. VoIP applications, however, use IP addresses and port numbers inside their data payloads. NAT cannot provide the necessary application and protocol transparency. Therefore, Application Level Gateways (ALGs) (Renesse and Dumitriu, 2002) are required to interact with NAT to provide end-to-end transparency. Because different applications employ different protocols or data formats, ALGs must be customized to each application.

Application Level Gateways (ALGs) are application specific translation agents that allow an application on a host in one address realm to connect to its counterpart running on a host in different realm transparently. An ALG may interact with NAT to set up state, use NAT state information, modify application specific payload and perform whatever else is necessary to get the application running across disparate address realms.

The H323 ALG component uses the ALG API to interface with NAT, as follows:

- To register ALG with NAT.
- As a NAT interface with ALG while an H323 session is active.
- For action occurring when the H323 session terminates.

Data structures and interface functions: An enhanced NAT developed by ourselves comes with a set of API routines that allows you to develop an ALG that closely interacts with NAT to modify application specific payloads and perform necessary tasks to get the application running across disparate address realms. This set of API routines enables the ALG to register itself and its callback routines to NAT, create and delete address binds and query bind information.

This section describes some important data structures and interface functions used in our enhanced ALG API. Figure 2 shows the sequence of calls between a typical ALG and NAT. The solid arrows represent required interactions. The dotted arrows represent interactions that may be optional depending on the ALG requirements.

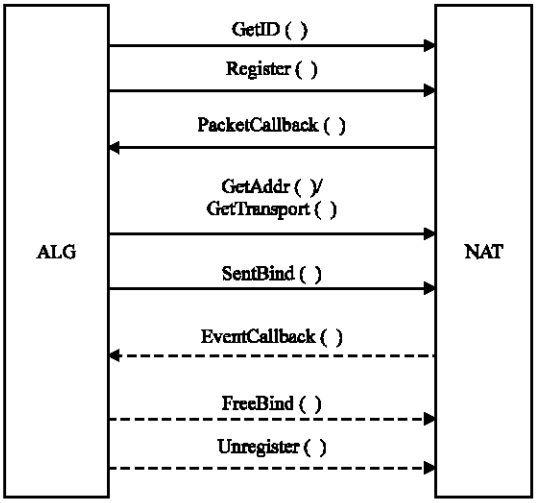

Fig. 2: ALG interface to NAT

1) AGENT_INFO

```
typedef struct {
    u_long id; /* Agent idenitfier, set by NAT */
    char desc[NAT_AGENT_DESC_LEN]; /* official description */
    char name[NAT_AGENT_NAME_LEN]; /* name of agent */
    NAT_AGENT_TYPE type; /* One of NAT_AGENT_TYPE */
    u_long flags; /* See NAT_FLAG */
    u_long events; /* bit-field of requested callback events */
    u_long timer_interval; /* timer event interval (in seconds) */
    NAT_EVENT_CALLBACK event_callback; /* agent callback function pointer */
    NAT_SESSION_TAG session_tag;
    /* Identifies type of packets to translate */
    NAT_PACKET_CALLBACK packet_callback;
    /* agent packet processing function */
} NAT_AGENT_INFO;
```

This structure is a global structure used to keep the information of registered agent.

2) NAT_STATUS GetID(NAT_ID_INFO* *id_info*)

The function is called by the external agent to obtain the NAT service identifier and its type. The external agent is responsible for allocating the NAT_ID_INFO data structure and passes its pointer as the argument to this function. NAT fills in the structure including the NAT ID.

3) NAT_STATUS Register (u_long nat_id, AGENT_INFO* *agent_info*)

The external agent calls the function to register with the NAT service, allocate the AGENT_INFO storage and fill in the agent's information before passing it to the

function. Upon successful completion of this function, NAT will assign and return the agent ID (i.e., handle) in the AGENT_INFO data structure.

4) NAT_STATUS GetAddr (u_long *nat_id*, u_long *local_addr*, u_long *global_addr*, BIND_INFO* *bind_found*)

The external agent calls this function to get address bind information. The caller may specify both or just the local address or the global address and set the other to zero. NAT will fill up the NAT_BIND_INFO data structure with the address bind information unless it cannot find a match for the addresses specified.

5) NAT_STATUS GetTransport (u_long *nat_id*, NAT_BIND_SESSION* *session*, NAT_BIND_INFO* *bind_found*, NAT_BIND_CHECK *check*)

The external agent calls this function to attain the transport ID bind information. The caller may specify both or just one of either local or global (address and transport). NAT will fill up the NAT_BIND_INFO data structure with the transport bind information unless it cannot find a match for the addresses specified. The check flag has two options, FULL to check for match of remote address and port and PARTIAL to check for match of remote address only.

6) NAT_STATUS SetBind (u_long *nat_id*, u_long *agent_id*, NAT_BIND_INFO* bind_info).

The external agent calls this function to create a new address bind or set certain parameters of an existing bind. The bind can be an address bind or transport bind. The caller is expected to fill in the NAT_BIND_INFO structure. A new bind request is made by setting the bind ID (in NAT_BIND_INFO) to 0. A non-zero bind ID means that the agent attempts to set some existing bind parameters.

7) NAT_STATUS FreeBind(u_long *nat_id*, u_long *agent_id*, u_long *bind_id*).

The external agent calls the function to terminate a bind and any sessions based on this bind.

8) NAT_STATUS Unregister (u_long *nat_id*, u_long *agent_id*).

The external agent calls this function to unregister itself from the NAT service.

Packet processing functions and H.323 sub-flow classifier: The performance of the H.323 protocol to setup a multimedia call can be poor mainly due to the poor performance of TCP with the support of a retransmission mechanism. Hence, there is a need to treat the H.323 media streams for interactive traffic (particularly, VoIP) differently from the H.323 control packet streams (e.g., H.225, H.245, RTCP messages). Also, due to the interactive nature of voice, VoIP packets should get higher priority over any in-band H.323 control packets within the session.

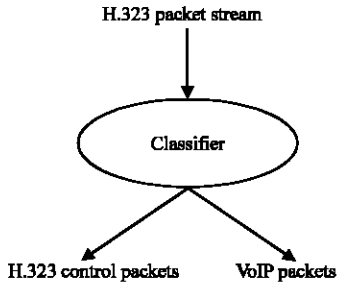

Fig. 3: H.323 sub-flow classifier

Two types of sub-flows requiring differentiation have been identified for VoIP service using H.323. These are:

- H.323 control packets including H.225, H.245 and RTCP (Schulzrinne *et al*., 2003). These packets need higher reliability but less stringent delay requirements compared to the media packets.
- H.323 media packets carried by RTP (Schulzrinne *et al*., 2003). These packets have lower reliability requirements than the control packets, but have stringent delay requirements.

We design a classifier to classify the sub-flows in implementation of NAT for VoIP using H.323 protocol, which distinguishes the control packets and VoIP packets and processes them respectively according to their requirements (Seung-Gu and Jong-Suk, 2000; Wanxiang and Zhenming, 2001). The functional requirements of this classifier are shown in Fig. 3.

In the next section, we explain some functions to process control packets and events, which are very important to implement NAT for VoIP. They are designed as follows:

1) BOOL H225Packet (u_long *nat_id*, u_long *agent_id*, u_long *session_id*, NAT_DIRECTION *direction*, void* *packet*)

This function handles the H.225/Q.931 call signaling packets. NAT calls this function after translating the address and port number in the IP and TCP headers and when the source or destination port is the H.323 port registered to NAT during the H.323 ALG registration (the standard port for H.323 is 1720). The H.225 call signaling protocol is used to establish connection between two end-points.

This function must check both an outbound and an inbound packet. H.323 connection can be attempted from local to global endpoint or vice versa.

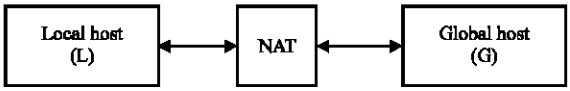

Fig. 4: Two-endpoint connection

The fields in the H.225 message are ASN.1 encoded. This function uses a simple strategy to look for the ip/port address in the H.225 payload by taking advantage of the fact that the port number always follows immediately after the IP address. Since the ALG can get the expected IP address from NAT, it can search byte by byte for this IP address to locate where it is in the H.225 payload. Of course, other search algorithm can be also employed. The tuple of IP address and TCP/UDP port number is sometimes called transport address in some publications and the same terminology will be used here.

There are two cases when the connection establishment as Fig. 4 shows.

Case 1: Local (L) endpoint to global (G) endpoint connection: L starts a TCP connection to G and the H.225 call signaling session starts. The establishment of this TCP connection is all handled by NAT, so NAT should have the bind entry for this connection. During the H.225 session, L will embed its Source Call Signal Address (ip/port) and G's transport address in the H.225 payload to G. Thus, this function must parse for each H.225 outbound packet, look for L's transport address and substitute it with the translated address obtained from NAT.

In addition, this function must also observe the H.225 inbound packets from G to look for the H.245 transport address (ip/port) in the connecting message from G. The port number provided by G here will serve as the H.245 port number. L will use this port number to open the H.245 TCP connection with G. So upon obtaining the H.245 port number, this function must also register the H.245 ALG agent to NAT.

Case 2: Global (G) endpoint to local (L) endpoint connection: G starts a TCP connection to L via the NAT's H.323 static entry and the H.225 call signaling session starts. As in case 1, the establishment of the TCP connection is all handled by NAT, so NAT should have the bind entry for this connection. During the H.225 session, G will embed its Source Call Signal Address and L's global transport address in the H.225 payload. So, this function must examine all inbound H.225 packets and substitute L's global transport address with its real transport address.

As in case 1, this function must also observe the H.225 outbound packets to look for L's H.245 transport address which is embedded in the connect message from L to G. The port number embedded in this message will serve as the H.245 port number.

G will use this port number to open the H.245 TCP connection with L. So, this function must also register the H.245 ALG agent to NAT upon obtaining the H.245 port number. Furthermore, since the H.245 TCP connection will be started from G to L, this function must create a new TCP bind entry for the impending H.245 port connection.

What should be noted is that TCP sequence adjustment is not required since it is a binary substitution of IP address and port number in the payload. However, the checksum in the TCP header must be adjusted when the TCP payload is modified.

2) BOOL H245Packet (u_long *nat_id*, u_long *agent_id*, u_long *session_id*,

NAT_DIRECTION *direction*, void* *packet*)

This function deals with the H.245 call control packets. NAT calls this function after translating the address and port number in the IP and TCP headers when the source or destination port is the H.245 port. The H.245 port was negotiated during the H.225 call signaling session and registered to NAT. The H.245 call control protocol is used in H.323 to negotiate call parameters between two end-points. For example, it negotiates the UDP connections for the open logical channels for RTP and RTCP streams between the two endpoints. In addition, it also negotiates the TCP connection for the T120 session.

For an outbound packet, L may send its transport addresses to G to let G make TCP/UDP connections to it. Similarly, for an inbound packet, G may send its transport addresses to L to let L make TCP/UDP connections to it. Since only the L's transport addresses in the payload need to be translated, it is sufficient to examine only the outbound packets.

EXPERIMENTS

The module of NAT for VoIP has been experimented by Microsoft NetMeeting. An end-to-end Microsoft NetMeeting VoIP session from Endpoint A (Caller) to Endpoint B (Callee) was created. NetMeeting 3.01 uses the H.323V2 call signaling (Q.931/H.225/H.245 over TCP) to setup VoIP session and the default Microsoft codec G.723. In these experiments, the call is considered successful only when the voice path is cut through (i.e., Endpoint B can hear the caller's voice). In addition, the maximum amount of time the called party waited for voice cut-through is 2 minutes, after which the call was marked as unsuccessful.

Table 1: Call success rate examination

Types	Times	Call setup success rate (%)
L to G	30	100
G to L	30	100

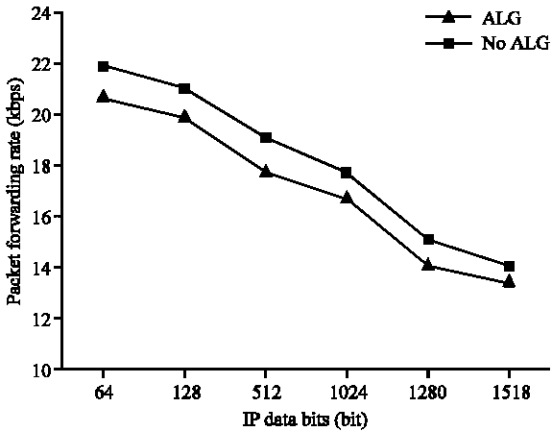

Fig. 5: Packet forwarding rate

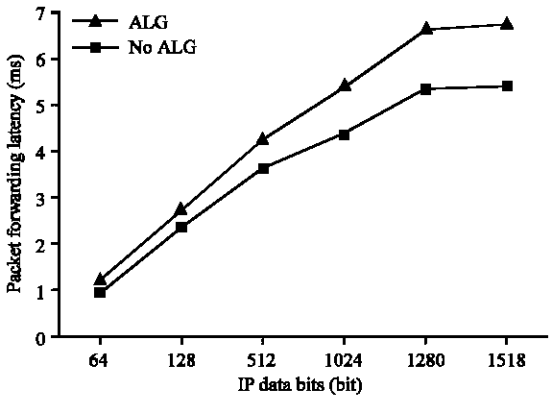

Fig. 6: Packet forwarding latency

Two sets of experiments were conducted, one with the call setup from Local to Global and the others from Global to Local. The success rate is shown in Table 1.

In order to test the packet processing functions, we used SmartBits 6000B, a professional network testing instrument, to generate test data of H.323 control packets.

The test results are demonstrated in Fig. 5 and 6. Figure 5 shows the packets forwarding rate and Fig. 6 shows the packet forwarding latency. No ALG in Fig. 5 and 6 presents ordinary packets that need not to be processed by ALG and ALG presents H.323 Control packets. We can find that the implementation method of ALG affects slightly on the performance of NAT when receiving H.323 control packets. As for ordinary packets, this method has no effect on them because it doesn't deal with those packets.

CONCLUSIONS

We proposed an implementing method of NAT for real-time application, which is ALG and supports the two most widely used NAT modes: Basic NAT and Network Address Port Translation (NAPT). In order to shorten the call set-up time, we proposed the concept of sub-flow classifier for H.323 packets. The classifier distinguishes the control packets and VoIP packets and processes them respectively according to their requirements. The efficiency of this method has been experimented by Microsoft NetMeeting and we have designed another experiment to test the control packet processing functions. The method has been also applied to the router developed by our network institute. The experiments and applications demonstrate it feasible to realize NAT for real-time service.

REFERENCES

Boucouvalas, A., 2002. Future evolution of network technologies. In: Proceedings of the 24th International Conference on Information Technology Interfaces. Begins, Sweden, IEEE., pp: 3-6.

ITU (International Telecommunication Union), 2003a. Packet-based multimedia communications systems. ITU-T Recommendation H.323, Geneva, Switzerland.

ITU (International Telecommunication Union), 2003b. Call signalling protocols and media stream packetization for packet-based multimedia communication systems. ITU-T Recommendation H.225.0, Geneva, Switzerland.

ITU (International Telecommunication Union), 2003c. Control protocol for multimedia communication. ITU-T Recommendation H.245, Geneva, Switzerland.

Renesse, V. and D. Dumitriu, 2002. Collaborative networking in an uncooperative Internet. In: Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems. Begins, Sweden. IEEE., pp: 51-60.

Schulzrinne, H. *et al.*, 2003. RFC3550 - RTP: A Transport Protocol for Real-Time Applications. IETF.

Senie, D., 2002. RFC 3235 - Network Address Translator (NAT)-Friendly Application Design Guidelines. IETF.

Seung-Gu, N. and A. Jong-Suk, 2000. TCP-like flow control algorithm for real-time applications. In: Proceedings of IEEE International Conference on Networks 2000. Singapore, IEEE., pp: 99-104.

Srisuresh, P. and K. Egevang, 2001. RFC 3022 - Traditional IP Network Address Translator (Traditional NAT). IETF.

Wanxiang, C. and L. Zhenming, 2001. A modified RTP adaptive algorithm. In: Proceedings of 2001 International Conferences on Info-tech and Info-net. Beijing, China, IEEE., pp: 33-38.