IPv4/IPv6 Translation

Allowing IPv4 hosts to communicate with IPv6 hosts without modifying the software on the IPv4 or IPv6 hosts

J. William Atwood[*]
Kedar C. Das
Xing (Scott) Jiang
Concordia University
Department of Computer Science
Montréal, Québec H3G 1M8
`bill@cs.concordia.ca, http://www.cs.concordia.ca/˜faculty/bill`

Abstract

As the Internet makes the transition from IP version 4 to IP version 6, it will be necessary to allow IPv4-based clients to access IPv6-based servers, and IPv6-based clients to access legacy services. Network Address Translation–Protocol Translation (NAT-PT) can provide network protocol translation, and Application Layer Gateways (ALGs) can handle the cases where peer addresses are embedded in application-layer messages. We describe an implementation on a Linux router/translation server, the necessary configuration of the IPv4 and IPv6 environments, and the operation of ALGs for the File Transfer Protocol and for the Session Initiation Protocol. We will present a demonstration of basic communication (web client to web server), and of multimedia communication (based on the open-source VOCAL project).

1 Introduction

IPv6 is the next generation protocol designed by the IETF to replace the current version of the Internet Protocol, IPv4. During the last decade, IP has conquered the world's networks. Most of today's Internet uses IPv4, which has been remarkably resilient in spite of its age, but it is beginning to have problems.

One motivation for developing IPv6 was the anticipated exhaustion of addresses for individual hosts. While the rate of depletion has been slowed through the use of Network Address Translation (NAT) [1], it does continue, and the other virtues of IPv6 (routing and network autoconfiguration, and enhanced support for IP Security (IPsec) and IP Mobility (Mobile IPv6)), will encourage its deployment much more widely in coming years.

Although a significant percentage of the clients and servers will be *dual stack* (i.e., capable of using either IPv4 or IPv6), there will be a large number of existing clients and servers (legacy systems) that will only be capable of using IPv4, and there will be a growing number of

[*] On leave at Ericsson Research Canada, Open Systems Research Laboratory, Montréal, Québec, Canada

clients and servers that will only be capable of using IPv6. For example, the Third Generation Partnership Project (3GPP) has mandated that third generation cellular networks will be "All-IP," and that the "IP" will be IP version 6 *only*.

In the same way as NAT has been used to connect hosts on private networks [2] with hosts on the public network, Network Address Translation–Protocol Translation (NAT-PT) [3] has been standardized as a way of connecting hosts in the IPv4 address space and hosts in the IPv6 address space. NAT and NAT-PT work by altering the IP headers. For NAT, only the address fields are replaced; for NAT-PT, the entire header is changed. This use of NAT-PT solves the network-layer problem for the IPv4/IPv6 transition, but it requires some auxilliary services to operate properly, and it does not solve a number of application-layer problems associated with crossing the IPv4/IPv6 boundary.

In this paper, we discuss the auxilliary services needed, report an experimental validation of the requirements, outline the solution to some of the application-layer problems, and speculate on the solution to the rest.

2 System Architecture

Figure 1 identifies the typical components that will be used to support communication between IPv4 hosts and IPv6 hosts. The IPv4 region represents the entire IPv4-based Internet of today. The IPv6 stub region contains the hosts that are to be granted the privilege of accessing legacy (IPv4-based) services. The IPv6 region represents the rest of the IPv6 address space. The v4/v6 Border Router provides the connection between the IPv6 stub hosts and the IPv4 hosts. The v6/v6 Border Router provides the connection between the IPv6 stub region hosts and the rest of the IPv6 address space. In some systems, the v4/v6 Border Router and the v6/v6 Border Router will be co-located. However, we leave them separate in the following, to make the explanations clearer.

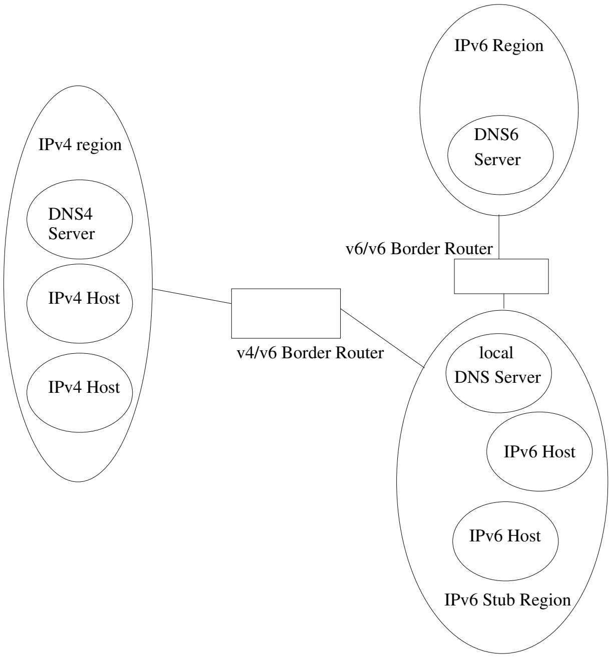

Figure 1: System Architecture

Each host has a *host name* and a *host address*. The host name is a (globally unique) character string that is intended to be human-readable. The host address is a (globally unique) 32-bit (IPv4) or 128-bit (IPv6) number. A particular host may have more than one name, and more than one address, especially if it has multiple interfaces.

Two address pools are associated with the v4/v6 Border Router. The *IPv4 address pool* is a sequence of addresses that are associated (temporarily) with the IPv6 hosts that are communicating with IPv4 hosts. Given the scarcity of IPv4 addresses, this pool will be sized to correspond to the number of IPv6 hosts (in the IPv6 stub region) that are *actively* communicating with IPv4 hosts at a particular time. The IPv6 address pool is a sequence of addresses that represent hosts in the IPv4 region. Given the large size of the IPv6 address space, this pool is structured as a 96-bit prefix, catenated with a 32-bit IPv4 address. A motivation for this will be presented later, in Section 3.6.

2.1 Translation Requirements

As packets move between the IPv4 region and the IPv6 region, it is necessary to rebuild their headers, since the IPv4 and IPv6 packet headers have different formats. This process is stateless—the necessary mapping information is determined by tables in the v4/v6 Border Router (for packets travelling from the IPv4 region to the IPv6 stub region) or by information carried in the packet address (for packets travelling from the IPv6 stub region to the IPv4 region). For NAT-PT, the mapping between an IPv4 pool address and the corresponding IPv6 host address is one-to-one. When there are insufficient IPv4 pool addresses available, then NAPT-PT can be used, with a mapping from (IPv4 address, IPv4 port) to (IPv6 address). This allows about 65,000 IPv6 hosts to be serviced using a single IPv4 address, as long as the IPv4 application does not care about the port that is being used to access it.

Certain packets require special treatment. In general, these packets contain application data that have embedded IPv4 or IPv6 addresses. They are identified when their headers are processed (usually by noting which port they are using), and they are handled by application-specific code called an *Application Layer Gateway* (ALG). The ALGs are application-specific, because they need to be able to parse the packets being exchanged by the application end-points. The specific ALG then modifies the contents of the packet, to reflect the address translation that has just taken place in the headers.

2.2 Centralized Architecture

One approach to handling the entire requirement is to co-locate the NAT-PT software and the set of ALGs needed to support the desired applications. In this case the interaction between the ALG and the translation tables in the NAT-PT software is simplified. However, the v4/v6 Border Router must provide processing power for all functions, which could overload it.

2.3 Distributed Architecture

An alternate approach is to separate the ALGs from the NAT-PT software. This lowers the processing requirements for the v4/v6 Border Router, but it introduces a requirement to define a protocol for interaction between the ALG and the NAT-PT. This approach is favoured when the application makes use of some form of “proxy” server, because the proxy functions and the ALG functions can often be advantageously combined in a single host, and separated from the v4/v6 Border Router. Commercial systems will need to adopt this approach, to handle the large number of IPv6 hosts that will be in a typical IPv6 stub region. However, our project was concerned with exploring issues relating to establishing the right environment, and we did not require high performance at this time.

3 NAT-PT Tool

The experimental system was based on a user-space NAT-PT implementation developed at ETRI [4]. The original implementation was based on a Linux 2.4.0 kernel, and required modification to make it work on the more recent (2.4.20) Linux kernel.

3.1 General Flow

To establish communication between IPv4 and IPv6 using NAT-PT we need interactions of at least 3 modules. These are-NAT, PT, ALG.

3.2 Network Address Translation (NAT)

The NAT module implements the transport/network layer translation mechanism. It uses a pool of IPv4 addresses for assigning to IPv6 nodes dynamically, and this assignment is done when sessions are initiated across the v4/v6 boundary.

3.3 Protocol Translation (PT)

As all the fields of IPv6 headers are not the same as that of the IPv4 header, the PT module translates IP/ICMP headers to make end-to-end communication possible. Due to the address translation function and because of possible port multiplexing, PT also makes appropriate adjustments to the upper layer protocol (TCP/UDP) headers, e.g., the checksum.

The IPv4-to-IPv6 translator replaces the IPv4 header of IPv4 packet with an IPv6 header to send it to the IPv6 host. Except for ICMP packets, the transport layer header and data portion of the packet are left unchanged. In IPv6, path MTU discovery is mandatory but it is optional in IPv4. This implies that IPv6 routers will never fragment a packet—only the sender can do fragmentation. Path MTU discovery is implemented by sending an ICMP error message to the packet-sender stating that the packet is too big. When an IPv6 router sends an ICMP error message, it will pass through a translator, which will translate the ICMP error to a form that the IPv4 sender can understand. In this case an IPv6 fragment header is only included if the IPv4 packet is already fragmented. The presence of df flag in the IPv4 header is the indication of Path MTU discovery. However, if the IPv4 sender does not perform path MTU discovery, the translator has to ensure that the packet does not exceed the path MTU on the IPv6 side. The translator fragments the IPv4 packet so that it fits in a 1280 byte IPv6 packet, since IPv6 guarantees that 1280 byte packets never need to be fragmented. Also, when the IPv4 sender does not perform path MTU discovery the translator must always include an IPv6 fragment header to indicate that the sender allows fragmentation.

3.4 Application Layer Gateway (ALG)

Several applications send IP addresses and host names within the payload of the IP packet. Because NAT-PT does not snoop the payload, it requires some Application Level Gateways (ALG) to extract that address to replace it either by IPv4 or IPv6. That is, ALG could work in conjunction with NAT-PT to provide support for many such applications. Two examples are the FTP-ALG and the SIP-ALG. These are discussed in Section 3.8 and Section 4.

While the FTP-ALG and the SIP-ALG are optional (provision of a specific ALG depends on the desire to support that particular application), the DNS-ALG is essential, because it is the trapping of the DNS queries that allows NAT-PT to discover the need for mapping between IPv4 addresses and IPv6 addresses.

In the DNS, “A” records represent IPv4 addresses and “AAAA” or “A6” records represent IPv6 addresses.

3.5 Communication from V4 to V6

Any packet originating on the IPv4 side destined to the IPv6 stub network should cross the v4/v6 Border Router, which is the NAT-PT device. When any packet is received on the IPv4 side, NAT-PT will check the destination address. If it is an IPv4 pool address, and if a mapping exists to an IPv6 host, then the IPv4 packet header is converted to an IPv6 packet header, otherwise the packet is dropped. The IPv6 source address will be the PREFIX catenated with the IPv4 source address; the IPv6 destination address will be the mapped IPv6 host address.

If the packet is a DNS query, then the DNS-ALG will change the query type from "A" to "AAAA", and alter the format of strings ending in "IN-ARPA.ARPA" to the IPv6 format ending in "IP6.INT". When the response is returned, then the DNS-ALG will change the "AAAA" record to an "A" record (if the resolution was successful), and change the resolved IPv6 address to the corresponding IPv4 (pool) address.

When any other response packet is returned to the NAT-PT, the IPv4 destination address will be found by removing the PREFIX from the IPv6 destination address. The source address to be used in the generated IPv4 packet is the IPv4 pool address corresponding to the IPv6 host.

3.6 Communication from V6 to V4

When a packet is received from the IPv6 side, its destination address will consist of the PREFIX catenated with the actual IPv4 destination, so the IPv4 packet can be created using this address as the destination, and the IPv4 pool address corresponding to the IPv6 host as the source address.

The key to the creation of the mapping is the DNS queries. If a DNS query is received from an IPv6 host, it is not known *a priori* whether the target host is a v4 host or a v6 host. The DNS-ALG will therefore split the query into an "A" query sent to the v4 DNS, and an "AAAA" query sent to the v6 DNS. If a v6 response is received, then any v4 response is discarded (the v6 path is preferred). Otherwise, a received v4 response triggers creation of a mapping entry, and then an "AAAA" response is generated using PREFIX catenated with the v4 address.

Returning traffic on the IPv4 side will arrive at a pool address. This is used to determine the correct IPv6 destination address, so that the packet can be forwarded. The IPv6 source address will be the PREFIX catenated with the original IPv4 source address.

3.7 TCP/UDP/ICMP Checksum Update

NAT-PT retains the mapping between a specific IPv6 host address and an IPv4 address from the pool of IPv4 addresses available. The mapping between IPv6 address and an IPv4 from the pool of IPv4 addresses is used in the translation of packets passing through NAT-PT. With the translation of IP header, TCP/UDP/ICMP checksum is also updated in NAT-PT according to specific algorithm.

3.8 FTP Application Layer Gateway (FTP-ALG)

Existing FTP works with IPv4 addresses. Two important FTP commands PORT and PASV will no longer exist in IPv6. The PORT command is used to specify a port different from the default one, and it contains the IPv6 address information. So it can't be used without translation. The PASV command is used to put the server into passive mode, which means the server listens on a specific data port rather than initiating the transfer. This command includes the host name and address of the FTP server and therefore does not work over IPv6 without modification. The PORT command is replaced by the EPRT command, which allows the specification of an extended address for the connection. The extended address specifies the network protocol (IPv6 or IPv4), as well as the IP address and the port to be used. The EPSV command replaces the PASV command. The EPSV command has an optional argument that allows it to specify the network protocol, if necessary. The server reply contains only the port number on which it listens, but the format of the answer is similar to the one used for the EPRT command and has a placeholder for the network protocol and address information might be used in the future to provide flexibility in using FTP through firewalls and NATs. An FTP control session carries the IP address and TCP port information for the data session in its payload; an FTP-ALG provides application level transparency.

If a V4 host originates the FTP session and uses PORT or PASV command, the FTP-ALG will translate these commands into EPRT and EPSV commands respectively prior to forwarding to the V6 node. Likewise, EPSV response from V6 nodes will be translated into PASV response prior to forwarding to V4 nodes.

If a V4 host originated the FTP session and was using EPRT and EPSV commands, the FTP-ALG will simply translate the parameters to these commands, without altering the commands themselves.

3.9 Payload Modifications for V6 originated FTP sessions

If a V6 host originates the FTP session the FTP-ALG has two approaches:

In the first approach, the FTP-ALG will leave the command strings “EPRT” and “EPSV” unaltered and simply translate the <net-prt>, <net-addr> and <tcp-port> arguments from V6 to its NAT-PT (or NAPT-PT) assigned V4 information. <tcp-port> is translated only in the case of NAPT-PT. The same goes for the EPSV response from V4 node. With this approach, the V4 hosts must have their FTP application upgraded to support EPRT and EPSV extensions to allow access from V6 hosts.

In the second approach, the FTP-ALG will translate the command strings “EPRT” and “EPSV” and their parameters from the V6 node into their equivalent NAT-PT assigned V4 node info and attach to “PORT” and “PASV” commands prior to forwarding to the V4 node. However, the FTP-ALG would be unable to translate the command “EPSVALL” issued by V6 nodes. In such a case, the V4 host, which receives the command, may return an error code indicating unsupported function, and this error response may cause FTP applications to simply fail. The benefit of this approach is that is does not impose any FTP upgrade requirements on V4 hosts.

3.10 Header updates for FTP control packets

All the payload translations considered in the previous sections are based on ASCII encoded data. As a result, these translations may result in a change in the size of packet. If the new size is the same as the previous, only the TCP checksum needs adjustment as a result of the payload translation. If the new size is different from the previous, TCP sequence numbers should also be changed to reflect the change in the length of the FTP control session payload. The IP packet length field in the V4 header or the IP payload length field in the V6 header should also be changed to reflect the new payload size. A table is used by the FTP-ALG to correct the TCP sequence and acknowledgement numbers in the TCP header for control packets in both directions.

The table entries should have the source address, source data port, destination address and destination data port for V4 and V6 portions of the session, sequence number delta for outbound control packets and sequence number delta for inbound control packets.

4 SIP-ALG Operation

Many communication applications on the Internet require a session protocol to negotiate and maintain the data exchange between endpoints in a session.

Moreover, as the evolution of the mobile computing and wireless networks, a session is required to handle user mobility, different media type, and media addition and removal in an existing session. Upon these requirements, the Internet Engineering Task Force (IETF) issued the Session Initiation Protocol (SIP) to enable the Internet endpoints (called user agents in SIP) to discover one another and to agree on the parameters of a session.

4.1 SIP overview

"SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls" [5]. SIP is specified as an agile and general-purpose tool that works independently of underlying transport protocols and without dependency on the various media types.

SIP establishes sessions by its invitations in which session descriptions are used to negotiate a set of compatible media types to be shared among participants. In addition, SIP can invite participants to join in an already existing session. SIP transparently supports name mapping and redirect services. SIP proxy severs could be used to facilitate routing SIP requests to the user's current location. SIP also provides a registration function that stores users' current locations that could be used by proxy servers to redirect requests.

The characteristics of SIP are simplicity and flexibility. SIP is not a complete communication system. SIP is rather a component that can cooperate with other IETF protocols to provide complete services to the users. Typically, the Real-time Transport Protocol (RTP) [6] could be combined with SIP to support real-time data transfer and provide QoS feedback; the Session Description Protocol (SDP) [7] could be used for describing multimedia sessions; the Real-Time Streaming Protocol (RSTP) [8] could be used to control delivery of streaming media; and the Media Gateway Control Protocol (MEGACO) [9] could be used for controlling gateways to the Public Switched Telephone Network (PSTN).

It should be noted that SIP does not depend on any of the protocols above that provide services. Rather, SIP provides basic functionalities and operations that can be used to implement different services. Furthermore, "SIP provides a suite of security services, which include denial-of-service prevention, authentication (both user to user and proxy to user), integrity protection, and encryption and privacy services" [5].

4.2 SIP Messages

SIP defines two distinct types of messages: requests and responses. "A SIP message is either a request from a client to a server, or a response from a server to a client. ... Both types of messages consist of a start-line, one or more header fields, an empty line indicating the end of the header fields, and an optional message-body" [5].

4.3 SIP Behavior

SIP requests can be sent directly from a user agent client to a user agent server, or they can traverse one or more proxy servers along the way. User agents send requests either directly to the address indicated in the SIP URI or to a designated proxy (outbound proxy), independent of the destination address. The current destination address is carried in the Request-URI. Each proxy can forward the request based on local policy and information contained in the SIP request. The proxy may rewrite the request URI.

4.4 SIP-ALG Behavior

There exists an increasing need of IP address translation because the networks based on IPv6 addresses have being extended and because the supply of IPv4 addresses is inadequate.

SIP is an application layer control protocol for establishing media sessions. It encounters problems with NAT-like devices, because the payloads of SIP packets carry the addresses for the sessions to be established. However, the NAT function in NAT-PT is application unaware and does not snoop the payloads. Along with NAT-PT and DNS-ALG, a SIP-ALG is needed on the boundary between IPv4 and IPv6.

This section describes a simple implementation of a SIP ALG to enable simple SIP sessions to pass through a NAT-PT box based on the Vocal system (an open source SIP implementation created by Vovida.org [10]). Rather than attempt to make a full specification for SIP-ALG, we have implemented a subset of the functionalities that is sufficient for typical use.

An IP packet carrying a SIP message is identified by NAT-PT box by the characteristic of the SIP protocol, using the port 5060 as the destination.

Whenever a Vocal user agent (UA) initiates a SIP session by the host name of a callee, it looks up the IP address from DNS services. DNS servers transparently provide the address as normal except that the DNS-ALG will setup an address mapping once the DNS query is traversing a boundary of IPv4 and IPv6, which has been described in the previous sections. If a mapping occurs, the SIP message is sent to the NAT box. The SIP-ALG in the NAT box will build a table storing two pairs of the source address and the corresponding destination address for both IPv4 and IPv6 domains. According to the table, the various fields in the SIP message will be modified. If the Content-Type is SDP, the SDP message and the Content-Length will also be adjusted. After that, the modified message is forwarded to another IP version of the network. Similarly, the response messages will be modified back to the original messages correspondingly when they return to the NAT box. Thus, it seems as if the IPv4 UA and the IPv6 UA are communicating with the NAT-PT box respectively.

5 DNS Considerations

The IPv4 region has a DNS server, which returns Address (A) records when queried about a host name that exists in the IPv4 region. The IPv6 stub region has a local DNS server, which returns IPv6 address (AAAA) records when queried about a hostname that exists in the IPv6 stub region. In addition, there is a “global” IPv6 DNS server.

If any IPv6 hosts in the IPv6 stub region are to provide services to IPv4 clients (initiated by the IPv4 clients), the NAT-PT must permanently associate an IPv4 (pool) address to the IPv6 address of the serving host. In addition, one of the following must be true:

1. The IPv4 DNS server must map a host name (probably different from its IPv6 host name) to the associated IPv4 (pool) address, *or*

2. The IPv4 DNS server must refer the DNS request for the associated host name to a DNS server located at a specific IPv4 pool address. Queries sent to this address are

processed by the DNS-ALG at the Border Router, and sent to the local IPv6 DNS (in the stub region). The returned reply is processed by the DNS-ALG, and sent to the original requesting host as a DNS "A" record. This record will contain the IPv4 pool address that was assigned to the IPv6 server.

We had to install and configure a *local* IPv4 DNS server to filter the requests for resolution of host names associated with the IPv6 stub domain. This was done on the host that would serve as the client, to minimize the impact on the rest of the system. Queries about names ending in ".ip6.lmc.ericsson.se" were sent to the statically assigned IPv4 pool address that was associated with the IPv6 DNS server. All other queries were forwarded to the regular IPv4 DNS server. (Note: this "extra" DNS server would be unnecessary in a production environment. The problem would be addressed by putting the necessary directives in the IPv4 DNS server itself.)

6 Conclusion and Future Work

We have demonstrated that it is possible to protect the investment in past hardware and systems, by installing a NAT-PT on the Border Router that provides access to IPv6-based equipment. To do so requires that special care be taken in configuring the Domain Name System servers (for both IPv4 and IPv6), the NAT-PT itself, and the IPv6 hosts (so that their DNS requests are sent to the NAT-PT).

The NAT-PT solution requires one IPv4 pool address for each IPv6 host that is concurrently accessing the IPv4 address space. This could clearly result in exhaustion of the IPv4 address pool. A potential solution is to use NAPT-PT [3], where host/port number pairs on the IPv6 side are mapped to a single IPv4 pool address and multiple port numbers on the IPv4 side. NAPT-PT bears the same relationship to NAPT that NAT-PT bears to NAT; the implementation details are more complex, but the use of NAPT-PT will be necessary when large IPv6 stub regions are able to use only a small pool of IPv4 addresses to provide the desired services.

7 Acknowledgments

The authors acknowledge the support of Ericsson Research Canada through the use of its laboratory. The first author acknowledges the support of the Natural Sciences and Engineering Research council of Canada, through its Discovery Grants program.

8 Availability

A link to the developed implementation is on the web site

```
http:
//www.linux.ericsson.ca/ipv6
```

References

[1] K. Egevang and P. Francis. The IP network address translator (NAT). Request for Comments 1631, Internet Engineering Task Force, May 1994.

[2] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. De, and E. Lear. Address allocation for private internets. Request for Comments 1918, Internet Engineering Task Force, February 1996.

[3] G. Tsirtsis and P. Srisuresh. Network address translation - protocol translation (NAT-PT). RFC 2766, Internet Engineering Task Force, February 2000.

[4] ETRI-PEC. User space NAT-PT implementation. http://www.ipv6.or.kr/english/natpt-overview.htm.

[5] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002.

[6] Henning Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. Request for Comments 1889, Internet Engineering Task Force, January 1996.

[7] M. Handley and V. Jacobson. SDP: Session Description Protocol. Request for Comments 2327, Internet Engineering Task Force, April 1998.

[8] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). Request for Comments 2326, Internet Engineering Task Force, April 1998.

[9] F. Cuervo, N. Greene, A. Rayhan, C. Huitema, B. Rosen, and J. Segers. Megaco protocol version 1.0. Request for Comments 3015, Internet Engineering Task Force, November 2000.

[10] VOVIDA. VOCAL home page. http://www.vovida.org.

Proceedings of the
Linux Symposium

July 23th–26th, 2003
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
Stephanie Donovan, *Linux Symposium*
C. Craig Ross, *Linux Symposium*

Review Committee

Alan Cox, *Red Hat, Inc.*
Andi Kleen, *SuSE, GmbH*
Matthew Wilcox, *Hewlett-Packard*
Gerrit Huizenga, *IBM*
Andrew J. Hutton, *Steamballoon, Inc.*
C. Craig Ross, *Linux Symposium*
Martin K. Petersen, *Wild Open Source, Inc.*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*