

An on Demand IPv4/IPv6 Multicast Translator

Tudor Mihai BLAGA, *Member, IEEE*, Virgil DOBROTA, *Member, IEEE*, Ferenc SZASZ and
Romeo VIDRASCU

Abstract — Several mechanisms were proposed to perform translation between IPv4/IPv6 packets in general. There are two dedicated approaches (MTP and UNINETT) designed for multicast traffic with implementations available only for Linux/Unix. This paper presents an original approach based on IPv4/IPv6 PIM-SM. The translator was developed for both, UDP packets and SAP/SDP (Session Announcement Protocol)/(Session Description Protocol) multicast session announcements. In order to save network resources the translation process starts only if there is at least one multicast receiver that wants the multicast content. The advantages of the implementation are the ease of installation and configuration, and the possibility of launching multiple instances of the service.

Keywords — multicast, PIM-SM, translation.

I. INTRODUCTION

During the transition to IPv6 it is expected that lots of IPv4 nodes will remain operational. IPv6-only nodes will appear, for cost reasons or as a result of exhaustion of the IPv4 address space, before IPv4 nodes disappear. Therefore, it is highly desirable to develop a mechanism which enables direct communication between IPv4 nodes and IPv6 nodes.

If the entities that communicate have applications and operating systems that support the same IP protocol version, there is usually no serious problem; except that one may have to establish tunnels through parts of the network. There is not much difference between unicast and multicast in this aspect. Not all tunnel techniques support multicast, but many do. For unicast one can try to make servers dual-stack so that IPv4-only and IPv6-only hosts can easily access them. Similarly, a multicast source could stream to both an IPv4 and an IPv6 group to allow all hosts to reach it. If the unicast server cannot be made dual-stack, hosts that support only the other protocol will

not be able to access it unless some translation is performed somewhere on the path between them.

Several types of translation solutions are available for multicast [1]:

- Multicast in dual-stack networks: When deploying multicast in a dual-stack network, one will probably want to deploy both IPv4 and IPv6 multicast. They are both mostly deployed using PIM-SM and the classical Any-Source Multicast (ASM) model with Rendezvous Points (RPs) and to some extent Source-Specific Multicast (SSM). There is no problem deploying both. Both routers and hosts can simultaneously do IPv4 and IPv6 multicast, and both can be done over the same network links. A router could also be an RP for both IPv4 and IPv6 groups.
- Application solutions: A single source is sending data to a number of clients. This can be audio and video streaming from a conference or from television. If RTP/RTCP is used, the receivers are also sources because they send RTCP reports. It is not critical to receive the reports though.
- Translation solutions: Translation can be performed in several ways. The idea is to have one or more translation devices on the path between sources that use one IP protocol and receivers that use another IP protocol. In some cases translation might also be carried out within the sending or receiving host. This can be useful on a dual-stack host where the application used only supports one IP protocol. In many cases the hardest part, when using translation devices, is to make sure that the translation device will be on the data path.

MTP (Multicast Translator Proxy) [2] and UNINETT [3] are translation mechanisms designed for multicast. The first one uses IGMP [4] and MLD [5] protocols and the second one uses PIM-SM [6] for IPv6 and IGMP for IPv4. Both use a special type of IPv6 addresses termed IPv4 compatible IPv6 multicast group address. The address is identified by a /96 prefix for IPv6 multicast and holds an IPv4 address in the low-order 32 bits [2].

An MTP device must be located between an IPv4 domain and an IPv6 domain, and it translates IPv4 multicast packets into IPv6 ones and vice versa. It consists of three sub-components: translator, IPv4 multicast proxy and IPv6 multicast proxy. The translator performs the conversion from IPv4 multicast to IPv6 multicast and vice versa. There are two translation modes: gateway and header conversion router. Gateway mode terminates data bound for an IPv4 multicast group at application layer, and relays the data to an IPv6 multicast group and vice versa. Header conversion router mode converts the IPv4 headers into IPv6 headers, fragments the IPv6 packets if necessary, and then forwards the packets. Likewise, when receiving IPv6 multicast packets, it converts the IPv6

Tudor Mihai BLAGA is with the Technical University of Cluj-Napoca, 400020 Constantin Daicoviciu 15, Cluj-Napoca, Romania (phone: +40-264-401816; fax: +40-264-597083; e-mail: tudor.blaga@com.utcluj.ro).

Virgil DOBROTA is with the Technical University of Cluj-Napoca, 400020 Constantin Daicoviciu 15, Cluj-Napoca, Romania (e-mail: virgil.dobrota@com.utcluj.ro).

Ferenc SZASZ is with the Technical University of Cluj-Napoca, 400020 Constantin Daicoviciu 15, Cluj-Napoca, Romania (e-mail: ferenc.szasz@gmail.com).

Romeo VIDRASCU is with the Technical University of Cluj-Napoca, 400020 Constantin Daicoviciu 15, Cluj-Napoca, Romania (e-mail: romeo.vidrascu@gmail.com).

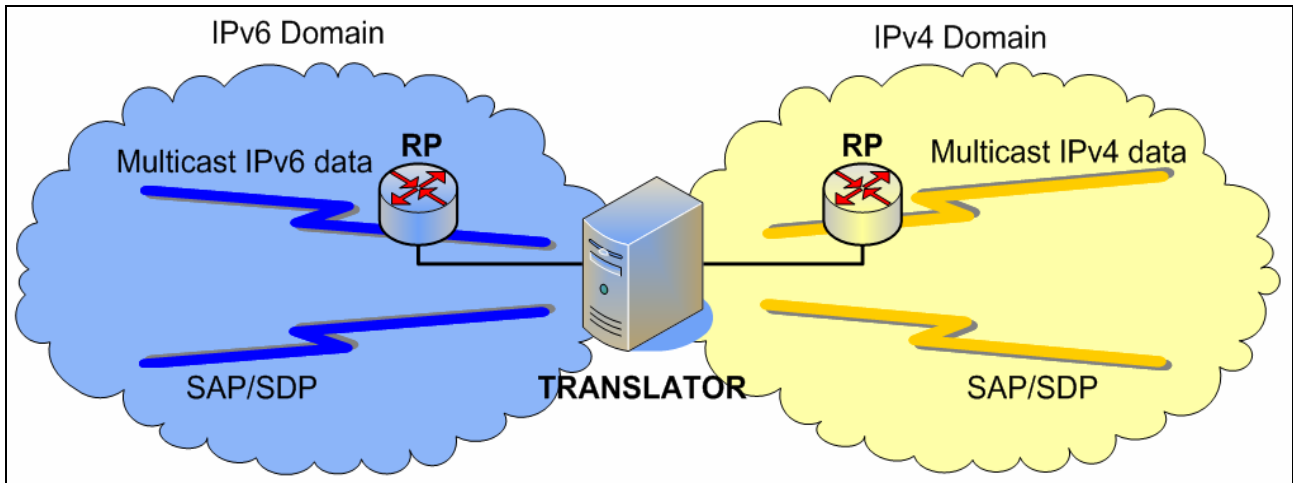


Figure 1. Translator operation

headers into IPv4 headers, and then forwards the IPv4 packets. The IPv4 multicast proxy joins the IPv4 multicast groups as a proxy for IPv6 receiver nodes and it receives packets bound for the IPv4 multicast groups, and then hands the packets to the translator. The IPv6 multicast proxy joins IPv6 multicast groups as a proxy of IPv4 receiver nodes and the received packets are sent to the translator. The disadvantages of this solution are that it requires configuration by the domain administrator and it can be applied only to small-scale networks [2].

UNINETT is developed by the Norwegian research network, and it works as a gateway between the IPv4 and IPv6 networks. It allows an IPv6 host full access to send and to receive from any IPv4 multicast group. The gateway can be deployed in an IPv6 PIM-SM domain, and the entire domain will be able to use it without any modification. The only requirement is that the gateway is the RP for a /96 multicast prefix. One major disadvantage of this approach is that IPv6 multicast is translated to the IPv4 domain even if there are no receivers for it [3].

II. TRANSLATOR ARCHITECTURE

The proposed translation mechanism allows IPv6/IPv4 hosts to send and to receive data to/from any IPv4/IPv6 multicast group. Traffic will be translated only on demand, if there are receivers for it in the IPv4/IPv6 domain. In the UNINETT case translation from IPv6 to IPv4 was performed even if no receivers were present. IPv4 compatible IPv6 multicast group address are used, but also arbitrary mappings between multicast address can be used. The administrator can configure these from the management module or from the configuration file.

The translator must be located on the edge between the IPv6 domain and the IPv4 domain, with interfaces configured for both IP versions (Figure 1). Its operation relies on information acquired from IPv4/IPv6 PIM-SM messages. This means that the translator must share the same link as the RP (Rendezvous Point) router from each domain. Thus it can keep track of all the IPv4/IPv6 sources and receive all their data. It will also know which groups there are listeners for, in either domain. From the implementation point of view this implies that the translator must include partial PIM-SM functionality, to be able to capture and understand the protocol messages. This provides a better solution than UNINETT, which requires the translator to perform all the functions of a PIM-SM RP for the multicast group.

An IPv6 Join PIM-SM message destined to the RP router or an MLD message on the local link causes the join of the IPv4 multicast group through IGMP. In a similar way an IPv4 Join PIM-SM or an IGMP message causes the join of the IPv6 multicast group through MLD.

As a result of sharing the same link as the RP router, the translator has information about all the active sources and receivers in both domains. All decisions regarding the starting or stopping the translation process is taken based on this information. Unfortunately UNINETT does not use PIM-SM in the IPv4 domain, so IPv6 multicast data is translated even if there are no receivers.

When a source starts sending the data will reach the translator. If the translator is on the same link as the source it will receive the packets natively and resend them as IPv4 multicast. Otherwise it will receive PIM Register messages that contain the multicast data that is resent as IPv4 multicast. When receiving Register messages, it may, join the IPv6 group to receive packets natively instead. These packets are also resent.

SAP/SDP session announcement messages are also translated together with the data traffic. The translator listens for these channels at the well-known IPv6 address FF0X::2:7FFE and IPv4 224.2.127.254 address. Only those messages are translated that refer to the data traffic.

III. IMPLEMENTATION ISSUES

The implementation of the proposed mechanism was split into two parts. One module performs the translation from IPv4 to IPv6 and the other from IPv6 to IPv4. This approach enabled the developers to work independently and as a result, depending on the intended use, we can install only one module. Also in the case of bidirectional translation between the IP domains the two modules can be installed on separate machines, thus providing fault tolerance.

The translator from IPv6 to IPv4 is implemented as a Windows Service. The service consists of 3 separate threads and has a Windows Installer attached to it called *setupserver64* (Figure 2).

The *verifyClient()* thread verifies by listening to PIM-SM Join/Prune messages if there are any IPv4 clients wanting to have access to IPv6 multicast content. A socket is created and bound to the ALL PIM ROUTERS address (224.0.0.13). The socket's type is Raw, so we have immediate access to the PIM message after the IPv4 header is removed. The analysis of PIM messages is absolutely crucial in the application. If a Join/Prune

message is received with the value of the *Number of Joined Sources* field different from zero and the value of the *Number of Pruned Sources* field equal to zero then the translation is started. If a Join/Prune message is received with the value of the *Number of Joined Sources* equal to zero and the value of the *Number of Pruned Sources* different from zero then the translation is stopped. In both case the *S*, *W*, *R* bits must have the value of 1. The value of the *Encoded Multicast Group Address* field must be the same as the address for which the translation is intended to be executed.

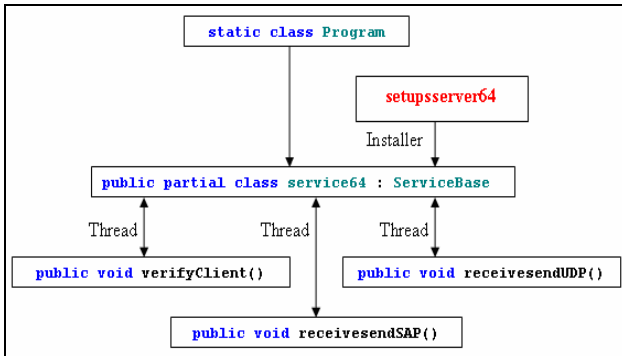


Figure 2. The structure of the IPv6 translator

The *receivesendUDP()* thread receives UDP packets from an IPv6 multicast group and sends them to an IPv4 multicast group. Two sockets are used, one for receiving, and one for sending these packets. The socket's type is datagram.

The sending and the receiving socket are made part of the multicast group. The sending socket needs to be modified because the default value of the Time To Live (TTL) field is 1, so the packet will not be forwarded by the first router.

The *receivesendSAP()* thread receives, modifies and sends SAP/SDP messages. Two sockets are used, one for receiving and one for sending these packets. Each socket's type is Raw, so we have immediate access to the SAP message after removing the IPv6 header and after removing the bytes corresponding to the SAP message, we can modify the SDP message, too.

The following modifications are required for every SAP/SDP message:

- The *IP address type* bit is changed,
- The *Authentication Length* field is checked,
- The *Message Identifier Hash* value remains unchanged,
- The *Originating Source* field is changed,
- The *Optional Authentication* field remains unchanged if exists,
- The *Payload Type* field remains unchanged,
- The "v=", "o=", "t=", "s=" and "c=" lines are changed or remain unchanged from the SDP message,
- The additional lines from the SDP message remain unchanged if they exist.

The parts and the connections between them, for the module that performs the translation from IPv4 to IPv6, are presented in Figure 3. There are three parts that make up the application: Translator, Manager and Monitor. The operation of the Translator is controlled by START/STOP

messages from the Manager, while the Monitor is used for configuration and operation monitoring based on the log files.

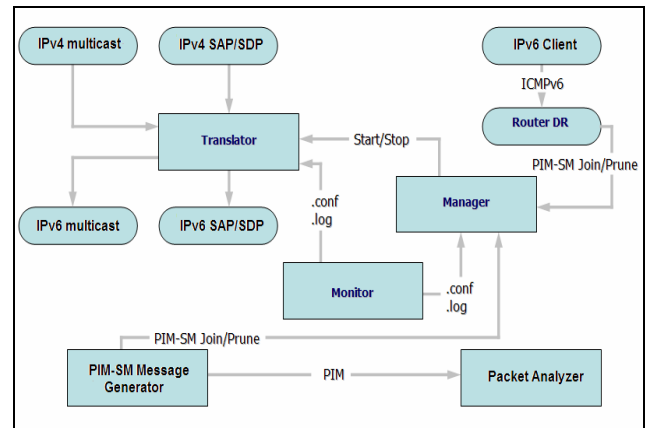


Figure 3. The structure of the IPv4 translator

Two applications were written to aid during the implementation stage: a PIM-SM Message Generator and a PacketAnalyzer. The first one creates PIM-SM message to be received by the Manager, thus replacing the need to deploy a full network with PIM-SM routers. The operation of PIM-SM Message Generator requires the use of NDIS Protocol Driver with the option Raw Packet enabled [7]. The second application was used to capture real PIM-SM messages and to verify that the generated ones are similar.

IV. CONFIGURATION ISSUES

The application that translates from IPv6 to IPv4 is installed by default in the "C:\Program Files\Feper\SetupSServer64\" folder. After the installation the configuration file: *fisierod.conf* must be edited according to the network setup (Figure 4) and multicast traffic. The configuration file has the following structure:

```

<Configuration file of the service. Do
not change the order of the parameters>
IPv6 local address:
2001::1:3
IPv6 multicast address:
FF0E::4444
IPv4 local address:
172.10.1.3
IPv4 multicast address:
233.11.11.11
IPv6 Port:
4444
IPv4 Port:
5555
IPv6 SAP address:
FF0E::2:7FFE
IPv4 SAP address:
224.2.127.254
SAP Port:
9875
PIM address:
224.0.0.13
PIM Listener Port:
3333
  
```

The *IPv6 local address* should be the global unicast address of the interface which is connected to the IPv6 network. The application works with local-link IPv6 address, too. The *IPv6 multicast address* is the multicast address of the IPv6 group for which the software translates the multicast content.

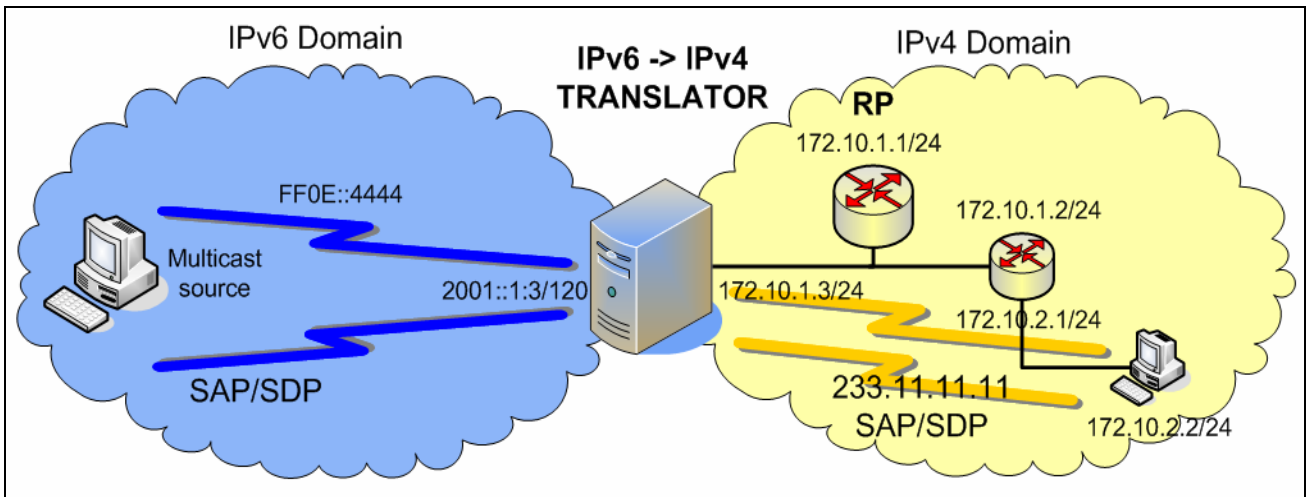


Figure 4. IPv6 to IPv4 translator testbed

The *IPv4 local address* should be the local IPv4 address of the interface which is connected to the IPv4 network. The *IPv4 multicast address* is the multicast address of the IPv4 group where the translator sends the multicast content.

The *IPv6 Port* specifies the port used to listen to the IPv6 multicast content. The *IPv4 Port* specifies the port used to send multicast content to the IPv4 multicast group. The *IPv6 SAP address* field identifies the IPv6 announcement channel. Usually two values can be used: *FF0E::2:7FFE* if global unicast addresses are used and *FF02::2:7FFE* if local-link addresses are used. The *IPv4 SAP address* field identifies the IPv4 announcement channel. Only one value is permitted: *224.2.127.254*.

The *SAP Port* field represents the port which is used to receive and send SAP/SDP messages. Only one value is permitted: *9875*. The *PIM address* field's value is the ALL PIM ROUTERS address: *224.0.0.13*. The *PIM Listener Port* specifies the port used to listen to the PIM messages.

The installation and configuration of the IPv4 to IPv6 translator is similar and is not presented here.

V. TESTBED ARCHITECTURE

The IPv6 to IPv4 translator testbed is presented in Figure 4. It uses two IPv4 PIM-SM routers with XORP (eXtensible Open Router Platform) [8]. One of them has only one interface, and it acts as the RP. This setup is known as RP-on-a-Stick [9]. The other one has two interfaces with a multicast receiver connected to IGMP interface.

In order to send multicast content the *VLC Media Player* version 0.8.4 was used [10]. This media player supports multicast both for IPv6 and IPv4. Packet were captured with *Ethereal* version 0.99 [11].

The first part of the test was aimed at verifying how the translation is performed. The original IPv6 packets were captured together with the translated IPv4 ones. Figure 5 represents the original UDP datagram which contains the multicast content that is received by the translator on its IPv6 interface. At the output we have the translated IPv4 multicast packet presented in Figure 6. As we can observe by comparing the two figures, several modifications were performed. The source address of the

modified datagram is the IPv4 address of the translator 172.10.1.3 and the destination address is the multicast IPv4 address 233.11.11.11, from the configuration file. The source port is generated dynamically by the operating system and the destination port is set from configuration file of this application. The total length of the modified packet is different from the length of the original packet because the IPv6 and IPv4 headers don't have the same length.

```

Internet Protocol Version 6
  version: 6
  Traffic class: 0x00
  Flowlabel: 0x000000
  Payload length: 1324
  Next header: UDP (0x11)
  Hop limit: 23
  Source address: 2001::1:3
  Destination address: ff0e:4444
User Datagram Protocol, Src Port: 1955 (1955),
  Dst Port: 4444 (4444)
  Source port: 1955 (1955)
  Destination port: 4444 (4444)
  Length: 1324
  Checksum: 0x6ca8 (correct)
  Data (1316 bytes)

```

Figure 5. IPv6 multicast packet

```

Internet Protocol, Src Addr: 172.10.1.3 (172.10.1.3),
  Dst Addr: 233.11.11.11 (233.11.11.11)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 1344
  Identification: 0x5709 (22281)
  Flags: 0x00
  Fragment offset: 0
  Time to live: 128
  Protocol: UDP (0x11)
  Header checksum: 0x3d80 (correct)
  Source: 172.10.1.3 (172.10.1.3)
  Destination: 233.11.11.11 (233.11.11.11)
User Datagram Protocol, Src Port: 4027 (4027), Dst Port: 5555 (5555)
  Source port: 4027 (4027)
  Destination port: 5555 (5555)
  Length: 1324
  Checksum: 0x226d (correct)
  Data (1316 bytes)

```

Figure 6. Translated IPv4 multicast packet

The translation of the SAP/SDP messages was also checked by capturing the original and the translated packet. The original announcement, valid only in IPv6

networks, is presented in Figure 7 and the IPv4 one in Figure 8.

The modified SAP/SDP message is generated by the translator based on the original announcement. The source port is generated dynamically by the operating system and the destination port must be the well-known port used by the SAP announcements 9875.

The flag field has the value of 00100000 which has the following meaning: the protocol's version is 1, the type of the source address is IPv4, the message is considered as an announcement of a session and it is not coded. The *Authentication Length* field's value is 0 so the *Optional Authentication Data* field is empty. The message identifier hash is 0x9a4f. These values were copied from the original announcement. The source address of the new announcement is the IPv4 address of the translator (172.10.1.3).

```

+ Internet Protocol Version 6
+ User Datagram Protocol, Src Port: 1953 (1953),
  Dst Port: 9875 (9875)
+ Session Announcement Protocol
  + Flags: 0x30
    Authentication Length: 0
    Message Identifier Hash: 0x9a4f
    Originating Source: 6b62:7073:2c20:3136:206b:487a:2c20:6d6f
    Payload type: no 1-pass CBR
+ Session Description Protocol
  Invalid line:
  + Owner/Creator, Session Id (c): - 92266404437 1432 IN IP6 ::
  Session Name (s): ch
  + Time Description, active time (t): 0 0
  + Connection Information (c): IN IP6 ff0e::4444/23
  + Media Description, name and address (m): video 4444 udp 33
  + Media Attribute (a): tool:v1c 0.8.4b
  + Media Attribute (a): type:broadcast
  + Media Attribute (a): x-plgroup:gr
  
```

Figure 7. IPv6 SAP/SDP packet

```

+ Internet Protocol, Src Addr: 172.10.1.3 (172.10.1.3),
  Dst Addr: 224.2.127.254 (224.2.127.254)
+ User Datagram Protocol, Src Port: 4026 (4026), Dst Port: 9875 (9875)
+ Session Announcement Protocol
  + Flags: 0x20
    Authentication Length: 0
    Message Identifier Hash: 0x9a4f
    Originating Source: 172.10.1.3
    Payload type: no 1-pass CBR
+ Session Description Protocol
  Session Description Protocol Version (v): 0
  + Owner/Creator, Session Id (c): - 92266404437 1432 IN IP4 172.10.1.3
  Session Name (s): ch
  + Time Description, active time (t): 0 0
  + Connection Information (c): IN IP4 233.11.11.11/23
  + Media Description, name and address (m): video 4444 udp 33
  + Media Attribute (a): tool:v1c 0.8.4b
  + Media Attribute (a): type:broadcast
  + Media Attribute (a): x-plgroup:gr
  
```

Figure 8. Translated IPv4 SAP/SDP packet

As well as the original message, this message begins with the $v=0$ line. The next line is an o type line which specifies the followings: the name of the multicast content source, the identifier and the version of this session, the network and IP address type. The c line specifies the parameters of the connection: the network type, the IP address type and the address of the connection (233.11.11.11/23). Generally a session descriptor contains one or more media sections. Every media section starts with a c line and its last line is of m type. In this message the media attribute is set to "video", the transport attribute is set to UDP which represents the transport layer protocol.

The same verification process was performed for the IPv4 to IPv6 translation module, and due to the similarities it is not presented here.

VI. TRANSLATION DELAY

Once the proper operation of the translator was proved, we determined the translation delay introduced. The translation delay is the time passed from the moment a packet was received to the moment a packet containing the same data is sent using a different IP version.

Table 1 presents the time of arrival and of sending the data packet, and the difference between them gives the translation delay. This delay and its average is represented as a chart in Figure 9. The average value is 0.598 ms, with a minimum of 0.405 ms and a maximum value of 0.971 ms.

Table 1. Translation delay from IPv6 to IPv4

| Nr. | Arrival(s) | Sending(s) | Delay (ms) |
|-----|------------|------------|------------|
| 1. | 54.652702 | 54.653243 | 0.541 |
| 2. | 54.668098 | 54.668559 | 0.461 |
| 3. | 54.683724 | 54.684129 | 0.405 |
| 4. | 54.699355 | 54.699862 | 0.507 |
| 5. | 54.699581 | 54.700167 | 0.586 |
| 6. | 54.715009 | 54.715783 | 0.774 |
| 7. | 54.732412 | 54.733269 | 0.857 |
| 8. | 54.732630 | 54.733601 | 0.971 |
| 9. | 54.746225 | 54.746668 | 0.443 |
| 10. | 54.761867 | 54.762308 | 0.441 |

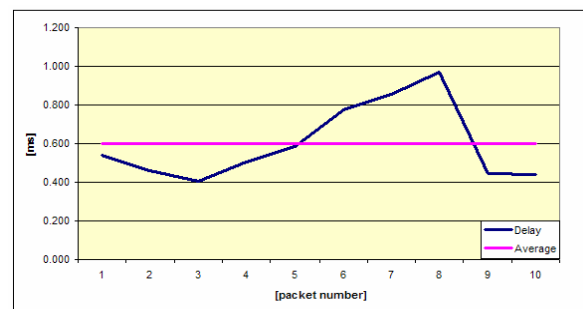


Figure 9. Delay for multicast data IPv6 to IPv4

Table 2 represents the difference between the arrival time of an original SAP/SDP message and the moment of sending of the modified SAP/SDP message. This difference is in fact the translation delay. The delay and its average is represented as a chart in Figure 10. The average processing time of the SAP/SDP messages is 0.604 ms

Table 2. Translation delay for SAP/SDP from IPv6 to IPv4

| Nr. | Arrival(s) | Sending(s) | Delay (ms) |
|-----|------------|------------|------------|
| 1. | 56.605677 | 56.606364 | 0.687 |
| 2. | 61.683797 | 61.684634 | 0.837 |
| 3. | 66.636935 | 66.637600 | 0.665 |
| 4. | 71.618849 | 71.619234 | 0.385 |
| 5. | 76.699393 | 76.700113 | 0.720 |
| 6. | 81.652544 | 81.653270 | 0.726 |
| 7. | 86.642094 | 86.642493 | 0.399 |
| 8. | 91.746311 | 91.746720 | 0.409 |

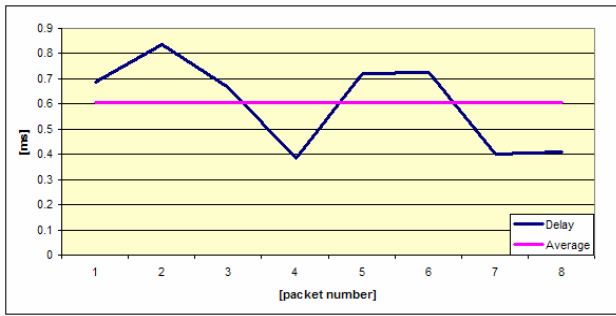


Figure 10. Delay for SAP/SDP IPv6 to IPv4

The charts for the translation delay in the case of IPv4 to IPv6 translation are presented in Figure 11 and Figure 12. The average for the multicast data is 0.987 ms, with the highest value of 2.3 ms, and for SAP/SDP it is 2.023 ms, with the highest value of 7.9 ms.

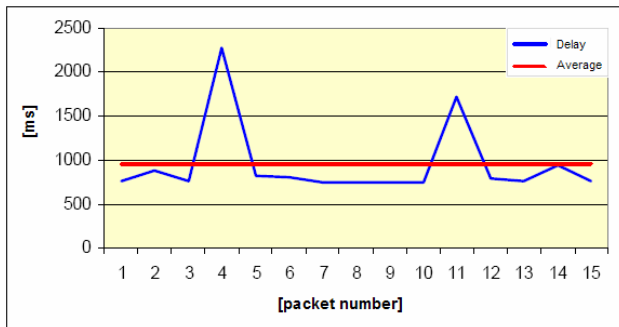


Figure 11. Delay for multicast data IPv4 to IPv6

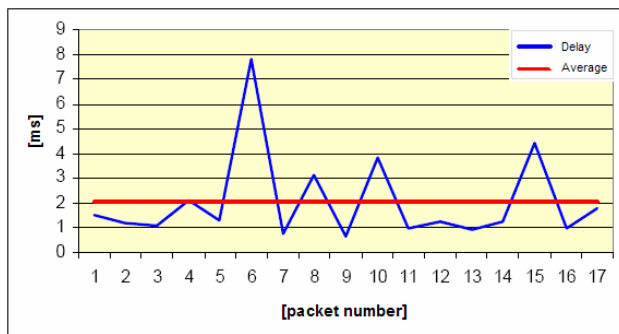


Figure 12. Delay for SAP/SDP IPv4 to IPv6

We observe a substantial difference in the case of transition from IPv4 to IPv6 compared to the reverse translation. This can only be explained by the differences in the software architecture of the two translation modules.

The values obtained for the translation delay could be used to compare the performance of our IPv4/IPv6 translator to other proposals. This was not possible due to the lack of functional implementations for MTP or UNINETT.

VII. CONCLUSION

The proposed mechanism uses PIM-SM to enable the translation of multicast data between domains using different IP versions. The translation process is started on-demand, only if there are IPv4/IPv6 clients that request the multicast data.

This software is implemented as a Windows Service (it can be started or stopped and works in the background), with different modules for each translation direction. This allows them to be used independently, depending on the requirements, and provides a fault protection mechanism, the two modules can be installed on separate machines. A multithreaded solution was adopted for receiving and the sending the UDP packets. The SAP/SDP message modification and the PIM-SM listener module are implemented in a separate thread.

We determined the translation delay cause by the implemented modules. For IPv4 to IPv6 data the average was 0.598 ms and for SAP/SDP it was 0.604 ms. For IPv6 to IPv4 data the average was 0.987 ms and for SAP/SDP it was 2.023 ms. This difference can only be explained by the independent implementation of the two software translation modules.

The values obtained for the translation delay could be used to compare the performance of our IPv4/IPv6 translator to other proposals. This was not possible due to the lack of functional implementations for MTP or UNINETT.

REFERENCES

- [1] L. Rossi and A. Pinizzotto, *IPv4/IPv6 Multicast Interoperability*. 2003, Consortium GARR (IIT-CNR): 6NET.
- [2] K. Tsuchiya, et al., *An IPv6/IPv4 Multicast Translator based on IGMP/MLD Proxying (mtp)*. 2002, www.ietf.org.
- [3] S. Venaas, *An IPv4 - IPv6 Multicast Gateway*. 2003: www.ietf.org.
- [4] B. Cain and S. Deering, *Internet Group Management Protocol, Version 3*. 2002.
- [5] R. Vida and L. Costa, *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*. 2004.
- [6] D. Estrin and D. Farinacci, *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*. 1998.
- [7] NDIS, <http://www.codeproject.com/cs/internet/sendrawpacket.asp>, in *Raw Ethernet Packet Sending*. 2006.
- [8] XORP. *eXtensible Open Router Platform* <http://www.xorp.org/>. 2006.
- [9] Beau Williamson, *Developing IP Multicast Networks*. Vol. Volume 1. 2001: Cisco Press.
- [10] VLC. *VLC - the cross-platform media player and streaming server* <http://www.videolan.org/vlc/>. 2006.
- [11] Ethereal. *A Network Protocol Analyzer* <http://www.ethereal.com/>. 2006.