Protocol Verification of Translation in Mobile Internet Protocol Version 4 and 6

Susanna S. Henry[1]*, B. Vijay Kumar[1], V. Santhosh Kumar[1], Gurwinder Singh[2]
[1] CS Dept, BPDC, Dubai, UAE.
[2] EEE Dept, BPDC, Dubai.

* Corresponding author. Tel.: 00971 55 7872662; email: susan@dubai.bits-pilani.ac.in
Manuscript submitted April 29, 2015; accepted June 20, 2015.
doi: 10.17706/jcp.11.2.149-158

Abstract: Verifying a protocol means ensuring that it is free of logical errors prior to implementing it. The goal of verifying the protocol is to guarantee that protocol does exactly what the designer intended. This paper presents the methodology to verify protocols in transition phase of Mobile IP. Due to insufficient addresses in IPv4 there is an urgent need to switch to IPv6. Translation is an effective technique that enables IPv4-only devices to communicate with IPv6-only devices. This paper deals with operation of Address translation & Mapping technique in Translation. The Design Analysis in this paper demonstrates methodology to verify translation of IPv4 data packet to IPv6 data packet.

Key words: Address translation, domain name system, mapping, stateless and stateful translation.

1. Introduction

Protocol verification is a crucial step to eliminate weaknesses and inaccuracies of effective network protocols [1]. Protocols can be verified against their design and implementation. In simple words, protocol can be verified during the initial phase before the system is implemented and since protocol verification can detect errors at the early stage, unnecessary or incorrect implementation can be avoided. Therefore, protocol verification has the potential to significantly reduce the cost of protocol development and testing [2]. IPv4 uses 32 bits addresses and can support 4,294,967,296 IPv4 addresses but all these IP addresses have now been allocated to various organizations and institutions [3]. Whereas, IPv6 utilizes 128-bits and can support 2128 addresses which count up to 340,282,366,920,938,463,463, 374,607,431,768,211,456 IPv6 addresses [4].

Translation refers to the direct conversion of protocols. The IETF has considered Translation as the most sensible transition model. IPv4/IPv6 Translation enables networks to have IPv4 and IPv6 coexist to a certain extent in a balanced manner while transitioning to an IPv6-only network.

Section 2 deals with details of Translation and Section 3 presents the methodology to verify protocols. Future scope is presented in Section 4 and conclusion is given in Section 5.

2. IPV4/IPV6 Translation

2.1.1. Scenarios for IPv4/IPv6 Translation

Currently, the proposed solutions for IPv6/IPv4 translation are classified as:

1) IPv6 network to the IPv4 Internet

2) IPv4 Internet to an IPv6 network
3) IPv6 Internet to an IPv4 network
4) IPv4 network to the IPv6 Internet
5) IPv6 network to an IPv4 network
6) IPv4 network to an IPv6 network
7) IPv6 Internet to the IPv4 Internet
8) IPv4 Internet to the IPv6 Internet

An IPv6 Network to the IPv4 Internet: Due to the lack of IPv4 addresses or due to other technical and economic constraints, the network is IPv6-only, but the hosts in the network require communicating with the global IPv4 Internet [5].

Ex: Wireless NW, enterprise N/W, sensor N/W.

The IPv4 Internet to an IPv6 Network: When the enterprise networks or ISP networks adopt Scenario 1, the IPv6-only users will not only want to access servers on the IPv4 Internet but also will want to setup their own servers in the network that are accessible by users on the IPv4 Internet. Thus, with a translation solution for this scenario, the benefits would be clear. Not only could servers move directly to IPv6 without trudging through a difficult transition period, but they could do so without risk of losing connectivity with the IPv4-only Internet.

The IPv6 Internet to an IPv4 Network: An inheritance of IPv4 network is required to provide services to IPv6 hosts. In this case, a Network Specific Prefix assigned to the translator will give the hosts unique IPv4-converted IPv6 addresses. There is no need to synthesize AAAA from A records, since static AAAA records can be put in the regular DNS to represent these IPv4-only hosts.

An IPv4 Network to the IPv6 Internet: Due to technical or economic constraints, the network is IPv4-only, and IPv4-only hosts (applications) may require communicating with the global IPv6 Internet. This scenario will probably only occur when we are well past the early stage of the IPv4, and the IPv4/IPv6 transition has already moved to the right direction [6].

An IPv6 Network to an IPv4 Network: In this scenario, both an IPv4 network and an IPv6 network are within the same organization. The IPv4 addresses used are either public IPv4 addresses or private addresses. The IPv6 addresses used are either public IPv6 addresses. The stateful and stateless translation schemes apply here.

An IPv4 Network to an IPv6 Network: This is another scenario when both an IPv4 network and an IPv6 network are within the same organization. The IPv4 and IPv6 addresses used are either public IPv4 addresses or private addresses or ULAs (Unique Local Addresses).

The IPv6 Internet to the IPv4 Internet: Any IPv6-only host or application on the global Internet can initiate communication with any IPv4-only host or application on the global Internet. Due to the huge difference in size between the address spaces of the IPv4 Internet and the IPv6 Internet, there is no viable translation technique to handle unlimited IPv6 address translation.

The IPv4 Internet to the IPv6 Internet: This case is very similar to Scenario 7. The analysis and conclusions for Scenario 7 also apply for this scenario [7].

2.2. Classification of Proposed Solution for IPv4/IPv6 Translation

Currently, the proposed solutions for IPv6/IPv4 translation are classified [8] into 1. Stateless translation and 2. Stateful translation.

Stateless Translation: For stateless translation, translation information is carried in the address and configuration information is carried in the translators which permit the initiation of both IPv4->IPv6 & IPv6->IPv4 sessions. Stateless translation supports end-to-end address transparency and has better scalability compared with stateful translation.

Stateful Translation: For stateful translation, the translation state is maintained between IPv4 and IPv6 address/port pairs, enabling IPv6 systems to open sessions with IPv4 systems [9].

The following points must be considered before choosing stateless or stateful translation.

1) No matter the direction is IPv6->IPv4 or IPv4->IPv6, a technology is needed that will permit systems acting as clients to be able to open sessions with other systems acting as servers. This in fact is stateless Translation [10]. Majority of accesses will be to servers in a carrier infrastructure which optimizes access to them. However, if the complexity is reduced and a stateless algorithm cannot be developed, a stateful algorithm is acceptable.
2) Whether the direction is IPv6->IPv4 or IPv4->IPv6, a technology is needed that will permit peers to connect with each other and if this was stateless it would be an ideal case [11]. However, if the complexity is reduced and a stateless algorithm cannot be developed, a stateful algorithm is acceptable [12].
3) Stateless or Stateful algorithm is not required in some cases to enable connections to the hosts where hosts are purely clients.

3. Protocol Verification

This section presents the methodology applied for Protocol Verification of Translation in MIPv4 / MIPv6. The methodology presented here consists of following steps,

1) Modeling of Translation in MIPv4 / MIPv6 using schema diagram.
2) Formulating design steps for each module from schema diagram.
3) Implementing each design step using Programming language C.
4) Noting down experimental results using screen shots.

3.1. Modeling of Translation in MIPv4 / MIPv6 Using Schema Diagram

Translation refers to the direct conversion of protocols. The network environment in Translation consists of translators that are responsible to convert AAAA records of IPv6 to A records of IPv4 and vice versa. In this thesis, the network scenario is assumed to be in IPv6 -> IPv4 direction which consists of IPv6 to IPv4 translators. This section presents the schema diagram developed for Translation model as shown in Fig. 1.

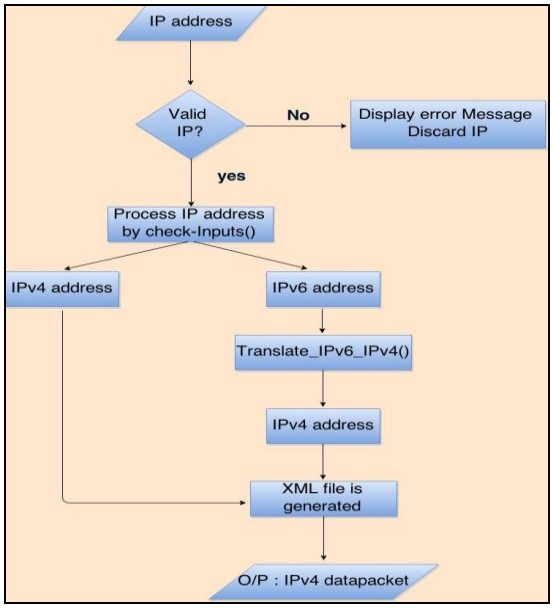

Fig. 1. Schema diagram of translation model.

3.2. Formulating Design Steps for Each Module from Schema Diagram

This section presents the design steps to demonstrate translation of IPv6 address to IPv4 address. This design will be implemented using the function TRANSLATE_IPv6_IPv4( ). After the translation process is complete, xml file gets generated to show the IP datapacket with translated IP address. Every datapacket can be classified into four categories as shown in Table 1.

Table 1. Datapacket with Source Address and Destination Address

Datapacket	Source address	Destination address
Case i.	IPv4	IPv4
Case ii.	IPv6	IPv6
Case iii.	IPv4	IPv6
Case iv.	IPv6	IPv4

During address translation, this design makes use of an internal table named IPADDRESSMAP.txt. The translator in this design translates by comparing the IPv6 address provided by the user with the IPv6 address present in the internal table. If match is found, corresponding IPv4 address is used for translation else default address gets selected. The function TRANSLATE_IPv6_IPv4( ) is used to translate IPv6 address to IPv4. After translation, XML file is generated which contains the translated IP datapacket.

3.3. Implementing Each Design Step Using Programming Language C

Function 1: TRANSLATE_IPv6_IPv4( )

Case i:

INPUT : Datapacket with source address 173.194.113.145 and destination address 50.97.149.10

OUTPUT: XML file with IPv4 datapacket as shown below

<IPv4>
<SOURCE_IP> 173.194.113.145 </SOURCE_IP>
<DESTINATION_IP> 50.97.149.10 </DESTINATION_IP>
<BODY > MESSAGE </BODY>
</IPv4>

Case ii:

INPUT: Datapacket with source address E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420 and destination address EEEE:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:EEEE

OUTPUT: Default IPv4 address gets selected and respective IPv4 datapacket gets generated as shown in XML file

<IPv4>
<SOURCE_IP> 101.101.101.101 </SOURCE_IP>
<DESTINATION_IP> 101.101.101.101 </DESTINATION_IP>
<BODY > MESSAGE </BODY>
</IPv4>

Case iii:

INPUT: Datapacket with source address 173.194.113.145 and destination address E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420

OUTPUT: Default IPv4 address gets selected for destination address and respective IPv4 datapacket gets generated as shown in XML

file
<IPv4>
<SOURCE_IP> 173.194.113.14 </SOURCE_IP>
<DESTINATION_IP> 101.101.101.101 </DESTINATION_IP>
<BODY > MESSAGE </BODY>
</IPv4>

Case iv:

INPUT: Datapacket with source address E3D7:0000:0000:0000:51F4:9BC8:C0A8:6420 and destination address 173.194.113.145

OUTPUT: Default IPv4 address gets selected as source address and respective IPv4 datapacket gets generated as shown in XML file

<IPv4>
<SOURCE_IP>101.101.101.101</SOURCE_IP>
<DESTINATION_IP>173.194.113.145 </DESTINATION_IP>
<BODY > MESSAGE </BODY>
</IPv4>

3.4. Noting down Experimental Results Using Screen Shots

Case i: Source and Destination address both are IPv4 and an IPv4 datapacket is sent for Translation as shown in Fig. 2.

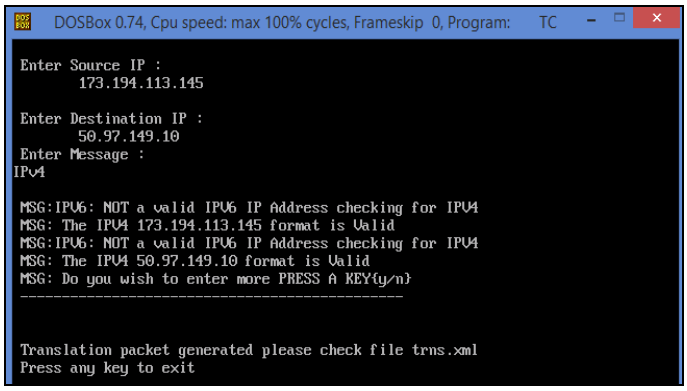

Fig. 2. IPv4 datapacket is sent for translation.

On providing valid IPv4 address for Source IP address and Destination IP address, the code will check for the validity of the IP address given as input. This step is carried out to check if the source IP address is a valid IPv4 address or a valid IPv6 address. As the source and destination IP address provided by the user is an IPv4 address, hence an IPv4 datapacket is generated as shown in the xml file (Fig. 3).

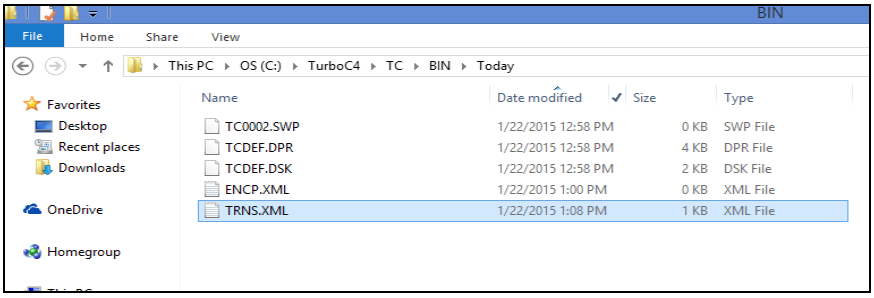

Fig. 3. XML file is generated.

The code used in this paper translates an IPv6 datapacket to IPv4 datapacket. But as the input given by

the for source and destination address is IPv4, hence Translation is not carried out and hence an IPv4 datapacket is generated as shown below in Fig. 4.

Case ii: In this case, the input given by the user for Source and Destination address is an IPv6 address, next step is to check if the source IP address is a valid IPv4 address or a valid IPv6 address. Since the input Source and destination is IPv6 address, an IPv6 datapacket is generated which is sent for Translation as shown below in Fig. 5.

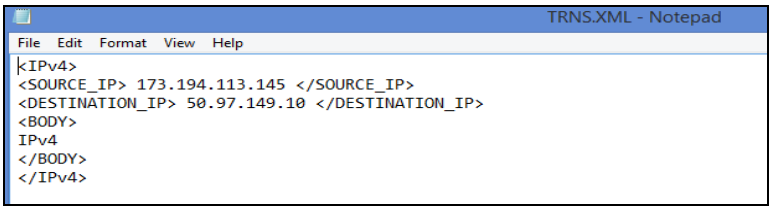

Fig. 4. Contents of XML file.

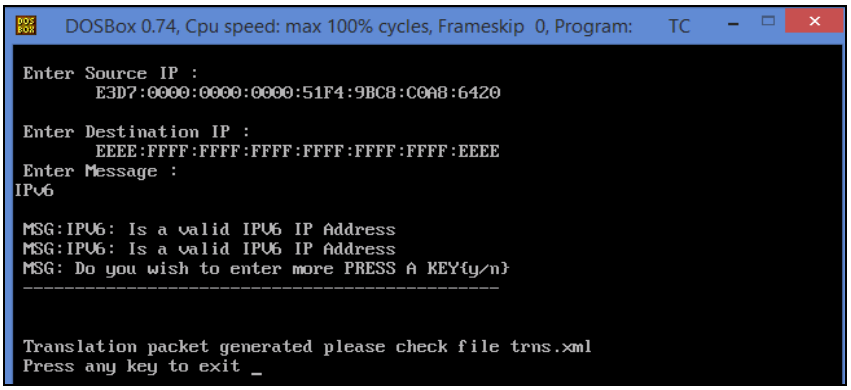

Fig. 5. IPv6 datapacket sent in IPv4 network.

Translation is successful and xml file is generated as shown in Fig. 6.

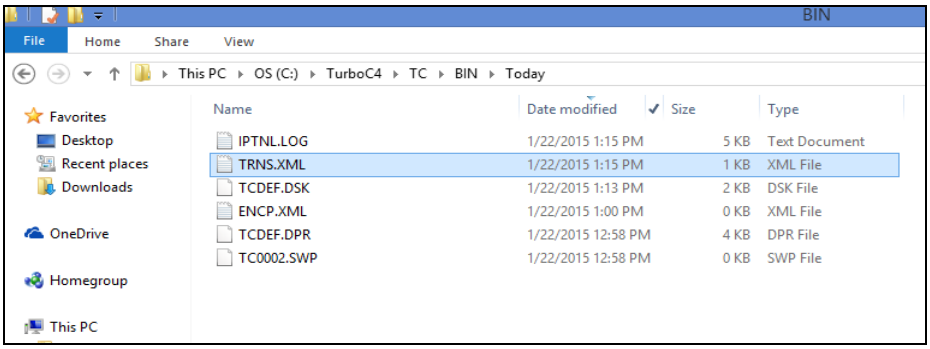

Fig. 6. XML file is generated.

Default IPv4 address gets selected and IPv4 datapacket is generated as shown below Fig. 7.

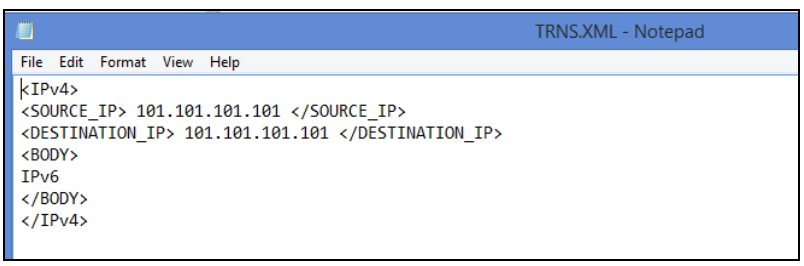

Fig. 7. IPv6 datapacket is translated to IPv4 datapacket.

Case iii: Datapacket with IPv4 Source Address and IPv6 Destination Address is sent for Translation as shown in Fig. 8.

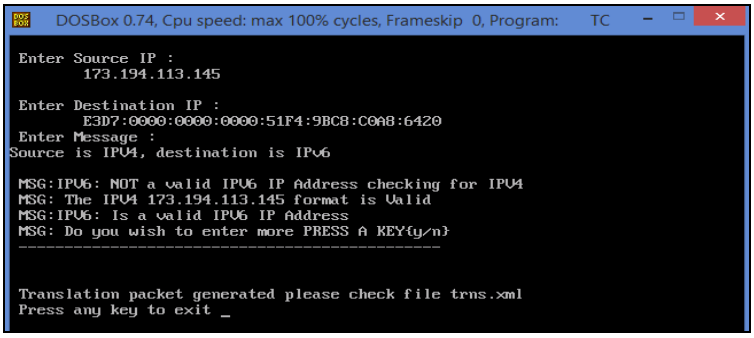

Fig. 8. Translation is complete.

After translation, XML file is generated, and the source address remains same while IPv6 destination address is translated to IPv4. The translated datapacket is shown below Fig. 9.

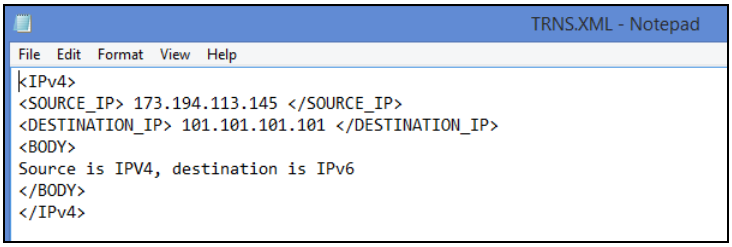

Fig. 9. Default IPv4 address is selected as destination address to generate IPv4 datapacket.

Case iv: Similarly when source address provided is IPv6 and Destination address is IPv4, a datapacket is generated which has an IPv6 source address and IPv4 destination address. This datapacket is sent for Translation as shown in Fig. 10.

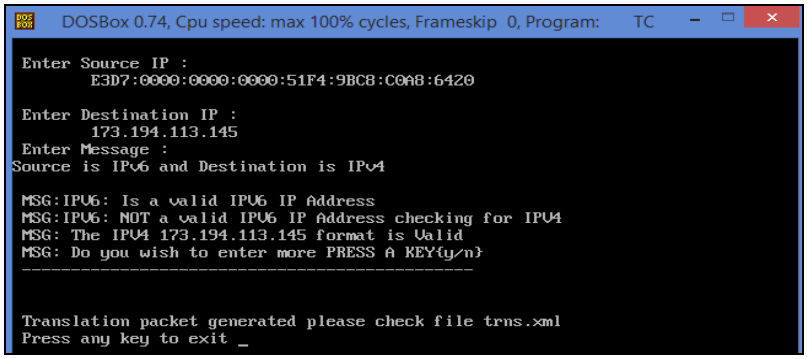

Fig. 10. Translation is complete.

After translation, XML file is generated in which the IPv6 source address is translated to IPv4 while the IPv4 destination address remains the same. The generated IPv4 datapacket is as shown in Fig. 11.

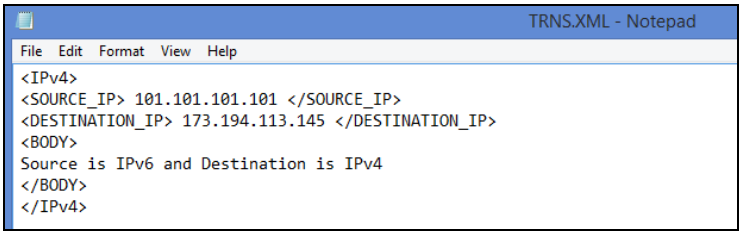

Fig. 11. Default IPv4 address is selected as source address to generate IPv4 datapacket.

4. Conclusion

4.1. Conclusion

Various types of Translation was described in detail and a model was developed to represent Translation. Design steps were formulated to translate IPv6 address to IPv4 address. The datapackets were viewed using a xml file. Four different cases for each data packet were identified as follows

Case 1: Source address :IPv4 Destination address: IPv4

Case 2: Source address :IPv6 Destination address: IPv6

Case 3: Source address :IPv4 Destination address: IPv6

Case 4: Source address :IPv6 Destination address: IPv4

In each case, the data packets with IPv6 address was successfully translated to IPv4 address. IPv4/IPv6 Translation is a vast quantitative aspect due to its eight different types of classifications. The internal mapping table used here helps to translate IPv6 addresses to IPv4 address. Thus, the design steps formulated provides an easy technique to model Translation.

4.2. Future Scope

The methodology that we suggested in this paper presents a new set of approach that will enable to verify any given protocols. The presence of logical errors due to alterations in the assumptions and environment of the given protocol may lead to one or more statements or entire routines that will never be executed. This causes unnecessary work for the network designers and may result in discarding well developed protocol after its implementation. Our design methodology presents an approach that allows analyzing important properties of the protocol under consideration. The results obtained eventually help in synthesis of specific systems. It allows improving the correctness and the performance of a system. Despite the existing wide variety of communication protocols, many new technologies are evolving. Because of this, the protocols, which define the network communication, are highly inter-related. Many protocols rely on others for operation. For example, many routing protocols use other network protocols to exchange information. Thus there is a need to verify these protocols. Protocol Verification would be valuable for verifying the intent of the network designer, troubleshooting, and performing “what-if” analysis of failure scenarios. However verification can never guarantee total correctness but it can increase or decrease confidence in the model by showing presence of bugs. It is always good enough to find bugs at the initial stage rather than to discard a well-developed protocol post its implementation. It is necessary for protocol designers to have techniques and tools to detect errors in the early phase of design, because the later a fault is discovered in any process, the greater the cost of rectifying it.

Appendix

Source Code - Protocol Verification of Translation in MIPv4 / MIPv6

```
#include <stdio.h>
void TRANSLATE_IPv6_IPv4();
char * getipv4addr(char inaddress[]);
int debug = 1;
const FILE *const_fp, *decap_file, *trans_file;
void main()
{
//IPV6 IP ADDRESS FORAMT - E3D7:0000:0000:0000:0001:0001:0001:0001
char source[40];
char destination[40];
char var_caseinput = 0;
```

```
//Open log file at global level
const_fp = fopen( "IPTNL.log", "a+" );
fprintf(const_fp,"\n\n\n###############NEWRUN############################\n")

clrscr();
printf("\n\n IP TUNNEING AND TRANSLATION \n");
printf("----------------------------------");
printf("\n\n MESSAGE SENDING FUNCTIONALITY");
printf("\n Press: 1 For Encapsulation For IP Tunnelling");
printf("\n Press: 2 For IP Translation");
printf("\n\n MESSAGE RECEVING FUNCTIONALITY \n");
printf("\n Press: 3 For Decapsulation For IP Tunnelling");
printf("\n Press: 4 For CSV For IP Translation \n");
printf("\n Please provide input _ ");
var_caseinput = getch();
clrscr();
switch(var_caseinput)
{ case '2':
fprintf(const_fp, "Selected 2 For IP Translation\n");
//Populate IPV6 stack only (previous functionality)
generate_packet_data(0);
printf("\n\n Translation packet generated please check file trns.xml ");
break;
default:
printf("\n Incorrect option code will exit");
fprintf(const_fp, "Incorrect option code will exit\n");
break; }
```

Acknowledgment

Ms. Susanna S Henry thanks Dr. B. Vijay Kumar and Dr. V.Santhosh Kumar for their valuable suggestions and also thanks BITS, Pilani Dubai Campus for providing with financial support.

References

[1] Martina, J. E. (2012). *Verification of Security Protocols Based on Multicast Communication*. Technical Report, Number 816. University of Cambridge. Computer Laboratory.

[2] Lin, F. J., Chu, P. M., & Liu, M. T. (1987). Protocol verification using reachability analysis: The state space explosion problem and relief strategies. *ACM SIGCOMM Computer Communication Review*, *17(5),* 126-135.

[3] Perkins, C. (Aug. 2002). IP mobility support for IPv4. *RFC 3344.*

[4] Johnson, D., Perkins, C., & Arkko, J. (June 2004). Mobility support in IPv6*. RFC 3775*.

[5] Baker, F., Li, X., Yin, K., & Bao, C. (April 2011). Framework for IPv4/IPv6 translation. *RFC 6144*.

[6] Tsirtsis, G., & Srisuresh, P. (February 2000). Network address translation — protocol translation (NAT-PT). *RFC 2766*.

[7] Levkowetz, H., & Vaarala, S. (April 2003). Mobile IP traversal of network address translation (NAT) devices. *RFC 3519*.

[8] Cisco Systems. From http://www.cisco.com

[9] Waddington, D. G., & Chang, F. Z. (June 2002). Realizing the transition to IPv6. *IEEE Communications Magazine*, Bell Research Laboratories.

[10] Sellers, C. (April 2009). *IPv6 Transition Mechanisms and Strategies*, NTT Communications. NTT America, Inc. New York, NY 10017. U.S.A. 212-661-0810.

[11] Li, X., Bao, C., & Baker, F. (April 2011). IP/ICMP translation algorithm. *RFC 6145*.

[12] Perreault, S. (November 5-6, 2009). DNS64 and NAT64. *Proceedings of IPv6 Migration Workshop for IETF and 3GPP*. Shanghai, China.

Susanna S. Henry is a senior lecturer of computer science at Bits Pilani Dubai Campus. She has 8 years of experience in teaching computer science and electronics & communication engineering students. She has a good knowledge of various computer applications and ability to provide the best practice to the students, capacity to solve the problems pertaining to the computer programs in C language. She is also a research scholar currently working on the topic of mobile internet protocol version 4 and 6.

B. Vijayakumar is currently the H.O.D of Computer science Dept at BITS, Pilani Dubai Campus. He has 20 years of teaching experience and more than 5 years of industrial experience. He has more than 25 journals published under his contribution.

Vasudevan Santhosh kumar is an assistant professor in the Computer Science Dept. at Bits Pilani Dubai Campus. He has a vast knowledge in the field of data mining with several years of experience in Industries. He has 6 years of teaching experience.

Gurwinder Singh is currently pursuing B.E (honors) electrical and electronics from Bits Pilani, Dubai. He has successfully completed 2 months internship at Verger Delporte, Sharjah, UAE. His research interests include network address translation & mobile IPv6.