A platform for porting IPv4 applications to IPv6

Yasir Mahmood[1], Ayad Abdulqader[2]
[1]*university of Mosul, yaser.ali@uomosul.edu.iq;* [2]*university of Mosul, ayad_alezzi@uomosul.edu.iq*

Abstract: Developing a new application goes through several stages which need a lot of effort from analysts, designers, and programmers which results in large sums of money as well as wasted time. It is also known that currently there are a lot of applications that only support IPv4 network. Building a new application from scratch that supports the IPv6 network is very expensive. Re-engineering these applications that support the IPv4 network to make them support the IPv6 network is the best solution to reduce effort and cost. The aim of this paper is to design and implementation of a platform used for automatic porting the C++ IPv4 network applications to IPv6, by using partial re-engineering approach the system is only partially re-engineered, the re-engineered system portion shall be integrated with the current non re-engineered portion, this reduces potential system improvements. the main process of the porting is done by replacing all the IPv4 dependent statement with its corresponding IPv6 dependent statement. Furthermore, replacing all constant value of IPv4 addresses to the suitable IPv6. The proposed platform will reduce the cost for porting network applications to support IPv6 in different proportion according to the size of the application to be ported.

Keywords: IPv4; IPv6; porting; re-engineering; platform.

1. Introduction

Every device connected to the internet needs a logical address in order to be defined on the Internet and this logical address is used in the process of directing information to the concerned device, and given that the number of devices connected to the internet is increasing, and this needs logical addresses for all these devices connected in the internet The current protocol used is the IPv4 protocol, as in the very near future this protocol cannot meet the needs of addresses for new devices, and this has become the need for a protocol that meets these needs, and here comes the role of IPv6 protocol to solve this problem as this new protocol Provides more logical addresses than IPv4.

Because the IPv4 cannot meet the needs of the current internet, the use of the IPv6 is necessary to solve the problems encountered by the IPv4.

Nowadays, the applications spread over the internet use the ipv4 network to communicate with servers or to communicate with each other.

This paper proposes the design and implementation of a porting platform, which port applications (written for the IPv4 network) to applications working on the ipv6 network with the lowest cost and effort, instead of building a new application from scratch that needs a great effort, which in turn produces a huge cost.

2. literature review

some expert researchers, [1] including companies and academic institutions, have been inclined to port existing applications to IPv6. In reference [2] IBM published an article; this article explores the principles behind a simple IPv6 system in particular how IPv6 addresses address space problems and large routing tables. the article is also about tunneling, mapped addresses, and porting IPv4 to IPv6 applications. In reference [3] This document outlines the improvements that must be made for an IPv4-compatible application to run within a network environment of IPv6. In reference [4] This paper discusses the transition from IPv4 to IPv6 network applications and the way to port the IPv4 broadcast application to IPv6. This IPv6 does not have an implemented broadcasting concept, Including a number of issues during the transition from the platform to IPv6, such as the easiest and easiest approach to the transition process. In reference [5] This article describes the author experiences in porting SIP

Email: yaser.ali@uomosul.edu.iq contact number:9647704151987

to IPv6. In reference [6] This article explains how IPv6 applications can be introduced, and how IPv4 applications can be ported to IPv6, and what specific application porting problems should be considered. In reference [7] This paper is intended to discuss different issues when porting an IPv4 application to IPv6 with an emphasis on issues facing an application developer. In reference [8] This document contains instruction with some examples of source code porting, used to illustrate the required changes in the client and server components. In reference [9] This paper includes the experiences on a case study for porting applications to IPv6. it presents the results of the effort to port OpenH323, an open-source H.323 platform to IPv6, that it can serve as guidelines for other projects with similar goals. In reference [10] The paper concludes with general recommendations for new IPv6 applications and provides some examples of code porting processes. In reference [11] This paper explains the application scenarios sensitive to changes to the IP protocol and the solutions that permit applications to work with different IP versions on heterogeneous networks.

3. Comparing related works to the proposed porting platform

The research mentioned above discusses how to port the application from IPv4 to IPv6 and give some instructions that help in the porting process, our work is implement porting platform that also provides guidance to the developer in the process of porting application, as well as the proposed platform reduces the effort and time to complete the porting process. The proposed platform provides the possibility of compile and build the source code after its ported and create executable file for the ported source code, as well as the proposed platform provides the possibility to test the ported application.

More importantly, the development process using the proposed platform is done automatically, Also, the proposed platform provides the possibility to update the configuration of the development process and this feature allows the platform to be dynamic and the ability to port applications in languages other than C, by placing the appropriate instructions in the configuration file. Table (1) summarizes the comparison.

Reference number	Instruction for Porting source code	Save Effort	Save duration	Make porting platform	Testing	Automatically porting	static or dynamic
1	√	×	×	×	×	×	S
2	√	×	×	×	×	×	S
3	√	×	×	×	×	×	S
4	√	×	×	×	×	×	S
5	√	×	×	×	×	×	S
6	√	×	×	×	×	×	S
7	√	×	×	×	×	×	S
8	√	×	×	×	×	×	S
9	√	×	×	×	×	×	S
10	√	×	×	×	×	×	S
11	√	×	×	×	×	×	S
proposed platform	√	√	√	√	√	√	D

Table (1) comparing related works and proposed platform

4. Problem statement

In the near future, the spread of the IPv6 protocol will become on a large scale and this spread will require applications that use the IPv6 protocol, and this means designing and programming new applications that meet the needs of this protocol, building applications from scratch is not an easy issue which needs effort and time and this requires financial sums, most applications currently use the IPv4 protocol, but by taking advantage of the IPv4 applications and re-engineering these applications and making them compatible with the IPv6 protocol is an important step in terms of reducing the effort, time and money needed to create new applications that work on the IPv6 protocol. We propose to implement a platform to porting the IPv4 applications to IPv6 automatically.

Figure (1) show the proposed algorithm used to implement the porting platform.

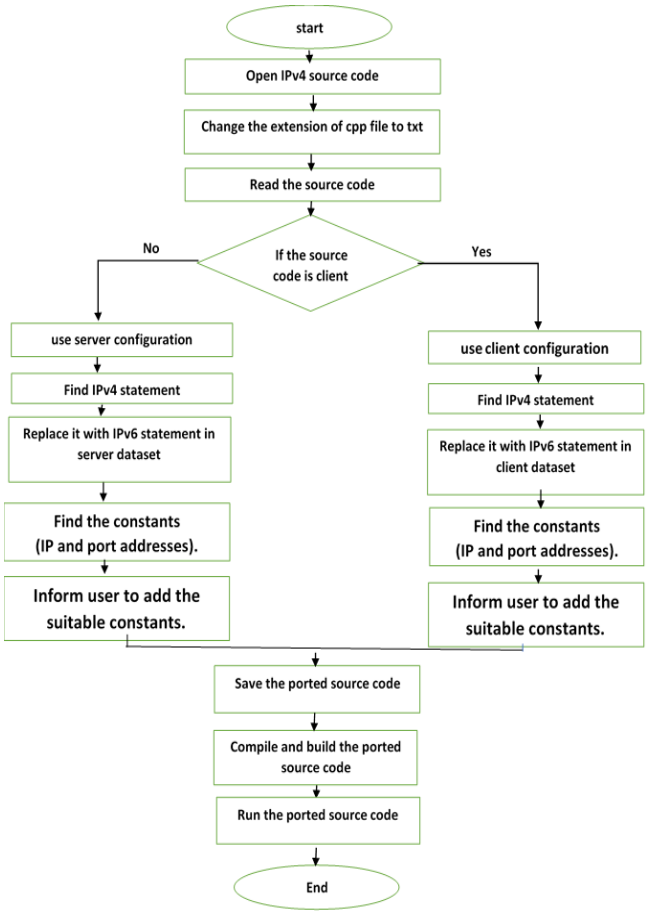

Figure (1) flow chart of the proposed algorithm

5. Software Re-engineering and its approaches

Re-engineering is the process of amending the product in some new form. Re-engineering means making fundamental changes to the code. Software re-engineering is concerned with re-implementing legacy systems to make them more maintainable. Re-engineering may involve re-documenting the system, organizing and restructuring the system, translating the system to a more modern programming language and modifying and updating the structure and values of the system's data. The functionality of the software is not changed and, normally, the system architecture also remains the same.

There are several specific approaches to reengineering, that approach involves a particular way of achieving the same goal: to build the target system (Reengineered system version).

5.1 Lump-sum re-engineering

This re-engineering approach takes a whole system in one wipe. Once the re-engineering initiative starts, all project goals are met and the target system is created. Figure (2) shows this approach as a black-box process with the existing system as input and the target system as an output. This approach results in monolithic projects and not always suitable.

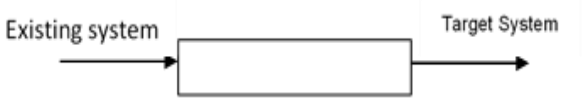

Figure (2) Lump-sum re-engineering

5.2 Incremental re-engineering

In this approach, the entire system is re-engineered gradually. One or more intermediate systems satisfy more project goals than preceding the intermediate system. The target system is produced when all project goals are satisfied. Figure (3) shows the incremental approach as a series of black boxes.

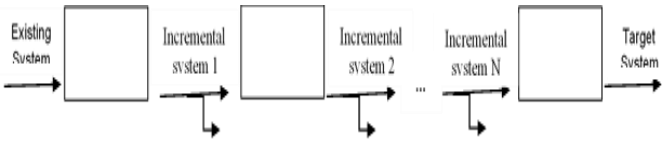

Figure (3) Incremental re-engineering

5.3 Partial re-engineering

In partial re-engineering, the system is only partially re-engineered. The re-engineered system portion shall be integrated with the current non re-engineered portion, this reduces potential system improvements. First, the existing system shall be split into two parts: a section which has to be re-engineering and a part which will not be re-engineering. The to be re-engineered portion can be part of the system that must be changed or a larger portion of the system that consists of the part that must be changed and another part whose inclusion simplifies the interface to the remainder of the system. Second, the re-engineering work must be done. Third, the portions of the system must be merged to produce the target system. The figure (4) shows three steps of partial re-

engineering approach. In this paper, a partial re-engineering was used to re-engineer the applications, because only part of the applications related to IPv4 should be changed to accommodate the applications with the IPv6 network, thus achieving the desired goal of reducing cost and effort.

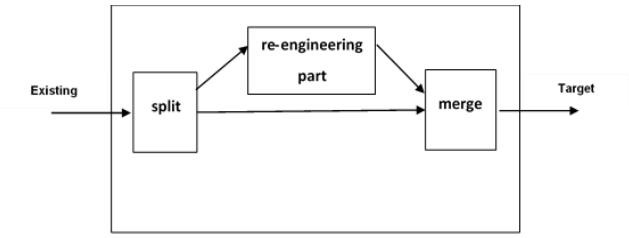

Figure (4) partial re-engineering

6. Porting process

The process of porting applications is considered very difficult unless the developer (the person who do the porting) aware of the special structure of IPv6 as well as knowledge of IPv6 programming, of course after doing a lot of efforts, through the use of partial re-engineering of applications we will get a quick method in the process of applying the porting as this method provides a process of separating a portion of the source code for the application of IPv4 that needs to be changed in order for the application to be suitable for work on the IPv6 network. The process of porting the source code needs to be familiar with all aspects of developing IPv6 applications like the details of header file and libraries, the porting process passes through two main phases the analysis and generation. The first main phase (analysis) used for finding all IPv4 dependent statements. The second phase used for generating the IPv6 version after replacing all IPv4 dependent statements into its corresponding IPv6 as show in figure (5).

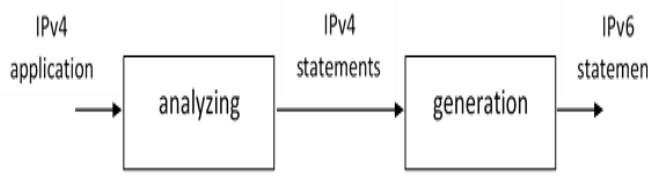

Figure (5) main phase of porting process

7. The design of the proposed platform

The porting process passes through several phases, the first phase is the reading phase where it works on reading the source code of the application, and the second phase is the analyzing phase where the IPv4 dependent statements are found, the third phase is the exchanging phase where it changes the IPv4 dependent statements to the equivalent IPv6 dependent statements, and the last phase is the testing phase where it works to ensure the integrity of the resulting source code after porting and generate IPv6 application.

figure (6) show the phases of the proposed porting platform and show the input and the output for each phase.

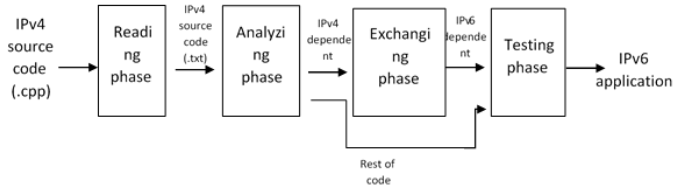

Figure (6) phases of the porting platform

7.1 Reading phase

The program works by reading the source code file written in C ++ and converts the file extension to text to be read, after reading the file text stored in the textbox and can edit it in the platform interface.

7.2 Analyzing phase

after reading the source code file, the program uses either the client configuration or server configuration as a dataset in the porting process depending on the source code that was read, the dataset contains all the statement that must be ported, the program uses it to detect the IPv4 statement that must be ported with the IPv6 statement, there is an executable file checkv4.exe that is included in the Microsoft Windows Software Development Kit (SDK) That is intended to furnish you with recognizes potential issues or features code that could profit by IPv6-competent capacities or structures, but the recognizes is limited on the basic items from table (2) like AF_INET, sockaddr_in, and the other items not recognized, the main process of this phase is to find the IPv4 dependent statements depending on the dataset and use it in the next phase.

IPV4 dependent statements	IPV6 dependent statements
127.0.0.1	::1
AF_INET	AF_INET6
SOCKADDR_IN	SOCKADDR_IN6
sockaddr_in	sockaddr_in6
sin_family	sin6_family
sin_addr.S_addr	sin6_addr
sin_addr.S_un.s_addr	sin6_addr
sin_port	sin6_port
sin_addr	sin6_addr
in4addr_loopback	in6addr_loopback
INADDR_ANY	in6addr_any

Table (2) IPv4 statement and corresponding IPv6 statement

7.3 Exchanging phase

After finding all the IPv4 dependent statements, the next step is to replace all the IPv4 dependent statements with the corresponding IPv6 dependent statement and inform the users to add the IP and port addresses that are used to communicate. The principal procedure of this phase is replacing the IPv4 dependent statement with the corresponding IPv6 dependent statement.

7.4 Testing phase

Save the ported source code and then compile it, and build the ported object code to generate (. exe) file and testing the new IPv6 application by running the ported source code. The compilation and building process are done by the Developer Command Prompt for VS 2019 In the windows environment by the command (Cl/EHsc mycode.cpp) to generate the mycode.exe.

8. Reducing cost evaluation

Depend on the basic COCOMO equations the effort and duration are increased if the line of code is increased.

The equation

$$E = a \times (KLOC)\text{^}b$$

$$D = c \times (E)\text{^}d$$

Where

E is the effort applied in person-months. D is the development time in chronological months. KLOC is the estimated number of lines of code for the project (express in thousands).

The coefficients a and c and the exponents b and d are given in Table (3).

Software Project	A	B	C	D
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Table (3) coefficients value

The process of calculating the effort and time of the application to be ported takes place in three stages. The first stage is calculating the effort and time during the reading phase, process of reading the source code of the application to be ported. After reading the source code we can easily count the line of code that contain in the source code file, and by applied the COCOMO equation we get the estimated effort and estimated time needed to develop the source code of the application.

By using the COCOMO equations the Effort is calculated by this equation:

$$\text{Effort} = 2.4 \times (kilo\ line\ of\ code)^{1.05}$$

$$\text{Duration} = 2.4 \times (Effort)^{0.38}$$

The second stage is calculating the effort and time during the analyzing phase, process of finding all the IPv4 dependent statements and highlighted it, By count the line of code of the IPv4 dependent statements and Appling the COCOMO equation we find the effort and time needed for analyzing and find IPv4 dependent statement

And the third stage is calculating the effort and time during the exchanging phase, process of exchange all the IPv4 dependent statements with corresponding IPv6

dependent statements process is also considered extra effort and extra time for general effort and time, and by applied COCOMO equation we find the effort and time needed for porting IPv4 dependent statement to corresponding IPv6 dependent statement

The proposed porting platform reduces the effort and time for developing the IPv4 application.

Table (4) Show the estimated effort and duration for a deferent number of lines of code for develop the source code. After applying the COCOMO equation, it is noted that the effort and duration increase progressively as the number of source code lines increases

Number of source code lines	Effort	Duration (month)
100	0.2139002	1.3912984
200	0.4428868	1.8345572
300	0.6779358	2.1567095
400	0.9170103	2.4190354
500	1.1591236	2.6442914
1000	2.4	3.4867458
1500	3.6737286	4.0990262
2000	4.9692716	4.5976007
2500	6.2812814	5.0257206
3000	7.6065646	5.4049498
3500	8.9429888	5.7478236
4000	10.289025	6.0623669
5000	13.005581	6.6268831
5500	14.374477	6.8837485

Table (4) estimated effort and duration for a deferent number of lines of code

The figure (7) and figure (8) show the curvature of the effort and duration required to develop a program, we note that effort and duration depend on the number of lines of code

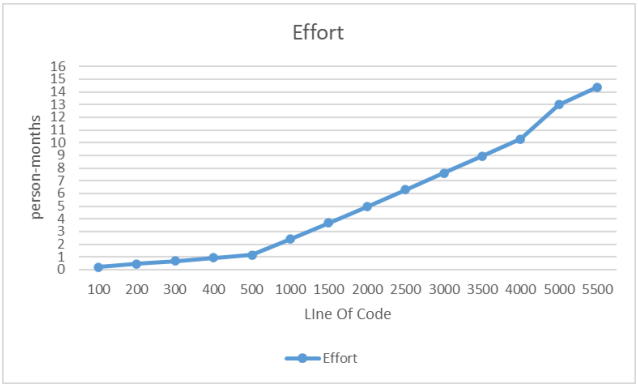

Figure (7)

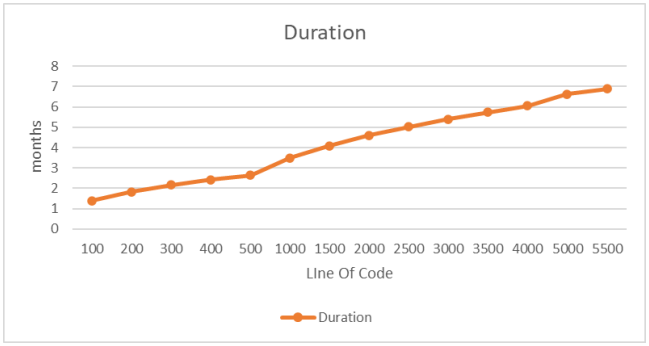

Figure (8)

9. Conclusion

Re-engineering turns out to be helpful for creating applications utilizing existing applications to deliver new application models that are all the more powerful. Re-engineering doesn't totally change the current applications, however, with the assistance of the current applications, it creates new and altered models, acquires a few capacities from the previously running applications, and adds a few capacities to deliver new applications. With re-engineering, more applications can be created from existing applications without changing the usefulness of utilizations that are as of now running. When we want to develop any network application, we have to look at the transport model in the source code. The transport model (socket procedures) is responsible for the network connection and its responsible for the data transmission, so in our case to porting IPv4 network applications to IPv6 we must porting the transport model to make the application run in IPv6 network. By using a partial re-engineering, we can separate parts of the code, which is

the transport model, that need to be ported from the rest of the code.

The effort and duration are reduced, using the proposed platform and the result is IPv6 application from IPv4 application with the lowest effort and time. The effort and cost are decreased using this platform if the source code was large in a line of code.

10. Bibliography

[1] R. Gilligan, S. Thomson, J. Bound, and W. Stevens, "RFC2553: Basic Socket Interface Extensions for IPv6." RFC Editor, 1999.

[2] senthil Sundaram, "Writing a simple IPv6 program." 2001, [Online]. Available: https://www.ibm.com/developerworks/library/wa-ipv6.html.

[3] S. S. Johar, "Porting IPv4 Applications to IPv6 By." 2002.

[4] J. Hanumanthappa and D. H. Manjaiah, "Transition of IPv4 Network Applications to IPv6 Applications [TIPv4 toTIPv6]," no. January 2009, 2014.

[5] T. Robles, R. Ortiz, and J. Salvachja, "Porting the session initiation protocol to IPv6," *IEEE Internet Comput.*, vol. 7, no. 3, pp. 43–50, 2003.

[6] "Implementing IPv6 Applications." 2010, [Online]. Available: http://www.6deploy.eu/tutorials/210-6deploy_devel_v0_4.pdf.

[7] K. Ettikan and T. W. Chong, "PORTABILITY ISSUES FOR IPv4 to IPv6 APPLICATIONS." .

[8] E. M. Castro, "Porting applications to IPv6 HowTo," *Lab. Over Next Gener. Networks*.

[9] C. Bouras, A. Gkamas, D. Primpas, and K. Stamos, "Porting and performance aspects from IPv4 to IPv6: The case of OpenH323," *Int. J. Commun. Syst.*, vol. 18, no. 9, pp. 847–866, 2005.

[10] T. De Miguel and E. M. Castro, "Programming guidelines on transition to IPv6," *Transition*, no. January. 2003, doi: 10.1.1.6.7440.

[11] E. M. Castro-Barbero, T. P. de Miguel-Moro, and S. Pavón-Gómez, "Transition of applications to IPv6," *IPv6 More than A Protoc.*, vol. 6, no. 2, p. 15, 2005.

[12] Harmandeep Singh, Software Reengineering: New Approach to Software Development, International Journal Of Research In Education Methodology, Council For Innovative Research Vol. 1, No.3, October, 2012.

[13] Stevens WR. Network Programming, vol. 1 (2nd edn). Prentice-Hall: Englewood Cliffs, NJ, 1998.

[14] Sun Microsystems, Porting Networking Applications to the IPv6 APIs.

[15] Microsoft Corporation, Adding IPv6 capability to Windows Socket Applications.

[16] A Brief Summary of the Original COCOMO Model, https://csns.calstatela.edu/site/s17/cs3337-1/item/5743412.

[17] MSDN Library, Windows Sockets Version 2. https://docs.microsoft.com/en-us/windows/win32/winsock/sockaddr-2.

[18] S. G. ZHANG, S. LIANG, and G. X. FU, “Transition of socket application from IPv4 to IPv6,” J. on Communications, Beijing, vol.27(11), pp.24-27, November 2006.

[19] Blanchet M, Cormier A, Parent F. Porting applications to IPv6: simple and easy!, May 2000. http://www.viagenie.qc.ca/en/ipv6/presentations/

[20] Gilligan R, Thomson S, Bound J, McCann J, Stevens W. Basic socket interface extensions for IPv6. Internet Engineering Task Force RFC 3493, February 2003.

[21] Zeadally S, Raicu I. Evaluating IPv6 on Windows and Solaris. IEEE Internet Computing 2003; May–June:51–57.

[22] Microsoft IPv6 Technology Preview for Windows2000. http://msdn.microsoft.com/downloads/sdks/platform/tpipv6.asp.

[23] Microsoft Corporation.IPv6 Guide for Windows Sockets Applications [EB/OL]. http://msdn2.microsoft.com/en-us/Library/ms738649.aspx.

[24] W. Stevens. RFC3542 - Advanced Sockets Application Program Interface (API) for IPv6, 2003.