2009 6th International Multi-Conference on Systems, Signals and Devices

WINDOWS-BASED ACTIVE-ROUTER DESIGN AND EVALUATION

Sufyan T. Faraj[1], Omar A. Athab[1], Kasim M. Al-Aubidy[2]

[1] Faculty of Computers, Anbar University, Iraq
e-mail: sufyantaih@yahoo.com
[2] Faculty of Engineering, Philadelphia University, Jordan
e-mail: alaubidy@gmail.com

ABSTRACT

The paper presents the design and evaluation of an Active Router (AR) architecture, which provides flexibility for the development of future network services. The hardware is based on a personal computer with 2GHz, Intel P4 processor. The designed AR depends on the use of Windows OS, enhancing the Active Network Encapsulation Protocol (ANEP) and the efficient use of C++ programming. Windows OS and C++ language are rarely used in such projects due to complexity of kernel-mode and network-oriented programming requirements. Enhancing ANEP gains novel service composition scheme. Finally, the success of the AR architecture and prototype implementation is evaluated by means of a practical application.

Index Terms— Active Network, Programmable Network, Active Router, Programmable Router, IM Driver.

1. INTRODUCTION

Traditional packet-switched networks, or exactly intermediate nodes, perform only the processing necessary to forward packets towards their destination. Over time, more and more functionality is being deployed inside the network, in an effort to provide better services to users[1]. For example, firewalls at the border routers (gateways) for security purposes[2] and network caching as a mechanism to reduce network load[3]. The need for qualitatively better communication mechanisms for real-time traffic has led to the investigation of Quality-of-Service (QoS) mechanisms for the Internet. Most of these network-side services are implemented as individual ad-hoc extensions. However, the network provides no architectural support for flexible extensibility [1].

Wetherall and Tennenhouse [4] have first pursued the idea of placing program fragments into IP packets as part of the ActiveIP project. Initially, they studied the potential of placing small programs within the option fields of IP packets. These so-called active options, encoded in Tcl language in their prototype implementation, were executed by modified network nodes as the packets traversed the network. SwitchWare Active Network Architecture [5] consists of three layers: active packets, active extensions and a secure active router infrastructure. Active packets carry programs consisting of code and data to replace both the header and payload of traditional packets.

Active Node Transfer System (ANTS) provides a capsule programming model[6]. Capsules are packets that encapsulate data with a customized forwarding code. Applications use the network by sending and receiving capsules via active nodes. When a capsule arrives at an active node, the corresponding routine is executed to forward the capsule. The demand-pull mechanism is used to obtain code from the previous node that the capsule visited. The ANTS prototype is implemented in Java under UNIX operating system.

The Smart Packets project [7] emphasizes at addressing problems that are inherent in typical polled managed devices rather than aiming for general transport mechanisms such as ANTS. Smart packets are encapsulated within ANEP packets and ANEP packets are encapsulated within an IP packet using a specific option (router alert). The Smart Packets architecture expects all programs to fit within one Ethernet frame. There is no existing language that had a compact enough representation for Smart Packets environment. As a result, Sprocket and Spanner languages are developed as part of the Smart Packets project.

This paper addresses an attempt to investigate a useful step towards active network mechanisms that considers flexible extensibility through programmability as part of the fundamental architectural design.

2. ACTIVE NETWORK FUNDEMANTELES

A central feature that distinguishes ANs from configurable ones is the programming model. A configurable network aims to establish a maximal set of high-level features that can be configured with a single action, while AN focuses on identifying a minimal set of primitives (for example, system calls of host operating systems) from which one can compose (or program) a broad spectrum of features[1]. By providing a programmable interface in network nodes, ANs expose the resources and mechanisms for constructing or refining new network services from those elements. In short, ANs support dynamic modification of the network behavior as seen by the user. The scope of network programmability varies from control plane to data plane programmability and extends from very limited to highly flexible forms depending on the programming interface[8].

978-1-4244-4346-8/09/$25.00 ©2009 IEEE

Several programming models have been suggested for ANs. A common approach is to provide a programmable engine at each intermediate node that can be programmed on a per-packet basis. Every packet contains in addition to the user payload (data) some form of active program that is executed on each intermediate node as it traverses the network. This is called active packet or in-band approach. Another approach in which active programs are loaded onto the active nodes in out-of-band fashion, prior to the transmission of data packets. This is called active extension or out-of-band approach [9].

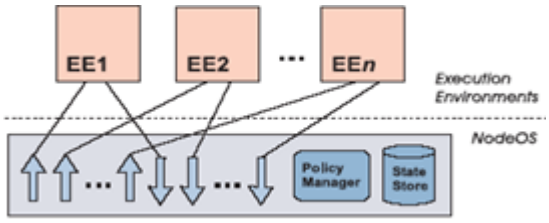

Figure 1. Active node architecture of DARPA.

2.1 Active Node Architecture

Active network working group at DARPA defines the fundamental parts of an active node and how they interoperate[10]. The functionality is divided into the Active Node Operating System (NodeOS) and the Execution Environment (EE). While the NodeOS manages access to node local resources and system configurations, the EE implements the active network APIs supported by the node. Figure 1 shows the envisioned general architecture for an Active Network node. Although this architectural framework considers only the integrated approach, it is considered by many researchers to be a de-facto standard[9].

2.2 Active Network Encapsulation Protocol (ANEP)

The ANEP document specifies a mechanism for encapsulating AN frames for transmission over different media[11]. The suggested format allows use of an existing network infrastructure (such as IPv4 or IPv6) or transmission over the link layer. This mechanism allows co-existence of different execution environments and proper demultiplexing of received packets. The program is executed by a receiving node in the environment specified by the ANEP. The format of the ANEP header is:

The version described by this document is 1. Only the most significant bit in the flags field is used. If its value is 0, the node could try to forward the packet using the default routing mechanism (if one is in use), if the necessary information is available in the Options part of the header. If the value is 1, the node should discard the packet. The ANEP Header Length field specifies the length of the ANEP header in 32 bit words. The Type ID field indicates the evaluation environment of the packet. The active node should evaluate the packet in the proper environment. If the value contained in this field is not recognized, the node should check the value of the most significant bit of the Flags field in deciding how to handle the packet. The ANEP also may contain option field(s). Some options are identified by the ANEP document [11] and the others are left to the implementer responsibility.

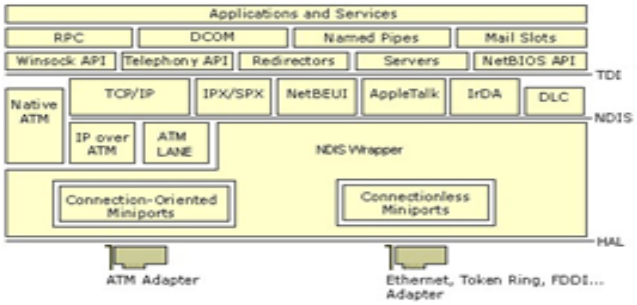

Figure 2. Windows OS network architecture.

2.3 Networking Stack in Windows OS

Windows OS network architecture may be imagined as shown in Figure 2. Components that contribute the same horizontal level, in the figure, provide similar functionality [12]. The Windows 2003 network layers are described below from the bottom of the network architecture model up to the top. Network Driver Interface Specification (NDIS) provides a communication path between network adapters and network protocols and manages the binding between these components. NDIS layer consists of the following [13]:

- NDIS wrapper represented by the NDIS library (Ndis.sys), which exports functions for use by transport protocols and adapter drivers.
- NDIS miniport drivers, which are responsible for interfacing transport protocols to particular network adapters.

According to the designer's requirements, NDIS layer may contain one or more NDIS Intermediate (IM) drivers that are located between transport drivers and miniport NIC drivers to perform additional functionality.

3. ACTIVE ROUTER ARCHITECTURE

The proposed networking stack model has to be "component-based" design and not layered. The advantages of component-based design, namely code modularity, reusability, and dynamic composition, facilitate the development and deployment of custom network services.

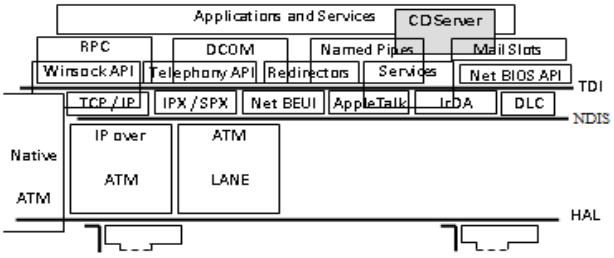

Figure 3. CD and PM units.

Actually, the protocol stacks are replaced by protocol components that can be tailored and composed to perform application specific functions.

Concerning the Programming Model, active packet (in-band) approach in AN tends to be fairly restrictive due to the limited programming capabilities. In the other side, the active extension (out-of-band) approach often lacks adequate service composition capabilities for software components. This project resolves these limitations by extending the active extension programming model by a flexible composition framework for software components.
NDIS IM driver has been chosen as a base in designing the packet interceptor in this project. It is located between the LLC and MAC sub layers, and this feature gives IM driver a lot of control over network packets, without affecting other network protocol stack components. Moreover, an IM driver could be layered above or below another IM driver without affecting its function.

3.1 Architecture Overview

The architecture of the proposed AN has been divided into two functional parts:
- the Component Distributor (CD) and
- the Packet Manipulator (PM) part.

The CD concerns the transferring and managing of User Components (UCs) from a Privileged End–System (PES) or network administrator (ADMN) to the AR. In other side, the PM functional part extends the OS networking stack such that it can intercept the in-bound packets that enter the AR and discriminate among the various types of packets. After distinguishing the type, the PM forwards the packet to the proper component to be serviced. The proposed architecture is designed to extend exiting routers by layering active network-specific functionality on top of the router operating system. The following sections explain the two parts of the proposed AR, as shown in Figure 3.

3.2 Component Distributor (CD)

The proposed CD allows the user to load new components in the AR. The UC is a program performing either protocol processing or value-added function to the packet. It is expected that the code of the UC is sent from any Privileged-End System (PES) to the Active Router (AR) using the CD unit. The end user who send and install a UC should be authenticated and authorized to add a UC into the router. Also, the code of the UC must be authenticated to ensure safe evaluation within the AR execution environment. Each UC has its own Component Identifier (CID) which is associated with the component during transmission.
PEU must firstly send the UC (which implements the required protocol or service) to the AR to be installed there. Then, PEU send his packets (which require active processing) such that it refers to the required component using the CID. In this manner, the AR will process the received ADPs by the indicated UC. To gain a certain active service, the AN user is responsible to determine which UCs and in which order they must be composed.

The proposed scheme of CD is the transfer of the UC to router(s) along the path that active packet using the service follows. The code is cached at these routers for later use. The CD unit provides the capability of packing (if more than one UC), and then uploading the UC from a PES to the AR using FTP protocol. Furthermore, the CD unit is also responsible for controlling (replace or uninstall) the installed UCs remotely. The CD unit operates in a client/server fashion. The AR represents the server, whereas the PES represents the client. Hence, jobs of the CD can be summarized as: packing, uploading and controlling the UC.

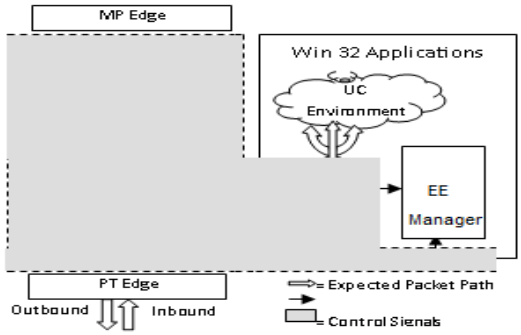

Figure 4. Packet manipulator architecture.

3.3 The Packet Manipulator

Certainly, manipulating a network packet demands capturing the packet itself. The IM driver has been proposed to be the foundation of the Packet Manipulator (PM) architecture. In addition to catching a packet, the PM performs a light firewalling, lifting the packet from the kernel to the user mode, recognizing its type, and finally dispatching the packet to the user component that it whishes for processing. Consequently, the PM architecture was further divided into the following functional units, as shown in Fig.4:
Packet Interceptor/Injector (PIJ): It provides the interface between the active network environment and the data path on the router. It is responsible for intercepting the network traffic traversing the node and passing it to the active network environment for processing. Also, it can re–inject the network data back into the default forwarding path on the node or sends it directly through one of the outgoing interfaces.
Packet Filter (PF): This project provides a programmable PF on the read handle. The PF actions are:

- **Block:** drop the matching packet from the normal packet flow.
- **Pass:** allow the matching packet to pass up to the PT driver as in the normal flow.
- **Read:** pass a copy of the received packet to the packet bridge which is the next unit in the AR.

Packet Bridge (PB): It is required to transfer network data to and from the UC. It is noteworthy to state that the PIJ and PF units are placed within the kernel space of the proposed AR. It is suggested to load the UCs in the user space of the OS. Accordingly, PB targets the transportation of Packets to the user mode to be, then, processed by UCs. Also, if required, the PB transport packets back to the PIJ to be re–injected into the default forwarding path on the node or sent it directly through one of the outgoing interfaces.

Packet Classifier (PC): The object of the PC is the discrimination among the various types of packets that may pass through the Packet Manipulator (PM). To be serviced properly, packets should be firstly classified. To clarify the ambiguity that may occur, these different types of packets can be categorized into the following: Component and Data packets. The designed PC distinguishes between these types, and then forwards each one to the correct path.

Packet Dispatcher (PD): It defines the "route" through the UC space for the ADPs passing the AR. The PD plays a central role in the service composition process. It determines, based on the ANEP header, which UC(s) are involved and in which order they should process the ADP. After the UC(s) finished it's processing on the ADPs, the PD returns the packet back to the windows network stack through the PB.

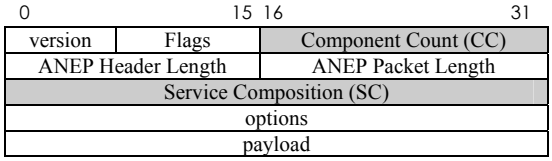

Figure 5. Format of the proposed ANEP**.**

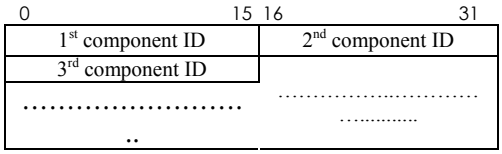

Figure 6. Format of "service composition" field**.**

3.4 Service Composition

The concept of component-based services is envisaged to achieve a good flexibility in introducing functionalities for the ADPs. This means that we have either multi Execution Environment (EE), where each one represents a single component, or single EE. In the two assumptions there is a limitation in determining which components(s) and in what order are appropriate to process the ADPs. According to the basic ANEP header, the type ID field in the entering ADP can assign only one component to process the packet. This is a big restriction. This will restrain the AR to be really active and flexible. Furthermore it may weak or omit the principal of component-based services in the AR; therefore, a proposal is presented below to enhance the original ANEP to overcome this limitation.

3.5 Enhancing ANEP

In the proposed ANEP enhancement, the type ID field is used to indicate the count of UCs that may be composed to introduce the service to the active packet. The Component ID (CID) of the component itself is described in a new proposed variable-length field called Service Composition (SC) field. It is placed after the basic header and before the options field as shown in Fig. 5. This new field consists of the CIDs of the components that must be composed to create the required active service. The order at which the CIDs appear in the service composition field is considered as the sequence of the components that will be executed in the AR. Using this format, the lack of service composition capability in the original ANEP can be avoided. For example, when an active packet want to be processed by three components; that are component 4, 6 and 3, respectively, in such case; the Component Count "CC" field would contain the value 3, and the SC field will contain the CIDs of components 4, 6, and 3 respectively. The proposed format of the SC field is shown in Fig. 6. As an example, a common service like IPv4 routing and forwarding may take 5 or 6 components (according to the designer of the components) to perform the packet integrity, TTL decrement, CRC calculation, lookup of routing table, and finally forwarding to the suitable next hop (or destination). Table 1 shows a comparison between the original ANEP format and the proposal for enhancing ANEP.

Table 1. Comparison between ANEP and enhanced ANEP.

Function	Original ANEP	ANEP Proposal
Service components	Single component services.	Single and multi- component services
functionality	fair Functionality	better Functionality
Processing time	Require the original ANEP processing time	Require more processing time than the original
Type ID range	Wide type ID range.	wide type ID range
Flexibility of service	Services are restricted because it depends on single component	Services are flexible because it is fully controlled by the end user
Bit overhead	Original bit overhead	Overhead is larger than the original

3.6 Modes of Operation

According to the specified architecture, four modes of operation of the proposed AN can be recognized;

Upload Mode: The upload mode of AN operation targets the transferring of one or more UC(s) from one of the PESs to the AR. As shown in Fig. 7-a, this mode does not involve any transmission beyond the AR.

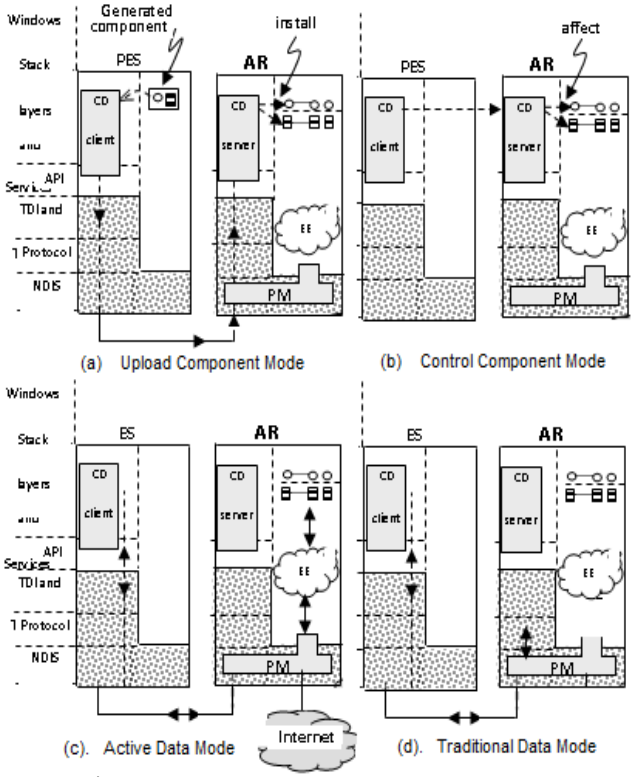

Figure 7. Modes of operations.

Control Mode: The purpose of this mode is the control (uninstall or replace) of the previously installed UCs, as shown in Fig. 7-b. This mode does not consist of any actual transmission of code or data; it only involves packets of commands issued by the CD client in the ES and implemented by the CD server in the AR.

Active Data Mode: This mode represents the envisaged operation of the designed AN (see Fig.7-c). It constitutes the transmission of Active Data Packets (ADPs) between the internet and any ES in the target LAN (in both directions), passing through the AR. ADP contains an ANEP header.

Traditional Data Mode: It is adopted to keep the backward compatibility with the existing computer networks. Packets transferred in this mode are exactly same as that are used in the current traditional computer networks, as shown in Fig. 7-d.

4. SYSTEM IMPLEMENTATION

Since conventional stand alone routers are typically closed commercial systems, it is virtually impossible to get source-level access to their software. Consequently, the AR prototype implementations outlined in this section is being built upon a Personal Computer with Microsoft's Windows 2003 server (which supports basic routing functionality).

The software linker between the CD server and the PM in the AR is the Execution Environment Manager (EEM).

4.1 Component Distributor Implementation

All Widows-based computers have internet explorer-based FTP clients; the user can always launch the FTP client from the address bar of the explorer. But FTP server capabilities are built into Internet Information Server (IIS) application that included with Windows 2003 servers and above. PEU can display the two sides' screens (FTP client and server) in his workstation. UCs can be uploaded easily by drag and drop from the PES screen to the AR screen. PEU, also, can control (delete partially or completely and rename) the installed UCs in the AR.

The proposed UC is implemented as two files: code and configuration files. The code file of the UC is a C++ language file converted to DLL library. It contains the required processing to be applied on the ADPs. The configuration file is an initialization file that contains configuration data such as the path of code file of UC (the .dll file name) and an optional control buffer.

4.2 Packet Manipulator Implementation

IM driver: The foremost step in realizing the PM architecture is the implementation of a simple "pass through" IM driver. The IM passthru simply re–warps the sent/received packet and pass it down/up to lower/higher–level drivers. The passthru driver has been implemented as six C language modules: Main, Adapter, NDISreq, Recv, Send and Status.

PIJ Unit: There are no comments from Microsoft about how to intercept (or inject) a packet from (into) an IM driver. Complicated way must be followed to get a copy of a received packet that may enter the AR processing. Two approaches are available; the developer either enforces a step–by–step (manual) C-programming or uses a cloned packet approach. However, the second approach was taken. PCAUSA Corporation[14] offered ready–to–use cloned–packet software which targets extending the Microsoft IM driver to achieve a buffered complete copy of received packets. The cloned–packet approach has been realized by adding a smart module "UTIL" to support Recv module.

PF unit: The base code of PIJ unit was reorganized by adding a new module to isolate the actual filtering code from the basic packet interception. The key functions provided in this module are:

- SetPktFilter: It is responsible to move the information of filter setting from the user application to the context pool of the currently opened adapter..
- ResetPktFilter*:* It resets what the SetPktFilter has been set.

PB Unit: The PB unit targets to facilitate the interface between the kernel and user sides of the PM. Hence, PB consists of modules in both the kernel and the Win32 application sides. The DEVMJFCN.C and WDMSUP.C are modules in the kernel side that encapsulates the routines related to the IOCTL interface. In user mode side, the modules are realized in C language and then converted to dll file such that they export a ready to use APIs for the Win32 applications of the AR. These APIs involve the following:

- OpenVirtual/LowerAdapter API.

- ReadOnAdapter API.
- WriteOnAdapter API.
- SetPKTFilter API.
- ResetPKTFilter API.

In this research, the IRP–based interface is used to implement the user/driver programming interface. In a consequence, applications can use the basic Win32 functions; CreatFile, DeviceIoControl, ReadFile, WriteFile; and CloseHandle in the user–mode side of the interface.

PC Unit: At this point, a copy of a complete, received, filtered packet has been gained and queued in the user-space of the AR. The PC is realized in a discrete function that includes the following steps:

- Examine the type of Network layer protocol.
- According to the protocol type, CLSF function can determine which field in the protocol header should be tested to know the packet is an ADP or not.
- According to the suitable header's field, if the packet is ADP, return non zero value.

PD Unit: After completing the filtering and classification operations, the PC will deliver only the Active Data Packets (ADP) to the Packet Dispatcher (PD). Jobs of PD unit are implemented as three C++ functions:

- ServNo function: it determines how many UCs must be called and executed on the received ADP.
- ServID function: it targets to withdraw the CIDs of the required UCs in a correct order.
- GetServInfo function: it gets a copy of contents of the configuration file associated with the UC.

5. SYSTEM EVALUATION

The evaluation of the AR architecture is to a large extent a theoretical analysis. Based on a concrete case study, this section evaluates the designed AR qualitatively and quantitatively.

5.1 Packet Generator/Injector

The computation of UC is typically driven by the data passing through the AR, therefore, additional software that generates data packets matching the tested component is required. In this work, an external program is proposed and implemented which creates and then injects data packets into the network stack of the testing machine. We will call this program a Packet Generator/Injector (PGI). In each testing course the testing machine which runs the PGI software is connected to the AR (to be tested).

5.2 Evaluation Methods

Since the main contribution of this paper is the architecture for ARs, a qualitative evaluation of the concepts and design of the architecture has been regarded as more meaningful. Since it has been feasible to implement only a subset of the overall AR architecture, a quantitative evaluation of the entire system (with applications) cannot be provided at this stage.

5.3 Qualitative Evaluation: Case Study

Assume that an establishment (Company, Bank, Library, Campus … etc) divided into headquarter and other branches distributed around wide area, such that they exchange information through the internet. It is envisioned that Active Routers (ARs) will form the core elements (access and gateway) of this network. ADMN of the network was decided to operate under TCP/IP suite and running the ARs under Windows 2003 server.

The challenge of the setting is to "open up" the establishment network and to provide public services (through internet) to remote authorized users (RAUs). RAUs may belong to the establishment staff or external customers. It is required to provide sufficient confidentiality for the transferred information from RAUs to the establishment (headquarter or one of the branches) services. It is decided to provide security by the developed encryption algorithm called "RC5" [15]. Since existing proposals for such example problem rely on enhanced support in network routers (which is not yet provided by most router manufacturers), it can be argued that the problem can be addressed more elegantly through the use of AN technology.

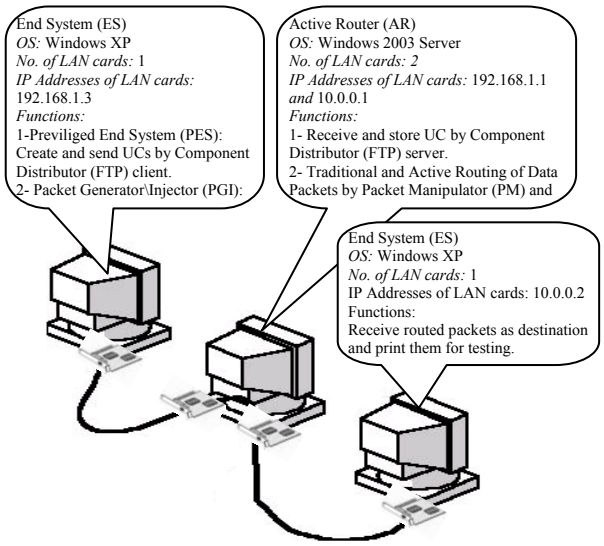

Figure 8. Setup of case study prototype realization.

For prototype implementation, the basic setting of case study is minimized into only three computers, namely A, B and C, as shown in Fig. 8. Computer B realizes the gateway active router whereas computer A realizes two entities. Before any information exchange, computer A will represent the PES; actually it is the software department of the establishment which will create and install the UC (the RC5 decryption component). After installing the UC, the function of this computer will be replaced to represent one of the RAUs who will send encrypted ADPs to the establishment (computer C), through the gateway AR (computer B). NIC

cards used in this prototypical setting are of type SURECOM 10/100M PCI adapter and uses Fast Ethernet Protocol.

In this proof-of-concept case study, RAU realizes the RC5 encryption algorithm as a single module (RC5.C) using VC++6 environment. After performing RC5 encryption of ADP payload, the RAU exploits the previously explained PGI unit to create and then send the packet. C++ language is used in programming the code and configuration files. Concerning the control buffer of the configuration file, it is exploited to store the users' supplied secret key.

In the AR computer (gateway), the pre-installed CD server receive the UC sent by software department of the establishment. CD server installs the two files (code and configuration) in a pre-defined folder in the AR. However, EEM continue running the Packet Manipulator for searching whether there is a new received packet or not. When the RAU begins send the RC5 encrypted ADPs, EEM will perform the required processing (decryption). Thousands of ADPs with varios payload contents and length are generated, encrypted and then sent from the RAU (computer A) to the establishment (computer C) passing through the AR gateway (computer B). All the original packets are arrived to computer C which means that successful decryption has been done in the AR.

5.4 Quantitative Evaluation

To evaluate the designed AR architecture and hence the performance of the envisaged AN, three types of tests were done; Control Test, AN Test, and Backward Compatibility Test. In all of them, a throughput and %CPU usage has been measured. The measurements are done using the Windows performance tool. In the three tests, a data of 360 MB size has been transferred from computer A to C, through B, and all measurements are accomplished at computer B. all results are tabulated in Table 2.

Table 2: Results of quantitative evaluation tests.

Test Sequence	Test type		Throughput (Packets/sec)	%CPU Usage
1	Control test		5679.8	30.2
2	AN test	PM test	2902.6	78.8
		5 UCs test	914.9	84.1
		10 UCs test	453.1	86.7
		15 UCs test	335.8	89.8
3	Backward compatibility test		4041.3	75.2

Control Test: It just tested the speed of the network, operating system overhead, and network stack overhead without installing the developed AR software. In control test, computer B is configured appropriately to operate as a traditional router. Packets of TDP type are used in this measurement. The next two tests, which involve the active enhancement software, will be compared with control test to measure how much degradation and processing cost is paid for active programmability.

Active Network Test: In this test, all the implemented software, which has been explained in section 5, is installed in computer B which still running Windows 2003 server. The object of this test is to gauge throughput and CPU usage required to achieve the active programmability of the router. AN test involves two parts, PM test and UC test.

PM Test: The measurements try to quantify the processing speed and load needed in PM units (namely, PIJ, PF, PB, PC and PD). The Component Loader (CL) is suppressed temporarily from the EEM to accomplish this target.

UC Test: It aims to check how many UCs can be used in each service such that the AR is not saturated. The evaluation in this paper is directed to evaluate the architecture only and not tied to a specific service or application. Therefore, the test checks the system performance based on the same configuration as used in the previous test except with the addition of a number of "Null" components to measure the extensibility of the system. The significant factor to be gauged is the loading of component from its local cache to the memory, when UC is called.

Backward Compatibility Test: In backward compatibility test, the developed AR software is also installed in computer B. A stream of TDPs (IP packets) is sent from computer A toward computer C through the AR. The purpose of this test is to ensure the backward compatibility to the current traditional routers and check the quantity of degradation in throughput and the cost in processing time when AR is used to route TDPs rather than traditional router.

6. DISCUSSION

PM test takes approximately double processing time and half throughput with respect to the control test. It is the price that must be paid to transport to user-level processing of the PM stage (i.e. PB, PC and PD). Because of the PIJ and PF units are operate within the kernel of the router, their effects are negligible.

Concerning the UC test, the values of measurements refer to high throughput degradation accompanied by consuming of most processing power, for 5, 10 and 15 UCs. The appended degradation in throughput and increase in CPU cost is mainly attributed to the successive loading of the component's code from the permanent cache to the EE (RAM). It is also possible to conclude that the system in its current state (experimental) is not extensible when more than 15 UCs are added to the path of ADPs. This can be solved, for example, by pulling the PC and PD units down to the kernel space which will lead to two times increase in throughput. Using multi-processor router (as in current traditional routers) or upgrading router's hardware by FPGA technology may also contribute in solving this shortcoming. At close to 453.1 packets/sec (3.6 Mbps), the developed system is usable, for example, in modems, wireless links, and access links through T1 (1.5 Mbps) applications. The results seem to be reasonable as a solution for AR in small-sized edge-networks.

It is important to note that there is tradeoff here between the performance and modularity. To get acceptable performance, count of UCs must be minimized. Decreasing the number of UCs leads to less flexibility. From other side, installing a new protocol or function with high modularity (divided into small UCs) means sever degradation in throughput. One of the intermediate choices is to insert the mainstream protocol (such as IPv4) as a standard network stack with the capability of support new protocols and functions as UCs.

The backward compatibility test is succeeded in proving that the AR routes and forwards the TDPs (IP packets) correctly without any error or side effects and it is not severely affects the standard throughput obtained in control test. The relative little degradation in throughput is due to the overhead of PM units (except PD unit) with paying double processing time.

7. CONCLUSIONS

Many concluded points of high importance can be declared as follows:

- Component-based active router architecture enables network programmability through extensibility of router functionality and services.
- The enhanced ANEP-based service composition enables transparent network programmability. New network functionality can be flexibly integrated into the packet processing chain on the router simply by inserting a CIDs into the ANEP header.
- Implementation of a NodeOS that provides resource control and safety features requires the NodeOS to be tightly coupled with the underlying (host) operating system.
- A split implementation across both kernel and user space of the underlying system appears to be a good choice. This approach takes advantage of the high flexibly programming environment in user-mode and sophisticated protection and safety mechanisms of today's OSs.
- Standard user-space implementations for active networks typically suffer largely from the performance hit resulting from the copy operations required to pass the network traffic "up" into user-space and back "down" again. As far as possible, packet processing must be in kernel space.
- It is recommended to use programmable devices to implement the hardware and software of the proposed system.

8. REFERENCES

[1]. P. L. Simeonov, The wandering logic intelligence, a hyperactive approach to network evolution and its application to adaptive mobile meltimeidia communication, PhD dissertation, faculty of informatics and automation, technology university of Ilmenau, Germany, 2002.

[2]. P. Xue, S. Chandra, Revisiting multimedia streaming in mobile ad hoc networks, NOSSDAV '06 conference, Newport, Rhode Island, USA, 2006..

[3]. G. Barish, k. Obraczka, World Wide Web catching: trends and techniques, IEEE communications magazine, May 2000.

[4]. D. J. Wetherall and D. L. Tennenhouse, The ACTIVE IP option, In 7th ACM SIGOPS European Workshop, Ireland, September 1996.

[5]. D. S. Alexander, W. A. Arbaugh, M. Hicks, P. Kakkar, and J. M. Smith, The SwitchWare active network architecture, IEEE Network, vol. 12, pp. 29-36, May/June 1998.

[6]. D. J. Wetherall, Service Introduction in an Active Network, PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, USA, 1999.

[7]. B. Schwartz, W. Zhou, A. W. Jackson, W. T. Strayer and D. Rockwell, Smart Packets for Active Networks, In 2nd Conf. on Open Architectures and Network Programming, OPENARCH'99, NY, Mar. 1999.

[8]. H. J. Wang, *et. al.*, Iceberg: An Internet-core Network Architecture for Integrated Communications, Project at Computer Science Division, U. C. Berkeley, 2000. Available at: http://iceberg.cs.berkeley.edu.

[9]. S. Schmid, LARA++ Design Specification, Work in progress report on the next generation active router architecture of Lancaster University, Computing Department, Lancaster University, UK, 2000.

[10]. K.L. Calvert (Ed.), Active Networks Working Group, Architectural Framework for Active Networks, Draft, August 1998.

[11]. D.S. Alexander *et al.*, Active Network Encapsulation Protocol (ANEP), Internet draft, IETF, July 1997.

[12]. D. A. Solomon and M. E. Russinovich, Inside Microsoft Windows 2000, 3rd Edition, Microsoft Press, 2000.

[13]. Microsoft Corporation, Microsoft Windows 2000 Driver Development Kit, Network Drivers, 2000.

[14]. T.F. Divine, NDIS IM driver samples for windows NT and higher, Available online at: www.pcausa.com, 2006.

[15]. R. L. Rivest, The RC5 encryption algorithm", MIT Lab. for computer science, USA, 1995.