J. of university of anbar for pure science : Vol.4:NO.3 : 2010

ISSN: **1991-8941**

A WINDOWS-BASED ACTIVE-ROUTER ARCHITECTURE

Sufyan T. Faraj * Omar A. Athab Kasim M. Al-Aubidy*****

***College of Computer, University of Anbar, Iraq**
****Al-Khawarizmi College of Engineering. University of Baghdad, Iraq**
*****Faculty of Engineering, Philadelphia University, Jordan**

Received: 10/1/2010 Accepted: 19/5/2010

Abstract : As computing power becomes cheaper, more and more functionality is being deployed inside computer networks, to provide better services to users. Examples of such services include support for Quality-of-Service (QoS), multicast, mobility and security. Nevertheless, such functionalities are still lacking in most networking systems. Active networking is a step towards enhancing the static and inflexible structures of current networks. This paper focuses on the design and development of technologies that allow rapid deployment of functionality throughout the network. The paper presents the design and implementation of a Windows-Based Active Router (WBAR) 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 and efficient C programming. Windows OS is rarely used in such projects due to complexity of kernel-mode programming.

Key words— Active Router, Active Network, Architecture, Windows

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]. In addition, the need for qualitatively better communication mechanisms for real-time traffic (for example, interactive audio and video) than simple best-effort forwarding has led to the investigation of QoS mechanisms for the Internet [1]. Most of these network-side services are implemented as individual ad-hoc extensions. However, the fundamental problem remains, namely that the network provides no architectural support for flexible extensibility [1]. Therefore, in this paper, an attempt to investigate a useful step towards active network mechanisms that considers flexible extensibility through programmability as part of the fundamental architectural design. Section 2 introduces some of the completed related works in the Active Network (AN) field. Section 3 includes different architectural approaches towards network programmability, various programming models, and Operating System (OS) support technologies. Section 4 presents the design of the windows-based active router (WBAR) architecture. Special focus is placed on the service composition framework. Section 5 describes the ongoing implementation efforts of developing prototype of the WBAR architecture. Finally, Section 6 concludes the paper by drawing together the main arguments of this work.

LITERATURE SURVEY

D. J. Wetherall and D. L. 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. As a consequence, a new programming language for Active Networks, known as PLAN, is designed and implemented. Active extensions, which are not mobile, form the middle layer of SwitchWare architecture. They communicate with other routers via active packets. It is programmed in Caml language. A secure active router infrastructure layer provides a secure foundation on which the above two layers are built. Its role is to ensure that the presumptions of the other system elements are true. Active Node Transfer System (ANTS) [6] provides a capsule programming model. 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 Active Network Encapsulation Protocol (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.

ACTIVE NETWORK FUNDEMANTELES

A central feature that distinguishes Active Netwrks (ANs) from configurable ones is the programming model. While a configurable network aims to establish a maximal set of high-level features that can be programmed (or configured) with a single action, 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) [6]. Many different 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 (code) 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 [8].

Active Node Architecture

DARPA Active Network Working Group (ANWG) [9] defines the fundamental parts of an active node and how they interoperate. 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 implement 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 in band approach, it is considered by many researchers to be a de-facto standard [8].

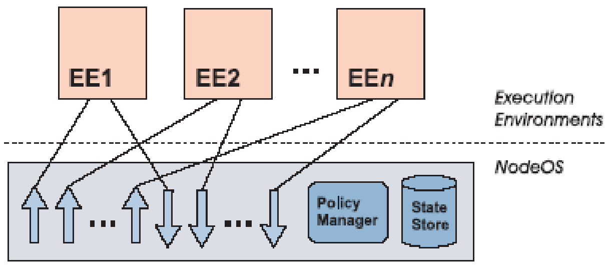

Figure 1. Active node architecture of DARPA

Networking Stack in Windows OS

Windows OS network architecture [10] may be imagined as shown in figure 2. Components that contribute the same horizontal level, in the figure, provide similar functionality. 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 [11]:

- 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.

Network Protocols (such as TCP/IP and IPX/SPX) Layer lays above the NDIS. Network protocols (also called transport protocols) provide services to their transport clients. At its upper edge, a transport protocol driver has a private interface to a higher-level driver in the protocol stack. This interface is called the Transport Driver Interface (TDI). The TDI provides a standard interface between network protocols and TDI clients of these protocols (such as network redirectors or networking APIs). TDI clients are kernel-mode device drivers that usually implement the kernel-mode portion of the upper networking API's implementation [10]. The network Application Programming Interface (API) provides standard protocol-independent programming interfaces for network applications and services to communicate across a network. Whereas, InterProcess Communications (IPC) layer support client/server computing and distributed processing. Some of the services that they support are RPC, Distributed Component Object Model (DCOM), named pipes, and mailslots. The top layer of the diagram is where user applications reside [10].

THE PROPOSED SYSTEM ARCHITECTURE

This section of the work introduces the design of the proposed active router architecture.

Design Specifications

The proposed Networking Stack Model has to be "component-based" and not layered. Actually, the protocol stacks are replaced by protocol components that can be tailored and composed to perform application specific functions. The advantages of component-based design, namely code modularity, reusability, and dynamic composition, facilitate the development and deployment of custom network services.

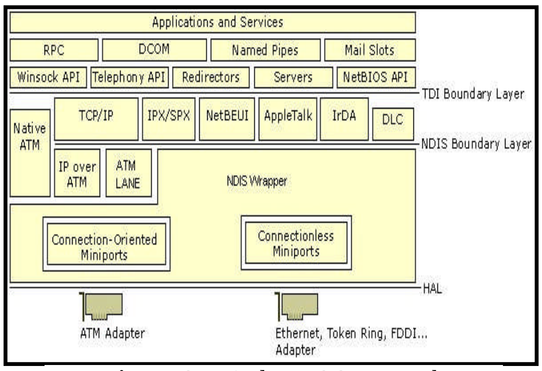

Figure 2. Windows OS Network

Concerning the Programming Model, active packet (in-band) approach in AN tends to be fairly restrictive because of the limited programming capabilities (for example, active programs are very limited in code size). 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.

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 WBAR. In other side, the PM functional part extends the OS networking stack such that it can intercept the in-bound packets that enter the WBAR 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 WBAR, which there positions are also shown in figure 3.

Component Distributor

The proposed Component Distributor (CD) allows the user to load new components in the WBAR.

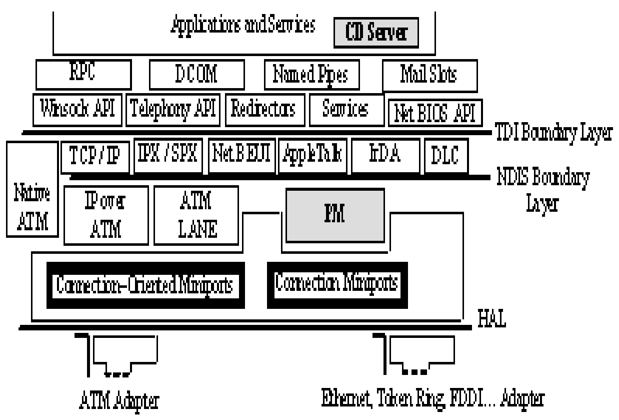

Figure 3. CD and PM units in Windows

The UC is a program performing either protocol processing or value-added function to the packet. UC may perform, for example, specialized filtering, routing, encrypting…etc. It is expected that the code of the UC is written by a Privileged-End User (PEU) or purchased from software third party, then sent from any Privileged-End System (PES) to the Active Router 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 WBAR 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 WBAR 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 WBAR 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 sequence they must be composed. Section 4.5 clarifies the necessary glue mechanism of UCs. 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 WBAR. 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 WBAR represents the server, whereas the PES represents the client. Hence, jobs of the CD can be summarized as: packing, uploading and controlling the UC.

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 wishes for processing. Consequently, the PM architecture was further divided into the following functional units: The Packet Interceptor\inJector (PIJ), Packet Filter (PF), Packet Bridge (PB), Packet Classifier (PC) and Packet Dispatcher (PD). A simple block diagram of the designed PM is shown in figure 4. The following sections illustrate the PM units briefly.

Packet interceptor/injector (PIJ)

PIJ 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. However, the PIJ consists of two edges (like any other IM driver), the protocol edge (sometimes called physical or lower adapter) and the miniport edge (sometimes called virtual or upper adapter). However, the delivered packet is not a simple sequential row of bytes. Instead it is configured in the NIC MiniPort driver in a memory descriptor list (MDL) fashion.

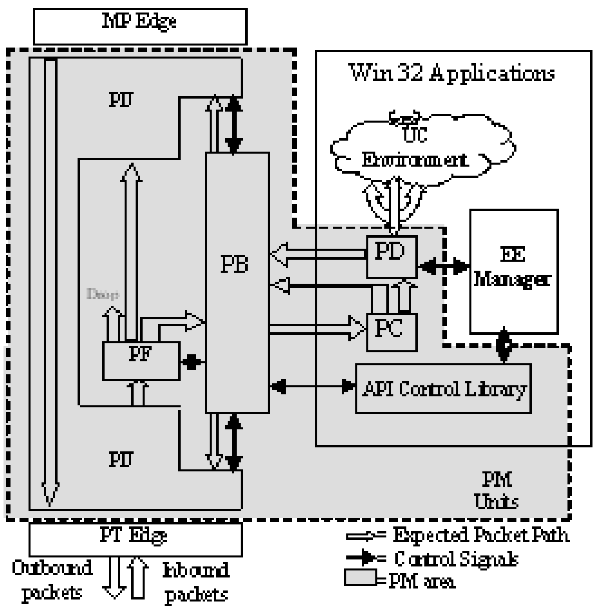

Figure 4. Block diagram of PM architecture Architecture

Packet Filter (PF)

The proposed project provides a programmable Packet Filter (PF) on the read handle. The read handle of IM driver may be the protocol edge or miniport edge, according to the direction of transmission of packets. The PF actions are:

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

Actions can be used in combination. For example, the combination of Block and Read (we will call this case "claim") action cause a matching packet to be redirected to the packet bridge and inhibited from continuing in its normal passage in the network stack of the WBAR.

Packet Bridge (PB)

The project establishes so–called Packet Bridge (PB) to provide the means 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 WBAR. However, to make the system more flexible 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)

If PB lifts a packet up to user mode, PC unit will receive it. 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.

A) Component Packets (CP):

It carries a code–related content in its payload. Accordingly, the CPs are further split into two types: the upload packets and control packets.

A.1) Upload Components Packets (UCP): It carries the code (or program) of the user component in its payload.

A.2) Control Components Packets (CCP): The installed UCs in the WBAR can be controlled (uninstalled or replaced) by sending CCP from a PES.

B) Data Packets (DP):

This type consists of packets that carrying data (not code) related contents. It may be traditional or active packets.

B.1) Traditional Data Packets (TDP): Packets that are exchanged usually in the current computer networks have been referred to as TDP in this paper.

B.2) Active Data Packets (ADP): ADPs are those packets expected to be interchanged among the AN elements (nodes and end–users). An IP option is inserted in the ADP to meet the requirements of AN from the point of view of composition of services.

Any packet belongs to the CPs is re–injected by the PC to the virtual adapter (miniport edge) of the proposed PIJ unit. This allows for these packets to complete their journey up to the component distributor. Also, the TDPs are feed backed by the PC to the virtual adapter to be processed traditionally in the protocol driver. In the proposed system, the windows 2003 server operates as a traditional router to serve such TDP packets. Only the ADPs are forwarded to the packet dispatcher unit to be processed actively by the suitable UC(s).

Packet Dispatcher (PD)

PD defines the "route" through the UC space for the ADPs passing the WBAR. The PD plays a central role in the service composition process. It determines, based on the IP 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.

Service Composition

A special working group has evolved within the DARPA active network program with the goal to investigate and standardize mechanisms for the composition of active services on EE. In this work, a router alert option of IP packet has been chosen as a means to assign packets passing a network node to active computations. This IP option is used for retrofitting active networks to current IP networks. The presence of such an option in the IP header alerts the router to the fact that it should look at the packet payload more closely. Seeing the active packet, the router then goes on to process the packet as required. A specific option which adhere to the generic type-length-value format of IP option is defined. Two fields in the option value (or data) are proposed. The Component Count (CC) field and Service Composition (SC) field. CC field (8 bit) 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 SC field which is a variable-length. The SC field consists of the CIDs of the components that must be composed to create the required active service. The sequence at which the CIDs appear in the service composition field is considered as the sequence of the components that will be executed in the WBAR. Using this format, a service composition capability and interoperability with active and non-active networks can be achieved. 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. In this paper, the concept of component-based services is envisaged to achieve a good flexibility in introducing functionalities for the ADPs.

Modes of Operation

According to the specified architecture, four modes of operation of the proposed AN can be recognized; Upload, Control, Active Data, and Traditional Data modes.

Upload Mode

The upload mode of AN operation targets the transferring of one or more UC(s) from one of the PESs to the WBAR. As shown in Figure 5, this mode of operation does not involve any transmission beyond the WBAR.

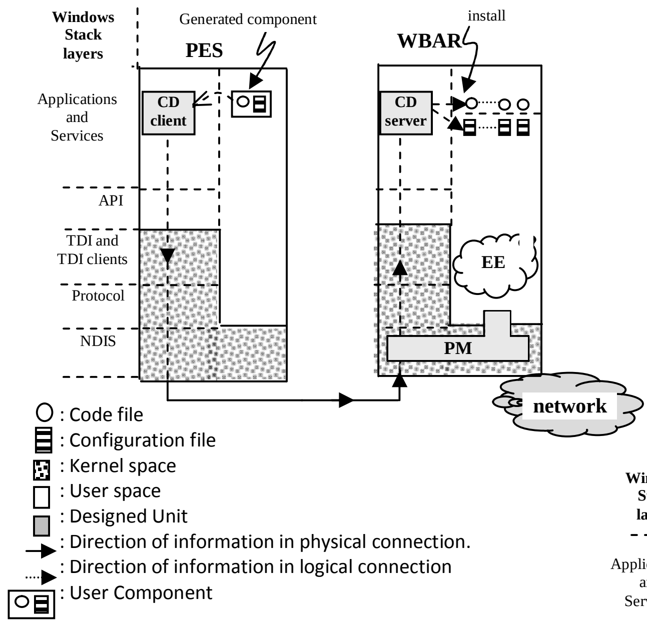

Figure 5. Upload Component mode

Control Mode

The purpose of this mode is the control (uninstall or replace) of the previously installed UCs, as shown in Figure 6. 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 WBAR.

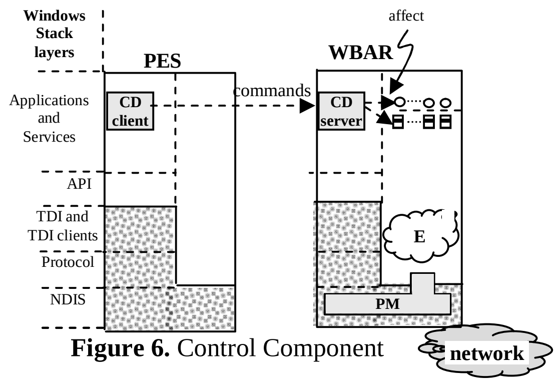

Figure 6. Control Component

. Active Data Mode

This mode represents the envisaged operation of the designed AN (see Figure 7). It constitutes the transmission of Active Data Packets (ADPs) between the network and any ES in the target LAN (in both directions), passing through the WBAR. ADP contains a router alert option in its header.

Traditional Data Mode

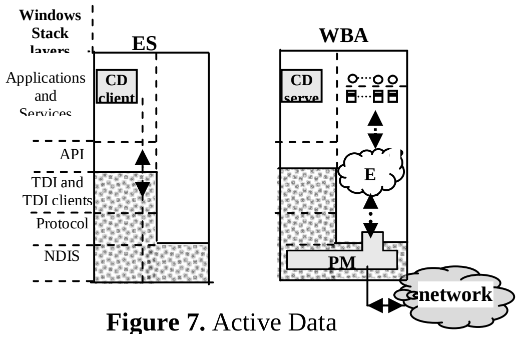

Figure 7. Active Data

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 Figure 8.

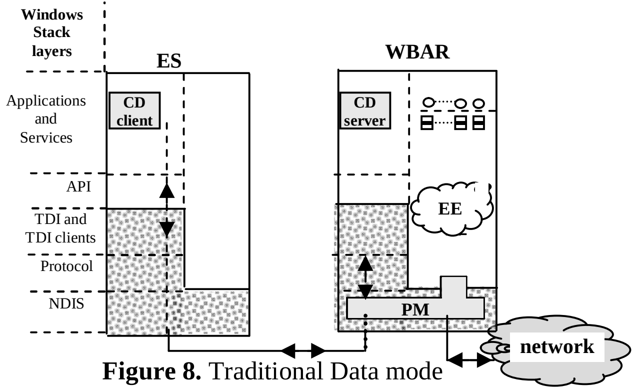

Figure 8. Traditional Data mode

PROTOTYPE IMPLEMENTATION

The WBAR prototype implementations outlined in this section is being built upon a Personal Computer with Microsoft's Windows 2003 server (which supports basic routing functionality). All widows' computers have internet explorer-based FTP service. Using FTP protocol, UCs can be uploaded easily by drag and drop from the PES to the WBAR. The proposed UC is implemented as two files: code and configuration files. The code file is a DLL library. It contains the required processing to be applied on the ADPs. The configuration file is an initialization (ini) file that contains configuration data such as the path of code file of UC.

The foremost step in realizing the PM

architecture is the implementation of a simple "pass through" IM driver. PCAUSA Corporation [12] 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 software is a group of c-language modules. In our work, cloned-packet software was modified and extended to implement the PIJ, PF and PB units of PM architecture. At this point, a copy of a complete, received, filtered packet has been gained

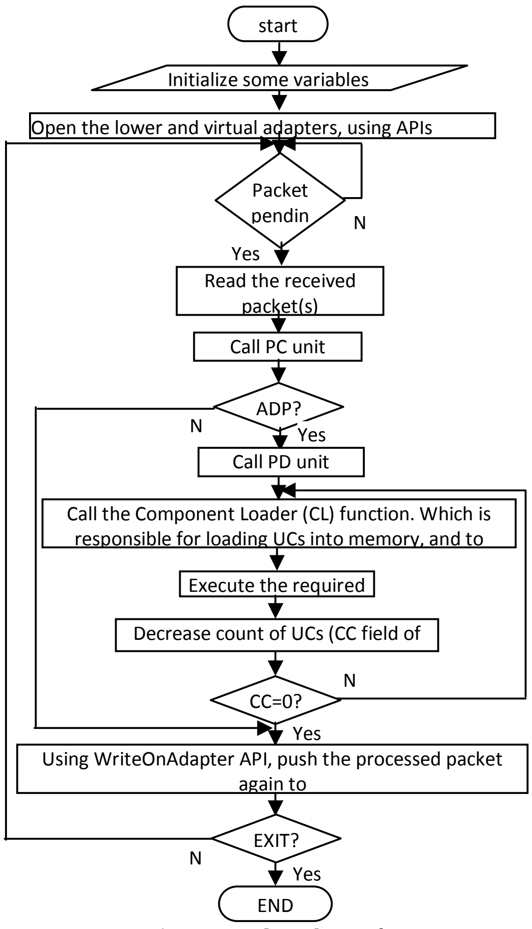
start
Initialize some variables
Open the lower and virtual adapters, using APIs
Packet pendin
N
Yes
Read the received packet(s)
Call PC unit
ADP?
N
Yes
Call PD unit
Call the Component Loader (CL) function. Which is responsible for loading UCs into memory, and to
Execute the required
Decrease count of UCs (CC field of
N
CC=0?
Yes
Using WriteOnAdapter API, push the processed packet again to
EXIT?
N
Yes
END

Figure 9. Flowchart of EEM

and queued in the user-space of the WBAR. The PC is realized in a discrete c++ function. After completing the 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. One determines how many UCs must be called and the second withdraws the CIDs of the required UCs in a correct order. The third one gets a copy of contents of the configuration file associated with the UC. The software linker between the CD server and the PM in the AR is the Execution Environment Manager (EEM). The implemented EEM is shown in Figure 9.

CONCLUSIONS

Several conclusions about the design of WBAR architecture can be drawn from this work:

* Component-based active router architecture enables network programmability through extensibility of router functionality and services.

* The IP option-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 IP header.

* A split implementation across both kernel and user space 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.

REFERENCES

[1] P. L. Simeonov (2002). The wandering logic intelligence, a hyperactive approach to network evolution and its application to adaptive mobile meltimeidia communication. PhD dissertation, F*aculty of Informatics and Automation*, Technology Uuniversity of Ilmenau.

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

[3] G. Barish and K. Obraczka (2002). World Wide Web catching: trends and techniques. IEEE Communications Magazine, May 2000.

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

[5] D. S. Alexander, W. A. Arbaugh, M. Hicks, P. Kakkar, and J. M. Smith (1998). The SwitchWare active network architecture. IEEE Network, vol. 12, no. 3, pp. 29-36.

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

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

[8] S. Schmid (2000). LARA++ Design Specification. Report on the next generation active router architecture of Lancaster University, Computing Department, Lancaster University, UK.

[9] Active Networks Working Group, K.L. Calvert (Ed.) (1998). Architectural Framework for Active Networks. Draft, August 1998.
[10] D. A. Solomon and M. E. Russinovich (2000). Inside Microsoft Windows 2000. Third Edition, Microsoft Press, USA.
[11] Microsoft Corporation (2000). Microsoft Windows 2000 Driver Development Kit, Network Drivers. Microsoft Press, USA.
[12] T. F. Divine. (2006). NDIS IM driver samples for windows NT and higher. Online article, USA. Available at: www.pcausa.com.

معمارية محدد المسار الفعال المبني على نظام النوافذ

سفيان تايه فرج عمر علي عذاب قاسم موسى العبيدي

E-mail: sufyantaih@yahoo.com

الخلاصة : نتيجة لأننا بدأنا نستطيع الحصول على قوة حاسوبية بكلفة أقل وبشكل مستمر، فإن كثيرا من الفعاليات والوظائف أصبح بالإمكان تأديتها داخل الشبكة، مما يتيح تقديم خدمات أفضل للمستخدمين. ومن الأمثلة على ذلك دعم نوعية الخدمة والإرسال المتعدد والحركية والأمنية. ورغم ذلك فإن جزءا مهما من هذه الفعاليات لازال مفتقدا في كثير من نظم الشبكات. ويعتبر التشبيك الفعال خطوة مهمة لتحسين التراكيب الثابتة وغير المرنة للشبكات الحالية. يركز هذا البحث على تصميم وتطوير التقانات التي تسمح بإطلاق كبير للفعاليات ضمن الشبكة، حيث يقدم البحث التصميم والتنفيذ لمعمارية محدد مسار فعال يعمل في بيئة نظام النوافذ من مايكروسوفت. ويقدم محدد المسار الفعال هذا المرونة اللازمة لتطوير الخدمات المستقبلية للشبكة. ولقد تم التنفيذ وإجراء التجارب باستخدام حاسوب يعمل بمشغل إنتل بنتيوم-٤ وسرعة ٢ غيغا هرتز. كما تم استخدام لغة البرمجة "سي" لغرض تطوير النظام في بيئة النوافذ. وقد أعطى هذا لبحثنا أهمية خاصة كون نظام النوافذ نادرا ما يستخدم لتطوير هكذا مشاريع بسبب تعقيد وصعوبة البرمجة في نواة النظام.