Research Journal of Applied Sciences, Engineering and Technology 5(7): 2384-2391, 2013
DOI:10.19026/rjaset.5.4669
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: July 26, 2012 Accepted: September 03, 2012 Published: March 11, 2013

Research Article

Terminal Design in Vector Network based on Windows Platform

[1]Aqun Zhao, [2]Yi Lu, [3]Yongnan Weng
[1]School of Computer and Information Technology, Beijing
Jiaotong University, Beijing, 100044, China
[2, 3]Beijing Subway Operation Technology Centre, Beijing, 100082, China

Abstract: The research work of this study focuses on the design and implementation technology of terminal in Vector Network (VN) based indows platform. The VN is a kind of new communication network with vector address as the switching adon Wdress. The premise of successful deployment of VN is its integration with the current IP networks, so it is necessary to study the implementation technology of VN terminal on the base of IP terminal. Firstly, a kind of software implementation method of VN terminal and a kind of integration method of VN and IP networks named "IP over VN" were proposed in this study. Secondly, the VN driver module was designed and implemented based on the NDIS driver interface and the key technique in the implementation was summarized. Finally, the experiment network was built to test the functions of VN terminal. The test results validated the rationality of the design and implementation scheme of VN terminal. The work of this study establishes the foundation for the deployment of VN and provides an example to the development of similar systems.

Keywords: IP over VN, NDIS driver, vector network, VN terminal

INTRODUCTION

With the extensive application of information technology, Internet has became the indispensable tool for people working and living, rather than a special experiment network only. Now communication networks including telecommunication network and Internet can no longer satisfy people's information communication demands for multimedia, broadband access, mobility, personalization and intelligent, which brings about the Next Generation Network. In recent years, research institutes both at home and abroad has started the study of Next Generation Network one after another. For example, GENI (2011) and NetSE (CCC, 2011) in America, FIRE (CORDIS, 2011) in European Union, AKARI (2011) in Japan, FGFN (ITU-T, 2011) of ITU-T and the universal network and Pervasive Services (Li *et al.*, 2010) in China.

Vector Network (VN) is a try in this background. Control separated from data, ID separated from locator and fractal architecture is its features. So it is possible to implement limitless expansion, security and credibility, QOS (Quality of Service) and other network demands based on the architecture of VN. This is a definite advantage to VN over IP network.

The built of Next Generation Network is a complex system engineering involving theory of foundations and engineering innovation. For the extensive application of IP network, it will be a long time process to design, deploy and popularize new network architecture. Therefore, to promote the research and built of Next Generation Network, we need to base the Next Generation Network on the existing Internet infrastructure, adopt an integrate and open method. By establishing revolutionary new network architecture and integrating it with IP network, we may promote the transference of network transaction from Internet to the new network and finally realize a smooth transition from IP network to Next Generation Network.

Here we use vector network technology to upgrade IP network. That is VN is used within a small area and coexist with IP network in the initial stage. While making upgrades to IP network, how to change IP terminal into VN terminal is a problem we should resolve first. The research work of this study focuses on the design and implementation technology of terminal in vector network based on Windows platform, meanwhile, the key technique in the implementation was summarized.

VECTOR NETWORK

The authors proposed a new kind of switching address called vector address (Liang, 2009; Zhao and Liang, 2012). Different from node-based coding and link-based coding methods, VA is coded by numbering the ports of a node machine which can be named port-based coding method. The network which is constructed

Corresponding Author: Aqun Zhao, School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China

based on this kind of forwarding address is called vector network. The forwarding machine in VN is named Vector Switch (VS). The routing device in VN is called vector router. Vector switch and vector router composed the VN route-switch system (Qun and Liang, 2008).

In VN, the ports of an electronic device is numbered from 1, the number allotted to a port is called port number. We define a VA from a source node to a destination node as a sequence of output port numbers of nodes along a path from the source to the destination. The port numbers in the sequence look like direction indicators guiding the packets to arrive at the destination step by step. This is why this kind of forwarding address is named VA. Each port number in VA is called element address. In practice, VA is expressed in binary format. The digit capacity of an address field is determined by how many ports the corresponding forwarding device has. For example, if an electronic device has seven ports, it will need three bits to describe all the ports.

The following data forwarding method is applied in VN. When a VS receives a data packet from one of its input ports, it extracts the first element address of the VA from the packet, deletes this element address from the packet and sends the packet to the output port pointed by this element address.

There are two kinds of vector-packets in VN: vector-data-packet to carry user data and vector-control-packet to carry control message. Both the data-packet and the control-packet include 1 byte of Head which can be extended to 4 bytes when necessary. The T bit was included in Head to allow different types of vector-packet (for example, T = 0 represents a data-packet while T = 1 represents a control-packet). In vector-data-packet, Head is followed by VA. VA consists of padding, flag and some address fields (the length of VA is variable theoretically, we assume it is 1 byte in the experiment), Data field follows VA is used to carry the upper-layer segment to be delivered to the destination. In vector-control-packet, Head is followed by Cmd field which is used to differentiate different kinds of control messages. Signaling field after Cmd field carries different kinds of control messages.

DESIGN OF VN TERMINAL

Implementation mode: There are two implementation modes of terminal in vector network: hardware implementation mode and software implementation mode. The so-called hardware implementation refers to linking a vector network terminal gateway device to the output port of the existing IP end and then IP network can connect to the vector network router-switch system via this gateway device. Vector network terminal gateway device which forms the VN terminal with

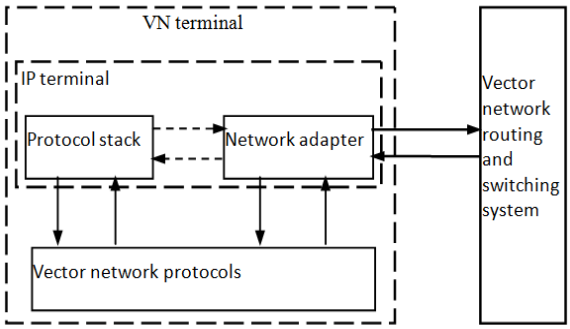


Fig. 1: Software implementation of VN terminal

physical IP terminal provides the transformation of protocols and data formats between IP network and vector network.

Figure 1 shows the software implementation mode. In this case, by adding vector network protocols to an IP terminal, then the IP terminal is physically a host or server while logically a VN terminal consists of network protocol stack and vector network protocols. This method has a lower cost and a better practicality in comparison to hardware implementation, because we just add some protocols into the software level rather than change the physical structure of the host. What's more, its flexibility and ease of modification features enabled us to upgrade some vector network protocols easily, what we need to do is amending the protocol software and reinstalling it. Hence, software implementation mode is much more convenient, flexible and high-performance.

In this study, we adopted the software implementation mode. The vector network protocol software is designed and implemented according to the functional requirements of the system. Then we installed the software to IP terminals and upgraded them to VN terminals. For Windows is one of the mainstream operating systems and most of the desktop platforms and business servers all use Windows, so we chose Windows Operating system as the development platform.

Integration method: The key point of the implementation technology of VN terminal on the base of IP terminal is the design of integration method of VN and IP networks. An integration method named "IP over VN" was proposed in this study, in reference to the fact that IP packet becomes Vector-packet after being added to a vector network head. Figure 2 shows the packing and unpacking process of data when using IP over VN method. At the source host, an application-layer message is passed to transport layer. In the simplest case, the transport layer takes the message and

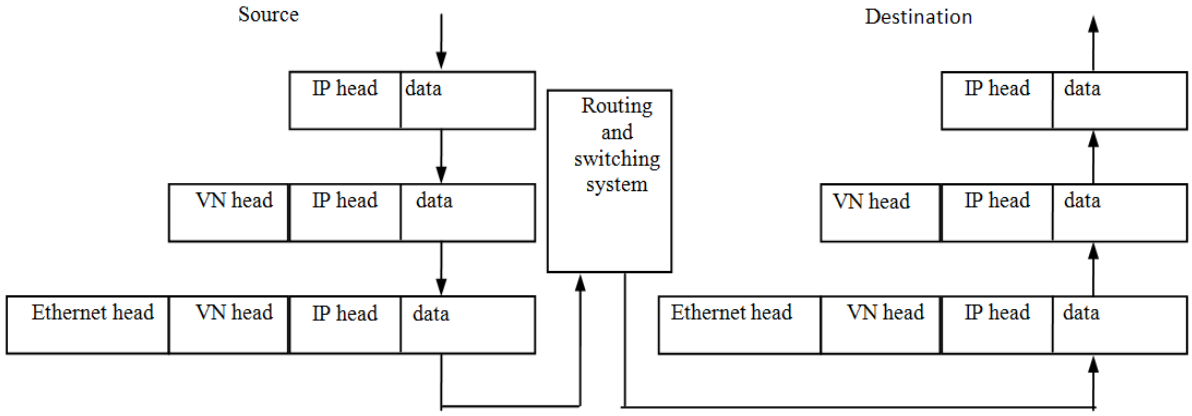


Fig. 2: IP over VN method

appends transport-layer header information. The application-layer message and the transport-layer header information together constitute the transport-layer segment. The transport layer then passes the segment to the network layer, which adds IP header information, creating an IP datagram. The datagram is then passed to vector network protocol-layer, which will add VN header information and create a Vector-packet. Later the packet is passed to the link layer (Here takes Ethernet for an example), which will add its own Ethernet header and tail information (Fig. 2 just illustrates the header information) and create an Ethernet frame. Then via physical layer the frame is sent into VN router-switch system, where the switches forward the data-packet according to its VA in VN header until to the end host. Once the destination host receives an Ethernet frame, network adapter will examine the Ethernet header information of the frame to identify whether it is a Vector-packet or not. If it is, then the vector-packet is delivered up to vector network protocol-layer which will handle VN header information and pass the IP datagram up to network layer. The datagram at last is delivered up to application program after the handling of network layer, transport layer and application layer in order.

From the packing and unpacking process we can see that the change of terminals focus on the part between network layer and link layer. That is, a vector network protocol-layer is added between the two layers. By using IP over VN method, IP datagram is successfully transformed into Vector-packet and this operation is transparent to users and upper protocols. Because, in the process of sending and receiving of data, both encapsulation and handling of application layer, transport layer and network layer are the same as before. Therefore, IP over VN method enables IP terminal just needs a little change to upgrades to VN terminal, thus, we can accomplish the integration of IP network and vector network.

Module structure: NDIS is not only able to capture all the datagram from and to the host but also able to analyze and change the datagram easily. NDIS driver interface is the most suitable network driver to cope with the intercommunication between vector network protocol layer and link layer and network layer (Wu, 2008). Therefore, NDIS driver is chose to help develop vector protocol in this study. We treat vector network protocol software as a NDIS Intermediate Driver and the implemented software module is named vector network driver module.

Figure 3 illustrates the relation between vector network driver module and the driver modules upper and lower layer. The upper layer, which is used to send-receive and handle network layer datagram, corresponds to NDIS Protocol Driver; The lower layer corresponds to NDIS Miniport Driver, its job is to accomplish sending-receiving and handling of link layer frame via network adapter.

Vector network driver module mainly consists of protocol driver receiving thread and Micro port driver sending thread. When Micro port driver module at the lower layer receives a data frame, its receiving thread will handle the frame and then deliver the data of the frame (vector-packet) up to the protocol driver receiving thread of vector network driver module, Which will call the user-defined de-packetize function to analyze the type and content of the packet, then removes the VN head, creating a new datagram (IP datagram). The datagram is then delivered up to the receiving thread of the upper layer protocol driver module. Upper layer

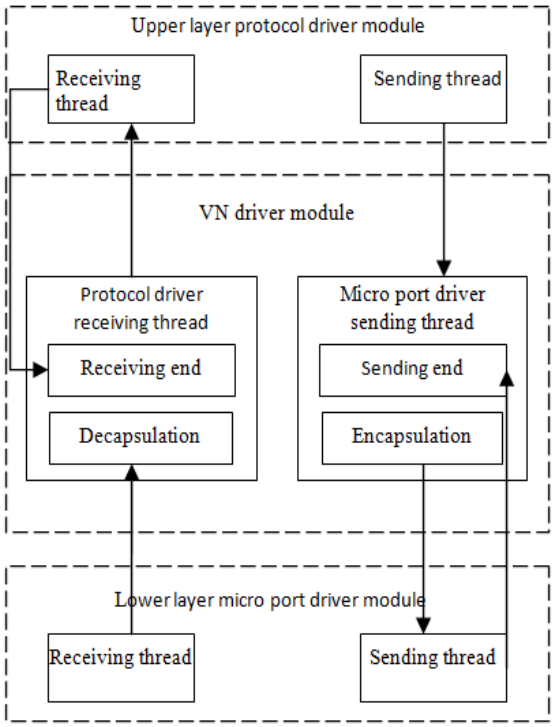


Fig. 3: Module structure of VN terminal

protocol driver module calls its sending thread to handle the incoming datagram and then passes the data (IP datagram) to Micro port driver sending thread of vector network driver module, which will call the user-defined packaging function to append VN head and create a new packet (vector-packet), then the new packet will be passed to the sending thread of the lower layer Micro port driver module.

In addition, the receiving and sending end function are also two very important functions to vector network driver module. After the receiving thread receives the incoming datagram completely, the upper layer protocol driver module will call its receiving end function to carry out subsequent operations, for example, release the system resource that no longer needed. Similarly, the lower layer Micro port will call its sending end function to carry out subsequent operations when the sending thread has finished its work.

KEY TECHNIQUES

In this study, we chose Visual Studio 6.0 (GAO *et al*., 2007), Windows 2003 Server SP1 DDK (Microsoft, 2005) and Driver Studio 3.2 (Compuware, 2009) as the tools to develop NDIS Intermediate Driver.

Packets identification and classification: When a frame arrives at a network adapter, the adapter needs to know to which upper layer protocol it should pass the contents of the data field. Ethernet frame use a type field to distinguish a datagram from others. For example, if the type number is 0x0800, then we know that the content of the data field is an IP datagram which should be delivered up to IP protocol. In order to support vector network, we give the definition that the type number of vector network protocol is 0x0810:

~~#include <hash_map>~~
~~typedef hash_map<ULONG, USHORT, hash<~~
~~ULONG>, equal_to<ULONG>, non_paged_alloc>~~
Map
~~typedef hash_map<ULONG, USHORT, hash<~~
~~ULONG>, equal_to<ULONG>,~~
non_paged_alloc>::iterator Iter
~~Map m_info~~

The process of packet identification and classification is shown in Fig. 4. Network adapter reads the type field of the incoming data frame and decides to which protocol it should pass. If type number is 0x0810, then the datagram will be delivered to vector network driver module, which reads VN header to know whether it is a vector-data-packet (T = 0) or vector-control-packet (T = 1). If the former, then the VA field will be checked, null indicates that it is the right data packet whose destination is the host. Or else, the data packet should be abandoned for it arrived at a wrong place. As to the latter, we should check its Cmd field further and carry out the control function according to the type of the data packet.

Datasheet storage and update: Because there are many tables to store a variety of information, we chose C++ STL standardized template library to implement information storage, and accomplished the function of handling regularly and refreshing of the data by using class KNdisTimer provided by Driver Studio.

Taking the storage of "destination IP Address, VA" for an example, the saving and querying of the destination IP Address at terminals are very frequent, so we could treat IP Address as the key and use map template to implement the mapping from IP Address to VA. Using the tree search method makes map template has a low efficiency when it comes to a large scale of data, while hash_map which uses hash function search method is much faster in comparison with map. Therefore, we decided to use hash_map to store information. The relevant code as follows:

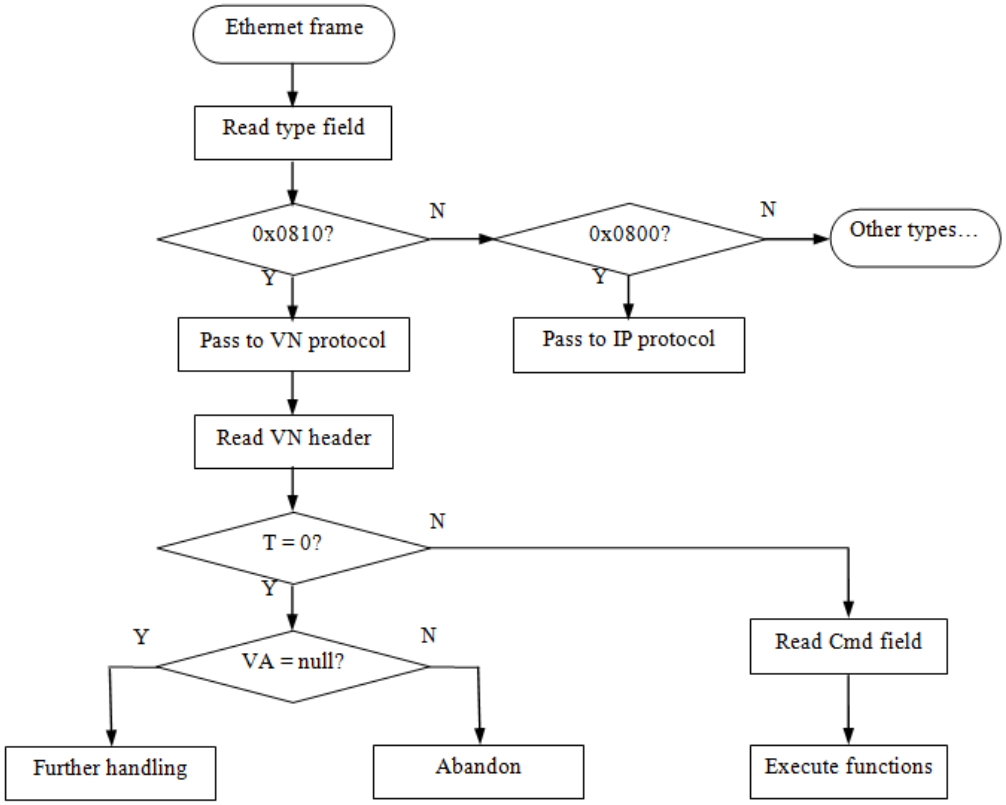


Fig. 4: The process of packet identification and classification

```
#include <hash_map>
typedef hash_map<ULONG, USHORT, hash<
ULONG >, equal_to< ULONG>, non_paged_alloc>
Map
typedef hash_map<ULONG, USHORT, hash<ULONG
>, equal_to<ULONG>,
non_paged_alloc>::iterator Iter
Map m_info
```

As described above, IP Address is stored in ULONG type and serves as the key of hash_map, while VA which is the value of hash_map is stored in USHORT type. Besides, we defined a iterator which is used to traverse hash_map. In order to examine hash_map regularly and delete the item whose value is null, we made use of class KNdisTimer provided by Driver Studio, and add the code as follows in the class when its member variables were defined:

```
KNdisTimer m_Timer
KNDIS-DECLARE-TIMERCALLBACK
(StlFilterAdapter, Timeout)
void Timeout ()
```

We used callback function register statement to assign the callback function of KNdisTimer in the driver, so that function Timeout could be callback when time arrives and the program can carry out preset operations. For example, the program could refresh data items regularly once the refresh task is added in function Timeout.

Construction of data packets: Vector network driver module uses IP over VN method to add VN header to the incoming IP datagram, creating user-defined data packet inherits from IP datagram and implement caching and queuing function by using data packet queue technology. To construct data packet, we used the class KNdisPacket and KNdisBuffer, while class KNdisBufferHeap and KNdisFilterPacketPool as the auxiliary classes. The relevant code as follows:

```
Typedef KNdisFilterPacketPool<PacketContext,
true>CTxPool
typedef KNdisFilterPacketPool<PacketContext, false>
CRxPool
KNdisBufferHeap m_BufferHeap
```

```
CTxPool m_TxPool
CRxPool m_RxPool
```

The constructed user-defined data packets then join dispatcher queue waiting for their turn to be sent. We used class KNdisInterlockedPacketList defined by Driver Studio as the queue container. The relevant code as follows:

```
KNdisInterlockedPacketList m_SendDelayQ;
KNdisInterlockedPacketList m_ReceiveDelayQ;
m_SendDelayQ (BOOLEAN(FALSE));
m_ReceiveDelayQ (BOOLEAN(TRUE));
```

Maximum transmission unit and network byte order processing: Maximum Transmission Unit refers to the max size of the datagram that communication protocol allowed. MTU of Ethernet is 1500 Bytes, which means that once the size of IP datagram is larger than 1500 Bytes, IP datagram will be fragmented into several small datagram's. Add VN header to IP datagram may result in the size of Vector-packet exceeds the limit of MTU and so causes error. To solve the problem above, we change changed the size of MTU to 1480 Bytes which is the new standard for fragmentation of IP datagram, reserving 20 Bytes for VN header. We change MTU in Windows registry to 1480 Bytes when we install vector network driver module and recover it as we uninstall vector network driver module.

The store order of data that is larger than 1 Byte in size in memory is called byte order. Byte order is different according CPU types, so byte order is also named host byte order. Little endian and Big endian are two Common types. The former treats the low-order byte as starting address; on the contrary, the latter treats the high-order byte as starting address. CPU of X86 series uses little-endian byte order. Network byte order is a kind of description format of data that TCP/IP protocol requires, which has nothing to do with CPU type and OS and applies big-endian form. As the above shows, host byte order is different from that of network, so it necessary to implement conversion operations in VN terminal. We defined four Macros to do it. The relevant code as follows:

```
#define  HTONS(x)((((x)>>8)&0xff) |(((x)&0xff)<<8))
#define NTOHS(x) HTONS (x)
#define HTONL(x)
((((x)>>24)&0xff)|(((x)&0xff)<<24)|
(((x)&0xff0000)>>8) | (((x)&0xff00)<<8))
#define NTOHL(x) HTONL(x)
```

EXPERIMENTS

In order to test the functions of VN terminal, the experiment network which consists of VN terminals and VN router-switch system was built (Fig. 5). Each of subnet 192.168.10.0/24 and 192.168.20.0/24 has two VN terminals which run on Windows XP operating system and have vector network driver module installed. Router-switch system consists of two NA820 switch, named A and B respectively, each running vector network router-switch protocol and runs on Linux operating system. Both A and B have seven ports, but only one to three of them were used.

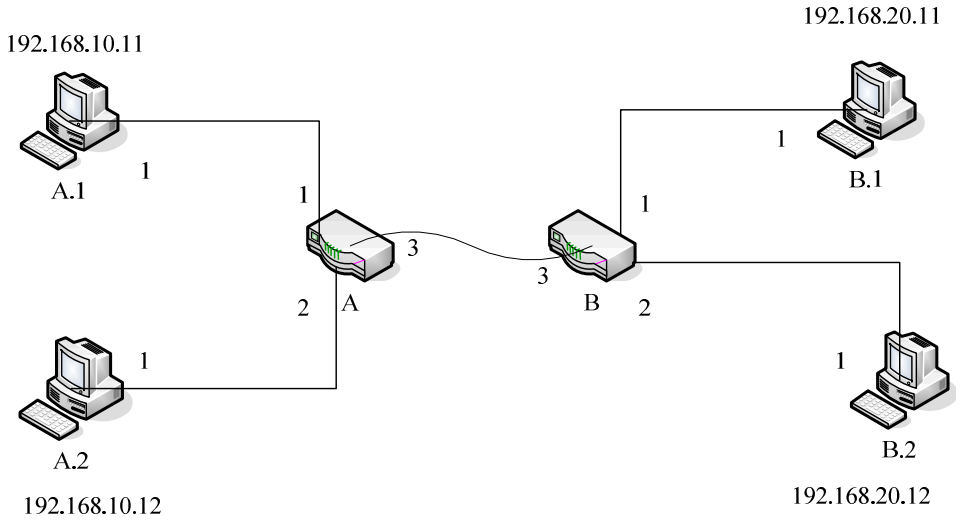
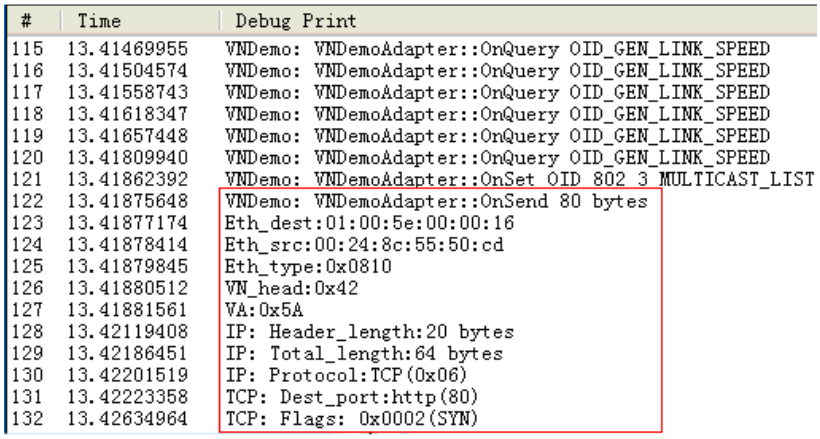
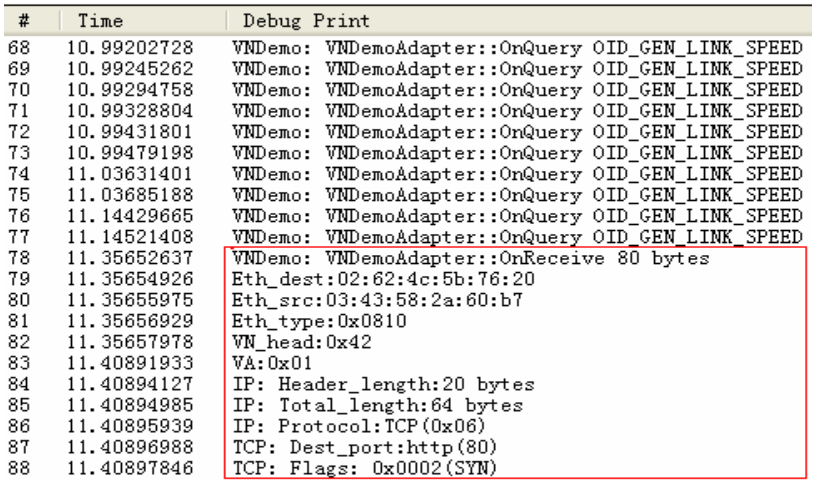


Fig. 5: Topology of the experiment network



#	Time	Debug Print
115	13.41469955	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
116	13.41504574	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
117	13.41558743	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
118	13.41618347	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
119	13.41657448	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
120	13.41809940	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
121	13.41862392	VNDemo: VNDemoAdapter::OnSet OID_802_3_MULTICAST_LIST
122	13.41875648	VNDemo: VNDemoAdapter::OnSend 80 bytes
123	13.41877174	Eth_dest:01:00:5e:00:00:16
124	13.41878414	Eth_src:00:24:8c:55:50:cd
125	13.41879845	Eth_type:0x0810
126	13.41880512	VN_head:0x42
127	13.41881561	VA:0x5A
128	13.42119408	IP: Header_length:20 bytes
129	13.42186451	IP: Total_length:64 bytes
130	13.42201519	IP: Protocol:TCP(0x06)
131	13.42223358	TCP: Dest_port:http(80)
132	13.42634964	TCP: Flags: 0x0002(SYN)

Fig. 6: The data of A.1



#	Time	Debug Print
68	10.99202728	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
69	10.99245262	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
70	10.99294758	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
71	10.99328804	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
72	10.99431801	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
73	10.99479198	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
74	11.03631401	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
75	11.03685188	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
76	11.14429665	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
77	11.14521408	VNDemo: VNDemoAdapter::OnQuery OID_GEN_LINK_SPEED
78	11.35652637	VNDemo: VNDemoAdapter::OnReceive 80 bytes
79	11.35654926	Eth_dest:02:62:4c:5b:76:20
80	11.35655975	Eth_src:03:43:58:2a:60:b7
81	11.35656929	Eth_type:0x0810
82	11.35657978	VN_head:0x42
83	11.40891933	VA:0x01
84	11.40894127	IP: Header_length:20 bytes
85	11.40894985	IP: Total_length:64 bytes
86	11.40895939	IP: Protocol:TCP(0x06)
87	11.40896988	TCP: Dest_port:http(80)
88	11.40897846	TCP: Flags: 0x0002(SYN)

Fig. 7: The data of B.2

VN terminal B.2 was treated as a server, opening www service. A1 as a client accessed B.2 for www pages via browser. The process of data packet switching was monitored by a kind of Kernel monitoring software called Debug View. Test result shows that the browser of A.1 could display www page normally and parts of the monitoring data are showed in Fig. 6.

In the screenshot of A.1 in Fig. 6, "On Send 80 bytes" means the size of data frame we sent was 80 bytes which contains 14 bytes of Ethernet header (each of source and destination MAC address occupies 6 bytes, the value of protocol type is 0x0810), 2 bytes of VN header (the value of Head field is 0x42, T = 0 indicates it's a vector-packet, VA:0x5A) and 64 bytes of IP datagram (20 bytes of IP header and 44 of TCP message). We also can see that host output port number has been deleted from VA, remaining 01011010, in which the first 1 is a tag followed by effective address. As to A and B both have 7 ports, so they need 3 bits to number all their ports. Therefore, the last two hops are 011 and 101 (3 and 2), which correspond to the output ports of A and B respectively.

Figure 7 shows the monitoring data of B.2. "On Receive 80 bytes" indicates the incoming data packet is 80 bytes in length. Note that, in the 2 bytes of VN header, the value 0x42 of Head field is same to that when the packet was sent. "VA: 0x01" means there is no address left, that is the packet received its destination.

CONCLUSION

By building the experiment network and functions test, this study validated the rationality of the design and implementation scheme of VN terminal, the availability of the integration method of VN and IP networks named "IP over VN" and the correctness of vector network

driver module. The work of this study establishes the foundation for the deployment of VN and provides an example to the development of similar systems. The future work will focus on implementing the function of VN control plane and the function performance experiment in large scale network environment.

ACKNOWLEDGMENT

This study was supported by the Fundamental Research Funds for the Central Universities (No. 2012JBM025) and in part by the Open Research Fund from Key Laboratory of Computer Network and Information Integration in Southeast University, Ministry of Education, China (No. K93-9-2010-08).

REFERENCES

AKARI, 2011. AKARI Architecture Design Project for New Generation Network. Retrieved from: http://akari-project.nict.go.jp/eng/index2.htm.

CCC, 2011. Network Science and Engineering (NetSE). Retrieved from: www.cra.org/ccc/netse.php.

Compuware, 2009. Driver Studio. Version: 3.2, Compuware Corporation Online Document.

CORDIS, 2011. FIRE- Future Internet Research and Experimentation. Retrieved from: http:// cordis. europa.eu/fp7/ict/fire/.

Gao, S.C., Y.M. Nie and J. Zheng, 2007. Visual C 6.0 Development Guide. Posts and Telecom Press, Beijing, China.

GENI, 2011. Exploring Networks of the Future. Retrieved from: http://www.geni.net/.

ITU-T, 2011. FGFN-Focus Group on Future Network. Retrieved from: http://www.itu.int/en/ITU-T/ focusgroups/fn/Pages/Default.aspx.

Li, S.Y., Y.J. Qin and H.K. Zhang, 2010. Mapping model for the service layer of universal network based on network utility maximization. Acta Electron. Sin., 38(2): 282-289.

Liang, M.G., 2009. A Method for Vector Network Address Coding. China Patent, ZL200610089302.6.

Microsoft, 2005. Windows DDK (Driver Development Kit) Version: Server 2003 SP1. Microsoft Corporation Online Document.

Qun, Z.A. and M.G. Liang, 2008. A new mobile network architecture. International Symposium on Computer Science and Computational Technology, Beijing, China, pp: 686-689.

Wu, G.Y., 2008. Computer Network Advanced Software Programming Technology. Tsinghua University Press, Beijing, China.

Zhao, A.Q. and M.G. Liang, 2012. A new forwarding address for next generation networks. J. Zhejiang Univ., Sci. C, 13(1): 1-10.