Journal of Information Hiding and Multimedia Signal Processing ©2014 ISSN 2073-4212
Ubiquitous International Volume **5**, Number **3**, July **2014**

Implementation of IP-over-DVB Network as A Wireless Internet Backbone

Chun-Hung Richard Lin, Kuan-Hua Tseng, Yu-Hsiu Huang and Cheng-Yi Lin

Department of Computer Science and Engineering
National Sun Yat-sen University
70 Lienhai Road, Kaohsiung, 804, Taiwan
lin@cse.nsysu.edu.tw; ethan@cse.nsysu.edu.tw; yhhuang@cse.nsysu.edu.tw; t0963162429@gmail.com

Received January, 2014; revised March, 2014

Abstract. *This paper focuses on implementing the DVB (Digital Video Broadcasting) based IP network. The main idea is to let DVB network interfaces emulate Ethernet network interface card (NIC). Then applications can keep utilizing socket interface for IP data stream transmitting without any modification. IP forwarding is used to carry the IP datagrams through the virtual DVB network interface. Then several NICs can be connected by DVB devices. We believe that DVB networks will be a good alternative to the wireless multimedia Internet backbone because of its reliability and robustness. Our implementation approach can keep socket-based applications working without making any modification. Broadcasting and multimedia application scenarios are especially suitable for this network system.*

Keywords: Digital Video Broadcasting, IP over DVB, MPEG-2 Transport Stream, Unidirectional Lightweight Encapsulation, High Definition, 1080p.

1. **Introduction.** Wireless transmission of full HD (High Definition) is always a critical problem for many applications. IP networks cannot provide a good solution because of bandwidth and QoS control issues. Except HDMI, we hardly find an acceptable and reliable solution. DVB, which is used for digital TV program transmission, looks like a choice. However, IP datagram transmission over DVB networks must be overcome to provide a signaling channel. Then DVB could be a solution of full HD video transmission. The DVB-T [1, 2, 3] protocol was designed for transmitting high definition digital video program through microwave instead of physical cable to reduce the constructing cost. Although there are some limitations caused by obstacles or interference, e.g., the weather, clouds and buildings, as shown in Figure 1 which are appearing on the way of video transmitting, be overcame, the reliability of DVB-T has been verified by the success of the digital television broadcasting system.

The main purpose of DVB-T is to reduce the cost on network hardware by replacing the physical and data link layers from cable to DVB-T devices as network backbone. The traditional television broadcasting system (DVB) [4, 5, 6, 7, 8, 9, 10, 11] is a one-way transmission, which is from the TV station as sender to TV set located at end-users home as receiver. If we can send IP datagrams over DVB-T channel in the other direction as illustrated in Figure 2. This makes applications on DVB-T system be able to send control signals to report status from client side to server side.y good localization features in time and frequency domains, but also symmetry, orthogonality, compact support and

high vanishing moments , which can present the face features better than scalar wavelet in each band.

A network infrastructure contains multiple LANs connected by wired routers. What we want to achieve is to replace these wired routers by wireless DVB devices as a wireless backbone. There are six sections in this paper which are organized as follows. Section 2 is related works, and Section 3 reviews the whole system. The implementation and performance test experiments are in Section 4 and Section 5. Section 6 draws our conclusions.

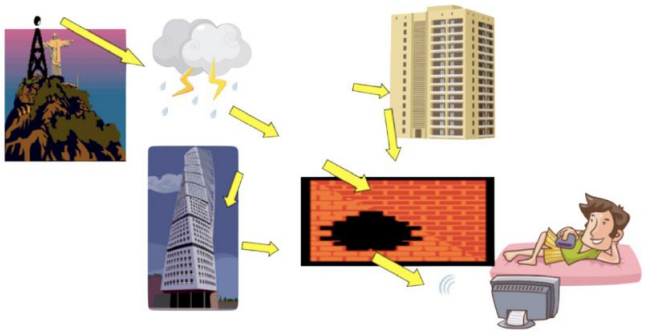

FIGURE 1. The obstacles may appear on the way of the video TV program transmission

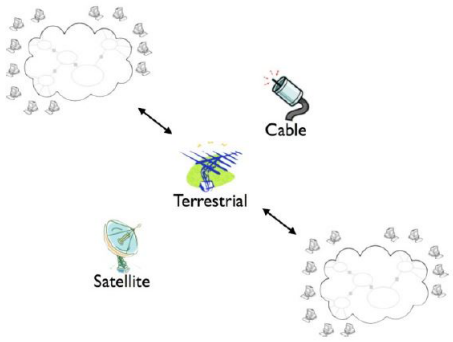

FIGURE 2. A DVB-T-backboned network

2. **Related Works.**

2.1. **Transmission protocol.** The OSI 7-layer model [12, 13, 14] defines several layers of protocols to deliver a packet from the source to the destination. DVB-T is not an exception. This main process is described as below. The minimal transmission unit of digital broadcast is Transport Stream (TS) [15, 16, 17, 18] packet. Figure 3 illustrates the packet format. It is composed of three parts, header, adaption field and payload. Header has fixed length of 4 bytes. The adapter field is an optional section for extra features. The payload is the data to be transmitted. When the sender receives a packet from upper layer, it fragments the packet into smaller units and encapsulates them as the TS packet, and then send them out. Once the receiver receives the TS packets, it decapsulates the packets, defragments them into the original packet and passes payload to the upper layer to complete the delivery process.

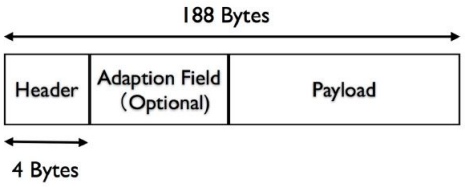

FIGURE 3. The packet format of the transport stream protocol

ULE (Unidirectional Lightweight Encapsulation) [4, 16, 19, 20] protocol is a data link layer protocol for transportation of IP packets over DVB transport streams. Figure 4 shows an IP packet generated from socket interface is fragmented and encapsulated into several DVB TS packets by the sender. Then TS packets are transmitted through DVB channel to receiver, and decapsulated and merged to the original IP packet.

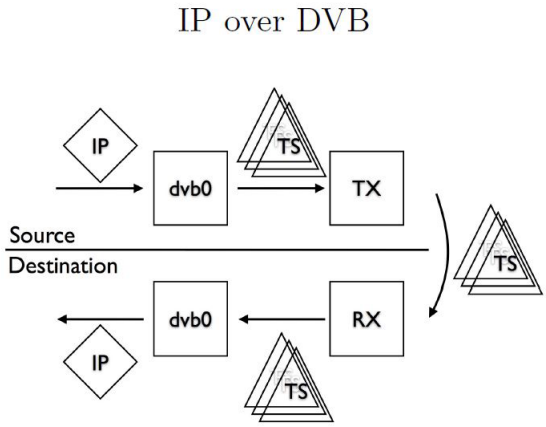

FIGURE 4. The procedure of the digital signal broadcasting

Figure 5 illustrates that when DVB driver gets IP packets from upper layer, it encapsulates these packets to ULE format with header and trailer first, then fragment the ULE packet into several TS packets, and finally send TS packets to DVB channel.

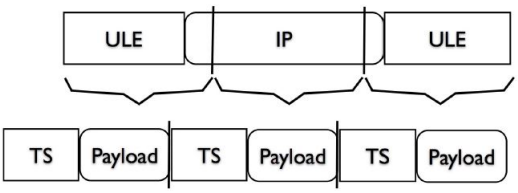

FIGURE 5. ULE packet to Transport Stream packet

The ULE packet is also called SubNetwork Data Unit (SNDU). There are two methods to fragment ULE packets into TS packets: one is padding and the other is packing.

(1). ULE Padding

Pad two bytes 0xFFFF after the Transportation Stream packet when the size of SNDU is less than the payload length of a TS packet. The remaining data of TS payload will be

treated as ignoring. This will cause bandwidth waste due to the useless data transmission, especially when every SNDU is less than the length of a TS payload. Illustrated by Figure 6.

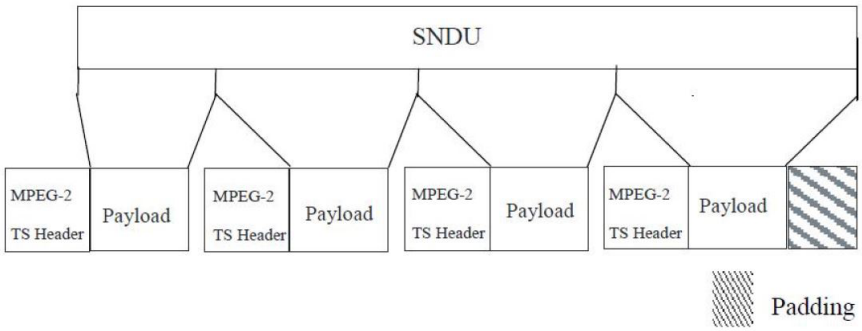

FIGURE 6. ULE padding into TS

(2). ULE Packing

Figure 7 depicts that if there still remains some space in TS payload after adding the first SNDU, the encapsulator may encapsulate further queued SNDU by starting the next SNDU in the next available byte of current TS packet payload. An 8-bit payload pointer must be inserted in the first byte directly following the TS packet header. The value of the payload pointer must be set to the position of the byte following the end of the first SNDU in the TS packet payload. If no further SNDUs are available, the encapsulator can wait a short period of time to fill the incomplete TS packet. This short period of time is known as the Packing Threshold.

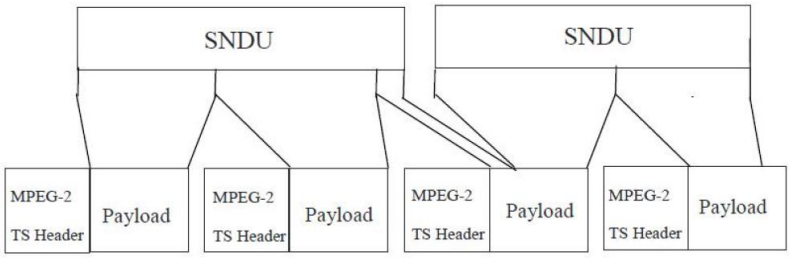

FIGURE 7. ULE packing

2.2. **The Design of I/O buffer.** URB stands for USB Request Block, which is the size of a buffer sending to or receiving from a USB device. For example, if we need to read some data from a USB device, the USB device will send back the data through the URB. Most drivers are now designed by a ring buffer which consists of multiple URBs.

The size of the ring buffer can be calculated by n URBs multiples the single URB size. Our experiment is using ITE 9507 DVB-T modulator chipset (http://www.ite.com.tw). From our test, the optimal URB size of DVB transmitter (TX) is 174*188 and the URB size of receiver (RX) is 348*188. This means that the TX will throw 174 Transport Stream (TS) packets and RX will throw 348 Transport Stream (TS) packets at a time. Both TX and RX buffers contain 16 URBs. The total buffer size of TX is 0.5 MBytes and RX 1 MBytes. This is our best experimental result.

TABLE 1. Buffer Size of Driver I/O

	TX	RX
URB size (Byte)	174*188 = 32712	348*188 = 65424
Number of URB	16	16
Total size (Byte)	523392	1046784

3. **System Overview.** Due to the uni-direction of DVB broadcasting, we have to bind a pair of transmitter and a receiver within a router in order to send and to receive TS packets at the same time. Then we connect two routers using a pair of different DVB frequencies to construct an IP-over-DVB backbone. Figure 8 illustrates the DVB backbone construction. All IP datagram can transport over a DVB channel whose bandwidth is 6 MHz and data rate is 23 Mbps. We may trunk multiple DVB channels to get a high data rate IP backbone.

Linux provides a Virtual Network Device (eth0) as an Internet I/O interface in user space. Thus, we need not directly access real network device for actually sending and receiving packets. This is demonstrated in Figure 9. Socket API is the standard of network programming. In order to make all network related applications working over the DVB network, we utilize the concept of virtual network device to implement a DVB virtual interface, say dvb0, which plays the same role as the eth0. Those system calls of Linux working in eth0 are also working in dvb0. The only different of eth0 and dvb0 is data link layer protocol and drivers. When dvb0 receives a packet from socket API, it forwards the packet through the TX module; if dvb0 receives a packet from RX module, it passes the packet to socket API to the application at user space.

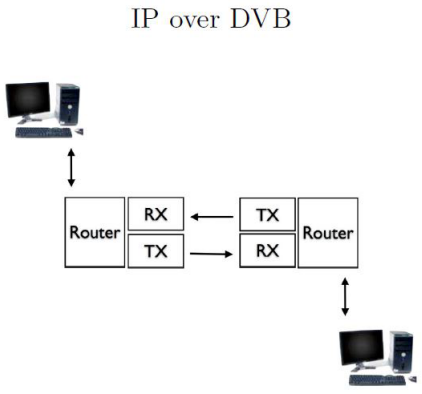

FIGURE 8. IP network backbone using wireless DVB channels

4. **Implementation of the System.**

4.1. **Modification of the Driver.** The virtual network device is in kernel space, and the char driver [21, 22, 23] cannot be used directly by applications in user space. Therefore, we additionally implement a set of primitive API within the char driver and a kernel API above the char driver which provides the same methods of the user API for kernel module

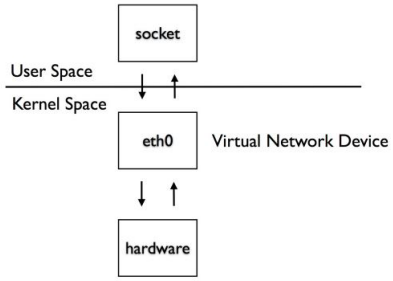

FIGURE 9. Virtual network device

usage. Then we create a virtual network device using these APIs. Figure 10 depicts the modified architecture of driver.

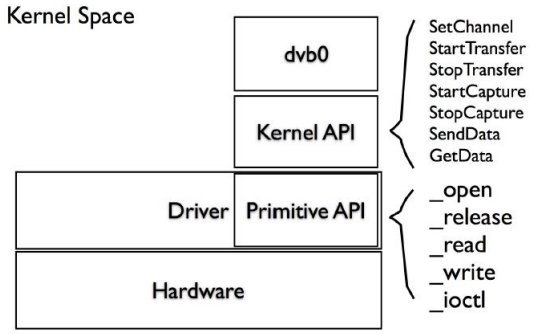

FIGURE 10. Modified architecture of driver

4.2. **Virtual network devices.**

- The lifecycle of a virtual network device:
  - Operating system detects a new hardware device installed.
    - Add a new virtual network device (dvb0).
    - Register dvb0 by calling system function "*register_net_dev*".
  - Operating system detects that a hardware device removed.
    - Close the virtual network device corresponding to the hardware.
    - Unregister dvb0 by calling system function "*unregister_net_dev*".
- Control the virtual network device
  Virtual network device must be enabled through Linux Command: *ifconfig*.
  - *ifconfig up*: Enable the virtual network device.
    - *startTransfer*: Start to transfer data.
    - *startCapture*: Start to receive data.
  - *ifconfig down*: Disable the virtual network device.
    - *stopTransfer*: Stop transferring data.
    - *stopCapture*: Stop receiving data.

To enable the DVB virtual network device, we have to notice the ARP flag. If the ARP flag is on, once a packet was received by virtual network device from an application, the operating system will issue an ARP lookup to find the MAC address of the destination. Since the DVB virtual network device (dvb0) has no MAC address, the ARP lookup

request will not be replied, and the packet will never be written into the TX module of the virtual network device. That is, this packet will not be sent out forever.

4.3. **Adjustment of the send buffer.** The data in the send buffer of the TX module is sent only when a URB is filled. This design is only considering the application of digital TV program transportation because video stream is generated in a constant data rate from the TV station. However, the Internet traffic is bursty, rather than constant. This will cause a huge waiting time when we must send a small amount of data immediately. For example, in the light load traffic situation, to avoid TCP timeout the ACK message of TCP, which is an extremely small packet, must be sent out immediately for flow control. But such a small packet cannot fill up a URB in a short time, and this will cause the ACK to fail to be sent out immediately.

To solve this problem, we add a mechanism that checks the ring buffer periodically to find whether the URB is filled or not. If not, it fills the URB with null packets so that this URB can be sent right away. However, continuously padding the URBs with null packets will degrades the bandwidth utilization greatly. This mechanism is illustrated in Figure 11.

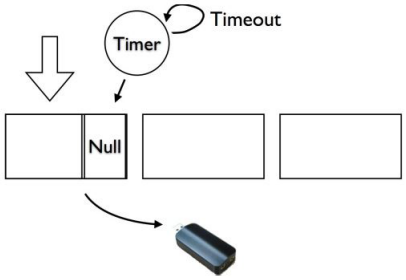

FIGURE 11. The *fast transmission mechanism* of send buffer on TX

Delay padding: In Figure 12, to consider the bandwidth utilization issue which we mention above, a small modification has been made. When the mechanism first checks the URB and finds it is not filled up, it only makes a note that this URB is not full instead of padding it with null packets. And on the second checking process, if the URB is filled up by some other data packets, then there is no need to make padding this URB again. Otherwise, on the third checking process, if this URB still remains not full, then it starts padding this URB and sending it.

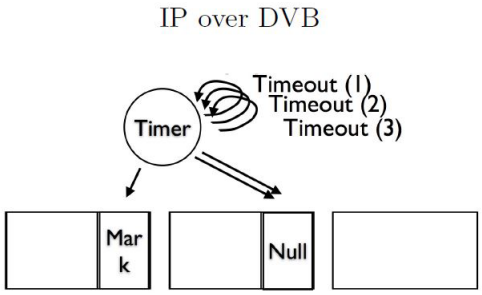

FIGURE 12. The *delay padding mechanism of send buffer on TX*

4.4. **Adjustment of the read buffer.** If we need to get the data from read buffer as soon as possible, we have to check the read buffer periodically by polling scheme. The polling operation is wasting CPU time in light load situation. Again, we make some modification to improve the performance.

The DVB driver will send null packets into read buffer to response application read request if there is no data coming RX module. If two continuous checking ring buffer and there is no any URB having data, we can treat this as an idle status and stop polling into sleeping mode for a while. Then the system just waits for the next response of the new request for whether there is new data coming. The reasons why we have to check the whole ring buffer are:

- To make sure all the data in ring buffer has already been read.
- To ensure the newer request from ring buffer is not issued by null packet.

5. Experiment Data.

5.1. **Parameters.** We are using ITE Technology Corporation DVB-T modulator 9507 chipset for our following experiments. The top data rate of bandwidth 6 MHz is 23Mbps. Show as in Table 2.

TABLE 2. Parameters of this experiment.

	RouterA	RouterB
TX Frequency (MHz)	666	533
RX Frequency (MHz)	533	666
Bandwidth (MHz)	6	6

5.2. **Performance tuning.**

1. The fast transmission mechanism of send buffer on TX: Figure 13 shows the effcient data rate of DVB channel is greatly downgraded if we padding the URB with null packets when this URB is not filled up.
2. The delay padding mechanism of send buffer on TX: Figure 14 shows the performance of the delay padding mechanism, i.e., three times URB checking mechanism. The data rate is obviously enhanced.

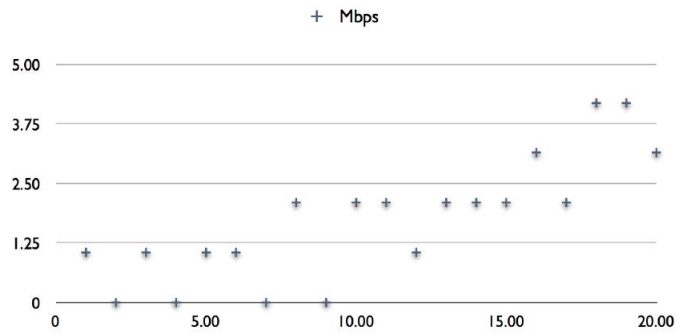

FIGURE 13. Effcient data rate of TX with rough null data padding (i.e., fast transmission padding mechanism) (X axis: seconds, Y axis: Mbps)

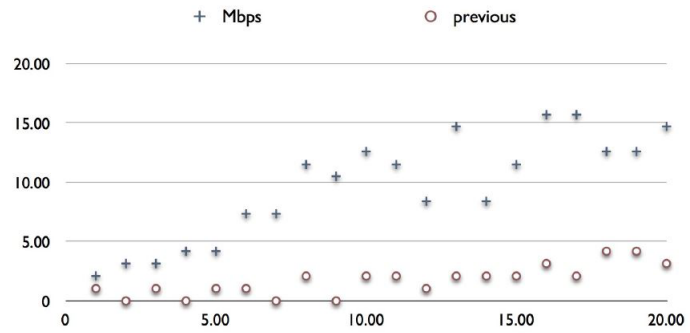

FIGURE 14. Data rate of TX with delay padding mechanism (X axis: seconds, Y axis: Mbps)

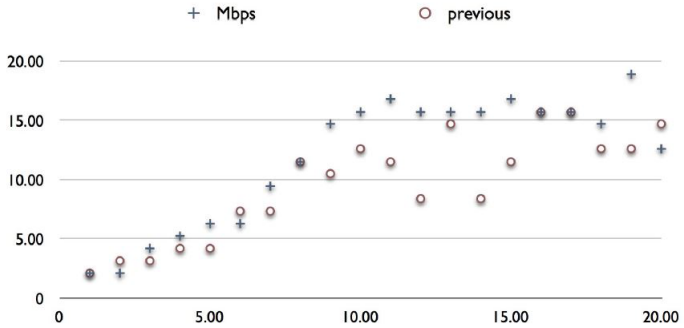

FIGURE 15. Data rate of polling mechanism with idle checking on RX ring buffer (X axis: seconds, Y axis: Mbps)

3. Polling mechanism with idle checking on ring buffer on RX: Figure 15 shows the performance of polling mechanism with idle status checking. It can further improve the data rate. The idle checking can control the frequency of DVB driver requests.
4. TCP throughput over DVB network: To speed up ACK delivery of TCP to avoid timeout in congestion controlling, we utilize the fast transmission padding mechanism

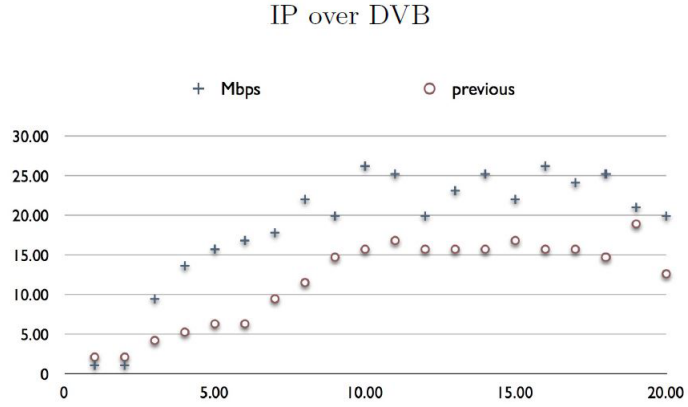

FIGURE 16. TCP throughput over DVB network (X axis: seconds, Y axis: TCP throughput in Mbps)

on send buffer of TX. ACK will be sent immediately. Figure 16 shows the TCP throughput over DVB network.

5. The average TCP performance The top speed of DVB wireless link of 6MHz bandwidth is 23 Mbps according to the test report of ITE DVB hardware equipment. However, the throughput of TCP/IP over DVB data link layer cannot have the data rate because of the overheads of the virtual network device in Linux kernel space. In this experiment, we want to measure the average TCP throughput by considering FTP application. The result is shown in Figure 17. The maximal data rate can be up to 22Mbps. However, the warm-up period (time for TCP slow-start and congestion avoidance) is up to 7 8 seconds. This should be further improved. Obviously, our implementation has very low overheads and achieve high effciency. It also reveals the importance of the enhancement of TX/RX ring buffer read/write schemes.

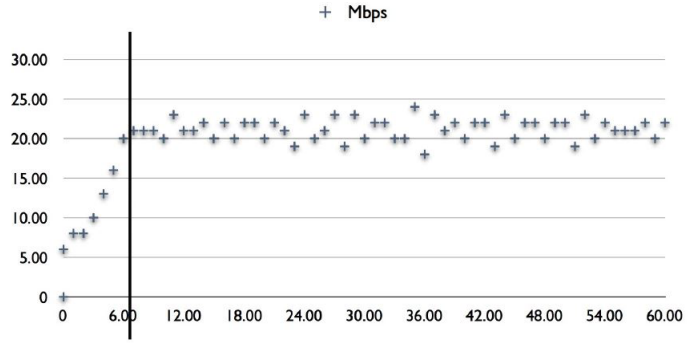

FIGURE 17. The average TCP throughput (X axis: seconds, Y axis: TCP throughput in Mbps)

5.3. **Multiple TCP connections within a DVB link.** 1. One hop of router to router We create multiple TCP data streams within a DVB link between two routers. Figure 18 shows the results:

- Two TCP connections (* marks the total speed)
  - Stream1: 471 Mbytes transferred in 360.3 seconds (throughput is 10.46 Mbps).
  - Stream2: 462 Mbytes transferred in 360.5 seconds (throughput is 10.25 Mbps).
  - Total throughput: 20.71 Mbps.
- Three TCP connections (□ represents the total throughput)
  - Stream1: 322 Mbytes transferred in 360.5 seconds (throughput is 7.38 Mbps).
  - Stream2: 308 Mbytes transferred in 360.5 seconds (throughput is 6.83 Mbps).
  - Stream3: 319 Mbytes transferred in 360.5 seconds (throughput is 7.08 Mbps).
  - Total throughput: 21.29 Mbps.
- Four TCP connections (□ represents the total throughput)
  - Stream1: 241 Mbytes transferred in 360.5 seconds (throughput is 5.35 Mbps).
  - Stream2: 232 Mbytes transferred in 360.5 seconds (throughput is 5.15 Mbps).
  - Stream3: 240 Mbytes transferred in 360.6 seconds (throughput is 5.32 Mbps).
  - Stream4: 240 Mbytes transferred in 361.0 seconds (throughput is 5.32 Mbps).
  - Total throughput: 21.14 Mbps.

2. Three hops of PC to PC through routers

In the test, there are two PCs connected to each DVB router. End-to-end connection is of 3-hop length. We want to demonstrate and measure the overheads in kernel space of

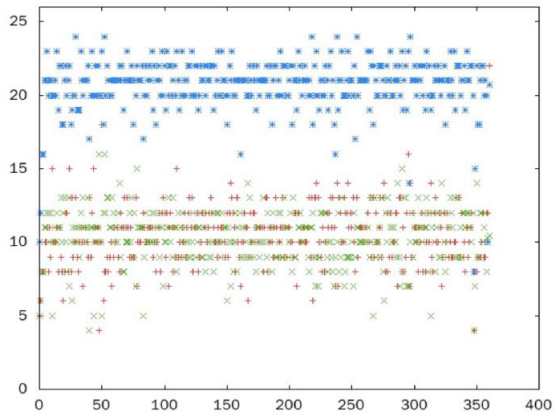

FIGURE 18. Throughput of two TCP connections (X axis: seconds, Y axis: Mbps)

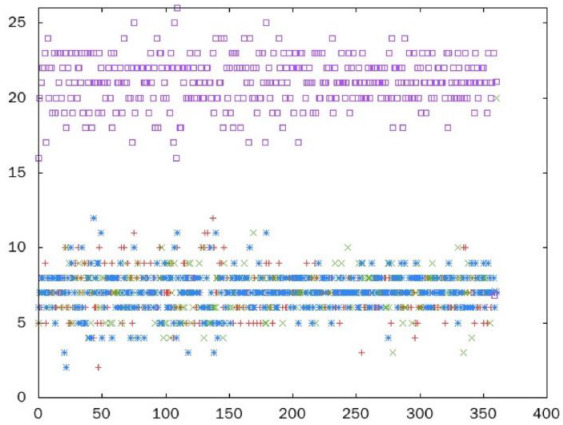

FIGURE 19. Throughput of three TCP connections (X axis: seconds, Y axis: Mbps)

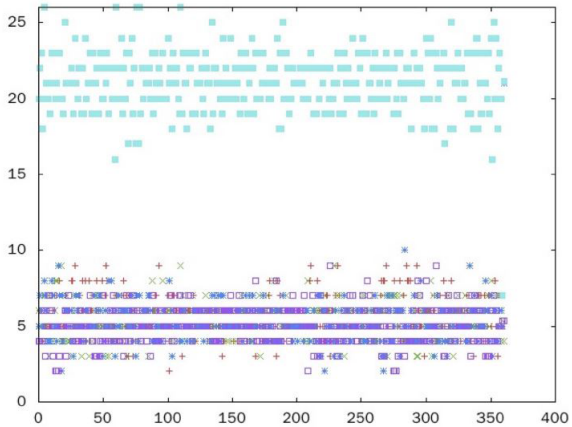

FIGURE 20. Throughput of four TCP connections (X axis: seconds, Y axis: Mbps)

store-and-forward within each router between Ethernet (eth0) and DVB (dvb0). From the experiment results shown below, the store-and-forward cost is not too high and acceptable.

- Two TCP connections (* represents the total throughput)
  - Stream1: 411 Mbytes transferred in 360.5 seconds (throughput is 9.12 Mbps).

– Stream2: 447 Mbytes transferred in 360.6 seconds (throughput is 9.92 Mbps).
– Total throughput: 19.04 Mbps.

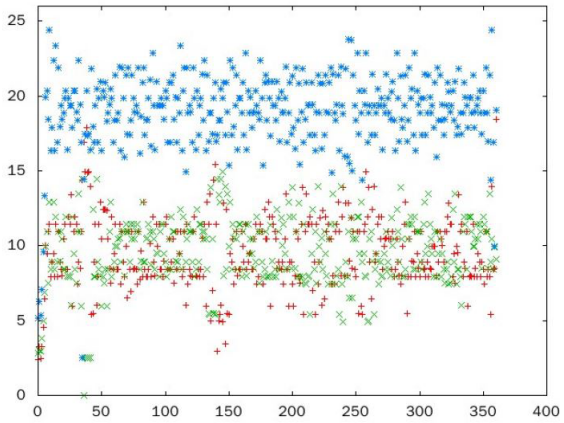

FIGURE 21. Throughput of two TCP connections (X axis: seconds, Y axis: Mbps)

- Three TCP connections (□ represents the total throughput)
  – Stream1: 297 Mbytes transferred in 360.6 seconds (throughput is 6.59 Mbps).
  – Stream2: 304 Mbytes transferred in 360.8 seconds (throughput is 6.74 Mbps).
  – Stream3: 264 Mbytes transferred in 360.9 seconds (throughput is 5.85 Mbps).
  – Total throughput: 19.18 Mbps.

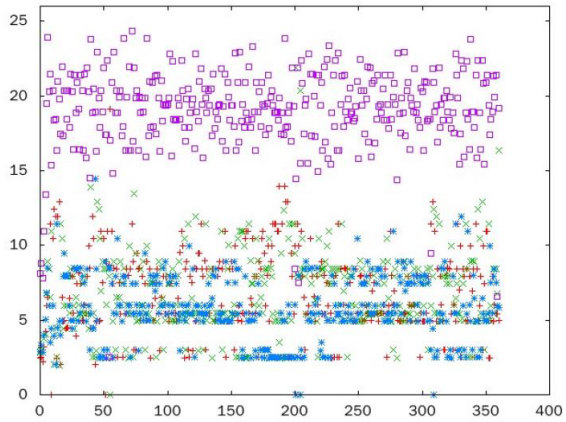

FIGURE 22. Throughput of three TCP connections (X axis: seconds, Y axis: Mbps).

- Four TCP connections (□ represents the total throughput)
  – Stream1: 221 Mbytes transferred in 360.4 seconds (throughput is 4.91 Mbps).
  – Stream2: 220 Mbytes transferred in 361.0 seconds (throughput is 4.88 Mbps).
  – Stream3: 215 Mbytes transferred in 361.1 seconds (throughput is 4.76 Mbps).
  – Stream4: 217 Mbytes transferred in 361.7 seconds (throughput is 4.80 Mbps).
  – Total throughput: 19.35 Mbps.

6. **Conclusions.** The paper is to present the implementation of IP packet transportation over DVB wireless network link. We design DVB network interface as the same as Ethernet virtual network device. But we replace the Ethernet driver to DVB driver and add more kernel space system calls for DVB network device. Thus, all Linux system calls and

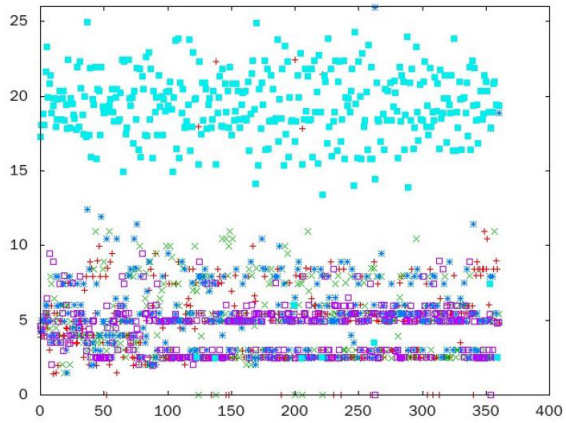

FIGURE 23. Throughput of four TCP connections (X axis: seconds, Y axis: Mbps)

socket API are all workable to DVB network device. There is no need for us to implement additional API for DVB network device in user space. All existing network utilities can run over DVB network device without any porting issues. Thus, data forwarding between Ethernet and DVB within DVB router can be simply solved by conventional IP forwarding utility. Accordingly, DVB backbone can be constructed by DVB routers with a special required topology. The Ethernet network device in the DVB router is the interface of IP packets to deliver through DVB backbone to the other IP networks.

From the experiment results, we can find our implementation overheads is low enough. Some performance improvement scheme are specially designed for Internet bursty traffic characteristics. The TCP throughput can almost reach DVB link throughput. However, it takes 7 8 seconds warn-up period to reach the maximal throughput due to the TCP flow control mechanism. We have to modify the operation mechanism of the ring buffer on DVB driver to further make the TCP flow control mechanism more effective. On the other hand, the TCP buffer need to be enlarged to reduce the packet loss by buffer overrun when passing the packets to upper layer, because the speed of Ethernet is much faster than the DVB network.

In our future research, we need to design some algorithms or mechanisms to manage TCP buffer and DVB ring buffer located in network device driver to make whole system stable more quickly. DVB has been developed for 20 years and its reliability and robustness are verified by digital TV applications. We believe that DVB network is a good alternative wireless Internet backbone solution for metropolitan area networks. Its multimedia delivery and broadcasting capability will enhance Internet services.

REFERENCES

[1] D. Negru, A. Mehaoua, Y. Hadjadj-Aoul, and C. Berthelot, Dynamic bandwidth allocation for effcient support of concurrent digital TV and IP multicast services in DVB-T netwoks, *Journal of Elsevier's Computer Communications*, vol. 29, no. 6, pp. 741-756, 2006.

[2] DVB-T transport stream, available at: http://csie.ntut.edu.tw/ labaspl/edu/MPEG2 TS(NCTU2)A.pdf.

[3] G. Gardikis, A. Kourtis, and P. Constantinou, Dynamic bandwidth allocation in dvb-t networks providing ip services,*Journal of IEEE Trans. on Broadcasting*, vol. 49, no.3, pp. 314-318, 2003.

[4] B. Collini-Nocker, and G. Fairhurst, ULE versus mpe as ab ip over dvb Encapsulation, *Proc. HET-NETs*, Ilkley, UK, 2004.

[5] DSM-CC multi-protocol Encapsulation, available at: http://www.interactivetvweb.org/tutorials/ dtv_intro/dsmcc/multi_protocol_encapsulation.

[6] *Taiwan digital television committee*, available at: http://www.dtvc.org.tw/knowledge 01.html.

[7] *DVB project*, available at: http://www.dvb.org/.
[8] ETSI EN 301 192 V1.4.2, *Digital video broadcasting (DVB): DVB specification for data broadcasting*, European Standard (Telecommunications series), 2008.
[9] H. Burklin, R. Schafer, and D. Westerkamp, DVB: from broadcasting to ip delivery, *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 65-67, 2007.
[10] U.H. Reimers, DVB-The Family of International Standards for Digital Video Broadcasting, *Proc. of the IEEE*, vol. 94, no. 1, pp. 173V182, 2006.
[11] A. J. Stienstra, Technologies for DVB Services on the Internet, *Proc. of the IEEE*, vol. 94, no. 1, pp.228V236, 2006.
[12] W. Richard Stevens, *TCP/IP Illustrated, Vol.1: The Protocols*, Addison-Wesley, USA, 1994.
[13] T. Nagano and, A. Ito, Packet loss concealment of voice-over ip packet using redundant parameter transmission under severe loss conditions, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 5, no. 2, pp. 285V294, 2014.
[14] F. C. Chang, H. C. Huang, and H. M. Hang, Layered access control schemes on watermarked scalable media, *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 49, no. 3, pp. 443V455, 2007.
[15] *MPEG transport stream*, available at: http://www.iptvdictionary.com/iptv_dictionary_MPEG_Transport_Stream_ TS_definition.html.
[16] G. Fairhurst and B. Collini-Nocker,*RCF4326-Unidirectional Lightweight Encapsulation (ULE) for Transmission of IP Datagrams over an MPEG-2 Transport Stream(TS)*, 2005.
[17] B. Collini-Nocker, *RCF 4259-A Framework for transmission of IP datagrams over MPEG-2 networks*, IETF Work in Progress, 2005.
[18] S. Gujjunoori, and B. B. Amberker, Reversible data embedding for mpeg-4 video using middle frequency dct coefficients, *Journal of Information Hiding and Multimedia Signal Processing*, vol. 5, no. 3, pp. 391V402, 2014.
[19] *Multiprotocol Encapsulation*, available at: http://en.wikipedia.org/wiki/Multiprotocol_Encapsulation.
[20] S. Epstein, Using multi-protocol encapsulation technology to develop optimum data broadcast systems, *IEE Colloquium on EUTELSAT - New Products and Services*, pp. 1V7, 2000.
[21] A. K. Pugalia, *Device Drivers, Part 4: Linux Character Drivers*, available at: http://www.linuxforu.com/2011/02/linux-character-drivers/, 2011.
[22] J. Corbet, A. Rubini, and G. Kroah-Hartman, *Linux Device Drivers*, 3rd Edition, O'Reilly, USA, 2005.
[23] M. Hunold, *Linux DVB API Version 4*, available at: http://www.linuxtv.org, 2005,.