Effective Software-Based Solution for Processing Mass Downstream Data in Interactive Push VOD System

Ni Hong, Wu Guobin, Wu Gang, and Pan Liang

***Abstract*—**Interactive push VOD system is a new kind of system that incorporates push technology and interactive technique. It can push movies to users at high speeds at off-peak hours for optimal network usage so as to save bandwidth. This paper presents effective software-based solution for processing mass downstream data at terminals of interactive push VOD system, where the service can download movie according to a viewer's selection. The downstream data is divided into two catalogs: (1) the carousel data delivered according to DSM-CC protocol; (2) IP data delivered according to Euro-DOCSIS protocol. In order to accelerate download speed and reduce data loss rate at terminals, this software strategy introduces caching, multi-thread and resuming mechanisms. The experiments demonstrate advantages of the software-based solution.

***Keywords*—**DSM-CC, data carousel, Euro-DOCSIS, push VOD.

I. INTRODUCTION

THE application of PVR (Personal video recorder) technology provides the opportunity to deliver content using a push-VOD model where a partition of the hard disk is reserved for caching content that is then made available on-demand. Push VOD is a bandwidth-saving technology that can push movies at high speeds, and deliver them at off-peak hours for optimal network usage. At the same time, the interactive television (ITV) technology has provided consumers with interactive digital services. Users can preview free promotional clip in the conventional interactive VOD manner to choose their favorite content to be downloaded.

In the interactive push VOD system, users select the films they want by means of a browser, and then send their requests to a head end server through upstream channels, in which the amount of data is very small. In response to terminals' requests, the head end packages the required clips in accordance with Euro-DOCSIS [1] standard. Upon receiving the interactive data, terminals analyze it by IP protocol stack. Here, the data is referred to as interactive data. In addition, at the head end, movie data are encoded and packed into MPEG-2 transport streams (TS) [2] according to DSM-CC data carousel specification [3][4] and transmitted to client terminals through HFC network. At terminals the received data are filtered, parsed and re-assembled, and then written into a hard disk as integrated movie data, which is referred to as carousel data. Due to the mismatching speed between software data-processing and hardware data-receiving caused by the resource restraint of embedded system, downstream data is more likely to be lost compared to upstream data. Therefore, the prominent problem to be solved is how to configure the packaging strategy at head end and to select the data-processing method at terminals.

In this paper, effective software-based solution is proposed to support the application of interactive push VOD system. At ISP head end, software package generator packs downstream data according to Euro-DOCSIS protocol, while at BSP head end, movie data is made into packages based on data carousel. The two kinds of packages are transmitted via HFC at different frequencies. For the received data, terminals adopt caching mechanism, multi-thread mechanism and resuming mechanism. Multi-thread includes several threads such as data-receiving thread, writing thread and the like. Meanwhile, the data-receiving thread group and the data-analyzing thread group can be designed correspondingly to the capacity of the demultiplexer (demux) applied to the terminal. Our experiments demonstrate that the proposed software-based solution can effectively process mass downstream data within a shorter time period and reduce packet-losing rate at terminals when compared to others. This strategy can help improve the performance of terminals which receive mass downstream data in similar service.

The rest of the paper is organized as follows. Section 2 provides structure of mass downstream data. In section 3, effective software-based solution for processing mass downstream data in terminal is presented in detail. Experimental environment and its results are described in section 4. Finally, in section 5 the conclusion is given.

Manuscript received September 26, 2006. The research is supported by China Next Generation Internet Foundations (CNGI) (No.CNGI-04-15-2A)

Ni Hong was with Institute of Acoustics, Chinese Academy of Sciences, Beijing, CO 100080, China (phone: 86-010-62565615; e-mail: nih@dsp.ac.cn).
Wu Guobin is with Department of Automation, University of Science and Technology of China, Hefei, CO 230027, China (e-mail: wuguobin@mail.ustc.edu.cn).
Wu Gang is with Department of Automation, University of Science and Technology of China, Hefei, CO 230027, China (e-mail: wug@ustc.edu.cn).
Pan Liang was with Institute of Acoustics, Chinese Academy of Sciences, Beijing, CO 100080, China (e-mail: panl@dsp.ac.cn).

II. Structure of Mass Downstream Data

Interactive downstream data, that is, interactive data in the downstream direction, is packaged in accordance with Euro-DOCSIS protocol. A MAC frame is the basic unit of transfer between the cable modem (CM) and the CMTS. Preceding each MAC frame is an MPEG-2 transmission convergence header in the downstream direction**,** as shown in Fig. 1.

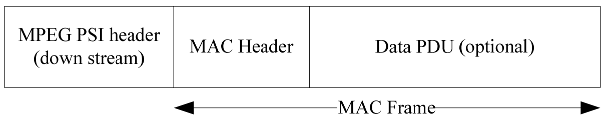

Fig. 1 Downstream MAC Frame format

Carousel data is organized for transmission according to DSM-CC data carousel protocol. The DSM-CC data carousel defines three important sections: Download Server Initiate (DSI), Download Info Indication (DII), and Download Data Block (DDB).

According to the DSM-CC data carousel standard, Block Number defines the number of blocks included in a module. As a result of the 16-bit Block Number, the maximum number of blocks contained in each module is 65536. When transmitted as DDB in a DSM-CC section, a block is 4 Kbytes. Therefore, a module has a fixed maximum capacity of 256 Mbytes (64K*4K). However, such module usually can't meet the requirements of bulk-data push service. According to the standard, a DSM-CC section contains 4096 bytes at most. In a DSM-CC section, DSM-CC header occupies 8 bytes, and DSM-CC CRC (Cyclic Redundancy Check) takes up 4 bytes. Packed as a DSM-CC payload， DII header needs 34 bytes. As a module is 13 bytes, a section can hold 311 modules at most due to the equation: (4096-8-4-34)/13=311. Thus, we use one or several groups to transmit bulk-data service. In practice, transmission structure should be chosen based on the size of multimedia data. Thus, in our implementation of bulk data such as a movie data which contains 2 Gbytes, we use one group structure which contains 8 modules to transmit data. It only needs DII and DDB section. The whole structure of the data carousel is shown in Fig. 2.

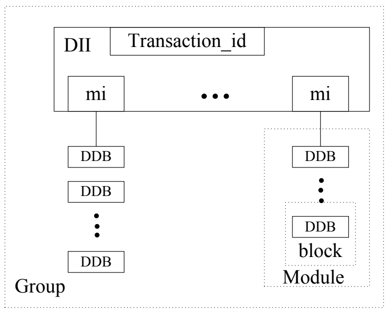

Fig. 2 Data carousel structure

III. Software-Based Solution for Processing Mass Downstream Data in Terminal

The receiving unit of the terminal is divided into two parts: CM unit and demux unit. The framework is shown in Fig. 3.

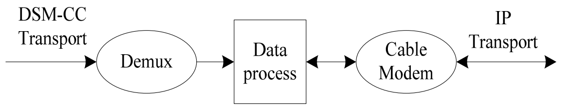

Fig. 3 Receiving unit of the terminal

The CM unit takes the responsibility of receiving interactive downstream data. The downstream function of the MAC include receiving MPEG-2 frames from the downstream receiver, extracting the Euro-DOCSIS MAC frames from the MPEG-2 frames, processing the MAC headers, filtering messages and data. The filtered downstream data packets will be transferred to memory space and then be analysis by IP stack.

The interactive downstream data is usually sent out by the head end in response to the terminal's request, thereby it is burst and random. Then, the terminal analyzes the data received by the CM unit.

The terminal demux unit receives carousel data as follows. Firstly, the terminal receives DSM-CC sections from MPEG-2 transport stream according to user demand. Then it analyses the received sections and re-assembles them into an integrated part. Lastly, by computing it determines the location on hard disk where data will be written. For users, they can conveniently manage the downloaded content through media management module.

The mechanism of data carousel is to divide data into sections and to broadcast them periodically. In fact, it is impossible for a terminal to receive all the data in the carousel during only one cycle. Once a section is missed, the terminal has to wait for another cycle, which is surely time-consuming. Therefore, it is important to reduce the down latency to satisfy viewer's requirements.

In order to give a full play to the whole resource of the terminal, we introduce effective software-based solution as shown in Fig. 4.

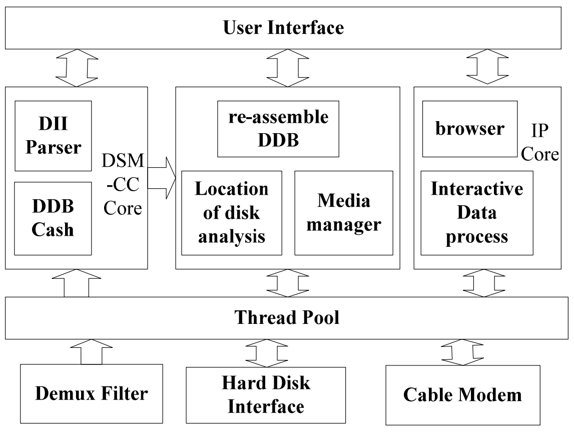

Fig. 4 Overview of effective software-based solution

A. The Cause of Performance Deterioration

The CM unit has a powerful function of receiving the interactive downstream data, which is burst and random as above-mentioned, at a high speed, for example, the highest of 38Mbps. However, having undergone the analysis by MAC layer of CM, these data is copied to the memory of the terminal for the analysis based on higher-level protocol. Thus, in the case that these data is not processed in time, they will take up a large portion of memory, which leads to the severe deterioration of the terminal's performance.

The data transport processor is an MPEG-2 transport stream message/PES parser and demux. [2] The module functions as receiving sections from a transport stream. In digital broadcasting, DDB sections make up of main data, while DII sections have a small data volume. When DDB filter is set in demux channels, the data transport processor can receive only corresponding DDB sections to hardware buffer instead of all the sections. If the buffer is full and DDB sections inside are not taken away in time, newly-coming DDB sections can't enter the buffer and are automatically abandoned. In general, data transport processors have a very rapid processing speed. For example, in our platform, the maximum data-receiving rate can reach 162 Mbps, and the size of hardware buffer can be set from 1 Kbytes to 512 Kbytes. On the contrary, the ability of CPU on embedded platform is limited and can't match the data transport processor. For example, the CPU on our platform can only achieve 133 MHz. In fact, analyzing sections by the software will take much more time than retrieving them by the hardware. So if we receive maximum sections in the buffer and then analyses them, there will be a high possibility to miss the next sections during this cycle. Even though we set hardware buffer maximum capacity as 512 Kbytes, there still exists section loss.

In conclusion, there is a need for an effective solution to prevent the deterioration from occurring, therefore to ensure the satisfactory experience for a user. One of effective methods is to make some change to software architecture.

B. Caching and Multi-Thread Mechanism

In order to avoid hardware buffer overflowed and give demux a full play, a circular cache of 1 Mbytes is built in memory to store DDB sections. After entering the hardware buffer, DDB sections are immediately copied to the circular cache in order that new DDB sections can enter hardware buffer. At the same time, a new thread is started from thread pool, which answers for analyzing DDB sections. The use of two-thread mechanism also avoids data loss due to the hardware buffer overflowing.

Here, thread group will be explained at first. As well known, the hardware capacity of demux differs on different platforms. In our research, there are two kinds of demux in two platforms: one can use a plurality of channels to receive a PID data, while the other can use only one channel for a PID data, that is, the other channels reject the same PID data. It is obvious that the capacity of the former is much higher than that of the latter. Accordingly, we can take advantage of the former to utilize a number of channels for data receiving, with each channel for a different section of DDB data. The more detailed description is as follows:

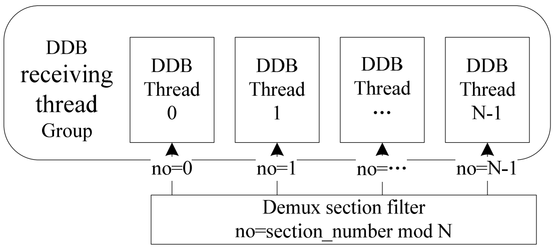

Fig. 5 Scheme of the DDB receiving thread group

In the DSM-CC protocol, a section number is filed in the DDB section header. DDB sections are sorted according to this section number and divided into corresponding threads to receive and analyze. The sorting method is as follows: Suppose N is the number of DDB receiving threads; it is equal to an integer power of two. The hardware section filter employs the section number to allocate the sections to various threads. The sections whose section numbers when divided by N yield a remainder of 0 are processed by Thread 0, and the sections whose section numbers when divided by N yield a remainder of 1 are processed by Thread 1. Further, the sections whose section numbers when divided by N yield a remainder of N – 1 are processed by Thread N – 1. Fig. 5 shows the DDB receiving thread group. The received data is put into the buffers corresponding to the assigned N buffers. In the same manner, the thread group for DDB data analysis creates as many threads as that of the thread group of DDB data reception, and carries out the analysis in correspondence with the reception thread group. Here, it is better not to make N too great. Since N indicates the number of the created threads, if too many threads are created, it will be disadvantageous that the system performance is degraded. In addition, the N buffers lead to the increase in memory consumption.

As to the demux with poor capacity, it is sufficient to assign 1 to the number N, and there is no need for any modification in the package generation at the head end.

These two threads group operate on the respectively corresponding circular cache. Receiving thread group is write-only and writes data to cache when there is free space; analyzing thread group is read-only and read data from cache while clearing up corresponding space. In the following experiment, we can find that using the caching and multi-thread mechanism enables parallel processing of data reception and local analysis so that to enhance data-processing efficiency, and to reduce the down latency and section loss rate.

For the multi-thread mechanism, there is a thread for writing hard disk besides receiving thread and analyzing thread. When analysis of DDB sections is completed, the payload of DDB sections needs to be written to hard disk. Because the writing operation takes some time, it is proper to introduce a new thread responsible for this task. Moreover, to guarantee the immediate processing for the interactive downstream data, a specific thread is developed, which has the function of monitoring the burst and random interactive downstream data, that is, entering the sleep mode when there is no data, and responding to the data as soon as it comes. The opening and closing of these threads are managed by an administer thread.

C. Re-Assembling Method and Resuming Mechanism for Carousel Data

It is impossible for a terminal to take a single cycle to receive all the DDB sections in the carousel. Therefore, it requires data re-assembling in the terminal. By calculating the total number of DDB sections from Module Size, Block Size and Module Number in the DII section, an index file of receiving information is generated locally, of which every element is initialized as zero. A DDB section is a block. For example, if the number of DDB block is M, an M-byte index file is built and initialized as 0x00. When a DDB section is received as a block, this block's index can be computed this block index from its Module Id and Block Number. For instance, if the block's index is calculated as N, the Nth byte in the index file will be read. After that, determine whether the Nth index is 0x00. If yes, payload of the block will be written to the location of N* Block Size in hard disk, and the Nth index will be set as 0xFF; otherwise, it indicates this block has been received and should be abandoned.

As a result, the re-assembling process is independent of user operation and can be resumed. Whenever a user starts his terminal, the terminal can continue to download and store data left from previous process.

IV. EXPERIMENTS AND RESULTS

A. Experiment System

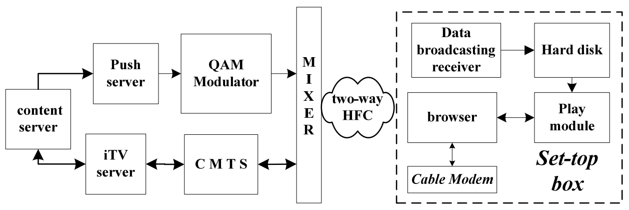

Fig. 6 Interactive push VOD system environment

Fig. 6 shows an overview of the push service system environment.

At front end we performed our tests with a MPEG-2 transport stream. This transport stream transmits multimedia data through data broadcasting DSM-CC Data Carousel. It is multiplex of 188 bytes packets consists of 75% DDB message, 25% DII message.

The application is tested on a MHP[5] compliant STB Bi-7111 running at 133 MHz, 64 Mbytes DDR memory, 16 Mbytes flash memory, 10 Gbytes hard disc, MPEG-2 hardware decoder, two tuners, 32 demux channels, Euro-DOCSIS 1.1 protocol stack, Linux OS 2.4.18, etc.

B. Experiments

It is difficult to carry out the experiment on the interactive downstream data due to its burst and random characteristics. Further, the amount of the interactive downstream data is much smaller than that of the carousel data. So, to evaluate the system performance of interactive push VOD system, the main factor is DDB section lose rate, that is, the ratio between the number of terminal-received sections and that of all the transmitted sections in one cycle. As referred above, because the pushed data is periodically displayed in a predetermined order, the loss of any section means that the terminal has to wait for at least another cycle to receive it, while the rest data within the waiting cycle don't need to be received again and become useless. As a result of the waiting, at least a cycle and bandwidth are wasted.

In the early implementation of interactive push VOD system, we use simple single-thread software-based solution without cache to process data in the terminal. Our experiment demonstrates that this method uses many cycles and is inefficient. After introducing effective software-based solution with caching mechanism, multi-thread mechanism and resuming mechanism, the experiment reveals satisfactory results of mass downstream data push service.

In our first experiment, we transmit 44-Mbytes file as one module. Then, we test terminal performance of simple software-based solution and effective software-based solution at different speeds of transport streams: 15.48 Mbps, 22.12 Mbps, 27.64 Mbps and 33.18 Mbps, respectively. Here, a single DDB thread and a two DDB thread schemes are employed correspondingly to demuxes with different capacities. DDB section lose rate and the number of cycles for receiving integrated file are shown in Table I.

TABLE I
EXPERIMENTAL RESULTS FOR A 44MBYTES FILE

Mbps	Simple		Effective N=1		Effective N=2	
	Lose rate%	Cycles	Lose rate%	Cycles	Lose rate%	Cycles
15.48	12.9	1.4	0	1	0	1
21.12	31.6	2.2	0.4	1.0	0.4	1.0
27.64	52.4	2.9	4.3	1.2	3.9	1.2
33.18	67.1	3.4	13.3	1.3	13.1	1.2

TABLE II
EXPERIMENTAL RESULTS FOR BULK FILES: LOSS RATE

Mbytes	Simple	Effective N=1	Effective N=2	Effective N=4
400	33	3.0	1.0	0.0
2000	70	16	2.8	0.0

(Unit: lose rate %)

TABLE III
EXPERIMENTAL RESULTS FOR BULK FILES: CAROUSEL CYCLE

Mbytes	Simple	Effective N=1	Effective N=2	Effective N=4
400	3.20	1.48	1.04	1.01
2000	10.6	1.99	1.28	1.00

(Unit: 1 carousel cycle)

As shown in Table I, it is obvious that the processing performance is remarkably enhanced with effective software-based solution. The data analysis indicates that speed of transport stream also influence system performance. As is seen, given a small amount of data, the improvement in capacity is not very significant for the demux which supports a same PID data-receiving with several channels. The comparison will be shown from the following experiment on a large amount of data. Therefore, it is determined to adopt 15.48 Mbps as transport stream speed for mass data in order to achieve minimum section lose rate. Table II and Table III shows our experimental results on a 400-Mbytes file (2 modules) and a 2-Gbytes file (8 modules).

As seen above, effective software-based solution is more advantageous than simple software-based solution in terms of system performance. Also, for the platform with high-capacity demux, it is more advantageous to adopt the strategy to use two thread groups, other than a single thread, for DDB data receiving and analyzing. It takes one loop to download a 2-Gbytes file, which can meet viewers' requirements for transmission speed.

V. CONCLUSION

It is of great importance for interactive push VOD system to improve the performance of processing downstream data. This paper proposes effective software-based solution for an interactive push service. This effective software-based solution gives a great promotion to terminals' processing capability by adding caching, multi-thread and resuming mechanisms. In particular, one thread group DDB data-receiving and another thread group DDB data-analyzing strategy are proposed for demuxes with different capacity, in which the single DDB data-receiving thread and the single DDB data-analyzing thread method is supported without any change in the head end. From the experimental results, it proves that the effective software-based solution satisfies viewer's requirements and will be helpful for accelerating interactive push service. Furthermore, the solution has been implemented in the software architecture of the terminal in a trail system of application of interactive services in two-way CATV networks.

ACKNOWLEDGMENT

The Author gratefully acknowledges the support of K. C. Wong Education Foundation, Hong Kong.

REFERENCES

[1] CM-SP-RFIv1.1, *Data-Over-Cable Service Interface Specifications DOCSIS 1.1,* 2005.
[2] ISO/IEC 13818-1, *Information technology -- Generic coding of moving pictures and associated audio information – part 1: Systems*, 1997.
[3] ISO/IEC 13818-6, *Information technology -- Generic coding of moving pictures and associated audio information – part 6: Extension for digital storage media command and control (DSM-CC)*, 1997.
[4] ETSI EN 301 192 V1.3.1 Digital Video Broadcasting (DVB), *DVB specification for data broadcasting,* 2003.
[5] ETSI TS 102 812 V1.2.1 Digital Video Broadcasting (DVB), *Multimedia Home Platform (MHP) Specification 1.1.1,* 2003.