7M−9

# PC BASED VIDEO SERVER SYSTEM IN ATM NETWORK - VIDEO SERVER -

DANG VAN TRAN,    KAWAI HIDEO
INFORMATION TECHNOLOGY R&D CENTER, MITSUBISHI ELECTRIC CORP.

## 1. Introduction

In realizing the desired video server system as described in [1], we have configured our video servers using only off-the-shelves, inexpensive personal computers (PC) running Windows NT OS. Compliant with the ATM Forum's MPEG-over-ATM standard, these PC-based video servers deliver high-quality MPEG2 encoded data to our simple set-top-boxes (STB) via the ATM network.

## 2. Configuration

Video Server's primary task is to handle STBs' requests for video segments. It does this by reading video segments from attached databased hard-disks and sending/pumping those read video segments to the appropriate STBs via the ATM network.

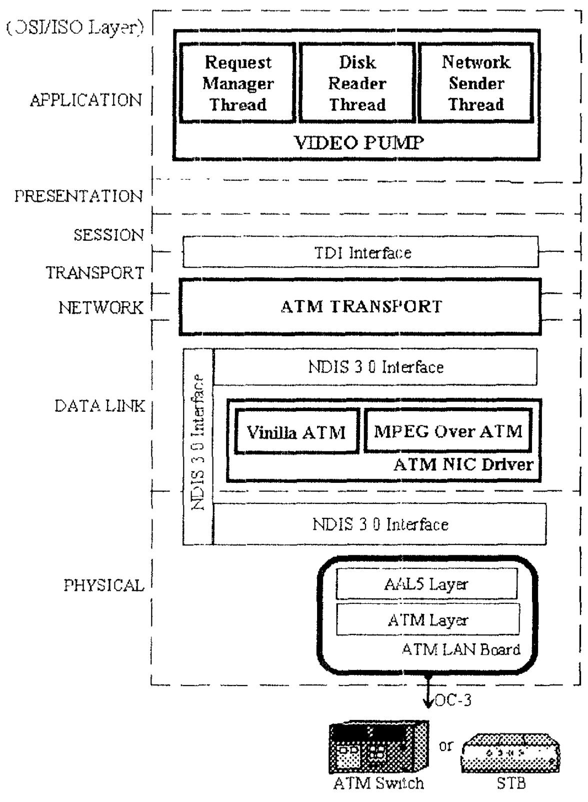

*Figure 1. Overall implementation protocol stack*

Figure 1 above illustrates our video server's implementing components and their relative locations within the OSI Protocol Stack. Note that the implementing components are dark boxed: Video Pump, ATM Transport, and ATM-NIC Device Driver.

## 3. Implementation Issues

### 3.1 MPEG-over-ATM

Reflecting the support of MPEG-Over-ATM data transmission specification (each transceiving packet size are in multiples of 188 bytes, with default size of 376 bytes) efficiently, the parsing and sending of the data are done in all three components: Video Pump, ATM Transport, and ATM-NIC Device Driver.

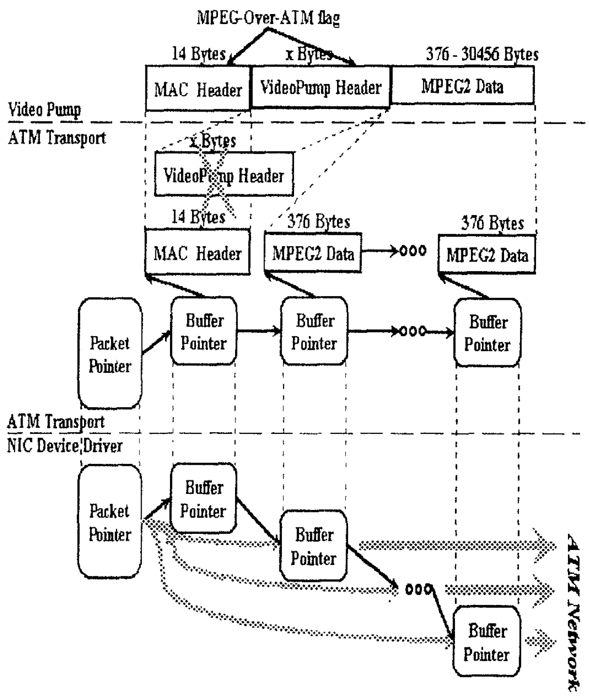

*Figure 2: MPEG-over-ATM mode*

The Video Pump component creates appropriate headers for the MPEG2 data payload. The ATM Transport component then creates the necessary packet pointer and buffer poiters according the

the Windows NT specification. Last, the ATM-NIC Device Driver iterates through these prepared buffer pointers and sends each as an individual data packet. See Figure 2 for pictorial explaination of the implementation.

### 3.2 Bandwidth Shaping

As described in [1], the precise controlling of the timed event of data transmission on a non-realtime OS is difficult and the variant of transmission speed is inevitable. One of our approaches to solving this problem is implementing a transfer-rate monitoring system with a dynamic manual or automatic "system tuning" feature onto each client.

We keep track of two transfer rate: "last transfer rate set" and "total average transfer rate". The "last transfer rate set" is actually an average calculation of the last 'x'[1] number of transmission rates. We utilize this value for the automatic "system tuning" such as stabilizing the transmission rate. The "total average transfer rate" can be useful for several applications such as system bench-marking.

| Transfer Rate Monitoring | Average Bit Rate (Mbps) | Negative Variation (Mbits) | Positive Variation (Mbits) |
|---|---|---|---|
| OFF | 2.9-3.0 | -.1 ~ -.1.5 | .2 ~ .6 |
| OFF | 5.8-6.1 | -.4 ~ 2.5 | .2 ~ .8 |
| ON | 2.9-3.0 | -.05 ~ .6 | .1 ~ .3 |
| ON | 5.8-6.1 | -.2 ~ -.8 | .2 ~ .5 |

*Table 1[2]: Transfer Rate Monitoring benchmarking*

Refer to Table 1[2] above for some basic bandwidth variation bench-marking. We used two types of MPEG2 video streams for our testing: 3Mbps and 6Mbps. When the transfer rate monitoring mode was not used (OFF), slight "disturbance" in the video transmission process can dramatically variate the tranmission rate, causing underflow (jitter) and overflow (mosiac breakout) during playback. With the transfer rate monitoring mode enabled (ON), dramatic improvement took affect to a more smoother playback.

### 3.3 Transmission Scheduling

When the video server has to host more than one client, request and transmission scheduling has to be handled efficiently, and also fairly. One key decision in designing an application is the number of threads to use. Adding more and more threads can degrade overall performance.

Our current solution is to create a minimum and constant number of threads needed, regardless of the number of hosting clients. Thus, three threads are utilized: the RequestManager thread handles calls/requests from/to all the clients, the DiskRead thread handles all the reading from disks and writing to buffers for all the clients, and the NetworkSend handles all the reading from buffer and sending to network for all the clients. These threads will iterate through all the hosting clients and round-robin serve them to insure the desired transmission "fairness."

## 4. Conclusions and Future Work

Eventhough this project is still in its infancy stage, it demonstrates that PC-based video servers are technically possible, if not also economically suitable to handle certain Video-On-Demand networks.

Much optimization solutions/ideas and feature enhancement are currently being developed and considered. Belows are some of the these planning:

- relocates video pump application to kernel environment
- handles larger MPEG2-TS packet size
- enlarges clients' buffer size
- handles client-initiative data requests
- disk stripping

## 5. Reference

[1] [7M-08], PC-Based Video Server in ATM Network - Overview -, Kawai Hideo and Dang Van Tran, Mitsubishi Electric Corp.

[1] the value 'x' is varied among different system to create a more accurate transfer rate calculation and a more stable transmission (less bandwidth variation), our experimenting value is 5

[2] data gathered from visual estimation on an ATM analyzer