Video Transmission over ATM and IP

Dorgham Sisalem
GMD-Fokus Berlin
(dor@fokus.gmd.de)

November 13, 1995

1 Introduction

In this short study we investigated the performance of a video sender when transmitting over different links using different protocol stacks. As testing cases we considered the protocol stacks UDP/IP/Ethernet, UDP/IP/AAL5 and AAL5. As a video source we used a simple video tool that we implemented during this study. A short overview of this tool is given in Sec. 2. In Sec. 3 the testing configuration that was used for the performance comparison is presented. The actual results are finally presented in Sec. 4.

2 The Network Video Terminal (NeViT)

While there are many powerful and freely available video tools [1] adjusting these tools to suite our own requirements is usually difficult and requires too much time. To avoid this, we implemented our own video tool called the Network Video Terminal (NeViT). At the current stage of development, the tool consists of a sender that transmits video data in RTP packets according to the Internet-Draft of the IETF AVT Working Group [2, 3]. The video packets can be either sent directly over an AAL5/ATM connection or a UDP/IP socket. Multicast IP is supported as well. As the receiver entity of the tool we can either use another RTP based video tool, such as vic [1], or our own simple receiver. Currently, our receiver is at a very early stage of development. Data can be received either through a UDP/IP socket or directly from the ATM Fore interface. Here, the arriving frames are reassembled, decompressed and simply displayed on the X-display. While the decompression is done is software, compressing and grabbing the video data is done using a Sun Video board. Currently, JPEG, CellB and YUV are supported for encoding the data. NeViT is being developed on Sun workstations with Solaris 2.4 as the operating system.

3 Test Setup

As a test setup we used two Sun-Sparc20 workstations with Super Sparc processors and Fore-SBA200 ATM cards. The two workstations called tao and rockmaster are connected to each other either through our local Ethernet or through a back-to-back ATM connection. That is, the output port of the ATM card of one machine is connected to the input port of the other one and vice versa. Further, we established a permanent virtual connection (PVC) between the two stations. By using a PVC and a direct connection we avoided any effects that switching and signaling might have on our results.

During our tests we used tao as the sender and rockmaster as the receiver. To fully dedicate the available processing power to the transmission actions we turned off the local preview window option that our video tool provides. With this option turned on, around half of the available processing power was taken up by the decompression and displaying the video data on the X-display. In this case a much lower transmission performance was noticed.

For grabbing the video data we used a Sun NTSC video camera with a maximum grabbing rate of 30 frames/s. The data was then compressed using a Sun Video board.

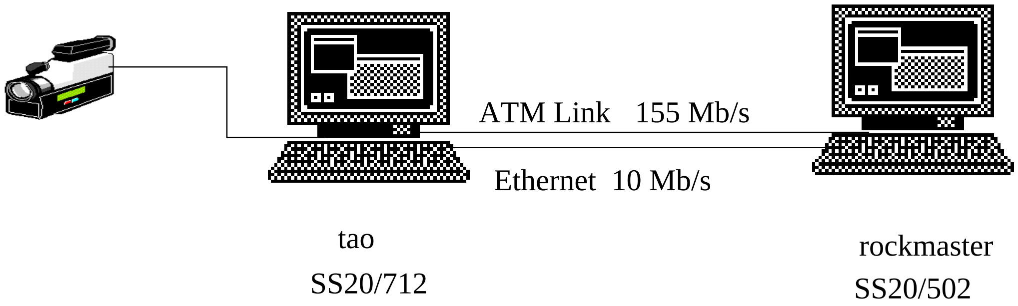

Figure 1: Test setup

4 Performance of Video Transmission over ATM and IP

As already mentioned, we investigated the transmission performance of a video sender when transmitting over UDP/IP/Ethernet, UDP/IP/AAL5 and directly over AAL5, see Fig. 2. For the measurements done here, we used a simplified version of NeViT without RTP.

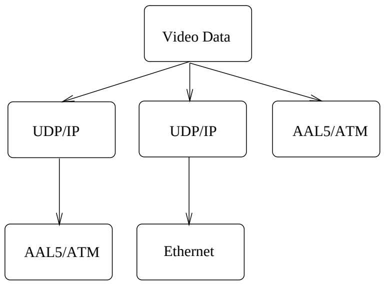

Figure 2: Data flow diagram

For our tests we measured the time required to send 2000 video frames from tao to rockmaster using the different protocol stacks. The time was measured at the sender site using the local clock as the difference between the time after sending the first packet and the time measured after sending the last packet. By not including the time required for sending the first packet, we avoided including the time required for initializing the video board which might not be the same for the different measurements. These measurements were run for different segment sizes, whereas a segment refers here to the maximum size of the data chunks in which the video frames were segmented into.

Each video frame is 320x240 pixels large. With YUV encoding, this results in 153.6 kbytes large packets. JPEG frames had an average size of around 6 kbytes.

The sending data and frame rates were measured at the sender tao using the local clock. Tables 1 and 2 show that with JPEG as the compression algorithm a similar data and frame rates were

Protocol stack	JPEG			YUV		
	1 kbytes	2 kbytes	8 kbytes	1 kbytes	4 kbytes	8 kbytes
UDP/IP/Ethernet	1.3	1.3	1.29	8.6	8.6	8.6
UDP/IP/AAL5	1.3	1.3	1.29	14	28	35
AAL5	1.3	1.3	1.28	29	36	36

Table 1: A comparison of the achievable data rate in Mb/s with UDP/IP/Ethernet, UDP/IP/AAL5 and AAL5 as a function of segment size and compression algorithm

measured for all cases. This was, however, to be expected. With an average frame size of 6 kbytes only 6 data packets were required per frame for the worst case of a segment size of 1 kbytes. This makes around 180 packets per second. It is obvious that the additional processing time required for the processing of the UDP/IP headers compared to the direct usage of AAL5 would be minimal in this case.

This is, however, entirely different when sending the raw YUV data. To keep up a frame rate of 30 frames/s we would require a link rate of around 36 Mb/s. With a maximum link rate of 10 Mb/s it is obvious that using the Ethernet in this case would lead to a severe performance degradation.

For the ATM connection with a link rate of 155 Mb/s the performance differences noticed in tables 1 and 2 for the cases UDP/IP/AAL5 and AAL5 stem mainly from the additional processing overhead caused by using UDP/IP. This is especially obvious for the case of a segment size of only 1 kbytes. In this case the sender would be required to process 4590 UDP/IP headers to keep up an average rate of 30 frames/s. The large processing overhead required in this case leads to a rate reduction of around 50%. Using larger payloads solves this problem. With UDP/IP/AAL5 and a segment size of 1 kbytes we reached a frame rate of only 11 frames/s, while using a segment size of 8 kbytes increased the frame rate to 29 frames/s.

Protocol stack	JPEG			YUV		
	1 kbytes	2 kbytes	8 kbytes	1 kbytes	4 kbytes	8 kbytes
UDP/IP/Ethernet	29	29	29	7	7	7
UDP/IP/AAL5	29	29	29	11	22	28
ALL5	29	29	29	24	29	29

Table 2: A comparison of the achievable frame rate in frames/s with UDP/IP/Ethernet, UDP/IP/AAL5 and AAL5 as a function of segment size and compression algorithm

References

[1] S. McCanne and V. Jacobson, "vic: A flexible framework for packet video," in *Proc. of ACM Multimedia '95*, Nov. 1995.

[2] W. Fenner, L. Berc, R. Frederick, and S. McCanne, "RTP encapsulation of JPEG-compressed video," Internet Draft, Kaman Sciences, Nov. 1994. Work in progress.

[3] M. F. Speer and D. Hoffman, "RTP encapsulation of CellB video encoding," Internet Draft, Sun Microsystems, Nov. 1994. Work in progress.