Experiments with Delivery of HDTV over IP Networks

Colin Perkins Ladan Gharai Tom Lehman Allison Mankin
USC Information Sciences Institute

15 March 2002

Abstract

We present the design, and preliminary performance results, for a system that transports uncompressed HDTV content over IP networks. Our system is motivated by the growth in use of digital video, and the ever increasing capacity of local- and wide-area Internet links. We aim to demonstrate the feasibility of IP as a transport for very high quality video, and to highlight areas where performance bottlenecks exist and further development is needed. To this end, our system is constructed from commodity components, and was tested over existing commercial IP backbone networks. Performance was shown to be good, with the end system being the main limiting factor.

1 Introduction

The conversion of broadcast television from the legacy analog PAL and NTSC standards to digital format has many exciting implications. These include the possible convergence of television distribution and computer network infrastructures, allowing interactive applications, and the increase in quality possible with high definition digital formats.

To date, the different aspects of this convergence have been studied in isolation: there has been much work on the transport of compressed standard definition TV over IP, and much work defining protocols and standards for high definition TV (HDTV), but few have studied the transport of HDTV over IP. In this paper we present our initial experiments with a system to deliver production quality uncompressed HDTV over IP networks.

Why do we chose to deliver uncompressed HDTV? Several reasons, primarily to maintain image quality and reduce latency. This is most useful in a production facility, where image degradation due to repeated compression cycles is undesirable, but may also be appropriate for very high quality telepresence applications. Delivery of compressed HDTV, using existing MPEG-2 over IP standards, may be more appropriate for other applications.

The outline of this paper is as follows: section 2 covers background in HDTV technology, protocols for transport of video over IP networks and network performance. This is followed, in section 3 with a discussion of the options for protocol development, with our design being outlined in section 4. Section 5 provides preliminary performance analysis of our system, demonstrating transmission of HDTV over a wide-area IP network, with section 6 outlining directions for further development. Finally, we summarize related work in section 7, and provide conclusions.

Format	Picture Format	Ratio	Frame Rate
HDTV	1920x1080	16:9	30I, 30P, 24P
HDTV	1280x720	16:9	60P, 30P, 24P
SDTV	704x480	16:9	30I, 60P, 30P, 24P
SDTV	640x480	4:3	30I, 60P, 30P, 24P

Table 1: Picture formats for digital televisions, defined by ATSC standard A/53.

2 Background

The television industry is in the process of a major transformation, from standard analog PAL and NTSC systems to high definition digital formats. These new formats provide significantly higher spatial and temporal resolution, and greater colour depth, than the existing formats. The digital nature of the new formats also allows for greater integration with computer systems and networks, providing a more interactive system.

High definition TV defines a range of picture formats distinguished by frame size and rate, aspect ratio, and scanning technique (see table 1). They encompass HDTV formats with 16:9 aspect ratios and both progressive and interlaced scanning, and digital equivalents of the standard PAL/NTSC picture formats with both 16:9 and 4:3 aspect ratios. HDTV content is broadcast at 19.34Mbps, using MPEG-2 for both compression and transport [11, 10].

Local area transport of uncompressed HDTV is via coaxial cable or optical fibre, using the SMPTE-292M standard [13]. This is the universal medium of interchange between various types of HDTV equipment (e.g. cameras, encoders, VTRs, editing systems, etc.) at data rates of 1.485 Gbps. It is widely used in studios and production houses, allowing HDTV content to be delivered uncompressed through various cycles of production, avoiding the artifacts that are an inevitable result of multiple compression cycles. If wide area transport is desired, the 292M bit-stream is typically run over dedicated fibre connections, but a more economical alternative is desirable. We consider the use of IP networks for this purpose.

Standards for real-time transport of video over IP networks have reached relative maturity recently, with the dominant protocol being the Real-time Transport Protocol, RTP [20, 21]. RTP provides media framing, identifies the payload type and source, and allows for timing recovery and loss detection. It typically runs on UDP/IP networks, inheriting their limitations: unreliable, best effort delivery. Receivers use information in the RTP headers to correct for packet loss, and to reconstruct media timing. A key feature is application level framing, where the codec output is intelligently fragmented and packetized, according to a payload format, so that each RTP packet can be decoded independently [4]. This makes careful design of receivers important, since they have the primary responsibility for correct playout of media disrupted by the vagrancies of an IP network.

IP networks provide a best-effort packet delivery service. There is no guarantee that the network will not discard, duplicate, delay or mis-order packets. Applications and transport protocols using IP must adapt to these issues, abstracting the network behaviour to give a usable service. RTP applications have developed sophisticated strategies for dealing with timing jitter and packet loss [16]. It is expected that a system for delivery of HDTV over IP will use these to provide a robust service. A critical area where RTP based systems are lacking is congestion control; adapting their behaviour to fit the available network capacity. The implication here is that it is necessary to either develop congestion control for RTP or to run applications only on a network provisioned with sufficient capacity to support their needs. Of course, if it is desired to transmit uncompressed HDTV over IP, the network will need a certain capacity anyway. For this reason, we defer discussion of congestion control to section 6 and concentrate instead on the issue of finding a network that can support gigabit rate IP flows.

There are several networks that have demonstrated sufficient capacity for these experiments. Internet2's Abilene network [17] and the DARPA SuperNet testbed [24] are examples to which we have access. SuperNet has been used to demonstrate performance of 740 Mbps of single stream TCP and 957 Mbps of multi-stream TCP traffic over a cross country path [18].

Our initial testing was conducted over SuperNet between ISI East (Arlington, VA) and CMU (Pittsburgh, PA). The path includes nine hops in each direction and has an RTT of approximately 10 ms (see figure 1). We also conducted tests on a SuperNet path where the packets flowed from ISI East to ISI West (Los Angeles, CA) and back to ISI East. In this configuration, both the sender and receiver were at ISI East. This path has twenty-two hops and an RTT of approximately 67 ms.

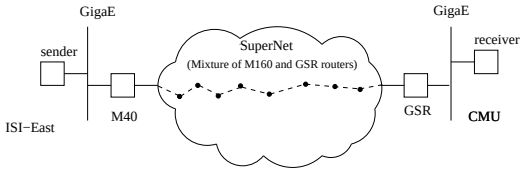

Figure 1: Data path over SuperNet from ISI-East to CMU

Our test configuration consisted of senders and receivers with gigabit Ethernet NICs connected to a switched gigabit Ethernet LAN infrastructure. The local area gigabit Ethernet connected to a site border router. Depending on the site, the border router was either a Juniper or Cisco router which connected to the wide area network via an OC48 POS connection. SuperNet uses a commercially available IP backbone for transport across the wide area. This wide area network consisted of a mix of Juniper and Cisco routers with OC48 and OC192 POS interfaces.

Prior to conducting the HDTV experiments, we first desired to ascertain that sufficient capacity was available across the wide area network. We also desired to accomplish this in a manner which would not significantly disrupt other traffic. TCP's congestion control mechanism provides a good gauge of available capacity. We used the iperf application [9] to measure both TCP and UDP bandwidth performance. Running iperf between our send machine at ISI East and the receiver at CMU, we were able to record a 702 Mbps TCP stream. Likewise we carried out the same experiment for UDP flows and were able to transfer flows in excess of 600Mbps. Performance was dependent on network load, with throughput being less at busy times.

These tests show it is possible to engineer an IP network to have low packet loss and jitter, and support gigabit rate flows. From this, we conclude that the network capacity should be available to conduct tests with HDTV over IP.

A system to transport HDTV over IP networks will use RTP as its transport, with the implication being that an RTP payload format needs to be developed for HDTV content. The options for the development of such a format are outlined in section 3, with the details of our design being presented in section 4.

3 Options for Transport of HDTV

A system for transport of HDTV over IP will accept a SMPTE-292M digital video signal and encapsulate it within RTP for transmission over IP. At the receiver, the SMPTE-292M signal can be regenerated, or the video can be displayed directly. There are a number of options in how this can be done, depending on the aim of the transport. If the intent is to link existing equipment the correct approach may be *circuit emulation*,

where the SMPTE-292M signal is mapped onto IP irrespective of its contents. The alternative is a *native packetization*, where an RTP payload format is defined to transport the video directly, with SMPTE-292M used only locally.

Circuit emulation provides transparent delivery of the HDTV bit-stream, suitable for input into other devices. It supports any format that SMPTE-292M supports, without having to be adapted to the details of that format. The main disadvantage is that the packetization is media unaware, and cannot optimise based on the video format. This makes circuit emulation somewhat loss intolerant.

Native packetization looks at the contents of the SMPTE-292M stream, acting on the video data within it. Hence, a native formats need to be defined for every possible video resolution, although those formats can be made more optimal. It also exposes the content to manipulation by end systems, rather than hiding it within another layer of framing.

We chose to use a native packetization, since one of our aims is to display and manipulate HDTV content on commodity workstations; we do not necessarily need to regenerate the SMPTE-292M output.

4 Design and Implementation

In the design and implementation of our HDTV system, our priority was to use commercial, off-the-shelf, components rather than to develop custom hardware. Accordingly, the core of our system is a high-performance PC, with gigabit Ethernet and an HDTV capture card.

The PC is a Dell Precision Workstation 620 MT with dual 1GHz Pentium III processors, running Linux 2.4.2. It has two 64 bit, 66MHz PCI slots and four 32 bit, 33MHz PCI slots. The 64 bit PCI cards are located on a separate PCI bus to the slower cards. The network interface is a 3Com 3c985 gigabit Ethernet.

For HDTV capture and playout, we use an HDstationOEM [22] card, providing SMPTE-292M input and output. This card can operate in several modes: captioning, capture and playback. We used it to capture HDTV into main memory, and to regenerate SMPTE-292M output at the receiver. Our system can also display HDTV on the workstation monitor, using a software-based decoder. The capture card supports a range of video formats, but our system uses only SMPTE-296M (1280x720 pixels, progressive scan, 60 frames per second) at this time.

The HDstationOEM card provides access to SMPTE-292M content using a FIFO API that transfers frames in order between the host and a capture/playout queue in the card's on-board memory. The API has two modes of operation: Mapped mode maps the memory on the card into the system address space. It is primarily for captioning applications, allowing small changes to be made as frames are filtered through the card. In DMA mode, the application supplies a pointer to a buffer in system memory, and the card fills that buffer with a complete frame. DMA mode is intended primarily for capture and playback applications. As noted later, we experimented with both modes of operation.

We used an updated version of the RTP library from the UCL Robust-Audio Tool [7] to provide the core network functions of our system. This a a complete RTP implementation, supporting IPv4, IPv6 and multicast. Transmission and reception were implemented as two separate programs, because the requirements of the system are such that it is not possible to transmit and receive on the same machine.

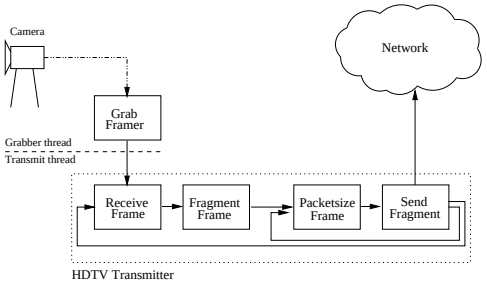

Figure 2: Block diagram of transmitter

4.1 Transmission

A block diagram of the transmitter is shown in figure 2. There are several parts to it: frame capture, fragmentation to match the network MTU, packetization and transmission. The capture process runs in a separate thread to the other operations, because the FIFO API provided by the grabber only supports blocking reads, that have to be overlapped with the other operations.

Once frames have been captured, they are fragmented to fit within the network MTU and transmitted in separate RTP packets. Frames are split into equal sized fragments, with an RTP payload header indicating the offset within the frame.

Our initial design smoothed transmission, spacing packets across the framing interval, rather than sending them in a burst. This proved hard to implement: the inter-packet spacing is on the order of microseconds, and system calls such as `nanosleep()` operate with a 10ms scheduling granularity under Linux, unless real-time scheduling is used. It also takes approximately 30μs to send a packet on our system, which is comparable to the desired packet spacing of 50μs. For these reasons, we reverted to a simple approach, sending packets back to back.

We initially used memory mapped capture, since we do not manipulate the video before transmission. Our hope was that it was not necessary to transfer the data into system memory. Rather we could calculate the fragment size, generate the RTP headers separately, and pass a pointer to the on-board memory on the capture card directly to the kernel via the `sendmsg()` system call. This performed very badly: the memory access patterns used to generate UDP packets are not optimal for the capture card.

Instead, we used DMA mode, where the capture card writes complete frames into memory. The rest of our system was unchanged: we calculate fragment sizes, and use `sendmsg()` to send the packets with a scatter/gather array, to avoid another copy in system memory. The result is that video data passes over the PCI bus twice: once from the capture card into system memory, and again from the system memory to the gigabit Ethernet card. We believe transmission performance could be greatly improved if the kernel was smart enough to use DMA for large copies in the `sendmsg()` system call.

4.2 Reception

A block diagram of the receiver is shown in figure 3. It operates in a classical `select()` loop, with a timeout on the order of the inter-frame time. Each iteration pulls a packet from the RTP stack, performs colour conversion if needed, and inserts the contents into a frame store at the appropriate point. If the packet

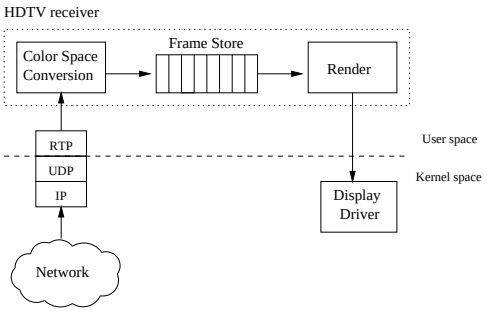

Figure 3: Block diagram of the receiver

is the last in the frame, rendering is triggered. The system also collects performance statistics and performs RTCP processing.

The first stage of reception occurs in the RTP stack. Packets are received from the kernel, validated as RTP, and then passed to the application. The RTP library used was originally written for a voice-over-IP system, and designed for flexibility rather than speed of operation. Despite this, it worked reasonably well at the rates needed for HDTV transport. The areas where performance was limited by the library included:

- Buffer allocation, since the `malloc()` system call is slow. Rather than allocate a new buffer for each packet, the library was modified to reuse buffers where possible.
- Packet validation, since the validity tests defined by RTP require a pass over the header including a number of consistency checks. This was found to take approximately 10% of the total runtime when using hardware rendering, so the library was modified to check only the RTP version number.
- Sequence number validation, as a further validity check, also uses a noticeable fraction of the system runtime, and we considered removing it. Instead, we limited our changes to a rewrite that improves the cache footprint of the code.

A number of system parameters also had to be adjusted before the full data rate could be sustained, as discussed in section 5.

Colour space conversion may be needed, depending on the display device. The HDstationOEM card can directly output the regenerated SMPTE-292M signal, but to render into a window it is necessary to convert from YUV colour space into the RGB space used by the display. Conversion is straightforward, but time consuming, since it requires arithmetic on every sample of the frame. We implement colour conversion in optimised C code, yet it takes over 90% of the total runtime when rendering into a window.

Once any necessary colour conversion has been performed, the fragment is copied to the correct location in the frame buffer. The offset is included as an RTP payload header within each packet, making this a straightforward matter. Since it is advantageous to reduce the number of copies, this step is integrated into the colour conversion code where possible.

The final packet of each frame is indicated by a marker in the RTP header, and this is used to trigger rendering. To regenerate SMPTE-292M output, the FIFO API of the HDstationOEM card is used in much the same way as for frame capture. To render into a window on the display, we use the shared memory extension of the X window system. Since rendering is triggered by receipt of the packet with the RTP marker set, our system is vulnerable to three failures:

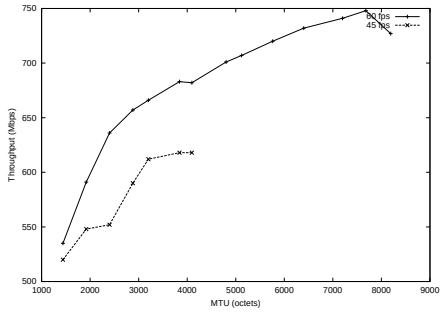

Figure 4: Effects of network MTU on throughout

- If the packet containing the marker is lost, the application will discard the frame.
- If the packet containing the marker is reordered, some fragments will be lost (they arrive after the frame has been displayed).
- If the packet containing the marker is delayed, the frame will be offset from its true playout time.

For our proof of concept system, these issues are considered an acceptable trade-off for implementation simplicity. A robust implementation would use a more sophisticated playout buffer algorithm, to smooth network jitter and to compensate for packet loss.

We have conducted a number of tests of the system performance, which we describe next, and based on these we propose a number of enhancements to our design in section 6.

5 Experimental Performance

5.1 Local area tests

Our initial trials were conducted between two hosts on the same Ethernet segment, connected via an Extreme 5i gigabit Ethernet switch. The aim was to demonstrate that our system could support HDTV delivery on an unloaded network, free from the effects of competing traffic. The tests were successful: when correctly tuned, our implementation is loss free in the local area tests. The tuning process was significant, however, requiring adjustments to the network maximum transfer unit (MTU), socket buffer size and network driver.

With the default 1500 octet MTU, the system throughput is 535 Mbps. This is insufficient for our needs, but gigabit Ethernet interfaces support the jumbo-frames extension, allowing us to increase the MTU to 9000 octets. Increasing the MTU affects throughput as shown in figure 4: larger MTU sizes result in higher throughout. The increase is due to the reduction in the header processing overhead relative to the amount of data, and the reduction in interrupt load on the host. We chose a 4470 octet MTU for our tests, even though that does not give best performance, since it is the maximum supported by the wide area network, and we desired to compare our local and wide area results.

The default 64k socket buffer was insufficient, and caused packet loss in the receiving host. Experience shows that the buffer may need to be large enough to store the packets for an entire frame of video. This because the receiver is not able to process packets continually: there are some periods when it is busy

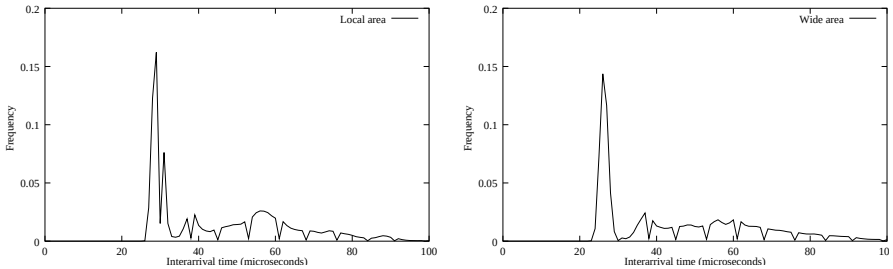

Figure 5: Inter-packet timing at the receiver

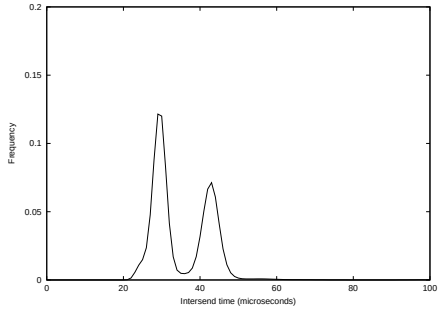

Figure 6: Inter-packet timing at the sender

processing the video, and cannot receive packets. This is a particular problem with our software decoder, since colour conversion takes a significant amount of time. Multithreading the receiver is expected to reduce the buffering requirements, since it will allow concurrent packet reception and media processing/decoding on dual processor systems.

The gigabit Ethernet driver performs interrupt coalescing and checksum offloading, and had delayed packet notification enabled. Adjusting these parameters does not appear to significantly affect performance, and after much experimentation we settled on the default values.

With these changes in place, the system can sustain a transfer rate of 615 Mbps, with no packet loss. This allows us to send 1280x720 pixel video at 45 frames per second, using 8 bits per colour component. Once packet loss was eliminated, we made two sets of measurements relating to the network timing jitter: the inter-packet timing and relative transit delay.

Figure 5 plots inter-packet timing against relative frequency of occurrence, for both local and wide-area tests. It should be compared with Figure 6, which shows timing measured at the sender. As can be seen, the inter-packet timing is strongly bi-modal, a result which surprised us since the transmitter sends the packets that comprise each frame in a tight loop (there is a much smaller peak at the location of the inter-frame interval, which is not visible in the figure). The bimodality seems due to the sender blocking in the transmit call, perhaps due to limited buffering in the network card. If the on-board buffer is full, we expect the system blocks until the packet is sent, causing some packets to be delayed. The effect of network transit is to smear this second peak out in time. The receiver sees the initial peak in the inter-packet timing, with the same interval as the sender, but a broader tail to the distribution.

The relative transit delay, the difference between the arrival time in RTP clock units and the send time in the same units, is illustrated in Figure 7. We note variation of approximately 10ms, equal to the Linux

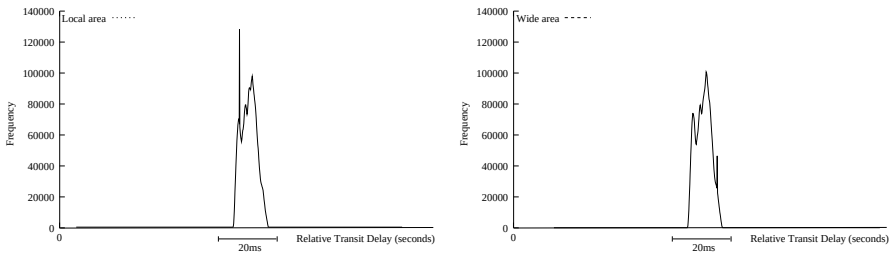

Figure 7: Relative transit delay

Loss event duration	Frequency
No loss	24697400
Single packet	85797
Two consecutive packets	587
Three consecutive packets	7
Four or more packets	0

Table 2: Observed packet loss rates

scheduling interval, and believe these measurements are heavily influenced by the time taken to wake the receiver on arrival of the first packet in a frame, and that such large variation in network transit time is largely a measurement artifact. Further study is needed to confirm this result.

Surprisingly, our tests also revealed the presence of a small amount of packet reordering between hosts on the same Ethernet segment. Typical measurements showing approximately 1 in 10000 packets being delivered out of order. Reordering persists when the two hosts are connected back to back using cross-over fibre, and can also be demonstrated with the iperf tool [9]. We have no good explanation for this, but suspect a race condition in the Linux 2.4 kernel, triggered by our use of dual processor systems.

Video quality was excellent during the tests, although 45 frames per second does not result in optimally smooth motion (every 4th frame of the original is dropped, so the frame timing is not uniform).

5.2 Wide area tests

We conducted a number of wide area tests of our system, using various paths across the DARPA SuperNet testbed. The first factor evaluated was packet loss on the wide-area network path. This was partly to ensure that we were not causing network congestion, and partly to determine the effects of packet loss on the video quality. Packet loss was difficult to measure, since the network commonly operated without loss. Table 2 shows typical measurements when the network was loaded: approximately 0.3% of packets were lost, with most loss events being of isolated packets. More common was the case where no loss was observed.

The distribution of inter-packet arrival times and relative transit delay for the wide area network path, shown in Figures 5 and 7, is almost identical to that for the local area tests. The network is lightly loaded, and hence there is no significant queueing jitter to impact the packet timing.

As to be expected, some small degree of packet reordering was present on the wide area path. Typical results showed 0.05% of packets delivered out of order, with the vast majority of reordering events being

of adjacent packets. In rare cases, we observed packets being delivered two or three out of sequence. This degree of reordering is not unusual, similar values have been reported by [2, 3, 15]

Video quality for the wide-area tests was subjectively identical to that observed in the local tests.

In addition to the closely monitored tests over DARPA's SuperNet, we also demonstrated the system at the SuperComputing 2001 conference in Denver, November 2001. In this demonstration, the system was run over a path from Washington D.C to Denver. The network path for his demonstration utilised Internet2's Abilene and the Washington D.C. area MAX gigapop. This path was 10 router hops with a RTT of approximately 43 ms. The backbone links along this path were OC48 POS and there was no advance resource reservation. We have no formal measurements of the system performance over this path, but informal observations of the system showed negligible packet loss and jitter. Very high quality video was received, with no apparent problems, for a period of several hours.

6 Limiting Factors and Future Directions

Our experiments were conducted using 615 Mbps media streams, comprising 1280x720 pixel images at 45 frames per second with 8 bits per colour component. This is insufficient for true uncompressed HDTV, which requires 850 Mbps to increase the frame rate to 60 frames per second, and 1.03 Gbps for full colour.

PCI bus contention appears as the main limiting factor. We initially put the HDTV capture card and the gigabit Ethernet into the two 64 bit/66 MHz PCI slots on the PC, but tests showed that performance *increased* when we moved the Ethernet card onto a slower 32 bit/33 MHz PCI slot. Investigation showed that the fast PCI slots shared a single bus, distinct from that used by the slower slots, leading us to believe that contention on the bus was an issue. We have under development a system using a PC with dual 64 bit/66MHz PCI bus architecture, which we expect to reach the 850Mbps data rate needed for full frame-rate HDTV, subsampled to 8 bits per component.

To support the full colour depth – 10 bits per component – we need a faster network interface, for example a PCI-based OC-48 interface. Our initial experiments with such interfaces have been disappointing, with the available cards being unable to exceed the transfer rates achievable using gigabit Ethernet. Use of dual gigabit Ethernet may also be possible, but we may again run into the limitations of the PCI bus.

An alternative to faster networks may be use of lossless video compression, to reduce the bandwidth requirements of the system. This is an area to be explored in future, although it is not clear that commodity systems can compress gigabit rate streams in real-time.

Memory bandwidth also limits performance; the system has been refactored several times to reduce the number of copies, increasing performance. This is especially an issue for the software decoder, rendering into a window, due to the need for colour conversion. Use of MMX extensions is expected to help, as will off-loading conversion using the hardware acceleration available in some display adaptors. Interrupt processing overheads are also a factor, evidenced by increased throughput when larger packets are used. This is an area where wide area networks limit performance, since they limit the MTU to 4470 octets.

Our implementation has a simple playout routine, and does not deal well with jitter or loss around frame boundaries. We plan to implement an adaptive playout buffer, to compensate for jitter and to correct frame playout time. Once this is done, we plan to study more sophisticated error correction and concealment algorithms. At present, packet loss is concealed by repeating part of the previous frame to cover the missing data. If frames are buffered before playout, it will be possible to use some form of FEC (e.g. [19]) or retransmission to correct lost packets.

Congestion control is a serious issue for high rate UDP applications on the current Internet. Our implementation is not currently congestion controlled, raising the issues of fairness to other traffic and potential congestion collapse of the network. Before we deploy our system outside a controlled environment, we need to implement some form of rate limiting or congestion control. The TFRC protocol [5] might be an appropriate means of congestion control, but more work is needed to implement and evaluate this.

Our implementation uses a simplistic RTP payload header, consisting only of the fragment offset within a frame (the fragment length being inferred from the packet length). A more general payload format for uncompressed video, better preserving frame metadata should be defined. The RTP payload format for BT.656 video [25] may be suitable as a base design, although it will need extension for HDTV formats.

7 Related work

A product from 2netFX [1] delivers compressed HDTV over IP. The system uses MPEG-2 compression at 19.2Mbps using the standard RTP payload [8]. The use of compression adds latency and makes this system unsuitable for environments where video editing is performed, or where full quality is needed.

The University of Washington have demonstrated a system for transport of HDTV over IP [14]. This system uses Sony HDCAM compression at 270 Mbps. This is a proprietary production quality compression scheme, supporting a limited number of edit cycles without significant quality degradation.

A prototype developed by Tektronix [23] uses custom hardware to deliver HDTV over an OC-48 POS interface. The system performs circuit emulation of SMPTE-292M over IP at 1.5 Gbps using an RTP payload format [6] developed in conjunction with the University of Washington and ourselves. This system was also demonstrated at the Super Computing 2001 conference.

Most similar to our work is the system built by NTT Laboratories, which was demonstrated in Tokyo, October 2001 [12]. This system is built around a multi-processor PC running Linux, with a commercial HDTV capture card, but uses a custom network interface.

These latter two systems suffer from being implemented using custom hardware. This makes them expensive and inflexible, compared to a system built using off the shelf components. Their advantage is that they have better performance at present, although we expect that Moore's law will close this gap rapidly.

8 Conclusions

We have successfully demonstrated a prototype system for transport of uncompressed HDTV over IP networks, which we believe is the first built using commodity components. The system currently supports SMPTE-296M format pictures at a reduced rate of 45 frames per second, with colour sub-sampled to 24 bits. An enhanced version is under development, which we expect to support the full 60 frames per second, and we further plan to extend the system to deliver full quality uncompressed video.

There are a number of challenges to supporting full uncompressed HDTV, primarily due to limitations of the end system. We have described a number of areas where performance may be improved; further work will implement some of these ideas. Many of these techniques are also valid for high bit rate compressed video, transport of HDTV over IP provides an appropriate tested, but is not the only application that may benefit from this work.

Acknowledgements

This work is supported by DARPA ITO under the Next Generation Internet program, and by hardware donated by Intel corporation. The MAX gigapop and Abilene NOC provided assistance in conducting the wide-area tests. Juniper Networks loaned equipment for our demonstration at SuperComputing 2001.

References

[1] 2netFX. Thundercastip advanced media server. http://www.2netfx.com/.

[2] J. C. R. Bennett, C. Partridge, and N. Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Transactions on Networking*, 7(6):789–798, December 1999.

[3] E. Blanton and M. Allman. On making TCP more robust to packet reordering. *ACM Computer Communication Review*, January 2002.

[4] D. D. Clark and D. Tennenhouse. Architectural considerations for a new generation of protocols. *Computer Communications Review*, 20(4):200–208, September 1990.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, 2000.

[6] L. Gharai, G. Goncher, C. S. Perkins, D. Richardson, and A Mankin. RTP Payload Format for SMPTE 292M. Internet Engineering Task Force, July 2001. Work in progress.

[7] O. Hodson and C. S. Perkins. Robust-audio tool, version 4. http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/.

[8] D. Hoffman, G. Fernando, V. Goyal, and R. Civanlar. RTP Payload Format for MPEG1/MPEG2 Video, January 1998. RFC 2250.

[9] Iperf. http://dast.nlanr.net/Projects/Iperf/.

[10] ISO/IEC. Generic coding of moving pictures and associated audio information: Systems, 1996. ISO/IEC 13818-1.

[11] ISO/IEC. Generic coding of moving pictures and associated audio information: Video, 1996. ISO/IEC 13818-2.

[12] NTT Innovation Laboratories. Uncompressed HDTV transmission system over the internet. NTT Press Release, October 2001. http://www.ntt.co.jp/news/news01e/0110/011026.html.

[13] Society of Motion Picture and Television Engineers. Bit-serial digital interface for high-definition television systems, 1998. SMPTE-292M.

[14] University of Washington. Internet hdtv. http://www.washington.edu/hdtv/.

[15] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM Transactions of Networking*, 7(3), June 1999.

[16] C. S. Perkins, O. Hodson, and V. Hardman. A survey of packet loss recovery techniques for streaming media. *IEEE Network Magazine*, September/October 1998.

[17] The Internet2 project. http://www.internet2.edu/.

[18] SuperNet Land Speed Record. http://www.ngi-supernet.org/wan-speed.html.

[19] J. Rosenberg and H. Schulzrinne. An RTP payload format for generic forward error correction, December 1999. RFC 2733.

[20] H. Schulzrinne. RTP profile for audio and video conferences with minimal control, January 1996. RFC 1890.

[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications, January 1996. RFC 1889.

[22] DVS Digital Video Systems. Hdstationoem card. http://www.dvs.de/english/products/HDStationPRO/HDStationOEM.htm.

[23] Tektronix. Universal network access system. http://www.tektronix.com/Measurement/commtest/darpa/darpa.html.

[24] The DARPA SuperNet testbed. http://www.ngi-supernet.org/.

[25] D. Tynan. RTP Payload Format for BT.656 Video Encoding, October 1998. RFC 2431.