Large Group Teleconferencing: Techniques and Considerations [*]

Ladan Gharai   Colin Perkins   Allison Mankin

USC Information Sciences Institute

3811 N. Fairfax Drive, Suite 200

Arlington, VA, 22203

March 8, 2002

Abstract

Much work has focused on the problems of small group communication and of one-to-many broadcast, while issues in large scale interactive networked teleconferences have received less attention. In this paper, we consider the problems inherent in conducting large scale conferences: teleconferences with hundreds, or perhaps thousands, of active participants. The lessons learnt from our design for a digital amphitheater – a system based on active agents, where about one hundred remote participants can conference together – are discussed. In that system we successfully overcame end system limitations by off-loading some processing into the network, thus creating parallelism and reducing the bottleneck inherent in the serial nature of the hosts managing each display. We expand on this architecture, further exploring parallelism by pushing functions from individual end systems, to clusters and the network, with the aim of scaling to thousands of users.

[*] This paper is based upon work supported by the Information Technology Office of the Defense Advanced Research Projects Agency. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. The authors may be contacted as {ladan|csp|mankin}@isi.edu

1 Introduction

The infrastructure of the Internet has grown at an incredible pace since its inception. With the widespread introduction of optical networking, bandwidth has become plentiful and wide-area networks operating at OC-48 rates, such as the SuperNet [1] and Internet2 [2], are common with OC-192 and other higher rate connections becoming available. This growth in wide area connectivity has been matched by improvements in LAN technology: 100 Mbit Ethernet is ubiquitous, 1 gigabit is becoming common, and 10 gigabit Ethernet is now being introduced.

Parallel to the growth of the infrastructure, consumers and academics have been busy envisioning new and far reaching applications: the traditional uses – email, netnews, file transfer – are giving way to more interactive applications based on the world wide web, to streaming media, digital radio, television and cinema, and to real-time interactive teleconferencing.

There is another variable which affects this happy growth in network bandwidth and application demand: the performance of the end-systems. According to Moore's Law, the number of transistors that will fit on a chip doubles every 18 months, and performance closely follows this. This is an impressive increase, but is dwarfed by the rate of increase in network capacity, which has grown at a much faster pace (figure 1). Given the availability OC-48 PCI network interface cards, gigabit – and soon 10 gigabit – Ethernet cards, the question remains: is current processing power capable of processing such data rates?

In this paper we explore end-system limitations in the context of scaling video conferencing, tele-conferencing hundreds, or perhaps thousands, of participants. Such a conference may be held worldwide with participants from different university campuses, corporate facilities, government organizations or even a lone participant from the Arctic. In this work we draw from our experience with the digital amphitheater, where we successfully video conferenced close to one hundred participants.

We begin, in section 2, by further describing the architectural implications of scaling teleconferencing systems.

[1] http://www.ngi-supernet.org/

[2] http://www.internet2.org/

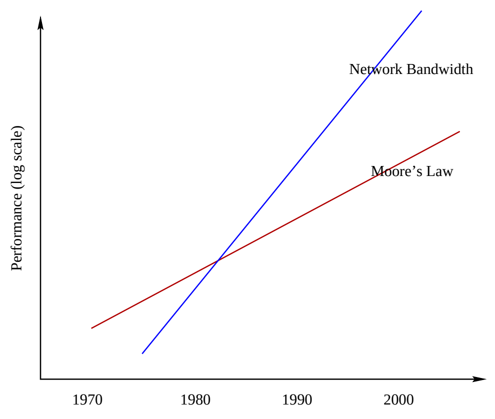


Figure 1: Moore's law vs. network growth.

Next, in section 3, we briefly describe our initial experiences with a prototype large-scale conferencing system – the digital amphitheater – and the lessons learnt from it. We expand on this experience to explore issues related to scaling up further, to possibly thousands of participants, in section 4. Finally we discuss related work in this area, section 5, and conclude in section 6.

2 Architectural Implications of Scaling

We envision a system capable of supporting several hundred, perhaps one thousand, simultaneous interactive users. The benefits of such a system are obvious: large organizations can have regular meetings with all levels of management involved without incurring high travel cost, long distance educational programs can meet as if within a lecture hall while students and lecturers join from geographically disparate locations, or it could be used for political and other debates.

Video teleconferencing among small groups of people is now quite common, and is supported by a number of commercial and open-source tools. However large structured meetings, on the scale that we are envisioning, have not yet been tried. There are a number of reasons for this: processing such a large number of video streams

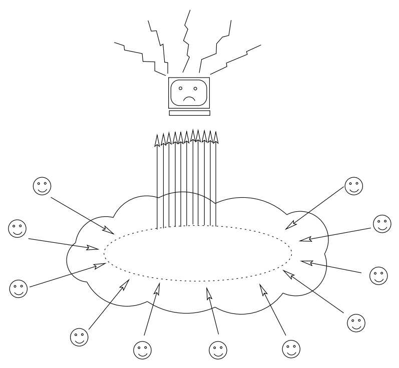


Figure 2: The end-system bottleneck.

presents a formidable challenge, both in the network and for the end-user application, and display technology is often a limiting factor. Processing over a thousand video streams can easily overwhelm most consumer grade workstations, in terms of bus access, interrupt processing, context switching, packet handling and demultiplexing, decoding, display processing and rendering.

Many of the current teleconferencing tools, especially the research oriented ones such as the popular multicast toolset maintained by University College London [13] have been designed with scaling properties in mind. However, their focus has been mainly on attaining scaling via multicast, and thereby reducing network load. This approach does not address the problem of the end-system bottleneck, and in fact it aggravates it. End-users can generate video content in parallel, this content moves through the network, but once received at its destination, must be processed by an inherently serial system. As all the video flows must be instantaneously reconstructed, decompressed and rendered (figure 2), thereby creating a performance bottleneck in the end-system.

Given that the processing limitations of end-systems are the main bottleneck and deterrent to very large scale video conferencing, what are the possible solutions? Our experience shows that the simple brute force technique of ‘faster end-systems’ is not a viable solution, as even the fastest available workstations cannot keep up with hundreds of video streams.

The implication is that we must distribute the processing, leveraging the increased communication ability rather

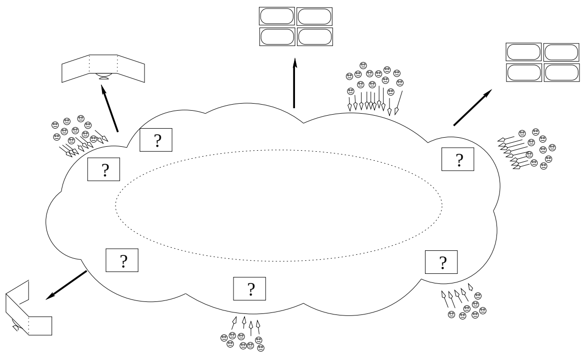


Figure 3: Large group conferencing: An architectural template.

than drinking from the firehose of the full set of input streams. Parts of processing must be pushed into the network infrastructure, offloading functions from the end-system to agents within the network (figure 3). The questions remains as to how much and which parts of the process can be off-loaded from the end-system, and exactly what are the tradeoffs involved.

We have explored some of these tradeoffs, and the performance which can be gained through the use of agents, in our prototype digital amphitheater. In the next section we discuss this in some detail, followed by an evaluation of the extensibility of this model to other large scale conferences.

3 The Digital Amphitheater: Prototyping Large-Scale Conferencing

When attempting to conduct a teleconference with upwards of one hundred participants using the standard multicast toolset, it rapidly becomes clear that those tools are unable to process the received data: the system load goes to 100%, data is dropped, and the visual appearance of the conference is destroyed.

There can be a number of reasons for this poor performance: it could be that the system cannot handle the total bandwidth of the incoming media streams, it could be the per-packet processing, it could be limited CPU performance, or it could be limited memory bandwidth in the host.

Our initial investigation led us to believe that the limit was not related to the raw bandwidth of the media streams: experiments with high rate TCP and UDP traffic on similar hosts have shown that they can receive significantly higher data rates, if tuned correctly. Those experiments also suggest a number of approaches to tuning the system, including: use of large frames, interrupt coalescing, zero copy networking, and checksum offloading [3].

When considering these, our first observation was that the media streams comprise a large number of small packets, due to the compressed nature of their payload. If these packets could be combined into larger frames, it might be possible to increase performance without having to tune the end host. This can also be expected to ease the application performance, by reducing the number of participants it must track.

This is a simple matter for TCP streams, since the communication is point to point and the data can be split arbitrarily. For real-time communication, however, the problem is more complex due to the following factors: (1) a video conferencing session naturally involves multiple sources; (2) the size of the packets generated depends on the compression scheme used, and cannot be arbitrarily varied. In particular, any change in the compression ratio to affect the packet size will vary the rate, defeating the point of the change.

We devised a technique we have term 'Spatial Tiling' which address the above constraints: combining data from multiple sources whilst maintaining the rate [7].

3.1 Spatial Tiling

Our concept of spatial tiling is to tile N frames from separate sources next to each other, and to modify the meta-data of the tiled frame, such that it represents a single frame. This is illustrated via an example in figure 4 where three individual video frames are placed side by side to form a single frame. Each individual frame is completely represented in the tiled frame, however the meta-data, in this case block coordinates, has been adjusted accordingly. We believe tiling satisfies both of our constraints: packetizing the tiled frame provides more opportunity for generating fuller packets, and the number of input sources is reduces from N to a single source.

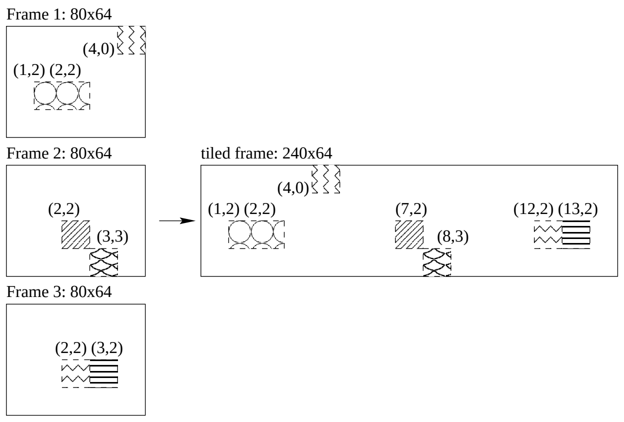


Figure 4: Tiling three frames into a single frame. Both frame size and block coordinates have been adjusted for the tiled frame.

It is important that spatial tiling does not add additional delay to the video stream. Tiling agents only parse and deconstruct the incoming video streams into smaller building blocks, whilst maintaining their relevant meta-data: no decompression is done in the tiling agent. To maintain independence between incoming and outgoing frame-rates, two sets of buffers are maintained per stream. The tiled frame is constructed at given intervals (determined by the outgoing frame rate) from the output buffers. New incoming frames are copied from the incoming buffer to the output buffers, once they are received in full.

Although, theorically, it is possible to tile an unlimited number of streams, we have restricted the tiling to 15 video streams. This restriction allows us to use the built in mixer functionality of RTP/RTCP [15], since an RTP packet can carry the contributing source identifiers for up to 15 different sources. The input streams can be tiled in any geometry requested: for 15 streams the agent can generate a single row of 15x1, a square of 4x4 (where the last square will be empty), a 5x3 rectangle, or even a single row/column.

In our current implementation, the spatial tiling agents support two video representations: high bandwidth raw YUV video with conditional replenishment (YUVCR) [8] and H.261 [17] using only intra-frame compression. Spatial tiling agents may be employed within a standard video conferencing session, or in conjunction with a

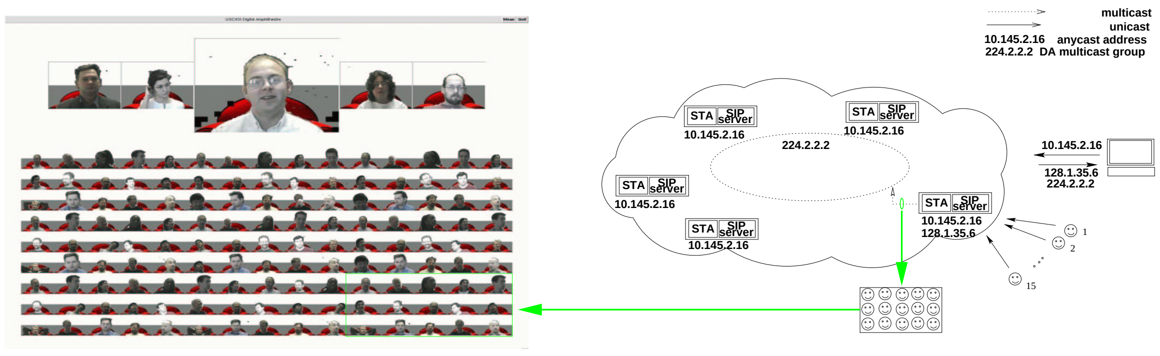


Figure 5: The digital amphitheater: user interface and architecture.

special purpose application, such as our digital amphitheater [14] system.

Figure 5 displays the digital amphitheater's user interface (with a conference in session). The amphitheater consists of the rows of attendees, four front panel members and the speaker. In all instances the background of the offices have been removed, in order engender a feeling of presence and location. To do so we have adapted a simple, low cost background substitution algorithm which runs in real time on the senders systems.

Each attendee in the amphitheater participates by unicasting video to the 'closest' agent, located via an anycast address. The agent, in turn, tiles together all the video streams it receives, and multicasts the tiled video stream to a multicast group. All participants join this group, receiving and displaying the combined audience video. The panel members and speaker send directly to the multicast group, thus avoiding the tiling.

3.2 Performance Gains

To determine the performance gains obtained by using the tiling agents we decided to measure and quantify: (1) bandwidth, in bits per second (bps); (2) packets per second (pps); and (3) the total number of streams the end system is capable of decoding and rendering (N). We compared the value of these variables in a conferencing session with and without the use of STAs. Here, we only present results obtained from the H.261 tests (results

Variable	seperate	tiled	gain%
pps	186	122	34.51%
bps	979	936	4.33%
N	55	97	43.2%

Table 1: Variables quantified with and without the STAs for H.261 video: average bit-rate, bps, average packet rate, pps, and N total number of streams.

with YUVCR support these conclusions, and are omitted due to lack of space).

The receiving system was what is currently considered an average user grade system: a 550Mhz Pentium III machine with 256M of memory, running Red Hat Linux 7. The tiling was initially run on a somewhat lower grade system, a 400Mhz Pentium II with 64M memory, running FreeBSD 3.4, but this was found to have insufficient memory, although it was sufficient in other ways. A more powerful system, with 512M of memory, was used to host the tiling agents during the tests we report. Work is underway to reduce the memory footprint of the tiling agents, since they are otherwise not very compute intensive and require only a few percentage of CPU time.

In our initial set of trials, we measured bps and pps. To do so, first we streamed the 15 test video streams individually to the receiver. Next, we ran the test video through the tiling process with the output frame rate set to 8 fps - essentially the same as the input frame rate of the test videos. To measure the bitrate, bps, and packet rate, pps, we instrumented the receiver such that it logged these variables, along with other decoding statistics, to a file. Figure 6 displays the results of these tests. In these graphs, pps clearly show a reduction for the tiled H.261 stream. Although not apparent, bps, also shows an overall reduction of 4.33% percent for the tiled stream over the entire test run. Table 1 summarizes these results.

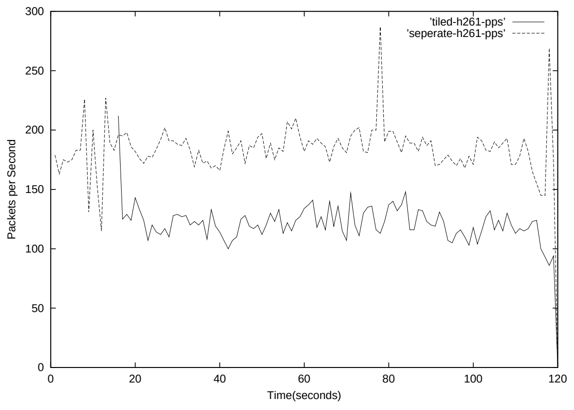
The reduction in pps is primarily due to the aggregation of smaller packets. The tiling process generates a single large frame, therefore there are fewer ‘half empty’ packets in the resulting stream. In the tiled H.261 stream pps is reduced by approximately 35%. Given the average size of H.261 packets such a decrease is to be expected.

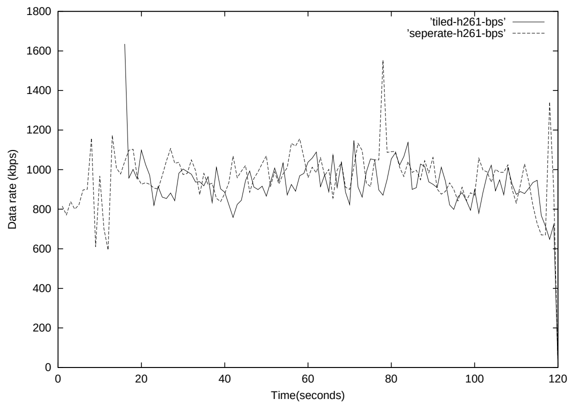
Figure 7 displays the cumulative packet size for 15 individual H.261 streams and their corresponding tiled video frame.

In terms of bandwidth, bps is reduced by 4%. Although bandwidth is reduced over the duration of the test runs, the graphs reveal that this in not the case on a per minute bases, as in some instances the bps of the separate streams appears to be less than the tiled stream. This is in part due to synchronization differences between the separate streams and the tiled stream, and in part due to measurement artifacts resulting from the averaging process. We also note the the low reduction in bandwidth is to be expected. In these tests the tiling agents reproduce the input video streams, exactly as they come in, without any temporal or spatial down sampling. Both the input and output frame rates are 8 fps and the tiling agents more or less copy each incoming frame to the outgoing tile frame. The existing reduction in bps is mainly a reflection of the reduced pps and lower packet overhead,

Finally, we turned our attention to the performance of the end-system, and quantifying N. Our decoder maintains statistics on the number of packets correctly decoded and on packets discarded due to late arrival or lack of rendering time. We used these statistics to measure the maximum number of streams, N, our end-system could receive without loss, both with and without tiling. This process was conducted by incrementally increasing the number of individual streams until the end-system reached the point of saturation. For the H.261 video streams it was found that the system could decode and render up to 55 individual video streams without loss. With this number of streams CPU was at 100% utilization. When receiving tiled H.261, the system could receive 6 tiled streams of 15 and an additional stream of 2 tiles, comprising a total of 97 individual streams, an overall increase of 43% in number of streams.

These numbers clearly demonstrate the reduction of workload on the end-system due to the spatial tiling process. Despite almost no reduction in bit-rate, the end-system is capable of receiving almost twice as many video streams, once the video streams are tiled and packet rate is reduced. This leads us to conclude that a primary load on end-systems is per-packet interrupt processing and per-source rendering, rather than the computational complexity of the decoding process and therefore spatial tiling is more amendable to relatively highly compressed video streams where the average packet size is significantly smaller than the network MTU. Having a significant number of 'half-full' packets, gives the tiling agents more leverage in reducing the overall packet rate.





a. b.

Figure 6: Comparison of 15 separate H261 QCIF video streams and the equivalent 5x3 tiled stream (a) packets per second (b) bits per second.

3.3 Limitations and Lessons Learnt

The spatial tiling agents were designed with two goals in mind: (1) reducing the number of sources visible to the receiver; and (2) reducing the number of smaller packets, combining them into larger output. Clearly this has resulted in performance increases: the number of streams received successfully by our end-system has almost been doubled, whereas the number of streams has actually been reduced from 55 to 5, as the 97 tiled streams, are really 5 separate streams.

By allowing the tiling agents to packetize bigger frames, we were able to produce 35% less packets of which over 80% are at MTU size (figure 7). This is in contrast to having only 40% of packets at full capacity.

Although, it is very promising that 80% of the packets are mostly full, it also indicates that we have reached the limit of what can be achieved via spatial tiling. Employing a hierarchy of tiling agents, or tiling more than 15

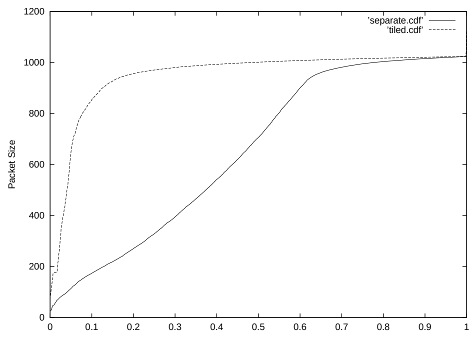


Figure 7: Cumulative distribution of packet sizes, showing increase in the fraction of large packets after tiling

streams, may reduce the number of sources received by an end-system, but it is unlikely that it reduce pps any further.

4 Challenges in Further Scaling

Experience with the digital amphitheater has shown the benefit which can be gained by coalescing packets within the network to reduce the load on an end-system. This is effective at reducing the packet rate up to a point, but rapidly runs into a bottleneck due to limitations of the network MTU.

To further scale the system we need to consider the other limiting factors, starting with general issues in optimizing the network stack, and moving on to issues with efficient RTP processing.

4.1 Optimizing the network stack

There are a number of ways in which the performance of the network stack can be improved, many of which have been explored in the context of high performance TCP implementations. These optimizations fall into two major categories: those which improve the per-packet performance of the stack, and those which reduce the per-byte overheads.

The major issue with respect to per-packet overheads is the interrupt processing load. High performance TCP implementations reduce this by sending larger packets, a solution which we have explored for teleconferencing through our use of spatial tiling.

Another solution is interrupt coalescing, where the network driver gathers several packets before signaling to the operating system that data has been received, rather than generating an interrupt for each packet. This is clearly of benefit, and needs little further discussion.

Reduction in the per-byte overhead is typically achieved with zero copy network stacks, where the network hardware writes into a buffer which is directly mapped into the application's memory space. This is useful, up to a point, but the gains which can be achieved are perhaps limited compared to some other applications.

The main issue is decompression of the media stream, which not only results in a copy of the data being made, but it's expansion. For example, a factor of ten compression is not unreasonable, yet the memory traffic generated by such a decompresser will dwarf the gains from a zero-copy network stack.

The other common technique used to improve performance of TCP is checksum offloading, where the network interface verifies the TCP or UDP checksum before passing the packet to the host. Once again, there is some gain for teleconferencing applications, but due to the processor intensive nature of these applications it may have limited impact (for example, the cost of computing a UDP checksum is negligible compared to that if MPEG decompression).

To conclude, there is some gain to be had from optimizing the UDP/IP network stack, but the nature of large scale teleconferencing is such that the RTP and application processing costs dwarf those of UDP/IP network processing. The reduction in packet rate achieved through spatial tiling is perhaps the most significant effect we can expect, followed by the gains from zero-copy stacks.

4.2 Optimizing RTP

Rather than considering the network stack alone, it is instructive to view the complete application, and consider how performance of the RTP protocol and media decoding can be optimized. Such an approach fits well with the nature of RTP, which was designed around the concepts of application level framing [4] and integrated layer processing.

There are two major issues to consider when optimizing RTP and application performance: participant state and media decoding performance.

An application using RTP will maintain a significant amount of state for each participant: the 32 bit synchronization source identifier, playout calculation details, decoding state, media data awaiting playout, reception quality statistics, source description information, etc. In total, this can easily comprise several hundred bytes per participant, excluding the media data.

When compared to the cache sizes of modern processors, where 64kBytes is considered large, it is clear that the complete state cannot fit into the level 1 cache. It may therefore be advantageous to split data structures into those parts necessary for decoding each packet and those parts which are required less often, so that unnecessary references to main memory can be avoided.

The use of tiling agents within the network has the unintended consequence of reducing the amount of state which has to be accessed during decoding. Since they act as synchronization sources for the media, it is necessary for the receiver to maintain state per-agent, rather than per participant.

As we have noted, media decoding is significant in terms of both processor utilization and memory bandwidth. The obvious result of this is that zero copy techniques must be adopted within an application, with media being rendered directly onto the display device if possible. In addition, the use of various SIMD extensions to processor instruction sets (e.g. MMX, AltiVec, VIS, etc.) can significantly reduce decoding times.

Media decoding is also a function which is parallelizable: each member of the session can be independently decoded and rendered. There is a clear benefit to be gained through the use of multiprocessor hardware, but this still suffers from a bottleneck due to the single network socket used to receive data.

We believe that there is a significant amount to be gained through the use of layered coding, in conjunction with multi-processor systems. The use of layered coding allows the network interface card to filter unwanted traffic, so each processor sees only a fraction of the total. This gives the benefit of parallel decoding, along with a signification reduction in the amount of state which needs to be kept.

It may be advantageous for agents near the edges of the network to combine outgoing data into a single stream, and split incoming data into layers. The trade-off for optimal performance at the end host and in routers is different: it is better for data to be layered at the edges, so the host can separate processing, but it is better to be combined in the core where packet switching is not an issue but per group state maintenance is. This is an ideal use for agents: offloading processing from groups of hosts at the boundary between relatively low speed local networks, and the high speed core network.

5 Related Work

The use of active service agents [1] to adapt the behavior of network traffic flows has been widely studied. Active services avoid the well known problems of active networks by restricting computations to the application layer, deploying services onto a network of computation servers placed within the network. As a result of this, they are readily deployable and form the basis of a number of commercial content distribution networks.

Whilst these commercial offerings have typically focused on efficient distribution of world-wide web content, a number of researchers have studied the problem of adapting streaming audio/video flows to match the network capacity.

One of the earliest such papers referred to self-organized transcoding of streaming audio/video media flows [11], leveraging from tools such as the video gateway developed at UC Berkeley [2]. More recently, implementations such as the ‘FunnelWeb’ Application Level Active Network [6], active routers [10], and overlay networks, such as the X-bone [16], add genericity and flexibility to the system.

The use of these techniques, whilst beneficial to the network, degrades the quality of the media stream. It would be desirable if the load generated by a media stream could be reduced whilst retaining its quality. Our proposed STA network has this property, at the expense of limited adaptability (when compared to schemes based on transcoding).

Critical to the operation of active services is the placement of the active elements within the network. This has received considerable attention in the literature [18, 12, 5], particularly when related to reliable multicast, leading to recent standards work in the IETF [9]. We do not seek to design new tree building mechanisms at present, rather we rely on existing work.

Our proposal seeks to leverage existing work in the field of active services: the concept of active agents within the network to adapt media flows to fit network/system constraints, the platforms for service creation and deployment, and mechanisms for placement of service agents.

6 Conclusion

We have presented the digital amphitheater, a system for large-scale teleconferences, based around the use of active agents to tile video streams, reducing the load on the receivers. This system illustrates one approach to scaling a teleconferencing system to large numbers of participants.

We have also present some preliminary thoughts on how we can further scale the system, to larger or higher quality conferences. In particular, we note the beneficial effect of RTP mixer/translators in reducing the state requirements for end systems, and their potential role layering media streams to enable efficient parallel decoding.

The requirements for conducting large conferences are difficult to meet, and no existing application is entirely successful. The digital amphitheater is a step towards the solution, and points the way to further development and performance improvements.

References

[1] E. Amir, S. McCanne, and R. Katz. An active service framework and its application to real-time multimedia transcoding. In *Proc. ACM SIGCOMM 1998*, Vancouver, BC, 1998.

[2] E. Amir, S. McCanne, and H. Zhang. An application level video gateway. In *Proc. ACM Multimedia'95*, San Francisco, CA, November 1995.

[3] J. Chase, A. J. Gallatin, and K. G. Yocum. End system optimizations for high-speed tcp. *IEEE Communications Magazine*, 39(4):68–74, April 2001.

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

[5] P. Francis. Yallcast: Extending the Internet multicast architecture. Technical report, NTT Information Sharing Platform Laboratories, September 1999.

[6] M. Fry and A. Ghosh. Application level active networking. *Computer Networks*, 31(7):655–667, July 1999.

[7] L. Gharai, C. Perkins, and A. Mankin. Scaling video conferencing through spatial tiling. In *Proc. NOSSDAV 2001*, Port Jefferson, NY, June 2001.

[8] M. Handley. YUV-CR codec for vic. Personal correspondance.

[9] M. Kadansky, B. Levine, D. M. Chiu, B. Whetten, G. Taskale, B. Cain, D. Thaler, and W. H. Koh. Reliable multicast transport building block: Tree auto-configuration. Internet Engineering Task Force, November 2000. Work in progress.

[10] R. Keller, S. Choi, D. Decasper, and M. Dasen. An active router architecture for multicast video distribution. In *Proc. IEEE Infocom 2000*, Tel Aviv, March 2000.

[11] I. Kouvelas, V. Hardman, and J. Crowcroft. Network adaptive continuous-media applications through self organised transcoding. In *Proc. NOSSDAV '98*, Cambridge, UK, July 1998.

[12] B. N. Levine, S. Paul, and J. J. Garcia-Luna-Aceves. Organizing multicast receivers deterministically by packet loss correlation. In *Proc. ACM Multimedia '98*, Bristol, UK, September 1998.

[13] University College London. Multicast conferencing applications archive. Software available online, 2001. http://www-mice.cs.ucl.ac.uk/multimedia/software/.

[14] A. Mankin, L. Gharai, R. Riley, M. Perez Maher, and J. Flidr. The design of a digital amphitheater. In *Proc. NOSSDAV 2000*, Chapel Hill, NC, June 2000.

[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-time Applications. Internet Engineering Task Force, January 1996. RFC 1889.

[16] J. Touch and S. Hotz. The X-Bone. In *Proc. 3rd Global Internet Mini-Conference/Globecom '98*, Sydney, November 1998.

[17] International Telecommunication Union. Video codec for audiovisual services at Px64 kbits/s. ITU-T recommendation H.261, 1993.

[18] R. X. Xu, A. C. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous media applications. In *Proc. NOSSDAV '97*, Washington University in St. Louis, May 1997.