Scaling Video Conferencing through Spatial Tiling *

Ladan Gharai, Colin Perkins, Allison Mankin
USC / Information Sciences Institute

ABSTRACT

We describe an approach to scaling video conferencing, with the use of active agents. Such agents tile N video frames into one, by modification of their respective meta-data and adjustment of their video frame rate if necessary. The spatial tiling agents are located within a network, and participants in the session unicast video to the 'closest' agent. The agent then multicast the tiled video to the group of all participants. Results show that spatial tiling increases the ability of the end-user to receive large numbers of video streams and reduces network load both in terms of bandwidth and in packets per second. The result is a significant scaling boost to video conferencing systems.

1. INTRODUCTION

In recent years, video conferencing systems have become increasingly prevalent, both as a business tool and as an environment for social interaction. Unfortunately, many of these systems support only limited numbers of participants, and large scale video conferencing is still hampered by end-system and bandwidth limitations.

To alleviate these bottlenecks, a number of authors have proposed the use of active agents within the network, to adapt the multimedia flows to match the characteristics of the network and end system. We propose a novel type of active service – spatial tiling agents – to aggregate video flows.

Spatial tiling agents may be employed within a standard video conferencing session, or in conjunction with a special purpose application, such as our Digital Amphitheater [1] system. Each spatial tiling agent (STA) tiles N video streams to produce a single output, containing the N original frames. There are several benefits to this approach: the tiling process can reduce the per-packet overhead by tiling small flows into a single large flow, and the STA can act as a bandwidth gauge, reducing the overall bandwidth and packet rate of the streams it receives; in turn this reduces the workload of the receiver (i.e., it is easier to process one video frame, albeit a bigger frame, than N smaller frames). The result is a reduction in the load on the network and receiver, without reducing the quality of the video.

The remainder of this paper is structured as follows: we begin, in section 2, with a discussion of active services for video manipulation within a network, in section 3 we explain the spatial tiling process, and quantify improvements gained through spatial tiling in section 4. Finally, we provide suggestions for future work and conclusions.

2. RELATED WORK

The use of active service agents [2] to adapt the behaviour 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 focussed 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-organised transcoding of streaming audio/video media flows [3], leveraging from tools such as the video gateway developed at UC Berkeley [4]. More recently, implementations such as the 'FunnelWeb' Application Level Active Network [5], active routers [6], and overlay networks, such as the X-bone [7], 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 [8, 9, 10], particularly when related to reliable multicast, leading to recent standards work in the IETF [11]. We do not seek to

*This paper is based upon work supported by the Defense Advanced Research Projects Agency Information Technology Office. Any opinions, findings and 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 at {ladan,csp,mankin}@isi.edu.

NOSSDAV'01, June 25-26, 2001, Port Jefferson, New York, USA.

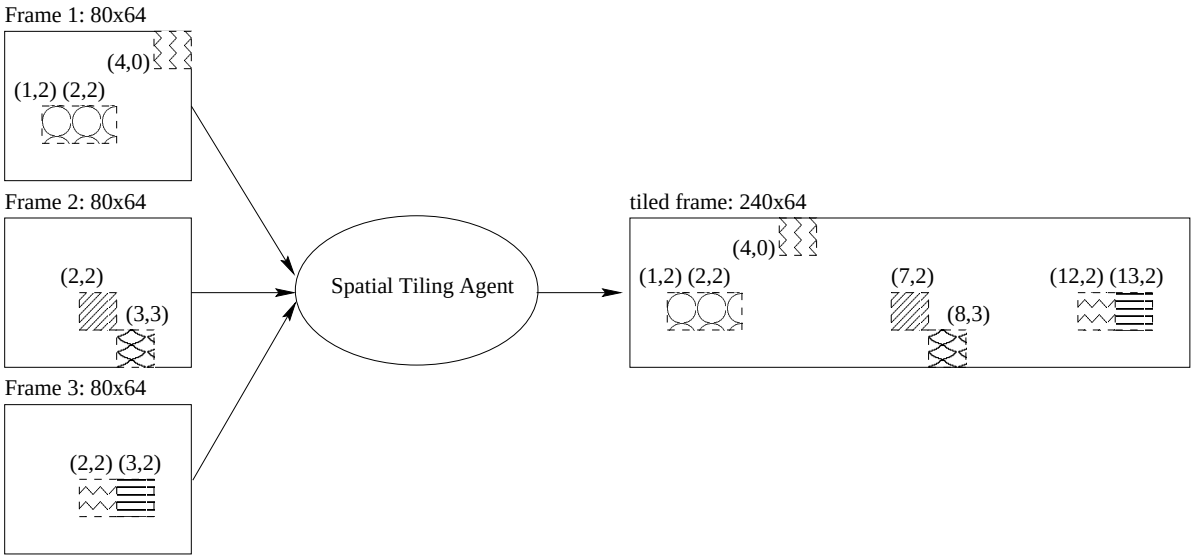


Figure 1: Tiling three frames into a single frame. Both frame size and block coordinates have been adjusted for the tiled frame.

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.

3. SPATIAL TILING: N FRAMES TO 1 FRAME

To illustrate the spatial tiling operation, consider the example in Figure 1: three streams of video are spatially tiled and represented as a single frame. The tiled frame consists of the three frames 'tiled' side by side, with each of the frames being completely represented. The meta-data for each of the frames, in this case the frame size and block coordinates, are adjusted accordingly. Frame size reflects the total frame size and block coordinates are transposed to the correct location.

To study the merits of spatial tiling we implemented STAs for two video representations: high bandwidth raw YUV video with conditional replenishment (YUVCR) [12] and H.261 [13] using only intra-frame compression. (Inter-frame spatial tiling is explored in section 3.3.) These two video formats have opposing characteristics: YUVCR is high bandwidth with little computational complexity, whereas H.261 is a relatively low bandwidth compression scheme, using discrete cosine transforms.

It is important that the STAs do not add additional delay to the video stream. In our implementation the 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 STAs. Each video stream is then processed within an STA as follows: First an incoming buffer collects all the packets for a given frame, this may be the entire frame or the last updated blocks of the frame. This frame is then added to the contents of the outgoing buffer for that video stream. This separation allows for independent frame rates between the various incoming streams and the outgoing tiled stream. At time intervals determined by the outgoing frame rate of the STAs, the tiled frame is constructed and packetized on the fly, from each of the individual output buffers, then the output buffers are cleared ready for the next packet.

At the receiver, any YUVCR decoder can receive and display a tiled YUVCR stream. However, for H.261, we have added an H.261 tiled decoder, H.261t, to the video conferencing tool vic [14, 15] in order to receive and decode a tiled H.261 stream. This was necessary as the tiled H.261 stream no longer complies with the standard H.261 stream. Both the standard H.261 syntax and the RTP payload headers have been slightly modified (see section 3.2).

Although, theorically, it is possible to tile an unlimited number of streams, we have restricted the STAs to 15 video streams. This restriction allows us to use the built in mixer functionality of RTP/RTCP [16], 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 STA 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 column.

In the following sections we discuss the tiling for each video format in more detail.

3.1 Tiling YUVCR

The simple structure of YUVCR is well suited to spatial tiling. Each video frame is divided into macro blocks of 16x16 pixels, represented in planar YUV format with a 4:2:0 color subsampling. The conditional replenishment algorithm insures that only updated video macro blocks are transmitted, and provides for some reduction in the data rate, in what is otherwise raw video. Unlike H.261, no meta-data related to the frame is carried in the frames themselves. The size of the video frames and the macro block coordinates are carried in an RTP payload header. For example, Figure 2 displays the RTP payload for frame 2 of our tiling example (Figure 1), where macro blocks (2,2) and (3,3) of

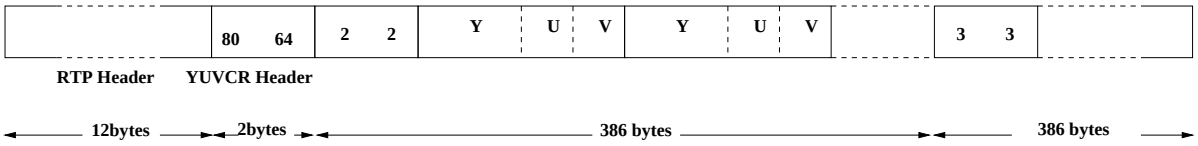


Figure 2: The RTP payload for Frame 2. Only updated blocks (2,2) and (3,3) are packetized. Each macro-block is 386 bytes long, with a 1500 mtu, an RTP packet can hold at most 3 macro-blocks.

the frame have been updated and must be transmitted. The RTP payload contains first the height and width of the video frame (80x64), followed by the coordinates and data of the two updated macro blocks. This structure not only makes for a high degree of flexibility in terms of frame sizes, up to 4096 pixels in each axis (the numbers are actually stored in multiples of 8), it also lends itself very well to spatial tiling.

Tiling YUVCR frames essentially consists of manipulating the size of the video frame in the RTP payload header and the coordinates of each 16x16 macro block such that it reflects the position of the macro block in the new tiled frame, much like the example in figure 1. As the information in the RTP payload is sufficient for tiled YUVCR, it is unnecessary to change the RTP payload, therefore the YUVCR decoder need not be altered either. At the receiving side, the YUVCR decoder is unchanged, it renders and displays the tiled frame in the usual manner.

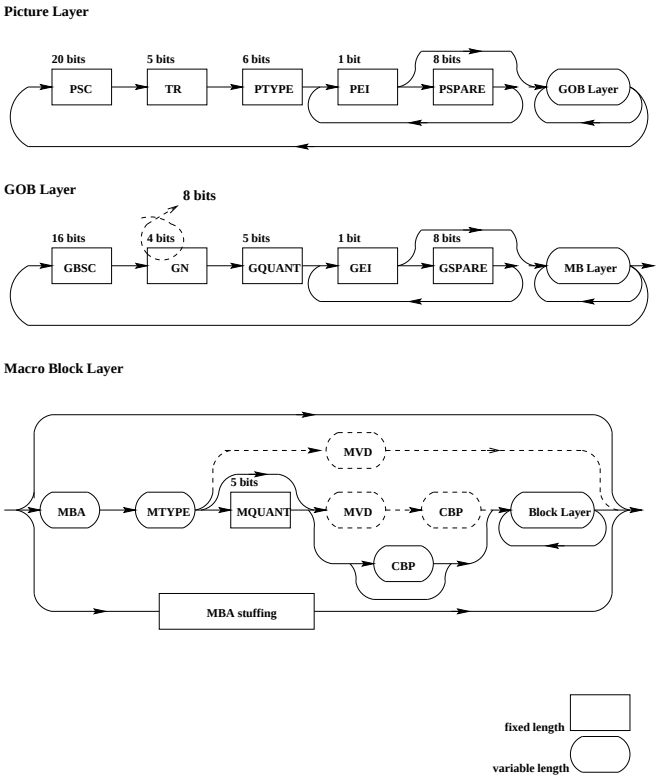


Figure 3: H.261 syntax diagram. The H.261t syntax is identical, except that GN in the GOB layer is extended to 8 bits.

3.2 Tiling Intra-frame H.261

Tiling H.261 frames is, in essence, the same as tiling YUVCR frames, since H.261 also divides each video frame into 16x16 macro-blocks, which are the smallest building blocks that the STAs process. However, tiling H.261 is somewhat complicated by intricate header system used to describe a frame and the use of Huffman encoding. Figure 3 shows the syntax diagram for an H.261 video frame. As is obvious from the diagram, extracting a macro-block requires manipulating non-byte aligned bit values and variable length fields. In this diagram following the solid lines in the macro block layer, produces the headers for intra-frame H.261.

An H.261 video frame consists of three layers: a picture layer, a group of block (GOB) layer and a macro block (MB) layer. Each GOB is divided into 33 macro-blocks, arranged in a 3x11 matrix. H.261 supports two scanning formats CIF and QCIF. A CIF frame contains 12 GOBs number consecutively from 1 to 12, whereas a QCIF frame contains 3 GOBs numbered 1, 3 and 5.

In the tiled H.261 frame, GOBs are numbered consecutively from 1 to Nx3 + Mx12, where N is the number of QCIF frames and M is the number of CIF frames tiled (currently both N and M cannot be non-zero, tiling of mixed CIF and QCIF frames planned for a future version). Since the standard H.261 GOB header only allocates 4 bits to the GOB Number (GN) field, it is necessary to extend this field to 8 bits in the tiled frame, so as to accommodate up to 15 CIF frames. GOB numbers are hence renumbered within the STAs prior to packetization. No changes are made to macro block headers: each macro block header is copied to the tiled frame as is. The tiled frame is preceded by a single picture header.

Overall, the primary changes made to the tiled H.261 frame, are replacing the individual picture headers by a single picture header and extending the GN field to 8 bits. For a tiled frame of 15, using a single picture header results in 56 bytes of savings (4 bytes per picture header) and the 4 bit increase in the GN field adds 12 bits per QCIF frame, or 22.5 bytes for a tiled frame of 15. All in all, 15 tiled QCIF frames save 33.5 bytes when compared with the non-tiled frames. For CIF frames the additional 4 bits per GOB, adds up to 6 bytes per frame. Therefore when 15 CIF frames are tiled, the tiled frame is 34 bytes larger than the individual non-tiled frames. However, this increase in ameliorated by the reduction in the number of packets and per packet overhead (40 bytes for each IP/UDP/RTP header) once the frame is packetized.

The increase in the size of the GOB number field must also be reflected in the standard RTP payload for H.261 [17]. As shown in figure 4, only 4 bits are allocated for GOBN in the RTP payload header. We define a new RTP payload header for tiled H.261 (figure 5) where GOBN is extended to 8 bits. To maintain the 4 byte size of the payload header,

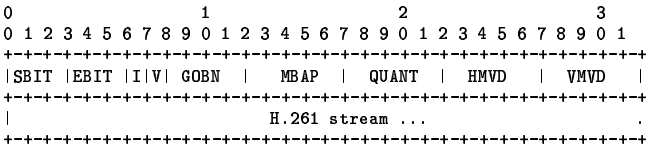


Figure 4: The standard H.261 header.

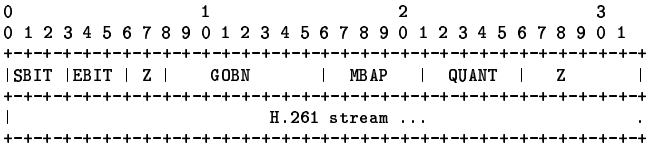


Figure 5: The current intra-frame H.261t payload headers. GOBN has been increased to a full byte to accomodate a fully tiled CIF frame.

for intra-frame H.261, we have eliminated the two motion vectors from the payload header (since our implementation does not currently support motion vectors).

Other information needed for by the H.261t decoder is the requested formation of the tiled frame, i.e., is the tiled frame a row of 15 or a block of 3x5? This information is signaled out of band to the STA and the number of tiled frames is extracted from the number of contributing sources. Thereby allowing the H.261t decoder to deduce the height and width of the tiled H.261t frame.

3.3 Tiling and Inter-Frame Codecs

Tiling inter-frame codecs can pose interesting challenges in terms of synchronization. Of course, in the best case scenario where the keyframe and intermediate frames of all incoming video streams are in sync, the STA will simply produce either keyframes or intermediate frames, based on the last frame it has received from all streams. However, it is unlikely that the incoming streams will always be in sync, further more, they may simply have different keyframe distributions. So what is an STA to do?

Figure 6 displays two possible solutions for when a tiling agent is confronted with a mixture of keyframes and intermediate frames. In the first solution, the STA simply produces two video frames at once. One frame tiled from the latest keyframes and an other from the latest intermediate frames. Although a simple solution, the frame rate of the tiled stream can become rather erratic, and could in the worse case be double that of the intended rate. The second solution avoids this pitfall, but is far more computationally intensive. As figure 6.b shows the tiling agent retains at all times the 'latest' frame for a video stream. This includes the last keyframe plus any updates since. When any number of actual keyframes must be tiled, the STA tiles the keyframes plus the update frames that it maintains for all streams.

Of course each technique has its own trade offs. The first solution increases the frame rate of the outgoing tiled video stream, although it does not increase bit-rate. Conversely the second solution increases bit-rate but maintains the frame rate of the outgoing stream, and is more resilient to packet loss. In addition this solution imposes additional computational overhead and buffering requirements on the

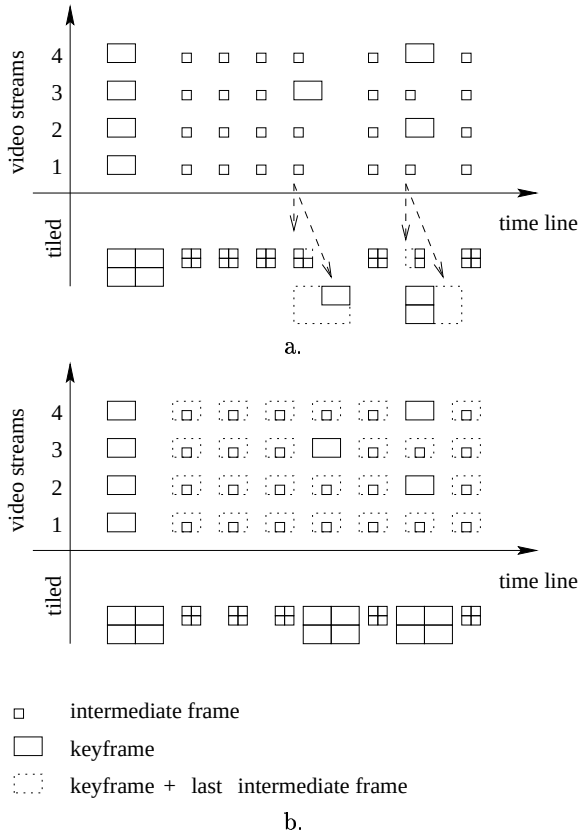


Figure 6: Two possible scenarios for tiling inter-frame codecs.

tiling agent. The appropriateness of each solution is application dependent.

3.4 Tiling Full H.261

In order to tile full frame H.261 the motion vectors, MVD, in the H.261 headers must also be processed. These motion vectors are computed relative to the macro-blocks, and only the position of the macro-block within the 3x11 matrix is relevant. As macro-block numbering in maintained within the tiled frame, tiling does not impact how motion vectors are processed.

In terms of the RTP payload header however, the motion vectors must be include in the payload. (We have eliminated this information in our current implementation). Figure 7 displays how we foresee the complete H.261t payload header to be. This particular structure was chosen to maximise the number of octet-aligned fields, at the expense of some small number of unused fields (labelled Z in the diagram).

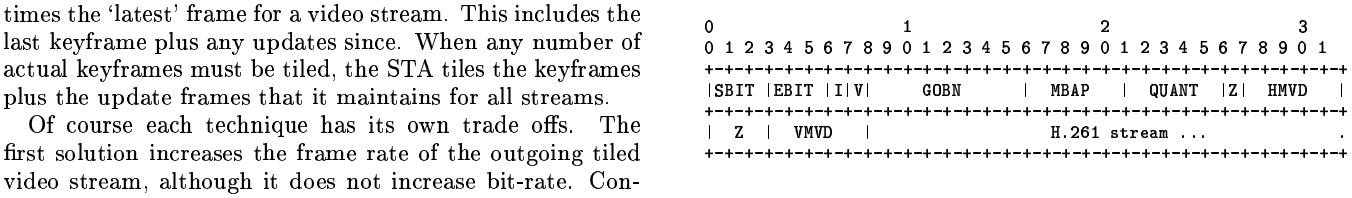


Figure 7: H.261t payload header with motion compensation.

	YUVCR			H.261		
Variable	seperate	tiled	gain%	seperate	tiled	gain%
pps	459	400	12.89%	186	122	34.51%
bps	3816	3717	2.59%	979	936	4.33%
N	42	57	26.3%	55	97	43.2%

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

4. PERFORMANCE MEASUREMENTS

The main goal of our performance measurements was to answer the following question: *What benefits are gained by using our STAs?* To identify potential performance gains 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. The test material comprised 15 YUVCR streams and 15 H.261 streams, all recorded at 8fps and 2 minutes long.

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 STAs were 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 to run multiple STAs, 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 vic. Next, we ran the test video through our STA with the output frame rate set to 8fps. To measure the bitrate, *bps*, and packet rate, *pps*, we instrumented vic such that it logged these variables, along with other decoding statistics, to a file.

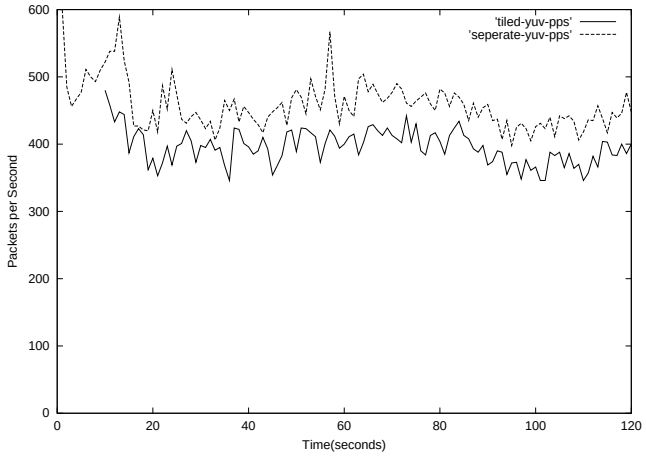
Figures 8-9 display the results of our tests for YUVCR and H.261, respectively. In all tests the tiled stream starts after a short delay, this delay is incurred while all individual streams reach the STA. All averages are computed from when the tiled stream commences. In these graphs, *pps* clearly show a reduction for the tiled streams of H.261 and YUVCR. Although not apparent, *bps*, also shows an overall reduction of 3 to 4 percent for the tiled streams over the entire test run. Table 1 summarizes these results.

The reduction in *pps* is primarily due to the aggregation of smaller packets. The STA generates a single large frame, therefore there are fewer 'half empty' packets in the resulting stream. In the tiled YUVCR stream *pps* is reduced by approximately 13% and in the H.261 tiled stream *pps* is reduced by approximately 35%. The higher reduction of *pps* in the H.261 tiled stream in our trials is the result of the more flexible nature of H.261 and its compression scheme. H.261 UDP packets, range anywhere in size from 48 bytes to 1024 bytes. A YUVCR packet, on the other hand, can only hold one, two or three macro blocks, which results in UDP packets sizes of 400 bytes, 786 bytes and 1172 bytes. This means there are less options on how to aggregate packets for the YUVCR tiled stream, whilst remaining within the 1500 byte Ethernet MTU, which results in a lower reduction of *pps*.

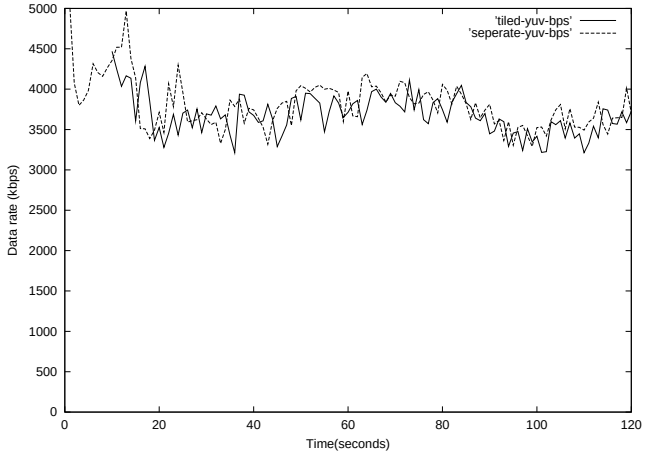
In terms of bandwidth, *bps*, the tiled YUVCR stream is reduced by an average of 3% and the H.261 stream by 4%. Although bandwidth is reduced over the duration of the test runs, the graphs reveal that this in not the case on a per minute bases, as in some instances the *bps* of the separate streams appears 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 STA reproduces the input video streams, exactly as they come in, without any temporal or spatial down sampling. Both the input and output frame rates of the STAs are 8 fps and the STAs more or less copy each incoming frame to the outgoing tiled frame. There are some instances where incoming frames are not pushed out by the STA due to synchronization variability between incoming and outgoing frame rates, this can results in a dropped frame. However as our data shows this does not happen often. 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 the STAs. 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. For YUVCR, our end-system was able to receive 42 individual streams without loss, while tiled, the end-system could process 3 fully tiled streams and one tiled stream of 12, a total of 57 streams.

These numbers clearly demonstrate the reduction of workload on the end-system due to the spatial tiling process. For our H.261 streams the end-systems is capable of receiving almost twice as many video streams, once the video streams are tiled and for YUVCR the number of streams is increased by about 25%. The higher increase in N for H.261 is in part a reflection of the greater reduction in *pps* for the H.261 streams. In fact in all our tests H.261 fared better than

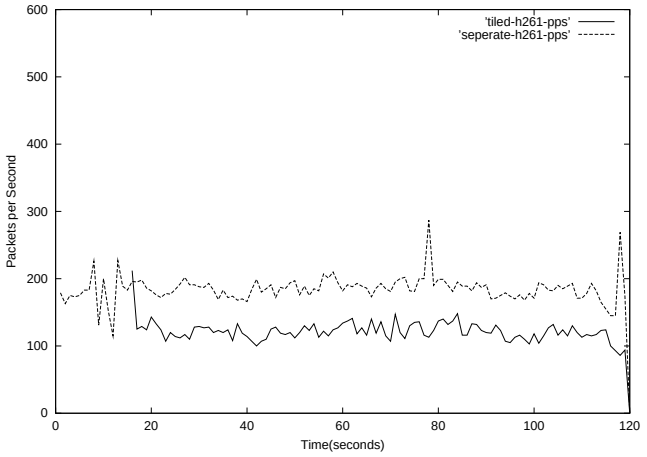


a.

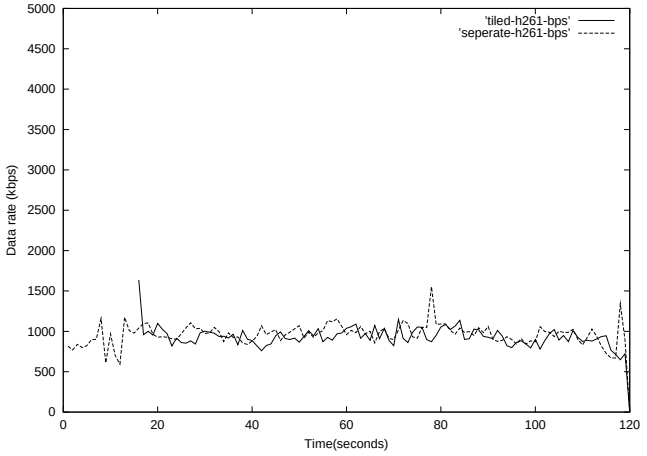


b.

Figure 8: Comparison of 15 separate YUVCR 80x64 video streams and the equivalent 5x3 (or 400x192) tiled stream (a) packets per second (b) bits per second.



a.



b.

Figure 9: Comparison of 15 separate H261 QCIF video streams and the equivalent 5x3 tiled stream (a) packets per second (b) bits per second.

YUVCR. Although YUVCR requires no processing time, its higher data rates and non-flexible packetizing scheme, seem to make it an unsuitable candidate for spatial tiling.

This leads us to conclude that a primary load on end-systems is per packet interrupt processing 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 STAs more leverage in reducing the overall packet rate.

5. FUTURE WORK

The initial results obtained from our experiments with spatial tiling have been very promising. We plan to extend the work in three directions:

Codec Support: As a first step we plan to complete the implementation of H.261 to include support for full H.261 with inter-frame compression. Second, we plan to expand the collection of codecs the tiling agents can support to include H.263 and MPEG. These pose more of a challenge to the tiling agents, due to the considerable emphesis on inter-frame compression and possibility of back-channel feedback.

Automatic Adaptation: The STAs are in a unique position to perform congestion control, and to adapt the streams according to the reception quality reports that are generated by RTP receivers. A number of congestion control algorithms have been proposed for streaming media (e.g. [18]), and we plan to experiment to see if these are suitable for a system with the adaptation range of the STAs.

Automatic STA Placement: As noted in section 2, there has been considerable work on the problem of automatic placement of agents, for reliable multicast and congestion control. This is not our main focus, but we will consider implementing such a system as part of the Digital Amphitheater [1].

6. CONCLUSION

Spatial tiling is a unique approach to scaling video conferencing, where N frames are tiled into a single video frame, and the relevant meta-data is adjusted accordingly. We hypothesized that receiving a single video stream instead of N separate video streams, would reduce the workload on end-systems. We tested this hypothesis with two varying video representations: YUVCR and H.261. Our tests confirmed that spatial tiling reduces network load in terms of bandwidth consumed and packet rate, and increases the number of streams an end-system can receive. We conclude that spatial tiling can potentially become an invaluable tool for large scale video conferencing.

7. REFERENCES

[1] 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.

[2] 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.

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

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

[5] M. Fry and A. Ghosh, "Application level active networking," *Computer Networks*, vol. 31, no. 7, pp. 655–667, July 1999.

[6] 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.

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

[8] 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.

[9] 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.

[10] P. Francis, "Yallcast: Extending the Internet multicast architecture," Tech. Rep., NTT Information Sharing Platform Laboratories, September 1999.

[11] 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.

[12] M. Handley, "YUV-CR codec for vic," Personal correspondance.

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

[14] S. McCanne and V. Jacobson, "vic: A flexible framework for packet video," in *Proc. ACM Multimedia'95*, San Franscisco, November 1995.

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

[16] 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.

[17] T. Turletti and C. Huitema, "RTP payload format for H.261 video streams," IETF, October 1996, RFC 2032.

[18] R. Rejaie, M. Handley, and D. Estrin, "RAP: an end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proc. IEEE Infocom '99*, New York, March 1999.