

Figure 7: Zonal bit-plane coding.

Competition mode

In practice, we have observed that although the two depicted algorithms give encouraging results, none is better than the other in all conditions and for all sequences. More precisely, their respective artifacts are very dissimilar and are located in regions with different properties. The modified Intra mode tends to blur the picture and to lose details in very textured areas. On the contrary, the scalable solution gives a sharper rendering in edges and rich textures but generates an unpleasing noise/ringing effect in smoother regions, due to the bit-plane truncation. Consequently, our solution is based on the competition between these two modes, to achieve the best performance. In order to limit complexity, we select the encoding mode through an *a priori* criterion: if the activity (or energy) of the current block is higher than a given threshold, then the scalable scheme is chosen, otherwise the modified H.264 Intra is applied. *A priori* selection is important because it generates a very limited computational overhead compared to an *a posteriori* solution where both schemes are processed before one gets selected.

4. EXPERIMENTAL RESULTS

We present our results on several dimensions: the memory aspects (size and transfers), the complexity issue, the objective and the subjective quality. Our test set is composed of CIF sequences at 25 frames/s with various contents as given in Table 1. We have encoded them with the JM6 reference software of H.264, set to respect the Baseline Profile, with an Intra period of 25 frames. Bit-rates are in the range [150…512] kb/s. This spatio-temporal resolution, these sequences and the encoding conditions were chosen as representative of the targeted applications.

Foreman	Face close-up, moderate motion.
Flower	Slow camera motion, rich-textured garden, cloudy sky.
Silent	Talking person moving hands and arms, painted background.
Stefan	Tennis scene, fast motions.
Weather	Weather forecast, synthetic map (background), talking person (foreground).

Table 1: Test sequences.

4.1. Memory gain

Our primary goal is to reduce power consumption through a decrease of the memory transfers. We have seen that we apply a Compression Ratio CR of 3 so that an 8x8 block maps a data burst of 256 bits. The first consequence is that the memory occupancy of the reference frame(s) is divided by three. Hence a smaller memory can be implemented, which is very positive in terms of both costs and consumption. Indeed, smaller memories can be embedded closer to the CPU, leading to less cable dissipation during accesses. Concerning transfers (and thus bandwidth), the gain is not so direct due to overhead (see paragraph 2.4). In the following, we describe how we measure it.

Overhead: statistical prediction

The memory accesses are divided by CR/Ovh at the reading side, where Ovh is the overhead. At the storing side, this overhead issue is irrelevant and transfers are thus divided by CR. Equation 1 expresses the global gain in memory transfers (load/store) with the approximation that there are as many loads as stores in the standard decoder. One can see that overhead can theoretically counter-balance the benefits of EC regarding memory transfers. We have to measure it precisely to assess the memory bandwidth that is actually saved.

Google Exhibit No. 1029
Page 007

Google Exhibit No. 1063
Page 1099

$$Transfer\_Gain = \frac{Memory\_Store_{EC} + Memory\_Load_{EC}}{Memory\_Store_{Std} + Memory\_Load_{Std}} = \frac{1 + Ovh}{2 * CR} \qquad \text{Equation 1}$$

Overhead strictly depends on the block size of the EC, on the size of the current motion-compensated block, and on the value of its motion vector. It is a part of our contribution to quantify their impact through a statistical analysis of the sequences of the test set.

There are four main inter prediction modes for luminance pixels in H.264: 16x16, 16x8, 8x16 and 8x8. The 8x8 mode can be further decomposed into 4x4, 8x4 and 4x8 partitions. For each mode and sub-mode, we have counted the occurences when a reference block is aligned with the chosen EC block grid (8x8). If it is not, then there is a separation between horizontal misalignment, vertical misalignment, and both. In each case, the number of encoded blocks to retrieve (and thus the number of pixels to get) is different. Then the statistical apparition of each mode is also computed. A synthesis permits to extract the average overhead from our test set.

Figure 8 illustrates a typical case, here for the Stefan sequence, where we also indicate the equivalent results for an EC block-grid of 4x4 and 16x16. A simple analysis shows that the result is qualitatively coherent. The 4x4 grid generates fewer overheads than the other ones because it is closer to pixel accuracy. Furthermore, for all grid sizes, overheads increase with the bit-rate. This is due to the H.264 encoder that allows macroblock partitioning and large motion vectors at high rates (small quantization step) while it favors the null vector with a 16x16 partition at low rates.

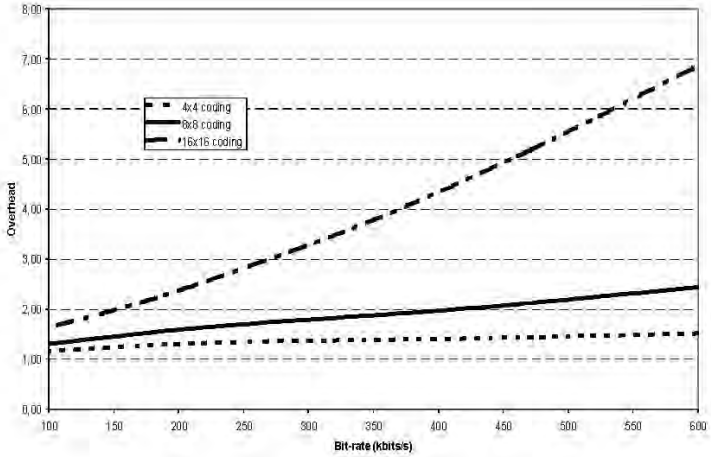


Figure 8: overhead vs bit-rate (Stefan sequence).

Sequence	4x4 EC Grid	8x8 EC Grid	16x16 EC Grid
Foreman	1.31	1.77	3.69
Flower	1.30	1.74	3.77
News	1.14	1.51	2.78
Silent	1.17	1.50	3.22
Stefan	1.51	2.44	6.95
Weather	1.17	1.49	3.18
All	**1.27**	**1.73**	**3.93**

Table 2: overhead vs EC block-grid for each sequence.

Table 2 summarizes the statistical analysis for each sequence of the test set. Not surprisingly, still sequences generate the smallest overhead values (News, Weather) because the motion vector is often equal to zero. On the contrary, a fast sequence like Stefan generates much overhead. In the remaining of the experiments we only present results for the 8x8 EC block grid, which turned out to offer the best trade-off between transfer overhead (that is minimized if the block grid is small) and coding efficiency (that prefers larger blocks).

Memory transfers

Thanks to the formula of Equation 1 and to the statistical results of Table 2, we can derive the average gain in memory transfers for each sequence. For the 8x8 grid, the transfers gain lies between 0.40 for the Weather sequence and 0.55 for the Stefan sequence, respectively corresponding to a 60% and a 45% reduction. The average gain for all the sequences is about 0.45 (or 55% of reduction), corresponding to a gain of 0.33 for the storing and 0.57 for the reading of a reference frame. These figures correspond to a Compression Ratio (CR) of 3 and lead to very good visual results as we

Google Exhibit No. 1029
Page 008

Google Exhibit No. 1063
Page 1100

will see in section 4.4. Higher CR values, and thus higher gains, can be achieved. For instance CR=8 implies an average reduction of 80% for memory transfers at the expense of more visible degradations.

4.2. Complexity evaluation (CFDG)

Although memory transfers represent the major part of the energy dissipation in modern architecture, computational and logical operations are not negligible. By applying Embedded Compression, we have added some modules to the conventional standard decoder, and consequently increased the complexity. In this paragraph we estimate the number of added operations through a CFDG analysis.
A Control Flow Graph (CFG) is an abstract data structure, a representation of a procedure or program. It consists of nodes performing operations and exchanging data. Each node in the graph represents a basic block, i.e. a straight-line piece of code without any jump. A CFDG is a CFG where Data dependencies are highlighted. In general, CF(D)G are internally built-up by compilers to optimize binary executables. In our case, we found it was a good tool to precisely decompose an algorithm at a high-level. It allows analyzing its complexity, memory accesses, level of data dependencies and degree of parallelism. All these notions directly give indications on the ease to implement a low-power system based on the studied algorithm.
The CFDG methodology has been used to evaluate the average complexity of the reference MC and of the proposed solution with EC. The results given in Table 3 represent the number of additions (Add), multiplex (Mux, related to 'if' statements) and multiplications (Mul) for performing the MC of one frame. It takes into account statistics about the motion vectors (including sub-pixel interpolation). The numbers of memory read and write are taken from the study of the previous paragraph, and also take into account the storing and reading of the motion vectors from the previous neighboring blocks (for motion vector decoding).

	Read (x1000)	Write (x1000)	Add (x1000)	Mux (x1000)	Mul (x1000)
MC of the H.264 decoder	226	152	709	1,060	200
Proposed MC with Embedded Compression	138	53	5,017	1,938	504
Ratio	0.61	0.35	7.00	1.72	2.52

Table 3: Complexity of the proposed MC compared to the standard one.

One can notice that EC performs a lot of additions: they are multiplied by 7 compared to the conventional MC. However, additions are less costly than multiplications. In general, one considers that one multiplication is three times as complex (and power-consuming) as an addition and a multiplex in most processing units. This approximation leads to an average ratio of 3.57 more operations (Add, Mux, Mul) for the proposed method. Finally, some studies estimate that the Motion Compensation represents around 40% of the total complexity of a Baseline H.264 decoder. Hence, our solution is globally between 1.5 and 2 times as complex as the standard one, while dividing the memory transfers by more than 2.

4.3. Objective quality (PSNR)

Embedded Compression generates an unavoidable reconstruction drift at the decoder, which impacts the resulting quality. However, in the introduction we have stressed the fact that we judge our algorithms on a power-rate-distortion basis, meaning that a large gain in power can justify a slight increase in distortion. We have to quantify this distortion to assess the acceptability of our solution.
Figure 9 and Figure 10 depict the rate-distortion graphs of the standard decoder and of the proposed one, for two representative sequences of the test set (Flower and Foreman). The gap between the two methods increases with the bit-rate because the decoded frames are better preserved and contain more details at higher rates, implying a non-negligible loss of data during the Embedded Compression. On the contrary, the decoded pictures at lower rates are easier to encode by the EC module because they are already degraded anyway.
The average PSNR loss on all the sequences of the test set is 0.7 dB. It is null at the low end of the bit-rate range and can reach more than 2.0 dB around 512 kbits/s. However, some authors have remarked that the visual impact of the EC is not necessarily as high as the gaps in the rate-distortion curves[14].

Google Exhibit No. 1029
Page 009

Google Exhibit No. 1063
Page 1101

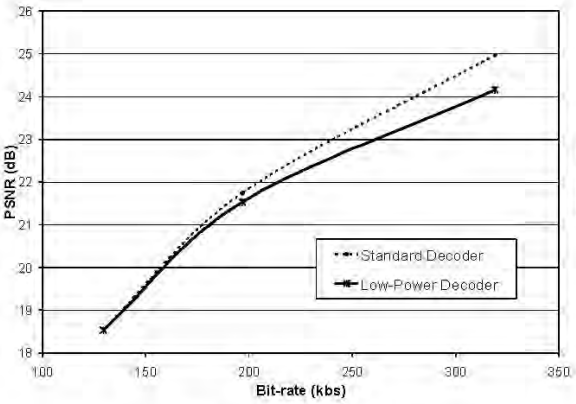


Figure 9: Flower sequence.

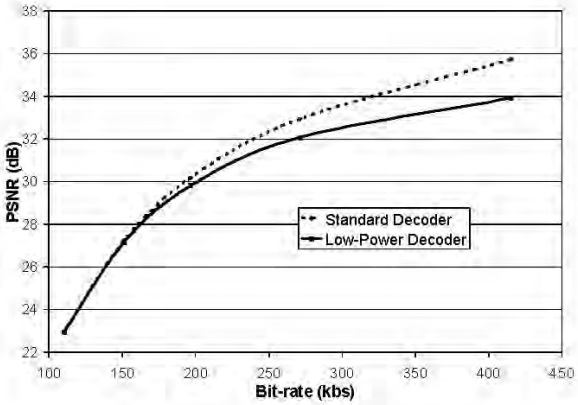


Figure 10: Foreman sequence.

4.4. Subjective quality (visual tests)

Rate-distortion graphs are a useful way of evaluating coding efficiency, especially during the process of designing and tuning algorithms. Yet, PSNR curves do not give a direct idea about the actual differences in visual quality between two solutions and can even be misleading in some special cases. Moreover, the subjective quality that viewers feel while looking at a sequence is much influenced by the viewing conditions. For instance the visibility of some artifacts (blocking, ringing, temporal drift, etc.) varies much according to whether people look at the palm-sized LCD screen of a PDA or at a professional monitor in a laboratory environment. Subjective tests are thus needed. Our solution targets mobile applications, so we performed the subjective tests on a portable device, following as much as possible the known protocols and recommendations proposed for television applications[17].
Tests were designed for Double Stimulus Impairment Scale (DSIS). Each test consists of a pair of stimuli: one sequence processed with the reference solution (in our case the standard decoder) and with the proposed one. Each pair is shown in this order, twice. Let us notice that the reference is not necessarily "artifact-free": typical coding artifacts (blocking, ringing, mosquito noise, lack of sharpness, temporal inconsistency) can be present. The viewers are asked to score the impairment level of the proposed technique regarding this reference. Table 4 shows the scoring scale.

Score	Impairment
5	Imperceptible
4	Perceptible but not annoying
3	Slightly annoying
2	Annoying
1	Very annoying

Table 4: Scoring scale.

The device is a mobile video player with a 320x240 LCD-TFT screen and a depth of 16 bits/pixel. The viewing distance is not imposed. Both expert and naïve viewers are asked to hold the mobile device at a convenient distance and to keep this distance constant during the test. The room illumination is low. Finally, the Mean Opinion Score (MOS) is computed. Results are summarized in Table 5.

Category	Number of viewers	Mean Opinion Score	Variance	Min Score	Max Score
Experts	9	**4.0**	0.4	2	5
Naïves	5	**4.3**	0.5	3	5

Table 5: Subjective Tests results.

Google Exhibit No. 1029
Page 010

Google Exhibit No. 1063
Page 1102

As usual, expert viewers are both more sensitive to artifacts and more consistent in their judgment than naïve viewers, leading to a smaller MOS and a smaller variance. However, the scores are rather coherent across the two categories, and the impairment level of the low-power solution is not considered annoying although it is perceptible. The proposed scheme is thus only 1 JND (Just Noticeable Difference[18]) away from the reference decoder, which is an excellent result.

4.5. General interpretation

In this section, the proposed low-power H.264 decoder has been evaluated regarding memory (size and transfers), complexity, objective and subjective quality. On the one hand, the cost of Embedded Compression is quite high in terms of computational operations compared to the standard Motion Compensation, leading to an estimated 1.7 complexity factor for the whole decoder. On the other hand, memory transfers, which are predominant in the global energy dissipation, have been divided by more than 2. As stated in section 2.2, this undoubtedly leads to a global gain in power consumption, especially in a hardware-only or a mixed software/hardware environment. It is difficult to quantitatively evaluate this gain: several tools exist[5], using various consumption models, but they need the implementation process to be well advanced. Finally, subjective tests proved that the proposed low-power decoder provides video sequences that are visually very close to the standard decoder. This scheme thus provides an excellent trade-off between power consumption and visual quality. It is possible to achieve larger gains by increasing the Compression Ratio of the EC at the expense of a degraded visual quality, which can be acceptable in some applications.

5. CONCLUSIONS

We have presented a modified H.264 decoder that gracefully degrades the output video quality in order to significantly reduce energy dissipation. We have first observed that the Motion Compensation was the most power-consuming part of the decoder, due to its numerous accesses to the frame memory. Embedded Compression (EC) was then identified as a promising technique to reduce these transfers. In order to limit the data overhead that is inherent to EC, we have imposed some requirements that help mapping the EC grid with the memory structure. Once these constraints have been set, we have applied a compression factor of 3 on the reference frames, using a combination of two complementary EC techniques. The selection criterion is *a priori* and low-cost.

Experimental results show that memory transfers, which dominate power consumption, are reduced by 55% in average for CIF sequences in mobile application conditions. The computational complexity of the decoder is globally increased by 1.7, which should not counter-balance the power gain due to memory transfers. Furthermore, subjective tests proved that our solution maintains the visual quality of a standard decoder. The proposed decoder thus provides an excellent trade-off between power consumption and visual quality, making it an advantageous solution for mobile applications.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Yi Li and Carolina Miro, from the Platform & Integration group of the Philips Digital Systems Labs in Paris, for their work on the evaluation of the complexity of the different decoders used in this paper.

REFERENCES

1. P. van der Wolf , P. Lieverse , M. Goel , D. La Hei , K. Vissers, "An MPEG-2 decoder case study as a driver for a system level design methodology", Proc. of the 7th Int. Workshop on Hardware/Software Codesign, pp.33-37, Roma, Italy, March 1999.
2. C.W. Yoon *et al.*, "An 80/20 MHz 160 mW multimedia processor integrated with embedded DRAM, MPEG-4 accelerator and 3D rendering engine for mobile applications", IEEE Journal of Solid-State Circuits, Vol. 36, Issue 11, pp. 1758–1767, November 2001.
3. L. Nachtergaele, F. Catthoor, B. Kapoor, S. Janssens and D. Moolenaar, "Low-Power Data Transfer and Exploration for H.263 Video Decoder System", IEEE Journal on Selected Areas in Communications, Vol. 16, No. 1, January 1998.
4. W. Badawy and M. Bayoumi, "Algorithm-based Low-Power VLSI Architecture for 2D-Mesh Video-Object Motion Tracking", IEEE Trans. On Circuits and Systems for Video Technology, Vol. 12, April 2002.
5. A. Chandrakasan and W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits", Proc. Of the IEEE, Vol. 83, April 1995.

Google Exhibit No. 1029
Page 011

Google Exhibit No. 1063
Page 1103

6. J. Jung and A. Bourge, "Power Scalable Video Encoder for Mobile Devices based on Collocated Motion Estimation", Proc. of VCIP, Electronic Imaging, San Jose, California, USA, January 2004.
7. B. Kapoor, "Low Power Memory Architectures for Video Applications", Proceedings of the 8th Great Lakes Symposium on VLSI, pp.2-7, Lafayette, LA, USA, 19-21 February 1998.
8. T.H. Meng, A.C. Hung, E.K. Tsern and B.M. Gordon, "Low-Power Signal Processing System for Wireless Applications", IEEE Personal Communications (USA), vol.5, no.3, pp.20-31, June 1998.
9. K. Masselos, P. Merakos, T. Stouraitis and C.E. Goutis, "A Novel Algorithm for Low-Power Image and Video Coding", IEEE Trans. On CSVT, Vol.8, No.3, pp.258- 263, June 1998.
10. N. Chaddha and M. Vishwanath, "A Low Power Video Encoder with Power, Memory and Bandwidth Scalability", 9th Int. Conf. On VLSI Design, pp. 358-263, January 1996.
11. W. Namgoong, N. Chaddha and T. Meng, "Low-Power Video Encoder/Decoder using Wavelets/TSVQ with Conditional Replenishment", ICASSP'96, Vol.6, pp. 3240-3243, May 1996.
12. Y. Chen, Z. Zhong, T.H. Lan, S. Peng and K. van Zon, "Regulated Complexity-Scalable MPEG-2 video decoding for Media Processors", IEEE Trans. On Circuits and Systems for Video Technology, Vol. 12, No. 8, August 2002.
13. E. Jaspers and P.H.N. de With, "Bandwidth reduction for video processing in consumer systems", IEEE Trans. On Consumer Electronics, Vol. 47, No. 4, pp. 885-894, November 2001.
14. E. Jaspers and P.H.N. de With, "Embedded compression for memory resource reduction in MPEG systems", Proc. Of 3rd Benelux IEEE Signal Processing Symposium, Leuven, Belgium, March 2002.
15. P.H.N. de With, P.H. Frencken and M. van der Schaar, "An MPEG decoder with embedded compression for memory reduction", International Conference on Consumer Electronics, Digest of Technical Papers, pp. 60-62, June 1998.
16. R.J. van der Vleuten, "Low-complexity lossless and fine-granularity scalable near-lossless compression of color images", Proceedings of the Data Compression Conference, pp.477, April 2002.
17. "Methodology for the subjective assessment of the quality of television pictures", ITU-R BT-500.10, updated in 2000.
18. J. Lubin and D. Fibush, "Sarnof JND vision model", T1A1.5 Working Group Doc. No. 97-612, T1 Standards Committee, 1997.

Google Exhibit No. 1029
Page 012

Google Exhibit No. 1063
Page 1104

Ex. 1029A

Low-power H.264 video decoder with graceful degradation

Arnaud Bourge*, Joël Jung*
Philips Digital Systems Laboratories - Paris
51, rue Carnot, BP301, 92156 Suresnes Cedex, France

ABSTRACT

In this paper, we present a novel H.264 video decoder where memory transfers and energy consumption are significantly reduced. A power-friendly solution is indeed required in mobile applications, where autonomy is a key feature. Whereas low-power design is usually regarded as a pure architectural and implementation problem, our solution is based on an algorithmic approach. We first observe that in modern hardware architectures, computational complexity plays a second role in terms of energy dissipation, which is dominated by memory transfers. We thus identify the motion compensation module as the most consuming part of the standard H.264 decoder, due to its numerous memory accesses to the reference frame(s). Through an algorithmic modification that uses concurrent embedded compression techniques and relies on the data structure of the memory, we reduce memory transfers, and hence power dissipation, while maintaining the visual quality. Experimental results prove that the new method is close to the reference H.264 baseline decoder both in terms of objective and subjective measurements. In the meantime, memory transfers have been reduced by 55% in average, implying power savings which lengthen the battery life of the mobile device, increase the reliability of the chip and lower production costs.

Keywords: Low Power, Battery life, H.264, Video Decoding, Memory Transfers, Complexity, Motion Compensation, Embedded Compression, Subjective Quality.

1. INTRODUCTION

The evolution of mobile devices shows a fast integration of multimedia functionalities. Today the capture and display of color images at reasonable resolutions are common features in cellular phones. Video is already present in some products (high-end mobile phones, PDAs) and is expected to spread with the emergence of the MMS (Multimedia Messaging System). However, this integration comes at the expense of increasing power consumption. Indeed, mobile devices providing video-decoding capabilities suffer from a shortened battery life, whereas autonomy remains a key-feature in the mobile world. Advances in battery capacities only partially answer this problem, because the complexity of codecs and the requirements in picture size and frame-rate keep increasing too. Directly reducing the energy dissipation of a video decoding circuit is thus crucial. It does not only extend battery life, but also increases the reliability of the chip. Furthermore, power saving allows a significant attenuation of the cooling effort, which simplifies the hardware management and lowers production costs.

Low-power design is usually regarded as a pure architectural problem. For instance, hardware/software partitioning allows an improved management of the power consumption[1] by allocating the most consuming parts to dedicated components. Embedded memories also help because they are located close to the CPU and co-processor, so an access to their data goes through less cable and dissipates less energy[2]. Finally, optimizing the implementation of a given algorithm for a specific platform is primordial[3]. In this classic approach, however, power matters are considered at the latest stage and result in a non-optimal solution.

On the contrary, we take the energy dissipation into account at the algorithmic stage. This approach, if combined with architectural considerations, leads to the most promising gains[4] because some algorithms intrinsically have the property to enable power-efficient implementations[5]. The counterpart is that evaluating the actual power consumption at the algorithmic stage is a tricky task that can only result in qualitative estimations. In practice, we translate consumption properties of a generic platform into guidelines and constraints for the design of algorithms[6]. First of all, low power should not be regarded as a mere complexity problem. Today, power consumption is led by memory transfers, and especially by accesses to an external memory. Computations only represent a minor (though not negligible) portion of

* {arnaud.bourge ; joel.jung }@philips.com

Visual Communications and Image Processing 2004, edited by Sethuraman Panchanathan, Bhaskaran Vasudev, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 5308
© 2004 SPIE and IS&T · 0277-786X/04/$15

Google Exhibit No. 1063
Page 1106

the overall dissipation[7]. As a consequence, a low-power video-decoding algorithm should mainly minimize the number of accesses to the memory, even at the expense of a moderate complexity increase[8]. Hence, we do not measure performance through a classical rate-distortion trade-off, but through a power-rate-distortion criterion. It means that a loss in visual quality can be compensated by a gain in power consumption.

We have chosen to start from the H.264 decoding scheme - that is the state-of-the-art standard in terms of coding efficiency - at the Baseline Profile, which is intended to be used in the mobile world. In this paper, we present a novel H.264 video decoder where memory transfers and power consumption are significantly reduced. Section 2 is a preliminary study. We present the state-of-the-art in low-power video decoding, show that the motion compensation is the most costly module, explain why we choose to apply embedded compression, and highlight some difficulties inherent to this kind of scheme. In Section 3 we define our strategy, which consists in imposing some specific constraints to the embedded compression, and we describe our algorithmic solution. Section 4 is dedicated to the experimental results, where we show that the memory size can be reduced by a factor of 3, and the associated memory transfers by a factor of 2.3 while maintaining the visual quality. Results also include complexity evaluations, rate-distortion curves and subjective quality assessments. Section 5 concludes this paper.

2. PRELIMINARY STUDY

2.1. State of the art

We have seen in the introduction some examples of low-power decoder designs based on architectural optimizations[1,2,3]. Although this is the most common approach, our method is more focused on the algorithmic side. Following the same principle, authors have built their decompression algorithm based on underlying hardware properties to minimize power dissipation. Most of them propose computational simplifications, but this method is not sufficient anymore to reduce energy consumption. Others put the strain on minimizing data transfers, both spatially and temporally. In the spatial domain, isometries were thus proposed as a way to significantly reduced a vector quantization (VQ) codebook[9]. Moreover in [10], "power scalability", depicted as the ability for the codec to change its power consumption depending on the available bandwidth and on the required video quality, is introduced through the use of a hierarchical VQ. Yet we will see that the spatial domain is not the most critical path regarding memory transfers in a conventional decoder. In the temporal domain, the Motion Estimation / Motion Compensation can be replaced by a so-called Conditional Replenishment[11] but at the cost of compression performance. Alternatively, an efficient trade-off between data transfers and coding efficiency is introduced in [6] with the notion of "Collocated Motion Estimation".
All the above solutions imply the use of a non-conventional encoder to generate the incoming bit-stream. They are not applicable in a context where interoperability and exchanges impose the use of an accepted standard, which is generally the case in mobile communications. Strictly speaking, one cannot come up with a completely new scheme only for the decoder: the corresponding encoding scheme must be accepted too. However, it is possible to decode a standard bit-stream in a manner that differs from the standard algorithm. The resulting sequence is necessarily degraded, but a sound control of the modifications can keep the loss small while some costly operations are saved. In fact this notion of "graceful degradation" introduces a trade-off between the quality of the reconstruction and the cost of the decoding[12]. Each modification of the standard scheme introduces a deviation that impacts the visual result. The goal is then to find modifications that maximize the gain in complexity while minimizing the visual losses. Finally the algorithm can adapt itself to the available resources: depending on the CPU utilization, some operations are done, simplified or even skipped. For instance the Inverse DCT can be truncated, or the sub-pixel interpolation filters of the motion compensation can be approximated.

2.2. Identification of the power-costly module: Motion Compensation

We have seen that in the literature "graceful degradation" is commonly applied to computational complexity. We extend this notion to power consumption by adapting the decoding scheme to the available energy resources, thus introducing power-related graceful degradation. We start with the identification of the power-consuming part of the H.264 decoder, which we will then modify.
Memory transfers, and especially accesses to an external memory, have a major influence on power consumption[8]. Computations (arithmetic and logical operations) still represent a significant portion of the overall dissipation, but there

Google Exhibit No. 1063
Page 1107

is a trend towards a reduction of its relative weight[7]. The design of low-power algorithms is thus primarily a matter of minimizing memory transfers, which is closely related to bandwidth reduction between the memory and the processing modules. This is especially true in hardware-only and in mixed hardware/software solutions. In software-only environments, however, the power repartition between memory transfers and computations is more balanced and difficult to measure.

In the scope of an H.264 video decoder, the Motion Compensation consumes most of the memory transfers. This is because the other parts of the decoder can generally be performed locally, on a block-by-block basis. In this paper, the term "Motion Compensation" (MC) defines all the operations (logical, arithmetic and memory) that are part of the feedback loop of the predictive scheme, as shown in Figure 1. It includes storing the decoded frame to be used as a reference for the next frame(s), calculating the Motion Vectors (MV), reading the memory to get the reference pixels, and finally performing the interpolation filters that generate the sub-pixel grid.

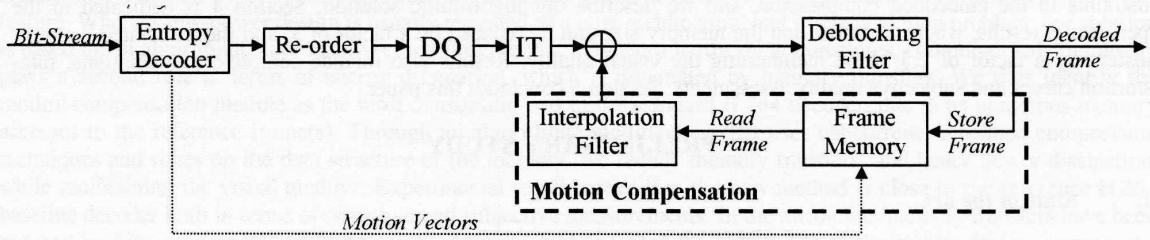


Figure 1: General scheme of a standard H.264 decoder.

The storing of the latest decoded frame is unavoidable in the standard scheme because all its pixels may serve as references for the next frame(s). In CIF, with a 4:2:0 format, this implies $352x288x1.5=152,064$ bytes to be stored. Moreover, even in the Baseline Profile, up to 5 preceding frames are used as references, so with the currently decoded frame there are 6 frames in the memory. Consequently, a 1MB external memory is required. It consumes much more power than a small internal memory, partly due to longer connections. During the decoding of one frame, around 300kB of data have to be transferred to and from this memory (read/write operations). No other module needs to store and access so much data: most of them can be performed on the fly, except the deblocking filter in case FMO (Flexible Macroblock Ordering) is enabled. We do not consider this possibility in the remaining of this paper. The Motion Compensation is thus responsible for a major part of the bandwidth between the external memory and the processing modules.

2.3. An existing technique for reducing memory size and transfers in the MC: Embedded Compression

Recent work has put the strain on reducing this bandwidth by presenting an appropriate memory management[13]. The pixels of the stored reference frame(s) are mapped to the structure of a DDR-SDRAM block-by-block instead of line-by-line, yielding excellent results. This method optimized the implementation of the MC without changing the decoding algorithm itself. It was then proposed to go further by using embedded compression techniques[14]. Embedded Compression (EC) consists in compressing the reference frame(s) in the feedback loop of a conventional predictive video coding scheme[15] as depicted in Figure 2. The additional encoder and decoder that perform the EC have a computational cost (complexity). This cost must be moderate, because in an extreme case the associated increase in CPU dissipation could exceed the expected power savings on memory accesses.

On quality aspects, if the selected compression technique is lossless it has no impact on the reconstructed sequence, but of course the compression ratio (and thus the expected gain) is limited (two at most). On the contrary if the compression is lossy the memory savings are much higher. It comes at the expense of a quality degradation that worsens along a Group of Pictures (GoP). The reference frames are compressed with loss, so the decoding of the next frame is erroneous. The predictive structure of video coding schemes make this reconstruction error propagate and accumulate: visual quality gets inexorably poorer and poorer. This phenomenon is known as the drift effect. Designing a low-power decoder with embedded compression, we have to minimize this unwanted artifact that is temporally very annoying.

Google Exhibit No. 1063
Page 1108

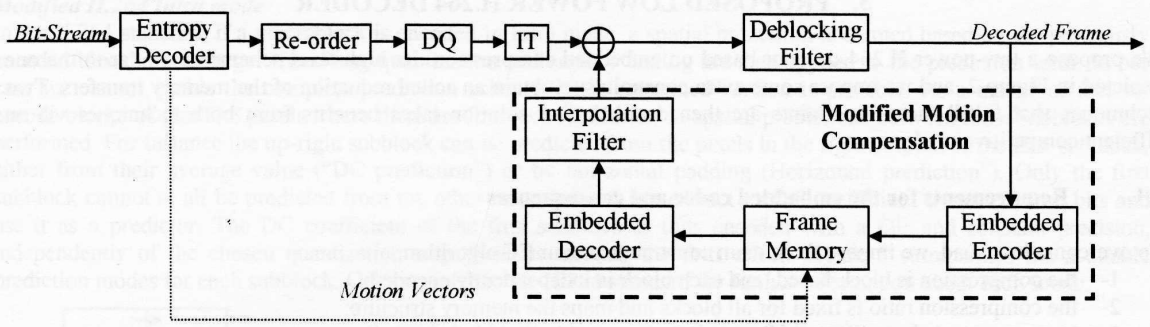


Figure 2: General scheme of an H.264 decoder with Embedded Compression.

2.4. Drawback of Embedded Compression: data overhead

When the goal of Embedded Compression (EC) is purely to reduce the required memory size for cost reasons, virtually any efficient still image compression technique can be performed. In practice, block-based schemes are convenient because they match the block-oriented structure of the incoming bit-stream, allowing on-the-fly processing. However, we also aim at reducing the bandwidth, and the relation between the Compression Ratio (CR) of the EC and the gain in memory transfers is not direct, due to overhead.

We define overhead as the ratio between the number of pixels that are actually accessed during the MC of a block and the number of pixels that are really useful in the reference block. This ratio is superior to 1 because of the block-based nature of EC techniques. For instance let us consider that the reference frames are encoded on an 8x8 block grid. The reference frame no longer has a pixel-oriented structure, but a block-oriented one (Figure 3): to access the value of one pixel we must fetch and decode the whole block it belongs to.

This is not a problem as long as the reference pixels in the MC form blocks that are aligned with the grid of the EC, but it becomes annoying when the motion vector points to a reference that overlaps several encoded blocks. It was already true with previous standards[15] but it becomes even more serious with H.264. Indeed, a 16x16 macroblock can be divided in 16x8, 8x16 or 8x8 blocks during the MC. Any 8x8 block can then be sub-divided into 8x4, 4x8 or 4x4 sub-blocks. In the case depicted in Figure 4, this has a dramatic impact on the overhead. Indeed the compensated block is a 4x4 block (allowed in H.264) that is not at all aligned with the 8x8 grid. Four encoded blocks, corresponding to 256 pixels, have thus to be read and decoded just to get the useful 16 pixels. The overhead in that worst-case scenario is 16. The long 6-tap filters that perform sub-pixel interpolation also contribute to this phenomenon.

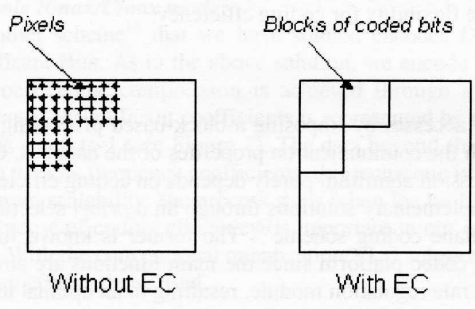


Figure 3: Block-oriented frame structure.

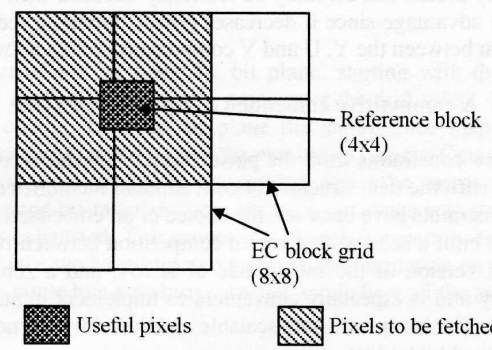


Figure 4: Overhead example.

Google Exhibit No. 1063
Page 1109

3. PROPOSED LOW POWER H.264 DECODER

We propose a low-power H.264 decoder based on embedded compression. Its high-level scheme is similar to the one depicted in Figure 2, and we impose some extra constraints to obtain an actual reduction of the memory transfers. Two techniques that fulfill these constraints are then detailed. Our solution takes benefits from both techniques via an efficient competitive mode.

3.1. Requirements for the embedded codec and consequences

To overcome overhead, we impose three main constraints on our EC algorithm:

1- the compression is block-based, and each block is independently encoded;
2- the compression ratio is fixed for all blocks and maps the memory structure;
3- luminance and chrominance planes are jointly stored in the memory.

First, having a block-based EC fits well with the structure of the standard decoder since it allows on-the-fly processing and avoids extra buffering. Furthermore, independent encoding of the blocks ensures that a limited amount of data must be fetched to recover some pixels pointed by the current motion vector. This excludes any kind of prediction between blocks, in particular spatial prediction, because one would have to load and decode also the previous blocks in the reference frame, which would annihilate the desired effect of memory transfer reduction.
The second constraint enables fast and easy access to any block in the reference frame. The start and end addresses of each encoded block are known, so that they do not have to be retrieved in an extra table in memory. The full benefit in terms of transfers and power comes when this constraint is coupled with a good data mapping strategy. In a typical SDRAM, memory addresses point to words of N bits (e.g. N=32), where N corresponds to the bus width. Moreover, data can be accessed by bursts of M consecutive words. Long bursts dissipate less energy than small ones relatively to the amount of transferred data. Generally, the maximum burst length is M=8 words (256 bits in the case of a 32-bit data bus). Hence, we set a fixed compression ratio CR so that the pixels of the source block fit in the length of one (or multiple) burst(s).
The third constraint increases data locality since motion vectors are identical for luminance and chrominance. Thus, pixels of the three planes can be accessed in one transfer, saving much energy in data-bus communications. We work with 4:2:0 progressive materials, so an 8x8 block in the pixel domain contains 512 bits of luminance and 256 bits of chrominance data. It exactly fits in one 8-word burst of 256 bits with the above memory structure if we apply a compression factor of 3.

While they clearly act in favor of a limited overhead effect, our first two constraints make the efficiency of our scheme difficult in terms of compression ratio and visual quality. Indeed, we cannot benefit from correlations between adjacent blocks. Furthermore, the fixed CR implies that too few bits might be used to correctly encode complex blocks while some easy blocks can certainly be losslessly encoded with less than 256 bits. On the contrary, the third constraint gives a double advantage since it decreases power through a reduction of the memory requests and also enables a joint bit-allocation between the Y, U and V components, which allows more flexibility for coding efficiency.

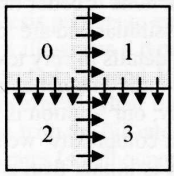
3.2. A competitive embedded compression scheme

The above constraints limit the power dissipation due to memory accesses by imposing a block-based processing that fits well with the data structure of conventional memories and with the communication properties of the data bus. Once these constraints have been set, the choice of an embedded compression algorithm purely depends on coding efficiency. We have built a scheme that uses a competition between two complementary solutions through an *a priori* selection: a modified version of the Intra mode of H.264, and a zonal bit-plane coding scheme[16]. The former is known for its efficiency and is especially convenient to implement in an H.264 codec platform since the main functions are already available. The latter is bit-rate scalable and thus does not need any rate regulation module, resulting in an optimal use of the allocated bit budget.

Google Exhibit No. 1063
Page 1110

Modified H.264 Intra mode

In the H.264 standard, if a macroblock is encoded in Intra mode, a spatial prediction is formed based on the previously encoded blocks. 4x4 integer transform, quantization and entropy coding are then applied to the prediction error. In our case we cannot enable the complete prediction scheme, but a restrained version is possible. As a matter of fact we encode blocks of size 8x8 pixels where there are four 4x4 subblocks, so spatial predictions between these subblocks are performed. For instance the up-right subblock can be predicted from the pixels in the right column of the first subblock, either from their average value ("DC prediction") or by horizontal padding (Horizontal prediction"). Only the first subblock cannot at all be predicted from the other ones. Instead, we extract its DC coefficient truncated on 8 bits and use it as a predictor. The DC coefficient of the first subblock is thus encoded with a fair and constant precision, independently of the chosen quantization step. It helps limiting the drift effect. Figure 5 summarizes the available prediction modes for each subblock. One can refer to the H.264 standard for further information about each mode.



0: predicted by its own average value on 8 bits
1: DC prediction or horizontal prediction from subblock 0
2: DC prediction or vertical prediction from subblock 0
3: DC, horizontal, vertical, vertical-right, diagonal-right, horizontal-down, or horizontal-up prediction from subblocks 0, 1 and 2

Figure 5: Spatial prediction modes available for each 4x4 subblock in the modified H.264 Intra mode.

One bit is enough to transmit the prediction mode of subblock 1, and another one for subblock 2, while 3 bits are needed for subblock 3, leading to a total of 5 bits. Chrominance pixels directly form 4x4 blocks, so we do not perform any spatial prediction there, except that their average value truncated on 8 bits is also losslessly encoded for each UV component. Then the 4x4 transform and the quantization process are similar to the standard, but with a limited number of 32 possible quantization steps. The Level and Run tables are encoded using a variable-length coding (VLC). In fact, a complex entropy coding (e.g. the Context-based Adaptive Binary Arithmetic Coding - CABAC) is not justified in this application because the context would have to be reset after each block, leading to a sub-optimal solution. VLC is just as efficient here. QP is adjusted so that the total length of the bit-stream does not exceed the chosen value (in our case 256 bits or a burst of eight 32-bit words). Finally, the bit-stream corresponding to one block is composed as depicted in Figure 6.

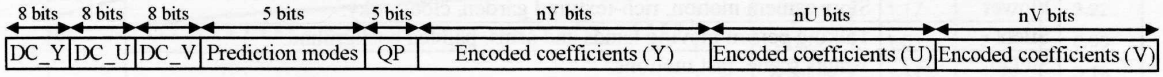


Figure 6: Bit-stream composition in the modified H.264 Intra mode.

Scalable Rmax/Cmax mode

The other scheme[16] that we have studied encodes DCT coefficients bit plane by bit plane, starting with the Most Significant Bits. As in the above solution, we encode the DC coefficients losslessly to counter the drift effect. For the AC coefficients, compression is achieved through a zonal coding: for each bit plane the significance map of the previously insignificant coefficients is represented by the maximum row Rmax and the maximum column Cmax where a bit is equal to 1 (see Figure 7). The data beyond this zone does not need to be transmitted. Since the energy usually lies in the low frequency coefficients, this technique implies a good bit-rate reduction. Its great advantage is to provide a fine-grain scalability: the process stops when the bit budget is exhausted. This property, although it generally comes at the expense of coding efficiency, is important in our context since we have imposed a severe rate limitation on a block basis. With the conventional quantization/VLC solution above, some bits are always wasted, while here all the available bits can efficiently be used.

Google Exhibit No. 1063
Page 1111

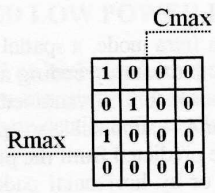


Figure 7: Zonal bit-plane coding.

Competition mode

In practice, we have observed that although the two depicted algorithms give encouraging results, none is better than the other in all conditions and for all sequences. More precisely, their respective artifacts are very dissimilar and are located in regions with different properties. The modified Intra mode tends to blur the picture and to lose details in very textured areas. On the contrary, the scalable solution gives a sharper rendering in edges and rich textures but generates an unpleasing noise/ringing effect in smoother regions, due to the bit-plane truncation. Consequently, our solution is based on the competition between these two modes, to achieve the best performance. In order to limit complexity, we select the encoding mode through an *a priori* criterion: if the activity (or energy) of the current block is higher than a given threshold, then the scalable scheme is chosen, otherwise the modified H.264 Intra is applied. *A priori* selection is important because it generates a very limited computational overhead compared to an *a posteriori* solution where both schemes are processed before one gets selected.

4. EXPERIMENTAL RESULTS

We present our results on several dimensions: the memory aspects (size and transfers), the complexity issue, the objective and the subjective quality. Our test set is composed of CIF sequences at 25 frames/s with various contents as given in Table 1. We have encoded them with the JM6 reference software of H.264, set to respect the Baseline Profile, with an Intra period of 25 frames. Bit-rates are in the range [150…512] kb/s. This spatio-temporal resolution, these sequences and the encoding conditions were chosen as representative of the targeted applications.

Foreman	Face close-up, moderate motion.
Flower	Slow camera motion, rich-textured garden, cloudy sky.
Silent	Talking person moving hands and arms, painted background.
Stefan	Tennis scene, fast motions.
Weather	Weather forecast, synthetic map (background), talking person (foreground).

Table 1: Test sequences.

4.1. Memory gain

Our primary goal is to reduce power consumption through a decrease of the memory transfers. We have seen that we apply a Compression Ratio CR of 3 so that an 8x8 block maps a data burst of 256 bits. The first consequence is that the memory occupancy of the reference frame(s) is divided by three. Hence a smaller memory can be implemented, which is very positive in terms of both costs and consumption. Indeed, smaller memories can be embedded closer to the CPU, leading to less cable dissipation during accesses. Concerning transfers (and thus bandwidth), the gain is not so direct due to overhead (see paragraph 2.4). In the following, we describe how we measure it.

Overhead: statistical prediction

The memory accesses are divided by CR/Ovh at the reading side, where Ovh is the overhead. At the storing side, this overhead issue is irrelevant and transfers are thus divided by CR. Equation 1 expresses the global gain in memory transfers (load/store) with the approximation that there are as many loads as stores in the standard decoder. One can see that overhead can theoretically counter-balance the benefits of EC regarding memory transfers. We have to measure it precisely to assess the memory bandwidth that is actually saved.

Google Exhibit No. 1063
Page 1112

$$Transfer\_Gain = \frac{Memory\_Store_{EC} + Memory\_Load_{EC}}{Memory\_Store_{Std} + Memory\_Load_{Std}} = \frac{1 + Ovh}{2 * CR} \qquad \text{Equation 1}$$

Overhead strictly depends on the block size of the EC, on the size of the current motion-compensated block, and on the value of its motion vector. It is a part of our contribution to quantify their impact through a statistical analysis of the sequences of the test set.

There are four main inter prediction modes for luminance pixels in H.264: 16x16, 16x8, 8x16 and 8x8. The 8x8 mode can be further decomposed into 4x4, 8x4 and 4x8 partitions. For each mode and sub-mode, we have counted the occurences when a reference block is aligned with the chosen EC block grid (8x8). If it is not, then there is a separation between horizontal misalignment, vertical misalignment, and both. In each case, the number of encoded blocks to retrieve (and thus the number of pixels to get) is different. Then the statistical apparition of each mode is also computed. A synthesis permits to extract the average overhead from our test set.

Figure 8 illustrates a typical case, here for the Stefan sequence, where we also indicate the equivalent results for an EC block-grid of 4x4 and 16x16. A simple analysis shows that the result is qualitatively coherent. The 4x4 grid generates fewer overheads than the other ones because it is closer to pixel accuracy. Furthermore, for all grid sizes, overheads increase with the bit-rate. This is due to the H.264 encoder that allows macroblock partitioning and large motion vectors at high rates (small quantization step) while it favors the null vector with a 16x16 partition at low rates.

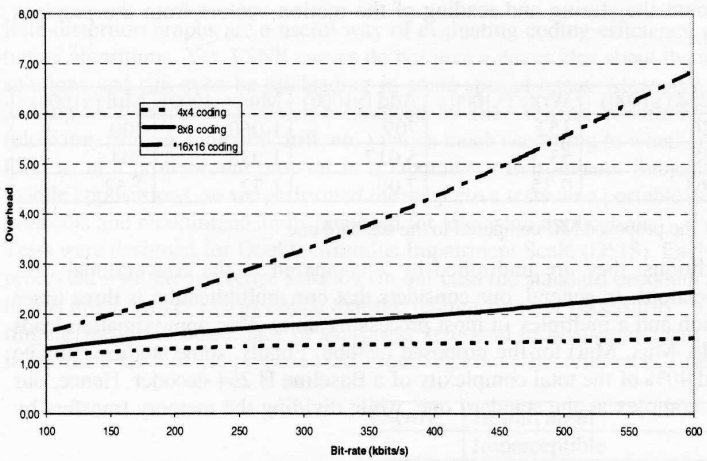


Figure 8: overhead vs bit-rate (Stefan sequence).

Sequence	4x4 EC Grid	8x8 EC Grid	16x16 EC Grid
Foreman	1.31	1.77	3.69
Flower	1.30	1.74	3.77
News	1.14	1.51	2.78
Silent	1.17	1.50	3.22
Stefan	1.51	2.44	6.95
Weather	1.17	1.49	3.18
All	**1.27**	**1.73**	**3.93**

Table 2: overhead vs EC block-grid for each sequence.

Table 2 summarizes the statistical analysis for each sequence of the test set. Not surprisingly, still sequences generate the smallest overhead values (News, Weather) because the motion vector is often equal to zero. On the contrary, a fast sequence like Stefan generates much overhead. In the remaining of the experiments we only present results for the 8x8 EC block grid, which turned out to offer the best trade-off between transfer overhead (that is minimized if the block grid is small) and coding efficiency (that prefers larger blocks).

Memory transfers

Thanks to the formula of Equation 1 and to the statistical results of Table 2, we can derive the average gain in memory transfers for each sequence. For the 8x8 grid, the transfers gain lies between 0.40 for the Weather sequence and 0.55 for the Stefan sequence, respectively corresponding to a 60% and a 45% reduction. The average gain for all the sequences is about 0.45 (or 55% of reduction), corresponding to a gain of 0.33 for the storing and 0.57 for the reading of a reference frame. These figures correspond to a Compression Ratio (CR) of 3 and lead to very good visual results as we

Google Exhibit No. 1063
Page 1113

will see in section 4.4. Higher CR values, and thus higher gains, can be achieved. For instance CR=8 implies an average reduction of 80% for memory transfers at the expense of more visible degradations.

4.2. Complexity evaluation (CFDG)

Although memory transfers represent the major part of the energy dissipation in modern architecture, computational and logical operations are not negligible. By applying Embedded Compression, we have added some modules to the conventional standard decoder, and consequently increased the complexity. In this paragraph we estimate the number of added operations through a CFDG analysis.

A Control Flow Graph (CFG) is an abstract data structure, a representation of a procedure or program. It consists of nodes performing operations and exchanging data. Each node in the graph represents a basic block, i.e. a straight-line piece of code without any jump. A CFDG is a CFG where Data dependencies are highlighted. In general, CF(D)G are internally built-up by compilers to optimize binary executables. In our case, we found it was a good tool to precisely decompose an algorithm at a high-level. It allows analyzing its complexity, memory accesses, level of data dependencies and degree of parallelism. All these notions directly give indications on the ease to implement a low-power system based on the studied algorithm.

The CFDG methodology has been used to evaluate the average complexity of the reference MC and of the proposed solution with EC. The results given in Table 3 represent the number of additions (Add), multiplex (Mux, related to 'if' statements) and multiplications (Mul) for performing the MC of one frame. It takes into account statistics about the motion vectors (including sub-pixel interpolation). The numbers of memory read and write are taken from the study of the previous paragraph, and also take into account the storing and reading of the motion vectors from the previous neighboring blocks (for motion vector decoding).

	Read (x1000)	Write (x1000)	Add (x1000)	Mux (x1000)	Mul (x1000)
MC of the H.264 decoder	226	152	709	1,060	200
Proposed MC with Embedded Compression	138	53	5,017	1,938	504
Ratio	0.61	0.35	7.00	1.72	2.52

Table 3: Complexity of the proposed MC compared to the standard one.

One can notice that EC performs a lot of additions: they are multiplied by 7 compared to the conventional MC. However, additions are less costly than multiplications. In general, one considers that one multiplication is three times as complex (and power-consuming) as an addition and a multiplex in most processing units. This approximation leads to an average ratio of 3.57 more operations (Add, Mux, Mul) for the proposed method. Finally, some studies estimate that the Motion Compensation represents around 40% of the total complexity of a Baseline H.264 decoder. Hence, our solution is globally between 1.5 and 2 times as complex as the standard one, while dividing the memory transfers by more than 2.

4.3. Objective quality (PSNR)

Embedded Compression generates an unavoidable reconstruction drift at the decoder, which impacts the resulting quality. However, in the introduction we have stressed the fact that we judge our algorithms on a power-rate-distortion basis, meaning that a large gain in power can justify a slight increase in distortion. We have to quantify this distortion to assess the acceptability of our solution.

Figure 9 and Figure 10 depict the rate-distortion graphs of the standard decoder and of the proposed one, for two representative sequences of the test set (Flower and Foreman). The gap between the two methods increases with the bit-rate because the decoded frames are better preserved and contain more details at higher rates, implying a non-negligible loss of data during the Embedded Compression. On the contrary, the decoded pictures at lower rates are easier to encode by the EC module because they are already degraded anyway.

The average PSNR loss on all the sequences of the test set is 0.7 dB. It is null at the low end of the bit-rate range and can reach more than 2.0 dB around 512 kbits/s. However, some authors have remarked that the visual impact of the EC is not necessarily as high as the gaps in the rate-distortion curves[14].

Google Exhibit No. 1063
Page 1114

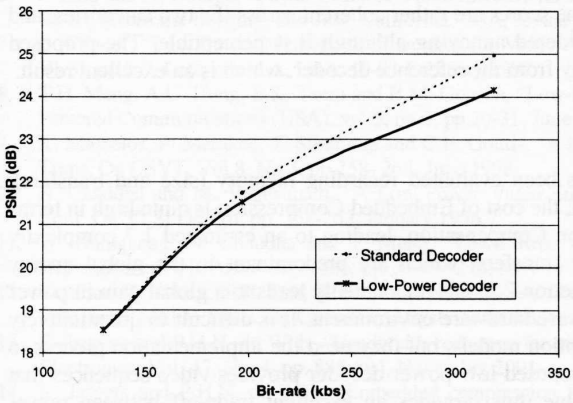


Figure 9: Flower sequence.

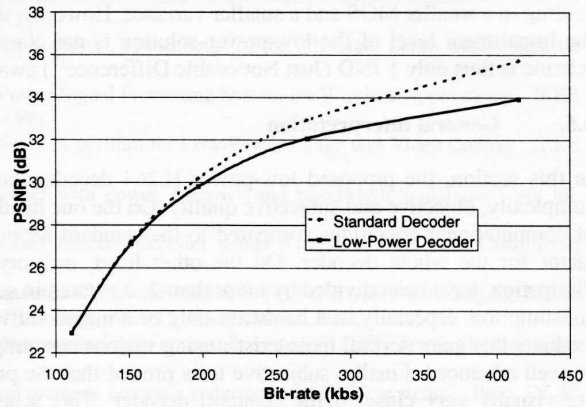


Figure 10: Foreman sequence.

4.4. Subjective quality (visual tests)

Rate-distortion graphs are a useful way of evaluating coding efficiency, especially during the process of designing and tuning algorithms. Yet, PSNR curves do not give a direct idea about the actual differences in visual quality between two solutions and can even be misleading in some special cases. Moreover, the subjective quality that viewers feel while looking at a sequence is much influenced by the viewing conditions. For instance the visibility of some artifacts (blocking, ringing, temporal drift, etc.) varies much according to whether people look at the palm-sized LCD screen of a PDA or at a professional monitor in a laboratory environment. Subjective tests are thus needed. Our solution targets mobile applications, so we performed the subjective tests on a portable device, following as much as possible the known protocols and recommendations proposed for television applications[17].

Tests were designed for Double Stimulus Impairment Scale (DSIS). Each test consists of a pair of stimuli: one sequence processed with the reference solution (in our case the standard decoder) and with the proposed one. Each pair is shown in this order, twice. Let us notice that the reference is not necessarily "artifact-free": typical coding artifacts (blocking, ringing, mosquito noise, lack of sharpness, temporal inconsistency) can be present. The viewers are asked to score the impairment level of the proposed technique regarding this reference. Table 4 shows the scoring scale.

Score	Impairment
5	Imperceptible
4	Perceptible but not annoying
3	Slightly annoying
2	Annoying
1	Very annoying

Table 4: Scoring scale.

The device is a mobile video player with a 320x240 LCD-TFT screen and a depth of 16 bits/pixel. The viewing distance is not imposed. Both expert and naïve viewers are asked to hold the mobile device at a convenient distance and to keep this distance constant during the test. The room illumination is low. Finally, the Mean Opinion Score (MOS) is computed. Results are summarized in Table 5.

Category	Number of viewers	Mean Opinion Score	Variance	Min Score	Max Score
Experts	9	**4.0**	0.4	2	5
Naïves	5	**4.3**	0.5	3	5

Table 5: Subjective Tests results.

Google Exhibit No. 1063
Page 1115

As usual, expert viewers are both more sensitive to artifacts and more consistent in their judgment than naïve viewers, leading to a smaller MOS and a smaller variance. However, the scores are rather coherent across the two categories, and the impairment level of the low-power solution is not considered annoying although it is perceptible. The proposed scheme is thus only 1 JND (Just Noticeable Difference[18]) away from the reference decoder, which is an excellent result.

4.5. General interpretation

In this section, the proposed low-power H.264 decoder has been evaluated regarding memory (size and transfers), complexity, objective and subjective quality. On the one hand, the cost of Embedded Compression is quite high in terms of computational operations compared to the standard Motion Compensation, leading to an estimated 1.7 complexity factor for the whole decoder. On the other hand, memory transfers, which are predominant in the global energy dissipation, have been divided by more than 2. As stated in section 2.2, this undoubtedly leads to a global gain in power consumption, especially in a hardware-only or a mixed software/hardware environment. It is difficult to quantitatively evaluate this gain: several tools exist[6], using various consumption models, but they need the implementation process to be well advanced. Finally, subjective tests proved that the proposed low-power decoder provides video sequences that are visually very close to the standard decoder. This scheme thus provides an excellent trade-off between power consumption and visual quality. It is possible to achieve larger gains by increasing the Compression Ratio of the EC at the expense of a degraded visual quality, which can be acceptable in some applications.

5. CONCLUSIONS

We have presented a modified H.264 decoder that gracefully degrades the output video quality in order to significantly reduce energy dissipation. We have first observed that the Motion Compensation was the most power-consuming part of the decoder, due to its numerous accesses to the frame memory. Embedded Compression (EC) was then identified as a promising technique to reduce these transfers. In order to limit the data overhead that is inherent to EC, we have imposed some requirements that help mapping the EC grid with the memory structure. Once these constraints have been set, we have applied a compression factor of 3 on the reference frames, using a combination of two complementary EC techniques. The selection criterion is *a priori* and low-cost.
Experimental results show that memory transfers, which dominate power consumption, are reduced by 55% in average for CIF sequences in mobile application conditions. The computational complexity of the decoder is globally increased by 1.7, which should not counter-balance the power gain due to memory transfers. Furthermore, subjective tests proved that our solution maintains the visual quality of a standard decoder. The proposed decoder thus provides an excellent trade-off between power consumption and visual quality, making it an advantageous solution for mobile applications.

ACKNOWLEDGEMENTS

The authors would like to acknowledge Yi Li and Carolina Miro, from the Platform & Integration group of the Philips Digital Systems Labs in Paris, for their work on the evaluation of the complexity of the different decoders used in this paper.

REFERENCES

1. P. van der Wolf , P. Lieverse , M. Goel , D. La Hei , K. Vissers, "An MPEG-2 decoder case study as a driver for a system level design methodology", Proc. of the 7th Int. Workshop on Hardware/Software Codesign, pp.33-37, Roma, Italy, March 1999.
2. C.W. Yoon *et al.*, "An 80/20 MHz 160 mW multimedia processor integrated with embedded DRAM, MPEG-4 accelerator and 3D rendering engine for mobile applications", IEEE Journal of Solid-State Circuits, Vol. 36, Issue 11, pp. 1758–1767, November 2001.
3. L. Nachtergaele, F. Catthoor, B. Kapoor, S. Janssens and D. Moolenaar, "Low-Power Data Transfer and Exploration for H.263 Video Decoder System", IEEE Journal on Selected Areas in Communications, Vol. 16, No. 1, January 1998.
4. W. Badawy and M. Bayoumi, "Algorithm-based Low-Power VLSI Architecture for 2D-Mesh Video-Object Motion Tracking", IEEE Trans. On Circuits and Systems for Video Technology, Vol. 12, April 2002.
5. A. Chandrakasan and W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits", Proc. Of the IEEE, Vol. 83, April 1995.

Google Exhibit No. 1063
Page 1116

6. J. Jung and A. Bourge, "Power Scalable Video Encoder for Mobile Devices based on Collocated Motion Estimation", Proc. of VCIP, Electronic Imaging, San Jose, California, USA, January 2004.
7. B. Kapoor, "Low Power Memory Architectures for Video Applications", Proceedings of the 8th Great Lakes Symposium on VLSI, pp.2-7, Lafayette, LA, USA, 19-21 February 1998.
8. T.H. Meng, A.C. Hung, E.K. Tsern and B.M. Gordon, "Low-Power Signal Processing System for Wireless Applications", IEEE Personal Communications (USA), vol.5, no.3, pp.20-31, June 1998.
9. K. Masselos, P. Merakos, T. Stouraitis and C.E. Goutis, "A Novel Algorithm for Low-Power Image and Video Coding", IEEE Trans. On CSVT, Vol.8, No.3, pp.258- 263, June 1998.
10. N. Chaddha and M. Vishwanath, "A Low Power Video Encoder with Power, Memory and Bandwidth Scalability", 9th Int. Conf. On VLSI Design, pp. 358-263, January 1996.
11. W. Namgoong, N. Chaddha and T. Meng, "Low-Power Video Encoder/Decoder using Wavelets/TSVQ with Conditional Replenishment", ICASSP'96, Vol.6, pp. 3240-3243, May 1996.
12. Y. Chen, Z. Zhong, T.H. Lan, S. Peng and K. van Zon, "Regulated Complexity-Scalable MPEG-2 video decoding for Media Processors", IEEE Trans. On Circuits and Systems for Video Technology, Vol. 12, No. 8, August 2002.
13. E. Jaspers and P.H.N. de With, "Bandwidth reduction for video processing in consumer systems", IEEE Trans. On Consumer Electronics, Vol. 47, No. 4, pp. 885-894, November 2001.
14. E. Jaspers and P.H.N. de With, "Embedded compression for memory resource reduction in MPEG systems", Proc. Of 3rd Benelux IEEE Signal Processing Symposium, Leuven, Belgium, March 2002.
15. P.H.N. de With, P.H. Frencken and M. van der Schaar, "An MPEG decoder with embedded compression for memory reduction", International Conference on Consumer Electronics, Digest of Technical Papers, pp. 60-62, June 1998.
16. R.J. van der Vleuten, "Low-complexity lossless and fine-granularity scalable near-lossless compression of color images", Proceedings of the Data Compression Conference, pp.477, April 2002.
17. "Methodology for the subjective assessment of the quality of television pictures", ITU-R BT-500.10, updated in 2000.
18. J. Lubin and D. Fibush, "Sarnof JND vision model", T1A1.5 Working Group Doc. No. 97-612, T1 Standards Committee, 1997.

Google Exhibit No. 1063
Page 1117

PROCEEDINGS

Electronic Imaging
Science and Technology

Visual Communications and Image Processing 2004

Sethuraman Panchanathan
Bhaskaran Vasudev
Chairs/Editors

20–22 January 2004
San Jose, California, USA

IS&T —The Society for Imaging Science and Technology
SPIE—The International Society for Optical Engineering

SPIE Vol. 5308
Part One of Two Parts

Google Exhibit No. 1063
Page 1118

PROCEEDINGS OF

Electronic Imaging
Science and Technology

Visual Communications and Image Processing 2004

Sethuraman Panchanathan
Bhaskaran Vasudev
Chairs/Editors

20–22 January 2004
San Jose, California, USA

Sponsored by
IS&T—The Society for Imaging Science and Technology
SPIE—The International Society for Optical Engineering

In Cooperation with
IEEE Circuits and Systems Society

Published by
IS&T—The Society for Imaging Science and Technology
SPIE—The International Society for Optical Engineering

SPIE Vol. 5308
Part One of Two Parts

Google Exhibit No. 1063

TA1632
.V56
2004
v.1

The papers published in this volume compose the proceedings of the technical conference cited on the cover and title page. Papers were selected by the conference program committee, and were subject to review by the editors and program committee. The inclusion of these papers in this publication does not necessarily constitute endorsement by the editors or by the publishers.

Please use the following format to cite material from this book:

Author(s), "Title of Paper," in *Visual Communications and Image Processing 2004*, edited by Sethuraman Panchanathan, Bhaskaran Vasudev, Proceedings of SPIE-IS&T Electronic Imaging, SPIE Vol. 5308 (2004) page numbers.

ISSN 0277-786X
ISBN 0-8194-5211-4

Published by
SPIE—The International Society for Optical Engineering
P.O. Box 10, Bellingham, Washington 98227-0010 USA
Telephone 1 360/676-3290 (Pacific Time) · Fax 1 360/647-1445
and
IS&T— The Society for Imaging Science and Technology
7003 Kilworth Lane, Springfield, Virginia 22151-4008 USA
Telephone 1 703/642-9090 (Eastern Time) · Fax 1 703 642-9094

Copyright © 2004, The Society of Photo-Optical Instrumentation Engineers and The Society for Imaging Science and Technology.

The papers published in these proceedings reflect the work and thoughts of the authors and are published herein as submitted. The publishers are not responsible for the validity of the information or for any outcomes resulting from reliance thereon.

Copying of material in this book for internal or personal use, or for the internal or personal use of specific clients, beyond the fair use provisions granted by the U.S. Copyright Law is authorized by SPIE and IS&T subject to payment of copying fees. The Transactional Reporting Service base fee for this volume is $15.00 per article (or portion thereof), which should be paid directly to the Copyright Clearance Center (CCC), 222 Rosewood Drive, Danvers, MA 01923. Payment may also be made electronically through CCC Online at http://www.copyright.com. Other copying for republication, resale, advertising or promotion, or any form of systematic or multiple reproduction of any material in this book is prohibited except with permission in writing from the publisher. The CCC fee code is 0277-786X/04/$15.00.

Printed in the United States of America.

Google Exhibit No. 1063
Page 1120

Contents

Part One

xv *Conference Committee*

SESSION 1 MULTIMEDIA TECHNOLOGIES FOR EMBEDDED SYSTEMS I

1 **A half D1 MPEG-4 encoder on the BSP-15 DSP** [5308-02]
L. Chen, Sarnoff Corp. (USA); Z. He, Univ. of Missouri/Columbia (USA); C. W. Chen, Florida Institute of Technology (USA); M. A. Isnardi, Sarnoff Corp. (USA)

6 **MPEG-4 advanced simple profile video encoding on an embedded multimedia system** [5308-03]
S. M. Akramullah, iVAST Inc. (USA); R. K. Giduthuri, viLOGIC, Inc. (USA); G. Rajan, iVAST Inc. (USA)

SESSION 2 MULTIMEDIA TECHNOLOGIES FOR EMBEDDED SYSTEMS II

18 **Resource-aware complexity scalability for mobile MPEG encoding** [5308-04]
S. Mietens, Philips Research Labs. (Netherlands); P. H. N. de With, LogicaCMG (Netherlands) and Technische Univ. Eindhoven (Netherlands); C. Hentschel, Technical Univ. Cottbus (Germany)

31 **Novel algorithm to reduce the complexity of quarter-pixel motion estimation** [5308-05]
P. S. S. B. K. Gupta, R. Korada, Emuzed India Private, Ltd. (India)

37 **Video coding for decoding power-constrained embedded devices** [5308-06]
L. Lu, V. Sheinin, IBM Thomas J. Watson Research Ctr. (USA)

SESSION 3 MULTIMEDIA TECHNOLOGIES FOR EMBEDDED SYSTEMS III

45 **Power-scalable video encoder for mobile devices based on collocated motion estimation** [5308-07]
J. Jung, A. Bourge, Philips Digital Systems Labs. (France)

57 **Power-rate-distortion analysis for wireless video communication under energy constraint** [5308-08]
Z. He, Univ. of Missouri/Columbia (USA); Y. Liang, I. Ahmad, Univ. of Texas at Arlington (USA)

69 **Compression of compound images and video for enabling rich media in embedded systems** [5308-09]
A. Said, Hewlett-Packard Labs. (USA)

83 **Layered Wyner-Ziv video coding** [5308-10]
Q. Xu, Z. Xiong, Texas A&M Univ. (USA)

LINDA HALL LIBRARY
Kansas City, Mo.

Google Exhibit No. 1063
Page 1121

92 **Optimal speech codec implementation on ARM9E (v5E architecture) RISC processor for next-generation mobile multimedia** [5308-11]
A. K. Bangla, M. K. Vinay, P. V. Suresh Babu, Emuzed India Private Ltd. (India)

SESSION 4 MULTIMEDIA TECHNOLOGIES FOR EMBEDDED SYSTEMS IV

102 **An overview of problems in image-based location awareness and navigation** [5308-12]
Y.-H. Lu, E. J. Delp III, Purdue Univ. (USA)

110 **Joint PHY and MAC layer power optimization for video transmission over wireless LAN** [5308-13]
X. Lu, Polytechnic Univ. (USA); Y. Chen, Philips Research USA (USA); Y. Wang, Polytechnic Univ. (USA)

119 **MPEG-7 meta-data enhanced encoder system for embedded systems** [5308-14]
K. Asai, H. Nishikawa, D. Kudo, Mitsubishi Electric Corp. (Japan); A. Divakaran, Mitsubishi Electric Research Labs. (USA)

124 **Post-processing of compressed video using a unified metric for digital video processing** [5308-15]
L. Boroczky, Y. Yang, Philips Research USA (USA)

132 **Efficient implementation of elliptic curve cryptography (ECC) on embedded media processors** [5308-16]
Y. Hu, Q. Li, L. Huang, C.-C. J. Kuo, Univ. of Southern California (USA)

144 **Exploiting temporal correlation with adaptive block-size motion alignment for 3D wavelet coding** [5308-17]
R. Xiong, Institute of Computing Technology (China); F. Wu, S. Li, Microsoft Research Asia (China); Z. Xiong, Texas A&M Univ. (USA); Y.-Q. Zhang, Microsoft Research Asia (China)

SESSION 5 IMAGE/VIDEO SEGMENTATION I

156 **Joint space-time motion-based video segmentation and occlusion detection using multiphase level sets** [5308-18]
M. Ristivojević, J. Konrad, Boston Univ. (USA)

168 **Geometric-model-based segmentation of the prostate and surrounding structures for image-guided radiotherapy** [5308-19]
X. Tang, Y. Jeong, R. J. Radke, Rensselaer Polytechnic Institute (USA); G. T. Y. Chen, Massachusetts General Hospital (USA)

177 **Human body posture recognition using MPEG-7 descriptors** [5308-20]
L. Goldmann, M. Karaman, T. Sikora, Technische Univ. Berlin (Germany)

SESSION 6 IMAGE/VIDEO SEGMENTATION II

189 **Multiple motion and occlusion segmentation with a multiphase level set method** [5308-21]
Y. Shi, J. Konrad, W. C. Karl, Boston Univ. (USA)

Google Exhibit No. 1063
Page 1122

198 **Segmenting focused objects using morphological filters and region merging** [5308-22]
C. Kim, Epson Palo Alto Lab. (USA)

210 **Investigation of implicit active contours for scientific image segmentation** [5308-23]
S. K. Weeratunga, C. Kamath, Lawrence Livermore National Lab. (USA)

SESSION 7 IMAGE/VIDEO/GRAPHICS CODING I

222 **Rate-distortion optimized interactive streaming for scalable bitstreams of light fields** [5308-24]
C.-L. Chang, B. Girod, Stanford Univ. (USA)

234 **Minimizing MPEG-4 sprite coding cost using multi-sprites** [5308-25]
D. Farin, Univ. Mannheim (Germany); P. H. N. de With, LogicaCMG (Netherlands) and Technische Univ. Eindhoven (Netherlands); W. Effelsberg, Univ. Mannheim (Germany)

246 **Adaptive lossless video compression** [5308-26]
S.-G. Park, E. J. Delp III, Purdue Univ. (USA)

255 **Automatic video object detection and mask signal removal for efficient video preprocessing** [5308-27]
Z. He, Univ. of Missouri/Columbia (USA)

264 **Block-based embedded color image and video coding** [5308-28]
N. Nagaraj, GE Global Research (India); W. A. Pearlman, Rensselaer Polytechnic Institute (USA); A. Islam, Nokia Research Ctr. (USA)

SESSION 8 IMAGE/VIDEO/GRAPHICS CODING II

276 **Coding-adaptive transforms** [5308-29]
D. Lelescu, F. Bossen, DoCoMo Communications Labs. USA, Inc. (USA)

288 **MINMAX rate control with a perceived distortion metric** [5308-30]
Y. Sermadevi, M. Masry, S. S. Hemami, Cornell Univ. (USA)

300 **Distributed source coding of multiview images** [5308-31]
M. P. Tehrani, T. Fujii, M. Tanimoto, Nagoya Univ. (Japan)

310 **Lossless compression for three-dimensional images** [5308-32]
X. Tang, W. A. Pearlman, Rensselaer Polytechnic Institute (USA)

320 **Progressive geometry encoder based on the octree structure** [5308-33]
J. Peng, C.-C. J. Kuo, Univ. of Southern California (USA)

331 **Near-lossless compression of digital terrain elevation data** [5308-34]
R. V. Panchagnula, W. A. Pearlman, Rensselaer Polytechnic Institute (USA)

Google Exhibit No. 1063
Page 1123

SESSION 9 VIDEO CODING H.264 I

343 **Improved video coding using long-term global motion compensation** [5308-35]
A. Smolić, Y. Vatis, H. Schwarz, P. Kauff, U. Gölz, T. Wiegand, Fraunhofer-Institut für Nachrichtentechnik Heinrich-Hertz-Institut (Germany)

355 **Multistage mode decision for intraprediction in H.264 codec** [5308-36]
C. Kim, H.-H. Shih, C.-C. J. Kuo, Univ. of Southern California (USA)

364 **Direct macroblock coding for predictive (P) pictures in the H.264 standard** [5308-37]
A. M. Tourapis, Thomson Corporate Research (USA); F. Wu, S. Li, Microsoft Research Asia (China)

SESSION 10 VIDEO CODING H.264 II

372 **Low-power H.264 video decoder with graceful degradation** [5308-38]
A. Bourge, J. Jung, Philips Digital Systems Labs. (France)

384 **Implementation of H.264 encoder on general-purpose processors with hyper-threading technology** [5308-39]
E. Q. Li, Intel Corp. (China); Y.-K. Chen, Intel Corp. (USA)

SESSION 11 MOTION ESTIMATION/MOTION COMPENSATION I

396 **True-motion estimation using feature correspondences** [5308-41]
R. A. Braspenning, G. de Haan, Philips Research Labs. (Netherlands)

408 **Predictive motion estimation with global motion predictor** [5308-42]
S. Sun, S. Lei, Sharp Labs. of America (USA)

416 **Statistical motion prediction with drift** [5308-43]
Z. Li, E. J. Delp III, Purdue Univ. (USA)

428 **Globally optimal wavelet-based motion estimation using interscale edge and occlusion models** [5308-44]
L. Şendur, Univ. of Cambridge (United Kingdom); O. G. Guleryuz, Epson Palo Alto Lab. (USA)

440 **An enhanced fast variable block-size motion estimation scheme for H.264** [5308-45]
C.-H. Kuo, M. Shen, C.-C. J. Kuo, Univ. of Southern California (USA)

SESSION 12 MOTION ESTIMATION/MOTION COMPENSATION II

452 **Transversal versus lifting approach to motion-compensated temporal discrete wavelet transform of image sequences: equivalence and tradeoffs** [5308-46]
J. Konrad, Boston Univ. (USA)

464 **Segmentation in the loop: an iterative object-based algorithm for motion estimation** [5308-47]
H. Blume, J. von Livonius, T. G. Noll, RWTH-Aachen (Germany)

Google Exhibit No. 1063
Page 1124

474 **Reduced complexity genetic algorithm for motion estimation** [5308-48]
R. Khanna, Intel Corp. (USA); M. van der Schaar, Univ. of California/Davis (USA)

486 **Estimation of multiple motions using block matching and Markov random fields** [5308-49]
I. Stuke, T. Aach, E. Barth, C. Mota, Univ. Lübeck (Germany)

497 **Video coding with lifted wavelet transforms and complementary motion-compensated signals** [5308-50]
M. H. Flierl, P. Vandergheynst, Swiss Federal Institute of Technology Lausanne (Switzerland); B. Girod, Stanford Univ. (USA)

509 **Adaptive windowing technique for variable block-size motion compensation** [5308-51]
S.-W. Park, S. H. Kim, Seoul National Univ. (South Korea); C.-S. Kim, Chinese Univ. of Hong Kong (China); S.-U. Lee, Seoul National Univ. (South Korea)

SESSION 13 SCALABLE/MULTIPLE-DESCRIPTION IMAGE/VIDEO CODING I

520 **Transform-domain Wyner-Ziv codec for video** [5308-52]
A. Aaron, S. D. Rane, E. Setton, B. Girod, Stanford Univ. (USA)

529 **Predicting optimal operation of MC-3DSBC multidimensional scalable video coding using subjective quality measurement** [5308-53]
Y. Wang, T.-T. Ng, Columbia Univ. (USA); M. van der Schaar, Univ. of California/Davis (USA); S.-F. Chang, Columbia Univ. (USA)

543 **Rate distortion analysis of layered video coding by leaky prediction** [5308-54]
Y. Liu, Purdue Univ. (USA); P. Salama, Indiana Univ. (USA) and Purdue Univ. (USA); G. W. Cook, Indiana Univ. (USA); E. J. Delp III, Purdue Univ. (USA)

SESSION 14 SCALABLE/MULTIPLE-DESCRIPTION IMAGE/VIDEO CODING II

555 **Fully scalable video coding in multicast applications** [5308-55]
S. Lerouge, R. De Sutter, P. Lambert, Ghent Univ. (Belgium); R. Van de Walle, Ghent Univ. (Belgium) and IMEC (Belgium)

565 **Temporal layered vs. multistate video coding** [5308-56]
S. Ekmekci, T. Sikora, Technische Univ. Berlin (Germany)

576 **Interframe wavelet video coding with operating point adaptation** [5308-57]
T. Rusert, RWTH-Aachen (Germany)

SESSION 15 IMAGE/VIDEO TRANSMISSION I

588 **Systematic lossy forward error protection for error-resilient digital video broadcasting** [5308-58]
S. D. Rane, A. Aaron, B. Girod, Stanford Univ. (USA)

Google Exhibit No. 1063
Page 1125

596 **Error-resilient video coding performance analysis of motion JPEG2000 and MPEG-4** [5308-60]
F. Dufaux, T. Ebrahimi, Swiss Federal Institute of Technology Lausanne (Switzerland) and Emitall SA (Switzerland)

608 **Robust transmission of JPEG2000 codestreams over packet erasure channels** [5308-61]
Z. Wu, A. Bilgin, L. Pu, M. W. Marcellin, Univ. of Arizona (USA)

617 **Error robustness evaluation of H.264/MPEG-4 AVC** [5308-62]
T. Halbach, S. Olsen, Norwegian Univ. of Science and Technology (Norway)

SESSION 16 IMAGE/VIDEO TRANSMISSION II

628 **Collaborative image transmission over wireless sensor networks** [5308-63]
M. Wu, Univ. of Missouri/Columbia (USA); C. W. Chen, Florida Institute of Technology (USA)

636 **Robust image transmission using a new joint source channel coding algorithm and dual adaptive OFDM** [5308-64]
M. Farshchian, S. Cho, W. A. Pearlman, Rensselaer Polytechnic Institute (USA)

647 **Adaptive streaming of high-quality video over wireless LANs** [5308-65]
P. van Beek, S. Deshpande, H. Pan, I. Sezan, Sharp Labs. of America (USA)

661 **Rate-distortion optimized video streaming with rich acknowledgments** [5308-66]
J. Chakareski, B. Girod, Stanford Univ. (USA)

669 **Joint mesh-texture optimization for progressive transmission** [5308-68]
S. Yang, M. Shen, C.-C. J. Kuo, Univ. of Southern California (USA)

680 **Block turbo codes for efficient image transmission over wireless channels** [5308-69]
S. Zaïbi, M. Ammar, K. Amis, R. Pyndiah, S. Saoudi, Ecole Nationale Supérieure des Télécommunications de Bretagne (France)

SESSION 17 IMAGE/VIDEO PROCESSING I

690 **Motion blur reduction for liquid crystal displays: motion-compensated inverse filtering** [5308-70]
M. A. Klompenhouwer, L. J. Velthoven, Philips Research Labs. (Netherlands)

700 **A two-dimensional generalized sampling theory and application to de-interlacing** [5308-71]
C. Ciuhu, G. de Haan, Philips Research Labs. (Netherlands)

712 **Optimization of the predict operator in lifting-based motion-compensated temporal filtering** [5308-73]
G. Pau, C. Tillier, B. Pesquet-Popescu, Ecole Nationale Supérieure des Télécommunications (France)

Google Exhibit No. 1063
Page 1126

Part Two

SESSION 18 IMAGE/VIDEO PROCESSING II

721 **Content-adaptive up-scaling of chrominance using classification of luminance and chrominance data** [5308-74]
M. Zhao, Technische Univ. Eindhoven (Netherlands); P. M. Hofman, Philips Research Labs. (Netherlands); G. de Haan, Technische Univ. Eindhoven (Netherlands) and Philips Research Labs. (Netherlands)

731 **Motion detection in color image sequence and shadow elimination** [5308-75]
J. Shen, Univ. Michel de Montaigne Bordeaux III (France)

741 **Y/C separation of composite color video signals using samples with non-opposite subcarrier phases** [5308-76]
N. Cordes, G. de Haan, Philips Research Labs. (Netherlands)

753 **Motion deblurring based on time-evolution of simultaneous nonlinear reaction-diffusion** [5308-77]
T. Saito, H. Harada, T. Komatsu, Kanagawa Univ. (Japan)

SESSION 19 IMAGE AND VIDEO APPLICATIONS I

765 **Closed-loop dialog model of face-to-face communication with a photo-real virtual human** [5308-78]
B. Kiss, B. Benedek, G. Szijártó, VerAnim (Hungary); B. Takács, Digital Elite Inc. (USA)

773 **Multistage feature extraction for accurate face alignment** [5308-79]
F. Zuo, Technische Univ. Eindhoven (Netherlands); P. H. N. de With, Technische Univ. Eindhoven (Netherlands) and Logica CMG (Netherlands)

782 **Face reorientation in video conferencing by approximating normal distributed depth** [5308-80]
B. Yip, J. S. Jin, Univ. of Sydney (Australia)

789 **Digital item for digital human memory—television commerce application: family tree albuming system** [5308-81]
J. Song, H. Lee, J. W. Hong, Electronics and Telecommunications Research Institute (South Korea)

SESSION 20 IMAGE AND VIDEO APPLICATIONS II

799 **Pixel- and compressed-domain color matching techniques for video mosaic applications** [5308-83]
M.-S. Lee, M. Shen, C.-C. J. Kuo, Univ. of Southern California (USA)

Google Exhibit No. 1063
Page 1127

808 **Multispectral image compression for high-quality color reproduction using JPEG2000** [5308-84]
R. Mase, Y. Kawasaki, Tokyo Institute of Technology (Japan); Y. Murakami, Tokyo Institute of Technology (Japan) and Telecommunications Advancement Organization of Japan (Japan); T. Obi, Tokyo Institute of Technology (Japan); M. Yamaguchi, N. Ohyama, Tokyo Institute of Technology (Japan) and Telecommunications Advancement Organizationn of Japan (Japan)

816 **Model-based quality enhancement of scalable video** [5308-85]
G. Feideropoulou, B. Pesquet-Popescu, Ecole Nationale Supérieure des Télécommunications (France)

POSTER SESSION

824 **Multispectral image fusion based on fractal features** [5308-86]
J. Tian, Institute of Acoustics (China); J. Chen, Beijing Institute of Technology (China); C. Zhang, Institute of Acoustics (China)

833 **Interpolative mode prediction for efficient MPEG-2 video encoding** [5308-87]
B. C. Song, K. W. Chun, Samsung Electronics Co., Ltd. (South Korea)

841 **Moving object detection using orthogonal Gaussian-Hermite moments** [5308-88]
Y. Wu, J. Shen, Univ. Michel de Montaigne Bordeaux III (France)

850 **Creating interactive 3D media with projector-camera systems** [5308-89]
N. L. Chang, Hewlett-Packard Labs. (USA)

862 **Spatially varying motion prediction algorithm for video codec** [5308-90]
J. W. Jeong, M.-C. Hong, Soongsil Univ. (South Korea)

871 **Turbo trellis coded modulation with two typical mapping strategies of 16QAM in image transmission** [5308-91]
Z. Bai, D. Yuan, Shandong Univ. (China)

876 **Possibilistic-clustering-based MR brain image segmentation with accurate initialization** [5308-92]
Q. Liao, Tsinghua Univ. (China) and Univ. de Caen (France); Y. Deng, Tsinghua Univ. (China); W. Dou, S. Ruan, D. Bloyet, Univ. de Caen (France)

881 **Robust techniques for background subtraction in urban traffic video** [5308-93]
S.-C. S. Cheung, C. Kamath, Lawrence Livermore National Lab. (USA)

893 **Object-based image restoration for multilayer autofocusing** [5308-94]
K. Kim, S. Hwang, J. Shin, J. Paik, Chung-Ang Univ. (South Korea); B. Abidi, M. A. Abidi, Univ. of Tennessee/Knoxville (USA)

902 **Error-resilient video coding using adaptive weighted multiple reference frame method** [5308-95]
S. H. Kim, Seoul National Univ. (South Korea); Y. K. Kim, Hoseo Univ. (South Korea); S.-U. Lee, Seoul National Univ. (South Korea)

Google Exhibit No. 1063
Page 1128

912 **Robust camera motion estimation and classification for video analysis** [5308-96]
H.-C. Chang, S.-H. Lai, National Tsing Hua Univ. (Taiwan)

924 **Tradeoff between picture resolution and quantization precision in video coding for embedded systems** [5308-97]
Y. Yuan, Tsinghua Univ. (China); D. Feng, Univ. of Sydney (Australia); Y. Zhong, Tsinghua Univ. (China)

932 **Early-stop and motion vector reuse for MPEG-2 to H.264 transcoding** [5308-98]
M. Kucukgoz, M.-T. Sun, Univ. of Washington (USA)

937 **Scalable texture-based video coding** [5308-100]
D. Chavira-Martínez, S. Pateux, Institut National de Recherche en Informatique et en Automatique (France)

946 **Hybrid image coding for real-time computer screen video transmission** [5308-101]
T. Lin, Peking Univ. (China); P. Hao, Queen Mary Univ. of London (United Kingdom) and Peking Univ. (China); C. Xu, J.-F. Feng, Peking Univ. (China)

958 **Modeling the user preference on broadcasting contents using Bayesian networks** [5308-102]
S. Kang, J. Lim, M. Kim, Information and Communications Univ. (South Korea)

968 **Automated classification of female facial beauty by image analysis and supervised learning** [5308-103]
H. Gunes, M. Piccardi, T. Jan, Univ. of Technology/Sydney (Australia)

979 **Fast wavelet transform domain texture synthesis** [5308-104]
D. S. Wickramanayake, E. A. Edirisinghe, H. E. Bez, Loughborough Univ. (United Kingdom)

988 **Camera-based man-machine interface for computer application control** [5308-105]
G. D. Morrison, SMART Technologies, Inc. (Canada)

996 **Effective quality analysis for video streaming over wireless ad hoc network** [5308-106]
Z. He, Univ. of Missouri/Columbia (USA); C. W. Chen, Florida Institute of Technology (USA)

1007 **Improved high-definition video by encoding at an intermediate resolution** [5308-107]
A. Segall, Pixonics, Inc. (USA); M. Elad, Technion-Israel Institute of Technology (Israel) and Consultant (USA); P. Milanfar, Univ. of California/Santa Cruz (USA) and Consultant (USA); R. Webb, C. Fogg, Pixonics, Inc. (USA)

1019 **Robust traffic event extraction from surveillance video** [5308-108]
A. Yoneyama, KDDI R&D Labs. Inc. (Japan); C. H. Yeh, C.-C. J. Kuo, Univ. of Southern California (USA)

1031 **Lossless compression of point-based data for 3D graphics rendering** [5308-109]
J.-Y. Sim, Seoul National Univ. (South Korea); C.-S. Kim, Chinese Univ. of Hong Kong (China); S.-U. Lee, Seoul National Univ. (South Korea)

1043 **Computation reduction for motion estimation in MPEG-4 video encoder** [5308-110]
C. Chen, R. Li, Intel China (China)

Google Exhibit No. 1063
Page 1129

1049 **Accurate bit rate and quality control for 3D multiview sequences** [5308-111]
J. Lim, K. Sohn, Yonsei Univ. (South Korea)

1060 **Distributed single source coding with side information** [5308-112]
J. E. Vila-Forcen, O. Koval, S. V. Voloshynovskiy, Univ. of Geneva (Switzerland)

1073 **Adaptive search range decision algorithm for fast motion estimation** [5308-113]
W. S. Song, M.-C. Hong, Soongsil Univ. (South Korea)

1082 **Assessment of the compression efficiency of the MPEG-4 AVC specification** [5308-114]
W. De Neve, P. Lambert, S. Lerouge, Ghent Univ. (Belgium); R. Van de Walle, Ghent Univ. (Belgium) and IMEC (Belgium)

1094 **JPEG 2000 coding of textured parametric surfaces** [5308-115]
M. N. Gamito, M. S. Dias, Instituto Superiour das Ciências do Trabalho e da Empresa (Portugal)

1106 **Image enhancement circuit using nonlinear processing curve and constrained histogram range equalization** [5308-116]
S. D. Cvetkovic, Bosch Security Systems BV (Netherlands); P. H. N. de With, LogicaCMG (Netherlands) and Technische Univ. Eindhoven (Netherlands)

1117 **Bit-rate prediction for look-ahead coding with AVC** [5308-117]
M. Beermann, RWTH-Aachen (Germany)

1127 **Watershed segmentation and region merging** [5308-118]
R. L. V. P. M. Pires, P. De Smet, W. Philips, Ghent Univ. (Belgium)

1136 **Analysis of multihypothesis motion-compensated prediction for error resilient video transmission** [5308-119]
W.-Y. Kung, C.-S. Kim, C.-C. J. Kuo, Univ. of Southern California (USA)

1148 **A multipath video delivery scheme over diffserv wireless LANs** [5308-120]
H. Man, Y. Li, Stevens Institute of Technology (USA)

1159 **Progressive compression of PointTexture images** [5308-121]
I.-W. Song, Seoul National Univ. (South Korea); C.-S. Kim, Chinese Univ. of Hong Kong (China); S.-U. Lee, Seoul National Univ. (South Korea)

1169 **A novel stereo image coding algorithm based on delaunay triangulation mesh** [5308-122]
J. Han, Z. Lu, Xidian Univ. (China)

1181 **Improving nonlinear up-scaling by adapting to the local edge orientation** [5308-124]
J. A. P. Tegenbosch, P. M. Hofman, M. K. Bosma, Philips Research Labs. (Netherlands)

1191 **Queue-based block matching algorithm for video compression and motion segmentation** [5308-125]
T.-K. Chiew, Univ. of Bristol (United Kingdom); J. T. H. Chung-How, ProVision Communication Technologies Ltd. (United Kingdom); D. R. Bull, C. N. Canagarajah, Univ. of Bristol (United Kingdom)

Google Exhibit No. 1063
Page 1130

1200 **Online smoothing for scalable media stream delivery** [5308-126]
K. Gao, W. Gao, Institute of Computing Technology (China) and Graduate School of Chinese Academy of Sciences (China); S. He, Institute of Computing Technology (China); Y. Zhang, Graduate School of Chinese Academy of Sciences (China) and Beijing Broadcasting Institute (China)

1209 **Optimal reconstruction value for DCT dequantization using Laplacian pdf model** [5308-127]
S.-Y. Kang, B.-U. Lee, Ewha Womans Univ. (South Korea)

1218 **Low-rate video coding with block reordering in wavelet domain** [5308-128]
P.-C. Chang, T.-T. Lu, National Central Univ. (Taiwan)

1226 **Arithmetic coding with adaptive context-tree weighting for the H.264 video coders** [5308-129]
D. Hong, Columbia Univ. (USA); M. van der Schaar, Univ. of California/Davis (USA); B. Pesquet-Popescu, Ecole Nationale Supérieure des Télécommunications (France)

1236 **Layered unequal loss protection for progressive image transmission over packet loss channels** [5308-130]
J. Cai, X. Li, Nanyang Technological Univ. (Singapore); C. W. Chen, Florida Institute of Technology (USA)

1244 **MPEG interlaced video transcoding for a networked video browsing system** [5308-131]
A. Shimizu, Sanyo Electric Co., Ltd. (Japan)

1253 **Edge degradation for objective video quality metrics** [5308-132]
C. Lee, S. Cho, J. Choe, T. Jung, W. Ahn, Yonsei Univ. (South Korea)

1261 **Interpolation-free subpixel refinement for block-based motion estimation** [5308-133]
T.-K. Chiew, Univ. of Bristol (United Kingdom); J. T. H. Chung-How, ProVision Communication Technologies, Ltd. (United Kingdom); D. R. Bull, C. N. Canagarajah, Univ. of Bristol (United Kingdom)

1270 **Framework design on video coding system for error-prone heterogeneous network** [5308-134]
W. Shen, Wuhan Univ. (China); Y. Chong, J. Xiao, WHU Firstlink Digital Technology Co., Ltd. (China); D. Li, Wuhan Univ. (China)

1277 **Two-way decodable variable-length data blocks for robust video transmission** [5308-135]
S.-S. Gao, Nanyang Technological Univ. (Singapore); C. Zhang, State Key Lab. of Information Security (China)

1286 **Tuning of perceptual technique for digital movie color restoration** [5308-136]
A. Rizzi, Univ. degli Studi di Milano (Italy); M. Chambah, Univ. de La Rochelle (France); D. Lenza, Univ. degli Studi di Milano (Italy); B. Besserer, Univ. de La Rochelle (France); D. Marini, Univ. degli Studi di Milano (Italy)

1295 **Universal motion prediction** [5308-137]
Z. Li, E. J. Delp III, Purdue Univ. (USA)

1305 **Wireless FGS video transmission using adaptive mode selection and unequal error protection** [5308-138]
J. Wu, J. Cai, Nanyang Technological Univ. (Singapore)

Google Exhibit No. 1063
Page 1131

1317 **Automatic calibration of multiple cameras on a nonplanar ground for visual surveillance systems** [5308-139]
J. Jung, H. Ki, J. Shin, J. Paik, Chung-Ang Univ. (South Korea)

1325 **Computation of discontinuous optical flow fields based on spatiotemporal bilateral filtering** [5308-140]
R. Feghali, A. Vincent, Communications Research Ctr. Canada (Canada)

1333 **EBCOT coprocessing architecture for JPEG2000** [5308-141]
H. Zhang, J. Fritts, Washington Univ. (USA)

1341 **A hybrid scheme for perceptual object window design with joint scene analysis and eye-gaze tracking for media encoding based on perceptual attention** [5308-142]
J. I. Khan, O. V. Komogortsev, Kent State Univ. (USA)

1353 **Novel precompression rate-distortion optimization algorithm for JPEG 2000** [5308-143]
Y.-W. Chang, H.-C. Fang, C.-J. Lian, L.-G. Chen, National Taiwan Univ. (Taiwan)

1362 **Motion-compensated wavelet video coding using adaptive mode selection** [5308-144]
F. Zhai, T. N. Pappas, Northwestern Univ. (USA)

1371 **Low-level behavioral analysis of the JVT/AVC decoder** [5308-145]
P. Lambert, L. Eeckhout, R. De Sutter, K. De Bosschere, Ghent Univ. (Belgium);
R. Van de Walle, Ghent Univ. (Belgium) and IMEC (Belgium)

1383 **Selecting the don't care bits in JPEG2000 ROI coding** [5308-146]
J. Ameli, J. Vaisey, T. Jin, Simon Fraser Univ. (Canada)

1392 *Author Index*

Google Exhibit No. 1063
Page 1132

Ex. 1029B

CONFERENCE PROCEEDING

Visual communications and image processing 2004 : 20-22 January, 2004, San Jose, California, USA / Sethuraman Panchanathan, Bhaskaran Vasudev, chairs/editors ; sponsored and published by IS&T--the Society for Imaging Science and Technology, SPIE--the International Society for Optical Engineering ; in cooperation with IEEE Circuits and Systems Society

Panchanathan, Sethuraman; Bhaskaran, Vasudev, 1956-; IS & T--the Society for Imaging Science and Technology; Society of Photo-optical Instrumentation Engineers; IEEE Circuits and Systems Society

c2004

Top
Send to
Get It
Details
Virtual Browse
Links

Available at Linda Hall Library Books - 2nd Floor (TA1632 .V56 2004) >

Send to

Email | QR | Permalink | Citation

EasyBib | Export RIS | Export to Excel

https://catalog.lindahall.org/permalink/01LINDAHALL_INST/19lda7s/alma993806573405961

Copy permalink to clipboard

Get It

LOCATION ITEMS

Linda Hall Library
Available , Books - 2nd Floor ; TA1632 .V56 2004

Item in place (0 requests)
Loanable

Item in place (0 requests)
Loanable

Details

Title	Visual communications and image processing 2004 : 20-22 January, 2004, San Jose, California, USA / Sethuraman Panchanathan, Bhaskaran Vasudev, chairs/editors ; sponsored and published by IS&T--the Society for Imaging Science and Technology, SPIE--the International Society for Optical Engineering ; in cooperation with IEEE Circuits and Systems Society
Other title	Proceedings of Electronic Imaging Science and Technology Electronic imaging science and technology Proceedings of SPIE--the International Society for Optical Engineering. v. 5308.
Creator/Contributor	Panchanathan, Sethuraman Bhaskaran, Vasudev, 1956- IS & T--the Society for Imaging Science and Technology Society of Photo-optical Instrumentation Engineers IEEE Circuits and Systems Society
Subjects	Image processing Coding theory
Identifier	ISBN : 9780819452115 ISBN : 0819452114 OCLC : (OCoLC)54456362
Series	Proceedings of SPIE--the International Society for Optical Engineering, v. 5308 SPIE proceedings series ; v. 5308
Publisher	Bellingham, Washington : SPIE
Creation date	c2004
Format	2 v. : ill. ; 28 cm.
General notes	"Proceedings of Electronic Imaging Science and Technology"--T.p. Includes bibliographic references and author index.

Virtual Browse




al munícati | Visual communicati | Visual communicati | Visual commu



Links

Visual communications and image processing 2004 (Proceedings of SPIE--the International Society for Optical Engineering. v. 5308)

Linda Hall Library MARC record copied from https://catalog.lindahall.org/discovery/sourceRecord?vid=01LINDAHALL_INST:LHL&docId=alma993806573405961&recordOwner=01LINDAHALL_INST

```
leader  01605nam a2200349Ia 4500
001     993806573405961
005     20040220132000.0
008     040220s2004    waua     b    101 0 eng d
020     ##$a9780819452115
020     ##$a0819452114
035     ##$a(OCoLC)54456362
035     ##$9527035
035     ##$a(MoKL)380657-lhalldb
035     ##$a(lhalldb)380657-lhalldb
040     ##$aLHL $cLHL
049     ##$aLHLA
090     ##$aTA1632 $b.V56 2004
245     00$aVisual communications and image processing 2004 : $b20-22 January,
2004, San Jose, California, USA / $cSethuraman Panchanathan, Bhaskaran
Vasudev, chairs/editors ; sponsored and published by IS&T--the Society for
Imaging Science and Technology, SPIE--the International Society for Optical
Engineering ; in cooperation with IEEE Circuits and Systems Society.
246     30$aProceedings of Electronic Imaging Science and Technology
246     30$aElectronic imaging science and technology
260     ##$aBellingham, Washington : $bSPIE, $cc2004.
300     ##$a2 v. : $bill. ; $c28 cm.
490     1#$aSPIE proceedings series ; $vv. 5308
500     ##$a"Proceedings of Electronic Imaging Science and Technology"--T.p.
504     ##$aIncludes bibliographic references and author index.
650     #0$aImage processing $vCongresses
650     #0$aCoding theory $vCongresses
700     1#$aPanchanathan, Sethuraman
700     1#$aBhaskaran, Vasudev, $d1956-
710     2#$aIS & T--the Society for Imaging Science and Technology
710     2#$aSociety of Photo-optical Instrumentation Engineers
710     2#$aIEEE Circuits and Systems Society
830     #0$aProceedings of SPIE--the International Society for Optical
Engineering. $vv. 5308.
948     ##$aLTI 09/01/2008
```

Portland State University

PDXScholar

Computer Science Faculty Publications and Presentations

Computer Science

12-1997

Flow and Congestion Control for Internet Streaming Applications

Shanwei Cen

Calton Pu

Jonathan Walpole
Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/compsci_fac

 Part of the Computer Sciences Commons, and the Digital Communications and Networking Commons

Let us know how access to this document benefits you.

Citation Details

Shanwei Cen ; Jonathan Walpole and Calton Pu, "Flow and congestion control for Internet media streaming applications", Proc. SPIE 3310, Multimedia Computing and Networking 1998, 250 (December 29, 1997); doi:10.1117/12.298426; http://dx.doi.org/10.1117/12.298426.

This Article is brought to you for free and open access. It has been accepted for inclusion in Computer Science Faculty Publications and Presentations by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

Google Exhibit No. 1032
Page 001

Google Exhibit No. 1063
Page 1139

Flow and Congestion Control for Internet Media Streaming Applications

Shanwei Cen[a], Jonathan Walpole[b] and Calton Pu[b]

[a]Tektronix, Inc., Video and Networking Division
P.O.Box 500, M/S 50-490 Beaverton, Oregon 97077 USA

[b]Department of Computer Science, Oregon Graduate Institute
P.O.Box 91000, Portland, Oregon 97291 USA

ABSTRACT

The emergence of *streaming* multimedia players provides users with low latency audio and video content over the Internet. Providing high-quality, best-effort, real-time multimedia content requires adaptive delivery schemes that fairly share the available network bandwidth with reliable data protocols such as TCP. This paper proposes a new flow and congestion control scheme, SCP (Streaming Control Protocol), for real-time streaming of continuous multimedia data across the Internet. The design of SCP arose from several years of experience in building and using adaptive real-time streaming video players. SCP addresses two issues associated with real-time streaming. First, it uses a congestion control policy that allows it to share network bandwidth fairly with both TCP and other SCP streams. Second, it improves smoothness in streaming and ensures low, predictable latency. This distinguishes it from TCP's jittery congestion avoidance policy that is based on linear growth and one-half reduction of its congestion window. In this paper, we present a description of SCP, and an evaluation of it using Internet-based experiments.

Keywords: Internet, flow control, congestion control, adaptive systems, streaming multimedia systems

1. INTRODUCTION

The real-time distribution of continuous audio and video data via streaming multimedia applications accounts for a significant, and expanding, portion of the Internet traffic. Many research prototype media players have been produced, including Rowe's MPEG player,[1] the authors' distributed video player,[2] and McCanne's vic,[3] etc. Over the past two years, many commercial streaming media players have also been released, such as RealAudio and RealVideo players, Microsoft Netshow, and Vxtreme.

Common characteristics of media streaming applications include their need for high bandwidth, smooth data flow, and low and predictable end-to-end latency and latency variance. In contrast, the Internet is a best-effort network that offers no quality of service guarantees, and is characterized by a great diversity in network bandwidth and host processing speed, wide-spread resource sharing, and a highly dynamic workload. Hence, in practice, Internet-based applications experience large variations in available bandwidth, latency, and latency variance.

To address these problems the new generation of Internet-based multimedia applications use techniques such as buffering and feedback-based adaptation of presentation quality. Buffering is used at the sender, receiver, or both, to mask short-term variations in the available bandwidth or latency. Through adaptation, applications scale media quality in one or more quality dimensions to better utilize the currently available bandwidth. For example, real-time play-out can be preserved in the presence of degraded bandwidth by adaptively sacrificing other presentation quality dimensions such as video frame rate or spatial resolution.

Dynamic adaptation is a powerful approach, but it requires new flow and congestion control mechanisms for accurate discovery and appropriate utilization of the available network bandwidth. Congestion control mechanisms must determine, dynamically, the share of the network bandwidth that can be fairly used by the adaptive application in the presence of competing traffic. If the mechanisms are not sensitive enough

Please send correspondence to Shanwei Cen at shanwei.cen@tek.com

SPIE Vol. 3310 • 0277-786X/97/$10.00

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 002

Google Exhibit No. 1063
Page 1140

to competing traffic the potentially high multimedia data rates could cause serious network congestion. On the other hand, if they are too sensitive they will under-utilize the bandwidth and presentation quality will be forced to degrade unnecessarily. In practice, such congestion control mechanisms must share bandwidth fairly among competing multimedia streams and must be compatible with TCP congestion control,[4] since TCP is the base protocol for currently-dominant HTTP and FTP traffic.

Flow control mechanisms should attempt to minimize end-to-end latency due to buffering, and maximize the smoothness of the data stream. Reliability through indefinite data retransmission is usually not desirable, since streaming applications can often tolerate some degree of quality degradation due to data loss, but are usually less tolerant of the delay introduced by the retransmission of lost data.

This paper presents a flow and congestion control scheme, called SCP (Streaming Control Protocol), for unicast media streaming applications. Like TCP, SCP employs sender-initiated congestion-detection through positive acknowledgment, and uses a similar window-based policy for congestion avoidance. The similarities in congestion avoidance between SCP and TCP make SCP robust and a good network citizen, and enable SCP and TCP to share the Internet fairly.

Unlike TCP, when the network is not congested, SCP invokes a hybrid rate- and window-based flow control policy that maintains smooth streaming with maximum throughput and low latency. Conversely, TCP repeatedly increases its congestion window size until packets are lost, then halves its window size (a behavior of TCP-Reno). One consequence of this behavior is that TCP sessions exhibit burstiness and develop long end-to-end latency due to the build up of packets in network router buffers. This behavior is particularly problematic over PPP links since some PPP servers have buffers that hold 15 seconds of data, or more. Also unlike TCP, SCP does not retransmit data lost in the network. Thus it avoids the associated unpredictability in latency and wasted bandwidth.

Further salient features of SCP include its support for rapid adaptation in the presence of drastic bandwidth fluctuations, such as those that occur when mobile computers migrate among different network types,[5] and its support for streaming-specific operations such as pausing.

This paper is organized as follows. Section 2 presents the design of SCP. Section 3 follows with an analysis of SCP's steady-state bandwidth sharing. Section 4 describes the implementation of SCP. Section 5 explains the experimental results. Section 6 then briefly discusses other congestion control schemes, and finally, Section 7 concludes the paper and discusses future work.

2. THE DESIGN OF SCP

A unicast streaming scenario with SCP involves a sender, which streams media packets over a network connection in real-time to a receiver. SCP policies are implemented at the sender side. Each packet contains a sequence number, and for each packet, SCP records the time at which it is sent, and initiates a separate timer*. The receiver acknowledges each packet it receives by returning an ACK containing the sequence number of the packet to the sender. Based on the reception of ACKs and expiration of timers, SCP monitors the available bandwidth, detects packet loss, and adjusts the size of its congestion window to control the flow and avoid network congestion. To achieve this task SCP maintains the following internal state variables and parameter estimators.

- *state* — The current state. SCP has several states, each corresponding to a specific network and session condition and associated flow and congestion control policy.
- W_t — The size of the congestion window (in number of packets).
- W — The number of outstanding packets sent but not acknowledged. When $W < W_t$, the congestion window is open, and more packets can be sent, otherwise it is closed and no packets can be sent.
- W_{ss} — The threshold of W_t for switching from the slow-start policy to the steady-state policy.
- $\hat{T}_{brtt}$ — An estimator of the base RTT (round trip time) – the RTT of a packet sent when the network is otherwise quiet.
- $\hat{T}_{rtt}$ — An estimator of the recent average RTT.
- $\hat{D}_{rtt}$ — An estimator of the standard deviation of the recent RTT.

*For clarity, we explain the design of SCP based on a timer per packet, but this aspect is optimized in the implementation.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 003

Google Exhibit No. 1063
Page 1141

State	Network and session condition	Congestion window adjustment policy
slowStart	Available bandwidth not discovered yet	SCP opens the congestion window exponentially by increasing the window size by one upon the receipt of each ACK.
steady	Available bandwidth being fully utilized	SCP maintains appropriate amount of buffering inside the network to gain sufficient throughput, avoid excessive buffering or buffer overflow, and trace the changes in available bandwidth.
congested	The network is congested.	SCP backs off multiplicatively by halving the window size. Persistent congestion results in exponential back-off.
paused	No outstanding packet in the network	When a new packet is sent, SCP shrinks the window size and invokes the slow-start policy.

Table 1. SCP states, network and session conditions, and flow and congestion control policies

- $\widehat{rto}$ — An estimator of the timer duration.
- $\hat{r}$ — An estimator of the receiving packet rate.

As long as the congestion window is open, the sender streams packets at a rate no more than $\frac{W_l}{\hat{T}_{brtt}}$, instead of sending them out in bursts, so as to improve the smoothness of the stream. When an ACK is received, or a timer expires, the congestion window size W_l is adjusted using a policy associated with the current state.

SCP adopts window-based policies similar to those of TCP for slow-start, and exponential back-off upon network congestion. It also estimates RTT in a way similar to TCP. Therefore, many parameters and state variables have counterparts in TCP. However, SCP differs from TCP in that it has a base RTT estimator $\hat{T}_{brtt}$ and a packet rate estimator $\hat{r}$. The use of these estimators to ensure smooth streaming and low latency is discussed in the following sections.

2.1. Overall architecture

SCP is based on the observation that excessive packets in the round-trip network connection accumulate in buffers at network routers and switches and lead to an increased RTT. As the accumulation of packets increases, these buffers overflow and packets are dropped. The goal of SCP is to ensure smooth streaming at a suitably high throughput, but without causing excessive buffering or congestion in the network. To achieve this goal, SPC monitors fluctuations in the available bandwidth and pushes an appropriate number of additional packets into the network connection. If network congestion is detected, SCP reacts immediately with exponential back-off. Depending on the condition of the network and the streaming session under control, SCP is in one of the following four states: *slowStart*, *steady*, *congested* or *paused*. Each state is associated with a specific condition and congestion window size adjustment policy as listed in Table 1.

SCP handles events that indicate changes in network and session conditions. Such events include indications that: the SCP session becomes paused or active; the available network bandwidth is fully utilized; the network is congested; the network interface has been switched. Upon these events, SCP updates its internal state and possibly switches to a new policy. Figure 1 shows the events that SCP handles and its associated state transitions.

2.2. Congestion window adjustment policies and state transitions

Initialization: Upon initialization, SCP sets $W = 0$, $W_l = 1$ and $W_{ss} = L_w$, where L_w is a limit on the congestion window size. $\hat{T}_{brtt}$, $\hat{T}_{rtt}$ and $\hat{D}_{rtt}$ are set to infinity. $\widehat{rto}$ is set to an initial default value. After initialization, SCP enters the *paused* state. Transmission of the first packet brings SCP to the *slowStart* state.

Slow-start: The slow-start policy is invoked after initialization or when SCP resumes from a pause. Its goal is to quickly grow the congestion window and discover the available network bandwidth. W_l is incremented by 1 when an ACK is received in-order, thus doubling the congestion window after each RTT amount of time. SCP leaves the *slowStart* state upon detecting events indicating network congestion, full utilization of available bandwidth, pause in streaming, or network interface switch.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 004

Google Exhibit No. 1063
Page 1142

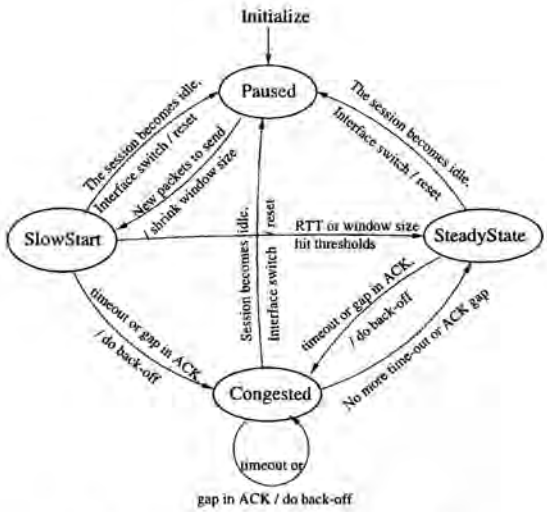


Figure 1. SCP state transition diagram

Steady-state smooth streaming: By pushing an appropriate number of extra packets, the steady-state policy enables sufficient utilization of the available bandwidth while avoiding over-buffering. It also traces the fluctuations in available bandwidth. In the *steady* state, estimation of packet rate $\hat{r}$ is enabled. Whenever a new $\hat{r}$ value is available, congestion window size W_l and threshold W_{ss} are adjusted according to Equation 1 below, where W_Δ is a constant referred to as the window-size incremental coefficient.

$$W_l = W_{ss} = \hat{r}\hat{T}_{brtt} + W_\Delta \tag{1}$$

The idea behind the flow control defined by Equation 1 is that SCP assumes that $\hat{r}$ is an approximation of the network bandwidth (in terms of packet rate) available to the session, calculates the bandwidth-delay product of the network connection with minimum buffering $\hat{r} \times \hat{T}_{brtt}$, and adjusts its congestion window size W_l accordingly. The product $\hat{r} \times \hat{T}_{brtt}$ is the amount of data SCP should keep outstanding inside the network in order to maintain sufficient throughput with minimum buffering. Since the network is shared and highly dynamic, the bandwidth available to the session under control fluctuates. If SCP just sets its congestion window size to $\hat{r} \times \hat{T}_{brtt}$, when the available bandwidth decreases, $\hat{r}$ would decrease, and SCP would reduce its congestion window size and thus the amount of outstanding data. However, if the available bandwidth increases, there is no way for SCP to detect that. This is because the receiving packet rate $\hat{r}$ could not be increased unless SCP increases the sending packet rate first, but SCP would not increase its sending packet rate by increasing the congestion window size unless it observes a increased $\hat{r}$. To solve this "chicken-and-egg" type of problem, SCP pushes W_Δ amount of extra outstanding data, which are held in router and switch buffers. When the network available bandwidth increases, releasing the extra data buffered in the network results in an increase in packet rate $\hat{r}$, which in turn results in larger amount of outstanding data as defined by the increased W_l. It can be seen that the increase in W_l is no more than additive, which is what TCP does in its steady state. Due to the smoothing effect of lowpass filtering in estimation of $\hat{r}$, W_l will be changed gradually, causing smooth changes in throughput. Also, with the flow control policy stated in Equation 1, though at the beginning $\hat{r}$ may not be a reliable estimation of the actual available bandwidth, if the network is stable enough, SCP will eventually bring $\hat{r}$ to the actual bandwidth through iterations. Overall, the steady-state policy is able to converge to a stable throughput, and to trace the fluctuation in available bandwidth smoothly.

Upon detecting network congestion, SCP backs off and enters the *congested* state. SCP enters the *paused* state when the session is paused, or when a switch in network interface is detected.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 005

Google Exhibit No. 1063
Page 1143

The steady-state policy defined by Equation 1 is what makes SCP different from TCP. If network conditions remain the same for a sufficiently long period of time, SCP converges its window size and data rate to stable values. SCP is also able to limit the buildup of end-to-end latency resulted from buffering in routers, regardless of the actual sizes of router buffers. These properties are desirable for media streaming applications. In contrast, TCP repeatedly increases its congestion window size linearly until packets are lost, and then backs off by halving the window size. The result is that TCP traffic is inherently bursty. TCP also tends to fill up router buffers no matter how big they are. A consequence is that across network connections with deep buffers, TCP yields long latencies. The deeper the buffers are, the longer the latencies are.

Exponential back-off upon network congestion: Upon network congestion, SCP invokes an exponential back-off policy similar to that of TCP. Whenever events such as gaps in ACK or timer expirations are detected, SCP backs off multiplicatively by reducing its congestion window size in half. The data rate estimator $\tilde{r}$ is no longer accurate and thus is reset and disabled. If the back-off is triggered by timeout, $\widehat{rto}$ may have been too short and is doubled. When the network is so congested that no packet can get through, with the multiplicative decrease in congestion window size and multiplicative increase in timer duration, SCP backs off exponentially until it virtually stops sending any further packet. This gives a chance for the network to quickly recover from the congestion.

At the time of a back-off, there may already be some outstanding packets inside the network. Loss of these packets does not correctly reflect the network condition after the back-off, and thus should be ignored. The *congested* state is designed for this purpose. After each back-off, SCP enters the *congested* state. Further back-off is disabled until the first packet sent in the current *congested* state is acknowledged, found lost or its timer expired. If the first packet is acknowledged, SCP enters the *steady* state, otherwise another round of back-off is initiated.

Pause when there is no packet to send: In a real-time streaming session, a user may want to pause in the middle. If the sender has no data to send for a while, W eventually decreases to 0. At this moment, the streaming session becomes idle, and SCP enters the *paused* state.

When an SCP session is paused, the bandwidth it previously took will gradually be discovered and taken by other sessions. When the session resumes at a later time, SCP invokes the slow-start policy with a reduced initial congestion window size. Currently, we adopt a ad-hoc policy of halving the congestion window size in every $\hat{T}_{brtt}$ time elapsed in the *paused* state.

Reset upon network interface switch: When either end of an SCP streaming session has its network interface switched (e.g., from Ethernet to cellular modem, or vice versa), the route from the sender to the receiver may be changed, and the new connection may go through links with totally different capacities. SCP handles this dynamic reconfiguration issue by providing a *reset* interface. Upon each network interface switch event, SCP discards the out-dated estimators, ignores all outstanding ACKs, and resumes with the slow-start policy to quickly discover the capacity of the new connection.

3. ANALYSIS OF BANDWIDTH SHARING BETWEEN SCP SESSIONS

With the steady-state rate- and window-based policy defined by Equation 1, it is possible for multiple SCP sessions to share network links in a fair and stable manner. In this section, we analyze a simple case, in which two sessions with the same packet size share a single network link. Both sessions send packets at maximum rate while the congestion window is open.

Suppose that, in a steady state, session A has estimators $\tilde{r}_a$, $\hat{T}_{brtta}$, $\hat{T}_{rtta}$, and $W_{la} = \tilde{r}_a\hat{T}_{brtta} + W_\Delta$; and session B has $\tilde{r}_b$, $\hat{T}_{brttb}$, $\hat{T}_{rttb}$, and $W_{lb} = \tilde{r}_b\hat{T}_{brttb} + W_\Delta$. Since A and B share the same network link, we have following observations:

(1) The aggregate packet rate of the network link, $\tilde{r}_l$, is the sum of the two sessions: $\tilde{r}_l = \tilde{r}_a + \tilde{r}_b$.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 006

Google Exhibit No. 1063
Page 1144

(2) Base RTT estimate is the actual link base RTT T_{brttl} plus some estimation error: $\hat{T}_{brtta} = T_{brttl} + e_a$ and $\hat{T}_{brttb} = T_{brttl} + e_b$. When $\hat{T}_{brtt}$ is estimated as the minimum of the past RTT measurements of a session, the main component of the estimation error is the residual buffering, which is caused by packets of other sessions preventing the network buffers from becoming empty. Another less important component is sampling noise.

(3) Sessions A and B have the same RTT estimator (when the sampling noise can be ignored): $\hat{T}_{rtta} = \hat{T}_{rttb} = \hat{T}_{rttl}$. Furthermore, the number of packets sent by a session in one RTT equals its congestion window size. Thus the packet rate ratio of A and B equals the ratio of their window size:

$$\frac{\bar{r}_a}{\bar{r}_b} = \frac{W_{la}}{W_{lb}} = \frac{\bar{r}_a(T_{brttl} + e_a) + W_\Delta}{\bar{r}_b(T_{brttl} + e_b) + W_\Delta} \quad \Longrightarrow \quad \bar{r}_a = \frac{\bar{r}_b W_\Delta}{\bar{r}_b(e_b - e_a) + W_\Delta} \tag{2}$$

Combining observations (1) and (3), it is clear that there is a single solution for $\bar{r}_a$ and $\bar{r}_b$. The session with a larger residual buffering tends to have a larger estimation error, thus gets a larger portion of the link bandwidth. In a special case where the two sessions have the same RTT estimation error, $e_a = e_b$, we have $\bar{r}_a = \bar{r}_b = \frac{\bar{r}_l}{2}$, indicating that the two sessions split the link bandwidth evenly.

4. IMPLEMENTATION OF SCP

SCP has been implemented as a layer on top of UDP, based on a toolkit for developing adaptive systems.[6] In this section, we briefly discuss the parameter estimators and various implementation issues. More details may be found in Cen's thesis.[6]

Several parameters are estimated by SCP. Base RTT $\hat{T}_{brtt}$, average RTT $\hat{T}_{rtt}$, RTT standard deviation $\hat{D}_{rtt}$, and timer duration $\widehat{rto}$ are estimated based on the history of the raw RTT measurements. $\hat{T}_{brtt}$ is estimated as the minimum of all the past RTT measurements. The estimations $\hat{T}_{rtt}$, $\hat{D}_{rtt}$ and $\widehat{rto}$ are done in a way similar to those in TCP.[4] The receiving packet rate $\bar{r}$ is estimated as the rate at which ACKs are received. This is reasonable since the receiver acknowledges every packet it receives.

Several performance issues have been addressed in the implementation. The first is that, in the design of SCP discussed in Section 2, each outstanding packet has a separate timer. The overhead of maintaining these timers would make SCP non-scalable. In the implementation, only an ordered table is maintained to record the sending time of all outstanding packets. Upon receipt of an ACK, SCP removes the entry of the acknowledged packet from the table. Only when there is a new packet to send and the congestion window is closed, SCP checks the older packets in the table to see if their timers should have been expired, and performs back-off if necessary. Other issues, such as out-of-order packet delivering, congestion-window adjustment in low-data-rate sessions, etc., are also addressed.

5. EXPERIMENTS AND RESULTS

To evaluate the performance of SCP, a test program has been built to stream dummy packets between Internet hosts using SCP or TCP. The test program reads parameters and experimental scripts from files, and collects statistics for analysis.

The network configuration for the experiments is shown in Figure 2. We have two LANs, one at Oregon Graduate Institute (OGI) and the other at Georgia Institute of Technology (Georgia Tech) connected through the long-haul Internet with 29 hops. Among the hosts, ***anquetil*** is a Linux PC notebook on subnet 1 of the OGI LAN. This notebook can optionally be moved to subnet 2 or a $28.8kb/s$ PPP link. This configuration covers typical types of Internet connections such as phone lines, LAN, and WAN. Network interface switching in mobile environments is simulated by having *anquetil* to switch between its Ethernet and PPP interfaces dynamically.

Experiments have been performed to evaluate the performance of SCP in sessions over different network connections. For all the experiments, unless stated explicitly, the following parameter values are used: steady-state congestion window size incremental coefficient $W_\Delta = 3$, and data packet size $1472B$[†]. The size

[†]IP packet size $= 20 + 8 + 1472 = 1500B$ is the Ethernet MTU. $1472B$ is the UDP/TCP payload size, including the SCP header (in the case of SCP), the various fields carried in the packet, followed by a block of randomized data.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 007

Google Exhibit No. 1063
Page 1145

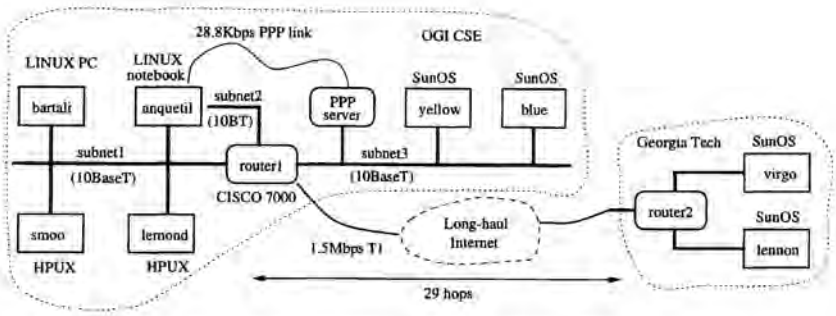
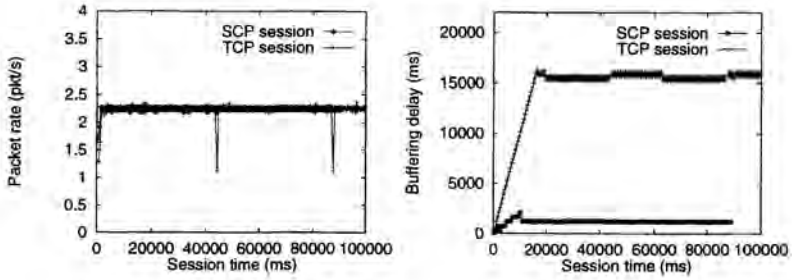


Figure 2. Network configuration for experiments



(a) Packet rate (b) Buffering delay

Figure 3. Performance of single SCP and TCP sessions from *lemond* to *anquetil* over PPP

of TCP and SCP socket sending and receiving buffers is up to $64KB$, which is the maximum supported in many Berkeley socket implementations. The value of W_Δ is selected in an ad-hoc manner, but experiments have shown that it yield satisfactory performance.

In every SCP or TCP session, the sender uses non-blocking mode to send packets. It repeatedly waits until the socket is ready to send (in the case of TCP) or the congestion window is open (in the case of SCP), timestamps a dummy packet, and sends it out immediately. Each packet carries a sender timestamp. The receiver records the time each packet is received. With the receiving timestamps, intervals between packets and packet rate can be measured. Assuming that the clocks of the sender and the receiver have the same rate, with the sender and receiver timestamps, it is possible to measure the application-level end-to-end *buffering delay* — the delay from when a packet leaves the sender application until when it is received by the receiver application, minus the minimum of all the delay measurements, which is the transmission and processing delay. For each packet, the buffering delay is measured as the difference between the sender and receiver timestamps, minus the minimum of all the measurements in the history. This end-to-end buffering delay include the times spent in the buffers in the sender, the receiver, and routers and switches. For all TCP and SCP sessions, buffering delay and receiving *packet rate* are measured. For SCP sessions, the congestion window size, and the gaps in ACKs are also measured.

5.1. Experiments over PPP link

In the first set of experiments, packets are streamed from *lemond* to *anquetil* over the $28.8kb/s$ PPP link with a MTU of $1500B$.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 008

Google Exhibit No. 1063
Page 1146

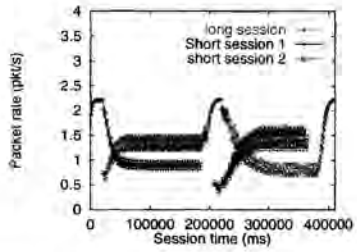


Figure 4. Smoothed packet rate of two SCP sessions from *bartali* to *anquetil* over PPP link

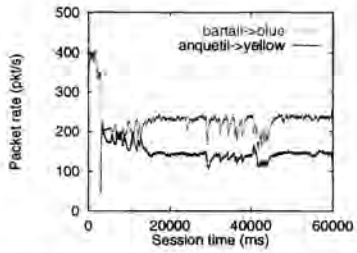


Figure 5. Smoothed packet rate of two SCP sessions through single router

First, single TCP and SCP sessions are played. For the SCP sessions, the receiving data rate is about $2.24pkt/s \approx 26.9kb/s$, which is close to the $28.8kb/s$ PPP link speed. Figure 3(a) shows the packet rate of a SCP session and a TCP session over time. It shows the raw packet rate (inverse of packet interval) versus the session time (the time relative to the beginning of the session). Figure 3(b) shows the buffering delay of these two sessions. After a startup phase, the SCP session yields a smooth packet rate, and utilizes the bandwidth sufficiently. The TCP session has similar data rate, except for the periodic downward spikes which are caused by packet loss (by the PPP server). However, SCP and TCP sessions maintain different levels of buffering inside the network. The SCP session has a steady-state buffering delay of about 1.2 seconds. In contrast, the TCP session pushes the buffering delay up to 16 seconds and stays there. As a matter of fact, different TCP implementations have very different behaviors over PPP. SunOS TCP has a steady-state behavior similar to that of HP-UX TCP, but a much worse start-up phase. Linux TCP pushes the delay to 7 seconds and stays there. Long network buffering delay for TCP over PPP is extremely detrimental to interactive streaming traffic. In an interactive streaming video presentation, each time the user changes playback speed, it is simply not tolerable to see no effect until after more than 16 seconds.

Careful readers may have noticed that 16 seconds is roughly the time for transferring $64KB$ (sender buffer size) of data over the $28.8kb/s$ PPP link. But this long delay is actually caused by a combination of PPP link's low bandwidth and the relatively large buffer in PPP server for individual PPP links. In our experiments, when UDP packets are sent from *lemond* or other hosts to *anquetil* over the PPP link, more than 15 seconds of data will be buffered in the PPP server before packets are dropped. A TCP connection over the PPP link will have its congestion window increased until either the sender or receiver socket buffer is full, or when packets are dropped when the buffer in PPP server overflows. To see that the long buffering delay is not caused by the big sender buffer itself, we also measured several TCP sessions in the reverse direction, from *anquetil* to *lemond*, and got buffering delays of about 2.5 seconds. It seems that LINUX has incorporate some techniques to reduce the latency for TCP over PPP to 7 seconds. Of cause, buffering delay can be reduced by reducing the TCP sender socket buffer size. But small sender socket buffer size is not always desirable. It will limit the size of congestion window, and will not be able to explore the bandwidth of long and fat network connections. There is not a single socket buffer size which fits all types of connections.

Figure 4 shows the smoothed[‡] packet rate of three SCP sessions, when two of them are played simultaneously from *bartali* to *anquetil* over the PPP link. It can be seen that the two competing sessions eventually reach a stable share of the PPP bandwidth. This experiment also shows the impact of residual buffering error in the based RTT estimation on bandwidth sharing. The base RTT estimation of the first half of the long session has a residual buffering caused by the first short session, thus it gets a larger share of the bandwidth. This result supports the observation in the bandwidth sharing analysis (Section 3). Later, after the first short session ends and the second short session starts, the latter gets a residual buffering error caused by the long session, and also gets a bigger portion of the PPP bandwidth. Further experiments show that if the start times of two competing SCP sessions are close enough, they will split the bandwidth evenly.

[‡]Smoothing is achieved by applying a lowpass filter to the raw packet rate measurements.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 009

Google Exhibit No. 1063
Page 1147

Experiment configuration	Single SCP *anquetil→yellow*	Single TCP *anquetil→yellow*	SCP: *anquetil→yellow* SCP: *bartali→blue*	SCP: *anquetil→yellow* TCP: *bartali→blue*
Packet rate (pkt/s)	320	380	150 : 230	SCP 100, TCP 300
Buffering delay (ms)	0~30 around 5	5~80 around 40	0~40 around 14	SCP 0~50, TCP 10~80

Table 2. Results of single TCP and SCP sessions through a single router

Experiments have also been carried out to play one SCP session and one TCP session simultaneously over the PPP link. Since SCP limits the amount of network buffering, the bandwidth partition depends on how aggressive a specific TCP implementation is. For sessions from *bartali* to *anquetil* the packet rates are $0.6pkt/s$ for SCP and $1.7pkt$ for TCP. HP-UX and SunOS TCP implementations are more aggressive. When SCP and TCP sessions are played from *lemond* to *anquetil*, virtually all bandwidth is taken by the TCP session.

5.2. Experiments on single subnet

All the single SCP sessions played from *bartali* to *anquetil* on subnet 1 yield smooth and stable throughput of about $700pkt/s = 8.4Mb/s$. The buffering delays of these sessions, primarily caused by MAC-level collisions and back-offs, remain within $4ms$. For comparison, individual TCP sessions yield $670 \sim 680pkt/s$ and up to $40ms$ of buffering delay. This indicates that SCP is efficient, smooth, and has low latency.

When two SCP sessions, one SCP and one TCP, or two TCP sessions are played *bartali→smoo* and *anquetil→lemond* on subnet 1, the two SCP and TCP sessions are able to share the Ethernet, but they are all very jerky, with highly variable throughput ($100pkt/s$ to $700pkt/s$) and buffering delay (up to 0.8 second). SCP also drops packets. This jerky throughput and packet drop is the result of severe MAC-level collisions and back-off which can not be detected by the sender application. Not much can be done by SCP or TCP to eliminate this problem except for explicitly limiting the data rate at the sender side.

5.3. Experiments through single router

To evaluate the performance of SCP streaming through a single router, *anquetil* is moved to subnet 2. Table 2 shows the configurations and the overall performance of the experiments. Figure 5 shows the smoothed packet rates of two simultaneous SCP sessions. These experiments were run late in the evening when the network was lightly-loaded.

A single SCP session yields a lower throughput than a single TCP session, but has a significantly lower buffering delay. Possibly reasons for SCP to have a lower bandwidth are that SCP sends more ACKs (one per packet), and maintains lower level of buffering thus the router buffers may be empty for more time.

Figure 5 shows that when two SCP sessions are competing for the output port of the single router, they are able to share the bandwidth in a stable manner. To understand why there is a packet rate difference between the two sessions, a closer look at the statistical data reveals that the two sessions have the same steady-state congestion window size of 5, but they have a different RTT. When played separately, the RTT for sessions *bartali→blue* and *anquetil→yellow* are around $15ms$ and $22ms$ respectively. When played simultaneously, they are around $22ms$ and $35ms$ respectively. The difference in RTT may be caused by the differences in host CPU speeds and other factors, and results in an uneven bandwidth partitioning. This doesn't mean that SCP itself is unfair.

When an SCP session competes against a TCP session, they are able the share the bandwidth. But SCP's steady-state control of network buffering make it less aggressive than TCP and gets a significantly smaller amount (1/4) of the bandwidth.

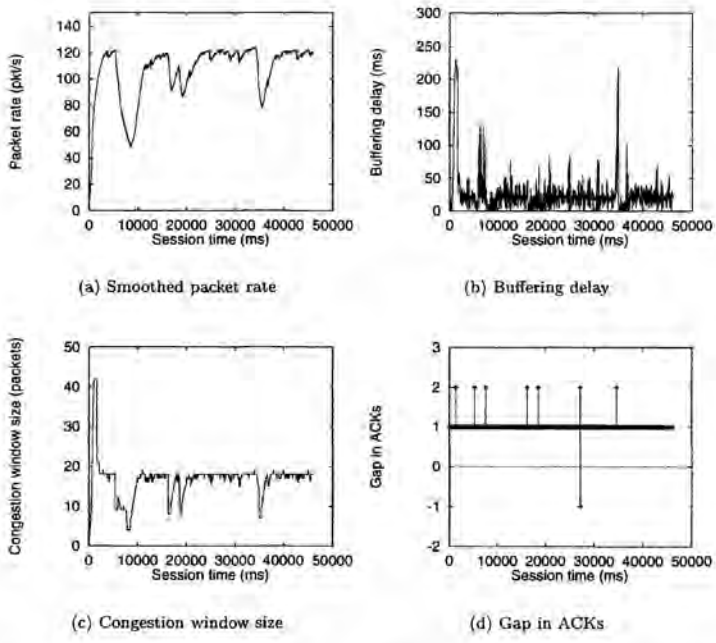
5.4. Experiments across the Internet

To evaluate the performance of SCP across the long-haul Internet, SCP and TCP sessions are played between hosts at OGI and Georgia Tech. There is no way to control the traffic in the Internet. Nevertheless, there are still times, such as midnights and weekends, when the Internet is relatively lightly loaded, and times such as weekdays when it is heavily loaded.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 010

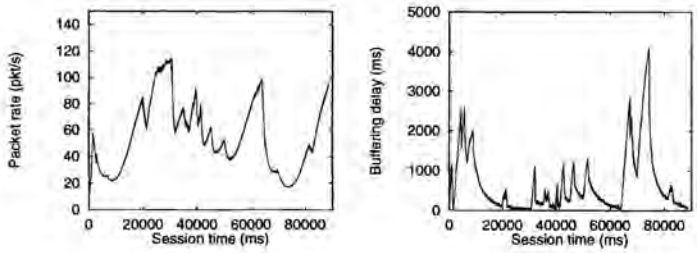
Google Exhibit No. 1063
Page 1148



(a) Smoothed packet rate (b) Buffering delay

(c) Congestion window size (d) Gap in ACKs

Figure 6. Performance of a single SCP session across the lightly loaded Internet



(a) Smoothed packet rate (b) Buffering delay

Figure 7. Performance of a TCP session across the lightly loaded Internet

As shown in Figure 6, when the network is lightly loaded, SCP is able to stream smoothly with low buffering delay, and to explore the network bandwidth sufficiently. From time to time, there are still periods of network congestion and packet loss. Figure 6(c) and (d) identify times when packets are lost, causing SCP to back off. The negative ACK gap shown in Figure 6(d) indicates an ACK delivered out of order. In this case, W_I is halved and then restored before any further packet is sent, thus the back-off is not shown in Figure 6(c). When packets are not dropped, SCP's steady-state rate- and window-based flow control policy maintains a stable congestion window size of about 18 packets, a stable throughput of about $115pkt/s (\approx 1.4Mb/s$, close

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 011

Google Exhibit No. 1063
Page 1149

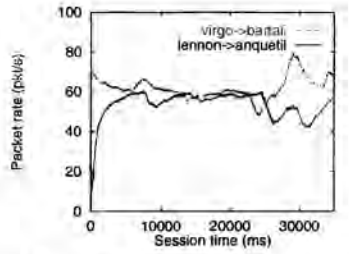


Figure 8. Smoothed packet rate of two SCP sessions across the Internet

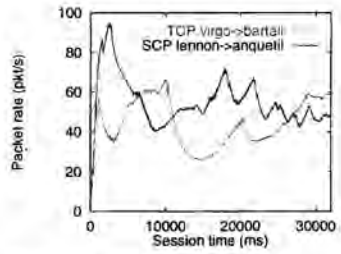


Figure 9. Smoothed packet rate of SCP and TCP sessions across the Internet

to the $1.5Mb/s$ T1 line speed), and low buffering delay (mostly lower than $50ms$). On the other hand, as shown in Figure 7, a single TCP session has more erratic throughput plus long and unpredictable buffering delays (up to 4 seconds)[§].

Two simultaneous SCP sessions are able to share the bandwidth in a fair manner. In our experiment, two competing SCP sessions: *lennon→anquetil* and *virgo→bartali* are played across the lightly loaded Internet. The ratio of the average bandwidth goes from $70pkt/s : 40pkt/s$ to $60pkt/s : 60pkt/s$. The buffering delay for the sessions are kept mostly below $100ms$. Figure 8 shows the packet rate of two simultaneous SCP sessions. Random packet drop causes competing sessions to back-off randomly, but on average the bandwidth sharing is fair, and the throughput is relatively smooth.

SCP and TCP sessions across the long-haul Internet share the network bandwidth more evenly than when through a single router. Figure 9 shows the packet rates of two simultaneous SCP and TCP sessions. The two competing sessions, SCP: *lennon→anquetil* and TCP: *virgo→bartali*, produce a throughput ratio around $SCP50pkt/s : TCP40pkt/s$. While TCP sessions have long latency, the buffering delay of the SCP sessions are still below $100ms$ for most packets.

During busy weekdays, it is more obvious that SCP yields smoother throughput and lower and more consistent delay than TCP, while still operating in harmony with the latter. The performance results of two SCP and TCP sessions from *virgo* and *bartali* are shown in Figure 10. The network is so congested that even "`ping virgo`" from *bartali* shows up to 10% packet loss. SCP sessions consistently get about $14pkt/s$ with buffering delays[¶] below $100ms$. SCP sessions observed up to 10% loss in ACKs. Due to the heavy packet loss, TCP sessions get only 2 to $6pkt/s$, with buffering delays up to 70 seconds[‖], and the throughput is very jerky. It is true that 10% packet loss will reduce the video presentation quality significantly, but a 70 second delay is simply intolerable to any streaming application!

5.5. Experiments with network interface switch

To evaluate SCP's ability to adapt to different network capacities resulted from network interface switches, both the PPP and Ethernet interfaces of the notebook *anquetil* are activated, and SCP sessions are played from *bartali* to *anquetil* through either the PPP or the Ethernet interface. Switching between PPP and Ethernet is simulated by reconfiguring the default IP route and sending a HUP signal to the receiver of the on-going SCP session. When the receiver gets a HUP signal, it re-establishes the control and data connections to the sender, optionally resets SCP, and continuous the session from where interrupted.

In our experiment, SCP is able to adapt to network change quickly and utilize the available bandwidth right away, with or without being reset. The reason is that the same congestion window size, around 5 packets, is optimal for both the PPP and the Ethernet link. If the network switch is between a PPP link and

[§]For TCP sessions, the end-end buffering delay measurement includes the time for back-off and retransmission of lost packets.

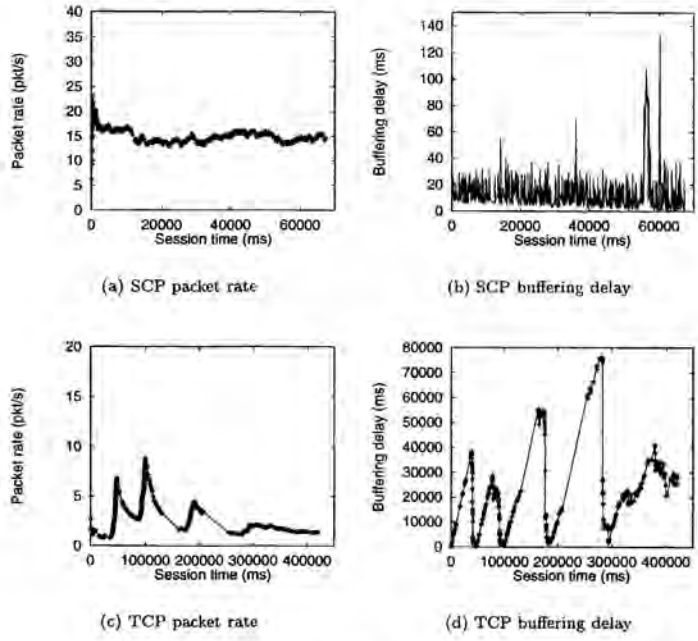
[¶]In a busy network when the buffers in routers are never empty, buffering delay measurements actually reflect the fluctuations in buffering delay.

[‖]In the presence of severe packet loss, most of the buffering delay measurement is the result of repeated back-off and retransmission of lost packets.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 012

Google Exhibit No. 1063
Page 1150



(a) SCP packet rate (b) SCP buffering delay

(c) TCP packet rate (d) TCP buffering delay

Figure 10. Performance of single SCP and TCP sessions across the busy Internet

a WAN connection with a large optimal congestion window size, we expect that reset on SCP would make a big difference. The slow-start policy following the reset helps figure out the new window size quickly.

6. RELATED WORK

Various schemes have been proposed for flow and congestion control in the context of streaming real-time multimedia data across the Internet. These schemes include HTTP/TCP, TCP without retransmission, rate-based flow and congestion control, and router-based congestion control. Most of these schemes address congestion control for unicast streaming. For multicast streaming there are also multilayered schemes.

HTTP/TCP based schemes: HTTP[7]/TCP[4] has been used in virtually all commercial streaming media players. The Internet has proven TCP's robustness and ability to share bandwidth among multiple sessions. Also important to commercial streaming players is HTTP's ability to penetrate firewalls. However, TCP is designed for reliable distribution of non-real-time data. When used for media streaming, it has the drawbacks of long and unpredictable latency, wasted network bandwidth, and bursty data flow. TCP implements reliability through infinite retransmission, causing unpredictable and unbounded latency and wasted bandwidth. TCP data flow is inherently bursty, due to its repeated process of increasing its congestion window size until packets are lost, then backing off exponentially. In network connections with deep buffers, such as PPP links, TCP's greediness in filling up the buffers incurs long latencies that are not tolerable in interactive streaming applications. SCP addresses the problems in TCP by not doing retransmission, taking increases in buffering latency as an early indication of network congestion, introducing a smooth steady-state policy, and doing packet pacing at the sender.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 013

Google Exhibit No. 1063
Page 1151

The problems of long latency caused by deep router buffers in (traditional) TCP has been addressed in TCP Vegas.[8] Vegas observes that in congested networks, router buffers will fill, causing increased latency, before they eventually overflow. A mechanism is used to measure the increase in latency and adjust the congestion window size accordingly. SCP is based on similar observations, but it uses a different flow control policy. The policy has been shown to ensure smooth streaming, low latency, and stable and fair sharing between multiple streams.

A congestion control protocol based on TCP, but without the retransmission is used in Jacobs' Internet video applications.[9] Removing reliability from TCP eliminates the resultant problems of unpredictability latency and wasted network bandwidth. However, other problems of TCP, including bursty data flow, and long latency in connections with deep buffers, remain the same. To reduce the impact of bursty data flow on video quality, Jacobs uses a buffer at the sender side, which introduces more latency. Instead, SCP tries to smooth the streaming itself. It directly paces out the sending of packets, and it employs a flow control policy with a steady state in which smooth streaming are maintained.

Rate-based schemes: Rate-based feedback is widely used for flow and congestion control in streaming applications. Examples are Smith's cyclic-UDP,[10] our first distributed MPEG player,[2] and the RTP[11] rate-based flow control proposed by Busse and Schulzrinne.[12] In a typical rate-based feedback scheme, the receiver continuously monitors the metrics of the incoming stream, such as the received data rate or data loss ratio, and sends feedback messages to the sender. The sender then adjusts the data rate accordingly. The goal is to maintain the stream quality metrics such as packet loss ratio at a controlled level. While rate-based feedback works well most of the time, it has the inherent danger of overflowing the network buffers, since the metric being directly controlled is the data rate, instead of the amount of buffering inside the network as in TCP. In the case of severe congestion, receiver-initiated negative feedback (the sender only reduces data rate when asked by the receiver) may not be able to react quickly enough, both because data rate or packet drop rate estimations are state-based and involve sluggish lowpass filtering, and feedback messages (to reduce the data rate) may not be able to get through to the sender in time.

Several sender-initiated rate-based schemes have also been proposed. In Keshav's control-theoretic packet-pair scheme,[13] packet-pairs are used by the sender to measure the available bandwidth and a control-theory-based policy is used for flow control. This scheme is based on the assumption that the routers/switches use round-robin-like queuing (fair queuing), but most, if not all, Internet routers/switches uses a totally different first-come-first-serve queuing.

A TCP-friendly unicast rate-based flow control scheme has been proposed by Mahdavi and Floyd.[14] By having the sender (instead of the receiver) monitor the overall packet loss ratio based on the acknowledgments from the receiver, this scheme solves the problem of feedback messages not getting through in time. However, the other problems such as sluggishness in estimation of packet loss rate, and the danger of network buffer overflow still exist. The scheme uses packet loss thresholds to detect congestion, and adopts a congestion avoidance policy similar to that of TCP. We believe that whether satisfactory thresholds exist, or if they can feasibly be computed, and how the scheme actually interacts with TCP, is yet to be seen. In contrast, SCP uses the same congestion avoidance policy as TCP. Only when there is no congestion, SCP's hybrid rate- and window-base policy will bring a stream to a steady state with a stable packet rate.

Router-base schemes: SCP as well as all the above-discussed schemes are end-host based. They are implemented exclusively on the sender or receiver sides. There is another category of schemes which are implemented on routers, such as the schemes proposed by Jeffay,[15] and by Floyd.[16] While router-based schemes may be more effective than host-based ones, they need to be implemented in routers. This requirement greatly slows down the speed at which the scheme can be deployed. Consequently, very few router-based flow or congestion control mechanisms can be found in the current Internet infrastructure.

Congestion control for multicast streaming: Our work focuses on unicast streaming, but some other researchers are examining rate adaptation policies for best-effort, real-time *multicast* streams. McCanne's receiver-driven layered multicast (RLM) combines layered content encoding with a layered transmission

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 014

Google Exhibit No. 1063
Page 1152

system.[17] Layered content is multicasted using multiple multicast groups, and receivers are responsible for adapting to network bandwidth variations by dynamically joining and leaving multicast groups.

7. CONCLUSIONS AND FUTURE WORK

We have presented SCP, an effective flow and congestion control scheme for Internet media streaming applications. SCP uses the same congestion avoidance mechanism as TCP. However, when the network is not congested, SCP's hybrid rate- and window-based flow control policy ensures stable, smooth streaming, and low latency. To adapt quickly to drastic changes in network bandwidth following a pause in the stream, or a switch of network interfaces, the state of SCP can be reset to begin a slow-start policy that uses a reduced initial congestion window size. Finally, SCP does not retransmit lost packets and hence eliminates that source of unpredictability in latency.

The experiments described in this paper test SCP over typical network configurations, including PPP, single subnet, single router (LAN), and across the long-haul Internet (WAN). The experiments demonstrate that SCP is able to stream data packets smoothly in all the cases. The buffering delay of SCP streams, even over PPP links with deep buffers, remains suitably low. Multiple SCP streams are able to share the network bandwidth in a stable, fair manner. Although our experiments show that residual errors in base RTT measurement can have a significant impact on network bandwidth partitioning among SCP sessions. Ensuring fair bandwidth partitioning without direct cooperation from network routers and switches, while not impossible, is not easy.

Our experiments also showed that SCP streams are able to share network bandwidth with TCP streams. SCP seems somewhat less aggressive than TCP, especially in the case of the PPP experiment. This behavior is a result of SCP's steady-state policy that limits packet buildup in router buffers, unlike TCP which tends to fill those buffers. This observation raises the question of whether TCP is too aggressive or SCP is too conservative. Filling up deep buffers in PPP servers merely results in more latency, but does not help to increase throughput. In interactive streaming applications, multiple seconds of latency is intolerable, and hence TCP's behavior over PPP links is clearly detrimental.

The properties described above make SCP a promising protocol for use with best-effort multimedia streaming across the Internet. We have incorporated SCP into our adaptive streaming video player.[6] Based on the available network bandwidth discovered by SCP, the player adapts the video presentation quality in multiple dimensions, including spatial resolution, frame rate, end-end latency, etc.

In the future, we plan to fine tune SCP's parameters and policies, study its execution overhead, optimize the number of ACK packets it generates, and evaluate SCP in more complex network scenarios. We also plan to compare SCP with other congestion control schemes - especially those used in commercial streaming media players - and further study, through analysis and experiments, the interaction between multiple SCP sessions and other congestion control schemes.

ACKNOWLEDGEMENTS

Discussions with the Quasar group at OGI motivated this work. Jon Inouye, Dylan McNamee and Ashvin Goel provided thoughtful comments on early drafts of this paper. Special thanks to Jon Inouye for his insights into TCP and other network protocols, and for discussions on the ideas behind this work. We are grateful to Prof. Karsten Schwan (Georgia Tech) for providing the account used to run the Internet experiments. This project was supported in part by DARPA grants N00014-94-1-0845, N66001-97-C-8522 and N66001-97-C-8523, and by Tektronix, Inc. and Intel Corporation.

REFERENCES

1. L. A. Rowe, K. D. Patel, B. C. Smith, and K. Liu, "MPEG video in software: Representation, transmission and playback," in *Symposium on Electronic Imaging Science and Technology*, pp. 15–26, (San Jose, California), February 1994.
2. S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole, "A distributed real-time MPEG video audio player," *NOSSDAV'95, Lecture Notes in Computer Science* **1018**, pp. 151–162. Springer-Verlag, 1995.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 015

Google Exhibit No. 1063
Page 1153

3. S. McCanne and V. Jacobson, "*vic*: a Flexible Framework for Packet Video," in *Proceedings of the Third ACM Conference on Multimedia*, pp. 511–522, (San Francisco, California), November 1995.
4. V. Jacobson, "Congestion avoidance and control," in *SIGCOMM'88*, pp. 79–88, August 1988.
5. J. Inouye, S. Cen, C. Pu, and J. Walpole, "System support for mobile multimedia applications," in *NOSSDAV'97*, pp. 143–154, May 19-21 1997.
6. S. Cen, *A Software Feedback Toolkit and Its Applications in Adaptive Multimedia Systems*. PhD thesis, Department of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology, October 1997.
7. T. Berners-Lee, R. Fielding, and H. Nielsen, "Hypertext transfer protocol – HTTP/1.0." Internet RFC 1945, `http://ds.internic.net/ds/rfc-index.html`, May 1996, [September 11, 1997].
8. L. S. Brakmo *et al.*, "TCP Vegas: New techniques for congestion detection and avoidance," in *SIGCOMM'94*, pp. 24–35, August 1994.
9. S. Jacobs and A. Eleftheriadis, "Adaptive video applications for non-QoS networks," in *International Workshop on Quality of Service'97*, pp. 161–166, (Columbia University, New York), May 1997.
10. B. C. Smith, "Cyclic-UDP: a priority-driven best-effort protocol." Computer Science Department, Cornell University, `http://www2.cs.cornell.edu/Zeno/papers/cyclicudp.pdf`, May 1994, [October 20, 1997].
11. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications." Internet RFC 1889, `http://ds.internic.net/ds/rfc-index.html`, January 1996, [September 11, 1997].
12. I. Busse, B. Deffner, and H. Schulzrinne, "Dynamic QoS control of multimedia applications based on RTP," *Computer Communications*, **19**, pp. 49–58, January 1996.
13. S. Keshav, "A control-theoretic approach to flow control," in *Proceedings of SIGCOMM'91*, pp. 3–16, (Zurich, Switzerland), Sept. 1991.
14. J. Mahdavi and S. Floyd, "TCP-friendly unicast rate-based flow control." In end2end mailing list, `ftp://ftp.isi.edu/end2end`, January 1997, [September 11, 1997].
15. K. Jeffay, M. Parris, F. D. Smith, and T. M. Talley, "A router-based congestion control scheme for real-time continuous media," in *Proceedings of NOSSDAV'96*, (Zushi, Japan), April 1996.
16. S. Floyd and K. Fall, "Router mechanisms to support end-to-end congestion control." Network Research Group, Lawrence Berkeley National Laboratory. `ftp://ftp.ee.lbl.gov/papers/collapse.ps`, February 1997, [September 11, 1997].
17. S. McCanne and V. Jacobson, "Receiver-driven layered multicast," in *SIGCOMM'96*, (Stanford, California), August 1996.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 01/17/2014 Terms of Use: http://spiedl.org/terms

Google Exhibit No. 1032
Page 016

Google Exhibit No. 1063
Page 1154

Ex. 1032A

Flow and Congestion Control for Internet Media Streaming Applications

Shanwei Cen[a], Jonathan Walpole[b] and Calton Pu[b]

[a]Tektronix, Inc., Video and Networking Division
P.O.Box 500, M/S 50-490 Beaverton, Oregon 97077 USA

[b]Department of Computer Science, Oregon Graduate Institute
P.O.Box 91000, Portland, Oregon 97291 USA

ABSTRACT

The emergence of *streaming* multimedia players provides users with low latency audio and video content over the Internet. Providing high-quality, best-effort, real-time multimedia content requires adaptive delivery schemes that fairly share the available network bandwidth with reliable data protocols such as TCP. This paper proposes a new flow and congestion control scheme, SCP (Streaming Control Protocol), for real-time streaming of continuous multimedia data across the Internet. The design of SCP arose from several years of experience in building and using adaptive real-time streaming video players. SCP addresses two issues associated with real-time streaming. First, it uses a congestion control policy that allows it to share network bandwidth fairly with both TCP and other SCP streams. Second, it improves smoothness in streaming and ensures low, predictable latency. This distinguishes it from TCP's jittery congestion avoidance policy that is based on linear growth and one-half reduction of its congestion window. In this paper, we present a description of SCP, and an evaluation of it using Internet-based experiments.

Keywords: Internet, flow control, congestion control, adaptive systems, streaming multimedia systems

1. INTRODUCTION

The real-time distribution of continuous audio and video data via streaming multimedia applications accounts for a significant, and expanding, portion of the Internet traffic. Many research prototype media players have been produced, including Rowe's MPEG player,[1] the authors' distributed video player,[2] and McCanne's vic,[3] etc. Over the past two years, many commercial streaming media players have also been released, such as RealAudio and RealVideo players, Microsoft Netshow, and Vxtreme.

Common characteristics of media streaming applications include their need for high bandwidth, smooth data flow, and low and predictable end-to-end latency and latency variance. In contrast, the Internet is a best-effort network that offers no quality of service guarantees, and is characterized by a great diversity in network bandwidth and host processing speed, wide-spread resource sharing, and a highly dynamic workload. Hence, in practice, Internet-based applications experience large variations in available bandwidth, latency, and latency variance.

To address these problems the new generation of Internet-based multimedia applications use techniques such as buffering and feedback-based adaptation of presentation quality. Buffering is used at the sender, receiver, or both, to mask short-term variations in the available bandwidth or latency. Through adaptation, applications scale media quality in one or more quality dimensions to better utilize the currently available bandwidth. For example, real-time play-out can be preserved in the presence of degraded bandwidth by adaptively sacrificing other presentation quality dimensions such as video frame rate or spatial resolution.

Dynamic adaptation is a powerful approach, but it requires new flow and congestion control mechanisms for accurate discovery and appropriate utilization of the available network bandwidth. Congestion control mechanisms must determine, dynamically, the share of the network bandwidth that can be fairly used by the adaptive application in the presence of competing traffic. If the mechanisms are not sensitive enough

Please send correspondence to Shanwei Cen at shanwei.cen@tek.com

SPIE Vol. 3310 • 0277-786X/97/$10.00

Google Exhibit No. 1063
Page 1156

to competing traffic the potentially high multimedia data rates could cause serious network congestion. On the other hand, if they are too sensitive they will under-utilize the bandwidth and presentation quality will be forced to degrade unnecessarily. In practice, such congestion control mechanisms must share bandwidth fairly among competing multimedia streams and must be compatible with TCP congestion control,[4] since TCP is the base protocol for currently-dominant HTTP and FTP traffic.

Flow control mechanisms should attempt to minimize end-to-end latency due to buffering, and maximize the smoothness of the data stream. Reliability through indefinite data retransmission is usually not desirable, since streaming applications can often tolerate some degree of quality degradation due to data loss, but are usually less tolerant of the delay introduced by the retransmission of lost data.

This paper presents a flow and congestion control scheme, called SCP (Streaming Control Protocol), for unicast media streaming applications. Like TCP, SCP employs sender-initiated congestion-detection through positive acknowledgment, and uses a similar window-based policy for congestion avoidance. The similarities in congestion avoidance between SCP and TCP make SCP robust and a good network citizen, and enable SCP and TCP to share the Internet fairly.

Unlike TCP, when the network is not congested, SCP invokes a hybrid rate- and window-based flow control policy that maintains smooth streaming with maximum throughput and low latency. Conversely, TCP repeatedly increases its congestion window size until packets are lost, then halves its window size (a behavior of TCP-Reno). One consequence of this behavior is that TCP sessions exhibit burstiness and develop long end-to-end latency due to the build up of packets in network router buffers. This behavior is particularly problematic over PPP links since some PPP servers have buffers that hold 15 seconds of data, or more. Also unlike TCP, SCP does not retransmit data lost in the network. Thus it avoids the associated unpredictability in latency and wasted bandwidth.

Further salient features of SCP include its support for rapid adaptation in the presence of drastic bandwidth fluctuations, such as those that occur when mobile computers migrate among different network types,[5] and its support for streaming-specific operations such as pausing.

This paper is organized as follows. Section 2 presents the design of SCP. Section 3 follows with an analysis of SCP's steady-state bandwidth sharing. Section 4 describes the implementation of SCP. Section 5 explains the experimental results. Section 6 then briefly discusses other congestion control schemes, and finally, Section 7 concludes the paper and discusses future work.

2. THE DESIGN OF SCP

A unicast streaming scenario with SCP involves a sender, which streams media packets over a network connection in real-time to a receiver. SCP policies are implemented at the sender side. Each packet contains a sequence number, and for each packet, SCP records the time at which it is sent, and initiates a separate timer*. The receiver acknowledges each packet it receives by returning an ACK containing the sequence number of the packet to the sender. Based on the reception of ACKs and expiration of timers, SCP monitors the available bandwidth, detects packet loss, and adjusts the size of its congestion window to control the flow and avoid network congestion. To achieve this task SCP maintains the following internal state variables and parameter estimators.

- *state* — The current state. SCP has several states, each corresponding to a specific network and session condition and associated flow and congestion control policy.
- W_l — The size of the congestion window (in number of packets).
- W — The number of outstanding packets sent but not acknowledged. When $W < W_l$, the congestion window is open, and more packets can be sent, otherwise it is closed and no packets can be sent.
- W_{ss} — The threshold of W_l for switching from the slow-start policy to the steady-state policy.
- $\hat{T}_{brtt}$ — An estimator of the base RTT (round trip time) – the RTT of a packet sent when the network is otherwise quiet.
- $\hat{T}_{rtt}$ — An estimator of the recent average RTT.
- $\hat{D}_{rtt}$ — An estimator of the standard deviation of the recent RTT.

*For clarity, we explain the design of SCP based on a timer per packet, but this aspect is optimized in the implementation.

Google Exhibit No. 1063
Page 1157

State	Network and session condition	Congestion window adjustment policy
slowStart	Available bandwidth not discovered yet	SCP opens the congestion window exponentially by increasing the window size by one upon the receipt of each ACK.
steady	Available bandwidth being fully utilized	SCP maintains appropriate amount of buffering inside the network to gain sufficient throughput, avoid excessive buffering or buffer overflow, and trace the changes in available bandwidth.
congested	The network is congested.	SCP backs off multiplicatively by halving the window size. Persistent congestion results in exponential back-off.
paused	No outstanding packet in the network	When a new packet is sent, SCP shrinks the window size and invokes the slow-start policy.

Table 1. SCP states, network and session conditions, and flow and congestion control policies

- $\widehat{rto}$ — An estimator of the timer duration.
- $\hat{r}$ — An estimator of the receiving packet rate.

As long as the congestion window is open, the sender streams packets at a rate no more than $\frac{W_l}{\hat{T}_{brtt}}$, instead of sending them out in bursts, so as to improve the smoothness of the stream. When an ACK is received, or a timer expires, the congestion window size W_l is adjusted using a policy associated with the current state.

SCP adopts window-based policies similar to those of TCP for slow-start, and exponential back-off upon network congestion. It also estimates RTT in a way similar to TCP. Therefore, many parameters and state variables have counterparts in TCP. However, SCP differs from TCP in that it has a base RTT estimator $\hat{T}_{brtt}$ and a packet rate estimator $\hat{r}$. The use of these estimators to ensure smooth streaming and low latency is discussed in the following sections.

2.1. Overall architecture

SCP is based on the observation that excessive packets in the round-trip network connection accumulate in buffers at network routers and switches and lead to an increased RTT. As the accumulation of packets increases, these buffers overflow and packets are dropped. The goal of SCP is to ensure smooth streaming at a suitably high throughput, but without causing excessive buffering or congestion in the network. To achieve this goal, SPC monitors fluctuations in the available bandwidth and pushes an appropriate number of additional packets into the network connection. If network congestion is detected, SCP reacts immediately with exponential back-off. Depending on the condition of the network and the streaming session under control, SCP is in one of the following four states: *slowStart*, *steady*, *congested* or *paused*. Each state is associated with a specific condition and congestion window size adjustment policy as listed in Table 1.

SCP handles events that indicate changes in network and session conditions. Such events include indications that: the SCP session becomes paused or active; the available network bandwidth is fully utilized; the network is congested; the network interface has been switched. Upon these events, SCP updates its internal state and possibly switches to a new policy. Figure 1 shows the events that SCP handles and its associated state transitions.

2.2. Congestion window adjustment policies and state transitions

Initialization: Upon initialization, SCP sets $W = 0$, $W_l = 1$ and $W_{ss} = L_w$, where L_w is a limit on the congestion window size. $\hat{T}_{brtt}$, $\hat{T}_{rtt}$ and $\hat{D}_{rtt}$ are set to infinity. $\widehat{rto}$ is set to an initial default value. After initialization, SCP enters the *paused* state. Transmission of the first packet brings SCP to the *slowStart* state.

Slow-start: The slow-start policy is invoked after initialization or when SCP resumes from a pause. Its goal is to quickly grow the congestion window and discover the available network bandwidth. W_l is incremented by 1 when an ACK is received in-order, thus doubling the congestion window after each RTT amount of time. SCP leaves the *slowStart* state upon detecting events indicating network congestion, full utilization of available bandwidth, pause in streaming, or network interface switch.

Google Exhibit No. 1063
Page 1158

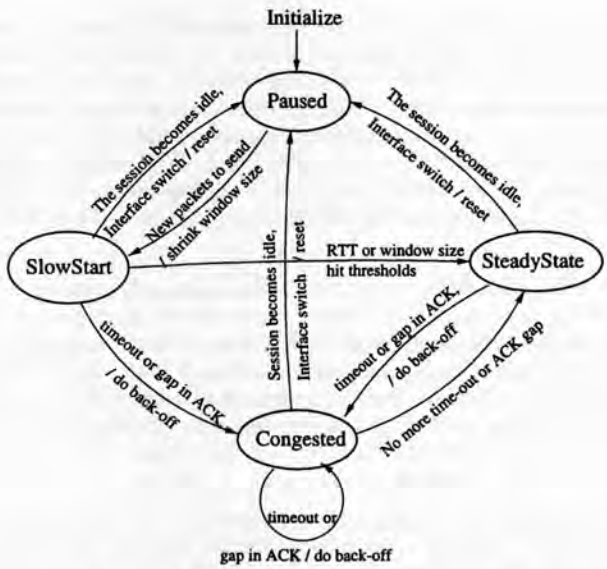


Figure 1. SCP state transition diagram

Steady-state smooth streaming: By pushing an appropriate number of extra packets, the steady-state policy enables sufficient utilization of the available bandwidth while avoiding over-buffering. It also traces the fluctuations in available bandwidth. In the *steady* state, estimation of packet rate $\hat{r}$ is enabled. Whenever a new $\hat{r}$ value is available, congestion window size W_l and threshold W_{ss} are adjusted according to Equation 1 below, where W_Δ is a constant referred to as the window-size incremental coefficient.

$$W_l = W_{ss} = \hat{r}\hat{T}_{brtt} + W_\Delta \tag{1}$$

The idea behind the flow control defined by Equation 1 is that SCP assumes that $\hat{r}$ is an approximation of the network bandwidth (in terms of packet rate) available to the session, calculates the bandwidth-delay product of the network connection with minimum buffering $\hat{r} \times \hat{T}_{brtt}$, and adjusts its congestion window size W_l accordingly. The product $\hat{r} \times \hat{T}_{brtt}$ is the amount of data SCP should keep outstanding inside the network in order to maintain sufficient throughput with minimum buffering. Since the network is shared and highly dynamic, the bandwidth available to the session under control fluctuates. If SCP just sets its congestion window size to $\hat{r} \times \hat{T}_{brtt}$, when the available bandwidth decreases, $\hat{r}$ would decrease, and SCP would reduce its congestion window size and thus the amount of outstanding data. However, if the available bandwidth increases, there is no way for SCP to detect that. This is because the receiving packet rate $\hat{r}$ could not be increased unless SCP increases the sending packet rate first, but SCP would not increase its sending packet rate by increasing the congestion window size unless it observes a increased $\hat{r}$. To solve this "chicken-and-egg" type of problem, SCP pushes W_Δ amount of extra outstanding data, which are held in router and switch buffers. When the network available bandwidth increases, releasing the extra data buffered in the network results in an increase in packet rate $\hat{r}$, which in turn results in larger amount of outstanding data as defined by the increased W_l. It can be seen that the increase in W_l is no more than additive, which is what TCP does in its steady state. Due to the smoothing effect of lowpass filtering in estimation of $\hat{r}$, W_l will be changed gradually, causing smooth changes in throughput. Also, with the flow control policy stated in Equation 1, though at the beginning $\hat{r}$ may not be a reliable estimation of the actual available bandwidth, if the network is stable enough, SCP will eventually bring $\hat{r}$ to the actual bandwidth through iterations. Overall, the steady-state policy is able to converge to a stable throughput, and to trace the fluctuation in available bandwidth smoothly.

Upon detecting network congestion, SCP backs off and enters the *congested* state. SCP enters the *paused* state when the session is paused, or when a switch in network interface is detected.

Google Exhibit No. 1063
Page 1159

The steady-state policy defined by Equation 1 is what makes SCP different from TCP. If network conditions remain the same for a sufficiently long period of time, SCP converges its window size and data rate to stable values. SCP is also able to limit the buildup of end-to-end latency resulted from buffering in routers, regardless of the actual sizes of router buffers. These properties are desirable for media streaming applications. In contrast, TCP repeatedly increases its congestion window size linearly until packets are lost, and then backs off by halving the window size. The result is that TCP traffic is inherently bursty. TCP also tends to fill up router buffers no matter how big they are. A consequence is that across network connections with deep buffers, TCP yields long latencies. The deeper the buffers are, the longer the latencies are.

Exponential back-off upon network congestion: Upon network congestion, SCP invokes an exponential back-off policy similar to that of TCP. Whenever events such as gaps in ACK or timer expirations are detected, SCP backs off multiplicatively by reducing its congestion window size in half. The data rate estimator $\hat{r}$ is no longer accurate and thus is reset and disabled. If the back-off is triggered by timeout, $\widehat{rto}$ may have been too short and is doubled. When the network is so congested that no packet can get through, with the multiplicative decrease in congestion window size and multiplicative increase in timer duration, SCP backs off exponentially until it virtually stops sending any further packet. This gives a chance for the network to quickly recover from the congestion.

At the time of a back-off, there may already be some outstanding packets inside the network. Loss of these packets does not correctly reflect the network condition after the back-off, and thus should be ignored. The *congested* state is designed for this purpose. After each back-off, SCP enters the *congested* state. Further back-off is disabled until the first packet sent in the current *congested* state is acknowledged, found lost or its timer expired. If the first packet is acknowledged, SCP enters the *steady* state, otherwise another round of back-off is initiated.

Pause when there is no packet to send: In a real-time streaming session, a user may want to pause in the middle. If the sender has no data to send for a while, W eventually decreases to 0. At this moment, the streaming session becomes idle, and SCP enters the *paused* state.

When an SCP session is paused, the bandwidth it previously took will gradually be discovered and taken by other sessions. When the session resumes at a later time, SCP invokes the slow-start policy with a reduced initial congestion window size. Currently, we adopt a ad-hoc policy of halving the congestion window size in every $\hat{T}_{brtt}$ time elapsed in the *paused* state.

Reset upon network interface switch: When either end of an SCP streaming session has its network interface switched (e.g., from Ethernet to cellular modem, or vice versa), the route from the sender to the receiver may be changed, and the new connection may go through links with totally different capacities. SCP handles this dynamic reconfiguration issue by providing a *reset* interface. Upon each network interface switch event, SCP discards the out-dated estimators, ignores all outstanding ACKs, and resumes with the slow-start policy to quickly discover the capacity of the new connection.

3. ANALYSIS OF BANDWIDTH SHARING BETWEEN SCP SESSIONS

With the steady-state rate- and window-based policy defined by Equation 1, it is possible for multiple SCP sessions to share network links in a fair and stable manner. In this section, we analyze a simple case, in which two sessions with the same packet size share a single network link. Both sessions send packets at maximum rate while the congestion window is open.

Suppose that, in a steady state, session A has estimators $\hat{r}_a$, $\hat{T}_{brtta}$, $\hat{T}_{rtta}$, and $W_{la} = \hat{r}_a\hat{T}_{brtta} + W_\Delta$; and session B has $\hat{r}_b$, $\hat{T}_{brttb}$, $\hat{T}_{rttb}$, and $W_{lb} = \hat{r}_b\hat{T}_{brttb} + W_\Delta$. Since A and B share the same network link, we have following observations:

(1) The aggregate packet rate of the network link, $\hat{r}_l$, is the sum of the two sessions: $\hat{r}_l = \hat{r}_a + \hat{r}_b$.

Google Exhibit No. 1063
Page 1160

(2) Base RTT estimate is the actual link base RTT T_{brttl} plus some estimation error: $\hat{T}_{brtta} = T_{brttl} + e_a$ and $\hat{T}_{brttb} = T_{brttl} + e_b$. When $\hat{T}_{brtt}$ is estimated as the minimum of the past RTT measurements of a session, the main component of the estimation error is the residual buffering, which is caused by packets of other sessions preventing the network buffers from becoming empty. Another less important component is sampling noise.

(3) Sessions A and B have the same RTT estimator (when the sampling noise can be ignored): $\hat{T}_{rtta} = \hat{T}_{rttb} = \hat{T}_{rttl}$. Furthermore, the number of packets sent by a session in one RTT equals its congestion window size. Thus the packet rate ratio of A and B equals the ratio of their window size:

$$\frac{\hat{r}_a}{\hat{r}_b} = \frac{W_{la}}{W_{lb}} = \frac{\hat{r}_a(T_{brttl} + e_a) + W_\Delta}{\hat{r}_b(T_{brttl} + e_b) + W_\Delta} \quad \Longrightarrow \quad \hat{r}_a = \frac{\hat{r}_b W_\Delta}{\hat{r}_b(e_b - e_a) + W_\Delta} \tag{2}$$

Combining observations (1) and (3), it is clear that there is a single solution for $\hat{r}_a$ and $\hat{r}_b$. The session with a larger residual buffering tends to have a larger estimation error, thus gets a larger portion of the link bandwidth. In a special case where the two sessions have the same RTT estimation error, $e_a = e_b$, we have $\hat{r}_a = \hat{r}_b = \frac{\hat{r}_l}{2}$, indicating that the two sessions split the link bandwidth evenly.

4. IMPLEMENTATION OF SCP

SCP has been implemented as a layer on top of UDP, based on a toolkit for developing adaptive systems.[6] In this section, we briefly discuss the parameter estimators and various implementation issues. More details may be found in Cen's thesis.[6]

Several parameters are estimated by SCP. Base RTT $\hat{T}_{brtt}$, average RTT $\hat{T}_{rtt}$, RTT standard deviation $\hat{D}_{rtt}$, and timer duration $\widehat{rto}$ are estimated based on the history of the raw RTT measurements. $\hat{T}_{brtt}$ is estimated as the minimum of all the past RTT measurements. The estimations $\hat{T}_{rtt}$, $\hat{D}_{rtt}$ and $\widehat{rto}$ are done in a way similar to those in TCP.[4] The receiving packet rate $\hat{r}$ is estimated as the rate at which ACKs are received. This is reasonable since the receiver acknowledges every packet it receives.

Several performance issues have been addressed in the implementation. The first is that, in the design of SCP discussed in Section 2, each outstanding packet has a separate timer. The overhead of maintaining these timers would make SCP non-scalable. In the implementation, only an ordered table is maintained to record the sending time of all outstanding packets. Upon receipt of an ACK, SCP removes the entry of the acknowledged packet from the table. Only when there is a new packet to send and the congestion window is closed, SCP checks the older packets in the table to see if their timers should have been expired, and performs back-off if necessary. Other issues, such as out-of-order packet delivering, congestion-window adjustment in low-data-rate sessions, etc., are also addressed.

5. EXPERIMENTS AND RESULTS

To evaluate the performance of SCP, a test program has been built to stream dummy packets between Internet hosts using SCP or TCP. The test program reads parameters and experimental scripts from files, and collects statistics for analysis.

The network configuration for the experiments is shown in Figure 2. We have two LANs, one at Oregon Graduate Institute (OGI) and the other at Georgia Institute of Technology (Georgia Tech) connected through the long-haul Internet with 29 hops. Among the hosts, *anquetil* is a Linux PC notebook on subnet 1 of the OGI LAN. This notebook can optionally be moved to subnet 2 or a $28.8kb/s$ PPP link. This configuration covers typical types of Internet connections such as phone lines, LAN, and WAN. Network interface switching in mobile environments is simulated by having *anquetil* to switch between its Ethernet and PPP interfaces dynamically.

Experiments have been performed to evaluate the performance of SCP in sessions over different network connections. For all the experiments, unless stated explicitly, the following parameter values are used: steady-state congestion window size incremental coefficient $W_\Delta = 3$, and data packet size $1472B$[†]. The size

[†]IP packet size $= 20 + 8 + 1472 = 1500B$ is the Ethernet MTU. $1472B$ is the UDP/TCP payload size, including the SCP header (in the case of SCP), the various fields carried in the packet, followed by a block of randomized data.

Google Exhibit No. 1063
Page 1161

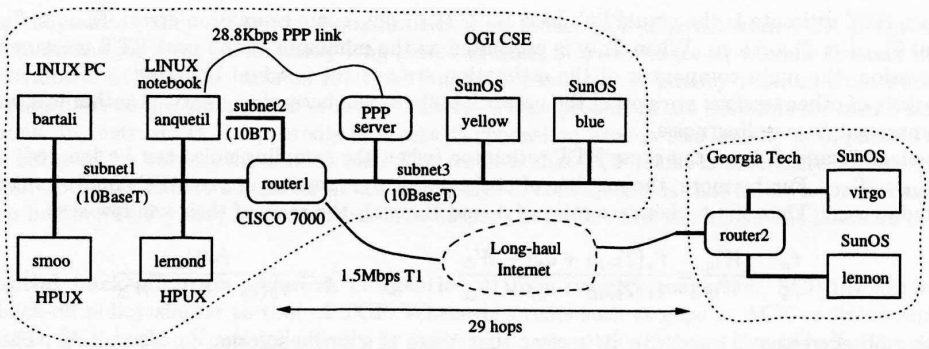
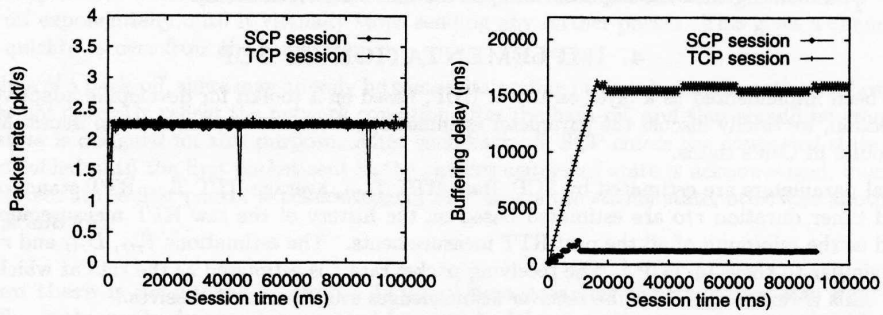


Figure 2. Network configuration for experiments



(a) Packet rate

(b) Buffering delay

Figure 3. Performance of single SCP and TCP sessions from *lemond* to *anquetil* over PPP

of TCP and SCP socket sending and receiving buffers is up to $64KB$, which is the maximum supported in many Berkeley socket implementations. The value of W_Δ is selected in an ad-hoc manner, but experiments have shown that it yield satisfactory performance.

In every SCP or TCP session, the sender uses non-blocking mode to send packets. It repeatedly waits until the socket is ready to send (in the case of TCP) or the congestion window is open (in the case of SCP), timestamps a dummy packet, and sends it out immediately. Each packet carries a sender timestamp. The receiver records the time each packet is received. With the receiving timestamps, intervals between packets and packet rate can be measured. Assuming that the clocks of the sender and the receiver have the same rate, with the sender and receiver timestamps, it is possible to measure the application-level end-to-end *buffering delay* — the delay from when a packet leaves the sender application until when it is received by the receiver application, minus the minimum of all the delay measurements, which is the transmission and processing delay. For each packet, the buffering delay is measured as the difference between the sender and receiver timestamps, minus the minimum of all the measurements in the history. This end-to-end buffering delay include the times spent in the buffers in the sender, the receiver, and routers and switches. For all TCP and SCP sessions, buffering delay and receiving *packet rate* are measured. For SCP sessions, the congestion window size, and the gaps in ACKs are also measured.

5.1. Experiments over PPP link

In the first set of experiments, packets are streamed from *lemond* to *anquetil* over the $28.8kb/s$ PPP link with a MTU of $1500B$.

Google Exhibit No. 1063
Page 1162

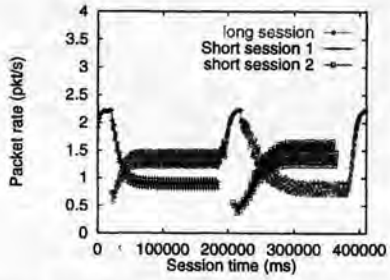


Figure 4. Smoothed packet rate of two SCP sessions from *bartali* to *anquetil* over PPP link

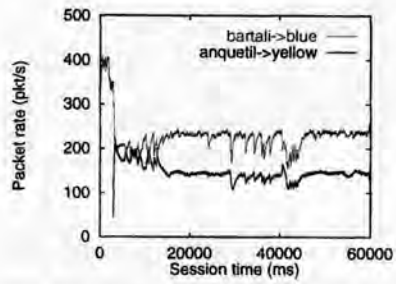


Figure 5. Smoothed packet rate of two SCP sessions through single router

First, single TCP and SCP sessions are played. For the SCP sessions, the receiving data rate is about $2.24pkt/s \approx 26.9kb/s$, which is close to the $28.8kb/s$ PPP link speed. Figure 3(a) shows the packet rate of a SCP session and a TCP session over time. It shows the raw packet rate (inverse of packet interval) versus the session time (the time relative to the beginning of the session). Figure 3(b) shows the buffering delay of these two sessions. After a startup phase, the SCP session yields a smooth packet rate, and utilizes the bandwidth sufficiently. The TCP session has similar data rate, except for the periodic downward spikes which are caused by packet loss (by the PPP server). However, SCP and TCP sessions maintain different levels of buffering inside the network. The SCP session has a steady-state buffering delay of about 1.2 seconds. In contrast, the TCP session pushes the buffering delay up to 16 seconds and stays there. As a matter of fact, different TCP implementations have very different behaviors over PPP. SunOS TCP has a steady-state behavior similar to that of HP-UX TCP, but a much worse start-up phase. Linux TCP pushes the delay to 7 seconds and stays there. Long network buffering delay for TCP over PPP is extremely detrimental to interactive streaming traffic. In an interactive streaming video presentation, each time the user changes playback speed, it is simply not tolerable to see no effect until after more than 16 seconds.

Careful readers may have noticed that 16 seconds is roughly the time for transferring $64KB$ (sender buffer size) of data over the $28.8kb/s$ PPP link. But this long delay is actually caused by a combination of PPP link's low bandwidth and the relatively large buffer in PPP server for individual PPP links. In our experiments, when UDP packets are sent from *lemond* or other hosts to *anquetil* over the PPP link, more than 15 seconds of data will be buffered in the PPP server before packets are dropped. A TCP connection over the PPP link will have its congestion window increased until either the sender or receiver socket buffer is full, or when packets are dropped when the buffer in PPP server overflows. To see that the long buffering delay is not caused by the big sender buffer itself, we also measured several TCP sessions in the reverse direction, from *anquetil* to *lemond*, and got buffering delays of about 2.5 seconds. It seems that LINUX has incorporate some techniques to reduce the latency for TCP over PPP to 7 seconds. Of cause, buffering delay can be reduced by reducing the TCP sender socket buffer size. But small sender socket buffer size is not always desirable. It will limit the size of congestion window, and will not be able to explore the bandwidth of long and fat network connections. There is not a single socket buffer size which fits all types of connections.

Figure 4 shows the smoothed[‡] packet rate of three SCP sessions, when two of them are played simultaneously from *bartali* to *anquetil* over the PPP link. It can be seen that the two competing sessions eventually reach a stable share of the PPP bandwidth. This experiment also shows the impact of residual buffering error in the based RTT estimation on bandwidth sharing. The base RTT estimation of the first half of the long session has a residual buffering caused by the first short session, thus it gets a larger share of the bandwidth. This result supports the observation in the bandwidth sharing analysis (Section 3). Later, after the first short session ends and the second short session starts, the latter gets a residual buffering error caused by the long session, and also gets a bigger portion of the PPP bandwidth. Further experiments show that if the start times of two competing SCP sessions are close enough, they will split the bandwidth evenly.

[‡]Smoothing is achieved by applying a lowpass filter to the raw packet rate measurements.

Google Exhibit No. 1063
Page 1163

Experiment configuration	Single SCP *anquetil→yellow*	Single TCP *anquetil→yellow*	SCP: *anquetil→yellow* SCP: *bartali→blue*	SCP: *anquetil→yellow* TCP: *bartali→blue*
Packet rate (pkt/s)	320	380	150 : 230	SCP 100, TCP 300
Buffering delay (ms)	0∼30 around 5	5∼80 around 40	0∼40 around 14	SCP 0∼50, TCP 10∼80

Table 2. Results of single TCP and SCP sessions through a single router

Experiments have also been carried out to play one SCP session and one TCP session simultaneously over the PPP link. Since SCP limits the amount of network buffering, the bandwidth partition depends on how aggressive a specific TCP implementation is. For sessions from *bartali* to *anquetil* the packet rates are $0.6pkt/s$ for SCP and $1.7pkt$ for TCP. HP-UX and SunOS TCP implementations are more aggressive. When SCP and TCP sessions are played from *lemond* to *anquetil*, virtually all bandwidth is taken by the TCP session.

5.2. Experiments on single subnet

All the single SCP sessions played from *bartali* to *anquetil* on subnet 1 yield smooth and stable throughput of about $700pkt/s = 8.4Mb/s$. The buffering delays of these sessions, primarily caused by MAC-level collisions and back-offs, remain within $4ms$. For comparison, individual TCP sessions yield $670 \sim 680pkt/s$ and up to $40ms$ of buffering delay. This indicates that SCP is efficient, smooth, and has low latency.

When two SCP sessions, one SCP and one TCP, or two TCP sessions are played *bartali→smoo* and *anquetil→lemond* on subnet 1, the two SCP and TCP sessions are able to share the Ethernet, but they are all very jerky, with highly variable throughput ($100pkt/s$ to $700pkt/s$) and buffering delay (up to 0.8 second). SCP also drops packets. This jerky throughput and packet drop is the result of severe MAC-level collisions and back-off which can not be detected by the sender application. Not much can be done by SCP or TCP to eliminate this problem except for explicitly limiting the data rate at the sender side.

5.3. Experiments through single router

To evaluate the performance of SCP streaming through a single router, *anquetil* is moved to subnet 2. Table 2 shows the configurations and the overall performance of the experiments. Figure 5 shows the smoothed packet rates of two simultaneous SCP sessions. These experiments were run late in the evening when the network was lightly-loaded.

A single SCP session yields a lower throughput than a single TCP session, but has a significantly lower buffering delay. Possibly reasons for SCP to have a lower bandwidth are that SCP sends more ACKs (one per packet), and maintains lower level of buffering thus the router buffers may be empty for more time.

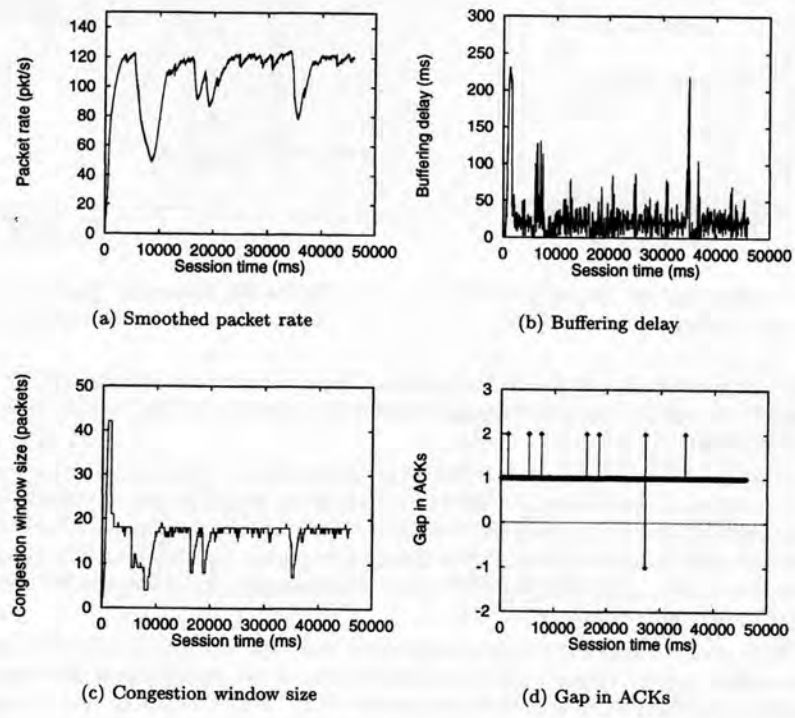
Figure 5 shows that when two SCP sessions are competing for the output port of the single router, they are able to share the bandwidth in a stable manner. To understand why there is a packet rate difference between the two sessions, a closer look at the statistical data reveals that the two sessions have the same steady-state congestion window size of 5, but they have a different RTT. When played separately, the RTT for sessions *bartali→blue* and *anquetil→yellow* are around $15ms$ and $22ms$ respectively. When played simultaneously, they are around $22ms$ and $35ms$ respectively. The difference in RTT may be caused by the differences in host CPU speeds and other factors, and results in an uneven bandwidth partitioning. This doesn't mean that SCP itself is unfair.

When an SCP session competes against a TCP session, they are able the share the bandwidth. But SCP's steady-state control of network buffering make it less aggressive than TCP and gets a significantly smaller amount (1/4) of the bandwidth.

5.4. Experiments across the Internet

To evaluate the performance of SCP across the long-haul Internet, SCP and TCP sessions are played between hosts at OGI and Georgia Tech. There is no way to control the traffic in the Internet. Nevertheless, there are still times, such as midnights and weekends, when the Internet is relatively lightly loaded, and times such as weekdays when it is heavily loaded.

Google Exhibit No. 1063
Page 1164



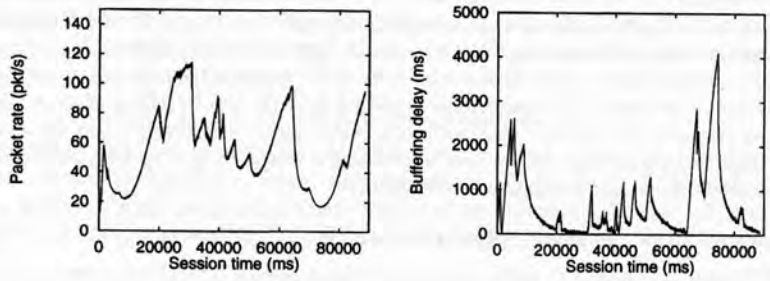
(a) Smoothed packet rate

(b) Buffering delay

(c) Congestion window size

(d) Gap in ACKs

Figure 6. Performance of a single SCP session across the lightly loaded Internet



(a) Smoothed packet rate

(b) Buffering delay

Figure 7. Performance of a TCP session across the lightly loaded Internet

As shown in Figure 6, when the network is lightly loaded, SCP is able to stream smoothly with low buffering delay, and to explore the network bandwidth sufficiently. From time to time, there are still periods of network congestion and packet loss. Figure 6(c) and (d) identify times when packets are lost, causing SCP to back off. The negative ACK gap shown in Figure 6(d) indicates an ACK delivered out of order. In this case, W_l is halved and then restored before any further packet is sent, thus the back-off is not shown in Figure 6(c). When packets are not dropped, SCP's steady-state rate- and window-based flow control policy maintains a stable congestion window size of about 18 packets, a stable throughput of about $115pkt/s (\approx 1.4Mb/s$, close

Google Exhibit No. 1063
Page 1165

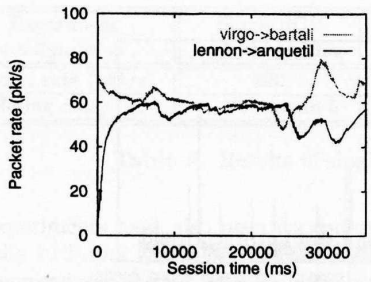


Figure 8. Smoothed packet rate of two SCP sessions across the Internet

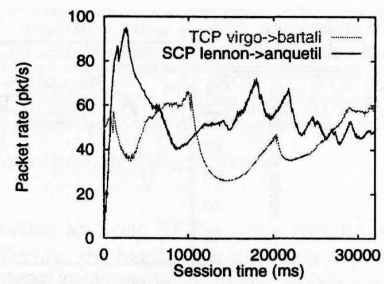


Figure 9. Smoothed packet rate of SCP and TCP sessions across the Internet

to the $1.5Mb/s$ T1 line speed), and low buffering delay (mostly lower than $50ms$). On the other hand, as shown in Figure 7, a single TCP session has more erratic throughput plus long and unpredictable buffering delays (up to 4 seconds)[§].

Two simultaneous SCP sessions are able to share the bandwidth in a fair manner. In our experiment, two competing SCP sessions: *lennon→anquetil* and *virgo→bartali* are played across the lightly loaded Internet. The ratio of the average bandwidth goes from $70pkt/s : 40pkt/s$ to $60pkt/s : 60pkt/s$. The buffering delay for the sessions are kept mostly below $100ms$. Figure 8 shows the packet rate of two simultaneous SCP sessions. Random packet drop causes competing sessions to back-off randomly, but on average the bandwidth sharing is fair, and the throughput is relatively smooth.

SCP and TCP sessions across the long-haul Internet share the network bandwidth more evenly than when through a single router. Figure 9 shows the packet rates of two simultaneous SCP and TCP sessions. The two competing sessions, SCP: *lennon→anquetil* and TCP: *virgo→bartali*, produce a throughput ratio around $SCP50pkt/s : TCP40pkt/s$. While TCP sessions have long latency, the buffering delay of the SCP sessions are still below $100ms$ for most packets.

During busy weekdays, it is more obvious that SCP yields smoother throughput and lower and more consistent delay than TCP, while still operating in harmony with the latter. The performance results of two SCP and TCP sessions from *virgo* and *bartali* are shown in Figure 10. The network is so congested that even "`ping virgo`" from *bartali* shows up to 10% packet loss. SCP sessions consistently get about $14pkt/s$ with buffering delays[¶] below $100ms$. SCP sessions observed up to 10% loss in ACKs. Due to the heavy packet loss, TCP sessions get only 2 to $6pkt/s$, with buffering delays up to 70 seconds[‖], and the throughput is very jerky. It is true that 10% packet loss will reduce the video presentation quality significantly, but a 70 second delay is simply intolerable to any streaming application!

5.5. Experiments with network interface switch

To evaluate SCP's ability to adapt to different network capacities resulted from network interface switches, both the PPP and Ethernet interfaces of the notebook *anquetil* are activated, and SCP sessions are played from *bartali* to *anquetil* through either the PPP or the Ethernet interface. Switching between PPP and Ethernet is simulated by reconfiguring the default IP route and sending a HUP signal to the receiver of the on-going SCP session. When the receiver gets a HUP signal, it re-establishes the control and data connections to the sender, optionally resets SCP, and continuous the session from where interrupted.

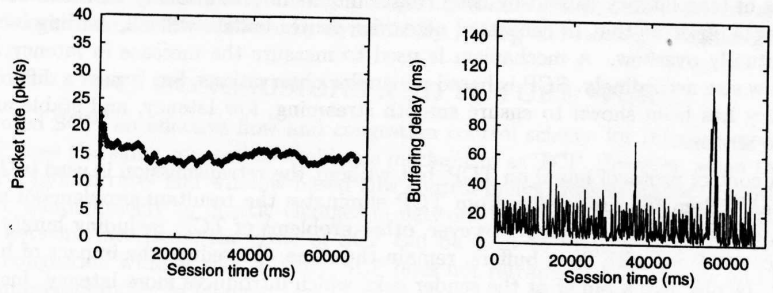
In our experiment, SCP is able to adapt to network change quickly and utilize the available bandwidth right away, with or without being reset. The reason is that the same congestion window size, around 5 packets, is optimal for both the PPP and the Ethernet link. If the network switch is between a PPP link and

[§]For TCP sessions, the end-end buffering delay measurement includes the time for back-off and retransmission of lost packets.

[¶]In a busy network when the buffers in routers are never empty, buffering delay measurements actually reflect the fluctuations in buffering delay.

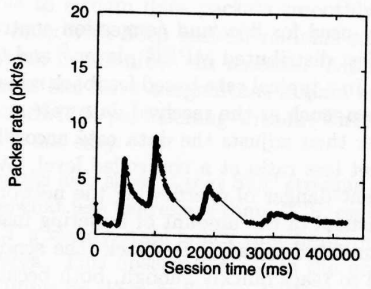
[‖]In the presence of severe packet loss, most of the buffering delay measurement is the result of repeated back-off and retransmission of lost packets.

Google Exhibit No. 1063
Page 1166

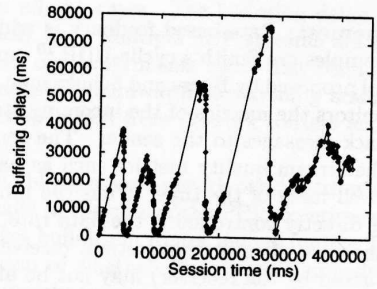


(a) SCP packet rate

(b) SCP buffering delay



(c) TCP packet rate



(d) TCP buffering delay

Figure 10. Performance of single SCP and TCP sessions across the busy Internet

a WAN connection with a large optimal congestion window size, we expect that reset on SCP would make a big difference. The slow-start policy following the reset helps figure out the new window size quickly.

6. RELATED WORK

Various schemes have been proposed for flow and congestion control in the context of streaming real-time multimedia data across the Internet. These schemes include HTTP/TCP, TCP without retransmission, rate-based flow and congestion control, and router-based congestion control. Most of these schemes address congestion control for unicast streaming. For multicast streaming there are also multilayered schemes.

HTTP/TCP based schemes: HTTP[7]/TCP[4] has been used in virtually all commercial streaming media players. The Internet has proven TCP's robustness and ability to share bandwidth among multiple sessions. Also important to commercial streaming players is HTTP's ability to penetrate firewalls. However, TCP is designed for reliable distribution of non-real-time data. When used for media streaming, it has the drawbacks of long and unpredictable latency, wasted network bandwidth, and bursty data flow. TCP implements reliability through infinite retransmission, causing unpredictable and unbounded latency and wasted bandwidth. TCP data flow is inherently bursty, due to its repeated process of increasing its congestion window size until packets are lost, then backing off exponentially. In network connections with deep buffers, such as PPP links, TCP's greediness in filling up the buffers incurs long latencies that are not tolerable in interactive streaming applications. SCP addresses the problems in TCP by not doing retransmission, taking increases in buffering latency as an early indication of network congestion, introducing a smooth steady-state policy, and doing packet pacing at the sender.

Google Exhibit No. 1063
Page 1167

The problems of long latency caused by deep router buffers in (traditional) TCP has been addressed in TCP Vegas.[8] Vegas observes that in congested networks, router buffers will fill, causing increased latency, before they eventually overflow. A mechanism is used to measure the increase in latency and adjust the congestion window size accordingly. SCP is based on similar observations, but it uses a different flow control policy. The policy has been shown to ensure smooth streaming, low latency, and stable and fair sharing between multiple streams.

A congestion control protocol based on TCP, but without the retransmission is used in Jacobs' Internet video applications.[9] Removing reliability from TCP eliminates the resultant problems of unpredictability latency and wasted network bandwidth. However, other problems of TCP, including bursty data flow, and long latency in connections with deep buffers, remain the same. To reduce the impact of bursty data flow on video quality, Jacobs uses a buffer at the sender side, which introduces more latency. Instead, SCP tries to smooth the streaming itself. It directly paces out the sending of packets, and it employs a flow control policy with a steady state in which smooth streaming are maintained.

Rate-based schemes: Rate-based feedback is widely used for flow and congestion control in streaming applications. Examples are Smith's cyclic-UDP,[10] our first distributed MPEG player,[2] and the RTP[11] rate-based flow control proposed by Busse and Schulzrinne.[12] In a typical rate-based feedback scheme, the receiver continuously monitors the metrics of the incoming stream, such as the received data rate or data loss ratio, and sends feedback messages to the sender. The sender then adjusts the data rate accordingly. The goal is to maintain the stream quality metrics such as packet loss ratio at a controlled level. While rate-based feedback works well most of the time, it has the inherent danger of overflowing the network buffers, since the metric being directly controlled is the data rate, instead of the amount of buffering inside the network as in TCP. In the case of severe congestion, receiver-initiated negative feedback (the sender only reduces data rate when asked by the receiver) may not be able to react quickly enough, both because data rate or packet drop rate estimations are state-based and involve sluggish lowpass filtering, and feedback messages (to reduce the data rate) may not be able to get through to the sender in time.

Several sender-initiated rate-based schemes have also been proposed. In Keshav's control-theoretic packet-pair scheme,[13] packet-pairs are used by the sender to measure the available bandwidth and a control-theory-based policy is used for flow control. This scheme is based on the assumption that the routers/switches use round-robin-like queuing (fair queuing), but most, if not all, Internet routers/switches uses a totally different first-come-first-serve queuing.

A TCP-friendly unicast rate-based flow control scheme has been proposed by Mahdavi and Floyd.[14] By having the sender (instead of the receiver) monitor the overall packet loss ratio based on the acknowledgments from the receiver, this scheme solves the problem of feedback messages not getting through in time. However, the other problems such as sluggishness in estimation of packet loss rate, and the danger of network buffer overflow still exist. The scheme uses packet loss thresholds to detect congestion, and adopts a congestion avoidance policy similar to that of TCP. We believe that whether satisfactory thresholds exist, or if they can feasibly be computed, and how the scheme actually interacts with TCP, is yet to be seen. In contrast, SCP uses the same congestion avoidance policy as TCP. Only when there is no congestion, SCP's hybrid rate- and window-base policy will bring a stream to a steady state with a stable packet rate.

Router-base schemes: SCP as well as all the above-discussed schemes are end-host based. They are implemented exclusively on the sender or receiver sides. There is another category of schemes which are implemented on routers, such as the schemes proposed by Jeffay,[15] and by Floyd.[16] While router-based schemes may be more effective than host-based ones, they need to be implemented in routers. This requirement greatly slows down the speed at which the scheme can be deployed. Consequently, very few router-based flow or congestion control mechanisms can be found in the current Internet infrastructure.

Congestion control for multicast streaming: Our work focuses on unicast streaming, but some other researchers are examining rate adaptation policies for best-effort, real-time *multicast* streams. McCanne's receiver-driven layered multicast (RLM) combines layered content encoding with a layered transmission

Google Exhibit No. 1063
Page 1168