TCP Streaming for Low-Delay Wireless Video*

Chi-Fai Wong Wai-Lam Fung Chi-Fai Jack Tang S.-H. Gary Chan
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong

Abstract

With the penetration and popularity of mobile devices such as pocket PCs and smart-phones, there is an increasing need of low-delay video streaming over wireless channel. Traditionally, UDP (User Datagram Protocol) is used for video streaming. However, due to unreliable transmission and fluctuating bandwidth of wireless channel, this requires error concealment and recovery mechanisms which greatly increases the complexity and delay of the system. Furthermore, UDP streams are often more difficult to penetrate firewalls. We hence propose using TCP (Transmission Control Protocol) for video streaming in this paper, due to its ease of use, reliability, and flexibility in selecting frames to transmit.

After discussing the wireless system architecture under consideration, we present a multi-worker model as implemented at wireless proxy (or encoder) which handles client requests independently. Our model makes use of a technique (selective packet drop) which selectively drops those unimportant frames so as to maintain video quality and low delay in the presence of congestion and fluctuating bandwidth. We have implemented a cellular and WLAN-based surveillance system using the model, and conduct real network measurement on its performance. Our model is simple and effective for mobile clients of heterogeneous bandwidth and computing power. Our results show that using TCP for streaming leads to good video quality in wireless networks.

1. Introduction

Driven by the popularity of mobile devices such as Smartphones and pocket PCs, there has been increasing interest for streaming applications over wireless medium. Many companies and software vendors (such as Oplayo and PacketVideo) have been set up to stream media to handhelds.

We show in Figure 1 an architecture for wireless video streaming. The video is first captured and encoded at a streaming server into multiple sub-streams using multiple description coding or layered coding [1, 2]. These sub-streams are then delivered, using either Internet or overlay multicast, to distributed proxies. Mobile clients of heterogeneous capability contact these proxies for services. A client of higher bandwidth and/or screen format may get more sub-streams incrementally from its proxy to maximize user's viewing quality.

The wireless network is scalable in the sense that the streaming server does not directly serve all the clients due to hierarchical architecture. By putting more proxies in the network, the system is able to incrementally serve more users. Though algorithms have been proposed and studied to adapt the encoding rate to client's decoding rate (see, for examples, [3–5]), they require receivers to periodically feedback its buffer state to sender. This increases the network bandwidth and the complexity of the server, and raises feedback-implosion issues. Our architecture does not require continuous feedback from the clients, and hence is simple and more scalable.

In this paper, we focus on the design of proxy in the network in terms of its buffer management to offer low-delay wireless video streaming. By "low delay," we mean that there is some target maximum frame delay which should not exceed. Our objective is to design and implement such a system, and conduct field trails on its performance. Note that because all sub-streams may be considered independently, without loss of generality, we focus on a single sub-stream and simply call it "stream" in our exposition in the paper.

We first argue that TCP is appropriate for video streaming from proxy to clients. An issue of using TCP for low-delay streaming is that TCP does not guarantee timely delivery due to retransmission. Because the bandwidth of wire-

*This work was supported, in part, by the Areas of Excellence (AoE) Scheme on Information Technology funded by the University Grant Council in Hong Kong (AoE/E-01/99), and by a grant of High Impact Area at HKUST (HIA03/04.EG04).

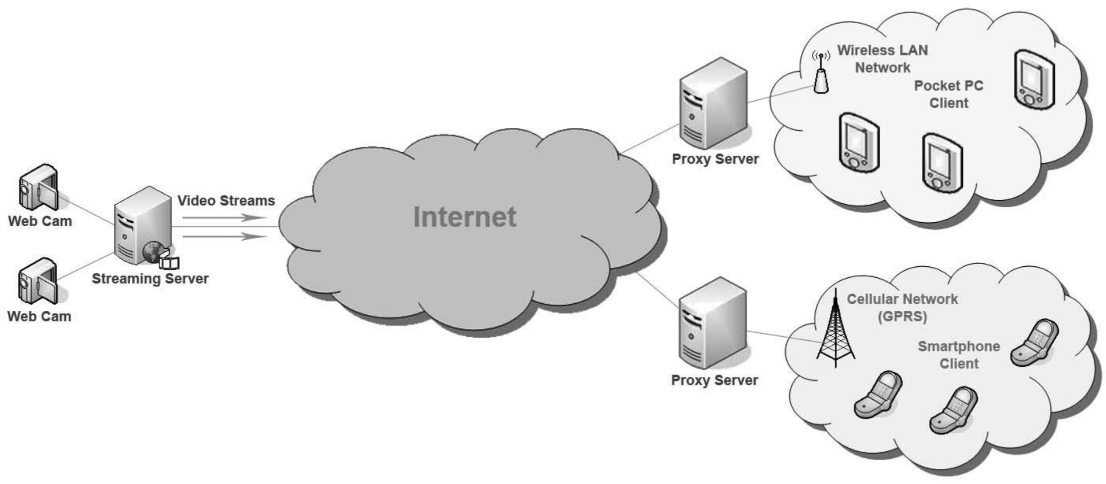

Figure 1. A hierarchical architecture for wireless video streaming.

less networks (the 802.11g LANs versus GPRS network) and client capability (high-end versus low-end phones) may vary widely, we propose and present a work model which uses flow-based buffer management at proxy. By treating each flow independently, we are able to isolate flows and tailor them for maximum quality, thereof achieving smooth video quality for the clients. The model is simple to implement and is based on our previous scheme of Selective Packet Drop (SPD) [6]. SPD meets a certain video delay requirement by allocating a finite-size buffer according to the delay tolerance of each client. The buffer keeps only those current, important and useful frames. However, our current work extends from [6] in the following ways: 1) we use TCP instead of UDP to address wireless error issue; 2) The SPD algorithm is implemented at the proxy rather than in the client. This is done so as to take advantage of the high-end proxies and to reduce the computational and memory requirements at the client. In this way, the computing resources at clients can be dedicated to decode and playback the incoming video packets, hence increasing the video quality.

We have implemented a surveillance system for real-time video streaming based on H.263+ [7]. In the system, a desktop PC captures and encodes video to H.263+ format and streams it through a wireless network (wireless LAN for Pocket PCs and GPRS network for smartphones) using TCP. Our performance study and field trials indicate that TCP streaming is effective to provide good-quality video over wireless channel. Using our model, the encoder does not need to encode its stream for each client, greatly reducing system complexity and increasing scalability of the number of clients.

This paper is organized as follows. We argue in Section 2 the reason of using TCP over UDP for streaming, and describe in in Section 3 the worker model. We implement our model and in Section 4, we present our experimental environment and measurement results. We conclude in Section 5.

2. UDP versus TCP Streaming

Wireless channel is characterized by fluctuating and low bandwidth with unpredictable error. Mobile devices, on the other hand, are characterized by their low processing/computational capability and low memory. Streaming low-delay high-quality video over wireless channel is hence challenging [8, 9]. Traditionally, User Datagram Protocol (UDP) is used for media streaming. However, UDP is not effective for wireless streaming, mainly due to the following reasons:

- *Complex error handling mechanism:* UDP is an unreliable protocol. As a result, packets may be lost during transit. To offer good-quality video, these losses have to be mitigated. Retransmission, FEC (Forward Error Correction), and error concealment are techniques which may be used (See [10–12] and references therein). However, efficient retransmission techniques are generally not easy to be implemented. They also increase the complexity at both proxy and client. FEC, and similarly for error resilience coding at the encoder, often increases the delay of the stream and tends to be designed for the worst-case scenario, leading to much bandwidth wastage. Error concealment, on the other hand, is effective for *random* error rather than burst error characterized by wireless channel. It also increases the complexity at the decoders.

- *Network unfriendliness:* UDP transmission is not elastic and hence not TCP-friendly. As a result, it either takes unfairly too much bandwidth or leads to high packet loss in the presence of fluctuating bandwidth. Though TCP-friendly UDP has been widely discussed their implementation is not straightforward [13–15].

- *Unselective data loss:* For video stream, some frames (e.g., those I frames) and some data fields (e.g., those synchronization bits) are more important than others and need to be protected. Since wireless error occurs at any time, these important data may be lost, leading to degradation in quality. If those more important frames or data fields can be selectively protected, better video quality would be achieved.

- *Firewall penetration:* Though some protocols make use of UDP (STUN, SIP, RTP, etc.), many more applications make use of TCP. Applications using UDP more likely experience firewall penetration problem than TCP.

In view of above, we propose and study the use of TCP for low-delay wireless video streaming. There are several advantages of using TCP:

- *Reliable transmission:* TCP is a reliable protocol, and hence effectively addresses the synchronization and retransmission problem as mentioned above. There is no need of complex error concealment and resilience mechanisms which need to be implemented in the client and proxy.

  Using TCP, the proxy can be designed to intelligently select and transmit those important frames/data in the presence of fluctuating bandwidth. There is hence more flexibility in choosing which frame to transmit and at what time. No extra framing overhead such as RTP and RTCP is required. One of the advantages of using UDP is its multicast capability. However, given that multicast capability is not pervasive in nowadays wireless networks, using TCP is a more natural and simpler choice than UDP.

- *Network fairness:* TCP is intrinsically friendly, which shares network resources with other data traffic/flows in the presence of congestion. There is no need to implement other mechanisms to achieve fairness. It also adapts its transmission rate according to the available network bandwidth, thereof allowing the video applications to make full use of the bandwidth.

- *Ease of deployment:* Using TCP in applications is easy, and TCP applications more readily penetrate firewalls (by means of, for example, http).

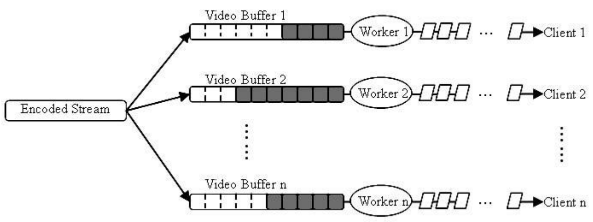

Figure 2. Multi-worker model as implemented in a proxy.

3. Multi-Worker Model

We show in Figure 2 the multi-worker model as implemented in a proxy. Each mobile client is associated with a proxy, which allocates a buffer corresponding to the client's delay requirement. A dedicated worker thread is created to serve each client. In other words, the buffer is managed independently using multiple threads. Encoded video frames coming into the proxy is replicated to the video buffer of each client. The frames in the buffer is emptied at the other end to be sent to the client using TCP. The buffer only keeps complete/full frames and may drop some frames in times of overflow (due to congestion).

Due to independent processing of buffers, the bandwidth and processing capability of a client would not affect the other clients in the network. Furthermore, the packet loss of a client would not affect the performance of other clients. In this way, video encoder does not need to adapt its stream on per-flow basis, thereof greatly reducing the complexity of the system.

As TCP is a reliable transport protocol, packets are retransmitted upon lost. Hence, all the frames emptied from the buffer would eventually arrive at the client. As frames are put into the buffer and be consumed at the other end with different rates, frames, and hence streaming delay, may accumulate at the buffer. When the buffer becomes full, those not-so-important frames need to be dropped to meet low-delay requirement.

When the buffer starts to overflow, we have used the technique of Selective Packet Drop (SPD) to maintain low delay and high video quality [6]. SPD is implemented at the work at the proxy so as to relieve the computation at the client. A client simply decodes the arrived frames and does not need to keep track of the delay problem.

To achieve high quality low-delay video, SPD makes use of the observation that the importance of video frames in a GoP (Group of Picture) of IPPP... sequence decreases from the first I to the last P. This is because each P frame in the GoP uses the previous frame as reference. When a P frame is lost due to buffer overflow, all the subsequent P frames

```
SPD_Worker_Thread($m$)
1  $Q \leftarrow$ Empty Queue of size $m$
2  while 1
3    do WaitPacket($P$)
4      if Full($Q$)
5        then SearchOldFrame($F$)
6          RemoveFrame($F$)
7      Enqueue($Q, P$)
```

Figure 3. The Selective Packet Drop (SPD) algorithm.

would not be useful and may be as well dropped.

We show in Figure 3 the SPD algorithm each dedicated worker thread runs for a buffer. In SPD, packets are allowed to accumulate in the buffer as long as there is space. SPD algorithm keeps the most recent I-frames and its following P-frames. In times of buffer overflow, a trailing P-frame or an isolated I-frame (i.e., an I-frame without any dependent P-frame) is dropped, whichever is the oldest. In SPD, frames are hence dropped to keep those most important and current ones in the buffer. This is done to keep the buffer in good utilization with useful frames. SPD always puts into the buffer the most recent I-frame and the P-frame whose reference frame has not been dropped. Clearly, the size of the buffer indicates the maximum delay of the stream.

4. Experimental Environment and Measurement Results

We have implemented the system as shown in Figure 1. In this section, we first present the experimental environment followed by measurement results.

4.1. Experimental Environment

The Foreman QCIF sequence is used as a representative video sequence in our experiment. The sequence consists of 400 frames. The frames are encoded in H.263+ format before delivered over the Internet.

The server-side video delivery program run on a Pentium IV 3GHz PC with 1GB memory. The server is connected to a 100Mbps LAN. The mobile access point offering wireless network connections is directly connected to the same LAN. The GPRS service was offered by China Resources Peoples Telephone Company Limited (China Resources Peoples Telephone Company Limited is a cellular network company in Hong Kong [16]). One of the client-side program runs on a HP iPAD h5450 Pocket PC. Besides the wireless LAN card, no other additional hardware was installed to the Pocket PC. Another client-side program ran on an i-mateTM SP3 SmartPhone with external storage card.

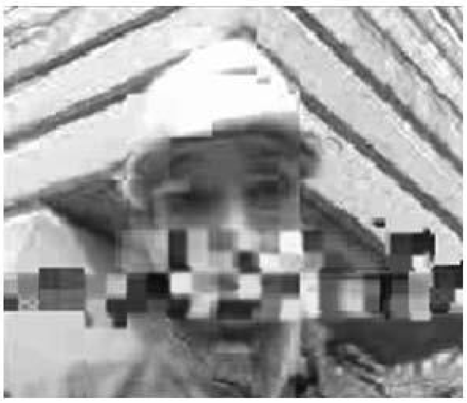

Figure 4. Subjective video quality of using UDP streaming in high error environment.

Regarding H.263+ encoder settings, we use a Quantization Parameter (QP) of 13, a search window size of 15, a GOP size of 4, and without error concealment. The encoded video stream is transmitted packet-by-packet to the clients. Each encoded frame is divided into fixed-size packets of 2,048 bytes. The buffer of each worker is a FIFO queue accommodating up to 10 frames.

We stream the video using TCP and UDP, and compare the video quality. in terms of subjective visual inspection and objective PSNR metric. We also examine the frames that are lost, and the delay incurred with or without the video buffer. We simulate the error environment by randomly dropping video frames at the exit of the proxy. We present the case of high error environment where we randomly drop 15% of frames. For TCP, this means that some frames are sent multiple times before the next one is transmitted. Besides the drop, the wireless networks are observed to have much lower loss rate during our measurement and hence may be ignored.

4.2. Measurement Results

In Figures 4 and 5 we show the subjective video quality for UDP and TCP streaming, respectively. Clearly, the one with TCP is better. For UDP in a high error environment, due to its unreliable and unselective data loss, the video quality is poor and may lead to dis-synchronization. This is not case for TCP.

Figure 6 shows PSNR for the decoded frames using UDP streaming. The gaps in the sequence mean dropped or missed frames. This due to loss of synchronization and unrecoverable errors. It is clear that the video quality decreases sharply upon a frame loss. The errors propagate to subsequent frames (due to inter-coded P frames to reduce temporal redundancy), leading to substantial reduction

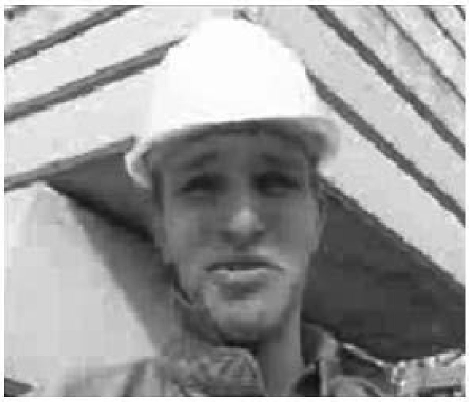

Figure 5. Subjective video quality of using TCP in high error environment.

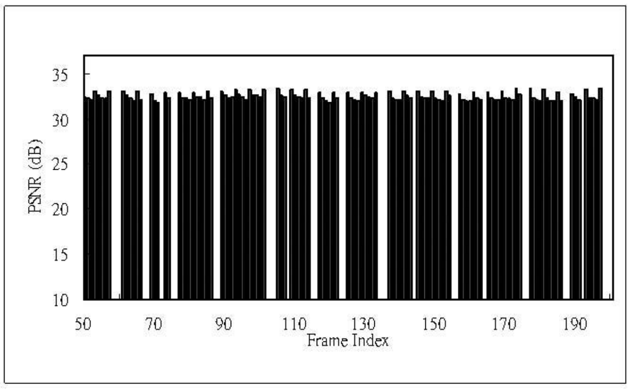

Figure 7. PSNR for the decoded frames using TCP streaming.

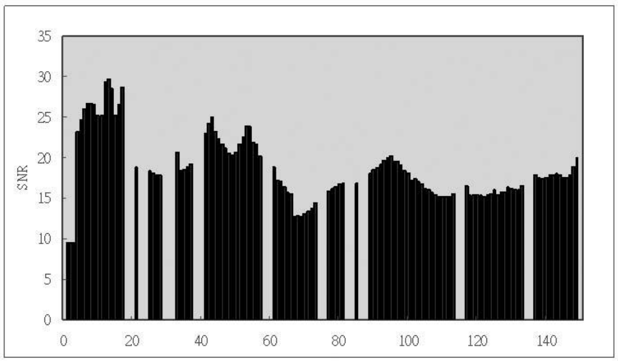

Figure 6. PSNR for decoded frames using UDP streaming.

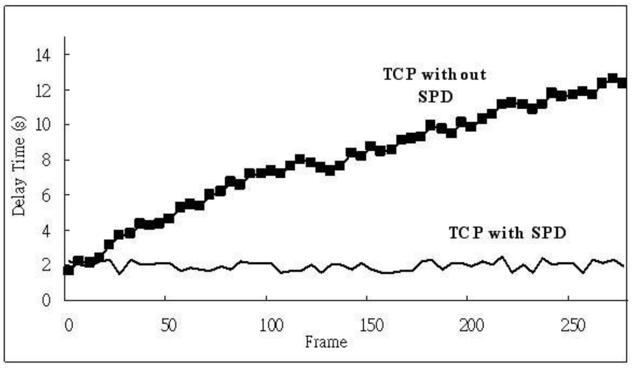

Figure 8. Delay time with respect to the proxy with and without SPD using TCP streaming.

in video quality and gaps. In this environment, much of the channel bandwidth is wasted to transmit poor-quality or useless frames. The resultant video quality is also not smooth.

We show in Figure 7 the PSNR for decoded frames using our TCP streaming. The PSNR is maintained at a high level, showing that our approach is robust to network loss. Since TCP retransmits all the lost frames, the important frames are recovered in high error environment. The gaps in the sequence are due to selective packet drop in times of overflow of video buffer in proxy. Since we drop frames intelligently and transmit those important frames, error propagation is eliminated and the channel bandwidth is used to delivered high-quality video. The video is clearly of much higher quality and smoother than the UDP case, as frames are occasionally and strategically dropped.

We finally compare the delay of each frame with and without SPD at sender using TCP in Figure 8, i.e., with finite and infinite buffer, respectively. Without SPD, there is a cumulative delay which increases quickly. This is because TCP does retransmission in high error environment. As a result of a reduction of throughput, the video incoming rate is higher than delivery rate, leading to frame and hence delay accumulation in the buffer. On the other hand, when SPD is used, the delay time is kept to a low value. This is because frames are dropped to accommodate more recent frames and the delay time is bounded by the video buffer size in proxy.

5. Conclusions

With the popularity of mobile devices such as smartphones and pocket PCs, there is an increasing need of low-delay video streaming over wireless networks. In this paper, we show that using TCP is effective for video streaming, due to its reliability, selective frame transmission and protection, and ease of implementation.

We present a hierarchical wireless architecture consisting of proxies for video streaming. The proxy makes use of TCP to serve its mobile clients by means of a multi-worker model and selective packet drop (SPD). To keep low delay, each client is allocated a finite-size buffer. SPD makes good use of the buffer by keeping the most useful and recent frames in the buffer.

We have implemented the model into a video surveillance system based on H.263+ and conduct experimental study on its performance. We measure both the subjective and objective video quality. Our experimental results show that using TCP and the model achieve high-quality, smooth, and low-delay video streaming over wireless channel.

References

[1] A. R. Reibman, H. Jafarkhani, Y. Wang, and M. T. O. R. Puri, "Multiple-description video coding using motion-compensated temporal prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 193–204, Mar. 2002.

[2] A. Sehgal A. Jagmohan and N. Ahuja "Wireless video conferencing using multiple description coding," in *The 2001 IEEE International Symposium on Circuits and Systems*, vol. 5, May 2001, pp. 303–306.

[3] L. A. Rowe and B. C. Smith, "A continuous media player," in *Proceedings of the Third International Workshop on Network and Operating System Support for Digital Audio and Video*, Aug. 1992, pp. 376–386.

[4] A. Goel M. Shor J. Walpole D. Steere and C. Pu "Using feedback control for a network and CPU resource management application," in *Proceedings of the 2001 American Control Conference (ACC)*, vol. 4, June 2001, pp. 2974 – 2980.

[5] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole, "A distributed real-time mpeg video audio player," in *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, Apr. 1995, pp. 151–162.

[6] K.-W. Cheuk, S.-H. Chan, K.-W. Mong, C.-M. Lee, and S.-S. Sy, "Developing PDA for low-bitrate low-delay video delivery," in *Proceedings of IEEE International Conference on Mobile and Wireless Communications Networks*, Oct. 2003.

[7] http://www.itu.int/ITU T/index.html.

[8] D. Wu, Y. Hou W. Zhu, Y.-Q. Zhang, and J. Peha "Streaming video over the internet: approaches and directions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 282–300, Mar. 2001.

[9] D. Wu, Y. Hou and Y.-Q. Zhang, "Scalable video coding and transport over broadband wireless networks," in *Proceedings of the IEEE*, vol. 89, Jan. 2001, pp. 6–20.

[10] T.-W. Lee, S.-H. Chan, Q. Zhang, W.-W. Zhu, , and Y.-Q. Zhang, "Allocation of layer bandwidth and FEC for video multicast over wired and wireless networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 1059–1070, Dec. 2002.

[11] A. Majumda, D. Sachs, I. Kozintsev, K. Ramchandran, and M. Yeung, "Multicast and unicast real-time video streaming over wireless LANs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 524 –534, June 2002.

[12] B. Girod and N. Farber, "Feedback-based error control for mobile video transmissions," in *Proceedings of the IEEE*, vol. 87, Oct. 1999, pp. 1707–1723.

[13] S. Jan and W. Liao, "Supporting non-adaptable multimedia flows by a TCP-friendly transport protocol," in *IEEE International Conference on Multimedia and Expo*, vol. 3, June 2004, pp. 2091 – 2094.

[14] Q. Wang, K. Long, S. Cheng, and R. Zhang, "TCP-friendly congestion control schemes in the Internet," in *2001 International Conferences on Info-tech and Info-net*, vol. 2, Oct. 2001, pp. 211 – 216.

[15] B. Mukherjee and T. Brecht "Time-lined TCP for the TCP-friendly delivery of streaming media," in *International Conference on Network Protocols*, Nov. 2000, pp. 165 – 176.

[16] http://www.peoples.com.hk/.