

Dynamic Handoff of Multimedia Streams

Roger Karrer and Thomas Gross
 Institute for Computer Systems
 ETH Zurich
 CH 8092 Zurich

Abstract— **Multimedia applications can perform a handoff to switch from an old, low performance connection to a new one. During such a handoff, the player should be fed with a constant data stream. This paper investigates two important factors that influence the application-perceived quality: the correct setup of the new connection, i.e. the first packet to be sent, and how long the old connection is allowed to send after the handoff initiation. Four methods for the first problem are developed and evaluated with an MPEG application.**

I. INTRODUCTION

The delivery of video over the Internet in good quality is challenging because the bandwidth offered by the network cannot always satisfy the demands of the applications. One way to address the lack of bandwidth is to use adaptive filters. An intelligent frame-dropping filter tries to optimize the video quality by selectively dropping the least important packets in case of congestion.

Another approach that is orthogonal to adaptivity is to perform a handoff. A handoff is the switch from one connection to an alternate connection. In contrast to adaptivity, which attempts the optimization of a single connection, a handoff takes alternate connections into account.

During a handoff, no frames should be lost, and the data stream to the video player should be kept as smooth as possible. These two goals can be accomplished by (i) starting the new connection at the correct position, (ii) correctly synchronize the two streams while they are simultaneously sending, and (iii) disconnecting the old connection at an appropriate time.

Section II describes the handoff challenges in more detail. In Section III, we investigate how different parameters influence the frame receiving. Based on this investigation, we create a model that allows an application to set up the new connection, and evaluate this model with an MPEG application. After a brief overview of related work, we conclude in Section VI.

II. DYNAMIC HANDOFF

A handoff denotes a switch from one connection to a new connection, where the new connection (hopefully) offers better connectivity. Handoffs are well-

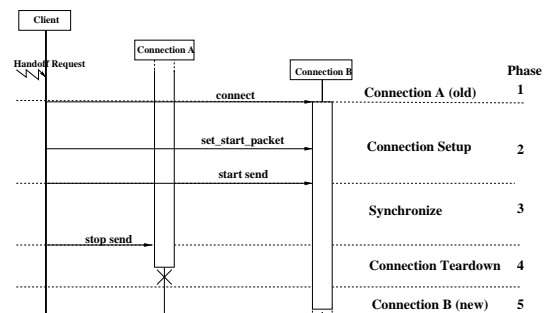


Fig. 1. Sequence diagram of the handoff, which can be divided into 5 phases: (i) connection A, (ii) connection setup, (iii) synchronize, (iv) connection tear down and (v) connection B only.

known techniques from mobile communication: when a user moves from one cell to another, the connection must be handed off from the old to a new base station. A handoff may as well be used for applications that transmit their data over networks, e.g., the Internet. If an application learns about a different connection that offers better bandwidth (e.g., a multicast client finds a better connection to another filter), it may switch to this new filter.

A handoff should be invisible to a client in two ways: the client must not know from which connection the data is received, and the data should be received at a steady rate even during the handoff.

Figure 1 shows a sequence diagram of a client-side handoff, which can be divided into 5 phases. In phase 1, the client receives data over the old connection A. During this transmission, a handoff request is triggered. Upon this trigger, a new connection B is established (phase 2) and the new connection starts streaming. During phase 3, packets are received via both connection, requiring a synchronization of the two streams before the player. In phase 4, the old connection is disconnected. Finally, in phase 5, data is only sent over the new connection B.

The most important issue during all phases is to deliver a constant data stream to the player. Two factors influence the smoothness: the correct setup of the new stream and the duration of the synchronization phase. First, the start of the new data stream must be synchronized with the old stream: if, e.g., a video has been played for a while, the new stream must start at the appropriate position. Second, the application can de-

lay the shutting down of the old connection. Thereby, packets continue to arrive over the old connection, hiding the connection setup. The disadvantage is that the same packet arrives over both connections and has to be discarded. Section III will present our solution to these problems.

The application discussed in this paper is a video streaming application. It consists of three components: server, filter and client. Clients retrieve movies from servers, and, in between, one or more filters can adapt or multicast the data.

We describe a client-based, application-layer handoff. Our solution, however, is not limited to this setup. We chose a client-based handoff because the application we use for the evaluation observes the incoming data stream for losses and triggers a handoff if the loss rate is too high or the connection is completely down. However, any other component may also trigger a handoff: the receiver of a filter may get aware of a lossy connection, and the server may request to reorganize its multicast tree.

In addition to the application-driven triggering, external tools, such as network monitors that scan a network for alternate paths may also trigger a handoff at any participant. Finding alternate paths, i.e. how they are detected and how their performance is estimated, is beyond the topic of this paper. We assume that at the time of the handoff triggering, the new path is known.

Finally, we do not discuss under what conditions a handoff may be triggered. We assume that the bandwidth of the new connection exceeds the old one, and that this difference persists for a certain time.

We restrict handoffs to application-layer connections because we want to test and evaluate our solution in existing networks, such as the Internet. A path thereby consists of a set of end-systems, running filters or servers. Using new network architectures, such as Active services [1] or active networks, would allow for an instantiation of filters inside a network.

Figure 2 shows a sample scenario of a handoff. The components of the application are distributed over three locations. Assume that a server at ETH Zurich originally sends its video data to a client at UFMG in Brasil (e.g., for teleteaching). The link between ETH and UFMG has only limited bandwidth. At some point in time, a client at CMU (Pittsburgh) joins the transmission. The video stream for the UFMG client could now be switched from the direct connection to a possibly better connection *via* CMU. In this case, a new connection between UFMG and CMU must be established. Then, the UFMG client switches to this new connection.

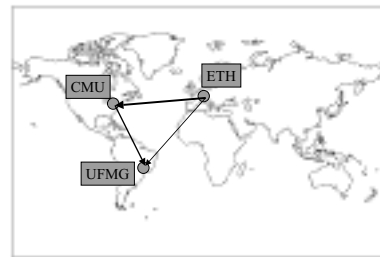


Fig. 2. A handoff scenario: the client at UFMG may switch from a direct connection to ETH to an indirect via CMU, depending on the bandwidth

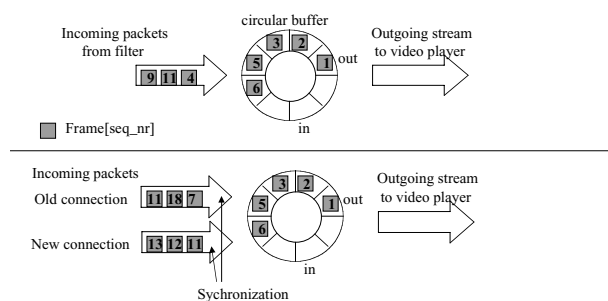


Fig. 3. Handoff at the client. Above: situation with only one connection streaming (phases 1,2,4 and 5). Below: synchronization required during phase 3.

III. APPLICATION MODEL

Figure 3 shows the implementation of the handoff at the client. Above, the situation for one incoming connection is shown. Typically, the incoming packets from the filter are put into a buffer according to their sequence number, and the video player reads the frames from this buffer. Below, the situation during the synchronization phase is shown. The synchronization of the two streams is done by the buffer: a packet that arrives over both connections is inserted only once - the second packet is simply discarded.

This section shows the influence of two parameters during a handoff: the correct setting of the first frame to be transmitted by the new connection and the timing for the shutdown of the old connection. Based on these experiences, Section IV will describe an algorithm that tries to optimize the handoff for different situations.

A. Stream Initiation

When a handoff is triggered, a part of the video has already been transmitted via the old connection. The new connection must start at an offset to match the

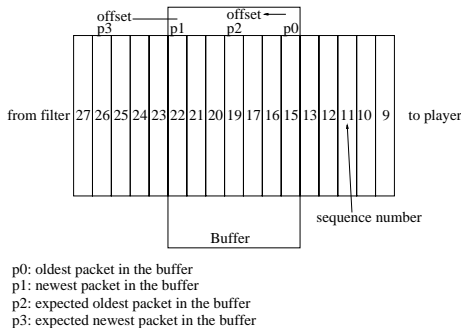


Fig. 4. Handoff timing: different methods to predict the first frame to be sent via the new connection

old stream. We call the number of the first frame that is to be transmitted over the new connection the *start packet*. In Figure 1, the client sends a set-start-packet request to the server, indicating this start packet.

Different definitions are possible as to what the start packet should be. Between the setup and the arrival of the first packet, frames drain from the buffer and packets may arrive via the old connection. The goal is to avoid that the same frame is transmitted over both connections because the buffer has to discard it. At the same time, however, no gap should be created between the frames of the old connection and the new one.

Figure 4 depicts four methods to define the start packet. It shows the video packets, streaming from left to right. Packets are first transmitted (packets 23 to 27), then, when they arrive at the client, are put into the buffer (packets 15 to 22). The video player fetches the frames from the buffer and displays them (packets 9 to 13).

The four methods are:

1. The oldest frame number in the buffer, i.e. the one that will be displayed next (p0).
2. The newest frame number in the buffer (p1).
3. The frame that is *expected* to be displayed when the first packet arrives over the new connection (p2).
4. The frame that is *expected* to have arrived last at the buffer when the first packet arrives over the new connection (p3).

The first possibility is expected to be of use if the buffer is missing many frames, e.g., if the old connection has deteriorated significantly before an alternate connection has been found. The new connection could then "repair" the holes of the missing frames. But method p0 ignores that frames may still be buffered or being transmitted over the old connection. Thereby, packets may be transmitted twice and have to be discarded.

An alternative is to send the last sequence number in the buffer (p1). If the old connection is known to be disconnected in advance, e.g., because the server is going down, the new connection could kick in at the correct position.

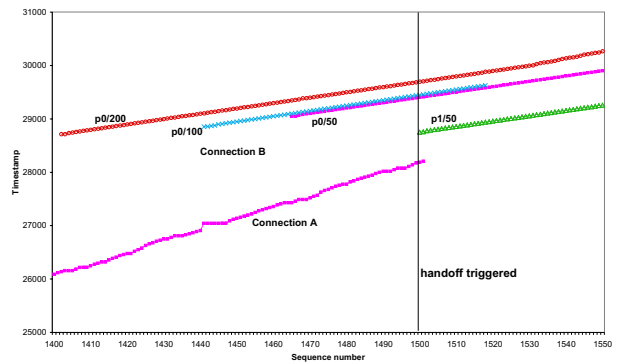


Fig. 5. 3 Experiments with variable buffer size (50,100,200 packets) for method p0, and 1 for p1.

Method p2 takes p0 as a base, but tries to avoid the unnecessary sending of packets that may arrive late, i.e. when the player has already consumed them. The sequence number for p2 is calculated as

$$p2 = p0 + \frac{rtt}{f} \quad (1)$$

where rtt is the round-trip time between the client and the new server, and f the average frequency with which the player fetches frames from the buffer.

Finally, p3 combines the ideas of p1 and p2. This method is used when the switching of the connection is known in advance and the old connection is still sending at a good quality.

Figure 5 shows 3 experiments with p0, varying the buffer size (50, 100 and 200 packets), and a single experiment with p1. On the x-axis, the incoming frame sequence numbers, on the y-axis the receiving time stamp of the packet at the client in milliseconds are shown. The old connection A is the same for all experiments. When the packet number 1500 arrives at the client, a handoff is triggered. With method p1, the new connection starts the transmission with the same packet number, and is independent of the buffer size. In contrast, with method p0, packets with lower sequence numbers are sent over the new connection. The buffer size determines here how many packets are sent previous to the handoff.

Two effects are visible: first, with p0, packets may arrive twice (over connection A and B). If the old connection has sent its packets contiguously until the handoff, as in Figure 5, most packets over the new connection have to be discarded up to the packet at the handoff trigger. The discard packet rate for the shown experiment is 98% for p0/200 and 43% for p0/50 (packets between frame 1400 and 1500). If, however, the old connection had had a lot of loss, the new connection would have been able to insert its packets into the stream and improve its quality. Second, using method p0 with a large buffer also leads to a later packet arrival: with strategy p0/200, a packet

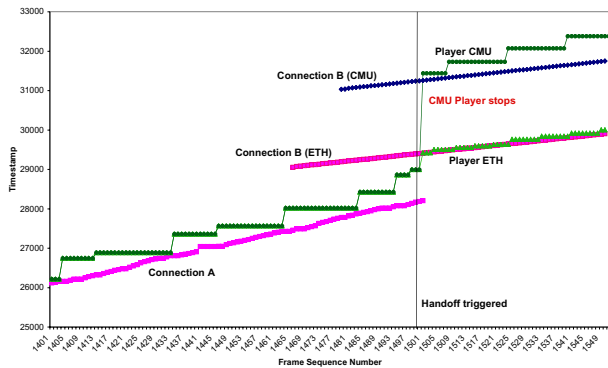


Fig. 6. Handoff to local filter/server (ETH) and remote (CMU).

arrives about 150 ms later than the same packet with p0/50.

B. Length of the synchronization phase

The duration of the synchronization phase can be determined by the application, unless the old connection breaks down completely. With this duration, the application can influence the incoming of the frames. Although the client cannot influence what frame numbers arrive from the filter, it can wait with the disconnection until the synchronization of the two streams is accomplished. In combination with the stream initiation, a late disconnection may close the gap till the new stream is fully sending. However, managing two streams may increase the load on both the network and the client. This is expressed by an increased packet discard rate in the buffer.

Figure 6 shows the situation of an immediate handoff, i.e. the old connection is disconnected immediately when the new connection has been established, but without waiting for the first packet to arrive. The two experiments shown here are (i) a handoff to a filter within the same LAN (ETH) and (ii) to a remote filter (CMU) using p0/50 as handoff strategy. The graph shows again connection A, which is the same for both experiments. After the handoff, packets from the remote server arrive much later than from the local server. The handoff to the local player is not noticed because the new frames arrive quickly enough. In contrast, the player that switches to the remote server has to wait until new frames arrive and the movie stops.

The gap in the transmission could have been omitted if the old connection had continued to send data. Therefore, an appropriate timing for the connection shutdown influences the video quality during a handoff.

IV. SELECTION AND EVALUATION

The above results help the application to improve the quality during the handoff. We have derived a sim-

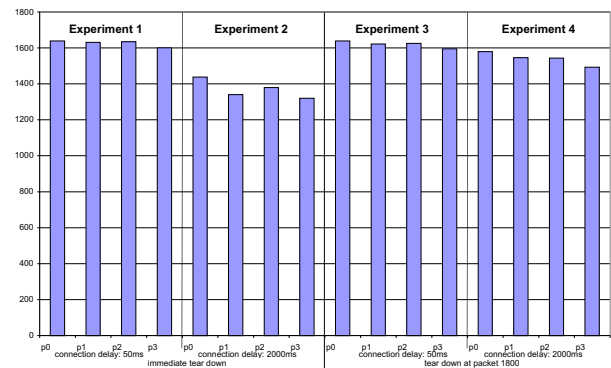


Fig. 7. The number of correctly received frames after frame 1400 (max: 1700 frames)

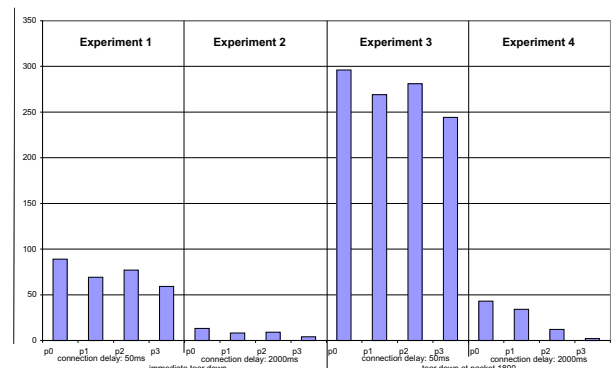


Fig. 8. The number of frames discarded during the handoff (between frame 1400 and frame 1800)

ple decision scheme that allows the application to determine the appropriate stream initiation:

- if *loss high* and *delay large* use p0;
- if *loss high* and *delay small* use p1;
- if *loss low* and *delay large* use p2;
- if *loss low* and *delay small* use p3;

Here, *loss* describes the number of lost packets of the old connection, and *delay* the expected delay of the new connection between the sending of the set-start-segment and the arrival of the first packet at the client.

We perform experiments with two extreme values for each parameter. For *loss*, we compare a connection that is disconnected immediately at the handoff triggering (corresponding to a loss rate of 100%) to a connection with 5% loss that is disconnected after the new connection is fully sending (packet 1800) For *delay*, we compare a handoff to a server within the same LAN (20 ms latency) to a WAN-server with 2000 ms latency. For each combination of these parameters, we measure for the each method p0 to p3 the number of correctly received frames. Hemy et al. [5] use this number to compare the quality of two MPEG video streams. In addition, we measure the number of frames that are discarded during the handoff phase, i.e. between frames 1400 and 1800. The buffer size is kept at 100 packets.

Figure 7 shows the number of correctly received frames. Only frames after packet 1400 are counted, as the previous quality is the same for all methods. The figure shows that a handoff to a local server (experiments 1 and 3) allows for more frames to be received correctly. However, with an appropriate strategy, also the handoff to a remote server almost reaches the same quality. Connecting to a local server shows little variation in the number of received frames, but large differences in the discard rate (Figure 8). Especially if the old connection continues to send (experiment 3), many frames have to be discarded. Transmitting unnecessary packets may cause losses of needed frames at the client, which would decrease the number of correctly received frames and degrade the quality. Our algorithm chooses p1 for experiment 1: p1 receives as much packets as p0 and p2, but has a lower loss rate. p3 loses even fewer packets, but the number of correctly received frames is smaller. In experiment 3, the difference in the discard rate is considered more important than the small difference in the number of received frames. Therefore, method 3 is chosen.

For the experiments with a remote server (experiments 2 and 4), the loss rate is much lower. If the old connection is shut down immediately (experiment 2), p0 is chosen because the number of frames is the highest and the loss rate is neglected. If the old connection continues to send (experiment 4), the differences in the number of received frames is smaller. Here, the loss rate may be taken into account, so that method p2 is chosen.

V. RELATED WORK

Handoff is applied in from mobile communication. When a user moves from one cell to another, the communication is moved from one base station to another. Snoeren and Balakrishnan [6], e.g., describe a handoff scheme for mobile hosts, but at transport layer (TCP). At a lower layer, Bejerano et al. [2] or Campbell [3] consider rerouting-algorithms for handoff. In contrast, our work is based on application-layer connections because these allow the application to actively influence the handoff and synchronize the video streams.

Stemm and Katz [7] treat the problem of the handoff speed: when a user moves between cells, the delay to establish the new connection may be too large for the application. This, e.g., holds for telephony. Some multimedia applications can use the buffer to overcome the delay, but, as we have shown, synchronization is needed.

Fei et al. [4] describe an active client buffer that must select the frames to be downloaded. The motivation here is that a user interactively modifies the video stream (moving forward or backward). The ac-

tive buffer should keep the playback point synchronized in the middle of the buffer. Although the motivation is very different from our work, the idea that the buffer tries to optimize the video stream is similar.

VI. CONCLUSIONS

Multimedia applications can switch from an old, slow, lossy connection to a new one. Such a handoff can be an alternative when adaptation is no longer efficient enough to adapt the data stream to the available resources. Setting up the video stream over the new connection and synchronizing the two connections during the handoff is important to allow for a smooth delivery of the video stream to the video player.

We have investigated the parameters that influence the handoff and thus the quality of the delivered video stream. Based on this investigation, we have derived an algorithm to select the first frame that the new connection should transmit. This algorithm allows for an almost maximal number of frames received at the client while at the same time minimizing the number of frames that have to be discarded. Thereby, the handoff is at worst only slightly noticed by the application.

From this evaluation we conclude that a handoff is another viable approach to improve the quality of multimedia applications

REFERENCES

- [1] E. Amir, S. McCanne, and R. Katz. An Active Service Framework and its Application to Real-time Multimedia Transcoding. In *ACM SIGCOMM*, pages 178–189, Vancouver, BC, Canada, September 1998.
- [2] Y. Bejerano, I. Cidon, and J. Naor. Efficient Handoff Rerouting Algorithms: A competitive on-line algorithmic approach. In *IEEE Infocom 2000*, Tel Aviv, Isr, March 2000.
- [3] A. Campbell. Mobiware: QOS-aware middleware for mobile multimedia communications. In *7th IFIP International Conference on High Performance Networking*, White Plains, NY, April 1997.
- [4] Z. Fei, M. Ammar, I. Kamel, and S. Mukherjee. Providing Interactive Functions through Active Client Buffer Management in Partitioned Video Broadcast. Technical Report GIT-CC-99-09, Georgia Institute of Technology, Atlanta, GA, September 1999.
- [5] M. Hemy, P. Steenkiste, and T. Gross. Evaluation of adaptive filtering of MPEG system streams in IP networks. In *IEEE Intl. Conference on Multimedia and Expo 2000*, New York, 2000.
- [6] A. Snoeren and H. Balakrishnan. An End-to-End Approach to Host Mobility. In *6th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '00)*, Boston, MA, August 2000.
- [7] M. Stemm and R. Katz. Vertical Handoffs in wireless overlay networks. *ACM Mobile Networking (MONET), Special Issue on Mobile Networking in the Internet*, 1998.