

Multipath streaming in Best-Effort Networks

Roger Karrer and Thomas Gross
Laboratory of Software Technology
Department of Computer Science, ETH Zurich
CH-8092 Zürich, Switzerland

Abstract—Multipath streaming is used by resource intensive applications to stream their data over multiple, disjoint paths, thereby cumulating the resources of the different subpaths. The trend towards application-layer implementations of communication protocols allows the deployment of a multipath streaming protocol in the current Internet. However, because resource availability fluctuates in the Internet, a multipath streaming protocol must be combined with other mechanisms, e.g., adaptation, to address these fluctuations. This paper describes two approaches to combine adaptation and multipath streaming. The first approach separates the two mechanisms and thereby allows multipath streaming to be transparent to the application, whereas the other approach combines the two mechanisms in the application context.

This paper compares the two approaches along various parameters. It shows that the first approach is easier to deploy from an engineering point of view, but the separation of multipath streaming and adaptation yields significant drawbacks. Especially, synchronization problems due to different latencies along the paths that form the multipath setup may lead to a significant drop in the quality of the data.

I. INTRODUCTION

The Internet is often not able to provide enough bandwidth for multimedia applications. Mechanisms are needed that allow an application to deal with shortages. Several approaches have been presented in previous work, e.g., resource reservation or adaptation.

However, there exists another approach to deal with this problem: since the Internet provides numerous paths between any two hosts[1], if a single transmission path does not suffice, the data can be streamed over multiple paths in parallel. Multipath streaming has been explored in the context of QoS networks (e.g., [2], [3]) and been proposed for the Internet (e.g., [4]), but this technique has not been deployed in best-effort networks so far.

One reason why multipath streaming has not been explored further in the Internet is that multipath streaming is hard to provide at the network layer. However, the recent trend in networking to consider routing or other networking activities at the application layer[5], [6], [7] opens new opportunities, and an application-layer multipath streaming mechanism is far easier to implement and to deploy than one within IP.

Multipath streaming in a best-effort network poses a set of new research issues. The main problem is that resources in such a network are not known, and, even worse, they dynamically change over time, so that continuous adaptation is needed even for a multipath streaming setup. The combination

of adaptation and multipath streaming, however, is far from easy:

- How can a multimedia stream efficiently and dynamically be split onto different substreams?
- Every path is likely to have different timing characteristics, e.g., latency or jitter. How can the different substreams be synchronized?
- How does the adaptation mechanism interact with multipath streaming? In contrast to QoS networks, neither bandwidth guarantees nor bandwidth information is directly available from the Internet; one aspect of this problem is selecting the various paths for a multipath setup.

We have implemented two different versions of multipath streaming for an existing adaptive MPEG application [8]. This application uses a frame-dropping filter to adapt the sending rate to the resource fluctuations in real networks, such as the Internet. The first version implements the multipath streaming completely transparent to the application. This approach follows the layering approaches of the traditional Internet design and current overlay networks by separating applications from the data transport.

In contrast, the second approach focuses on an integration of the data transport into the application context and allows a combination of adaptation and multipath streaming. This integration results in a software architecture that is similar in style to those scenarios that provide application-driven data transport. The motivation in both cases is also the same: some parameters that decide how the data should be transported are only known within the context of the application.

The comparison of the two approaches yields interesting insight about the benefits of both approaches. Both approaches significantly differ in their approach to address the questions listed above. Because both approaches operate at the application layer, we are able to compare the two approaches using real Internet experiments.

II. APPLICATION-LAYER MULTIMEDIA STREAMING

Adaptive filters, such as frame dropping filters for MPEG streams, are used to reduce the amount of data streamed over fluctuating, low-bandwidth connections [9], [8]. In times of congestion, these filters estimate the amount of data that can be sent over a connection and drop frames of less importance while maintaining synchronization between audio and video. When congestion is reduced, the “drop level” is reduced and more (or all) of the data is sent. Multipath streaming provides another dimension to address the bandwidth requirement problem. Multiple, disjoint paths provide another way to increase the available bandwidth to an application.

The work presented in this paper was supported, in part, by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

Criteria	TLMS	ALMS
filter changes	no	coordinator
client changes	no	merging
other changes	socket	no
portability	yes	no
split granularity	(UDP) packet	MPEG frame
number of streams	unlimited	frame layout
split metric	bandwidth	packet loss
adaptation metric	packet loss	packet loss
stream adaptation	all together	individual

TABLE I
COMPARISON OF TLMS AND ALMS.

increase or decrease the filtering level must no longer be handled by the corresponding filter but must be forwarded to the coordinator because a change in the filtering level for one path affects all paths. If one path reduces the filtering level to transmit additional frames, the coordinator must ensure that the streams are still complimentary. Similarly, if one stream increases the filtering level, the coordinator must ensure that the high priority frames are still transmitted first. Assume, e.g., that a multipath streaming filter has 2 substreams, one transmitting I-frames, the other P- and B-frames. When the first connection slows down, it cannot simply start dropping I-frames. The coordinator must adjust the filtering level of both paths. An I-frame that can no longer be transmitted over the first path must be transmitted over the second path, which in turn may have to start dropping B- and P-frames.

C. Comparison

Table I compares the two approaches according to different design, implementation and performance criteria. To enable an existing conventional single-path application, TLMS limits any source code changes to using the new splitting and merging sockets, whereas ALMS requires changes to the application code. These changes may be hard to implement, especially if the application's module structure does not encapsulate the communication activities.

The splitting granularity of the two implementations is different. TLMS, which is closely related to the network, works with UDP packets. In contrast, ALMS works with application-specific data types (MPEG frames in this case). The splitting granularity for ALMS is coarser than for TLMS because an average MPEG frame is larger than a UDP packet.

The number of parallel streams for TLMS is only limited by the host communication (network) interface (and the host's memory bandwidth), so for many settings, the number of paths can be quite large. ALMS on the other hand may be limited by application restrictions. The MPEG filtering process distinguishes frames within a GOP, but not among them. That is, the filter knows whether a B-frame is the first or the second after an I-frame, but it does not know the difference between the first B-frames in different GOPs. As a consequence, the number of parallel streams is limited by the number of frames in a GOP. Typical numbers are 12 or 15 possible parallel streams.

The split metric is also related to the network for transport-layer stream splitting. Bandwidth is a typical split metric, but others network-related metrics, e.g., error rate for wireless connections, could be used as well. In contrast, the coordinator of

Paths		1	2	3	5	10	15
Sysload	ALMS	2%	2.6%	3.4%	5.0%	7.6%	11.2%
	TLMS	2.1%	2.1%	2.0%	2.1%	2.0%	2.2%
Delay	ALMS	1.00	1.06	1.06	1.06	1.07	1.07
	TLMS	1.02	1.02	1.01	1.02	1.02	1.02

TABLE II
PERFORMANCE OF TLMS AND ALMS.

the application-layer splitting distributes frames (not packets) according to packet loss along the paths. The adaptation metric is packet loss for both implementations, but the TLMS adapts all substreams together, as if it were one single stream, whereas ALMS adapts each substream individually.

IV. EVALUATION

Table I shows that the two multipath streaming approaches not only differ architecturally but also in many parameters. All these differences influence the streaming performance of each approach in various ways. This section discusses this influence along three lines of thought. A first part looks at the overhead of each approach with respect to the system load on a filter and the delay overhead for the streaming. The second part discusses synchronization problems in multipath streaming, e.g., how the two approaches deal with large delay differences along the subpaths. Finally, the third part discusses the interaction between multipath streaming and adaptation and its impact on the final multimedia quality.

A. Performance

Table II compares the load imposed on the filtering host and the delay for data forwarding of the two approaches, as a function of the number of paths in the multipath setup. The system load is measured on a 933 MHz Pentium III with 256 MB running Linux 7.2. The results are averages of 100 tests and are expressed in % of the total system load. The results show that the load imposed by TLMS is independent of the number of paths, whereas for ALMS the load increases almost linearly. The reason for this increase is that the whole filtering process is replicated for each additional stream in the application-streaming approach. ALMS has therefore a significantly worse scalability than TLMS.

The second set of rows shows the delay overhead of multipath streaming normalized to the corresponding single-stream implementation. For TLMS, the delay overhead is measured between the sending command by the application and the sending command by the splitting socket. For ALMS, the measurement compares the time between the fetching of a frame from the buffer and the sending of a frame. The numbers show that the delay overhead is negligible; in practice the absolute values are in the order of milliseconds.

A second issue related to the performance is the efficiency of the stream splitting. For ALMS, this splitting is implicitly integrated into the adaptation mechanism. The filtering process of each substream fetches the data from the buffer at its own rate, matching the capacity of the outgoing streams. In contrast, an explicit splitting mechanism is needed for TLMS. This splitting mechanism, which is part of the splitting socket,

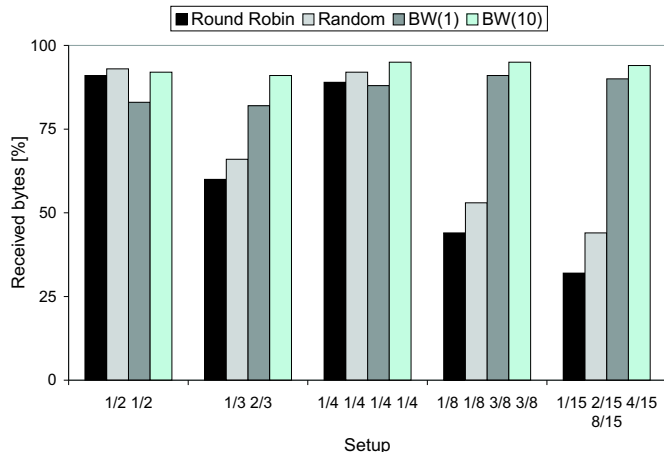


Fig. 3. Transmitted packets (network emulation based on real traces)

receives UDP packets from the application and must distribute them onto the outgoing connections. Three principal splitting strategies are evaluated in this paper. Round robin distributes the packets sequentially at an equal share. Random uses a uniform distribution. Finally, the third strategy measures the available bandwidth along each path and distributes the frames based on the actual bandwidth availability. These bandwidth measurements are also integrated into the splitting and merging socket and are therefore hidden to the application. We report on two different bandwidth splitting strategies: $BW(1)$ splits the streams according to a bandwidth sample that is taken anew every second, whereas $BW(10)$ takes a value averaged over the last 10 samples to smooth heavy fluctuations.

To compare the different splitting strategies, we collected a set of Internet bandwidth traces and emulate them between the filter and the client. Two different videos are run over these emulated traces. The amount of transmitted data is measured as a function of the splitting strategy, the number of parallel streams and the bandwidth distribution among the paths. Figure 3 shows the amount of transmitted bytes, relative to total bytes sent by the filter, in %. A value of less than 100% shows that some packets had to be dropped because they could not be sent due to congestion. The x-axis denotes the relative bandwidth distribution of the paths. A value of $(1/2, 1/2)$, e.g., means that the multipath setup consists of two subpaths with an equal average bandwidth (50%). To achieve this distribution, the original traces are scaled so that the average bandwidth of a trace matches the target average bandwidth, while still keeping its fluctuation patterns. The total amount of streaming bandwidth is always chosen to match the bandwidth requirements of the video for this experiment.

The first and the third group of columns in Figure 3 exhibit an equal share of bandwidth for each subpath. The results of all 4 splitting strategies are similar. In contrast, the two static splitting strategies (random and round robin) significantly drop in performance when the bandwidth share is no longer equal for the different paths, as shown in the 2nd, 4th and 5th group of columns. Regarding the two dynamic splitting strategies, we note that $BW(10)$ performs 5-10% better than $BW(1)$.

We conclude from this figure that a good splitting strategy is essential to get a decent performance for TLMS. Static

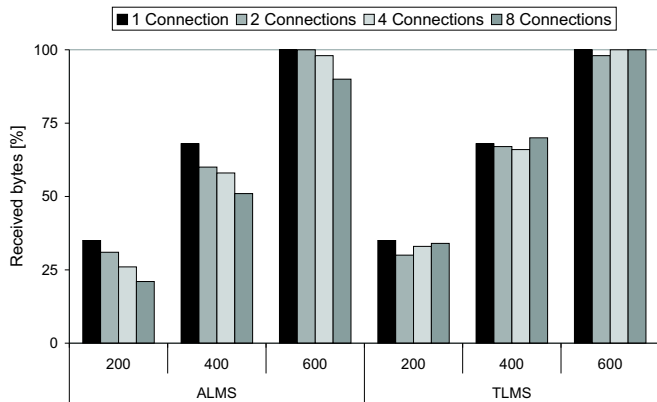


Fig. 4. Amount of received data as a function of the number of parallel connections, the available bandwidth.

splitting strategies are not useful when the paths do not have an equal bandwidth, which is hardly the case in real best-effort networks.

TLMS and ALMS differ in a third factor: splitting granularity and splitting metric. The result of this difference is shown in Figure 4. The experiment that leads to this figure is similar to the previous one. Two different videos are sent over the emulated traces for both approaches. The x-axis denotes the total amount of available bandwidth for the multipath setup, and the bandwidth of each of the n paths has $1/n$ -th of the total bandwidth. TLMS uses $BW(10)$ as a splitting metric. The y-axis shows the amount of received bytes, in % of the total bytes sent.

The performance for a single connection is always higher because it does not have to pay the multipath streaming overhead. Interestingly, the relationship within one group of bars is different for TLMS and ALMS. The performance of ALMS decreases with an increasing number of parallel paths while it increases for TLMS. This different behavior is due to the differences in the splitting mechanism. The decrease in performance for the ALMS is due to the coarse and slow splitting-adaptation mechanism. The adaptation mechanism is lazy to ignore small bandwidth fluctuations to smooth the video and, especially, to avoid packet loss. If a request to change the filtering level arrives in ALMS, this request has a much larger impact than for the corresponding request for TLMS because frames are a factor of 2-5 larger than UDP packets. It is therefore much more difficult to coordinate the different filters.

In contrast, TLMS is more efficient for a larger number of connections. We explain this observation with bandwidth mispredictions. If there are only two parallel connections, a misprediction has an impact on 50% of the of the streams, whereas with 8 connections, only $1/8$ of the substreams is affected.

The discussion shows that splitting granularity and metric affect both approaches in a different way. Comparing the two approaches, ALMS is generally better suited for a lower number of parallel streams. TLMS does not only scale better to a large number of streams from an implementation's point of view, it is also more efficient when the number of paths increases.

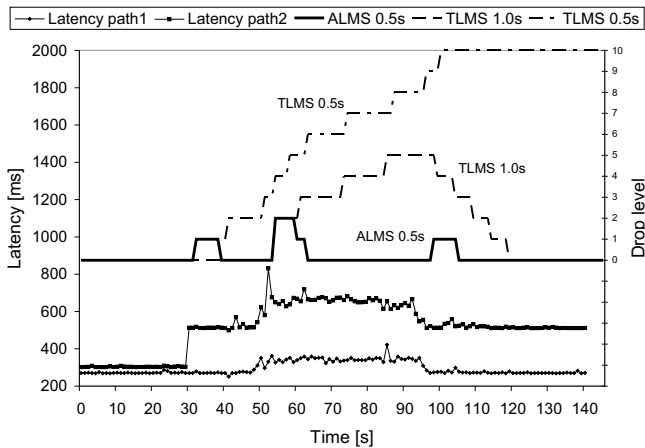


Fig. 5. Change of filtering level due to synchronization problems

B. Synchronization

Real Internet paths are not only different in the available bandwidth, they also differ in other metrics, such as delay or jitter. These differences cause a synchronization problem for multipath streaming. Packets that travel over a slower path may arrive late at the client. In the worst case, they must be discarded because they are too late to be displayed. Discarding or losing packets due to congestion has a negative impact on the video quality. Because a packet is smaller than a frame, a single packet loss may invalidate other packets as well, i.e., they are transmitted in vain. Even worse, the loss of a packet containing an I-frame leads to an invalidation all the frames that depend on this I-frame.

The ability to deal with a large difference in the path latencies varies for the two multipath streaming approaches. To show the differences, a video stream is sent over two emulated paths based on Internet traces. The latency of these traces is shown on the lower part of Figure 5. The x-axis denotes the time, the *left* y-axis the latency in milliseconds. The two lower lines show the latency of the two paths that form the multipath. Their latency is similar until $t + 30$ seconds, when one trace doubles its latency in two major steps whereas the latency of the other path is only slightly increased.

The increase in the latency difference has an impact on the filtering level if the client buffer is not much larger than the latency difference. We therefore set the buffer size corresponding to data for 0.5 seconds and 1.0 seconds, resp., for TLMS and to 0.5 seconds for ALMS. These buffer size values are relatively small for Internet applications, but the latency difference in the real Internet may also be much larger than for the shown traces, so the same effect may happen on a larger buffer.

The top of Figure 5 shows the filter drop level (y-axis on the *right*) for TLMS and ALMS. (In the case of ALMS, the filtering level of the coordinator is shown). All three implementations increase their filtering level 30 seconds into the experiment as a reaction to the increase in the latency. The ALMS reduces the drop level again after 10 seconds. The reason is for this reduction is that the adaptation mechanism (of the single-path streaming) is able to adjust the speed of a stream (the data rate) by delaying the sending of a packet

at the filter to keep a steady data flow to the client. Because ALMS adapts every substream individually, it can increase the speed of the high latency connection until the two substreams are synchronized and no more packets are dropped. The same adjustment in the speed is visible after 53 seconds and 96 seconds.

TLMS does not have the ability to react accordingly because the multipath streaming is hidden. The adaptation algorithm only notices that some packets are discarded by the client buffer. Assuming that the discarding is a result of congestion, TLMS increases the drop level at the filter. However, this increase has no effect because the latency difference is still there. The “0.5s buffer” client continues to increase the drop level until the maximal level of 10 is reached (i.e., 10 out of the 12 frames in each GOP are filtered out). The effects on the video quality are devastating, especially compared to ALMS with the same buffer size! Doubling the buffer size for TLMS to “1.0s” also leads to an increase in the filtering level, but with two differences with respect to the “0.5s” buffer. First, the filtering level increases later because the larger buffer can hide the effects of the first latency increase. Second, when the latency difference decreases again after 95 seconds, the filtering level is also reduced.

We make three observations here. First, the buffer size is a critical parameter for multipath streaming, especially at the transport layer. A larger buffer is advantageous to deal with synchronization problems. However, there are other factors that limit the buffer size, e.g., constraints imposed by the client hardware.

Second, the ability to deal with synchronization problems is limited for TLMS *because* the multipath mechanism is separated from the adaptation mechanism. The adaptation mechanism has no way to find out the right cause of the packet loss in this implementation. One way to deal with the synchronization would be to integrate synchronization into the multipath streaming (socket) as well. However, several changes are necessary. First, the socket has to measure the latency along each path. Second, latency differences must either be reported to the application or be compensated using a sophisticated packet scheduling where packets are buffered and scheduled according to the target arrival time. Finally, the API has to be opened to allow an application to specify which latency differences can be tolerated and what to do if these bounds cannot be met. The last point violates the transparency assumption, however, leading towards an integrated adaptation–multistreaming approach again.

Finally, synchronization is only one kind of the “asymmetry” problems that can occur with multipath streaming. In a wireless environment, e.g., the different paths have an “asymmetric” probability of losing packets due to transmission errors. ALMS can detect these differences and react, e.g., by sending the most important frames over the most reliable connection. A transport-layer implementation cannot deal with an unequal error rate because it does not know that multiple paths are used. Implementing a scheme to send high priority frames over the most reliable connection within the splitting socket would be very difficult, especially because the splitting socket does not know which part of a frame is contained in a packet.

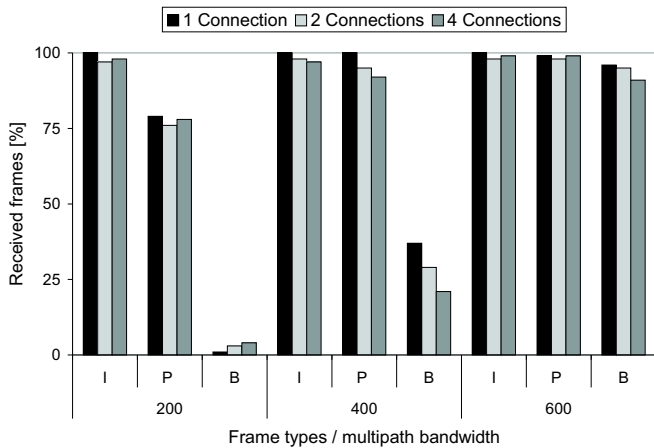


Fig. 6. Transmitted video frames as a function of the available bandwidth of the multipath for *transport-layer* streaming.

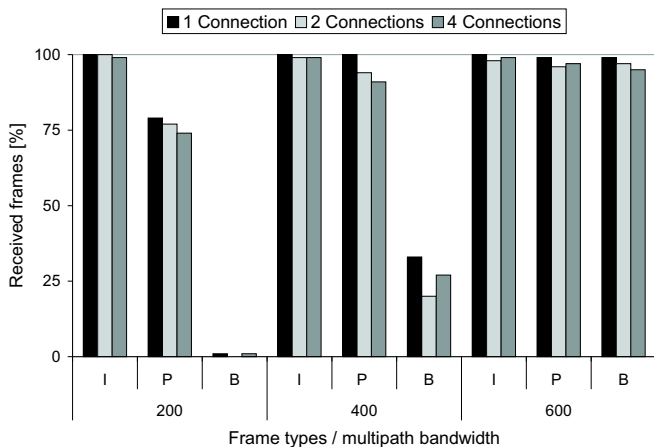


Fig. 7. Transmitted video frames as a function of the available bandwidth of the multipath for *application-layer* streaming.

C. Multipath streaming and adaptation

The performance analysis in Section IV-A has measured only the amount of transmitted bytes. Important for the video quality at the client, however, is not the amount of transmitted data, but the number of correctly displayable frames, which determines to the video quality.

To assess the video quality at the client, we stream again two different videos over the emulated Internet traces with a varying number of paths. Each run is repeated 50 times with different Internet traces. The traces are again scaled to match the average target bandwidth. For each run, we measure the number of displayable frames, separately for each frame type. The results of this experiment are shown in Figures 6 and 7.

Both figures show the number of displayable frames on the y-axis, grouped by frame types, as a function of the multipath bandwidth and the number of paths that make up the multipath bandwidth. Both figures are similar in the layout of the received frames, that is, if the multipath bandwidth is low, only I-frames are transmitted. P-frames are sent when more bandwidth becomes available, and finally B-frames get through. This shows that the adaptation and multipath streaming mechanism work well together. Analyzing the data further reveals that ALMS results in a slightly higher number

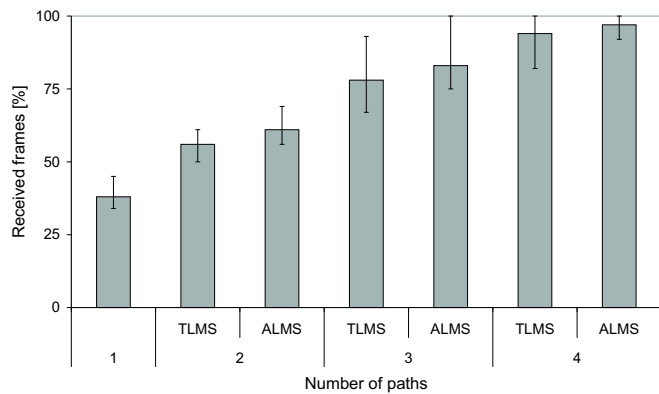


Fig. 8. Multipath streaming in a best-effort network.

of frames. Especially the probability of losing an I-frame is almost zero, which means that this approach almost perfectly combines adaptation and multipath streaming. The results for TLMS, however, are almost as good as those of ALMS. However, it must be noted that these values are achieved only by a good setup of the transport-layer streaming, i.e., we used the $BW(10)$ splitting strategy, the paths are free of error, and the client-side buffer is large enough to avoid synchronization problems. Any of these problems can significantly reduce the performance of TLMS whereas ALMS is more robust and only slightly affected.

D. Multipath streaming in best-effort networks

The previous discussion compares ALMS and TLMS and highlights its pros and cons; the net benefit of multipath streaming for an application is shown in Figure 8. It shows the number of received frames as a function of the number of paths and the multipath streaming strategy. For this experiment, a video stream of 1.5 Mbps is streamed over a different number of parallel connections. Each connection is emulated with Internet traces whose average bandwidth is between 300 and 500 Mbps. Every experiment is repeated 10 times with different traces. Every bar shows the average number of frames received, with the min and max numbers. The first bar denotes the number of received frames over a single path, i.e., the common single-path streaming. All other bars use multipath streaming. With two streams, the number of received frames is almost doubled and the video quality is greatly increased. Finally, four parallel streams transmit the video almost in the original quality. These results show that the presented approaches to multipath streaming can significantly increase the video quality. The key to this improvement lies in the exploitation of the resources that are available in the network via alternative paths.

V. RELATED WORK

The idea of using multiple paths to stream data over a network is not new at all. The networking community has applied the graph theory algorithms, developed in the 60s, to QoS networks. Rao and Batsell [3], e.g., show the relationship between the network metrics, such as bandwidth and latency, and the routing algorithms. Because several algorithms to evaluate a

graph are NP-complete, the authors describe approximations that can be applied to QoS networks. A summary of further routing algorithms is described by Chen and Nahrstedt [2], who also refer to multipath routing as a future direction in networking when talking about next-generation high-speed networks. This paper, in contrast, evaluates two solutions for multipath streaming in a best-effort network where no resource guarantees are available. Its focus is therefore on the interaction between adaptation and multipath streaming.

Multipath streaming in the Internet is closely related to the recent advances of overlay networks [6], [5], [10]. Overlay networks allow the routing of data based on application-specific metrics. So far, overlay networks have only considered single-path streaming. Multipath streaming is an almost natural extension to the capabilities of overlay networks. Especially the transport-layer multipath streaming implementation exactly fits into the model of overlay network routing.

Similarly, the presented approach also fits well into the Internet Indirection Infrastructure i3 [18]. This infrastructure defines a rendez-vous based model for different communication types that can be expressed by indirection, e.g., multicast or mobility. The same triggers which are injected into the network to allow the implementation of different communication types can also be used to implement multipath streaming. A first trigger (a kind of filtering trigger) splits the data and sends each stream to a different, secondary trigger, and so forth until each substream reaches the final destination.

Another area that is closely related to this work are active services [7]. Active services advocate for the placement of application-specific services inside a network. The authors propose to use video gateways (streaming filters in our terminology) inside the network. The adaptive MPEG filter in this paper is in fact implemented as a kind of service that is embedded in a larger framework for future applications [17].

VI. SUMMARY AND CONCLUSIONS

This paper presents two different approaches to multipath streaming in best-effort networks. Multipath streaming in best-effort networks is challenging because it must be combined with adaptation mechanisms that address resource fluctuations. The first approach, transport-layer multipath streaming (TLMS), splits the data after the adaptation, whereas application-layer multipath streaming (ALMS) splits the data first and then adapts every substream individually.

From an engineering point of view, TLMS has the advantage that no changes are needed in the (possibly complex) application code. The splitting can be hidden from the application, which makes TLMS also portable to different applications. It fits conceptually well into other overlay networking approaches. TLMS also has a low overhead and a low impact on the system performance.

However, TLMS has significant disadvantages. Every path in a best-effort network has its own dynamic behavior, e.g., different latencies or error rates. Any asymmetry in this behavior must be addressed by an application, e.g., to maintain synchronization. TLMS only sees the effects of the asymmetry on the whole stream, but it is neither able to identify the misbehaving path nor can it take appropriate reactions.

In contrast, ALMS integrates the multipath streaming and the adaptation into the application context. Because it first splits the data and adapts every substream individually, it is able to deal with path asymmetries.

We also observed that although TLMS has a lower overhead and is more efficient with respect to the number of bytes, ALMS is able to deliver a better video quality because the adaptation and the splitting mechanisms use application-layer metrics. The mapping of networking metrics (bandwidth) onto application-layer metrics (MPEG frames) is not easily performed. So we caution to expect that the promising performance results of overlay network will automatically translate into noticeable quality increases for an application.

REFERENCES

- [1] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of Internet path selection," in *Proceedings of ACM SIGCOMM '99*, Boston, Massachusetts, Aug. 1999, pp. 289–299.
- [2] S. Chen and K. Nahrstedt, "An overview of quality-of-service routing for the next generation high-speed networks: problems and solutions," *IEEE Network Magazine, special issue on transmission and distributed of digital video*, vol. 12, no. 6, pp. 64–79, Nov. 1998.
- [3] N. Rao and S. Batsell, "QoS routing via multiple paths using bandwidth reservation," in *Proceedings of IEEE Infocom '98*, San Francisco, CA, Mar. 1998, pp. 11–18.
- [4] C. Villamizar, "OSPF Optimized Multipath (OSPF-OMP)," Internet Draft (draft-ietf-ospf-omp-02), work in progress, Feb. 1999.
- [5] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, "Resilient overlay networks," *Operating Systems Review*, vol. 35, no. 5, pp. 131–145, Dec. 2001.
- [6] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the Internet using an overlay multicast architecture," in *Proceedings of ACM SIGCOMM '01*, San Diego, CA, Aug. 2001, pp. 55–67.
- [7] E. Amir, S. McCanne, and R. Katz, "An active service framework and its application to real-time multimedia transcoding," in *Proceedings of ACM SIGCOMM '98*, Vancouver, BC, Canada, Aug. 1998, pp. 178–189.
- [8] M. Hemy, P. Steenkiste, and T. Gross, "Evaluation of adaptive filtering of MPEG system streams in IP networks," in *Proceedings of the IEEE International Conference on Multimedia and Expo 2000 (IDME 2000)*, New York, NY, Aug. 2000, pp. 1313–1317.
- [9] N. Yeadon, F. Garcia, D. Hutchison, and D. Shepherd, "Continuous media filters for heterogeneous Internetworking," in *Proceedings of SPIE - Multimedia Computing and Networking (MMCN96)*, San Jose, CA, Jan. 1996, pp. 118–134.
- [10] S. Shi and J. Turner, "Routing in overlay networks," in *Proceedings of IEEE Infocom 2002*, vol. 3, New York, NY, June 2002, pp. 1200–1208.
- [11] P. Francis, "Yallcast: extending the Internet multicast architecture," NTT Information Sharing Platform Laboratories, Tech. Rep., Sept. 1999.
- [12] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, "Overcast: reliable multicasting with an overlay network," in *Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI 2000)*, San Diego, CA, Oct. 2000, pp. 197–212.
- [13] K. Lee, S. Ha, J. Li, and V. Bharghavan, "An application-level multicast architecture for multimedia communications," in *Proceedings of ACM Multimedia 2000*, Los Angeles, CA, Oct. 2000, pp. 398–400.
- [14] I. Stoica, T. Ng, and H. Zhang, "REUNITE: A recursive unicast approach to multicast," in *Proceedings of IEEE Infocom 2000*, Tel Aviv, Isr, Mar. 2000, pp. 1644–1653.
- [15] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, Aug. 2002.
- [16] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet Indirection Infrastructure," in *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, Aug. 2002.
- [17] R. Karrer and T. Gross, "Communication in future applications: where should the intelligence go?" ETH Zurich, Tech. Rep., June 2002.
- [18] I. Stoica, D. Adkins, S. Ratnamsamy, S. Zhuang, S. Shenker, and S. Surana, "Internet Indirection Infrastructure," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Cambridge, MA, Mar. 2002.