Alternative Transmission Strategies for Multipath Transport of Multimedia Streams over Wireless Networks

Martin Becke*, Thomas Dreibholz§, Andreas Bayer*, Markus Packeiser*, Erwin Paul Rathgeb*

*University of Duisburg-Essen, Institute for Experimental Mathematics
Ellernstraße 29, 45326 Essen, Germany
{martin.becke,erwin.rathgeb}@iem.uni-due.de, {andreas.bayer, markus.packeiser}@uni-due.de

§Simula Research Laboratory, Network Systems Group
Martin Linges vei 17, 1364 Fornebu, Norway
dreibh@simula.no

***Abstract*—With the strongly growing popularity of mobile devices like smartphones and tablet computers, the number of end-systems with more than one network access – like UMTS/LTE and WLAN – is also increasing. This so-called multi-homing also leads to the desire of utilising multiple network paths simultaneously, in order to improve application payload throughput. Clearly, this so-called multi-path transfer feature is also very useful for the transport of multimedia contents, particularly when a single network access alone is not fast enough to fulfil the bandwidth requirements of the application.**

In many cases, multimedia transport is also sensitive for delays and packet losses. However, the focus of the current multi-path transfer approaches has been on bandwidth only. In order to tackle this challenge, our paper introduces two new send strategies to map payload data to different wireless paths. Finally, by using measurements, we show that a significant performance improvement for delay and loss-sensitive applications can be achieved in comparison to the existing approaches.[1][2][3]

Keywords: Multi-Path Transfer, Multimedia Transport, Scheduling Strategies, Real-Time, Wireless

I. Introduction and Related Work

Classic TCP-based communication is based on the assumption that end-systems are accessing the Internet with only a single network interface. However, with the growing success of e.g. smartphones and tablet computers, multi-homed devices with more than one network interface – like 2G/3G/4G, WLAN and Ethernet – become increasingly popular. While the SCTP [1] protocol just utilises this property for redundancy purposes out of the box, there is a significant demand to use multiple network accesses *simultaneously* in order to improve the application payload throughput.

For instance, there is very active work on the Concurrent Multipath Transfer for SCTP (CMT-SCTP) [2]–[4] as well as the Multi-Path TCP (MPTCP) [5], [6] for TCP protocol extensions for this so-called multi-path transfer purpose. Currently, the focus of research on multi-path transfer performance is on scenarios with dissimilar path setups [2], [7]. That is, the used paths have different quality of service (QoS) characteristics like bandwidths, delays and loss rates – which is clearly the usual case for Internet setups. Particularly, such dissimilar paths also affect the multi-path congestion control behaviour [8] when applying path-coupled strategies [9], [10]. While throughput is the main performance goal of many Internet applications, interactive multimedia services are also sensitive to delays and packet losses.

Furthermore, multi-path transfer can also be used to improve the resilience in case of possible path failures, by redundantly retransmitting the affected data on other paths in order to reduce packet losses [11]. This works fine if the complete traffic could be shifted to alternative paths. Otherwise, the scheduler could filter for important data, e.g. special video compression picture types like I-frames [12]. The organisation of the send behaviour is also part of current research: [13] shows that energy-efficient interface scheduling using MPTCP could provide a way to minimise energy consumption. Furthermore, [14] presents an optimised scheduling approach for transporting multiple logical streams with multi-path transfer.

Particularly, there is also interest [15] in applying multi-path transfer for media streaming applications, where the bandwidth of a single path alone would only lead to a poor user perception of the presented contents. However, with the current scheduling strategies for SCTP [2] and MPTCP [5], the dissimilarity of path delays will have a negative impact on latency-sensitive applications, as we will show in this paper. Therefore, we propose two approaches to tackle this issue.

This paper is structured as follows: first, we introduce the existing approaches for multi-path transfer and describe the currently-used sending strategies. This is followed by the description of our alternative approaches. By using an experimental testbed setup, we furthermore evaluate these approaches and compare their performance to the standard behaviour. Finally, we conclude our work and provide an overview on future goals.

[1]Parts of this work have been funded by the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG).

[2]The authors would like to thank Michael Tüxen for his friendly support.

[3]The authors would like to thank the anonymous reviewers for their helpful comments.

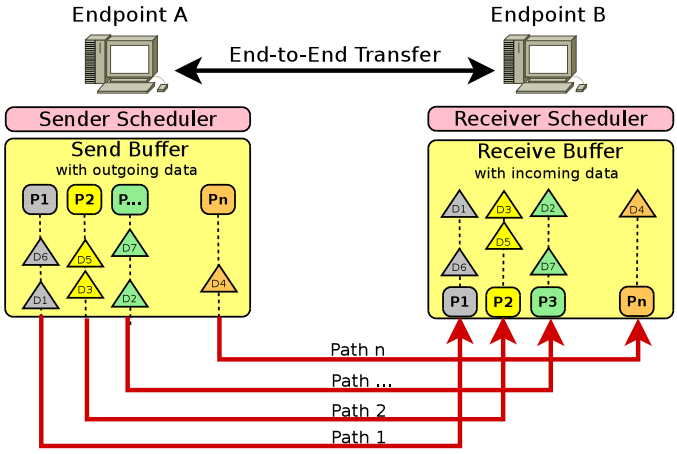


Fig. 1. Retransmission Latency Issue for Delay-Sensitive Data

II. Basics

In order to understand our approaches, it is important to understand how the sender currently schedules messages. The sender-side scheduler decides about the order in which messages are sent over which path. On the other hand, the receiver-side scheduler is organising the order in which the data is passed to the application. For example, in case of CMT-SCTP [3], it depends on the per-message configuration of in-sequence delivery and reliable transport. However, the CMT-SCTP and MPTCP specifications do not specify *how* these schedulers actually work. For instance, CMT-SCTP in the FreeBSD kernel applies a mapping of the application payload segments to the paths for transmission by applying round-robin over the paths. That is, the decision of which message is transmitted on which path completely depends on the corresponding protocol implementation.

Clearly, the scheduling of an implementation is based on the assumption that the segments reach their destination in a timely manner. However, the delay of a path may increase – e.g. due to buffer bloat [16] – as depicted in Figure 1: Path #1 has a Round-Trip Time (RTT) of 10 ms and Path #2 an RTT of 1000 ms. Let the scheduler apply a simple round-robin scheduling of payload segments onto the two paths. In case of a reliable, in-sequence delivery, a lost segment on Path #2 would cause a significant delay by the need to detect its loss and perform a successful retransmission. That is, if the receive buffer is already filled with segments awaiting the missing segment for in-sequence delivery, or if the send buffer is occupied with segments awaiting a cumulative acknowledgement, the whole transmission – i.e. on *all* paths – gets stalled until a successful retransmission of the missing segment. Obviously, this implies a negative impact on delay-sensitive applications.

In case of utilising only a single network interface, i.e. without multi-path transfer, there is of course no reason to think about how the segments could be mapped to different paths. The paradigm change from just using a single path (e.g. like classic TCP) towards multi-path transport (e.g. by using CMT-SCTP or MPTCP) puts a new challenge on the design of transport protocols. As we will show in Section V, just coupling the paths and arbitrarily distributing segments among them will not work properly in certain situations. Instead, caution must be taken for a more appropriate scheduling. We have therefore developed two advanced scheduling approaches.

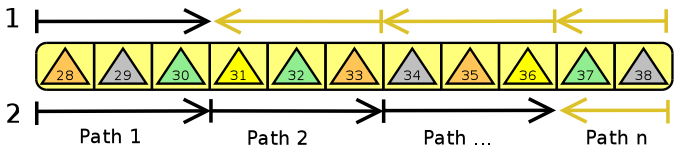


Fig. 2. The Principle of the Confluent Sequence Numbers Strategy

III. Alternative Transmission Strategies

A. Confluent Sequence Numbers

Our first alternative scheduling strategy is denoted as *Confluent Sequence Numbers* (ConSN). Figure 2 illustrates its principle: instead of round-robining the segments among the paths (as performed in current multi-path transfer implementations, see Section II), the segments of the send buffer are split up among the n paths. Two scenarios are possible:

1) Path #1 gets the segments in ascending order and the other $n-1$ paths in descending order, or
2) Paths #1 to #$n-1$ get the segments in ascending order and Path #n in descending order.

The key idea of our strategy is that the segment sequences of the paths will be confluent (hence the name Confluent Sequence Numbers). Now, if there is a problem – either a jump of the delay or a complete failure – on Path #j ($1 \leq j < n$), the transmitted sequence numbers of the other paths run towards the missing segments' sequence numbers. Therefore, a successful retransmission of the missing segments on a "good" path can easily solve a blockade of that path due to a lack of space in the send buffer or receive buffers (see also [2] for details on buffer handling).

To make that clearer, Figure 3 presents an example with only two paths. Segments #40 to #43 are transmitted on Path #1, while segments #44 to #49 are sent – in descending order – on Path #2. All segments have to be delivered to the remote application in sequence. A loss of Segment #43 (on Path #1) would block the Segments #44 to #49 (on Path #2) in the receive buffer, due to the need for in-sequence delivery. Then, a successful retransmission of Segment #43 on Path #2 allows to release the whole receive buffer space occupied for Path #2.

In case of problems on a path, round-robin scheduling would scatter missing segments over the whole sequence number range of the send buffer. ConSN, on the other hand, leads to consecutive blocks of missing segments. This allows for a space-efficient signaling of missing segments by the receiver (e.g. by providing the first missing segment's sequence number and the length of the block, instead of listing many scattered segment sequence numbers). Furthermore, having consecutive blocks instead of scattered missing segments reduces the buffer blocking issues described in [17]. Also, it can improve the handover performance in mobility scenarios by just having to retransmit some consecutive segment blocks in break-before-make scenarios.

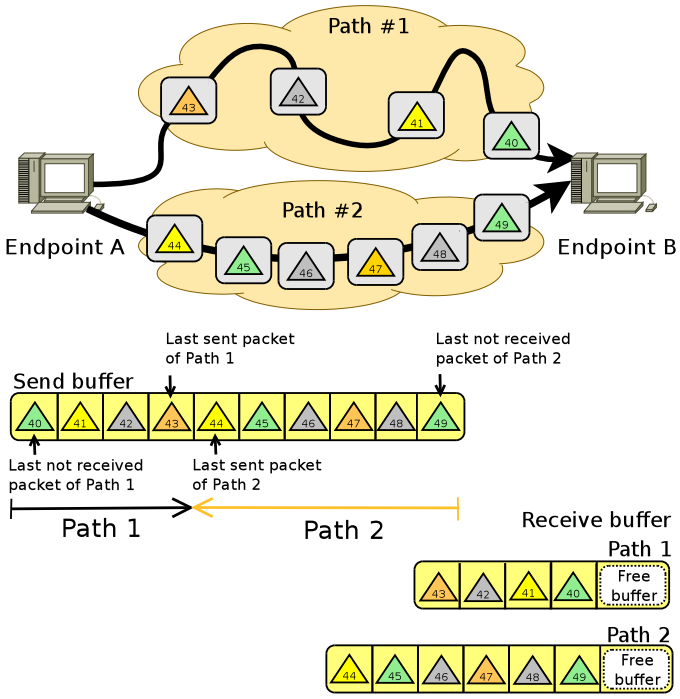


Fig. 3. An Example for the Usage of the Confluent Sequence Numbers

Furthermore, for delay-sensitive payload, it is possible to perform retransmissions preventively, in order to avoid buffer blocking. That is, given the example in Figure 3, Path #1 could continue with segments from #44 and higher after having sent segments #40 to #43. In this case, possible missing segments higher than #44 are recovered by Path #1. On the other hand, Path #2 could continue with segments from #43 and lower after having sent Segments #49 to #44. These overlapping – and therefore redundant – segment transmissions on both paths reduce the delay (by avoiding the need for retransmissions) at the cost of an increased bandwidth usage. A delay-sensitive heuristic may perform such redundant transmissions when segments seem to be late (and possibly lost). This can be seen as an extension of the chunk rescheduling mechanism introduced in [7] and [2], which triggers retransmissions on upcoming buffer blocking only. Note that preventive retransmissions are congestion-controlled like *new* segment transmissions [8], i.e. they do not introduce any unfairness to concurrent flows within the network.

B. Path Delay Compensation

In order to compensate significant delay differences among the paths, [18] has added another mechanism. It introduces a Min-Max approach which divides the flow on the paths such that the end-to-end delay for all the paths remains the same. While, of course, this mechanism introduces an additional delay for packets going over otherwise low-latency paths, the compensated path delays make the handling of multipath transfer easier, but also increases the overall transmission delay. To avoid this disadvantage, we introduce an alternative approach, which we denote as *Path Delay Compensation* (PDC). Here, for each path, the smoothed RTT (sRTT) is used to estimate its RTT and perform an educated guess on which packets should be sent over the path with the higher delay, in order to let these

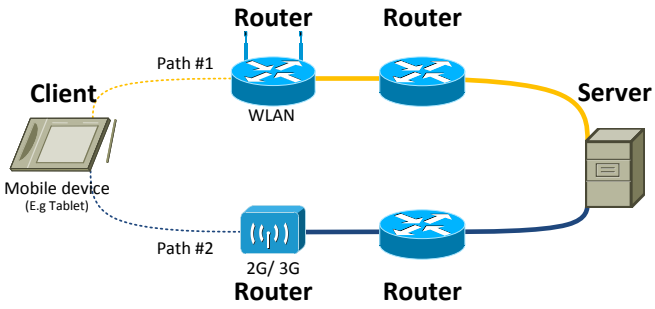


Fig. 4. The Testbed Setup

packets arrive at the right time. Of course, the RTT difference between the paths could be so high that the throughput of the higher-performing link is not sufficient to fill the exiting gap caused by the delay (as we will later demonstrate in Subsection V-C).

C. Split Error Correction

Bit errors due to interferences and noisy channels are a very common problem for wireless transmission. That is, even a single or very few bit errors in a packet lead to dropping a possibly full-sized packet. [19] proposes the idea of applying *Forward Error Correction* (FEC) for delay-sensitive video communications. Its goal is to improve the reliability of the video transmission by adding redundancy information to repair damaged data blocks. The proposed solution works on the Data Link Layer (WLAN in this case), which requires appropriate support by the underlying hardware. Our second scheduling strategy – which is denoted as *Split Error Correction* (SEC) – transfers the idea of FEC into the Transport Layer. The advantage of this higher-layer approach is the independence of the underlying protocol layers; i.e. the end-to-end principle makes a deployment easier. Furthermore, the sending endpoint – which has a more detailed knowledge of the different network paths – can make a better choice of distributing the original data (i.e. the multimedia stream) and the redundancy information onto the available paths. Note, that SEC requires the delivery of damaged packets to the receiver-side Transport Layer (similar to e.g. SCTP packet drop reporting [20]).

A quite simple FEC mechanism is introduced by [21]. This algorithm is able to correct one bit error in 120 bits with an overhead of 7 bits. It therefore produces an overhead of 88 bytes for a packet of 1,500 bytes (i.e. about 6%). This may help to save a full packet (and avoid a retransmission) of 12,000 bytes (i.e. 8*1,500). Our idea is to use a high-bandwidth path for the original multimedia stream and a possibly low-bandwidth one for the redundancy information. Clearly, more sophisticated FEC approaches like [22] may be used to better adapt a system to the specific error characteristics of the underlying channels. Nevertheless, the principle of SEC remains the same. In the first step, as a proof of concept, we focus on the splitting of the information itself, not on the optimisation of the FEC mechanisms.

Network	Payload Download	Payload Upload	Delay	Loss
3G HSDPA	4.0 Mbit/s	1.5 Mbit/s	160 ms ± 150	0 %
WLAN	25.0 Mbit/s	6.0 Mbit/s	31 ms ± 4	<1 %

TABLE I. THE BASELINE PERFORMANCE IN THE TESTBED

IV. TESTBED SETUP

In order to evaluate the two new scheduling strategies, we have set up a testbed environment as illustrated in Figure 4. The testbed is based on Linux-based routers as well as two FreeBSD-based endpoints. Two disjoint paths connect the endpoints via the routers. The first path has been configured with a 3G High-Speed Downlink Packet Access (HSDPA) connection, via the mobile Internet service provider E-PLUS. A WLAN connection has been used for the second path. A performance evaluation tool based on NETPERFMETER [23] has been applied for the performance evaluation. In order to obtain reproducible and comparable results, we have first obtained a baseline performance in the described configuration. After that, we have replaced the actual mobile communication by link emulation using NETEM [24]. Table I shows the results of the baseline measurement. Note, that the total payload bandwidth of both paths is 8.5 Mbit/s.

In order to demonstrate the effects of the alternative sending strategies, we have extended a simple control protocol with some basic mechanisms for acknowledgement, round trip time calculation, packet counting and bandwidth measurement. For sending, we have utilised our traffic generator to send a payload of 25,000 bytes in an interval of 25 ms. That is, we have sent data with an average payload data rate of 8 Mbit/s – which is e.g. realistic for a high-definition movie stream – in total, split up between the two paths. For the ConSN strategy, we schedule the data to the paths by using a simple weighted scheduling (weight $\hat{w}_P$ for path P) that is based on the calculation of the sRTT [25]:

$$\hat{w}_P = 1 - \frac{\text{sRTT}_P}{\sum_i \text{sRTT}_i}$$

$$\text{sRTT} = (\alpha \cdot \text{RTT}) + ((1-\alpha) \cdot \text{sRTT}),\ \alpha = 0.125$$

Note, that we use this simple strategy in order to focus on the ConSN strategy and not on the scheduling itself. Clearly, here is a potential for future work.

V. EXPERIMENTAL EVALUATION

A. Sufficient WLAN Bandwidth

In the first scenario, we have configured the WLAN link with a downstream bitrate of 25 Mbit/s and an upstream bitrate of 6 Mbit/s; the delay has been 30 ms and the loss rate has been less than 1%. The 3G HSDPA link has had a downstream bitrate of 4 Mbit/s and an upstream bitrate of 1.5 Mbit/s, with a delay of 160 ms and a loss rate of 0%. From time t=16 s to t=24 s, the delay has been increased to 280 ms in order to introduce a temporary transmission difficulty. Note, that in fact just using the WLAN link would have been sufficient for successfully transferring the 8 Mbit/s video stream here.

Figure 5 presents the percentage of unusable packets for this scenario, where unusable packets are all packets that

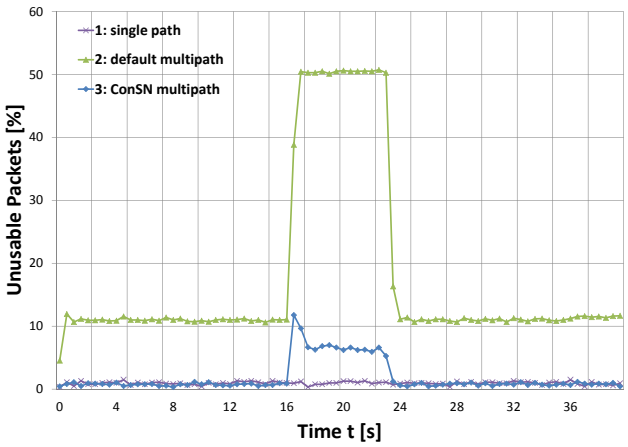


Fig. 5. Coupling WLAN and 3G Links (WLAN alone would be sufficient)

have actually been lost as well as the packets that had to be dropped because they have arrived too late (by more than 200 ms in this scenario, i.e. they have become useless for the real-time playback). By just using the WLAN path (i.e. a single path setup, in curve 1), the resulting percentage of unusable packets is almost 0%. However, by turning on multipath transfer, the percentage of unusable packets for default multipath scheduling (curve 2) increases to more than 10% for a delay of 160 ms, and even jumps to more than 50% when the 3G path delay is temporarily increased to 280 ms. That is, the application of multipath transfer with the default scheduling – which is intended for bulk data transfer, but not for delay-sensitive multimedia streaming – can lead to a significant performance loss in comparison to a single-path transmission. By using ConSN (curve 3), the performance is similar to the single-path performance for a 3G path delay of 160 ms. However, in case of the delay jump, the delay limit of 200 ms is exceeded. That is, in this case there are again losses. However, they are significantly smaller in comparison to the default scheduling (compare to curve 2).

Clearly, a heuristic could be introduced to avoid using a path when such a delay issue persists for a longer time period. Note, however, that it is not generally useful to completely turn off a path in advance. In wireless scenarios, transmission conditions vary frequently, due to interferences, handovers to other segments, different congestion conditions, etc.. Therefore, a high degree of adaptability to changing network behaviour is needed.

B. Insufficient Bandwidth for Single-Path Transfer

In the second scenario, we have reduced the WLAN downstream bitrate to 6 Mbit/s, i.e. the WLAN link alone has become insufficient for transferring the 8 Mbit/s video stream. Such a bandwidth reduction is a very usual case in situations of interferences or bad reception conditions. Figure 6 presents the corresponding percentage of unusable packets. Clearly, when using single-path transfer (curve 1), there is a percentage of unusable packets of more than 30%. Applying multipath transfer with default scheduling (curve 2) solves the bandwidth issue (since both paths together achieve a sufficient bandwidth), but there is still a percentage of unusable

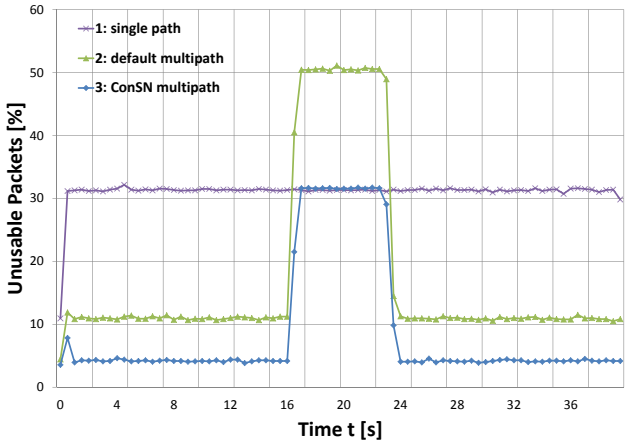


Fig. 6. Coupling WLAN and 3G Links (WLAN capacity too small)

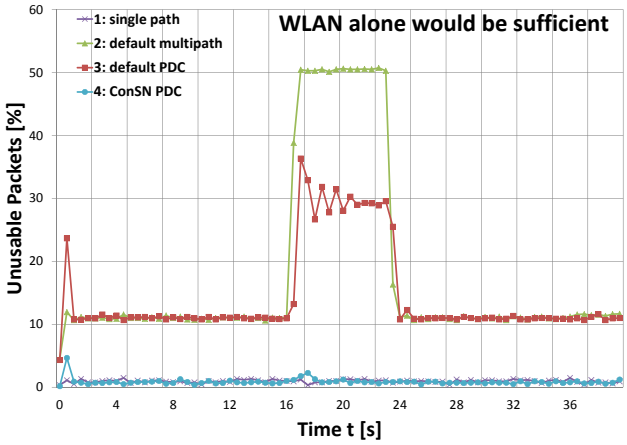
packets of more than 10%. This is caused by the packet delay, which results in too-late arrivals. ConSN (curve 3) is able to significantly reduce the percentage of unusable packets to less than 4%. Even in the case of a link delay increase from 160 ms to 280 ms on the 3G path (t=16 s to t=24 s), the loss rate does not become worse than for the single-path transfer (compare to curve 1).

Note, that – in this increased-delay case – the bandwidth of the 3G path would not be usable anyway, since the delay difference of this path (280 ms vs. 30 ms) violates the playback delay constraint of 200 ms. Here, the video transmission would only be possible by relaxing this playback delay constraint to a higher value (which, of course, would degrade the interactivity of the user).

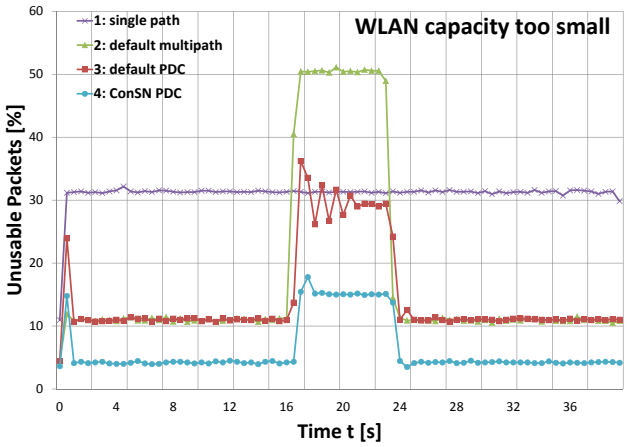
C. Applying Path Delay Compensation

In order to demonstrate the effects of PDC, Figure 7 presents the corresponding results in the previous two scenarios, i.e. sufficient WLAN bandwidth (as described in Subsection V-A) in Subfigure 7(a) and WLAN capacity alone being too small (as in Subsection V-B) in Subfigure 7(b). For comparison, curve 1 shows the unusable packets percentage rate for a single-path transmission, while curve 2 presents the unusable packets percentage rate for a standard multipath scheduling.

Clearly, activating PDC alone – i.e. without ConSN – as shown by curve 3 does not help much. The unusable packets percentage rate remains at more than 10% in both scenarios, and it even jumps to more than 30% when the delay on the 3G path is temporarily increased. However, by using PDC in combination with ConSN, a significant improvement can be achieved. In the first scenario, with sufficient WLAN bandwidth available, the same performance as for a single-path transfer (compare to curve 1) is reached. Also, in comparison to ConSN alone (see curve 3 in Figure 5), the delay jump of the 3G path does not affect the unusable packets percentage rate any more. A similar observation can also be made for the second scenario, where WLAN bandwidth alone is insufficient. PDC in combination with ConSN significantly reduces the percentage of unusable packets in comparison to PDC alone



(a) WLAN alone would be sufficient



(b) WLAN capacity is too small

Fig. 7. Coupling with Path Delay Compensation

(curve 3). Also, in comparison to ConSN alone (see curve 3 in Figure 6), it about halves the unusable packets percentage rate.

Particularly, PDC in all cases reaches at least the performance of the case without PDC. That is, the recommendation is to always apply PDC with ConSN. In order to further reduce the loss rate in cases where the path delay differences are too large (here: from time t=16 s to t=24 s), future work could combine PDC with the idea from [18] to apply additional buffering on the faster paths in order to achieve more similar path delays. However, instead of buffering all paths to the delay of the highest-latency path, the application of PDC may lead to a relaxed need for additional buffering (and therefore result in a reduced overall message delay).

D. Using Split Error Correction

In the first scenario (see Subsection V-A), we have shown that in certain cases it may be useful to just use a single path. Then, it is possible to utilise the second path for redundancy information by applying SEC as introduced in Subsection III-C. Therefore, we have taken the scenario settings from Subsection V-A, but using the 3G path for SEC only and keeping its

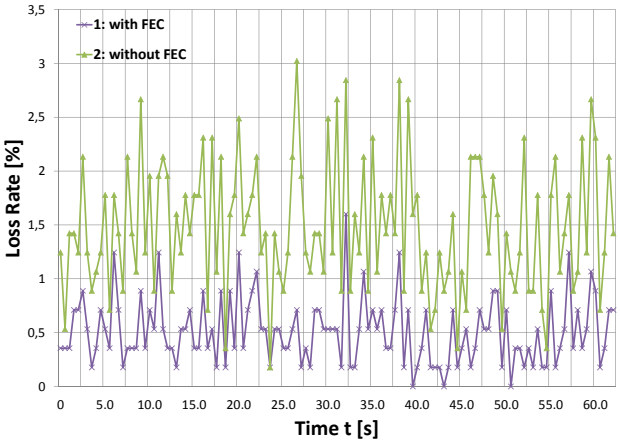


Fig. 8. Using the Small-Bandwidth Path for Split Error Correction

delay of 160 ms fixed. Figure 8 presents the resulting results for SEC turned on (curve 1) and off (curve 2). Bit errors on the WLAN path are introduced as follows: with a probability of $\frac{1}{30}$, a packet is affected by bit errors. In this case, the error probability of each 15-byte block in the packet is $\frac{1}{20}$. 90% of the errors are 1-bit errors; 10% are 2-bit errors. Note, that our error model is quite simple, since it is only intended to demonstrate the SEC capabilities and not to accurately model particular WLAN channel characteristics. As it is shown, the application of SEC significantly decreases the loss rate. That is, a second path – even if it is not actually necessary to achieve a certain bandwidth requirement, and particularly if its bandwidth is actually not very high – can be utilised in a useful way to decrease losses due to bit errors on the main, high-bandwidth path.

VI. Conclusion

Multipath transfer is becoming increasingly interesting, due to the quickly growing number of multi-homed devices. Particularly, it would also be useful for multimedia streaming applications, where the bandwidth of a single path alone may not be sufficient. However, the standard procedures for data scheduling among the paths are not useful for delay-sensitive traffic. Therefore, in this paper, we have introduced two alternative scheduling approaches and have shown their usefulness in a proof-of-concept evaluation.

As part of future work, it is clearly necessary to examine these mechanisms in more complex setups. A main goal is to evaluate each mechanism separately and define a policy-based approach for mechanism combination. We intend to perform parameter studies with a broad range of system configurations in reality (by using the FreeBSD kernel CMT-SCTP implementation, based on the NorNet multi-homed Internet testbed infrastructure [26]–[28]) as well as in simulations (by using the OMNeT++-based CMT-SCTP simulation model [2]). Furthermore, we are going to contribute our results – from research to application – into the ongoing IETF standardisation process of the multipath transport protocol extensions CMT-SCTP as well as MPTCP.

References

[1] R. R. Stewart, "Stream Control Transmission Protocol," IETF, Standards Track RFC 4960, Sep. 2007, ISSN 2070-1721.

[2] T. Dreibholz, "Evaluation and Optimisation of Multi-Path Transport using the Stream Control Transmission Protocol," Habilitation Treatise, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems, Mar. 2012.

[3] P. D. Amer, M. Becke, T. Dreibholz, N. Ekiz, J. R. Iyengar, P. Natarajan, R. R. Stewart, and M. Tüxen, "Load Sharing for the Stream Control Transmission Protocol (SCTP)," IETF, Network Working Group, Internet Draft Version 06, Mar. 2013, draft-tuexen-tsvwg-sctp-multipath-06.txt, work in progress.

[4] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-End Paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, Oct. 2006, ISSN 1063-6692.

[5] C. Raiciu, S. Barré, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving Datacenter Performance and Robustness with Multipath TCP," in *Proceedings of the ACM SIGCOMM*, Toronto/-Canada, Aug. 2011.

[6] A. Ford, C. Raiciu, M. Handley, S. Barré, and J. R. Iyengar, "Architectural Guidelines for Multipath TCP Development," IETF, Informational RFC 6182, Mar. 2011, ISSN 2070-1721.

[7] T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tüxen, "On the Use of Concurrent Multipath Transfer over Asymmetric Paths," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Miami, Florida/U.S.A., Dec. 2010, ISBN 978-1-4244-5637-6.

[8] M. Becke, T. Dreibholz, H. Adhari, and E. P. Rathgeb, "On the Fairness of Transport Protocols in a Multi-Path Environment," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Ottawa, Ontario/Canada, Jun. 2012, pp. 2666–2672.

[9] C. Raiciu, D. Wischik, and M. Handley, "Practical Congestion Control for Multipath Transport Protocols," University College London, London/United Kingdom, Tech. Rep., 2009.

[10] T. Dreibholz, H. Adhari, M. Becke, and E. P. Rathgeb, "Simulation and Experimental Evaluation of Multipath Congestion Control Strategies," in *Proceedings of the 2nd International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, Fukuoka/Japan, Mar. 2012, ISBN 978-0-7695-4652-0.

[11] M. Subramaniam and D. Manjula, "Effective Transport using Concurrent Multipath Transfer by Redundant Transmission during Path Failure," *European Journal of Scientific Research*, vol. 58, no. 2, pp. 247–256, Aug. 2011, ISSN 1450-216X.

[12] S. Srinivasan, J. Vahabzadeh-Hagh, and M. Reisslein, "The Effects of Priority Levels and Buffering on the Statistical Multiplexing of Single-Layer H.264/AVC and SVC Encoded Video Streams," *IEEE Transactions on Broadcasting*, vol. 56, no. 3, pp. 281–287, Sep. 2010, ISSN 0018-9316.

[13] C. Pluntke, L. Eggert, and N. Kiukkonen, "Saving Mobile Device Energy with Multipath TCP," in *Proceedings of the 6th ACM International Workshop on MobiArch*, Bethesda, Maryland/U.S.A., Jun. 2011, pp. 1–6, ISBN 978-1-4503-0740-6.

[14] T. Dreibholz, R. Seggelmann, M. Tüxen, and E. P. Rathgeb, "Transmission Scheduling Optimizations for Concurrent Multipath Transfer," in *Proceedings of the 8th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT)*, vol. 8, Lancaster, Pennsylvania/U.S.A., Nov. 2010, ISSN 2074-5168.

[15] T. Zinner, K. Tutschku, A. Nakao, and P. Tran-Gia, "Using Concurrent Multipath Transmission for Transport Virtualization: Analyzing Path Selection," in *Proceedings of the 22nd International Teletraffic Congress (ITC)*, Amsterdam, Noord-Holland/Netherlands, Sep. 2010, pp. 348–349, ISBN 978-1-4244-8837-7.

[16] J. Gettys, "Bufferbloat – Dark Buffers in the Internet," Jan. 2011.

[17] H. Adhari, T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tüxen, "Evaluation of Concurrent Multipath Transfer over Dissimilar Paths," in *Proceedings of the 1st International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, Singapore, Mar. 2011, pp. 708–714, ISBN 978-0-7695-4338-3.

[18] S. Shailendra, R. Bhattacharjee, and S. K. Bose, "Optimized Flow Division Modeling for Multi-Path Transport," in *Proceedings of the*

IEEE International Conference on Communication Systems (ICCS), Kolkata/India, Nov. 2010, pp. 1–4, ISBN 978-1-4244-9072-1.

[19] M.-F. Tsai, N. Chilamkurti, J. H. Park, and C.-K. Shieh, "Concurrent Multipath Transmission with Adaptive Forward Error Correction Mechanism for Delay-Sensitive Videostreaming in Wireless Home Networks," in *Proceedings of the 5th IEEE International Conference on Future Information Technology (FutureTech)*, Busan/South Korea, May 2010, pp. 1–6, ISBN 978-1-4244-6948-2.

[20] I. Rüngeler, "SCTP – Evaluating, Improving and Extending the Protocol for Broader Deployment," Ph.D. dissertation, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems, Dec. 2009.

[21] C. Shi-yi and L. Yu-bai, "Error Correcting Cyclic Redundancy Checks based on Confidence Declaration," in *Proceedings of the 6th IEEE International Conference on Telecommunications Proceedings (ITS)*, Chengdu, Sichuan/People's Republic of China, Jun. 2006, pp. 511–514, ISBN 0-7803-9586-7.

[22] A. Nafaa, T. Taleb, and L. Murphy, "Forward Error Correction Strategies for Media Streaming over Wireless Networks," *IEEE Communications Magazine*, vol. 46, no. 1, pp. 72–79, Jan. 2008, ISSN 0163-6804.

[23] T. Dreibholz, M. Becke, H. Adhari, and E. P. Rathgeb, "Evaluation of A New Multipath Congestion Control Scheme using the NetPerfMeter Tool-Chain," in *Proceedings of the 19th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Hvar/Croatia, Sep. 2011, pp. 1–6, ISBN 978-953-290-027-9.

[24] S. Hemminger, "Network Emulation with NetEm," in *Proceedings of the Linux Conference Australia (LCA)*, Canberra/Australia, Apr. 2005.

[25] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," IETF, Standards Track RFC 2988, Nov. 2000, ISSN 2070-1721.

[26] T. Dreibholz and E. G. Gran, "Design and Implementation of the NorNet Core Research Testbed for Multi-Homed Systems," in *Proceedings of the 3nd International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, Barcelona, Catalonia/Spain, Mar. 2013, pp. 1094–1100, ISBN 978-0-7695-4952-1.

[27] T. Dreibholz, "The NorNet Project – A Research Testbed for Multi-Homed Systems," Invited Talk, Karlstad, Värmland/Sweden, Nov. 2012.

[28] ——, "The NorNet Project – An Introduction to NorNet for the Site Deployment at NTNU Trondheim," Invited Talk, Trondheim, Sør-Trøndelag/Norway, Apr. 2013.