# An Analysis of the Heterogeneity and IP Packet Reordering over Multiple Wireless Networks

Dominik Kaspar*, Kristian Evensen*, Audun F. Hansen*, Paal Engelstad†*, Pål Halvorsen*‡, Carsten Griwodz*‡
*Simula Research Laboratory, Norway †Telenor R&I, Norway ‡University of Oslo, Norway
email:{kaspar, kristrev, audunh, paalee, paalh, griff}@simula.no

***Abstract*—With the increasing deployment of wireless technologies, such as WLAN, HSDPA, and WiMAX, it is often the case that simultaneous coverage of several access networks is available to a single user device. In addition, devices are also often equipped with multiple network interfaces. Thus, if we can exploit all available network interfaces at the same time, we can obtain advantages like the aggregation of bandwidth and increased fault tolerance. However, the heterogeneity and dynamics of the links also introduce challenges. Due to different link delays, sending packets of the same flow over multiple heterogeneous paths causes the reordering of packets.**

**In this paper, we quantify the impact of network heterogeneity and the use of multiple links on IP packet reordering. We show with practical measurements, according to commonly used metrics, that packet reordering over multiple links exceeds the reordering caused by common connections in high-speed, wide-area networks. We also demonstrate that heterogeneity and reordering exceed the assumptions presented in related work.**

**By using sufficiently large buffers, packet reordering can be avoided. However, for devices with high resource constraints, the workload of using large buffers is expensive. Sender-side solutions of dividing and scheduling a packet sequence over multiple links can reduce the buffer requirements at the receiver. Initial experiments with a static scheduler, that has knowledge of average link delay and throughput estimates, show that packet reordering can be reduced by only 38 % due to the dynamic heterogeneity of the two links.**

## I. Introduction

A growing infrastructure of heterogeneous wireless technologies, such as WLAN, HSDPA, and WiMAX, often places a single user device into the coverage areas of multiple access networks simultaneously. Currently, from an application's point of view, Internet connectivity is usually provided using a single link at each point in time. It is, however, increasingly common that devices such as laptops, phones, and PDAs are equipped with multiple wireless interfaces. Motivated by these trends, wireless providers are looking for solutions that can fully utilize multiple technologies concurrently when present.

Exploiting multiple links simultaneously has several benefits: increased throughput by bandwidth aggregation, added fault tolerance by sending redundant data over independent links, and increased mobility and connectivity by overlapping several coverage areas [1]. However, the deployment of a readily usable multilink solution has so far been hindered by substantial network heterogeneity. High variances in the characteristics of wireless links cause packets to be severely reordered, which greatly impairs the performance of transport layer congestion control mechanisms.

Due to the currently existing challenges, end users do not yet have easy access to exploiting multiple networks at once. However, with modifications of an operating system's routing tables, it is possible to achieve a multilink configuration, which is able to achieve aggregated throughput, provided that an application is used that opens many concurrent connections.

A BitTorrent client is a perfect example of such an application, because the underlying protocol logically splits a large file into segments, which are downloaded independently and concurrently from various locations. A new connection is opened to each provider of a data segment, and using a multilink setup, this can be exploited without any changes to the protocol. After defining an equal-cost multipath route (ECMP) [2] for each available interface, each new connection will automatically be opened over one of the interfaces. For a large number of connections, traffic will be equally balanced according to the bandwidths of the links. Figure 1 illustrates the benefit of a multilink setup using an HSDPA and a WLAN link aggregated through the use of ECMP.

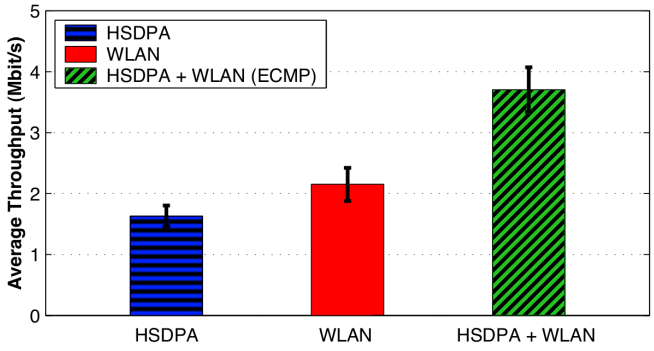


Fig. 1. Throughput aggregation with BitTorrent over simultaneous HSDPA and WLAN links (results were obtained by downloading a total of 75 copies of a 700 MB large file from a popular source).

These results show the potential of utilizing multiple links simultaneously. However, building this functionality per application is not a viable approach. From an Internet service provider's (ISP's) point of view, we envision the users to access any type of services on the Internet using multiple wireless links. For such an approach to be deployed, we think that a solution must be able to split a single transport connection over the alternative links. This can be solved by a proxy or server implementation. The latter requires that, for instance, the ISP controls the server.

Real-time and streaming applications (e.g., video conferencing) serve as other examples of why simply dividing the traffic over multiple links, using a vast number of connections, is not a viable approach. The packets in such streams should be delivered in the order of application data playout, not risking a long pause due to a delayed bulk of packets on a slow link.

The problem of packet reordering may be mitigated by using buffers that hold at least as much data as is "in flight" at all times over all interfaces. However, for low-performance and battery-operated devices, such as cell phones and PDAs, large buffers are very resource-intensive. Keeping in mind the goal of reducing the workload of such devices, this paper studies the fundamental characteristics of packet reordering when using multiple links. We also evaluate the potential gain of a static, sender-side packet scheduler that tries to minimize reordering at the client by sending packets ahead of time over links with high delay.

This paper's main contribution is to shed some light into the effects of splitting a packet sequence over multiple wireless interfaces and how the dynamic heterogeneity of these links cause reordering at the destination. In Section II, currently proposed approaches to data transfer over multiple links are presented. Based on measurements of the heterogeneity of wireless links, Section III then indicates the dynamics of wireless links and points out the challenges with link aggregation. In Section IV, metrics for the quantification of packet reordering are introduced, before Section V experimentally examines the effects of using multiple links on IP packet lateness. Methods of reducing packet reordering are discussed in Section VI, and the conclusions of this work are finally drawn in Section VII.

## II. Related Work

For several years, contributions to multilink transfer have been proposed on many layers of the protocol stack, ranging from network level multihoming [3], over multipath transport for homogeneous and independent wired paths [4], to methods for application layer striping [5]. A good overview of recent solutions to transport layer striping is given in [6].

Most of the related literature focuses on multipath routing in wired, high-speed networks. In addition, the majority of existing solutions to network striping assume substantial protocol and end-host modifications, which may hinder general deployment. The fact that most of them have only been tested in simulations, often based on very simple assumptions about heterogeneity, enforces our skepticism on deployability [7], [8]. Our field measurement results (see Section III) contradict frequently made assumptions about the characteristics of heterogeneous links, which are often modeled too evenly and not very realistically (e.g., 30 milliseconds delay for both WLAN and UMTS [7]).

The effects of multipath routing on packet reordering have been addressed in various publications ranging from simulation-based analyses [9], over real-time measurements in wide-area networks [10], [11], [12], [13], to IETF standardization efforts that define metrics for measuring packet reordering [14], [15]. However, no publications have come to our attention that address how severely packets get disordered when using multiple interfaces.

## III. Measurements on Link Heterogeneity

Having access to the Internet over multiple links at the same time has many potential benefits, but it also introduces new difficulties. The main challenge addressed in this work is that splitting a packet sequence over heterogeneous IP paths will most likely cause the sequence to be out of order at the destination. Especially in wireless networks, the amount of reordering is heavily dependent on a large number of parameters that are beyond the user's control, such as unpredictable interference and varying signal quality, other users contending on the same channel, overloaded routers, etc.

Link heterogeneity has several aspects, of which the transmission delay is the most crucial factor for causing packets split over different paths to arrive out of sequence. Table I shows the round-trip delay of ping packets measured through an entire day over different wireless technologies. From these numbers, we expect that packet reordering over multiple links will be far beyond the 0.01 % to 1.65 % level [11] stated in the literature.

TABLE I
Round-trip time measurements over HSDPA and WLAN

| | min. | max. | mean | stdev. | loss |
|---|---|---|---|---|---|
| WLAN1 | 2.2 ms | 5010 ms | 10.1 ms | 61.2 ms | 2.7 % |
| WLAN2 | 3.2 ms | 1807 ms | 11.2 ms | 59.0 ms | 1.5 % |
| HSDPA | 67.5 ms | 1350 ms | 212 ms | 106 ms | 0.8 % |

A solution to multilink transfer must also consider that link characteristics may vary significantly over time. An example of variability over time is illustrated in Fig. 2, which shows the throughput dynamics of our HSDPA link.

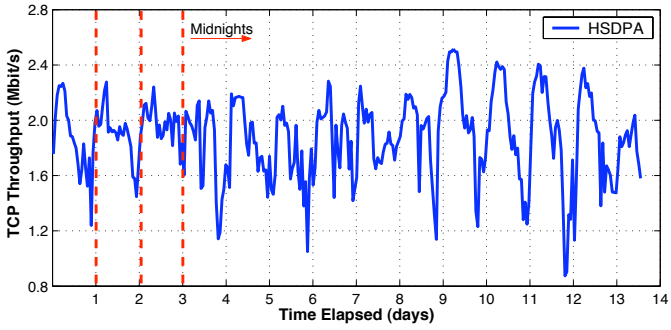


Fig. 2. HSDPA throughput varying over a period of 14 days. The results were obtained by repeatedly downloading a 5 MB large file over HTTP.

At present, no complete and readily deployable multilink solution exists due to implementation challenges and substantial network heterogeneity. It is a great challenge to deal with transport layer packet reordering and to optimally schedule packets (or sessions) to available links that vary significantly in their characteristics.

## IV. Packet Reordering

The Internet Protocol has no mechanisms to ensure that packets arrive at a destination in the same sequence that they were sent out at the source. There are various reasons for the occurrence of IP packet reordering; traditional causes include link layer retransmissions and priority scheduling in routers. When dividing IP traffic over heterogeneous paths, however, much higher reordering can be expected. This section describes our testbed for conducting field measurements for packet reordering and presents the metrics used in the evaluations.

### A. Testbed Setup

For measuring the effects of packet reordering, a dedicated server with a single 100 Mbit/s Ethernet network interface was used. As illustrated in Figure 3, a client laptop that is connected to both a WLAN and an HSDPA access network receives a packet sequence from the server. In the initial connection request, the client informs the server about its available interface addresses and about all other parameters used for the intended test run (i.e., bitrate, duration, packet size, type of scheduling, etc.). When the server has received all necessary information, it sends UDP packets at the specified constant bitrate to the client addresses. A scheduler at the server decides for each packet to which client address it will be sent. If not otherwise specified, round-robin scheduling is used. A static scheduler based on predetermined average link delays will be discussed in Section VI-B.

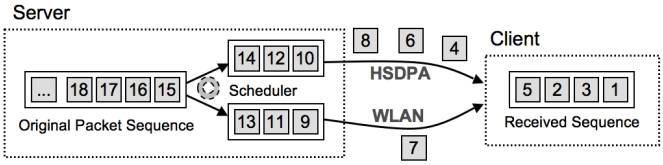


Fig. 3. Example of the experimental testbed, in which a round-robin scheduler is used for splitting a UDP packet sequence over two heterogeneous paths to a multilinked client.

This testbed was used in all the experiments described in the following sections. The multilinked laptop is stationary and it has access to two WLAN networks $N_{WLAN_1}$ and $N_{WLAN_2}$, and an HSDPA network $N_{HSDPA}$.

### B. Packet Reorder Metrics

Various metrics have been defined for getting a notion of the amount of reordering that occurs in a packet sequence. In RFC 4737 [14], several metrics for packet reordering are defined, each one focusing on certain application-specific properties. For instance, packet reordering can be expressed as the number of packets that arrived late (i.e., a packet is late, if another packet with larger sequence number had previously arrived). For expressing the percentage of packets that arrived late, we will use the *Type-P-Reordered-Ratio-Stream* metric defined in RFC 4737 and denote the metric as $M_{TPRRS}$.

If reordering occurs to only a very small fraction of all packets, $M_{TPRRS}$ is an intuitive metric for getting a sense of the reordering. Most of the current research discussing packet reordering is focused on single-path flows through wired, high-speed networks [12]. The common portion of packets affected by reordering is in the order of 0.01 % to 1.65 % [11], or 1 % to 1.5 % [10].

However, $M_{TPRRS}$ does not contain any information about the severity of disorder in a sequence. In an extreme case of receiving a packet sequence, such as $S = \{2, 1, 4, 3, 6, 5, ...\}$, where pairs of sequence numbers are "flipped", $M_{TPRRS}$ would be equal to 0.5, indicating that 50 % of all packets arrived late. In our testbed, where two links induce different delays, $M_{TPRRS}$ is therefore naturally expected to approach 0.5. Thus, for the case of splitting a stream over heterogeneous links, a more sophisticated metric that is able to convey the total magnitude of disorder in the received sequence would be valuable.

One such metric, called *Reorder Density*, is introduced in [13] and RFC 5236 [15]. To compute the Reorder Density $M_{RD}$, the sequence number of each packet is compared to the position in the sequence received at the destination. For each packet, this allows the calculation of a displacement within the sequence, negative values indicating "early" packets and positive displacements indicating "late" packets. $M_{RD}$ is calculated by counting the number of packets with a specific displacement. Figure 4 shows a measured example of $M_{RD}$, obtained by sending packets over $N_{WLAN_1}$ and $N_{HSDPA}$ at 2.0 Mbit/s. The reason why the depicted $M_{RD}$ histogram is not symmetric is that 2.1 % packets were lost over the WLAN link, while the HSDPA link suffered no loss.

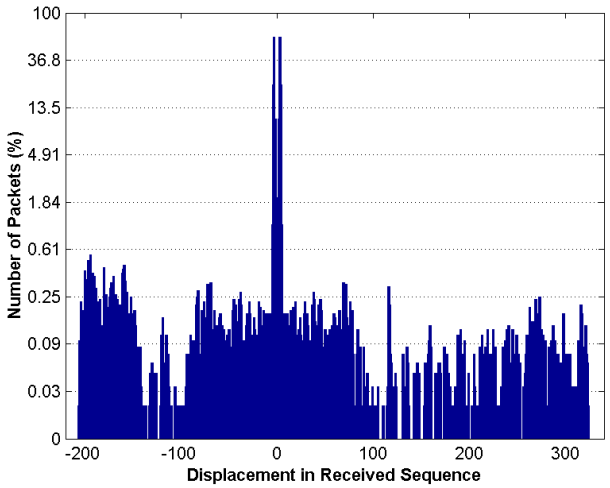


Fig. 4. An example Reorder Density, obtained at a bitrate of 2.0 Mbit/s, splitting packets in a round-robin fashion over an HSDPA and WLAN link. *For illustration purposes, the y-axis is in logarithmic scale.*

Although $M_{RD}$ provides an exact summary of the reordering in a packet sequence, it is less intuitive than $M_{TPRRS}$ and not suitable for comparing the disorder of two independently measured sequences. A density function does not give a clear indication of the system (e.g., scheduling solutions) that performs the best. Thus, a singleton metric, called *Reorder*

*Entropy*, has been formulated in [12]. The metric $M_{RE}$ expresses the total disorder of a packet sequence, including both the fraction of displaced packets as well as the degree to which the packets are displaced. The total Reorder Entropy $M_{RE}$ is directly derived from $M_{RD}$ and defined as:

$$M_{RE} = -\sum_{i \in D} M_{RD}(i) * ln(M_{RD}(i)) \quad (1)$$

The set $D$ contains all existing displacements in $M_{RD}$. It will be shown how $M_{RE}$ grows with increasing bitrate.

## V. Analysis of Packet Lateness

This section presents results obtained using the testbed mentioned in Section IV-A and the packet reorder metrics introduced in Section IV-B.

Figure 5 depicts the ratio of late packets according to the metric $M_{TPRRS}$ in two sets of measurements. The two curves were obtained by scheduling packets over $N_{HSDPA}$ and both of the two available WLAN networks. In this experiment, the average delay over $N_{HSDPA}$ was 80 ms, the average delay over $N_{WLAN_1}$ was 42 ms, and for $N_{WLAN_2}$, it was 44 ms.

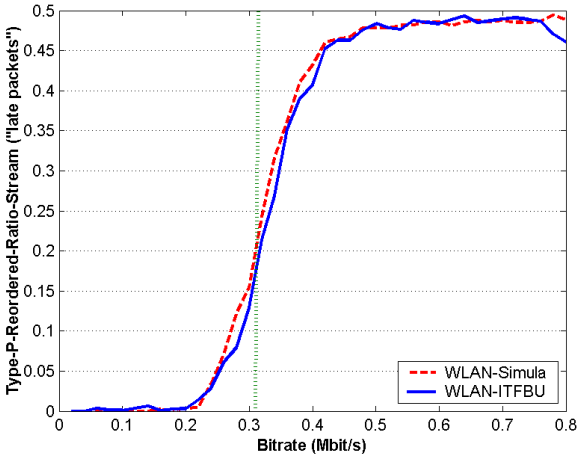


Fig. 5. The metric *Type-P-Reordered-Ratio-Stream* expresses the percentage of packets in a sequence that arrives late. For different WLAN networks, the ratio of late packets is alike.

The vertical line located at roughly 0.32 Mbit/s indicates the critical bitrate $B_C$ at which packets of the WLAN link start to "overtake" packets sent over HSDPA, therefore causing the latter to arrive late. Ideally, when each packet travels with a constant delay, $M_{TPRRS}$ would jump from 0 % to 50 % when the bitrate surpasses $B_C$. If $d_1$ and $d_2$ denote the average transmission delays over two different access networks and $p$ designates the packet size in bits, then the critical bitrate $B_C$, at which late packets are expected, is defined as:

$$B_C = p/(|d_1 - d_2|) \quad (2)$$

In the experiment described above, the average difference in transmission delays $|d_1 - d_2|$ is 37 ms, and the total packet size used was 1480 bytes, which leads to a critical bitrate of $B_C$ = 0.32 Mbit/s. In a practical experiment, such as ours, $B_C$ is not sharply defined because of delay variations. Therefore, with an increasing bitrate, a gradual increase in reordering can be observed. From the point where almost every packet on the slow link is passed by a packet with higher sequence number on the fast link, $M_{TPRRS}$ remains close to 50 %.

As opposed to the previously mentioned related work [10], [11], this experiment confirms that in a multilink scenario, the ratio of late packets goes far beyond the stated 0.01 % to 1.65 % and quickly approaches 50 % for bitrates greater than the critical bitrate $B_C$.

## VI. Multilink Packet Reorder Reduction

There are several methods of reducing packet reordering at the receiver. This section discusses two methods, a receiver- and a sender-side approach, and presents experimental results to show their gain.

### A. Receiver-side Reorder Buffer

On the receiver side, packet reordering can be reduced through the use of a buffer. Every packet that has a sequence number greater than the currently expected sequence number is put into a reorder buffer. If more out-of-order packets arrive than fit into the buffer, the packet in the buffer with the lowest sequence number (the most excessively delayed packet) will be discarded. From an application's point of view, such discarded packets are lost (another method, not treated here, would be to remove packets from the reorder buffer once a certain application-specific delay has expired).

When a lot of reordering happens, the reorder buffer will be heavily occupied, while in a scenario of permanent in-order delivery, the reorder buffer occupancy would amount to zero. Figures 6 and 7 illustrate that for the combined use of the two heterogeneous links $N_{WLAN_1}$ and $N_{HSDPA}$, the required buffer size increases with increasing bitrate and that, if a certain number of discarded packets can be tolerated, the buffer size can be reduced.

In order to capture the workload of such a reorder buffer, RFC 5236 also defines the *Reorder Buffer-Occupancy Density* metric, denoted here as $M_{RBD}$. Figure 6 shows the results of a reorder buffer implemented in our testbed as well as the $M_{RBD}$ metric values obtained from it. For illustrative purposes, a small buffer that can hold 4 packets was used. When the bitrate was set to a low value, such as 0.1 Mbit/s, no reordering occurred, and the reorder buffer was never utilized. With a higher bitrate, such as 1.0 Mbit/s, packets experienced reordering, and the buffer was occupied by 2 packets in 45 % of the time and by an overall maximum of 4 packets in 5 % of the time. If the buffer had been allocated to fit only 3 packets, 5 % of the received packets would have been discarded.

If the discard of packets is not allowed, the required size of a reorder buffer quickly grows with an increasing bitrate (see Fig. 7). In other words, the buffer size would be as large as the most displaced packet in the corresponding $M_{RD}$ histogram. On the other hand, applications that are resilient to a certain degree of packet loss require a smaller buffer. In any case,

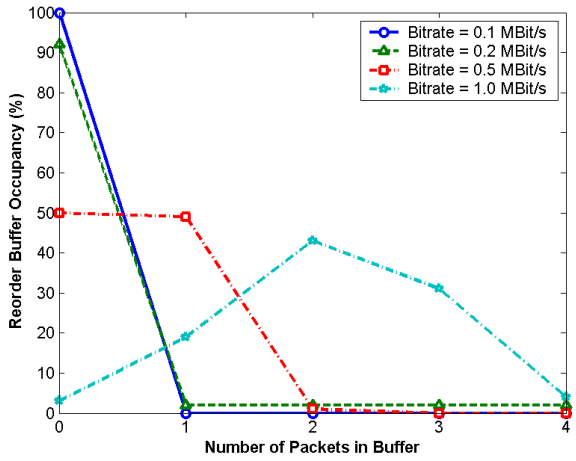


Fig. 6. The *Reorder Buffer Occupancy Density* metric gives a sense of a receiver's buffer requirements to reduce packet reordering. This example shows the load of a buffer with a capacity of 4 packets at different bitrates. The sum of each curve amounts to 100 %, indicating no packets were discarded.

packets with very high displacements might be delayed so much that they are useless to the application.

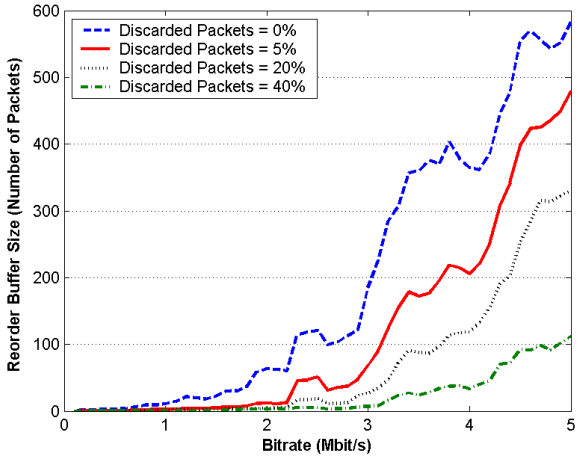


Fig. 7. Receiver-side buffer requirements depend on the bitrate used and on the number of packet discards that can be tolerated by the application.

Client-side buffers are not the only method to reduce the occurrence of packet reordering. As shown in the following section, the sender, or an intermediary proxy, is also able to impact the total disorder in a packet sequence.

### B. Sender-side Multilink Packet Scheduler

Packet reordering can also be mitigated by scheduling packets in a more sophisticated way over the available network interfaces than the simple round-robin assignment used in the experiments discussed so far. The sender has two degrees of freedom in deciding where to send packets. First, a link can be favoured according to its bandwidth, so that more packets travel over a fast link than over a slow one, and second, packets can be sent at different points in time in the attempt to compensate for heterogeneous delays.

To get a deeper insight into the heterogeneity of multiple wireless links, we design a static scheduler. If the heterogeneity shows little variation, a static scheduler built on the average delay and throughput should work close to perfect. An outline of a static scheduler is given in Algorithm 1. In order to guarantee a constant bitrate, packets are sent in equal time intervals, and in each step the scheduler selects the network interface to be used. The scheduler needs to have precognition of the available interfaces and their estimated delay $d_1$ and $d_2$. The main idea of the scheduler is to delay sending packets over the low-delay $N_{WLAN_1}$ link by putting them into a FIFO queue. According to the delay difference $|d_1 - d_2|$ of both interfaces and the bitrate used, it is calculated how large the FIFO queue must be to delay the packets over $N_{WLAN_1}$ in a way that they should experience a delay similar to the packets sent over $N_{HSDPA}$.

**Algorithm 1** Static Scheduler using a FIFO Queue

```
1:  Time interval [ms] T_I = (bitrate*10^6)/(packetsize*8) * 1000;
2:  Allocate queue Q of size q = ⌊|d_1 − d_2| / T_I⌋;
3:  for sequence number s = 1 to n do
4:      loop
5:          interface i = getInterfaceBySeqno(s);
6:          if (i == N_HSDPA) then
7:              exit loop;
8:          else if (i == N_WLAN1) then
9:              Q.put(s);
10:             if Q.full() then
11:                 Q.get(s);
12:                 exit loop;
13:             else
14:                 s++;
15:             end if
16:         end if
17:     end loop
18:     send sequence number s on interface i;
19:     wait(T_I);
20: end for
```

The function *getInterfaceBySeqno()* used in Algorithm 1 decides to which of the receiver's network interfaces a packet is sent. The same function is also used at the receiver, which makes it possible to determine on which interface a lost packet should have arrived. The round-robin version of *getInterfaceBySeqno()* is displayed in Algorithm 2.

**Algorithm 2** "Round-Robin" getInterfaceBySeqno(s)

```
1: if (s modulo 2 == 0) then
2:     return N_HSDPA;
3: else if (s modulo 2 == 1) then
4:     return N_WLAN1;
5: end if
```

Figure 8 shows how the total packet disorder reduces when the static scheduler is used. Compared to the purely round-robin packet allocation, $M_{RE}$ decreases on average by 38 % when using a queue that buffers packets before sending them on the fast link.

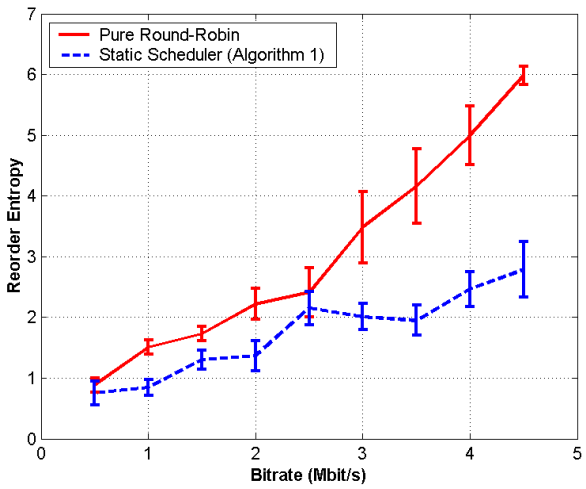


Fig. 8. Comparison of the *Reorder Entropy* $M_{RE}$ for a purely round-robin versus a static scheduler that is based on average link estimates (assuming WLAN vs. HSDPA: delays = 40:80 ms, bitrate ratio = 3:1).

The main problem of static schedulers is that the delay may vary over time and that they are dependent on the bitrate used. The closer the bitrate approaches link saturation, the more will the experienced delays increase. Therefore, a static scheduler based on fixed delay estimates cannot be expected to perfectly solve packet reordering in an environment with dynamic heterogeneity as has been demonstrated in wireless networks. The design of a scheduler that adaptively adjusts to the link delays is indispensable.

## VII. Conclusions and Future Work

Through the aggregation of multiple concurrent links, it is possible to achieve increased throughput, but at the cost of high packet reordering at the receiver. Recent studies on multihoming and multipath routing have focused either on high-speed, wide-area networks or ignored the severe heterogeneity of wireless links.

In contrast, our work focuses on dividing packet sequences to hosts over multiple last-hop wireless networks. For this scenario, we have demonstrated with practical experiments that the dynamics and heterogeneity of multiple wireless links causes reordering of IP packets that clearly exceeds the one observed for common Internet connections.

By using sufficiently large buffers, packet reordering can be avoided. However, for devices with high resource constraints, the workload of using large buffers is expensive. Instead of discarding many packets that arrive out of order at a receiving device, it is also possible to reduce packet reordering by intelligently scheduling packets on the sender-side. Our experimental results show that a static sender-side scheduler, that has knowledge about the average delay and throughput of the links, is able to reduce client-side buffering by 38 % on average.

However, a static scheduler that is guided by average delay and throughput estimates works inaccurately in a highly dynamic wireless environment. Therefore, our future research will include the design of packet scheduling mechanisms that dynamically adapt to the current link characteristics based on feedback from the receiver.

Furthermore, we will analyze the impact of multiple links on packet reordering in TCP connections. We plan to implement a proxy solution without the requirement of server-side modifications, enhancing the research done in [16]. Tunneling mechanisms will be applied to transparently split traffic of the same TCP stream over the proxy to all available client-side interfaces.

## VIII. Acknowledgements

The authors would like to express their gratitude to Haakon Riiser from Netview Technology AS (www.netview.no) for his support and for providing us with a research license of the Sigma Network Analyzer. We also thank Erlend Viddal from Lividi, Eunah Kim from ETRI, Peter Kaspar from ETH Zurich, and Matthias Wille from Avaloq for their valuable feedback.

## References

[1] L. Golubchik, J. C. S. Lui, T. F. Tung, A. L. H. Chow, W.-J. Lee, G. Franceschinis, and C. Anglano, "Multi-path continuous media streaming: what are the benefits?" *Perform. Eval.*, 2002.

[2] C. Hopps, "Analysis of an equal-cost multi-path algorithm," IETF RFC 2992, November 2000.

[3] D. S. Phatak, T. Goff, and J. Plusquellic, "IP-in-IP tunneling to enable the simultaneous use of multiple IP interfaces for network level connection striping," *Computer Networks*, 2003.

[4] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, 2006.

[5] B. Wang, W. Wei, Z. Guo, and D. Towsley, "Multipath live streaming via TCP: scheme, performance and benefits," in *ACM CoNEXT*. New York, NY, USA: ACM, 2007, pp. 1–12.

[6] I. Gapanova, "Concurrent multipath transfer for heterogeneous networks," Master's thesis, TU Braunschweig, 2008.

[7] C.-M. Huang and C.-H. Tsai, "WiMP-SCTP: Multi-path transmission using stream control transmission protocol (SCTP) in wireless networks," in *AINA Workshops (1)*, 2007, pp. 209–214.

[8] J. Liao, J. Wang, and X. Zhu, "cmpSCTP: An extension of SCTP to support concurrent multi-path transfer," *ICC*, 2008.

[9] N. M. Piratla and A. P. Jayasumana, "Reordering of packets due to multipath forwarding – an analysis," in *International Conference on Communications (ICC)*, 2006.

[10] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and classification of out-of-sequence packets in a Tier-1 IP backbone," *ACM SIGCOMM Internet Measurement Workshop*, 2002.

[11] L. Gharai, C. Perkins, and T. Lehman, "Packet reordering, high speed networks and transport protocol performance," *13th International Conference on Computer Communications and Networks (ICCCN)*, pp. 73–78, October 2004.

[12] B. Ye, A. P. Jayasumana, and N. M. Piratla, "On monitoring of end-to-end packet reordering over the Internet," in *ICNS '06: Proceedings of the International conference on Networking and Services*. Washington, DC, USA: IEEE Computer Society, 2006, p. 3.

[13] N. M. Piratla and A. P. Jayasumana, "Metrics for packet reordering - a comparative analysis," in *International Journal of Communication Systems*, 2007.

[14] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser, "Packet reordering metrics," IETF RFC 4737, November 2006.

[15] A. Jayasumana, N. Piratla, T. Banka, A. Bare, and R. Whitner, "Improved packet reordering metrics," IETF RFC 5236, June 2008.

[16] K. Chebrolu, B. Raman, and R. R. Rao, "A network layer approach to enable tcp over multiple interfaces," *Wireless Networks*, vol. 11, no. 5, pp. 637–650, 2005.