Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 3307–3320
computer communications
www.elsevier.com/locate/comcom

Concurrent multipath communication for real-time traffic

M. Fiore [a], C. Casetti [a,*], G. Galante [b]

[a] Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi 24, 10129 Torino, Italy
[b] Networking Lab, Istituto Superiore Mario Boella, via P. Carlo Boggio 61, 10138 Torino, Italy

Available online 22 December 2006

Abstract

The growing availability of multiple network interfaces on both mobile and fixed hosts makes concurrent multipath transfer (CMT) an appealing option to improve the performance of increasingly bandwidth-hungry multimedia applications. The Westwood Stream Control Transmission Protocol with Partial Reliability (W-PR-SCTP) is a partially reliable, SCTP-based transport protocol featuring a novel adaptive traffic-scheduling algorithm enabling CMT of multimedia real-time traffic on multihomed hosts. This paper introduces W-PR-SCTP, reports a thorough evaluation of the new protocol with the ns-2 network simulator under several traffic scenarios, outlines its implementation in Linux and its testing in a simple experimental setup.

Keywords: Concurrent multipath transfer; Multihoming; SCTP; Partial reliability; Multimedia; Real-time traffic

1. Introduction

It is more and more common for computer equipment in general and for laptops and palmtops in particular to have more than one network interface: alongside the ubiquitous digital subscriber line (DSL)/cable modems and Ethernet ports, wireless 802.11× cards are becoming a basic feature for every manufacturer, and other technologies, such as the General Packet Radio Service (GPRS), the Universal Mobile Telecommunications System (UMTS), Bluetooth or satellite links offer further access capabilities to network services. Moreover, the dramatic drop of broadband connectivity price could easily lead small- and medium-sized enterprises to prefer multiple low-cost broadband connections to an expensive, high-bandwidth Internet connection in the near future.

It is widely agreed that using several interfaces at the same time (i) would bring enhancements to the connection persistence, reliability and fault tolerance, (ii) would help during handovers, and (iii) would generally increase the available bandwidth, when sending data over several network interfaces at the same time. Nevertheless, most current transport- and network-layer protocols are generally unable to exploit multiple interfaces.

The Stream Control Transmission Protocol (SCTP) [1] standardized by the Internet Engineering Task Force (IETF) is a transport protocol that introduces support for non-simultaneous transmission over multiple paths. This is achieved through the concept of *association*, which is a set of transport-layer *connections* between the same pair of *endpoints*, each connection being routed over one of the available paths. However, as far as the standard is concerned, SCTP always uses at most one path at a time for transmission, leaving the remaining ones for redundancy (mainly for packets' retransmission, or for backup in case of link failure).

New, improved versions of SCTP have been proposed in several multihoming/multipath scenarios. These approaches can be classified in two categories: *best-path selection* and *concurrent multipath transfer* (CMT). The former assumes that the protocol always tries to choose the best available path (according to one or more metrics) and funnels all traffic onto it, while the latter allows more than one path

* Corresponding author. Tel.: +39 011 5644126.
E-mail addresses: marco.fiore@polito.it (M. Fiore), claudio.casetti@polito.it (C. Casetti), galante@ismb.it (G. Galante).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.12.016

to carry packets from the same connection and with the same source-destination endpoints.

An interesting approach to best-path selection using packet-pair bandwidth estimation and round-trip-time (RTT) measurements can be found in [2], whereas a combination of packet-pair and ack-based bandwidth estimation techniques is used in [3]. In [4] multihoming was applied within a UMTS/wireless local area network (WLAN) overlay architecture to improve throughput performance; it can also be successfully used by a mobile station to exploit an association over multiple WLANs at the same time, reducing the latency caused by a handoff that [5] showed to be around 50–400 ms.

Researches on extending SCTP to CMT currently in progress [6–10], agree on the need for structural changes to the original SCTP standard.

In [6,7] several modifications at the sender are proposed to compensate for the problems introduced by using a unique sequence-number space for data transfers occurring concurrently over multiple paths. A multihomed sender maintains per-destination virtual queues and spreads chunks across all available paths as soon as the congestion window allows it. Retransmission are triggered only when several selective acknowledgments (SACKs) report missing chunks from the same virtual queue.

Load-Sharing SCTP (LS-SCTP) [8] aggregates the bandwidth of the transmission paths between the communicating endpoints, and dynamically adds new paths as they become available. Path monitoring is used to stripe packets among all available paths ensuring that the association does not stall as a consequence of high loss rates or temporary path unavailability. The key idea is to introduce a per-association, per-path data-unit sequence-numbering that extends the per-association SCTP congestion control to a finer-grained per-path congestion control. This, however, requires a change in the SCTP packet format, and modifications at both the sender and the receiver.

Independent Per-path Congestion Control SCTP (IPCC-SCTP) [9] builds upon the LS-SCTP idea of enhancing SCTP with a per-path congestion control, but tries to avoid modifications in the packet format and in the receiver by keeping more state information at the sender about which path the individual data units are sent on.

Another transport-layer protocol that modifies the original TCP to support multiple end-to-end paths is mTCP [11]. This solution assumes a special routing layer that implements a resilient overlay network (RON), from which an mTCP source learns which paths are available to reach the destination. It also outlines a heuristic for the detection of disjoint paths.

Little attention has instead been paid to partially reliable, CMT transport protocols supporting multimedia real-time traffic. The focus of this paper is on Westwood Partial-Reliability SCTP (W-PR-SCTP), recently introduced in [10], which is based on the Partial Reliability extension to SCTP (PR-SCTP) [12], and on TCP Westwood+ [13]. *Partial reliability* is obtained by imposing an upper limit on the number of retransmissions after which an *SCTP data chunk* (i.e., the data payload unit in SCTP jargon) is *advance acknowledged* at the source and considered as delivered, even in the absence of an explicit acknowledgment from the destination. This may have marginal relevance, or even be detrimental, for data traffic, but becomes fundamental for multimedia audio/video applications, where timely delivery is more important than full reliability. The Westwood extension also provides a scheduling algorithm, based on the Westwood bandwidth-estimation technique [13], to determine the load sharing among the paths in the association. As far as the implementation is concerned, the key features of W-PR-SCTP are that (i) it only affects the sender, leaving the receiver unmodified, and (ii) it does not require any change in the SCTP packet format.

Our work draws on previous studies on bandwidth and bottleneck estimation and is especially based on the framework of TCP Westwood+ [13].

The paper is organized as follows. Section 2 introduces Westwood PR-SCTP. Section 3 presents simulation results for several *static scenarios*, where the protocol performance is evaluated both in absence of any interfering traffic and in presence of steady-state interfering TCP flows. On the other hand, Section 4 focuses on simulation results obtained in *dynamic scenarios* featuring time-varying interfering UDP traffic. Then, the experimental testbed is described in Section 5 and the measurements collected in simple static and dynamic traffic scenarios are compared with the results obtained by simulation on an equivalent topology, thus validating the implementation's correctness. Finally, Section 6 concludes the paper.

2. The Westwood PR-SCTP protocol

Although the original PR-SCTP features per-path congestion window, slow start threshold (SSTHRESH), and round-trip-time variables, along with per-path retransmission timers, the per-association Cumulative TSN Ack point, receiver window, and sequence-number space introduce a few problems for data transfers occurring concurrently over multiple paths. Section 2.1, reports the changes required at the sender to decouple the different paths. As a general remark, we found our modifications to be similar to those recommended by other CMT-related papers [6,7,9,14]. In the remainder of the paper, we will refer to the different paths with the variable i and differentiate among per-path variables using the i subscript.

Since RTT and bandwidth may be different across the paths, packets may reach the destination out of order. The solution to mitigate reordering is based on: (i) estimating the available bandwidth on each path, as described in Section 2.2, and (ii) using an appropriate bandwidth-aware packet-scheduling algorithm to stripe packets across all paths, as reported in Section 2.3, so that they reach the destination almost in order. Finally, Section 2.4 completes the description of W-PR-SCTP commenting on an improve-

ment to the fast-recovery mechanism enabled by the per-path bandwidth estimate. Section 2.5 introduces a simple bandwidth-agnostic greedy scheduler used as a benchmark for the bandwidth-aware scheduler. This allows to define a new protocol, dubbed g-PR-SCTP, that differs from W-PR-SCTP only because it uses the greedy scheduler as a replacement for the bandwidth-aware scheduler in Section 2.3.

2.1. Concurrent multipath support

The single-buffer architecture of PR-SCTP must be replaced by a *multibuffer* structure, giving to each interface its own send buffer. The multibuffer structure guarantees path independence as far as transmission is concerned, but introduces the need for modifications to SACK handling at the source.

We add a per-path congestion control, avoiding the limitations of the standard PR-SCTP, which only implements a per-association congestion control. Indeed, the standard PR-SCTP only allows for CWND adjustments when the association cumulative TSN Ack point is updated by an incoming SACK. Although this is correct when at most one path is used at any given time and consecutive packets arrive in-order at the receiver, when packets are concurrently transmitted to multiple destinations, the assumption of ordered reception does not hold anymore, and SACKs that do not update the association cumulative TSN Ack point may acknowledge one or more chunks that were received in-order on different paths. In this case, the standard SCTP would behave incorrectly, not updating the CWND of such paths, and missing all CWND updates that do not move the association cumulative TSN Ack point. This problem can be easily solved by introducing a per-path cumulative TSN Ack point and updating it after processing a meaningful SACK, as reported in the following pseudocode fragment, where $\texttt{cumTsnAckPoint}_i$ is the cumulative TSN Ack point on path i.

```
for(each path i in the association) {
  if(cumTsnAckPoint_i < cumTsnAckPoint)
    cumTsnAckPoint_i = cumTsnAckPoint;
  for(each gap block z in the received SACK)
    for(each outstanding chunk x) {
      j = path x was last sent on;
      if(i == j && x is not ack'd &&
        TSN (x) is before or after z)
          exit;
      if(i == j && TSN (x)> cumTsnAckPoint_i
      &&
        TSN (x) is inside z)
          cumTsnAckPoint_i = TSN (x);
    }
}
```

Another point that requires attention is the Fast Retransmit algorithm: the original PR-SCTP triggers it after four consecutive SACK missing reports, leading to congestion window reduction and retransmissions on the connection over which the presumably-lost packets were last sent. However, when splitting a traffic flow over multiple connections, the path diversity induced by different bandwidth and delay in each path in the association, may lead to a high number of out-of-order packet arrivals at the receiver. This, in turn, causes the receiver to advertise large, lasting gaps in the received chunk sequence, forcing several fast retransmissions and triggering unneeded CWND reductions. The solution to this problem too is based on per-connection SACK processing. Chunk missing reports are trusted only if the SACK acknowledges for the first time at least one chunk that (i) was transmitted over the same path of the missing chunk, and (ii) had a TSN higher than that of the missing chunk. In other words, the number of missing reports for a chunk is incremented only if one or more chunks, sent on the same path as the missing one and after it, were acknowledged, thus implying the possibility of a real loss. The pseudo-code for the per-connection handling of Fast Retransmit performed upon SACK reception is as follows.

```
for(each path i in the association) {
  highestNewTsnAcked_i = cumTsnAckPoint_i;
  numNewlyAckedBytes_i = 0;
}
for(each gap block z in the received SACK)
  for(each chunk k in z) {
    if( k is ack'd for the first time) {
      i = path k was last sent on;
      numNewlyAckedBytes_i += size of (k);
      if (TSN (k)> highestNewTsnAcked_i)
        highestNewTsnAcked_i = TSN (k);
    }
  }
for(each outstanding chunk x ) {
  i = path x was last sent on;
  if(numNewlyAckedBytes_i > 0 &&
    highestNewTsnAcked_i > TSN (x)) {
      numMissingReports_x ++;
      if(numMissingReports_x > 3)
        fast retransmit on path i;
  }
}
```

Notice that the receiver's architecture and behavior remained unmodified: Westwood PR-SCTP, like standard PR-SCTP, needs only a single receiver buffer collecting the chunks from all the connections in the association. SACKs are generated as in PR-SCTP and transmitted over the channel from which the last packet was received. Further, Westwood PR-SCTP introduces no overhead because it does not require any change to the original SCTP packet format.

We also remark that out-of-order SACKs are not trusted as in the standard PR-SCTP, and are discarded, because CMT may easily lead to SACK reordering.

Finally, no detailed discussion of retransmission handling is given in this paper: since we focus on real-time traffic support, the number of retransmissions after which a chunk is advanced acknowledged is set to *zero*, disabling retransmissions. For a discussion of the issue of CMT retransmission, the interested reader can refer to [6,7].

2.2. Westwood bandwidth estimation

The per-path SACK processing allows to perform independent bandwidth estimations on each path. Westwood SCTP is named after TCP Westwood+ [13], which bases its estimate on measurements taken on path i over contiguous, non-overlapping time windows, lasting either one round-trip time RTT_i, or 50 ms, whichever is larger. Therefore, in this section, we denote all variables with the integer superscript k, to indicate the window they refer to. The k-th sampling window on path i can then be written as:

$$\Delta t_i^{(k)} = \max\{RTT_i^{(k)}, 50\ \text{ms}\}$$

and the current available bandwidth estimate $\hat{B}_i{}^{(k)}$ is computed as a smoothed auto regressive moving average (ARMA) of the *bandwidth sample* $\hat{s}_i^{(k)}$ and the bandwidth-estimate on window $k-1$.

$$\hat{B}_i^{(k)} = \hat{s}_i^{(k)}\left(1 - \mathrm{e}^{-\Delta t_i^{(k)}/\tau}\right) + \hat{B}_i^{(k-1)}\mathrm{e}^{-\Delta t_i^{(k)}/\tau}$$

where τ is set to 1 in our implementation, and the bandwidth sample

$$\hat{s}_i^{(k)} = D_i^{(k)}/\Delta t_i^{(k)}$$

is obtained on all the data $D_i^{(k)}$ sent on path i over the window $\Delta t_i^{(k)}$. Note that the smoothing filter for computing $\hat{B}_i^{(k)}$ has time-varying coefficients unlike the classical exponential moving-average filters because the time-intervals over which the bandwidth samples are taken are not constant. In addition, the bandwidth estimate is always lower-bounded by 5 kbps so as to prevent unused paths experiencing very-low throughput, and, consequently, near-zero bandwidth estimates, from completely starving. when the bandwidth-aware scheduler described after is employed.

2.3. Bandwidth-aware packet-scheduling algorithm

Whenever a new chunk becomes available for transmission, the scheduler computes its *reception index* for each path i, as

$$R_i = \frac{O_i + S_i + C}{\hat{B}_i^{(k)}}$$

where C is the chunk size, while O_i and S_i are, respectively, the total size of the outstanding and of the buffered chunks for path i. The resulting R_i reflects on the one hand the congestion-window emptying speed of the connection over path i: a higher bandwidth $\hat{B}_i{}^{(k)}$ leads to a faster packet acknowledgment and therefore to a quicker window clearing; opposingly, a higher outstanding- and queued-chunk size ($O_i + S_i$) increases the time needed to empty the congestion window. Obviously, a path whose window is emptied faster has a higher probability to transmit the chunk first. On the other hand, R_i also includes the term $C/\hat{B}_i{}^{(k)}$, accounting for the time that the current chunk will take to reach its destination over path i. The path with the lowest R_i is therefore selected as the chunk destination: this means that the chunk could be allocated to a destination whose CWND is currently full, but which the scheduler estimates will deliver the chunk first, due to higher bandwidth.

Since this scheme can be used only after at least one bandwidth estimate has been performed for all the involved paths, just after opening the association, a simple round-robin scheduler uniformly distributes the chunks among the paths, until a $\hat{B}_i{}^{(k)}$ value is obtained for every path. In this study, we assume that the advertised RWND is infinite, so that it does not constrain the sender.

2.4. Westwood fast recovery

The per-connection Westwood bandwidth estimate is leveraged every time a retransmission-timeout (RTO) expires, or a fast-retransmission is triggered on path i, and the CWND must be reduced, to set $SSTHRESH_i$ to the path's estimated bandwidth-delay product as in

$$SSTHRESH_i = \max\{\lfloor \hat{B}_i^{(k)} \cdot \min RTT_i \rfloor, 2MPDU\}$$

instead of halving its previous value, as the original SCTP would have done. Note that, in the formula above, min-RTT_i is the minimum RTT on path i, whereas $MPDU$ is the maximum SCTP protocol-data-unit size. As shown in [13], this approach allows for a more accurate congestion control.

2.5. Greedy packet-scheduling algorithm

We also define as *greedy* scheduler a scheduler that sends data over a path comprised in the association as soon as some room is available in its congestion window. The aim is to show that g-PR-SCTP, the protocol using the greedy scheduler as a replacement for the bandwidth-aware scheduler featured by W-PR-SCTP, being agnostic to the different paths' bandwidth, may cause packets sent over slower paths to arrive at the receiver much later than those sent over faster paths, producing high packet reordering at the receiver and triggering large duplicate-SACK transmissions.

3. Simulation of static scenarios

W-PR-SCTP and g-PR-SCTP were implemented in the network simulator ns-2 and tested in several network sce-

narios. Section 3.1 considers a multipath scenario comprising two, three or four independent paths featuring different bandwidth and RTT. Section 3.2 evaluates the performance of PR-SCTP and g-PR-SCTP in a two-path, multi-bottleneck topology including a common bottleneck link. Finally, Section 3.3 assesses the performance of W-PR-SCTP on the same topology as in Section 3.2, when steady-state interfering TCP flows and random link-losses are introduced.

3.1. Multipath scenario

A first set of simulations was run on a simple network with two multihomed nodes connected by two, three or four paths. The aim of such preliminary test was to assess the W-PR-SCTP and the g-PR-SCTP capability to provide smart load balancing on multiple paths with different bandwidth-delay product, as summarized in Table 1.

A single constant-bit-rate (CBR) application was run on one of the two nodes, generating data at 300 kbps. Notice that the aggregated 350-kbps bandwidth available on the paths in each configuration was more than enough to transport the traffic generated by the CBR application.

Table 1
Bandwidth/delay setup for the multipath scenario

Number of paths	Path id	Bandwidth (kbps)	Propagation delay (ms)
2	1	300	40
	2	50	80
3	1	175	40
	2	125	80
	3	50	60
4	1	125	30
	2	100	70
	3	75	80
	4	50	90

On the left-hand side of Fig. 1, obtained in the four-path scenario using W-PR-SCTP, the traffic is balanced across all the available paths according to their bandwidth, reaching the expected 300-kbps throughput. The right-hand side of Fig. 1 shows the instantaneous jitter at the receiver as a function of the number of paths. In all cases, the jitter remained below 25 ms. This, together with average delays in the 150- to 200-ms range, and packet losses around 0.2%, led to excellent real-time-application performance. When a multimedia player with a three-second playout buffer located at the receiver was simulated, it was found capable of uninterrupted playback for 79.6 out of the 80-s run.

The same scenario was tested with g-PR-SCTP: the throughput results were similar to those for W-PR-SCTP, but the performance for real-time traffic dramatically deteriorated. The jitter at the receiver is depicted in Fig. 2: if data were sent as soon as the paths' CWNDs allowed it, the packet ordering at the receiver became unpredictable, especially when the paths' bandwidth and propagation delay were different, and the jitter experienced peaks of 200 ms. We also recorded average delays ranging from 500 ms (two-path scenario) to 1.1 s (four-path scenario) and losses ranging from 1% (two-path case) to 6% (four-path case): these values were caused by an unbalanced load sharing, which led to alternated buffer filling and emptying on both paths. As a consequence, the multimedia player at the receiver suffered from frequent freezing due to playout-buffer emptying.

3.2. Multi-bottleneck scenario: standalone association

Additional simulations were run in the scenario represented in Fig. 3. The traffic was generated by a 300-kbps CBR application sending traffic from the single-homed to the dual-homed node. We studied three different cases, corresponding to the three bandwidth/delay settings for links

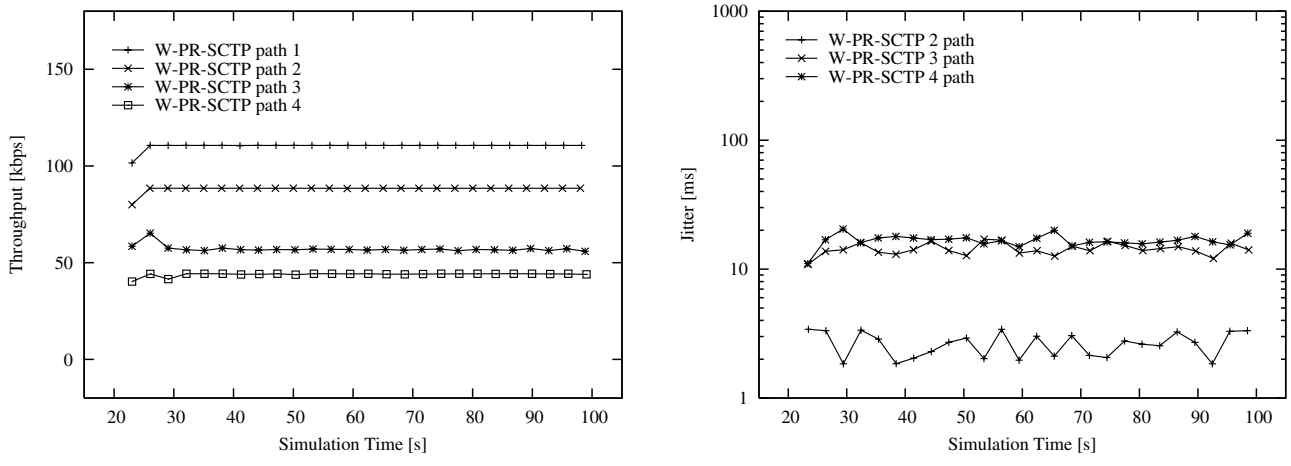


Fig. 1. Multipath scenario: W-PR-SCTP per-path throughput versus simulation time for a four-path CMT (left), and W-PR-SCTP jitter versus simulation time for two-, three-, and four-path CMTs (right).

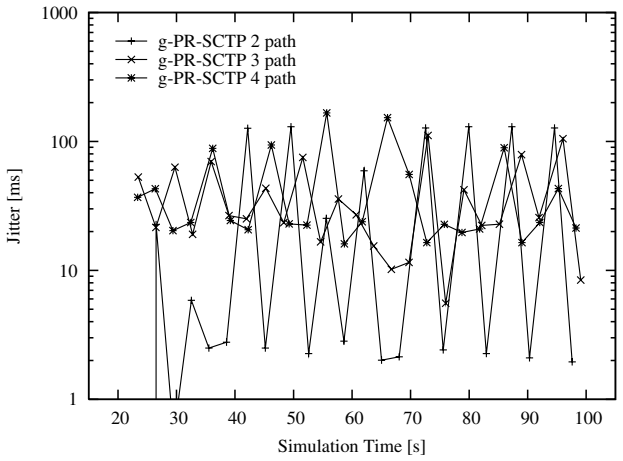


Fig. 2. Multipath scenario: g-PR-SCTP jitter versus simulation time for two-, three-, and four-path CMTs.

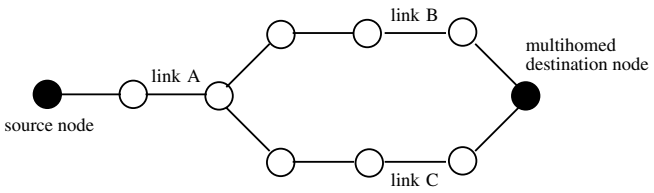


Fig. 3. Multi-bottleneck scenario.

A–C (Fig. 3), as reported in Table 2. All the other links had a 800-kbps bandwidth and a 10-ms propagation delay. The path comprising link B will be referred to as *upper path*, whereas the one including link C as *lower path*.

In case 1, the disjoint links B and C were the upper- and lower-path's bottleneck, whereas in case 2 there was a common bottleneck on shared link A, and a second bottleneck on link C, on the lower path. In both cases, W-PR-SCTP was able to split traffic on the two available paths, as shown on the left-hand side of Fig. 4, representing the per-path throughput in case 2, and on the right-hand side of Fig. 4, depicting the jitter at the receiver in case 1 and 2.

The same did not hold for g-PR-SCTP. Fig. 5 shows that the jitter at the receiver exhibited frequent variations and was generally more unstable in case 1 than in case 2. Moreover, W-PR-SCTP had better performance in terms of end-to-end delay, packet-loss rate, and reordering delay.

Table 2
Bandwidth/delay settings for the multi-bottleneck scenario

Case	Link	Bandwidth (kbps)	Propagation delay (ms)
1	A	350	10
	B	300	10
	C	50	10
2	A	250	10
	B	300	10
	C	50	10
3	A	250	10
	B	800	10
	C	800	10

In case 3, both paths were constrained at the shared bottleneck-link A. Simulation results show that W-PR-SCTP still managed to balance the load between the paths, but the jitter, in Fig. 6, was greater than the previous cases and comparable with that obtained by g-PR-SCTP: this was caused by the shared bottleneck that made it more difficult for the sender to infer the paths' characteristics. Delay, packet loss, and playout statistics suggest that, in presence of a shared bottleneck, W-PR-SCTP performs similarly to g-PR-SCTP.

In no case among those studied did W-PR-SCTP perform *worse* than g-PR-SCTP, while in most situations it brought noticeable improvements to jitter, end-to-end delay, packet loss rate, and reordering delay at the receiver.

3.3. Multi-bottleneck scenario with background traffic and channel losses

We next tested W-PR-SCTP in a multi-bottleneck scenario featuring channel losses and background traffic. In our first set of simulations, we had TCP connections sharing the upper path of the topology of Fig. 3 with the same W-SCTP-PR association described in the previous subsection. Independent channel packet losses were introduced on Link B. We tested 1 and 5 TCP connections, and packet loss rates up to 10%. However, for the sake of brevity, we just report the most interesting results, i.e., those for 1 TCP connection, comparing the cases of no losses and 10% loss rate, and setting bandwidth and delay as in cases 1 and 3 of Table 2.

Fig. 7 reports the comparison of W-SCTP-PR throughputs in case 1 with (right-hand side) and without losses (left-hand side). The left plot is reminiscent of the left plot in Fig. 4 (standalone case), except for a throughput surge caused by the presence of the concurrent TCP connection and a slow convergence to a slightly lower throughput than in the standalone case. If the upper path is affected by packet losses (right plot), the throughput oscillates quite dramatically, but no effect is perceived on the lower path, which is already filled to the brim.

On the contrary, in case 3, when both paths are overprovisioned after the bottleneck (link A), the presence of the concurrent TCP connection yields wild throughput spikes on either path in the no-losses case (Fig. 8, left plot). Beside the scheduler uncertainty already observed in the standalone case, the perturbation introduced by the downstream TCP connection is enough to undermine the stability of the scheduler. However, if losses affect the upper path, the scheduler reins in the W-PR-SCTP connections so that the lower path, unaffected by losses, carries most of the traffic (right plot).

4. Simulation of dynamic scenarios

A set of simulations was carried out to assess the performance of W-PR-SCTP in presence of time-varying interfering traffic. Section 4.1 compares the effectiveness

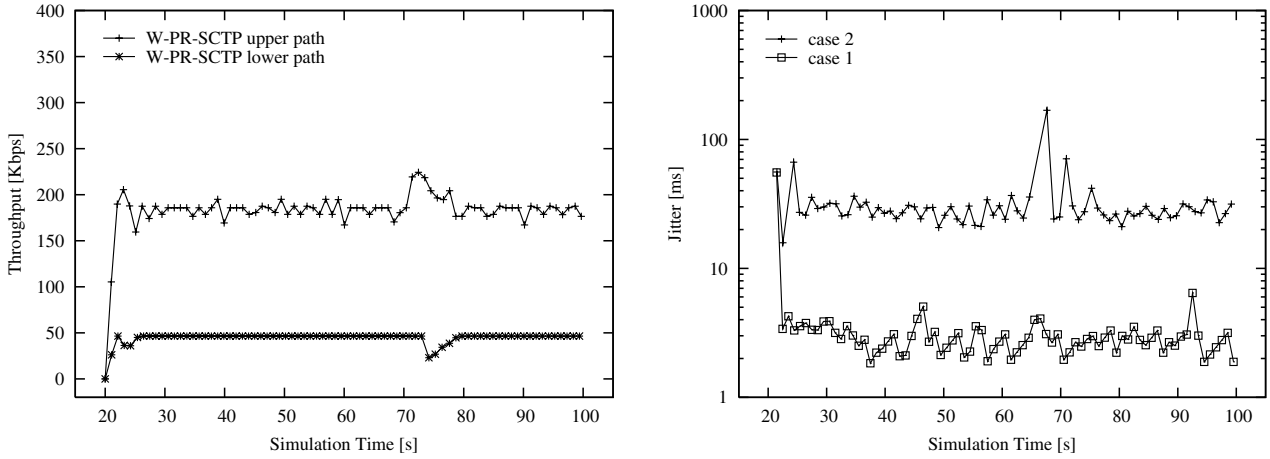


Fig. 4. Multi-bottleneck scenario: W-PR-SCTP per-path throughput versus simulation time in case 2 (left), and jitter versus simulation time in case 1 and 2 (right).

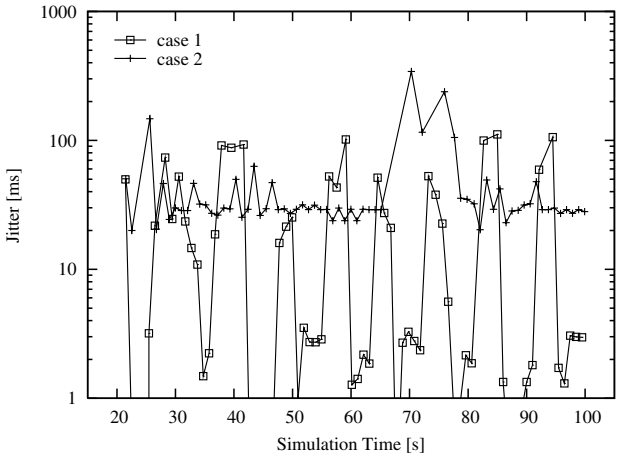


Fig. 5. Multi-bottleneck scenario: g-PR-SCTP jitter versus simulation time in case 1 and 2.

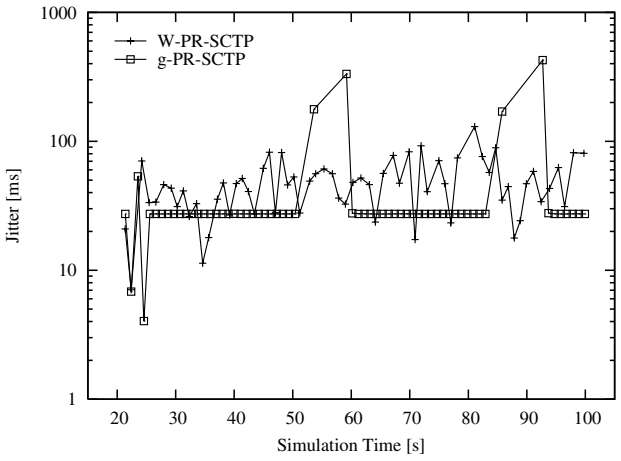


Fig. 6. Multi-bottleneck scenario: jitter for W-PR-SCTP and g-PR-SCTP versus simulation time in case 3.

of W-PR-SCTP, g-PR-SCTP and UDP as multimedia transport protocols. Section 4.2 assesses the impact of several UDP flows on the performance of W-PR-SCTP.

4.1. Symmetric interference

The first scenario involved two dual-homed nodes, connected by two paths, with a 384-kbps bandwidth each and the same 60-ms propagation delay. Alongside W-PR-SCTP, which carried traffic generated by a 300-kbps CBR application, we ran six 300-kbps UDP streams evenly split among the two paths. The UDP flows fully congested a path when they started up, and we timed them so that the two paths between the W-PR-SCTP source and destination nodes became alternately congested. The left-hand side of Fig. 9 shows that W-PR-SCTP managed to switch from the congested to the uncongested path, thus keeping its throughput constant.

In an equivalent single-homed benchmark simulation scenario, using UDP as the transport layer protocol for the 300-kbps CBR application and a single channel with bandwidth 768 kbps (i.e., 2×384 kbps) becoming congested in the same way, the recorded throughput was 241 kbps and losses around 20% were observed. The resulting throughput is plotted on the right-hand side of Fig. 9. The metrics achieved by W-PR-SCTP were a 298-kbps throughput and a mere 0.25% loss percentage.

Fig. 10 reports the jitter measured in the multihomed scenario when W-PR-SCTP and g-PR-SCTP were used as the transport protocol. By simply looking at jitters, the effectiveness of W-PR-SCTP (almost always below 0.1 ms, except for the instant when the interfering traffic switched on) is evident Fig. 10 also depicts the tendency of g-PR-SCTP to stick to the primary path, even when congested, a behavior confirmed by the simulations shown in the remainder of the paper. However, further evidence is provided by Fig. 11, depicting the status of the receiver playout buffer when the transport protocol was W-PR-SCTP, g-PR-SCTP or UDP. The curves show the performance of W-PR-SCTP with a one- or an eight-second playout buffer, as well as the performance of W-PR-SCTP and that of UDP with an eight-second playout buffer.

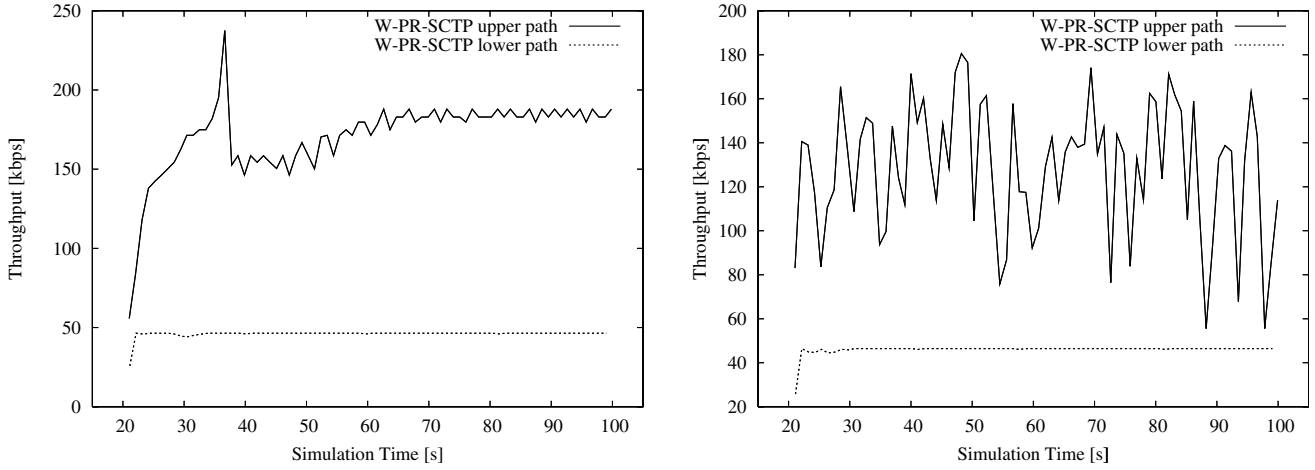


Fig. 7. Multi-bottleneck scenario with TCP background traffic: W-PR-SCTP per-path throughput versus simulation time without channel losses (left) and with a 10% channel loss rate on link B (right) – case 1.

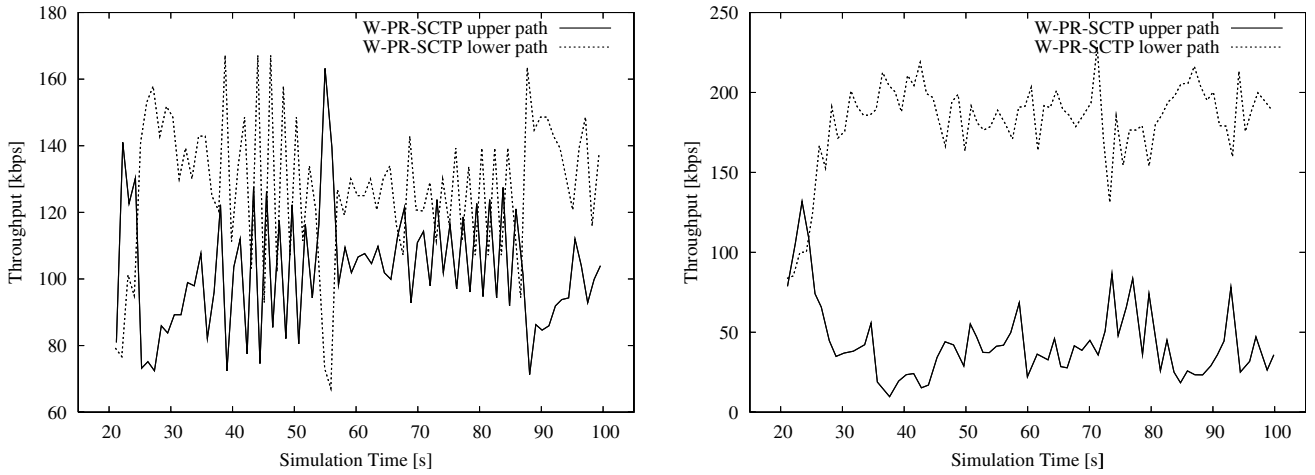


Fig. 8. Multi-bottleneck scenario with TCP background traffic: W-PR-SCTP per-path throughput versus simulation time without channel losses (left) and with a 10% channel loss rate on link B (right) – case 3.

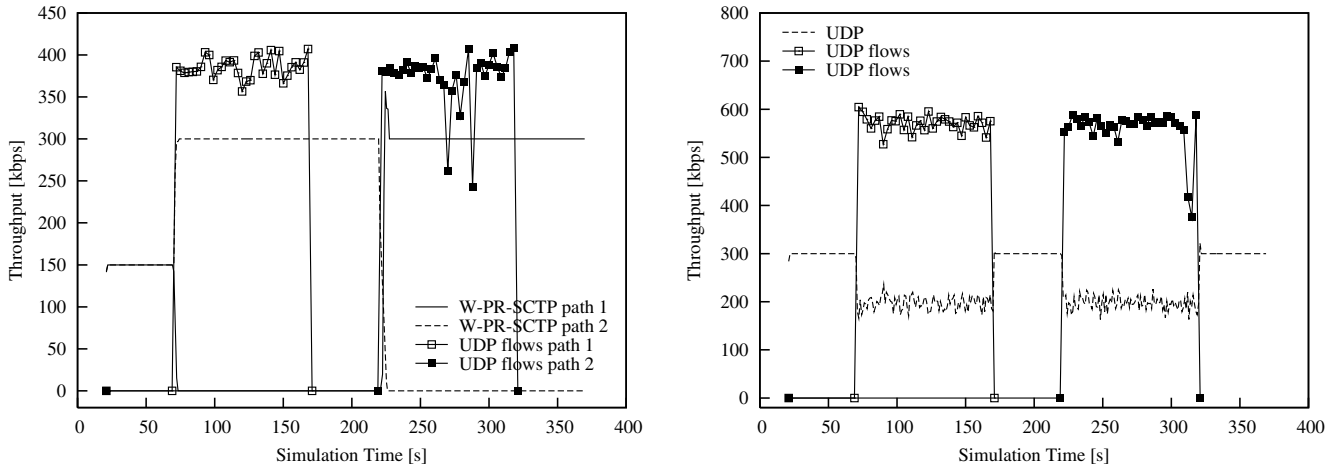


Fig. 9. Multimedia streaming over W-PR-SCTP (left) and UDP (right): per-flow throughput versus simulation time.

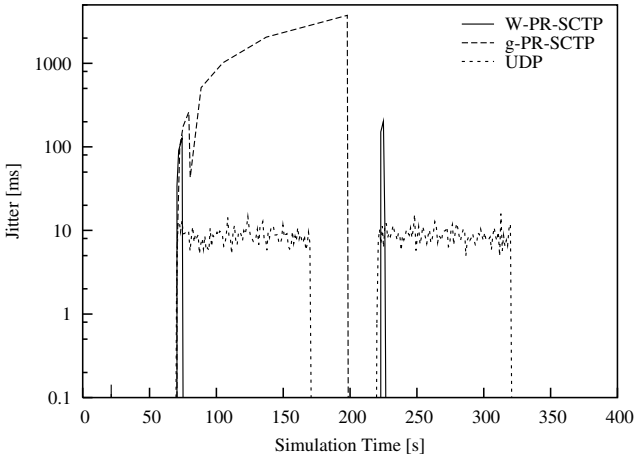


Fig. 10. Multimedia streaming over W-PR-SCTP and g-PR-SCTP: jitter versus simulation time.

Notice that W-PR-SCTP always performed well, with an almost continuous playout. Obviously, the larger the buffer size, the better the performance. On the other hand, g-PR-SCTP and UDP exhibited a very poor multimedia quality because of frequent freezing, even though the playout buffer never emptied in the latter case.

4.2. Asymmetric interference

In this second scenario, the topology and traffic flows depicted in Fig. 12 were simulated. A W-PR-SCTP association was established over the two paths connecting a 300-kbps CBR source with its destination. All point-to-point links had a propagation delay of 20 ms. Several competing UDP flows were then started over the middle links of both paths, with different timings, changing over time the bandwidth available for each connection and the position of the related bottleneck.

Fig. 13 shows the resulting W-PR-SCTP throughput along with the traffic generated by the superposition of the interfering UDP flows. For a better understanding,

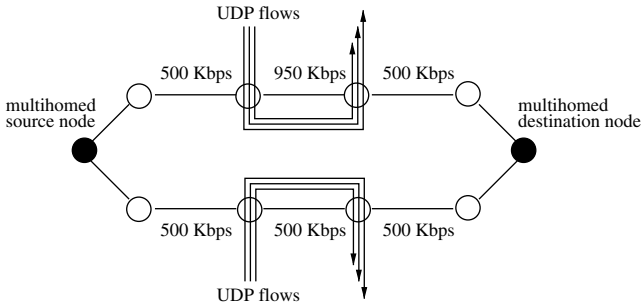


Fig. 12. Interfering UDP cross traffic scenario.

we labeled the different phases in the scenario evolution with increasing numbers, at the bottom of Fig. 13.

In phase 0, no UDP traffic was injected in the network. The two overprovisioned 500-kbps connections led the scheduler to split the W-PR-SCTP traffic evenly on the two paths.

Then, in phase 1, a first UDP flow, carrying traffic at a 300 kbps rate, was started over the middle link of the first path, in Fig. 12. Since such link had a 950-kbps bandwidth, the W-PR-SCTP flow, which used little more than 150 kbps on such link, and the UDP flow could coexist with no performance reduction.

The same was true during phase 2, when a second UDP flow, identical to the first one, was introduced on the same link.

Instead, when a third 300-kbps UDP connection was established over that link in phase 3, the overall UDP offered load summed up to 900 kbps, reducing the bandwidth available for W-PR-SCTP over the first path to a mere 50 kbps. The scheduler promptly identified the new bottleneck, and moved all the traffic over to the second path.

In phase 4 we began congesting the second path as well, by establishing a 100-kbps UDP connection on the middle 500-kbps link. However, the remaining bandwidth was still enough to accommodate the whole W-PR-SCTP traffic load.

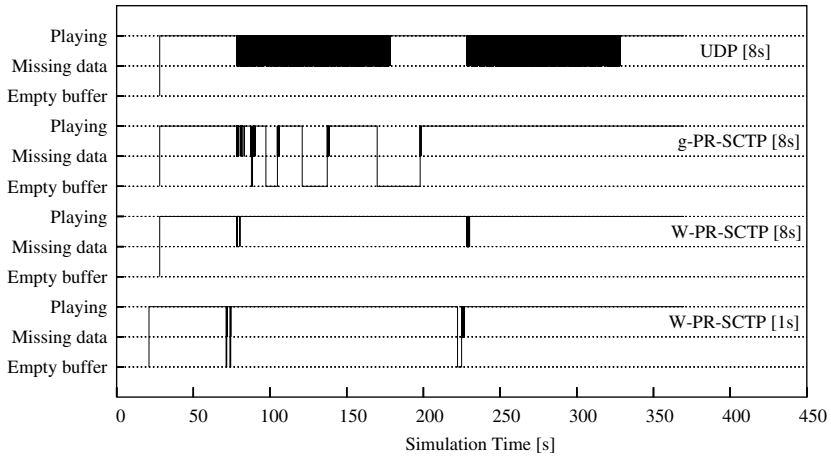


Fig. 11. Multimedia streaming over UDP, g-PR-SCTP and W-PR-SCTP: receiver playout status versus simulation time.

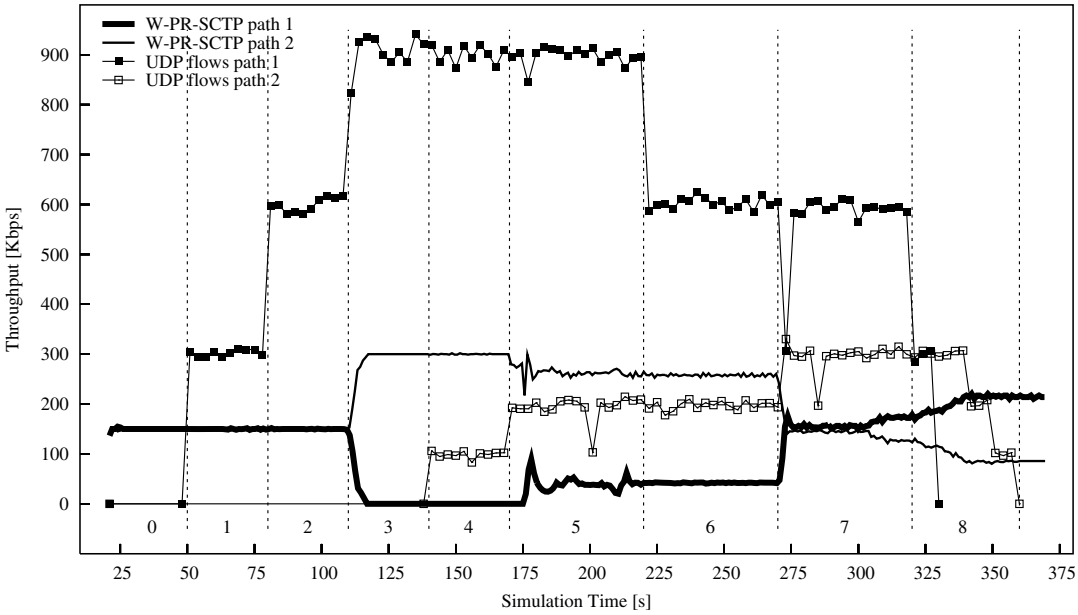


Fig. 13. W-PR-SCTP per-flow throughput versus simulation time.

This was not true anymore when, in phase 5, a second 100-kbps UDP flow was injected on that same link. The bandwidth left on the second path, amounting at around 300 kbps was not sufficient to transfer all the data and overhead generated by the CBR source attached to W-PR-SCTP. Thus, the protocol switched again part of the traffic to the first path, exploiting the small bandwidth left there to send all the data arriving from the application.

During phase 6, one of the UDP connections over the first path was torn down, but this did not influence the throughput, as W-PR-SCTP was already able to transfer all its traffic load.

On the other hand, the introduction of a third 100-kbps UDP flow over the second path, occurring in phase 7, further reduced the bandwidth usable by W-PR-SCTP on such connection, and forced the scheduler to move a larger part of the data traffic over to the first path, which, at that time, could accommodate it.

Finally, during phase 8, all the interfering UDP flows were stopped, one after the other.

Notice that the final load distribution was different from the initial one. This happened because we did not consider saturation conditions, which could have been obtained, for example, using a greedy traffic generator. In absence of saturating traffic, the scheduler tended to conserve the load distribution as long as it allowed to transfer all the data received by the upper layers. Consequently, competing traffic dynamics influenced the sharing of the constant load. This behavior possibly leads in time to different configurations than that obtained initiating the association on an unloaded network.

In any case, the different load balancing conditions experienced by W-PR-SCTP had a very-little impact on chunk ordering at the receiver. Fig. 14 shows that, regardless of how the traffic was split between the two paths, the bandwidth-aware scheduling guaranteed an almost-in-order chunk arrival at the receiver, with a jitter always lower than 20 ms. The time interval where the jitter was lower than 1 ms corresponds to the configuration in which W-PR-SCTP transmitted all the data over the same unused path.

The same scenario was simulated using the g-PR-SCTP protocol, resulting in the throughput shown in Fig. 15. The greedy scheduler favored the first connection directing all the traffic there. This did not spell any problem as long as the bandwidth available at the bottleneck was sufficient to carry all the traffic generated by the CBR source; however, when this condition did not hold anymore, g-PR-SCTP kept using the first path, relaying the exceeding traffic on the second path. The

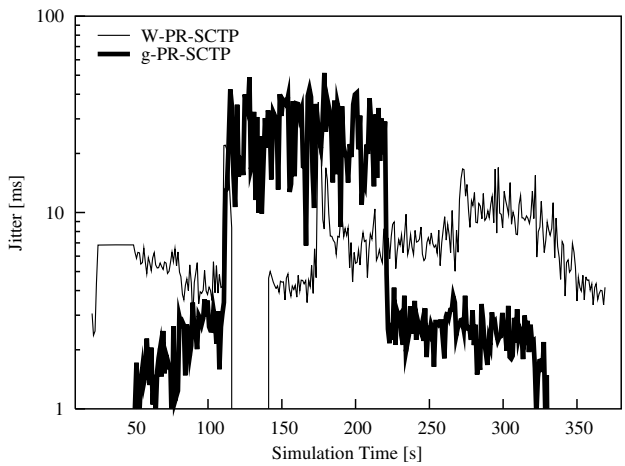


Fig. 14. W-PR-SCTP jitter versus simulation time.

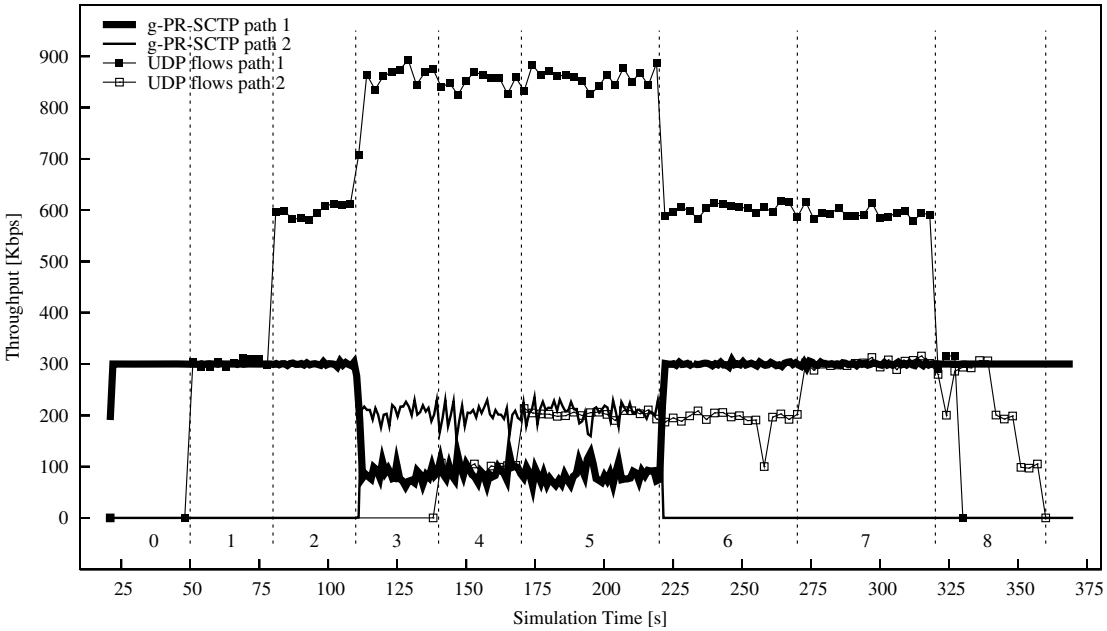


Fig. 15. Per-flow g-PR-SCTP and UDP throughput versus simulation time.

negative consequence of this behavior is shown in Fig. 14, where, in correspondence of the time interval when the traffic was split over the two connections, a jitter around 50 ms was experienced.

The playout at the receiver, in the two cases described above and considering a 3-s buffer, can be observed in Fig. 16. As a consequence of the jitter dynamics, g-PR-SCTP performed poorly when the first path have been congested, dropping many chunks. W-PR-SCTP instead provided a sufficiently ordered packet arrival at the receiver, resulting in a continuous data playout.

5. Experimental and simulation results

This section describes the testbed setup and the simple traffic scenarios under which the experimental results were compared with ns-2 simulations that tried to mimic similar network conditions and W-PR-SCTP protocol implementations.

5.1. Testbed setup

The experimental testbed depicted in Fig. 17 consisted of 5 personal computers (PCs) equipped with several Fast Ethernet network interface cards (NICs), one Fast Ethernet switch, and two Fast Ethernet hubs, all running at 100 Mbps.

The *Client* and the *Server* were two dual-homed PCs running SCTP-enabled client–server file-transfer programs on top of a patched Linux-2.6.12.2 kernel (patch available from [15]) supporting W-PR-SCTP.

The *Router* was a PC equipped with four Fast Ethernet NICs running Dummynet on FreeBSD 4.10 [16]. Dummynet [17] is a firewall extension to select packets using

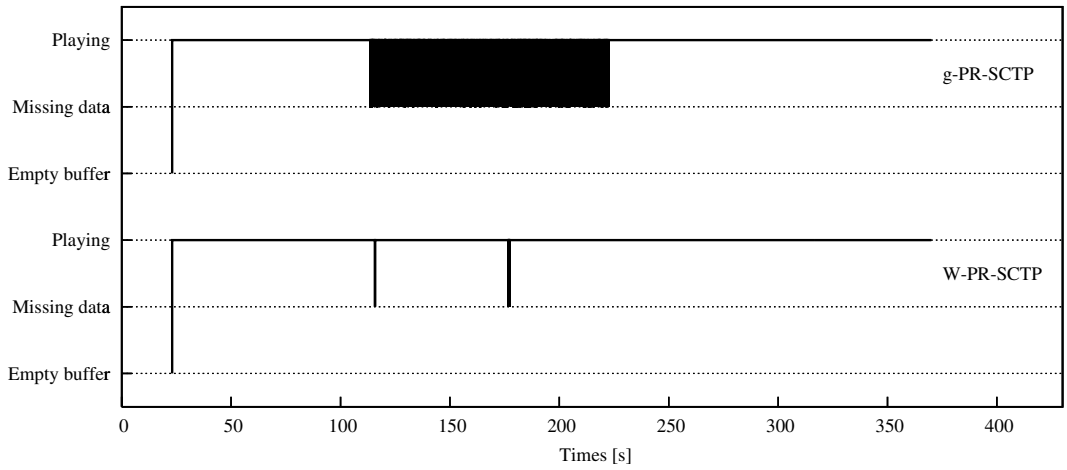


Fig. 16. Receiver playout status using g-PR-SCTP and W-PR-SCTP versus simulation time.

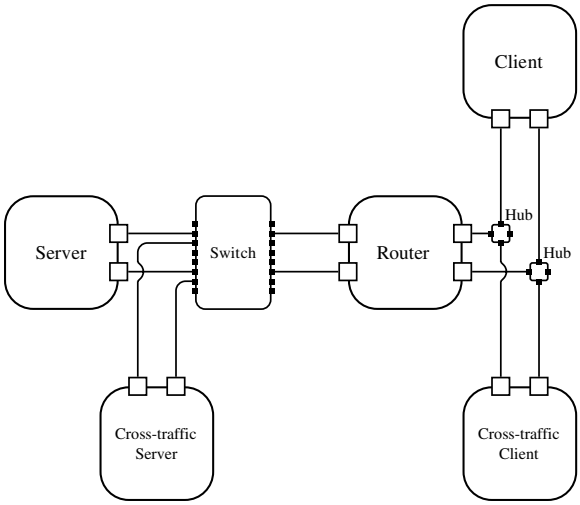


Fig. 17. Experimental testbed setup.

programmable rules and pass them through objects called *pipes*, which are used to emulate bandwidth and resource limitations, propagation delays and packet losses. Here, Dummynet was used to set up two parallel pipes, called *path 1* and *path 2*, having adjustable bandwidth and propagation delay, running between the left-hand side and the right-hand side Router interfaces. The FreeBSD real-time-clock-granularity `HZ` affecting the Dummynet timing precision was set to 10,000, corresponding to a 10-kHz clock tick-rate and a 100-μs resolution.

The *Cross-traffic Server* and the *Cross-traffic Client* were two dual-homed PCs running Linux 2.6.12.2, which could be used to generate interfering UDP traffic with the multi-generator toolset (MGEN) [18]. The two interfaces on the Cross-Traffic Server could be used to inject interfering traffic in the Switch, where it mixed with the traffic coming from the SCTP Server before entering the Dummynet pipes in the router.

A functionally equivalent topology was implemented in ns-2 for replicating by simulation the experiments carried out on the testbed and comparing the results in different scenarios. The main idea was to set up a W-PR-SCTP association between the Server and the Client and inject a variable amount of interfering non-elastic traffic to check the responsiveness of W-PR-SCTP to available-bandwidth oscillations.

5.2. Traffic scenarios

In our tests, we considered the simple topology shown in Fig. 17, comprising two multihomed W-PR-SCTP hosts and two cross-traffic generators, each pair connected through two distinct paths: one path was assigned a capacity of 2 Mbps (path 1), while the other a capacity of 1 Mbps (path 2). A propagation delay of 25 ms was artificially introduced on each path. A single W-PR-SCTP association was established between the Server and the Client and split into two connections running over a path each. The Server was engaged in sending greedy traffic to the Client.

First, we explored a static scenario. An experiment was run in which the Cross-traffic Server was sending UDP traffic to the Cross-traffic Client and was loading path 1 with a 500-kbps UDP stream (Fig. 18). The left-hand side of Fig. 18 shows the throughput achieved by W-PR-SCTP on each testbed path, while the right-hand side plot allows a comparison with simulation results in the same scenario. Correctly, the scheduler detected the UDP traffic throttling path 1 and lowered the portion of SCTP load on it, according to the bandwidth estimation. Both in simulation and testbed, SCTP losses ranged around 4%. Although both testbed and simulation experiments showed that W-PR-SCTP balancing is effective, a lower total throughput was observed for the testbed (i.e., 1.36 Mbps versus 1.43 Mbps on path 1), caused by a slightly different implementation of the Westwood filter with respect to simulation, due to Linux internal fixed-point algebra. The lower throughput was thus determined by a more sluggish response of the testbed filter.

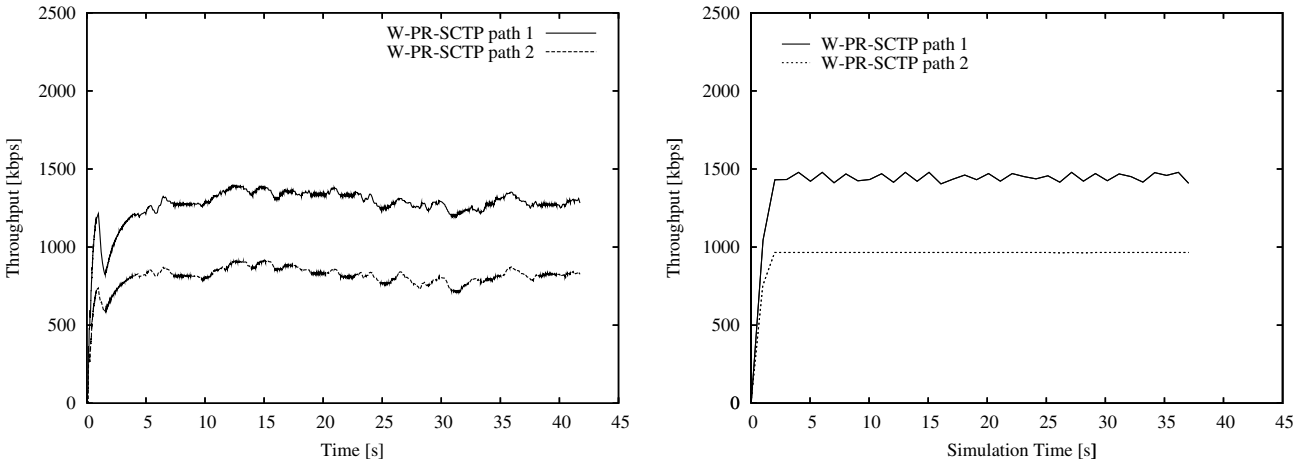


Fig. 18. Comparison between experimental (left) and simulation (right) results; steady 500-kbps UDP stream on path 1.

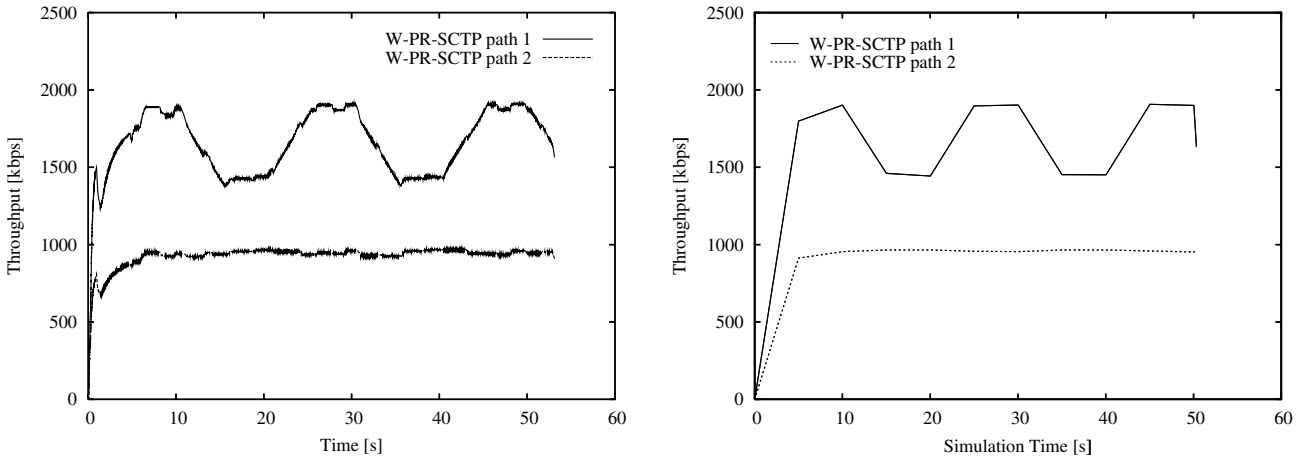


Fig. 19. Comparison between experimental (left) and simulation (right) results; intermittent 500-kbps UDP stream on path 1.

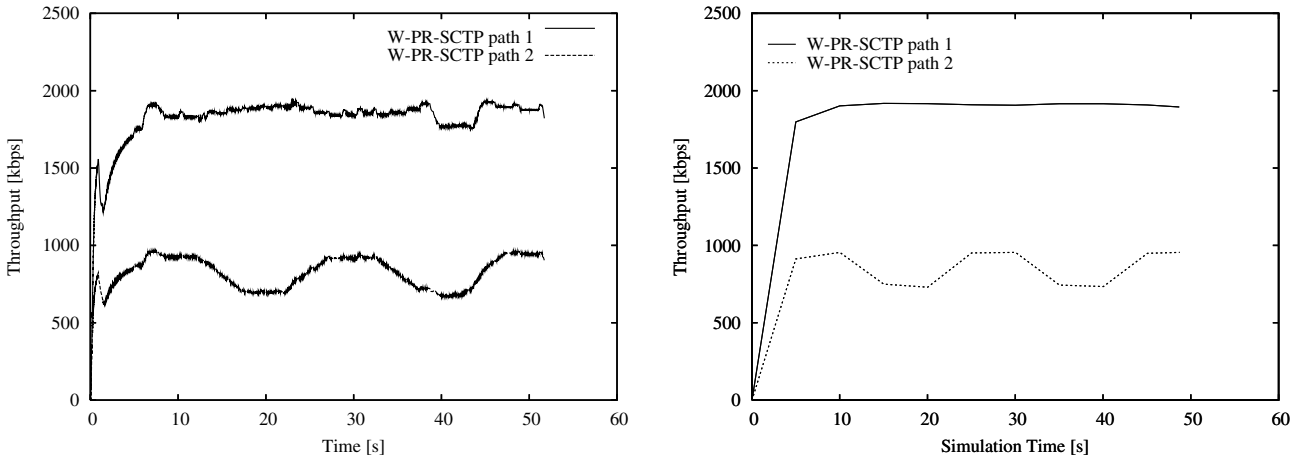


Fig. 20. Comparison between experimental (left) and simulation (right) results; intermittent 250-kbps UDP stream on path 2.

Next, we undertook the study of a dynamic scenario, where the UDP cross-traffic had an intermittent behavior, turning on and off every 10 s (Figs. 19 and 20), showing intermittent traffic on path 1 and path 2, respectively. The bit rate of the interfering UDP traffic was 500 kbps when affecting path 1 and 250 kbps when it ran over path 2. Again, the testbed results (on the left-hand side) were validated by simulation (on the right-hand side), and both confirmed the degree of adaptivity of W-PR-SCTP to changing traffic conditions. The slope of the instant throughput changes is not very steep because the points on the plot were averaged over the last 5 s, thus showing a slower transition rate.

6. Conclusions

This paper presented and discussed a novel approach to concurrent multipath communication, represented by the W-PR-SCTP transport protocol. W-PR-SCTP is capable of exploiting all the bandwidth available on multiple interfaces through a bandwidth estimation process and a sender-based scheduler that tries to predict the delivery time of each chunk of data. The effectiveness of the new protocol was investigated through extensive simulation in both static and dynamic scenarios. Also, a testbed implementation of W-PR-SCTP on Linux was described and simulation results were provided to validate its correctness.

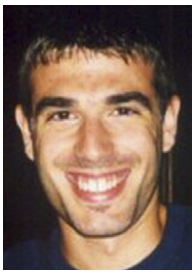
Acknowledgements

This work has been carried out in the framework of PRIMO, a project on reconfigurable platforms for wideband wireless communications partly funded by the Italian Ministry of University and Research (MIUR). The cooperation among the authors has been fostered by the Network of Excellence in Wireless Communications (NEWCOM), funded by the European Union in the Seventh Framework Program.

References

[1] R. Stewart, RFC 2960, Stream Control Transmission Protocol (SCTP) (October 2000).
[2] S. Kashihara, T. Nishiyama, K. Iida, H. Hoga, Y. Kadobayashi, S. Yamaguchi, Path selection using active measurement in multihomed

wireless networks, in: Proceedings of IEEE SAINT'04, Tokio, Japan, 2004, pp. 273–276.
[3] R. Fracchia, C. Casetti, C.-F. Chiasserini, M. Meo, A wise extension of sctp for wireless networks, in: Proceedings of IEEE ICC 2005, Seoul, S. Korea, 2005, pp. 1448–1453.
[4] L. Ma, F. Yu, V.C.M. Leung, A new method to support umts/wlan vertical handover using sctp, IEEE Wireless Communications 11 (4) (2004) 44–51.
[5] A. Mishra, M. Shin, W. Arbaugh, Concurrent multipath transfer using SCTP multihoming, in: ACM SIGCOMM Computer Communication Review, vol. 33, San Jose, CA, USA, 2003.
[6] J.R. Iyengar, K.C. Shah, P.D. Amer, R. Stewart, Concurrent multipath transfer using SCTP multihoming, in: Proceedings of SPECTS 2004, San Jose, CA, USA, 2004.
[7] J.R. Iyengar, P. Amer, R. Stewart, Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths, accepted for publication in IEEE/ACM Transactions on Networking 14 (5) (2006) 951–964.
[8] A.A.E. Al, T. Saadawi, M. Lee, LS-SCTP: a bandwidth aggregation technique for Stream Control Transmission Protocol, Computer Communications 27 (10) (2004) 1012–1024.
[9] G. Ye, T.N. Saadawi, M. Lee, IPCC-SCTP: an enhancement to the standard SCTP to support multi-homing efficiently, in: Proceedings of the IEEE International Conference on Performance, Computing, and Communications (ICPCC 2004), Phoenix, AZ, USA, 2004, pp. 523–530.
[10] M. Fiore, C. Casetti, An adaptive transport protocol for balanced multihoming of real-time traffic, in: Proceedings of IEEE GLOBECOM 2005, St. Louis, MO, USA, 2005, pp. 523–530.
[11] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, R. Wang, A transport layer approach for improving end-to-end performance and robustness using redundant paths, in: Proceedings of the Usenix Annual Technical Conference, Boston, MS, USA, 2004, pp. 99–112.
[12] R. Stewart, RFC 3758, Stream Control Transmission Protocol (SCTP – Partial Reliability extension) (Jan. 2002).
[13] S. Mascolo, L.A. Grieco, R. Ferorelli, P. Camarda, G. Piscitelli, Performance evaluation of Westwood+ TCP congestion control, Performance Evaluation 4 (55) (2004) 93–111.
[14] G.D. Marco, M. Longo, F. Postiglione, S. Loreto, A. Senatore, On some open issues of load sharing in SCTP, in: Proceedings of CCCT 2003, Orlando, FL, USA, 2003, pp. 304–309.
[15] F. Perotto, G. Galante, Westwood-SCTP-PR Linux kernel patch. <http://www.lipar.polito.it/wsctp-pr/>.
[16] FreeBSD. <http://www.freebsd.org/>.
[17] L. Rizzo, Dummynet. <http://info.iet.unipi.it/~luigi/ip_dummynet/>.
[18] N.R. Laboratory, MGEN. The Multi-Generator Toolset. <http://mgen.pf.itd.nrl.navy.mil/>.



Marco Fiore was born in Asti, Italy, in 1979. He received his MS degrees in Computer Science from University of Illinois at Chicago, on December 2003, and from Politecnico of Torino, on July 2004. From July 2004 to December 2004 he collaborated with the Electronics Department of Politecnico of Torino. Since January 2005 he has been a PhD student in the Telecommunication Group at Politecnico di Torino, under the supervision of Prof. Claudio Casetti. From September 2006, he is a visiting student at Rice University, Houston, TX, in the Networks Group led by Prof. Ed Knightly. His research interests include mobile and vehicular wireless communications, novel transport protocols solutions and wireless mesh networking.

Claudio Casetti (M'05) graduated in Electrical Engineering from Politecnico di Torino in 1992 and received his PhD in Electronic Engineering from the same institution in 1997. In 1995, he was a visiting scholar with the Networks Group of the University of Massachusetts, Amherst. In 2000, he was a visiting scholar with the Networking Group at UCLA. He is an Assistant Professor at the Dipartimento di Elettronica e Telecomunicazioni of Politecnico di Torino. He has coauthored more than 80 journal and conference papers in the fields of networking and holds three patents. His interests focus on performance evaluation of TCP/IP networks and wireless communications. He is a member of IEEE.

Giulio Galante obtained his Dr. Ing. degree in Electronics Engineering in 1998 and his PhD in Telecommunications Engineering in 2003 both from Politecnico di Torino.

During summer 2000, he was an intern with Lucent Technologies, Bell Labs, Holmdel, NJ. During 2001–2002, he visited the research group of Prof. Nick McKeown at Stanford University, CA. Currently he is with Istituto Superiore Mario Boella, Turin, Italy. His research is mainly focused on the design of vehicular and mesh networks, the performance evaluation of wireless extensions to the TCP/IP protocol suite, and switching architectures based on off-the-shelf hardware and open-source software.