Self-management of Hybrid Networks – Hidden Costs Due to TCP Performance Problems

Giovane C.M. Moura, Aiko Pras, Tiago Fioreze, and Pieter-Tjerk de Boer

University of Twente
Centre for Telematics and Information Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Design and Analysis of Communications Systems (DACS)
Enschede, The Netherlands
{g.c.m.moura,a.pras,t.fioreze,p.t.deboer}@utwente.nl

Abstract. Self-management is one of the most popular research topics in network and systems management. Little is known, however, regarding the costs, in particular with respect to performance, of self-management solutions. The goal of this paper is therefore to analyze such hidden performance costs. Our analysis will be performed within the context of a specific example, namely automatically moving elephant flows from the routed IP layer to optical light-paths (lambdas) in hybrid networks. The advantage of moving elephant flows to light-paths is that such flows will experience lower delays, lower jitter and lower loss, thus better Quality of Service (QoS), while reducing the load at the IP-level, which means that the remaining flows will also experience better QoS. The lower delay at the optical level may cause temporary reordering of packets, however, since the first packet over the light-path may arrive at the receiver side before the last routed IP packet has arrived. Such reordering may lead to short but severe performance problems at the TCP level. We systematically analyze under which conditions such TCP performance problems occur, and how severe these problems are. Although our conclusions are specific to self-management of hybrid networks, it demonstrates by means of an example that self-management solutions may also introduce new problems, which must further be investigated before conclusions can be drawn regarding the pros and cons of self-management.

1 Introduction

One of the most popular research topics within the network management community is *self-management* [1–5]. The key idea behind self-management is to develop computer systems that are able to manage their own operation without (or with little) human intervention. Although self-management is often seen as a solution for many problems, little is known about the hidden costs in terms of potential new problems that may result from self-management. Such problems may vary from case to case, and therefore need to be investigated within the context of specific examples.

T. Bauschert (Ed.): EUNICE 2013, LNCS 8115, pp. 54–65, 2013.

To get a better understanding of such hidden problems, this paper tackles the example of hybrid networks in which routed flows at the IP level can be moved to the optical level. In such networks, IP packets can be forwarded either through a chain of IP routers, or through an end-to-end lightpath. Such a "lightpath" uses optical switching technology, either at the level of entire fibers, a wavelength within a fiber (lambda), or a TDM-based channel (SONET/SDH) within a wavelength; in any case, switching delays will be smaller, and data rates may well be higher than at the IP level. Moving large flows from the routed IP level to a direct optical lightpath thus enables these flows to experience a faster and more reliable service, while at the same time reducing the load on the IP level equipment, which is typically much more expensive than lightpath equipment.

Current approaches to manage such hybrid networks mostly rely on human operators in order to: (*i*) select the flows to be moved to the optical level, and (*ii*) create and release the necessary lightpaths. Since these decisions can be slow and error prone, we envision a move towards self-management approaches, which automatically move IP flows to lightpaths on the fly [6–8]. Network operators would only be required to initially configure the self-management process with decision policies, such as setting thresholds and priorities. After this initial setup, the self-management process runs by itself. For more details, see [9].

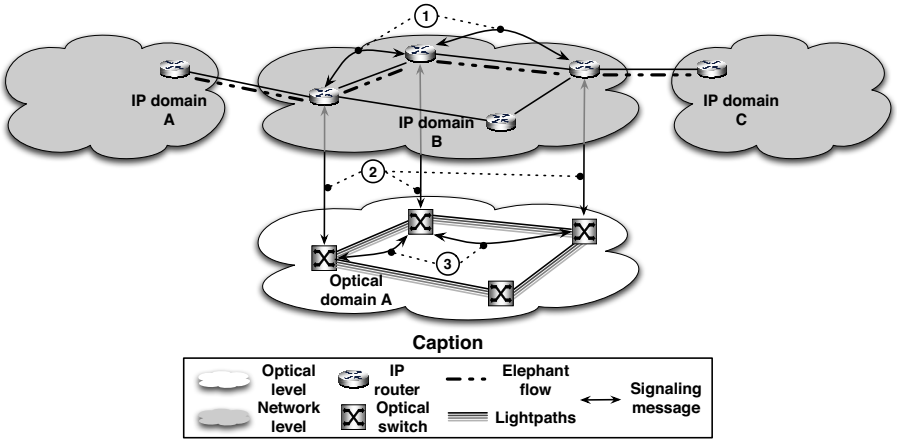


Fig. 1. Self-management of lightpaths in hybrid optical and packet networks: in step 1, the elephant flows are identified; in step 2 the optical switches are informed about the decision to move a flow; and in step 3, the optical switches establish the lightpath.

Self-management of hybrid networks can result in performance problems at the TCP level, however. In a manual process, a lightpath is established a-priori, before the start of the flow (note that within this paper the term "flow" relates to a single TCP connection). Since our self-management process cannot rely on human intelligence to identify a-priori which flows should be transferred over a lightpath, it should analyze existing flows at the IP level to find those flows that are worthwhile to be moved to the optical level. Flows are thus moved *on the fly*,

which means that the first packets of a flow may be forwarded at the IP level, whereas later packets can be moved to the lightpath. The lower delay at the optical level may cause temporary reordering of packets however, since the first packet over the lightpath may arrive at the receiver side before the last packet has arrived via the IP level. Under certain conditions, such reordering may lead to short, but severe performance problems at the TCP level.

The goal of this paper is to investigate the potential performance problems at the TCP level, to gain a better understanding of the potential hidden costs of self-managing hybrid networks. With 'cost', we do not necessarily mean monetary costs, but more generally problems and disadvantages (although those may well translate into monetary costs for a commercial network operator). Also, we will not look at the direct costs such as the costs of (re)configuring the equipment when moving a flow.

The approach taken in this paper is to first identify the factors that may limit the throughput of a TCP flow, and define four scenarios to cover these factors, thus extending a previous paper [10] in which we analyzed only one scenario. We will then use the *ns-2* network simulator [11] to investigate, for each scenario, the effect of moving TCP flows on the fly.

As noted above, the essence of the problem is the reordering of packets and TCP's reaction to it. Of course, packet reordering is not limited to moving flows to the optical domain, or to hybrid networks in general. It can also have many other causes, such as multipath routing, load balancing, and route changes. In fact, [12] shows that packet reordering is a very common occurrence in the Internet, and there have been many studies on its impact and potential countermeasures, such as [13, 14]. Our study is different in that the reordering affects a single very high rate flow, making the impact much larger than when the same amount of reordering is spread out over many smaller or lower rate flows. Also, in our case the reordering is intentional, and if its consequences are found to be too severe, it would be an argument against the use of the self-management system.

The paper structure is as follows. Section 2 discusses the factors that limit the TCP throughput, defines the four scenarios that cover these factors, and introduces the ns-2 simulation setup used. Following that, Section 3 presents the simulation results for all four scenarios. In Section 4, the insights gained for the four scenarios are generalized to other scenarios and transport protocols. Finally, Section 5 presents our conclusions.

2 Simulation Setup

This section discusses the simulation setup: we discuss the topology, the relevant thoughput limiting factors, the TCP variant to be used, and the simulator.

Network Topology. Figure 2 shows our simulation topology. It consists of three routers (`r1`, `r2`, and `r3`) and two nodes (`Sender` and `Receiver`) connected by two different paths: the IP path (`r1-r2-r3`) and the optical path (`r1-r3`).

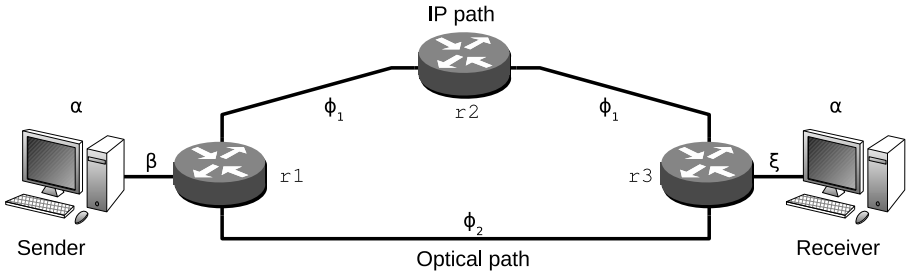


Fig. 2. Topology used in the simulations and limiting factors (Greek letters)

The simulation starts with the sender opening a single TCP connection to the receiver and sending data, forwarded via the IP path. After reaching a predefined throughput value, the decision is made to move that flow to the optical path: from that moment on, router `r1` will forward all subsequent data to the receiver via the optical path. Note that this only affects the data direction (`sender` → `receiver`), while the acknowledgment packets continue to use the IP path. For a short period of time after the switch – the *transient phase* – there will be data packets on both the IP and optical paths; after this phase, all data packets are on the optical path.

Simulation Scenarios. Although many factors can limit the throughput of TCP [15], since the scope of this paper is on self-management of network configurations, we focus on those factors that relate to the configuration of the network, namely the capacity of network links and the size of the TCP buffers. Thus, we have identified the following scenarios:

- **Scenario A**: the size of the TCP buffers (α in Figure 2) are the limiting factors for the TCP throughput.
- **Scenario B**: the capacity of the sender's local link (β in Figure 2) is the limiting factor for the TCP throughput.
- **Scenario C**: the capacity of the core links (ϕ_1 and ϕ_2 in Figure 2) acts as the limiting factor. We distinguish between the cases where the capacity of IP links is equal to that of the optical links ($\phi_1 = \phi_2$) and where the capacity of the optical links is larger than that of the IP links ($\phi_2 > \phi_1$).
- **Scenario D**: the receiver local link is the limiting factor (ξ in Figure 2).

For each scenario we simulate three limiting data rates: 100 Mbps, 1 Gbps, and 10 Gbps. In each simulation, the flow is moved to the optical path once the actual throughput reaches the limiting rate. We simulate three base RTT values, namely 10, 100 and 1000 ms, with corresponding optical path RTTs of 6, 60 and 600 ms, respectively. Table 1 summarizes the values of the parameters used in our simulation scenarios for RTT=10 ms; an equal number of simulations was conducted for 100 ms and 1000 ms RTT, with α in Scenario A scaled accordingly.

Table 1. Scenarios and values used for the limiting factors for RTT equal to 10 ms

Scenario	Limiting Rate	α (rtt=10ms)	β	ϕ_1	ϕ_2	ξ
A	100 Mbps	0.125 MB	622.08 Mbps	622.08 Mbps	622.08 Mbps	622.08 Mbps
	1 Gbps	1.25 MB	2.488 Gbps	2.488 Gbps	2.488 Gbps	2.488 Gbps
	10 Gbps	12.5 MB	39.813 Gbps	39.813 Gbps	39.813 Gbps	39.813 Gbps
B	100 Mbps	1.16 GB	100 Mbps	622.08 Mbps	622.08 Mbps	622.08 Mbps
	1 Gbps	1.16 GB	1 Gbps	2.488 Gbps	2.488 Gbps	2.488 Gbps
	10 Gbps	1.16 GB	10 Gbps	39.813 Gbps	39.813 Gbps	39.813 Gbps
C1	100 Mpbs	1.16 GB	622.08 Mbps	100Mbps	100Mbps	622.08Mbps
	1 Gbps	1.16 GB	2.488 Gbps	1 Gbps	1 Gbps	2.488 Gbps
	10 Gbps	1.16 GB	39.813 Gbps	10 Gbps	10 Gbps	39.813 Gbps
C2	100 Mbps	1.16 GB	622.08 Mbps	100 Mbps	622.08 Mbps	622.08 Mbps
	1 Gbps	1.16 GB	2.488 Gbps	1 Gbps	2.488 Gbps	2.488 Gbps
	10 Gbps	1.16 GB	39.813 Gbps	10 Gbps	39.813 Gbps	39.813 Gbps
D	100 Mbps	1.16 GB	622.08 Mbps	622.08 Mbps	622.08 Mbps	100 Mbps
	1 Gbps	1.16 GB	2.488 Gbps	2.488 Gbps	2.488 Gbps	1 Gbps
	10 Gbps	1.16 GB	39.813 Gbps	39.813 Gbps	39.813 Gbps	10 Gbps

TCP Version and Simulator. For this study, we decided to use TCP CUBIC (version 2.1) [16], since it is adapted to links with a large bandwidth-delay product (as is the case in our scenarios), and nowadays widely used because it is the default in Linux. For the simulations we use the ns-2 network simulator, version 2.33, which can run the actual TCP CUBIC code from Linux [17].

3 Simulation Results

In this section, we show some simulation results for all four scenarios, illustrating the various problems that may occur.

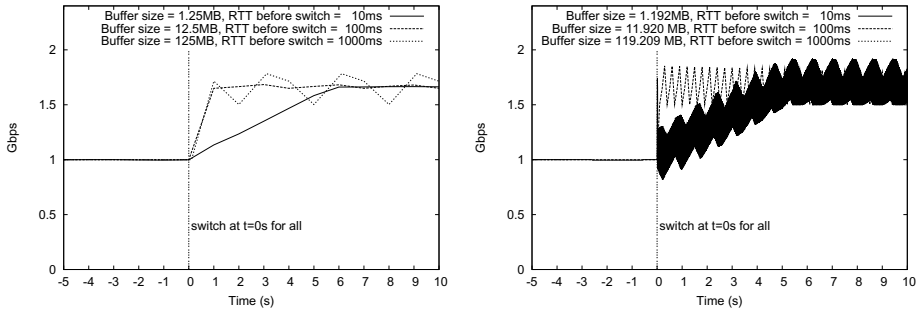
3.1 Scenario A: TCP Buffers as the Limiting Factor

In this scenario we configured the TCP buffers to act as the limiting factor of the TCP throughput, as discussed earlier in [10]. This is done by setting the other potentially limiting factors (link capacities β, ϕ, and ξ) high enough, as presented in Table 1. The TCP buffers were set equal to the bandwidth-delay product, calculated from the limiting flow rate (100 Mbps, 1 Gbps or 10 Gbps) and the RTT *before* the switchover.

For all cases of Scenario A, only one type of behavior was seen, independent of RTT and buffer sizes: after the switchover, the TCP throughput increases without any packet loss. Figures 3(a) and 3(b) show the throughput as a function of time for the 1 Gbps case, before and after the switch at $t = 0$ (vertical line).

These graphs differ in their "granularity", i.e., the time interval over which the data rate is averaged. This is a compromise between cluttering the picture at too small granularity, and masking interesting effects at coarse granularity. We henceforth show only graphs for a granularity of 1000 ms; if this hides any interesting effects, these are discussed in the text.

As can be seen, before switchover the flows present a stable rate of 1 Gbps, limited only by the TCP buffers. After the switchover, CUBIC's throughput increases quickly to the expected new theoretical value of 1.667 Gbps (i.e., the buffer size divided by the new RTT).



(a) Granularity = 1000 ms (b) Granularity = RTT (before switch)

Fig. 3. Throughput of TCP flows in Scenario A (1 Gbps limiting rate)

3.2 Scenario B – Sender's Local Link as the Limiting Factor

In Scenario B, the sender's local link is the limiting factor for TCP throughput. Intuitively, one might expect that in this case, the optical switch has no influence, since the bottleneck does not change. However, this is not true; Figure 4 shows that there is a brief decrease of the throughput immediately after the switch (solid line), compared to the case without switching (dotted line).

With a finer granularity, one can also see a *peak* of 1.3 Gbps in the first 0.1 s after the switch. This is due to packets arriving from both the IP and optical paths during this transient phase. The packets arriving via the optical path have a higher sequence number than the ones coming from the IP path, because of the optical path's lower delay. This causes the receiver to perform reordering and

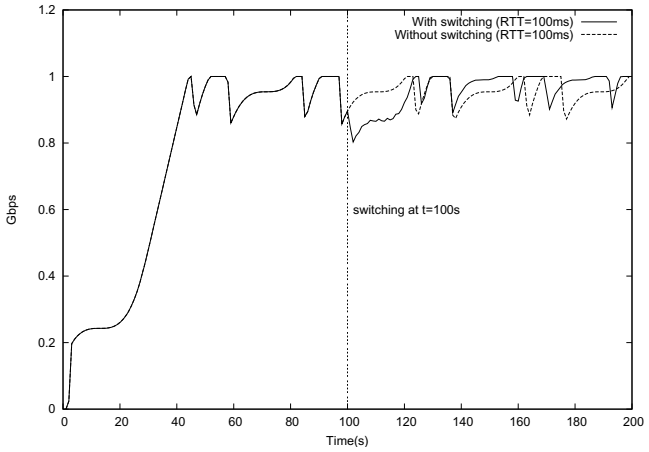


Fig. 4. Throughput of TCP flows in scenario B for RTT=100 ms, β = 1 Gbps and granularity equal to 1000 ms

ask for retransmissions (through a total of 14186 duplicate ACKs), resulting in the reduction of the congestion window and thus a throughput reduction. This can be seen at $t = 102$ s, where the throughput reaches 803 Mbps.

3.3 Scenario C – Core Link as the Limiting Factor

In this scenario, the core link is the limiting factor, and we distinguish two cases: C1 with the IP and optical links having the same capacity ($\phi_1 = \phi_2$), and C2 with the optical link being faster ($\phi_2 > \phi_1$). In scenario C1, the behaviour turns out essentially the same as in scenario B, so we focus on scenario C2.

In scenario C2, similar behavior was seen as in scenarios C1 and B during the transient phase: throughput oscillation. However, after this oscillation, the throughput increases significantly, using the optical path's higher capacity. This is shown in Figure 5 for a 10 Gbps IP link and a 39.813 Gbps optical link, at RTTs of 10 and 100 ms. In either case, we see a mostly linear[1] growth of the throughput until the IP link is fully utilized; then the switch to the optical domain is made, and mostly linear growth resumes until the optical link speed is reached.

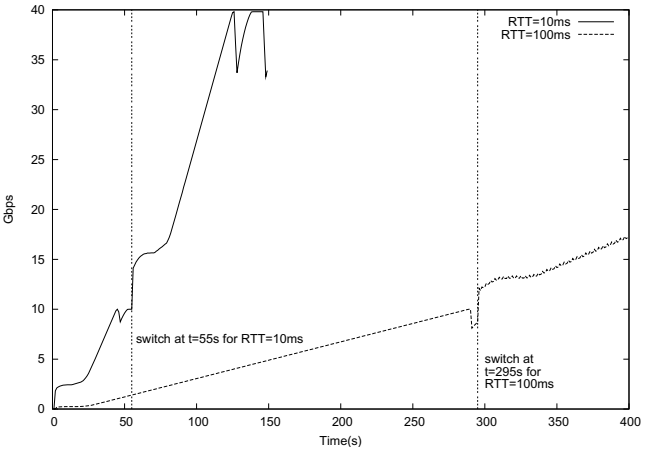


Fig. 5. Throughput of TCP flows in Scenario C2 (1000 ms granularity)

3.4 Scenario D – Receiver's Local Link as the Limiting Factor

Finally, we consider the case where the receiver's local link is the bottleneck and all other links are overdimensioned. Results for the 10 Gbps case are shown in Figure 6. As can be seen, there is a severe reduction in the throughput after the switchover.

[1] One might expect cubic rather than linear growth; however, version 2.1 of TCP CUBIC had an upper bound on the growth rate. This was removed in version 2.2 [16].

Like in the other scenarios, after the switchover router `r3` temporarily receives data via both the IP and the optical links simultaneously, for a total of 20 Gbps. However, unlike in the other scenarios, the link from this router to the destination was already fully loaded at 10 Gbps, and cannot cope with this temporary double data flow. Queueing all these packets is not possible either (in our simulation, `r3` has a 500 packet buffer), causing many packets to be dropped (335107 packets in the first 200 ms after the switchover in the 10ms RTT case). This in turn causes the receiver to send a massive number of duplicate ACKs, which leads the sender to reduce its congestion window, causing the throughput to drop to a minimum value of 1.6 Gbps.

Figure 7 shows how `r3`'s queue size changes in time. Since TCP CUBIC (like most other TCP versions) measures the available bandwidth on the path by increasing its congestion window until packet loss occurs, the queue also builds up at other moments than the switchover, for example at $t = 46s$. When the optical switchover occurs at $t = 55$ s, the queue already contains 343 packets due to TCP CUBIC probing for bandwidth. After the switchover, packets arrive from both paths, causing the queue to fill up very quickly and resulting in a massive discarding of packets, as discussed above. The queue increases at later times are again due to CUBIC's normal bandwidth probing.

Note that the problem seen here is fundamentally different from that seen in scenarios B and C. In either case, temporarily two streams of packets (namely via the IP path and via the optical path with less delay) are merged, causing reordering. However, in scenarios B and C, no packets were dropped, whereas in the current scenario many packets are dropped. This causes a much larger reduction in throughput, and recovery from this also takes much more time.

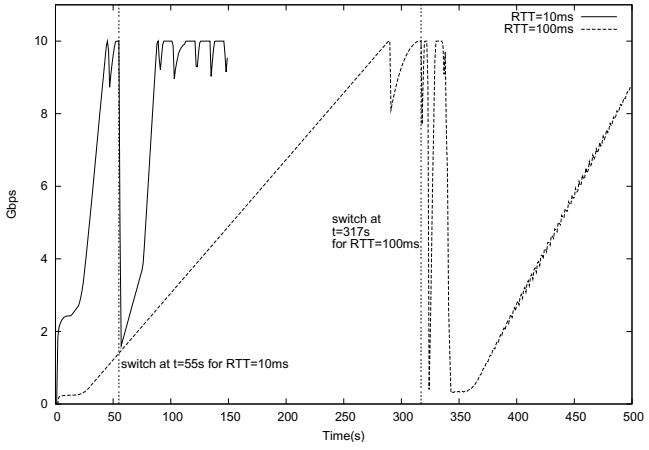


Fig. 6. Throughput of TCP flows for Scenario D (1000 ms granularity)

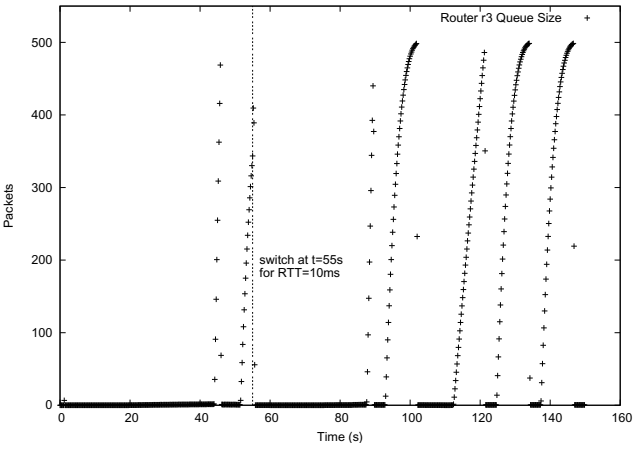


Fig. 7. Router `r3` Queue Size for 10Gbps case with RTT = 10ms

4 Discussion and Generalization

In the previous section, we have seen three types of behaviour when switching a TCP flow from the IP to the optical path:

- When the TCP buffers were the limiting factor (scenario A), nothing untoward happens: the system converges quickly to the new higher throughput made possible by the lower RTT.
- When the sender's link or the IP and optical paths are the bottleneck (scenarios B and C), temporarily packets from both paths arrive at the receiver, causing reordering but no loss. We have seen that TCP reacts to this rather benignly, with only a small ($< 20\%$) and short-lived reduction in throughput.
- When the receiver's link is the bottleneck (scenario D), it gets overloaded due to the temporary simultaneous arrival of packets via both paths. This causes much packet loss and a severe ($> 80\%$) reduction of throughput, from which recovery takes longer (thousands of RTTs).

These conclusions can easily be generalized to other situations, as follows.

Other TCP Variants. The choice for TCP CUBIC was already motivated in Section 2. Still, we have done similar experiments with three other TCP variants (Reno, Vegas, Compound). The results for Compound were similar to those for CUBIC reported here, sometimes recovering even faster.

Reno and Vegas performed far worse: due to their at most linear growth of the congestion window, they need much more time to recover. In fact, for the same reason they need much time to fully utilize link with a large delay bandwidth product even before the changeover, making them unsuitable for the kind of elephant flows considered here. Therefore, their problems with recovering from the optical switchover can be safely ignored.

Other Protocols. Although TCP is the dominant transport protocol in the internet, there are others, such as UDP, SCTP and DCCP. Clearly, after the switchover they will experience the same packet reordering and loss as TCP.

UDP does not by itself have a congestion control mechanism, so the reordering and loss will not directly affect its throughput. However, the reordering and loss will be passed on to the application, so detailed knowledge of the application would be needed to predict the effects for the user.

New transport protocols such as SCTP (Stream Control Transmission Protocol) [18]) and DCCP (Datagram Congestion Control Protocol) [19] have congestion control mechanisms that are directly based on TCP variants [20]. Therefore we can expect them to behave similarly to TCP, with possibly additional application layer effects like with UDP.

Other Scenarios. In the scenarios considered so far, the same parameter was the limiting factor before and after the switchover. Of course, in reality after the switchover a different parameter can be the limiting factor. What happens in those cases will generally be a mixture of the "pure" behaviours studied here.

Also the numerical values of parameters (link speeds etc.) can be different. Clearly, such changes will change the outcomes quantitatively (e.g., recovery will take longer or shorter), but no qualitative differences are to be expected.

One might hope that installing a large buffer at the output router would prevent the massive packet loss in scenario D, but this is not true. TCP's congestion control algorithm would have filled such a buffer (because TCP needs to detect loss to know when to back off), so there would not be space to temporarily accept the extra packets when the flow is moved to the lightpath.

5 Conclusions

In this paper we investigated hidden costs of self-management within the specific context of hybrid networks. In particular we analyzed the performance problems that may result if an elephant TCP flow at the IP level is moved on the fly to the optical level using ns-2 simulations for four different scenarios. During such move, a temporary but massive reordering of packets occurs since the first packets transferred over the optical level will arrive before the last packets arrive at the IP level.

We found three qualitatively different behaviours. In case the TCP throughput of the elephant flow was limited by the size of the TCP send and receive buffers, the throughput *increases* substantially due to the lower RTT of the optical path. In case the throughput was limited by the sender's local link or by the backbone links, TCP reacts benignly to the reordering, with only a small and short-lived reduction of throughput. On the other hand, if the throughput was limited by the receiver's local link, a *significant drop* in the throughput occurs for a relatively long period. In this case, the router at the receiver side needs to drop a large number of packets when they arrive simultaneously via both links, because its outgoing link was already fully used.

The significant throughput drop and long recovery period observed in the last scenario can be very noticable to the end user, and thus is something the operator of the network might want to avoid. However, whether or not this problem is going to occur depends on the bottleneck in the receiver's network, which may well be out of sight of the operator of the hybrid network, making this truly a *hidden* cost of implementing self-management.

The conclusion that can be drawn from the hybrid networks case presented in this paper, is that self-management mechanisms may introduce unexpected side-effects, which require further analysis before such self-management mechanisms can be widely adopted. Research on self-management mechanisms should therefore not be performed in isolation, but always consider the context in which these mechanisms will be applied, to fully understand their pros and cons.

Acknowledgments. The research reported in this paper is supported by the FP7 ICT UniverSelf project (#257513). The authors would like to thank Wouter Kooij for his work on extending the simulations presented in this paper to other TCP flavors – Vegas, Reno, and Compound.

References

1. Pras, A., Schönwälder, J., Burgess, M., Festor, O., Pérez, G., Stadler, R., Stiller, B.: Key Research Challenges in Network Management. IEEE Communications Magazine 45(10), 104–110 (2007)
2. Jennings, B., van der Meer, S., Balasubramaniam, S., Botvich, D., Foghlu, M., Donnelly, W., Strassner, J.: Towards Autonomic Management of Communications Networks. IEEE Communications Magazine 45(10), 112–121 (2007)
3. Lupu, E., Dulay, N., Sloman, M., Sventek, J., Heeps, S., Strowes, S., Twidle, K., Keoh, S.L., Schaeffer-Filho, A.: AMUSE: Autonomic Management of Ubiquitous e-Health Systems. Concurr. Comput.: Pract. Exper. 20(3), 277–295 (2008)
4. Cheng, L., Galis, A., Mathieu, B., Jean, K., Ocampo, R., Mamatas, L., Rubio-Loyola, J., Serrat, J., Berl, A., de Meer, H., Davy, S., Movahedi, Z., Lefevre, L.: Self-organising Management Overlays for Future Internet Services. In: van der Meer, S., Burgess, M., Denazis, S. (eds.) MACE 2008. LNCS, vol. 5276, pp. 74–89. Springer, Heidelberg (2008)
5. Derbel, H., Agoulmine, N., Salaün, M.: ANEMA: Autonomic Network Management Architecture to Support Self-configuration and Self-optimization in IP networks. Comput. Netw. 53(3), 418–430 (2009)
6. Fioreze, T., Granville, L., Sadre, R., Pras, A.: A Statistical Analysis of Network Parameters for the Self-management of Lambda-Connections. In: Sadre, R., Pras, A. (eds.) AIMS 2009 Enschede. LNCS, vol. 5637, pp. 15–27. Springer, Heidelberg (2009)
7. Fioreze, T., Pras, A.: Self-management of Lambda-connections in Optical Networks. In: Bandara, A.K., Burgess, M. (eds.) AIMS 2007. LNCS, vol. 4543, pp. 212–215. Springer, Heidelberg (2007)
8. Fioreze, T., van de Meent, R., Pras, A.: An Architecture for the Self-management of Lambda-Connections in Hybrid Networks. In: Pras, A., van Sinderen, M. (eds.) EUNICE 2007. LNCS, vol. 4606, pp. 141–148. Springer, Heidelberg (2007)

9. Fioreze, T.: Self-Management of Hybrid Optical and Packet Switching Networks. PhD thesis, Universiteit Twente (February 2010)
10. Moura, G.C.M., Fioreze, T., de Boer, P.-T., Pras, A.: Optical switching impact on TCP throughput limited by TCP buffers. In: Nunzi, G., Scoglio, C., Li, X. (eds.) IPOM 2009. LNCS, vol. 5843, pp. 161–166. Springer, Heidelberg (2009)
11. The Network Simulator NS-2, http://nsnam.isi.edu/nsnam/index.php/Main_Page
12. Bennett, J., Partridge, C., Shectman, N.: Packet reordering is not pathological network behavior. IEEE/ACM Transactions on Networking 7(6), 789–798 (1999)
13. Zhang, M., Karp, B., Floyd, S., Peterson, L.: RR-TCP: a reordering-robust TCP with DSACK. In: 11th IEEE International Conference on Network Protocols, pp. 95–106 (November 2003)
14. Karlsson, J., Hurtig, P., Brunstrom, A., Kassler, A., Di Stasi, G.: Impact of multipath routing on TCP performance. In: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–3 (June 2012)
15. Timmer, M., de Boer, P., Pras, A.: How to Identify the Speed Limiting Factor of a TCP Flow. In: 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services, pp. 17–24 (April 2006)
16. Ha, S., Rhee, I., Xu, L.: CUBIC: a new TCP-friendly high-speed TCP variant. SIGOPS Oper. Syst. Rev. 42(5), 64–74 (2008)
17. Wei, D.X., Cao, P.: NS-2 TCP-Linux: an NS-2 TCP Implementation with Congestion Control Algorithms from Linux. In: WNS2 2006: The 2006 Workshop on NS-2: The IP Network Simulator. ACM, New York (2006)
18. Steward, R. (ed.): Stream Control Transmission Protocol (SCTP). RFC 4960 (2007)
19. Kohler, E., Handley, M., Floyd, S.: Datagram Congestion Control Protocol (DCCP). RFC 4340 (2006)
20. Floyd, S., Kohler, E.: Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control. RFC 4341 (2006)