# Scalable Router Aided Congestion Avoidance for Bulk Data Transfer in High Speed Networks

Michael Welzl
Institute of Computer Science
Leopold-Franzens-University of Innsbruck
A-6020 Innsbruck, Austria
Email: michael.welzl@uibk.ac.at

***Abstract*— This paper justifies using explicit performance signaling for bulk data transfer by means of a simple yet efficient congestion control scheme called "Congestion Avoidance with Distributed Proportional Control (CADPC)". It thereby contradicts a common belief that no additional forward traffic should be sent by such mechanisms. CADPC, which solely relies on rare performance feedback from routers, was shown to outperform several TCP-friendly mechanisms and TCP "flavors" in various aspects in simulations; it is particularly well suited for fast long-distance networks due to its rapid convergence that does not depend on the RTT. One disadvantage of our scheme is its requirement for traffic isolation; on the other hand, it may be possible to deploy it without updating the software in routers.**

## I. Introduction

TCP causes congestion in order to avoid it. Since it bases its decisions on a binary congestion signal — traditionally packet loss, more recently the ECN flag — a sender can only react when it is already too late (or almost too late, as with ECN). This behavior repeatedly causes the bottleneck queue to grow, which leads to increased delay and eventually causes packets to be dropped. With its additive increase rate update policy, recovering from packet loss can take an unacceptably long time in the face of links with a high bandwidth $\times$ delay product. Therefore, it must be a goal to *avoid* congestion (keep queues low and prevent loss) while efficiently using the available bandwidth in order to obtain reasonable performance. To this end, an entirely different approach may be necessary.

We describe one such alternative: "Congestion Avoidance with Distributed Proportional Control" (CADPC). Our mechanism only uses rare performance feedback that can be attained by explicitly querying routers for the relevant data; since it does not depend on the loss ratio or the RTT, the (very simple) rate calculation can be performed by any network node which sees acknowledgments from the receiver. Also, other than TCP, RTT deviations due to link layer retransmissions and packet loss from link noise do not hinder the convergence of the mechanism. Our scheme has a number of additional advantages:

- it quickly reaches a stable state instead of a fluctuating equilibrium
- it has a smooth rate
- it showed greater throughput than TCP and several TCP-friendly mechanisms, almost no loss and a small queue length over a wide range of parameters in simulations
- it realizes precise max-min fairness in a fully distributed manner
- while it requires an occasional forward signaling packet, it generates a significantly smaller number of feedback messages from the receiver to the sender than TCP[1] — leading to better behavior across highly asymmetric links

In the aforementioned simulations, signaling was carried out with the *Performance Transparency Protocol (PTP)*, which we describe in the next section together with CADPC. Results as well as an in-depth description of both CADPC and PTP can be found in [1]; this paper complements this work in that it focuses on its applicability to fast long-distance networks (section 3), explains a potential scenario for relatively easy deployment (section 4) and provides a brief comparison with XCP (section 5) [2].

## II. Mechanism description

### A. PTP

PTP is designed to efficiently retrieve any kind of performance related information (so-called "performance parameters" — these could be the average bottleneck queue length, the "Maximum Transfer Unit (MTU)" of the path or the maximum expected bit error ratio, for example) from the network. It is scalable and lightweight: the code to be executed in routers is reduced to the absolute minimum and does not involve any per-flow state — all calculations are done at end nodes. In order to facilitate packet detection, the protocol is layered on top of IP and uses the "Router Alert" option [3]. In *Forward Packet Stamping* mode, PTP packets carrying information requests are sent from the source to the destination and updated by intermediate routers. The receiver builds a table of router entries, detects the relevant information and feeds it back to the sender. PTP packets must not be fragmented and will not exceed the standard 576 byte fragmentation limit on typical Internet paths.

CADPC requires information regarding the available bandwidth in the network. With PTP, this means that intermediate routers have to add the following information:

- The address of the network interface

[1]This may seem like an unfair comparison because TCP requires this signaling in order to realize reliability. Yet, without it, the congestion control behavior of TCP would also be entirely different.

- A timestamp
- The nominal link bandwidth (the "ifSpeed" object from the "Management Information Base (MIB)" of the router [4])
- A byte counter (the "ifOutOctets" or "ifInOctets" object from the MIB of the router [4])

At the receiver, two consecutive such packets are required to calculate the bandwidth that was available during the period between the two packets. Then, the nominal bandwidth $\mathcal{B}$, the traffic $\lambda$ and the interval $\delta$ of the dataset which has the smallest available bandwidth are fed back to the sender, where they are used by the congestion control mechanism.[2]

A complete specification of the PTP protocol with all its features is beyond the scope of this paper; please consult [1] or [5] for further details.

### B. CADPC

CADPC is a distributed variant of the "Congestion Avoidance with Proportional Control" (CAPC) ATM ABR switch mechanism by Andrew Barnhart [6]. Roughly, while CAPC has the rate increase or decrease proportional to the amount by which the total network traffic is below or above a predefined "target rate", CADPC does the same with the relationship between the rate of a user and the available bandwidth in the network. In mathematical terms, the rate $x(t+1)$ of a sender is calculated as

$$x(t+1) = x(t)\Big(2 - \frac{x(t)}{\mathcal{B}(t)} - \frac{\lambda(t)}{\mathcal{B}(t)}\Big) \tag{1}$$

whenever new feedback arrives. The old rate, $x(t)$, can be arbitrarily small but it must not reach zero.

Assuming a fluid-flow model, synchronous RTTs and $n$ users, equation 1 becomes

$$x(t+1) = x(t)\big(2 - x(t) - nx(t)\big) \tag{2}$$

(normalized to $\mathcal{B} = 1$), which is a form of logistic growth and is known to have an unstable equilibrium point at $\overline{x} = 0$ (hence the rule that the rate must not reach zero) and an asymptotically stable equilibrium point at $\overline{x} = 1/(1+n)$ [7]. Thus, the total traffic in the network converges to

$$n\overline{x} = \frac{n}{1+n} \tag{3}$$

which rapidly converges to 1 as the number of users increases. We decided that this convergence behavior is acceptable because it is very unlikely that only, say, two or three users would share a high capacity link, and eqn. 3 already yields a value that is very close to the bottleneck capacity for 10 or more users. In fact, this may be a more "natural" way of allocating bandwidth, as users tend to be very dissatisfied if their experience is degraded significantly. This can happen with a protocol that always saturates the bottleneck capacity when new users enter the network.

[2]Actually, $\delta$ is not used by the mechanism described in this paper; it is included in the feedback for the sake of completeness as these three parameters fully characterize the dataset.

| *Parameter* | *Value* |
|---|---|
| Topology | Dumbbell |
| Packet Sizes | 1000 |
| Bottleneck link bandwidth | 10 Mbit/s |
| Bandwidth of all other links | 1000 Mbit/s |
| Link Delay | 50 ms each |
| Queuing Discipline | Drop-Tail |
| Duration | long-term: 160 seconds |
| All flows start after | 0 seconds |
| Flow type | greedy, long-lived |
| Number of flows | short-term: 10, long-term: 100 |
| CADPC update frequency | 4 RTTs |
| CADPC smoothness factor $a$ | 0.5 |
| CADPC startup enhancement | active |
| TCP "flavor" | TCP Reno |
| ECN | active for all TCP "flavors" |

TABLE I
DEFAULT SIMULATION PARAMETER VALUES

Simulation results indicate that the stability of the control remains intact in the face of asynchronous RTTs; in fact, its convergence does not depend upon the RTT at all. In order to speed up the mechanism at the beginning of a connection (similar to "Slow Start" in TCP), an alteration of the rate update rule that temporarily allows a flow to be slightly more aggressive was designed; a detailed description of this change can be found in [1].

## III. SIMULATION RESULTS

The Performance Transparency Protocol was implemented for Linux (both configured as a router and as an end system) for simple functionality tests; an update to the latest kernel version and a Linux implementation of CADPC are currently in the works. So far, CADPC/PTP performance evaluations were only carried out using the *ns* network simulator[3]; except otherwise noted, the simulation parameters were always set according to table I.

Two simulation examples are shown in fig. 1: the diagrams depict the total transferred bytes of 10 CADPC and 10 TCP flows (only flows of one kind at a time, two simulations per scenario) sharing a 10 Mbit/s and a 100 Mbit/s bottleneck link, respectively. Clearly, the rate of CADPC is smoother than the rate of TCP. Also, in the case of TCP, the speed of convergence and bandwidth utilization decreases in the case of higher link capacities whereas CADPC is always capable of rapidly converging to its steady rate. There is a clear advantage of using CADPC in the face of high-speed links.

The diagram on the left-hand side of figure 2 shows the startup behavior of 10 flows sharing a single bottleneck; they rapidly converge to the same value. While equation 1 yields perfectly smooth convergence to the same rate, the startup time of a new flow that implements CADPC as described in the previous section and enters a system which already contains a significant amount of traffic is too long for realistic conditions. Therefore, in the style of the Slow Start algorithm in TCP, a startup mechanism was added to CADPC. Roughly, the idea is

[3]Linux and *ns* code is available from http://www.welzl.at/ptp

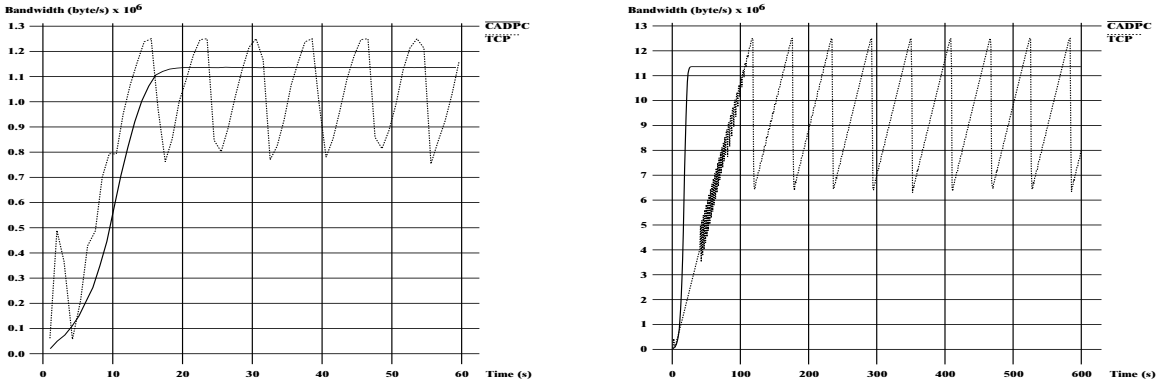


Fig. 1. CADPC vs. TCP with a 10 Mbit/s (left) and a 100 Mbit/s bottleneck (right)

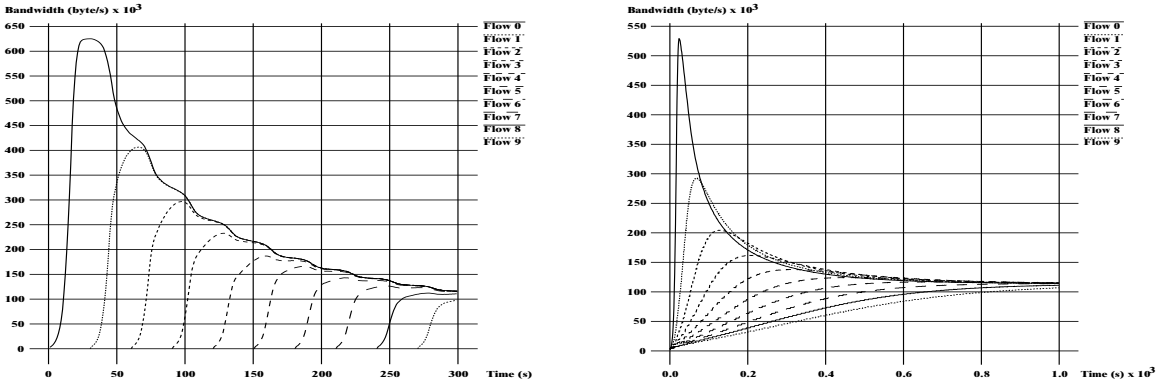


Fig. 2. Staggered startup (left) and fairness with $RTT_i = (i+1) \times RTT_0$ (right)

to act like a larger number of flows at the very beginning and have this behaviour linearly converge to the standard behaviour of a single flow as the rate approaches the final converged rate. This simple scheme does not endanger stability of the system, and it was used in all simulations shown in this paper and [1].

CADPC attains a max-min fair rate allocation, which means that it is easy to apply proper weights and thereby realize a QoS scheme based upon the mechanism: in a max-min fair allocation, two flows utilize exactly twice as much bandwidth than one flow, and therefore, a flow that behaves like two flows (because a QoS scheme allowed it to do so) attains twice the bandwidth of all others. The diagram on the right-hand side of figure 2 indicates that this fairness measure is indeed achieved: it shows the rates of ten CADPC flows, where the RTT of each flow is a multiple of the first flow. This was realized by changing the rate update frequency from 4 RTTs to multiples thereof (8, 12, 16, ..) from one flow to another. Thus, this diagram also shows that losing a signaling packet does not change the point of convergence. Our mechanism also correctly converged to max-min fairness in a simulation with multiple bottlenecks; this is documented in [1].

In order to evaluate the long-term performance of CADPC, its behavior was compared with TCP Reno, TCP SACK and TCP Vegas (each combined with ECN for scenarios with active queue management) and several TCP-friendly congestion control protocols in simulations — RAP, GAIMD with $\alpha = 0.31$ and $\beta = 7/8$ (which was realized by tuning RAP parameters), and the slowly-responsive TFRC and TEAR [8]. Flows of one particular kind at a time shared a single bottleneck link in a "dumbbell" topology. An excerpt of our result is shown in figure 3. Here, the simulation time was 160 seconds and link delays were 50 ms each, which, neglecting queuing delays, accounts for a RTT of 300 ms — thus, a simulation took approximately 500 RTTs when the network was lightly loaded. As indicated in table I, all flows were started at the same time; this could have caused a traffic phase effect [9], leading to the strange outliers of TCP Vegas. PTP was used in a mode where packets have an initial size of 32 bytes and grow by 16 bytes at each router. Only the payload flow is shown in the diagrams.

The behavior of CADPC shown in fig. 3 underlines a fundamental property of the mechanism: it is good at rapidly utilizing the available bandwidth and thus, its throughput increases linearly with the capacity. Packet loss of CADPC was negligible in all simulations, and fairness was almost perfect. Interestingly, CADPC maintains a small but constant queue length independent of the bottleneck bandwidth; this potential problem might be corrected by using a smaller target rate (normalizing CADPC with a smaller $\mathcal{B}$) depending on the amount of background traffic or the level of statistical multiplexing, which could be estimated by monitoring traffic changes.

CADPC always outperformed its competitors in our simu-

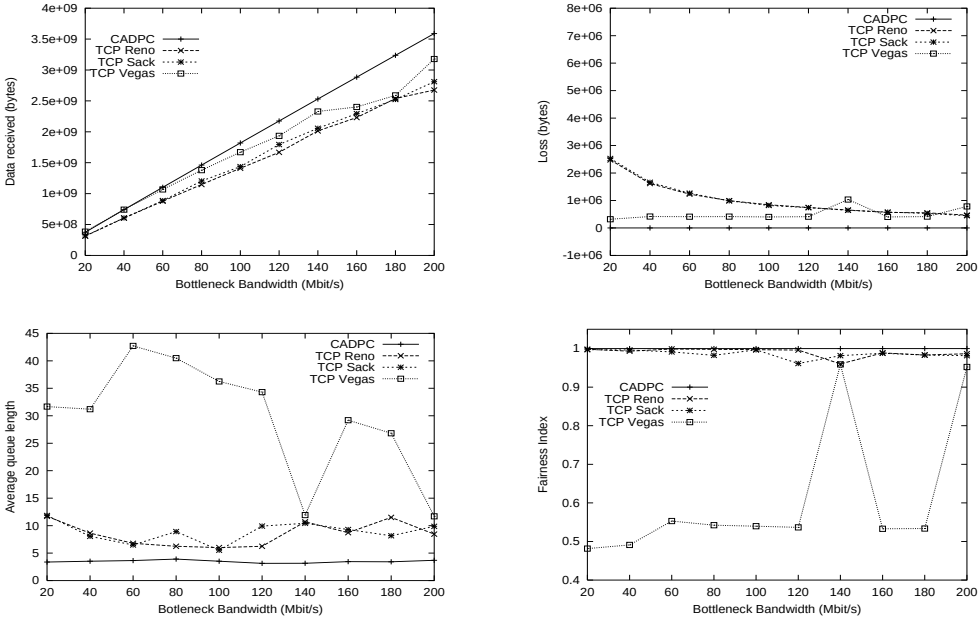


Fig. 3. CADPC and TCP, 50 fbws, varying bottleneck

lations: it achieved greater throughput than almost all other mechanism while maintaining a very low loss ratio. Also, while CADPC is the only mechanism that does not acknowledge every packet, it almost always showed the smallest average queue size — it should be no surprise that CADPC also worked very well in simulations with active queue management, which we did not include due to space restraints. Interestingly, CADPC achieves these results based on a single PTP packet that is sent every 4 RTTs while all other mechanisms acknowledge each and every packet that reaches the receiver.

In addition to the shown examples, the following scenarios were simulated (here, "performance" is used synonymously for throughput, loss, average queue length and fairness in comparison with TCP Reno, Sack and Vegas, RAP, GAIMD, TFRC and TEAR) and led to good results:

- Robustness against fast path changes
- Mixing of converged flows
- Convergence to "max-min-fairness" with two different bottlenecks ("parking lot" topology)
- performance as a function of the bottleneck bandwidth
- performance as a function of end-to-end delay
- performance with the active queue management mechanisms *RED*, *REM* and *AVQ*

We also examined the performance with varying web-like background traffic; since such traffic is much more dynamic than CADPC by nature, the mechanism cannot react in time. Therefore, this was the only scenario that led to unsatisfactory results.

## IV. Deployment considerations

Realizing a CADPC/PTP based service in a real, operational network is a difficult task because of two specific problems: first, PTP requires explicit support from at least every other router along the path. This means that the software in routers must be changed. While this is a minor issue when Linux routers are used, it is quite troublesome in a real network, where such devices are often hardware boxes that are not programmable. Often, only a firmware update can achieve the desired effect. The second problem with CADPC/PTP is its requirement for traffic isolation. In what follows, we will discuss possibilities for eliminating both these problems.

### A. Replacing PTP with SNMP

Except for a timestamp, PTP solely accesses data that are available in the Management Information Base (MIB) of routers when it is used in support of CADPC. A PTP packet collects information from all routers along a path: like a "Resource Management Cell" in ATM ABR, it is updated on-the-fly as it moves from the source to the destination. This is a more reasonable way to obtain the data than via, say, SNMP [10], because it is faster and generates less signaling traffic. Using PTP however means that routers must know and support the protocol – the possibility of replacing it with SNMP is therefore worth considering.

Retrieval of all the necessary information listed in section II-A from all routers along the path would essentially mean that the sender carries out the following procedure:

- Do a traceroute to the receiver
- Query all routers in the traceroute result list with SNMP and store the results
- If there are no earlier results available (i.e. this is the first measurement), wait for a while, then query again and store the new data
- Calculate the bandwidth that was available at each router between the two measurements

- Using the dataset with the minimum available bandwidth, feed back the link capacity $\mathcal{B}$ and the traffic $\lambda$ between the second and first measurement to CADPC

Notably, this method does not even require any support from the receiver. While the SNMP based deployment approach could theoretically be feasible, there are quite a number of operational limitations to consider:

- While SNMP access does not require the software in routers to be updated, it is at least necessary for administrators to grant MIB read access to the individuals who use CADPC. This might or might not be a reasonable assumption, depending on the specific usage scenario. In the context of a computational Grid, service providers often participate in the effort by providing specific resources and manpower (e.g., when they are partners of a funded research project). Then, asking for read access may be reasonable. A thorough discussion of reasons why service providers should not regard these data as secrets can be found in [1].
- Doing a traceroute does not guarantee that all the necessary router (or actually *interface*) addresses along a path are detected. PTP is designed to compensate for single intermediate routers that do not support the protocol by also taking information regarding the incoming interface into account. If, however, a router does not show up in the traceroute for the SNMP based approach, there is not much one can do other than asking the service provider to change the configuration of the router. This is not entirely unrealistic because there needs to be personal support from the ISP's side anyway.
- When designing a new protocol, one can simply prescribe certain router behavior. The PTP specification mandates that the values of the highly dynamic MIB object ifOutOctets reflect the present state of the hardware counters when a PTP packet is updated. When router software remains unaltered, setting up such a rule may not be feasible. It is therefore possible that the ifOutOctets object in the MIB is only rarely updated (e.g., once per second). Depending on the router software, this may or may not be the case, and it may or may not be possible to configure the update interval. Once again, communication with the ISP is probably necessary – if possible, routers should be configured to update these values very often, or (ideally) whenever a new request comes in. If this is not possible, CADPC is potentially fed imprecise traffic data.
- Time values in MIBs are often very coarse grained (updated every second). It may therefore be better to use the local time at the sender as a timestamp value – but then, the potentially fluctuating delay between the submission of an SNMP request and the moment it reaches the router comes into play, potentially rendering the results imprecise.

To conclude, the SNMP based approach suffers from two basic deficiencies: it is inevitable to obtain some explicit support from the service providers along the path, and even then, the data that are retrieved may be somewhat imprecise. The method may still be feasible because the robustness and update time independence of CADPC could compensate for the latter issue.

### *B. Isolating traffic*

When flows are used in the public Internet, they should be fair towards TCP; this requirement is known by the name *TCP-friendliness* or *TCP-compatibility*. It is defined in [11] as follows:

> A TCP-compatible flow is responsive to congestion notification, and in steady-state it uses no more bandwidth than a conformant TCP running under comparable conditions (drop rate, RTT, MTU, etc.)

When the intention is to design a mechanism which performs better than legacy TCP across fast long-distance networks, not using more bandwidth than TCP is exactly what one does *not* desire. The very idea of coming up with such new developments that outperform the protocol is at odds with this definition. Therefore, other deployment possibilities must be sought for CADPC than simply degrading its performance to that of a TCP-friendly flow (although the idea of incorporating additional packet loss feedback and behaving TCP-friendly *in the presence of TCP* might be worth considering[4]).

Rather than changing the CADPC mechanism itself in order to make it applicable in a real world scenario, we suggest to change the environment. In its current form, CADPC is a mechanism which only works well for long-lived and greedy traffic such as bulk data transfer. It is not TCP-friendly, and, as mentioned earlier, it turned out to be a poor match for highly fluctuating web traffic due to its rare feedback. It would therefore be best to recommend its usage for such traffic only and isolate all CADPC flows from everything else in the network. This can theoretically be attained by placing CADPC flows in a DiffServ class and prescribing that no other flows are allowed to become part of this traffic aggregate. Further, it must somehow be ensured that these flows are long-lived and greedy.

Since the behaviour of CADPC with a certain number of flows is deterministic and its converged rate can easily be calculated in advance, it makes sense to regard a DiffServ aggregate comprising nothing but CADPC flows as a fine-grain QoS scheme [12]. Individual flows can be weighted by allowing them to act like a fixed number of flows because of the convergence to a max-min fair rate allocation. In order to realize meaningful QoS based upon CADPC, a certain degree of additional control is necessary, i.e. admission control is inevitable. When flows are allowed into the system and who is given how much bandwidth must be negotiated between end systems and a bandwidth broker, which could be realized as a distributed information system that does not involve routers.

[4]In [1], several additional possibilities for future research on CADPC/PTP deployment are outlined; here, we only focus on what we believe to be the most realistic one.

| *Property* | *XCP* | *CADPC/PTP* |
|---|---|---|
| Feedback | in payload packets | extra signaling |
| Routers examine... | every packet | 1 packet every 4 RTTs |
| Convergence to... | bottleneck capacity | capacity*users/(users+1) |
| "web mice" bg traffic | works well | does not work well |

TABLE II
DIFFERENCES BETWEEN XCP AND CADPC/PTP

The only additional router function necessary for this to work is policy based admission control in edge routers.

Once again, computational Grids appear to be a good match here, as Grid applications typically cause two types of traffic — short, sporadic procedure ("Grid Service") calls and bulk data transfer. Since transmitting large amounts of data as part of a Grid service call is known to be highly inefficient, it is common to use a special service (often the *GridFTP* element of the well-known *Globus* middleware) [13] for this purpose. In other words, programmers of Grid applications are already used to distinguishing between these two types of services. In such a setting, it seems feasible to provide a CADPC communication service for bulk data transfer only and require the related QoS negotiation to be carried out via Grid Service calls. Developing such a system is intended as future research.

## V. CONCLUSION

In this paper, we have shown that CADPC/PTP is a good match for fast long-distance networks because of its ability to efficiently utilize a high-speed link as well as its quick convergence and RTT independence. It is by no means the only mechanism that was designed for such environments. One particularly noteworthy alternative is XCP, which resembles CADPC/PTP in several aspects: both quickly saturate the available capacity, converge to max-min fairness (which facilitates applying weights at the edges in order to realize service differentiation) and require router support. Also, both mechanisms require their traffic to be isolated from TCP flows (reference [2] describes a method to separate flows via two different queues in routers in order to achieve TCP-friendliness).

Table II shows some major differences between the two mechanisms. Briefly put, we believe that CADPC/PTP imposes less burden on routers at the cost of responsiveness; clearly, XCP is a better choice in highly dynamic environments. CADPC, on the other hand, could theoretically also utilize SNMP to retrieve the relevant information. Such an implementation would be slightly less efficient, but it might be possible to deploy it without installing a software update in routers. The CADPC/PTP requirement for long-lived flows seems to make the mechanism a particularly good match for bulk data transfers in computational Grids; we are currently investigating such usage scenarios, where we intend to isolate CADPC/PTP traffic from the rest by exclusively placing them in a DiffServ class.

## ACKNOWLEDGMENT

The author would like to thank his Ph.D. thesis supervisors Max Mühlhäuser and Jon Crowcroft for their continuous support.

The following figures were taken from [1] (copyright 2003 by Kluwer Academic Publishers) with kind permission of Springer Science and Business Media:

- Figure 1, which is a part of figure 4.10 on page 111 (chapter 4) of [1].
- Figure 3, which is figure 4.23 on page 122 (chapter 4) of [1].
- The left diagram in figure 2, which is a part of figure 4.9 on page 109 (chapter 4) of [1].
- The right diagram in figure 2, which is a part of figure 4.12 on page 113 (chapter 4) of [1].

## REFERENCES

[1] M. Welzl, *Scalable Performance Signalling and Congestion Avoidance*. Kluwer Academic Publishers, August 2003.

[2] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2002, pp. 89–102.

[3] D. Katz, "IP router alert option," Internet Engineering Task Force, RFC 2113, Feb. 1997. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2113.txt

[4] K. McCloghrie and M. T. Rose, "Management information base for network management of TCP/IP-based internets:MIB-II," Internet Engineering Task Force, RFC 1213, Mar. 1991. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1213.txt

[5] M. Welzl, "The performance transparency protocol (ptp), internet-draft draft-welzl-ptp-05, available from http://www.welzl.at/ptp," June 2001.

[6] A. W. Barnhart, "Explicit rate performance evaluations," ATM Forum Technical Committee, Traffic Management Working Group, Tech. Rep. Contribution ATM Forum/94-0983, October 1994.

[7] D. G. Luenberger, *Introduction to Dynamic Systems*. John Wiley & Sons, 1979.

[8] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *Special Issue of the IEEE Network Magazine "Control of Best Effort Traffic"*, vol. 15, no. 3, pp. 28 – 37, May/June 2001.

[9] S. Floyd and V. Jacobson, "Traffic phase effects in packet-switched gateways," *SIGCOMM Comput. Commun. Rev.*, vol. 21, no. 2, pp. 26–42, 1991.

[10] J. D. Case, M. S. Fedor, M. L. Schoffstall, and C. Davin, "Simple network management protocol (SNMP)," Internet Engineering Task Force, RFC 1157, May 1990. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1157.txt

[11] B. Braden, D. D. Clark, J. Crowcroft, B. S. Davie, S. E. Deering, D. Estrin, S. Floyd, and V. Jacobson, "Recommendations on queue management and congestion avoidance in the Internet," Internet Engineering Task Force, RFC 2309, Apr. 1998. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2309.txt

[12] M. Welzl and M. Mühlhäuser, "Scalability and Quality of Service: A Trade-off?" *IEEE Communications Magazine*, vol. 41, no. 6, pp. 32–36, June 2003.

[13] T. G. toolkit, "http://www-unix.globus.org/toolkit/."