Router Aided Congestion Avoidance with Scalable Performance Signaling

Michael Welzl

Institute of Computer Science, University of Innsbruck, Austria

Abstract. This paper justifies using explicit performance signaling in support of congestion control by means of a simple yet efficient 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 bandwidth feedback from routers, is shown to outperform several TCP-friendly mechanisms and TCP "flavors" in various aspects in simulations; its robustness against RTT fluctuations and loss from link noise makes it particularly well suited for heterogeneous network scenarios.

1 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. Therefore, it must be a goal to *avoid* congestion (keep queues small and prevent loss) while efficiently using the available bandwidth in order to obtain good performance. To this end, an entirely different approach may be necessary.

We present one such alternative: "Congestion Avoidance with Distributed Proportional Control" (CADPC). Our mechanism only uses rare bandwidth querying packets to calculate the rate and needs no other feedback. Since it does not depend on the loss ratio or the round-trip time (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, 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, leading to better behavior across highly asymmetric links

Signaling is carried out with the *Performance Transparency Protocol (PTP)*, which resembles ATM ABR explicit rate feedback, but 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. PTP is briefly explained in the next section (for an in-depth description of the protocol, the reader is referred to [1] and [2]). In section 3, we motivate the design of the endpoint control law and show that, assuming a fluid model and equal RTTs, our mechanism converges to an asymptotically stable equilibrium point. We present simulation results in section 4 and describe related and future work in section 5; section 6 concludes.

2 The Performance Transparency Protocol (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. 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 needs 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
- A timestamp
- The nominal link bandwidth (the "ifSpeed" object from the "Management Information Base (MIB)" of the router)
- A byte counter (the "ifOutOctets" or "ifInOctets" object from the MIB of the router)

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 λ and the interval δ of the dataset which has the smallest available bandwidth are fed back to the sender, where they are used by the congestion control mechanism.

3 Congestion Avoidance with Distributed Proportional Control (CADPC)

3.1 Fluid flow model design

CADPC is a distributed variant of the "Congestion Avoidance with Proportional Control" (CAPC) ATM ABR switch mechanism by Andrew Barnhart

[4]. 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 $\overline{x} = 1/(1+n)$ [5]. 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 very high values for 10 or more users. In fact, this may be a more "natural" way of allocating bandwidth; users tend to be very dissatisfied if their performance is degraded significantly due to new users entering the network. CADPC can also be expected to properly adapt to background traffic: if we assume constant background load b, the measured traffic becomes $nx(t) + b$, which yields the equilibrium point $(1-b)/(1+n)$ for the rate of a user. The total traffic then converges to $1 - b$ as the number of users grows.

Simulation results indicate that the stability property of the control remains intact in the face of asynchronous RTTs; in fact, its convergence does not depend upon the RTT at all, rendering the mechanism robust against a broad range of potentially harmful effects in heterogeneous environments (e.g., delay spikes from link layer ARQ). In order to speed up the behavior at the beginning of a connection (similar to "Slow Start" in TCP), an alteration of the rate update rule which temporarily allows a flow to be slightly more aggressive was designed; further details can be found in [1].

3.2 Discrete model design

In the case of a real network with packets and queues, some additional details need to be taken into account. Two of them quickly became evident in early simulations; they are i) the rate update frequency and ii) operational limitations due to packet sizes.

Update frequency Receiving and reacting to feedback more often than every RTT (as calculated from the departure and arrival times of PTP packets) causes oscillations. This is a natural phenomenon: as a fundamental principle of control, one should only use feedback which shows the influence of one's actions to control a system [6]. Since CADPC differs from TCP in that it reacts more drastically to precise traffic information, increasing the rate earlier can cause the traffic to reach a level where the control law tells the source that it should back off and vice versa. RTTs fluctuate — thus, the update frequency should be chosen large enough to ensure that it is larger than a RTT; as a general rule of thumb, four times the RTT is such a value.

As an alternative, it may seem to be a good idea to calculate the rate update frequency similar to the TCP retransmit timeout, which uses an exponentially weighted moving average process; this function appropriately weighs the most recent measurement against the history of measurements. However, in the case of CADPC, the situation is different because measurements are only received at the approximate rate of TCP retransmit timeouts — thus, the most recent measurement is much more important. In simulation experiments, using the TCP timeout calculation led to undesirable traffic phase effects [7] when all flows were started at the same time.

Operational limitations The fact that a calculated rate cannot be reached precisely (because the rate granularity depends on the packet size) causes oscillations, rendering the mechanism potentially unstable. Consider a single link and a single flow traversing it: if, for example, the calculated rate is 2.5 packets per measurement interval, there will be an interval with 2 and another interval with 3 packets. Following the interval with 2 packets, the calculated rate will be much higher, causing the mechanism to overshoot the point of convergence. During the next interval, chances are that there will be too many packets, leading to a severely reduced rate and so on. These fluctuations are hard to come by and may not be so bad in the long run: on average, the source sends at the appropriate rate.

Yet they narrow the operational range of the mechanism — it can become unstable if there are too many flows, the smoothness parameter a is set too high or packets are too large with respect to the link bandwidth. In other words, the stability of the mechanism is directly proportional to the link bandwidth and indirectly proportional to packet sizes, the number of flows and a.

With a fixed value of 0.5 for a, the packet size imposes a lower limit on the bottleneck bandwidth and an upper limit on the number of users that CADPC can support. For values beyond these limits, oscillations lead to growing queues (and, thus, larger delay) and packet loss — in other words, the quality of the mechanism is significantly degraded. This is illustrated in fig. 1: in cases with a small bandwidth and a large number of flows, the packet loss ratio grows — in the extreme case of 100 flows traversing a 1 Mbit/s link, half of the packets are lost. The increased loss ratio coincides with an increased average queue length. Figure 2 shows the same scenario with a packet size of 500 instead of 1000 bytes;

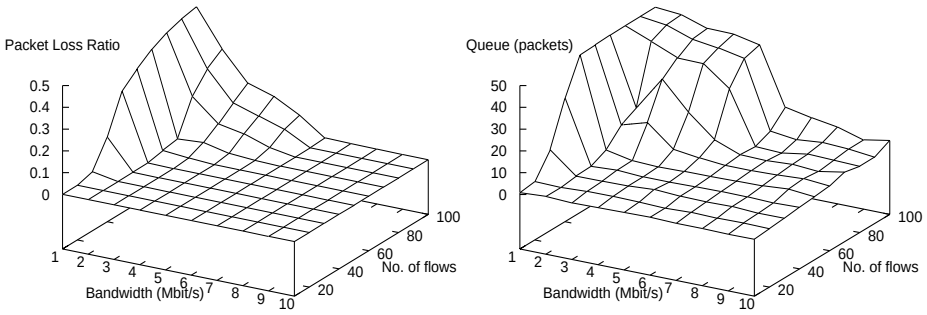

Fig. 1. Packet loss ratio (left) and average queue length (right) of CADPC

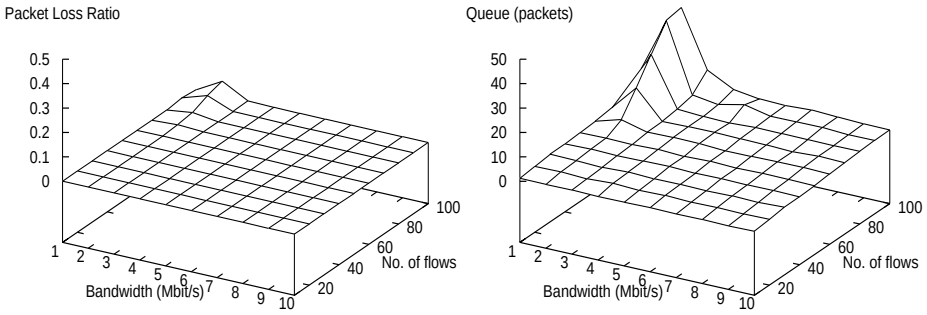

Fig. 2. Packet loss ratio (left) and average queue length (right) with smaller packets

obviously, the region with a packet loss ratio and queue length close to 0 grows as the packet size is decreased. Instead of changing the packet size, it would also be possible to choose a smaller value for a at the cost of having a less responsive system.

4 Performance Evaluation

The Performance Transparency Protocol was implemented for Linux (both configured as a router and as an end system) for simple functionality tests. So far, CADPC performance evaluations were only carried out using the *ns* network simulator[1]; except otherwise noted, simulation parameters were always set according to table 1. This includes the simulations in the previous section.

4.1 Dynamic Behavior

Two simulation examples are shown in fig. 3: the diagrams depict the total throughput 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

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

Table 1. Default simulation parameter values

Parameter	Value
Topology	Dumbbell
Packet size	1000 byte
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 (FTP in the case of TCP)
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"

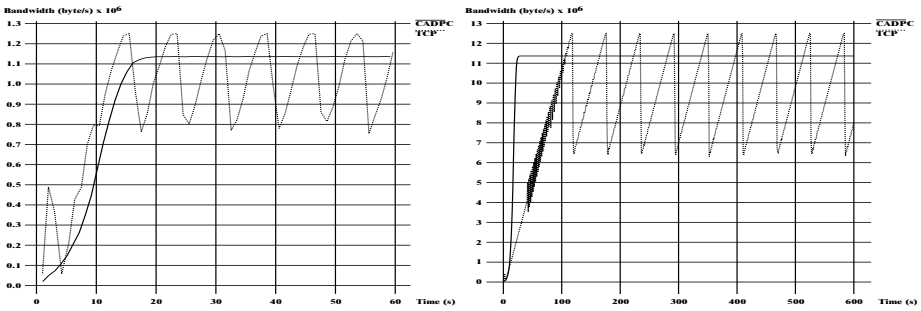

Fig. 3. CADPC vs. TCP with a 10 Mbit/s (left) and a 100 Mbit/s bottleneck (right)

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.

The diagram on the left-hand side of figure 4 depicts the rates of 10 flows that were started at the same time but had a different RTT. It was obtained by increasing the update frequency by 4 with the number of the flow, leading to rate updates of 4, 8, 12 ... RTTs. Clearly, the rates even converge towards their fair share of approximately 10 Mbit/s / $(1 + n)$ = 113636 byte/s in this very extreme example — neglecting queuing delay, the rate update frequency of flow no. 0 was 4*300=1200 ms and the frequency of flow no. 9 was 12000 ms, which is similar to changing the RTT from 300 to 3000 ms. This result additionally shows that it is possible for a flow to "artificially prolong" its RTT by updating the rate slower than every 4 RTTs. This is an important feature of CADPC,

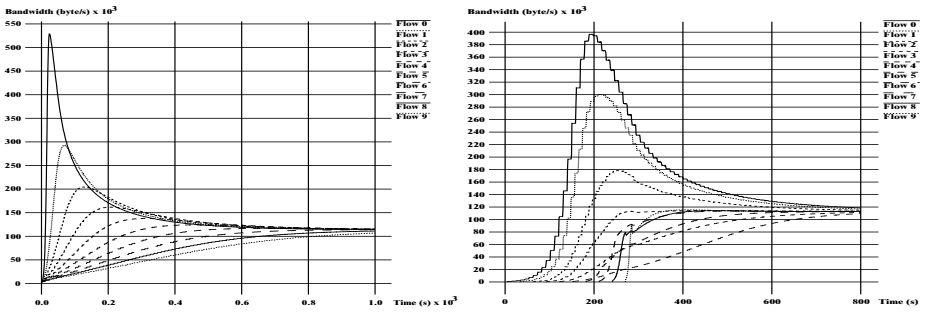

Fig. 4. Fairness with heterogeneous RTTs

as it enables imposing an upper limit on signaling traffic and therefore ensures scalability of the mechanism.

It can be seen that the enhanced startup behaviour almost degrades to regular logistic growth in this example — it takes flow no. 9 approximately 1000 seconds to reach its fair share — but the convergence itself is not hindered by the extreme diversity of RTTs. The diagram on the right-hand side shows a somewhat more realistic case; here, the RTT of each flow was greater than the RTT of the previous flow by 117 ms while a new flow entered the system every 30 ms. Apparently, some flows (e.g., flow 8) were capable of using the startup mechanism while others (e.g., flow 5) were not.

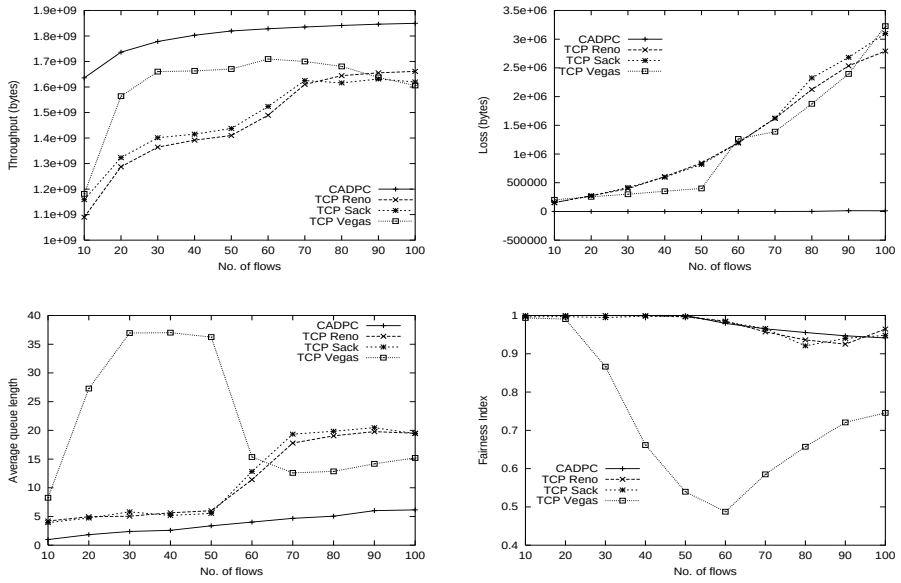

Fig. 5. CADPC and TCP, bottleneck 100 Mbit/s

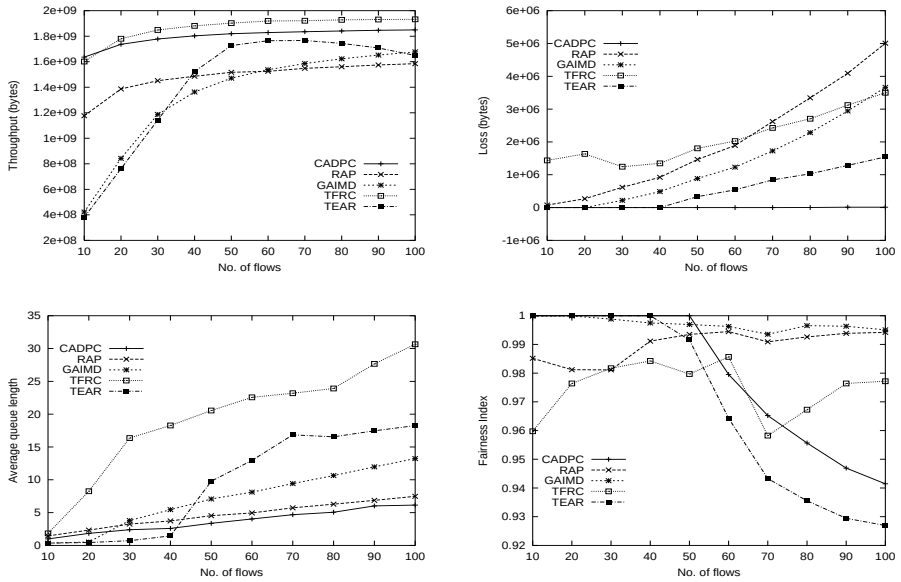

Fig. 6. CADPC and TCP-friendly mechanisms, bottleneck 100 Mbit/s

4.2 Long-term performance evaluation: CADPC vs. TCP(-friendly) Congestion Control

In oder to evaluate the performance of CADPC, its behavior was compared with TCP Reno, TCP SACK and TCP Vegas (each combined with ECN) and several TCP-friendly congestion control protocols in simulations — RAP [8], GAIMD [9] with $\alpha = 0.31$ and $\beta = 7/8$ (which was realized by tuning RAP parameters), and the slowly-responsive TFRC [10] and TEAR [11]. Flows of one particular kind at a time shared a single bottleneck link in a "dumbbell" topology. For all the results in figures 5 and 6, 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. PTP was used in a mode where packets have an initial size of 32 bytes and grow by 16 bytes at each router [2]. Only the payload flow is shown in the figures.

Note that in our simulation scenario, unresponsive senders which transmit at a high data rate would always achieve the greatest total throughput; therefore, high throughput can only be considered a good sign if the packet loss ratio is low. For example, the high throughput of TFRC is not necessarily a good sign because it corresponds with high loss. Yet, TFRC appears to work better with many flows than RAP because RAP shows less throughput as well as more loss.

CADPC outperformed its competitors in our simulations: it achieved more 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, "QoS" 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 (details can be found in [1]):

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

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

5 Related and future work

The advantage of moving a step further from binary to multilevel feedback is outlined in [12]. Actual ABR-like signaling based on RTCP is used with per-flow state in the "Adaptive Load Service" (ALS) [13] and with a window-based approach, no per-flow state but somewhat strenuous router calculations where the header of every payload packet is examined in XCP [15]. In [14] and [16], the focus is on the interaction between ABR-like signaling and DiffServ.

Among these approaches, XCP is particularly noteworthy, as it resembles CADPC 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 ([15] describes a method to separate flows via two different queues in routers in order to achieve TCP-friendliness).

Table 2 shows the most important differences between the two mechanisms. Briefly put, we believe that CADPC imposes less burden on routers at the cost of responsiveness; however, XCP is clearly a better choice in highly dynamic environments. CADPC, on the other hand, could also utilize SNMP to retrieve the relevant information. Such an implementation would be slightly less efficient but could be deployed without installing a software update in routers. The CADPC

Table 2. Differences between XCP and CADPC

Property	XCP	CADPC
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

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 traffic from the rest by exclusively placing it in a DiffServ class [17]. Other envisioned future work includes:

- CADPC converges to max-min fairness — such a rate allocation can also be attained by maximizing the linear utility function of all users. Although this may in fact suit certain special applications, common network services like file transfer and web surfing are known to have a logarithmic utility function; as of yet, it remains an open question whether CADPC could be extended to support such utility functions and converge to proportional fairness [18].
- So far, CADPC was designed to strictly rely on PTP feedback and not include any additional signaling. It is planned to extend the mechanism with regular per-packet acknowledgments and see whether it can be tuned to be TCP-friendly; also, this variant may be rendered gradually deployable by allowing the mechanism to use a default TCP-like behaviour in the absence of complete path information from PTP.
- Possible extensions for non-greedy flows and multicast scenarios should also be examined.
- Most importantly, gradual deployment methods should be pursued. In addition to the aforementioned idea of mandating CADPC behavior within a DiffServ class, it might be possible to utilize PTP signaling between edge routers of a domain and have these devices control traffic aggregates instead of end-to-end flows. This could be done in support of traffic engineering, or as dynamic resource allocation scheme for DiffServ.

6 Conclusion

In this paper[2], a novel approach to congestion control based on explicit and lightweight feedback was presented. The control law converges to a fixed and stable point, which accounts for higher throughput, less loss, reduced fluctuations and therefore better performance than TCP. Moreover, the rate calculation is simple and can be performed wherever feedback packets are seen; it is easy to enforce appropriate behavior and implement load-based and differentiated

[2] This paper is an overview of [1], which contains further details on the design rationale, extensive simulation results and a complete specification of the PTP protocol.

pricing. While our mechanism has the obvious drawback of requiring router support, we believe that future work on gradual deployment can lead to a service that is fit for the Internet.

References

1. Welzl M: Scalable Performance Signalling and Congestion Avoidance. Kluwer Academic Publishers, Boston, 2003. ISBN 1-4020-7570-7.
2. Welzl, M: A Stateless QoS Signaling Protocol for the Internet. Proceedings of IEEE ICPADS'00, July 2000.
3. Katz, D.: IP Router Alert Option. RFC 2113, February 1997.
4. Barnhart, A. W.: Explicit Rate Performance Evaluations. ATM Forum Technical Committee, Traffic Management Working Group, Contribution ATM Forum/94-0983 (October 1994).
5. Luenberger, D. G.: Introduction to Dynamic Systems - Theory, Models, and Applications. John Wiley & Sons, New York, 1979.
6. Jain, R., Ramakrishnan, K. K.: Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals and Methodology. Computer Networking Symposium, Washington, D. C., April 1988, pp. 134-143.
7. Floyd, S., Jacobson, V.: On Traffic Phase Effects in Packet-Switched Gateways. Internetworking: Research and Experience, V.3 N.3, September 1992, p.115-156. Earlier version: Computer Communication Review, V.21 N.2, April 1991.
8. Rejaie, R., Handley, M., Estrin, D.: RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. Proceedings of IEEE Infocom 1999, New York City, March 1999.
9. Yang, Y. R., Lam, S. S.: General AIMD Congestion Control. Technical Report TR-2000-09, Dept. of Computer Sciences, Univ. of Texas, May 2000.
10. Floyd, S., Handley, M., Padhye, J., Widmer, J.: Equation-Based Congestion Control for Unicast Applications. Proceedings of ACM SIGCOMM 2000.
11. Rhee, I., Ozdemir, V., Yi Y.: TEAR: TCP emulation at receivers – flow control for multimedia streaming. Technical Report, Department of Computer Science, North Carolina State University.
12. Durresi, A., Sridharan, M., Liu, C., Goyal, M., Jain, R.: Traffic Management using Multilevel Explicit Congestion Notification. Proceedings of SCI 2001, Orlando Florida 2001.
13. Sisalem, D., Schulzrinne, H.: The Adaptive Load Service: An ABR-like Service for the Internet. IEEE ISCC'2000, France, July 2000.
14. Li, N., Park, S:, Li, S.: A Selective Attenuation Feedback Mechanism for Rate Oscillation Avoidance. Computer Communications 24 (1), pp. 19–34, Jan. 2001.
15. Katabi, D., Handley, M., Rohrs, C.: Internet Congestion Control for Future High Bandwidth-Delay Product Environments. ACM SIGCOMM 2002, Pittsburgh, PA, 19-23 August 2002.
16. Kim, B. G.: Soft QoS Service (SQS) in the Internet. Proceedings of SCI 2001, Orlando Florida 2001.
17. Welzl, M., Mühlhäuser, M.: Scalability and Quality of Service: a Trade-off? IEEE Communications Magazine Vol. 41 No. 6, June 2003, pp. 32-36.
18. Kelly, F.: Charging and rate control for elastic traffic. European Transactions on Telecommunications, 8. pp. 33-37. An updated version is available at http://www.statslab.cam.ac.uk/frank/elastic.html