Composing QoS from Congestion Control Elements

Michael Welzl
University of Innsbruck
Institute of Computer Science
Distributed and Parallel Systems Group
A-6020 Innsbruck, Austria
Phone: +43 (512) 507-6110
Fax: +43 (512) 507-2977
michael.welzl@uibk.ac.at

Abstract

We sketch a QoS architecture which is exclusively based on congestion control as the key enabling mechanism. The idea is not to prioritize traffic at all within a network "cloud", but to rely upon a congestion control framework that is fully distributed and clearly defined; flows are weighted appropriately, which leads to fine-grain services that are defined and can be enforced at the edges, where they are negotiated as a traffic contract with end nodes. We believe that our architecture is feasible, much more in line with the open design of the Internet than existing approaches and can lead to a true multi service network.

1. Introduction

All major efforts that were made towards the realization of Internet QoS share a commonality: traffic is prioritized. In the case of the IntServ model, the RSVP signaling protocol dynamically sets up per-flow QoS — leading to special treatment of some packets — along a path (or multicast tree) whereas in DiffServ, a fixed set of core prioritization schemes (per hop behaviors) is predefined [1]. DiffServ uses edge routers to enforce traffic shaping, perform admission control and realize several other functions. It may be combined with economic models, and there is the concept of a bandwidth broker that can be used to set up appropriate QoS with signaling [2]. Such means can help to increase the service granularity at the cost of sometimes dramatically increased complexity [3]; so far, none of these extensions have led to the desired global success of any mechanism.

One fundamental problem of all legacy approaches is that they enforce special treatment of high-class as opposed to low-class packets somewhere within the network while facing the need for high flexibility of service definitions and fine service granularity. It has long been noticed that centralized mechanisms of any kind are not a good (i.e. scalable) design choice for Internet congestion control. In our perception, the same applies to QoS, where the enabling mechanisms should also be fully decentralized in order to attain increased flexibility, enable gradual deployment and provide ease of implementation. Prioritization in the network core does not seem to match the original Internet philosophy, trying to impose it creates a mixture of contradicting philosophies that cannot be expected to effectively work together. Current attempts to Internet QoS at least partly try to borrow approaches from ATM technology / philosophy, which is an orthogonal approach to network design and therefore cannot work well for the Internet.

We propose a departure from all of these efforts — an approach that is much more in line with the original Internet design concepts. Instead of flow aggregation, prioritization, dynamic resource management in core routers through signaling, and complex per-hop behaviors, our idea is to provide QoS solely by controlling what enters the core network. This is achieved by employing congestion control at edges; flows are weighted appropriately, which leads to fine-grain services that can be enforced at the edges and negotiated as a traffic contract between end nodes and edge routers or bandwidth brokers. The main technical advantages are that:

- services can be defined at the very edge of a network cloud without requiring any kind of reconfiguration in the core
- it is much more lightweight than existing approaches
- there is a finer service granularity than with DiffServ because the scheme does not use state aggregation; still, it is scalable

- since congestion control is used as a means to provide QoS, there is no adverse interaction between QoS and congestion control (as with TCP over DiffServ [4])

This paper is organized as follows: first, we explain why we believe that congestion control can be effectively applied as a key ingredient for QoS. In section 3, we sketch our envisioned architecture, which should be gradually deployable; example usage scenarios are given in scenario 4. Section 5 concludes.

2 Congestion Control for QoS

Congestion control did not stop to evolve when it was first described for TCP in [5]: among many other things, active queue management [6] and explicit communication between end nodes and the network [7] were introduced and more implicit information have been discovered and used [8] [9]. However, in order for a mechanism to be gradually deployable in the Internet, it should still be compatible with TCP, which is the prevalent transport protocol. The idea is to protect existing TCP flows from flows that are unresponsive or too aggressive — a single such flow can do severe harm to a large number of TCP flows. The common definition for a flow to be TCP-compatible (often called TCP-friendly) is [10]:

> *A TCP-compatible flow is responsive to congestion notification, and in steady-state it uses no more bandwidth than a conforming TCP running under comparable conditions.*

TCP causes congestion in order to avoid it: its rate is known to depend on the packet size, the round-trip time, the retransmit timeout (which is a function of instantaneous round-trip time measurements) and loss [10]. This means that it reacts upon changes of the round-trip time (usually due to changed queuing delay) and in response to dropped packets. Both of these factors obviously have a negative influence on the rate — and they are, to some degree, under control of the congestion control mechanism itself. We therefore believe that a new mechanism could perform better than TCP (actually *avoid* congestion, i.e. prevent queues from growing and minimize loss) by dropping the requirement of TCP-friendliness and incorporating the state-of-the-art of network measurement. Consider the following three examples:

1. The “eXplicit Control Protocol (XCP)” adds a “congestion header” to each packet. This header is sometimes updated by intermediate routers; timers ensure that most packets only need to be counted. In simulations covering a wide range of scenarios, XCP was shown to outperform TCP in several aspects. Its relative gain grows with the bandwidth $\times$ delay product; among its outstanding features is the fact that it decouples congestion from fairness control, causing the mechanism to converge to precise max-min fairness without requiring any per-flow state in routers [11].

2. The “Congestion Avoidance with Distributed Proportional Control (CADPC)” scheme changes the rate in proportion to the current rate and the available bandwidth. It uses the “Performance Transparency Protocol (PTP)” to efficiently query routers for performance related information in a scalable manner. While this scheme resembles XCP in some aspects, it differs in that routers do not perform any calculations but merely add some data to rare measurement packets; payload packets remain untouched. CADPC/PTP is slowly responsive; it was shown to outperform TCP in several aspects in simulation scenarios with long-lived “greedy” flows (sources that use all the bandwidth that they are given). Its relative gain grows with the bottleneck link capacity; just like XCP, it converges to precise max-min fairness without requiring any per-flow state in routers [12] [13].

3. The “Distance Weighted Additive Increase and Loss Rate Dependent Multiplicative Decrease Scheme (DWAI/LDMD)” does not require any explicit help from within the network. As with CADPC, its rate control function is proportional to the current rate and a certain allowable rate and depends on the actual value of the reported packet loss rate — i.e. it does not simply multiply the rate by a fixed factor in response to packet loss [14]. It is only one of many end-to-end mechanisms that are enhancements over TCP; other examples are FAST TCP [15] and HighSpeed TCP [16], which was specifically designed for links with high bandwidth $\times$ delay product.

None of these mechanisms are TCP-friendly by nature (XCP is tuned to be TCP-friendly by separately queuing XCP and TCP packets in [11]), and most of them achieve outperform TCP, in particular in environments that are a poor match for the protocol (high-speed, long-delay links). We believe that mechanisms such as the one in [8] and [9] could also be tuned to work better than TCP by dropping the requirement of TCP-friendliness. As another link between QoS and congestion control, it was discovered that the QoS perceived by users can be quantified with a utility function; ideally, the respective utility functions of all senders should be maximized [17]. While not strictly a congestion control mechanism, one such effort is [18]. If all utility functions are logarithmic (which is the normal case for applications such as web surfing and ftp downloads), such a scheme converges to “proportional fairness” [19]. XCP was shown to be capable of realizing this fairness measure [11]; since this

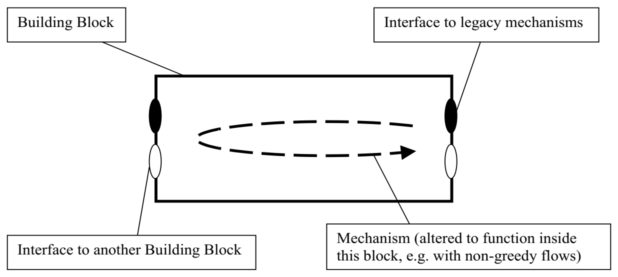
Building Block
Interface to legacy mechanisms
Interface to another Building Block
Mechanism (altered to function inside this block, e.g. with non-greedy flows)

Figure 1. QoS Building Block

is merely a result of applying weights to a mechanism that would otherwise converge to max-min fairness, the same method can be expected to work for CADPC/PTP as well. Their predictable and stable rate makes these mechanisms particularly attractive for QoS provisioning — in the case of CADPC, a sender can simply reach twice the rate of all others by acting like two senders (doubling the rate in its calculation). The applicability of XCP for bandwidth differentiation and hence QoS support is explained at length in [20].

There is a wealth of information on congestion pricing [21], where the general idea is to feed back congestion (sometimes based on the ECN signal) to users in the form of an increased price, which should cause an incentive to reduce the rate. Thus, the system is self-regulatory. Even TCP, which was misinterpreted to realize proportional fairness [22], can be used to differentiate between users and provide some sort of QoS [23].

3 Architecture

We envision a heterogeneous network which is composed of various interacting QoS building blocks that can be deployed at will depending on the environment and user or ISP requirements, respectively; each yields a certain local advantage and contributes to the enhancement of the whole network at the same time. At this point, let us regard building blocks as fully isolated environments (we will discuss isolation methods later). Within a block, existing QoS methods are combined with congestion control; as we will also see later, our architecture can serve as the "glue" that allows putting them together.

3.1 Building Blocks

Figure 1 depicts a generic building block — a fully functioning QoS environment of its own. As an instantiation of such a block, consider the following example:

- When a new flow wants to enter (or leave) the block, it contacts a bandwidth broker. This could be a dedicated network node, or it could be realized in a distributed manner, similar to the underlying information service when searching for a file in a peer-to-peer application.

- Depending on the current amount of traffic in the block as well as the QoS requirements of the new flow, the bandwidth broker either rejects or accepts the request. Upon acceptance, it tells edge routers about the admitted flow. This signaling could be carried out via COPS, whereas the admission process could use RSVP.

- All flows must behave according to the CADPC/PTP rules. The rate of a CADPC flow always converges to $c/(n+1)$, where c is the bottleneck capacity and n is the number of flows in the system; since the number of flows is known by the bandwidth broker, this value can be calculated in advance, and it is easy to check a flow for conforming behavior [13]. CADPC/PTP flows can be grouped together in order to provide differentiated services: if, for example, a flow acts like two flows by simply duplicating its calculated rate, it attains exactly twice the throughput of all other flows [12].

The CADPC/PTP prioritization within by this block could be used to realize the services defined for IntServ — grouped together with RSVP signaling, the whole block could emulate an IntServ environment. Note that, since CADPC/PTP is not TCP-friendly (and deterministic QoS would be very hard to attain in the presence of TCP), the aforementioned isolation is quite necessary in this example.

3.2 Gradual Deployment

The idea is to construct usage scenarios which can be seen as virtual environments that provide just the right degree of isolation; depending on the scenario in question, specific non-TCP-friendly congestion control mechanisms may work better than others. As we will see, three key techniques for isolation should be developed as a prerequisite for gradual deployment:

1. Control of heterogeneous flows — this applies to flow aggregates (which means that our end nodes are edge routers and our architecture is used as a dynamic traffic management tool) as well as non-greedy sources

2. Mechanism translation — segment-based deployment could be done by converting a TCP-friendly mechanism to a non-TCP-friendly one and vice versa, using typical QoS tools such as traffic shapers

3. Traffic isolation — special and uniform control of individual flows within a QoS aggregate ("QoS in the small")

A building block is supposed to be capable of interacting with legacy mechanisms; for example, it might be possible to create a network “cloud” that emulates DiffServ so that it can interact with DiffServ domains (satisfy SLAs). In what follows, some examples are described; for ease of reading, we use “CC” as a replacement for any applicable and highly efficient, typically non-TCP-friendly congestion control mechanism.

3.3 CC Traffic Management

Typically, traffic management takes place on a long time scale (possibly through manual intervention of network administrators). Traffic management based on CC differs from traditional traffic management in these aspects: it is supposed to dynamically adapt to changing network state by itself and can react much faster. It was nevertheless decided to choose the term “traffic management” because of the similar goal: to make the most efficient use of network infrastructure in a transparent manner (i.e. without individual end nodes being aware of it).

So far, most mechanisms were only studied assuming greedy flows. While this is a valid model for certain end-to-end applications such as file transfer, it is sometimes necessary to control flows that show a different behaviour. This may, for example, be the case for streaming media applications that cannot adapt their rate with the fine granularity that is mandated by the network. Most importantly, flow aggregates can be seen as special flows with certain traffic characteristics; these flows are not greedy by nature. Controlling non-greedy flows can be performed in various ways:

- The controlling mechanism can be altered appropriately
- QoS elements that the DiffServ architecture is built upon can be used to control the traffic so as to make it suitable — for example:
  - Traffic shapers can impose a limit on the burstiness of a flow. Here, special care must be taken not to degrade the performance of adaptable flows such as TCP, which is known to be a poor fit for token buckets [4].
  - Buffers can generally be used to delay packets
  - Policing can be used to enforce a certain dynamic behaviour of flow aggregates even when the individual flows are not capable of adapting in response to bandwidth changes
- These elements can be installed in various places: in the end systems, as a network layer service, in middle boxes or domain edge routers

In the case of TCP “over CC”, it may be possible to apply the vast amount of existing knowledge on TCP behavior with an underlying ATM ABR service because of the similarities between XCP or CADPC/PTP and ATM ABR (two example research studies in this area are [24] and [25]. Once it is clear how to control non-greedy flows, this knowledge could be applied to control flow aggregates within a domain[1], with edge routers acting as end nodes.

3.4 Mechanism Translation

Transparency is a key factor when deploying a mechanism in a gradual manner: ideally, it should be possible for edge routers to use CC with per-flow granularity along a path segment without TCP senders noticing it and the other way round (two CC end nodes should be able to seamlessly tunnel through a non-CC-capable network cloud). At the far end of the spectrum, one could even imagine one sender using CC without the other one noticing it — like, for example, an RTP-based streaming audio application where the sender uses CC to control its data rate without having to inform the receiver.

In any case, it is necessary to take a close look at typical protocols that CC needs to be “translated” to (and vice versa); among a few RTP based data streams, the new Datagram Congestion Control Protocol (DCCP) [26] that is being worked at in the IETF, its profiles and TCP play the most important roles here. Translating one congestion control mechanism into another means that traffic has to be shaped appropriately; from the perspective of one side, the other side appears to be a possibly non-greedy application that would like to transmit data — thus, knowledge of controlling heterogeneous data flows is also required for this building block.

3.5 “QoS in the Small”

In order for the architecture to work, it should be possible to isolate CC traffic from all other flows, including traffic that exhibits long-range dependence (web traffic) and TCP. QoS mechanisms are intended to isolate traffic; if, for example, a user reserves a certain guaranteed bandwidth, the expected behaviour is essentially the behaviour of the user on a leased line (or “empty pipe”). Our proposed solution in this context is to co-design a DiffServ class and a congestion control mechanism. Simply put, if there was a totally isolated traffic class for high class users and all users within this class would deploy the same congestion

[1]In this context, a “domain” is a network “cloud” but not necessarily an ISP domain because building blocks may be embedded in the network wherever they lead to a performance gain. Edge routers are therefore not necessarily edge routers of an ISP domain but routers at the edge of the “cloud”.

control mechanism, there would be a clear benefit for everybody because the chosen congestion control mechanism may simply work better than TCP; this was also identified as a viable way to realize fine-grain QoS in a scalable manner in [27]. A possible requirement of router support as with XCP or CADPC/PTP would be less of a burden in such a scenario because it would be restricted to high-class users only.

The assumption of a fully isolated DiffServ class may be unrealistic; the tricky part would be to appropriately design traffic isolation elements while applying tools such as traffic shapers to keep the functionality of CC intact. We believe that this is possible, or at least worth trying.

3.6 “QoS in the Large”

This building block builds on the “CC based traffic management” idea: once it is clear how to control traffic within a domain, this knowledge can be applied to enhance the functionality of DiffServ, where traffic is classified into traffic aggregates at edge routers and the inner network infrastructure of a domain must be exploited based on these traffic aggregates. The goal of this work is to transparently enhance the service provided by DiffServ; CC is carried out among edge routers without sources being aware of it. In this context, sources may be individual DiffServ-aware or non-aware users or traffic aggregates from an adjacent domain that are classified accordingly.

4 Scenarios

There are numerous ways of using the building blocks, depending on the function required; some may only be a technical enhancement (e.g., if only the “QoS in the large” building block is used) while others allow for deployment of a complete QoS architecture with new service definitions; such an architecture would have to encompass an additional signaling plane for service negotiation with edge routers or bandwidth brokers. In what follows, some example scenarios will be described.

4.1 Example 1: Corporate Network with QoS Traffic Isolation

In this simple example, some users of a large corporate network occasionally participate in strictly internal video conferencing sessions. These users are important — the video conferencing quality should be preserved at the potential cost of degrading the service of all other users (who use Intranet applications, surf the web and send email). This can already be done using the DiffServ QoS mechanism to distinguish between the high-class video conferencing participants and all other users.

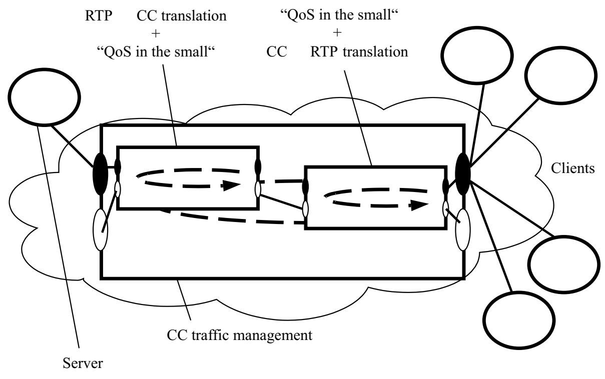

Figure 2. The streaming audio scenario

With the “QoS in the Small” building block, it is possible to use our architecture for the high-class users only. This is done with the traffic isolation technique, which consists of some changes to both the “outer” and the “inner” QoS mechanism — DiffServ and CC in this case. In this scenario, the video conferencing applications would be fully CC-capable, thereby achieving a better video quality without any notable disadvantage for any of the low-class users. Service differentiation within the CC class would be possible; in a sense, this would mean stacking a QoS architecture onto another.

4.2 Example 2: ISP providing CC Based Services

An ISP provides its customers with additional high-quality streaming audio from a server that is located within the same domain as the customers. As the number of customers has grown, the QoS has degraded to a degree where customers complain.

Simply turning on DiffServ would not suffice: there is just not enough network capacity to provide the service to all interested customers. One solution in this scenario may be overprovisioning, but there is a cheaper one — CC within a DiffServ class (“QoS in the small”). Technically, this example is an extension of the previous one: in an isolated network domain, DiffServ is activated and CC is used within a high-class traffic aggregate, but this time, neither the audio server nor the clients are CC-capable, legacy software is used — installing new software on both sides may be too much burden for both the ISP and the end users. Also, the ISP deploys CC traffic management to enhance the overall network utilization in the domain.

The scenario is shown in fig. 2. There are three building blocks; the big one represents CC based traffic management. The two inner ones each perform the function of translating from the outer mechanism to CC (and vice versa), while being protected from the impact of all other traffic as in the first example. The inner blocks are connected via CC and both interact with the outer block using CC. While this is not vital for the basic functionality, it may help the outer block fulfill its task in that it facilitates dealing with the traffic aggregates.

All the clients and the audio server are connected using non-CC-capable links (RTP in the figure); thus, CC operates totally transparently in this example. If the connections in this domain are known to be symmetric, the ISP has another advantage: using CADPC/PTP or XCP feedback, it would be straightforward to monitor the rate of each user and thereby charge per traffic. Also, this information could be used to control traffic shapers at the edges of the domain, thereby enforcing appropriate behavior from all the clients within the CC traffic aggregate. As in the previous example, installing a complete CC based QoS architecture would be possible in this scenario.

4.3 Example 3: ISP using CC Traffic Management only

This scenario is simple: an ISP can use the "CC Traffic Management" building block only to achieve better utilization of its network resources. This can lead to better service for all customers or similar service with cheaper network infrastructure (less capacity). The traffic management block can or cannot communicate with other building blocks; should a peering ISP use CC traffic management as well, CC communication between both ISPs would enhance the quality of the CC traffic management block even further. Service differentiation between aggregates could be used to satisfy DiffServ SLAs with peering ISPs.

4.4 Example 4: Heterogeneous Video Conferencing Scenario

The scenario depicted in fig. 3 shows an example of using CC in support of an RTP based adaptive video conferencing software; in reaction to RTCP feedback, its sending rate is changed to suit the currently available bandwidth. In this case, there are two different end user applications — one is not aware of CC (the upper nodes) and one is (the lower right node).

Once again, there is a domain where CC is used to control traffic aggregates; this time, it is not CC based traffic management but a DiffServ enhancement ("QoS in the large"). The inner block translates RTP from the non-CC-capable application and the stream from above to CC. Also, as in the second example, it communicates with the outer block to facilitate controlling the outer traffic aggregates. The upper left application is not CC-capable and not greedy; unknowingly, it connects to a block that controls the traffic of the application somewhere outside the domain. This block is connected to a second block that takes care of translation to RTP as it is not known whether the next hop is going to be a peering application or a router of some kind — but at the other end of the RTP stream, there is the next CC block. CC based service differentiation would be possible but less meaningful in this scenario because the CC-capable regions are not consecutive: RTP acts as a tunnel through a non-CC-capable region.

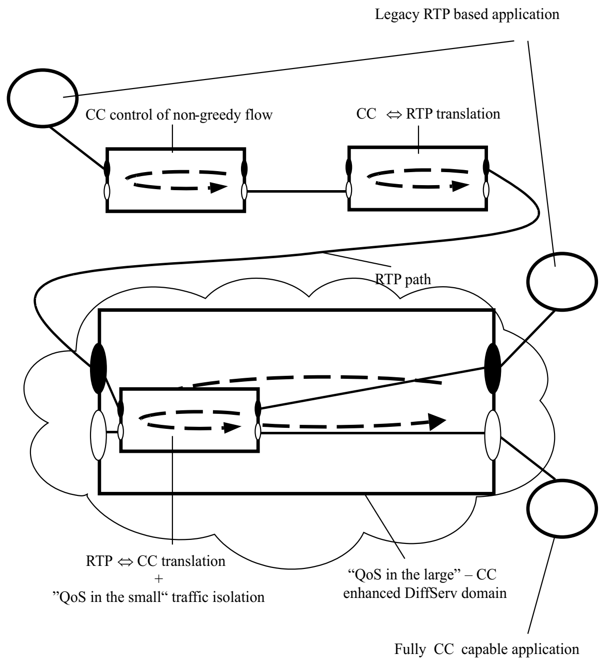
Legacy RTP based application
CC control of non-greedy flow
CC ⇔ RTP translation
RTP path
RTP ⇔ CC translation
+
"QoS in the small" traffic isolation
"QoS in the large" – CC enhanced DiffServ domain
Fully CC capable application

Figure 3. A usage scenario in a heterogeneous environment

5 Related Work

There are many mechanisms that relate both to the QoS and congestion control domain in some aspect in addition to the approaches mentioned in section 2 — for instance, the whole research domain of adaptive real-time multimedia applications that alter the quality of a media stream to adapt to the currently available bandwidth [8]. Congestion control mechanisms such as TFRC [10] are designed to support QoS by means of a smooth rate that makes them especially suitable for streaming media. A good overview of TCP-friendly congestion control mechanisms can be found in [28].

To the best of our knowledge, the idea of a gradually deployable QoS architecture where, apart from traffic contract enforcement at the edges, congestion control is the only enabling mechanism and no prioritization is enforced in the core is new — but some ideas come quite close: controlling traffic aggregates in the sense of combining congestion control and QoS is described in [29]. Ideas to enhance DiffServ with ATM ABR-like signaling were proposed in [30] and [31].

One unique feature of our architecture is the fact that it can easily be tailored to work in heterogeneous environments by choosing a building block that is based on an appropriate CC mechanism. CADPC/PTP and XCP, for example, rely on byte counting and therefore do not encounter the problems seen with TCP over noisy (wireless) links; moreover, both are designed to work well across so-called "long fat pipes" (links with a large bandwidth $\times$ delay product) [11] [12]). In other environments, however, mechanisms such as DWAI/LDMD [14] and FAST [15], which do not require router support and rely on packet loss as an indication of congestion, may work perfectly fine.

6 Conclusion

In this paper, we have sketched an architecture that realizes QoS in a fully distributed manner without requiring core routers to implement any prioritization. The key element that can enable this functionality at per-flow granularity is Congestion Control; we argue that Congestion Control and QoS should not be seen as separate but rather complementary network functions.

Most of the things described in this paper have not yet come into being (this concerns, for example, the admission control functionality that is needed at network edges); they will require a lot of work, including an in-depth study of potential impacts on existing technology. Yet, we believe that the architecture itself is beneficial and, given the number of possible building blocks, feasible. The potential to interface emulate and interface with IntServ and DiffServ services would once again facilitate gradual deployment — a key success factor in the Internet.

Rather than demonstrating the value of a specific research effort, the goal of this paper was to explain why QoS without prioritization in routers could work and thereby motivate research in this direction. We believe that actual development and deployment of our proposed architecture could eventually turn out to be one of the long-awaited steps towards a realistic multi-service Internet.

7. Acknowledgments

The author would like to thank Costas Courcoubetis, Jon Crowcroft, Max Mühlhäuser, Michael Smirnov and Ioannis Stavrakakis.

References

[1] Grenville Armitage, "Quality of Service In IP Networks: Foundations for a Multi-Service Internet", *Macmillan Technical Publishing*, April 2000.

[2] Zhi-Li Zhang, Zhenhai Duan, Lixin Gao, Yiwei Thomas Hou, "Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Services", *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, August 28 - September 1 2000.

[3] Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski and E. Felstaine, "A Framework for Integrated Services Operation Over DiffServ Networks", *RFC 2998*, November 2000.

[4] G. Huston, "Next Steps for the IP QoS Architecture", *RFC 2990*, November 2000.

[5] Van Jacobson, "Congestion Avoidance and Control", *Proceedings of ACM SIGCOMM* 1988, pp. 314-329.

[6] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, August 1993.

[7] K Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", *RFC 3168*, September 2001.

[8] Dorgham Sisalem and Adam Wolisz, "LDA+: A TCP-Friendly Adaptation Scheme for Multimedia Communication", *Proceedings of ICME 2000*.

[9] Guanghui He, Gao, J.Hou, and K. Park, "A Case For Exploiting Self-Similarity of Network Traffic in TCP Congestion Control" *Proceedings of IEEE ICNP 2002*, Paris, 12-15 November 2002.

[10] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer, "Equation-Based Congestion Control for Unicast Applications", *Proceedings of ACM SIGCOMM 2000*.

[11] Dina Katabi, Mark Handley, and Charlie Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", *Proceedings of ACM SIGCOMM 2002*, Pittsburgh, PA, 19-23 August 2002.

[12] Michael Welzl, "Scalable Performance Signalling and Congestion Avoidance", Kluwer Academic Publishers, August 2003.

[13] Michael Welzl, "Traceable Congestion Control", *Proceedings of ICQT 2002 (International Workshop on Internet Charging and QoS Technologies)*, Zürich, Switzerland, 16-18 October 2002.

[14] Pantelis Balaouras and Ioannis Stavrakakis, "A Congestion Control Scheme for Continuous Media Streaming Applications", *Proceedings of ICQT 2002 (International Workshop on Internet Charging and QoS Technologies)*, Zürich, Switzerland, 16-18 October 2002.

[15] Cheng Jin, David X. Wei and Steven H. Low, "FAST TCP: motivation, architecture, algorithms, performance", *Proceedings of IEEE Infocom*, March 2004.

[16] Sally Floyd, "HighSpeed TCP for Large Congestion Windows", *RFC 3649*, December 2003.

[17] Scott Shenker, "Fundamental Design Issues for the Future Internet", *IEEE Journal on Selected Areas in Communications*, 13, pp. 1141-1149, 1995.

[18] Narayanan Venkitaraman, Jayanth P. Mysore, and Mike Needham, "A Core-Stateless Utility Based Rate Allocation Framework", *Proceedings of PFHSN 2002* (Protocols for High Speed Networks 2002 - IFIP TC6 WG6.2 / IEEE Comsoc TC on Gigabit Networking), Springer Verlag, Berlin, Germany, 22-24 April 2002.

[19] Frank Kelly, "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

[20] Dina Katabi, "Decoupling Congestion Control from the Bandwidth Allocation Policy and its Application to High Bandwidth-Delay Product Networks", PhD Thesis, MIT, March, 2003.

[21] Costas Courcoubetis and Richard Weber, "Pricing Communication Networks", Wiley 2003.

[22] Milan Vojnovic, Jean-Yves Le Boudec, and Catherine Boutremans, "Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times", *Proceedings of IEEE Infocom 2000*, Tel Aviv, Israel, March 2002.

[23] P. Oechslin and J. Crowcroft, "Differentiated End-to-End Internet Services using a Weighted Proportional Fair Sharing TCP". *ACM CCR*, 1998.

[24] Charoenpanyasak, S., Kamolphiwong, S. and Thongnoo K. (2001). TCP Throughput Performance on ABR/UBR Services in Large Scale ATM Switch Networks. *Proceedings of SCI 2001*. Orlando Florida 2001.

[25] Abdelrahman, A. M. and Djuanda, J. N. S. (2001). Internet over ATM between the Enhancement of ABR Flow Control and the Impact of TCP Parameters. *Proceedings of SCI 2001*, Orlando Florida.

[26] Eddie Kohler, Mark Handley, and Sally Floyd, "Datagram Congestion Control Protocol (DCCP)", *Internet draft* (work in progress) draft-ietf-dccp-spec-07.txt, July 2004, available from the on-line Internet draft directories via http://www.ietf.org

[27] Michael Welzl, Max Mhlhuser, "Scalability and Quality of Service: a Trade-off?", *IEEE Communications Magazine*, Vol. 41 No. 6, June 2003, pp. 32-36.

[28] J. Widmer, R. Denda, and M. Mauve, "A Survey on TCP-Friendly Congestion Control", *IEEE Network Magazine*, Special Issue "Control of Best Effort Traffic" Vol. 15, No. 3, May 2001.

[29] D. Harrison, S. Kalyanaraman, and S. Ramakrishnan, "Congestion Control as a Building Block for QoS", *Student Poster, SIGCOMM 2001*, August 2001. Reprinted in Computer Communication Review, Volume 32, Number 1, January 2002.

[30] N. Li, S. Park, S. Li, "A Selective Attenuation Feedback Mechanism for Rate Oscillation Avoidance", *Computer Communications*, Vol. 24., No. 1, pp. 19–34, Jan. 2001.

[31] B. G. Kim, "Soft QoS Service (SQS) in the Internet", *Proceedings of SCI 2001*, Orlando Florida 2001.