A Case for Middleware to enable Advanced Internet Services

Michael Welzl
Institute of Computer Science, Distributed and Parallel Systems Group
University of Innsbruck, Technikerstr. 5-7, A-6020 Innsbruck
Austria
Michael.Welzl@uibk.ac.at

***Abstract* - Despite the many research efforts related to Internet congestion control and Quality of Service, new innovations in these two areas hardly ever make it into the TCP/IP stacks of standard end systems. We believe that this is due to lack of a well-defined interface to the application as well as a lack of transparency – a problem that could be solved by introducing middleware above TCP/IP.**

***Key-Words* - Congestion Control, DCCP, QoS, Middleware, Adaptation Layer**

1 Introduction

The broad range of modern Internet applications, ranging from a web browser to highly complex Grid computing software, calls for flexible and efficient methods to use the underlying network infrastructure. Nowadays, flexibility is quite restricted as the standard TCP/IP stack in a typical Internet end system provides only two different transport services:

1. Unreliable datagram delivery (based on the User Datagram Protocol, UDP)

2. Reliable in-order delivery of a consecutive data stream (based on the Transmission Control Protocol, TCP)

While we may be used to this limitation, it is in fact somewhat strange if we consider the immense amount of research on TCP and potential alternatives that is carried out; somehow, the results of all this research hardly seem to make it into the operating systems of our personal computers. New services were once promoted in the form of "Quality of Service" (QoS), which is quite a different approach — a set of mechanisms for service providers to sell better and accordingly priced services to high-class customers. Deployment of Internet QoS towards end users also never happened.

We propose to add middleware — an "Adaptation Layer" — on top of TCP/IP; by providing a unique service oriented interface and realizing a certain degree of transparency, we believe that our Adaptation Layer could enable the deployment of new transport services, no matter if they are realized as TCP alternatives or as actual end-to-end Internet QoS with strict guarantees.

This paper is organized as follows: in the next two sections, we take a closer look at the aforementioned problems. In section 4, we explain the Adaptation Layer, followed by a discussion of its potential implications. Section 6 concludes.

2 Deployment Problem 1: Transport Layer Developments

TCP is known to be the prevalent protocol in the Internet; a recent study has undermined this fact with a number of approximately 83% [1]. While UDP merely adds identification of different communicating entities at a single site (port numbers) and data integrity (checksum) to the "best effort" service provided by the IP protocol, TCP encompasses a variety of functions — some of them are:

Reliability: connections are set up and torn down; missing segments are retransmitted. If something goes wrong, the application is informed.

In-order delivery: segments are ordered according to sequence numbers by the receiver.

Flow control: the receiver is protected against overload.

Congestion control: the network is protected against overload using a variety of mechanisms, encompassing:

- **Ack-clocking:** roughly, in equilibrium, no packet enters the network unless a packet leaves the network; this helps to ensure network stability.
- **Rate control:** the rate of a sender is reduced in response to congestion, following a rule called "Additive Increase, Multiplicative Decrease" (AIMD) for reasons of network stability and fairness.
- **Round-trip time estimation:** acknowledgments are used to update an internal estimate of the round-trip time at the sender; this variable is important to fine-tune the other mechanisms.

Congestion control alone consists of several intertwined algorithms; together, they make TCP a somewhat complex beast. This does not pose a problem in practice because its implementation is entirely left up to operating system designers.

As the Internet grows, new applications arise. Following the most traditional ones (file download and remote login) and the rise of the World Wide Web, the most important application class that has emerged is probably real-time streaming media, encompassing live radio or TV transmissions, Internet telephones and video conferencing tools. Such applications differ from the typical earlier applications in that they do not necessarily require reliability, but timely transmission of data — thus, the protocol of choice is UDP, which does not retransmit (possibly old) packets like TCP does.

This is where problems occur: the most complex part of TCP is without a doubt its congestion control functionality. When it was added to TCP to protect the network from overload and ensure its stability [2], this appeared to be a sensible choice because i) TCP was the prevalent protocol and ii) congestion control in TCP is combined with its window-based flow control, which was already available in the protocol. In the case of UDP, implementing congestion control is entirely left up to application designers — and it is clearly necessary to have at least some kind of congestion control in order to maintain the stability of the Internet [3].

While applications using UDP should contain a congestion control mechanism that ensures fairness towards TCP ("TCP-friendliness"), the window-based stop-and-go behavior and heavy rate fluctuations of TCP are a poor match for real-time streaming media applications. Thus, a large amount of research on more suitable ("smoother") yet TCP-friendly congestion control has been carried out; some examples are TFRC [4], RAP [5], LDA+ [6] and Binomial Congestion Control [7]. An overview is given in [8].

The fact that the burden of implementing such a mechanism should not be placed upon the application designer — who would also have to be convinced to implement something which does not necessarily provide an immediate benefit for her application but rather for the network as a whole — has only recently been acknowledged: the "Datagram Congestion Control Protocol (DCCP)", which is currently undergoing IETF standardization, is a means to support applications that do not require the reliable transport of TCP but should nevertheless perform (TCP-friendly) congestion control [9]. DCCP can be regarded as a framework for such mechanisms which provides a wealth of additional features, encompassing partial checksums — this makes it possible for an application to utilize corrupt data instead of not having it delivered at all (corrupt data is suitable for certain video and audio codecs); also, partial checksums can be used to circumvent the common problem of corruption being interpreted as a sign of congestion.

The "Stream Control Transmission Protocol" (SCTP) is another new transport protocol which, like DCCP, is not yet commonly available in the Internet. Unlike DCCP, it is already standardized and ready for deployment [10]. SCTP was designed to transfer telephone signaling messages over IP networks, but is capable of broader applications; among other things, it extends the TCP service model by separating reliability from in-order delivery. SCTP provides the following features [11]:

Multi-stream support: logically independent user messages can be delivered independently, which can alleviate the so-called "head-of-line-blocking" problem caused by the strict sequence delivery of TCP. Nowadays, applications requiring this type of functionality typically utilize multiple TCP connections; this is similar to the new SCTP feature from an application programmer's point of view, but different for the network as there is only one congestion control instance for multiple streams in the SCTP case.

Multi-homing support: with SCTP, a connection is not associated with a single IP address and a port number on each side but with a set of IP addresses and port numbers — thus, in the case of failure, a connection can transparently switch from one host to another. Such things are already done at a higher level for websites, but with SCTP, multi-homing is realized where it belongs: at the transport layer. This makes the feature independent from the application — SCTP based FTP, for instance, would automatically use it.

Preservation of message boundaries: this is useful when application data are not a continuous byte stream but come in separate logically independent chunks.

Unordered reliable message delivery: data that come in logical chunks which do not require ordered transmission can sometimes be handed over to the application faster than with TCP (once again, this is due to elimination of the "head-of-line-blocking" problem).

Notably, the congestion congestion control behavior of TCP remains unchanged in SCTP, regardless of the type of application which uses the protocol.

To summarize, an Internet application programmer can be expected to face the following choice of transport services (provided by her operating system) within the next couple of years:

- Simple unreliable datagram delivery (UDP)
- Unreliable congestion controlled datagram delivery (DCCP)
  - with a choice of congestion control mechanisms
  - with or without delivery of erroneous data
- Reliable congestion controlled in-order delivery of
  - a consecutive data stream (TCP)
  - multiple data streams (SCTP)
- Reliable congestion controlled unordered but potentially faster delivery of logical data chunks (SCTP)

This is only a rough overview: each protocol provides a set of features and parameters that can be tuned to suit the environment or application — for example, the size of an SCTP data chunk (or UDP or DCCP datagram) represents a

trade-off between end-to-end delay and bandwidth utilization (small packets are delivered faster but have a larger per-packet header overhead than large packets). Work based on TCP parameter tuning is described in [12] and [13].

Nowadays, the problem of the transport layer is its lack of flexibility. In the near future, this may be alleviated by the availability of DCCP and SCTP — but then, it is going to be increasingly difficult to figure out which transport protocol to use in what mode and how to tune its parameters. The problem will not go away. Rather, it will turn into the issue of coping with transport layer complexity.

3 Deployment problem 2: End-to-end QoS

In modern networks with applications such as video and audio transmission or interactive online games, it becomes more and more interesting for ISPs to sell differentiated end user services which are accordingly paid for. Thus, following early efforts with ATM networks, the Internet QoS architecture "Integrated Services (IntServ)" was devised. Along with its special (albeit architecturally separated) reservation protocol "RSVP", IntServ was designed to provide strict service guarantees at the granularity of individual data flows, but it failed as a commercial product. This failure was generally accredited to the fact that its scalability was limited by the fine service granularity — which led to the design of "Differentiated Services" (DiffServ), where users are classified into traffic aggregates at domain endpoints in order to reduce the state for core routers [14].

In practice, neither the IntServ nor the DiffServ architecture is used as a service to end users and their applications. According to [15], this is at least partially due to a "chicken-egg-problem": network operators are not going to invest in deployment and support of a distinguished service infrastructure unless there is a set of clients and applications available to make immediate use of such facilities. Clients will not pay for enhanced services unless they see a performance gain in their applications; application programmers will not bother to implement support for enhanced network services unless these services are deployed. We believe that this problem stems from a certain lack of transparency in end systems; it could therefore be alleviated by the middleware described in this paper.

There are, however, additional reasons for the failure of QoS as a service for end customers. One problem is the fact that end-to-end QoS requires everything from application 1 to application 2 to fulfil the service guarantees — this includes the operating system in both computers and all the routers in between them as well as the underlying link layer and even physical layer technology (e.g., strict bandwidth guarantees may never be possible when CSMA/CD is involved and a part of the network is set up in a way that may lead to collisions). Then, there is the problem of traversing multiple ISP domains — there is no globally agreed upon set of regulations for all the related economic issues (how much does a high-quality service costs, and who is given which share?). Here, the Adaptation Layer may help by motivating ISPs to provide QoS because it has an immediate benefit; we will discuss this issue in section 5.

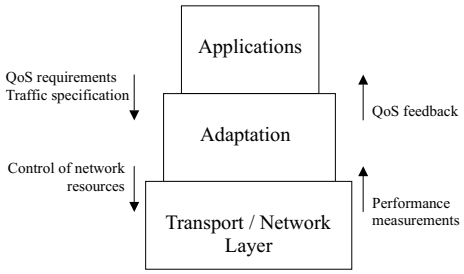

Figure 1. The Adaptation Layer

4 The Adaptation Layer

The difficulty of choosing and tuning a transport service can be alleviated by hiding transport layer details from applications; an underlying "Adaptation Layer" could simply "do its best" to fulfill application requirements by choosing from what is available in the TCP/IP stack. This method also makes the application independent of the given network infrastructure — as soon as a new mechanism becomes available, the Adaptation Layer can use it and the application works better.

The envisioned architecture is shown in fig. 1; here, an application specifies:

- the network requirements in a standardized and appropriate manner; e.g., instead of requesting a guaranteed bandwidth, it provides appropriate weights to tune the trade-off between factors such as bandwidth, delay, jitter, rate smoothness and loss.

- the behavior that it will show. Is it "greedy" (i.e. does it use all the resources it is given)? If not, what is the traffic pattern — e.g., are there long transmissions with short interruptions or vice versa? Only with knowledge of the expected behavior will the Adaptation Layer be able to ideally use the network. For example, interrupting a TCP connection can cause the congestion window to decay [16]. Therefore, the underlying AIMD policy, which has the protocol slowly increase the rate, can lead to a severe throughput reduction that might be compensated for by the middleware via buffering if i) delay is known to be less important than bandwidth and ii) the data stream is known to have short interruptions only.

  In contrast with strictly interactive applications, where the traffic pattern mainly depends on user behavior, this is especially important for Grid software — applications which often generate traffic on their own, which means that the underlying pattern may be predictable and could perhaps be specified precisely.

  As another example, an adaptive multimedia application (a real-time Internet application which adapts multimedia content based on network measurements) may not always be able to satisfy the criteria dictated by the network: in the case of layered data transmission, there may only be a limited number of rate steps

(i.e. the application may have a coarser rate granularity), or there can be upper and lower bounds. When tuning the compression factor of a data stream to suit the available bandwidth, it may not even be possible to predict a precise rate but only an average of some sort. Knowing about these things could be helpful for the middleware.

The Adaptation Layer controls resources by:

- choosing and tuning an appropriate mechanism from the transport or network layer — depending on the environment, this could mean choosing an appropriate congestion control mechanism and tuning its parameters via the DCCP protocol, making use of a protocol such as UDP Lite [17], which is tailored for wireless environments, or even using an existing network QoS architecture such as IntServ or DiffServ.

- performing additional functions such as buffering or choosing the ideal packet size (this can be a tricky task: while large packets generally cause less overhead as long as they do not exceed the fragmentation limit (the "Maximum Transmission Unit (MTU)" of a path), small packets yield a shorter delay — the choice must therefore depend on the bandwidth and delay requirements that were specified by the application). The packet size could, of course, also be mandated by the application or a congestion control mechanism.

The Adaptation Layer needs to monitor the performance with respect to the requirements that were specified earlier. It provides QoS feedback to the application, which can then base its decisions upon this tailored high-level information rather than generic low-level performance measurements. While the middleware is concerned with many complex tasks, some lower level issues remain out of its scope: retaining the stability of the network, for instance, can be left up to the transport and network layer mechanisms that it chooses.

5 Implications

Clearly, introducing a layer above TCP/IP comes at the cost of service granularity — it will not be possible to specify an interface that covers each and every possible service and ideally exploit all available mechanisms without actually knowing which ones are available. On the other hand, this appears to be the only viable solution to the aforementioned long-standing deployment problems that may otherwise never be solved.

The fact that applications could transparently use new services as they become available — much like SCTP based TCP would provide multihoming without requiring to actually implement the feature at the application level — makes it possible to incrementally enhance implementations of the Adaptation Layer and thereby automatically further the quality attained by users. For instance, if a service provider provides QoS to its users (who are running Adaptation Layer enabled applications), there would be an immediate benefit; this is different from the situation of today, where someone would first have to write a QoS-enabled application that may not run in different environments. This could add a new dimension of competitiveness to the ISP market as well as the operating system vendor market.

The latter would be possible by the realization of sophisticated mechanisms in the Adaptation Layer — this could even become a catalyst for related research. Something like this has happened before: since the ATM Available Bit Rate service specification [18] does not comprise an actual algorithm to be followed in switches but only a set of rules that must be adhered to, it is possible to come up with new algorithms that do a better job than existing ones — this led to the large amount of research on ATM ABR Explicit Rate Feedback schemes we are facing today (some examples are [19], [20] and [21]).

In the case of the Adaptation Layer, related research would encompass the "Congestion Manager" [22] — the idea of realizing a single congestion control instance for a single pair of communicating hosts — and the question whether it would suffice to statically choose the right service or whether an Adaptation Layer should dynamically adapt to changing network characteristics. While it seems obvious that the latter would be more efficient, this brings about a great number of new questions, e.g., if the middleware switches between several TCP-friendly congestion control mechanisms depending on the state of the network, is the outcome still TCP-friendly? Despite these open problems, this seems to be a promising and new way of realizing congestion control.

Since the Adaptation Layer could be aware of the network infrastructure, it could also make use of TCP-unfriendly but more efficient transport protocols such as XCP [23], CADPC/PTP [24], HighSpeed TCP or FAST — an overview of such protocols can be found in [25].

6 Conclusion

In this paper, we have motivated and discussed the seemingly simple idea of placing middleware (an "Adaptation Layer") on top of TCP/IP; the goal of this concept is to provide a unique and sufficiently flexible interface to all kinds of Internet applications, thereby enabling gradual deployment of a wide range of network services. We believe that such an Adaptation Layer would have quite a positive impact on the evolution of the Internet as a whole.

However, the potential implications are numerous, making the development of such an Adaptation Layer rather tricky; especially the application interface should be designed with special care in order to maintain the greatest possible flexibility while ensuring the best possible service granularity. The following obstacles must be overcome before our middleware can have any serious impact:

- The interface should be standardized. Considering the broad range of open questions and issues that the Adaptation Layer brings about, it appears to be questionable whether the required community consensus can ever be reached.

- Once it is standardized, it will still take quite a long

time for it to become deployed in operating systems. As a motivating factor, it is therefore necessary to implement several prototypes and representative demonstration applications that can be shown to gain benefits by using the Adaptation Layer.

- Once the interface is available in the API's of operating systems, it will take a while for application programmers to use it — the well-known TCP/IP API will obviously have to be available as an additional alternative in order to support legacy applications. At this point, we must consider migration strategies: is it possible (and cost effective) to port the large base of existing TCP/UDP sockets based applications to the new model? Are there automatic, or at least half automatic ways of doing so?

It is easy to see that the goal of a unique middleware above TCP/IP can only be reached if the whole process is well prepared — this mandates progressing at a slow pace. At this point, our concept of what the middleware and its interfaces should look like is only very rough. Since carelessly adding layers has the potential to render a network technology inefficient, and due to the realization obstacles listed above, we considered it worthwhile to approach our goal in very small steps and start by closely examining the reasoning behind our idea as well as its possible implications and issues before taking the next step: the actual design.

References

[1] Marina Fomenkov, Ken Keys, David Moore and k claffy, "Longitudinal study of Internet traffi c in 1998-2003", CAIDA technical report, available from http://www.caida.org/outreach/papers/2003/nlanr/

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

[3] Sally Floyd and Kevin Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", IEEE/ACM Transactions on Networking, August 1999.

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

[5] Reza Rejaie, Mark Handley, and Deborah Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", Proceedings of IEEE Infocom 1999, New York City, New York, 21.-25. 3. 1999.

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

[7] Deepak Bansal and Hari Balakrishnan, "Binomial Congestion Control Algorithms", Proceedings of INFOCOM 2001, Anchorage, Alaska, April 2001.

[8] Jörg Widmer, Robert Denda, and Martin Mauve, "A Survey on TCP-Friendly Congestion Control", IEEE Network Magazine, Special Issue "Control of Best Effort Traffi c" Vol. 15, No. 3, May 2001.

[9] Eddie Kohler, Mark Handley, and Sally Floyd, "Datagram Congestion Control Protocol (DCCP)", Internet-draft draft-ietf-dccp-spec-06.txt (work in progress), February 2004.

[10] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[11] L. Coene, "Stream Control Transmission Protocol Applicability Statement", RFC 3257, April 2002.

[12] Wu-chun Feng, Mark K. Gardner, Michael E. Fisk, Eric H. Weigle, "Automatic Flow-Control Adaptation for Enhancing Network Performance in Computational Grids", Kluwer Journal of Grid Computing vol. 1, issue 1 (2003) , pp. 63-74.

[13] Eric Weigle and Wu-chun Feng, "A Comparison of TCP Automatic Tuning Techniques for Distributed Computing", Proceedings of High Performance Distributed Computing (HPDC), Edinburgh, Scotland, 23-26 July 2002.

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

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

[16] M. Handley, J. Padhye and S. Floyd, "TCP Congestion Window Validation", RFC2861, June 2000.

[17] L-A. Larzon, M. Degermark, S. Pink, L-E. Jonsson, and G. Fairhurst, "The UDP Lite Protocol", *Internet draft* (work in progress) draft-ietf-tsvwg-udp-lite-02.txt, August 2003. Approved by the IESG for publication as Proposed Standard RFC.

[18] The ATM Forum Technical Committee, "Traffi c Management Specifi cation, Version 4.1", Contribution *AF-TM-0121.000*, March 1999.

[19] Aleksandar Kolarov and G. Ramamurthy, "A Control Theoretic Approach to the Design of Closed Loop Rate Based Flow Control for High Speed ATM Networks", *Proceedings of IEEE Infocom 1997*, Kobe, Japan, April 1997.

[20] Eitan Altman, Tamer Basar, and R. Srikant, "Robust Rate Control for ABR Sources", *Proceedings of Infocom 1998*, San Francisco, CA, USA, vol. 1 (pp. 166-173).

[21] Kuan Su-Hsien and Lachlan L. H. Andrew, "Performance of Fuzzy Logic ABR Rate Control with Large Round Trip Times", *Proceedings of Globecom 1998*, 2464-2468, Sydney, Australia, 1998.

[22] H. Balakrishnan and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.

[23] 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.

[24] Michael Welzl, "Scalable Performance Signalling and Congestion Avoidance". Kluwer Academic Publishers, 2003, ISBN 1402075707.

[25] Michael Welzl (ed.), Matthieu Goutelle (ed.), Eric He (ed.), Rajkumar Kettimut, Sanjay Hegde, Yunhong Gu, William E Allcock, "A Survey of Transport Protocols Other than Standard TCP", GGF draft (work in progress), Data Transport Research Group, February 2004.