

Improving the Performance of TCP on guaranteed bandwidth connections

Hartmut Ritter, Klaus Wehrle, and Lars Wolf

Institute of Telematics, University of Karlsruhe,
Zirkel 2, D-76128 Karlsruhe, Germany
Phone: +49 721 608 6411, Fax: +49 721 388097

{[ritter](mailto:ritter@telematik.informatik.uni-karlsruhe.de),[wehrle](mailto:wehrle@telematik.informatik.uni-karlsruhe.de),[wolf](mailto:wolf@telematik.informatik.uni-karlsruhe.de)}@telematik.informatik.uni-karlsruhe.de

Abstract. This paper discusses the performance of the Transmission Control Protocol under two aspects: First, the future Internet will provide some kind of service differentiation and bandwidth guarantees. Nevertheless, TCP connections often cannot fully exploit the reserved bandwidth over time. The congestion control mechanisms like slow start and congestion avoidance have to be revised in the context of guaranteed services, where no congestions occur for special flows.

Second, short-lived TCP connections have become more and more important in the Internet. Short-lived connections are needed in transaction-oriented network applications and client-server scenarios; a typical example is the common use of the World Wide Web. TCP has typical performance flaws in these cases.

Former approaches of improving the performance of TCP did not succeed because their objective was the adaption of all involved TCP stacks. This leads to a change in every TCP stack in the Internet – at least 80 millions hosts nowadays. In this paper another way has been chosen: only one host within a connection will be modified, particularly with regard to client-server communication. Within this context, several modifications to improve TCP's performance with given bandwidth guarantees and their evaluation will be presented in this paper.

1 Introduction

The provisioning of Quality of Service (QoS) has been intensively studied for ATM and IP networks. Whereas QoS support in ATM networks seems to be solved, in IP networks the technical details are not yet clear. The former Integrated Services Architecture [BrCS94] as well as the currently discussed Differentiated Services Architecture [NiJZ99] are still not broadly deployed in existing IP networks. Nevertheless, it seems to be obvious, that some kind of bandwidth guarantees will be available in the next generation Internet.

The transport protocol TCP, most likely to be used in the current Internet as well as in future networks, shows performance problems in an environment providing guaranteed bandwidth. These problems of TCP increase, when short-lived connections are considered, because the greatest loss of performance takes place in the early stages of a connection.

Short-lived TCP connections have to be taken more into account nowadays. On the one hand Web traffic gains more importance, not at least due to the economic interests in the World Wide Web. On the other hand, legacy applications formerly using different networks, like online brokerage or typically transaction-oriented ticket reservation systems, tend to be integrated into the Internet.

But using TCP in the next generation Internet, where on the one hand short-lived connections dominate the traffic and on the other hand, Quality of Service mechanisms are used to assure the bandwidth, today's TCP will lead to performance problems as the following sections will indicate. TCP can not exploit the entire bandwidth that has been negotiated for the connection. This is not in the end user's interest, above all he has to pay for the value-added IP service.

To solve this problem, the Transmission Control Protocol has to be modified according to the new requirements. Nevertheless, it is impossible to change a transport protocol, that is used in over eighty millions hosts. Furthermore, it is impossible to change the underlying transport protocol of the mostly used applications in the Internet from TCP to another (more QoS friendly) protocol from one day to the next. TCP is the most used transport layer instance. This is the status quo and will not change within the next decade.

Consequently, the focus of this paper is not the change of the TCP instances in all Internet hosts, but a modification of a small amount of them, which have strategic positions in the Net as described in the next section. In the opinion of the authors, this would solve the described problems in the most use cases.

1.1 The Internet is a client-server network

Client-server communication, where the bigger part of the traffic is sent from the server to the client, is the most used communication pattern in the Internet, e.g. DNS, SMTP, FTP, and above all the World Wide Web. Only a small amount of traffic is really load-balanced between the two peers, e.g. chat traffic, IP-telephony, interactive audio/video-conferences.

In the following, the focus is put on Web communication, because it forms the majority of today's Internet traffic. TCP connections in this scenario are typically very short. Nevertheless, Quality of Service will also be introduced into this environment, enabling users to select the quality of the server's response. A framework for such a QoS-enabled web server, with focus on different kinds of signaling for the desired service level between client and server, is discussed in [RiPW00].

In such a scenario, the TCP implementation of the server has to be modified anyway for guaranteed rate connections on the downlink from the server to the client, e.g. a traffic shaper has to be used for limiting the outgoing traffic to the negotiated rate, and the server can initiate the reservation on behalf of the client. So, it is possible to modify also the TCP stack in the server to improve the behavior of the transport protocol on the leased line. The TCP implementation in the client remains *unchanged*. This is a necessary condition

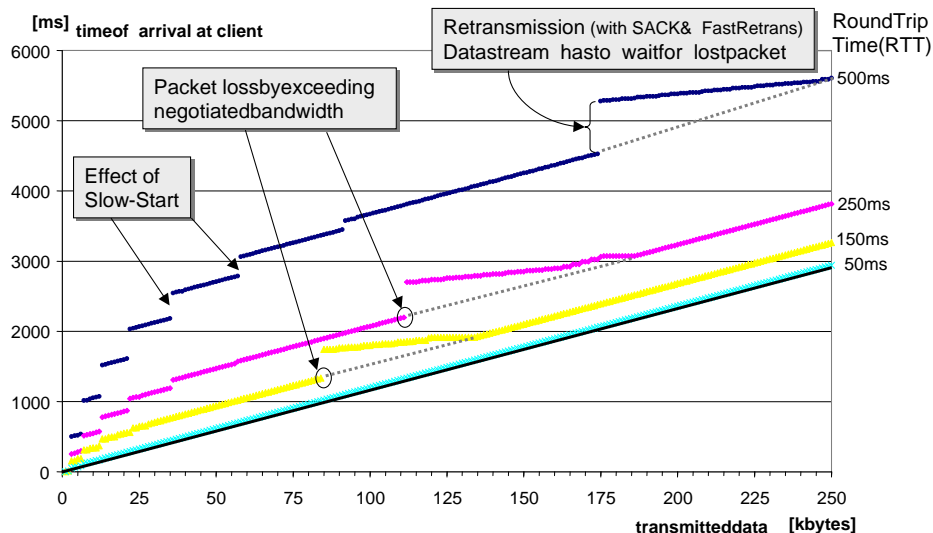


Fig. 1. Performance evaluation of a TCP connection using guaranteed bandwidth (with varying round trip times)

for a practical solution. The interoperability of modifications of TCP with legacy TCP implementations is therefore a main prerequisite for all approaches.

The next section discusses some reasons why TCP cannot use the whole bandwidth in the case of an environment providing bandwidth guarantees. The related TCP mechanisms will be discussed and analyzed in the context of short-lived connections and QoS-supporting networks. Section 3 provides a description of the testbed, describes the realized modifications of TCP and presents measurements. In section 4 the integration of the modifications in a single TCP implementation applicable in heterogeneous networks with and without guarantees is discussed.

2 Problems of TCP in a Guaranteed Rate Environment

TCP is one of the most frequently used transport protocols. It is almost twenty years old and several modifications have been integrated into TCP over the years, e.g., RFC1323. Moreover, many proposals have been made in order to enhance TCP (T/TCP (RFC1644), I/TCP [BaBa95], mobileTCP [BrSi97], etc.).

While the behavior of TCP in ATM networks with guaranteed bandwidth connections was studied by some research groups, e.g. [Bona98], [KJFG+96], the behavior of TCP in native IP networks with guaranteed bandwidth services has not been analyzed in detail so far. The problems of TCP in the context of guaranteed bandwidth services result from design decisions which target the

provisioning of fairness in the TCP protocol. This is due to the fact that in best-effort networks it is necessary that the protocol adapts to the current network load and slows down the sending rate in order to avoid resp. reduce congestion and avoid the starvation of other users flows.

The following two subsections describe the main components used within TCP to provide for this fairness. While they are necessary in best-effort networks, they result in a waste of bandwidth in a guaranteed bandwidth environment.

2.1 Congestion Avoidance in Non-Congested Links

Congestion situations occur due to a lack of bandwidth, i.e., if packets arrive faster at a node than they can be transmitted via a specific link, this link's output queue fill, and if the network load is not reduced in such a situation, more and more packets must be dropped. To prevent the Internet from this situation, mechanisms for congestion avoidance and control have been integrated into TCP - the Slow Start algorithm and the Congestion Avoidance algorithm [Jaco88].

When congestion occurs and the communication partners note the resulting packet losses, they reduce the amount of data they are sending into the network by applying the Slow Start and Congestion Avoidance algorithm. A typical characteristic of the Slow Start algorithm is the sawtooth behavior of the congestion window (which is also reflected in the measurements given in Fig. 1). The behavior of Slow Start and Congestion Avoidance can be described briefly as follows:

- TCP starts with a congestion window of one segment and increases this after a successful transfer of the whole window.
- The congestion window and thus the send rate is further increased. In the slow-start-phase this window opens up exponentially, after exceeding a slow-start threshold, TCP enters the congestion avoidance phase where the further increase of the window size occurs linearly. The slow start phase can be seen in Fig. 1. With bigger round trip times the effect can be seen better, because the acknowledgements needed for opening up the congestion window suffer from higher delay.
- If a packet has been lost, TCP returns to a very low transmission rate. This is not the case, if the SACK option is used, like it was done in the example of Fig. 1. The effect can be seen at the point of the curves, where a packet was dropped from the meter. In the case of SACK TCP the selective acknowledgement enables the sender to retransmit just the one lost packet. The packets sent after the transmission of the lost packet are cached on the receiver side and passed up to the receiving application not until the retransmitted packet arrives. As a result, the packets are passed to the receiving application in a burst with a higher rate.

These algorithms are necessary for the best-effort Internet in order to handle the offered traffic without collapse even if the principal transmission demand is much higher than the available bandwidth. Nevertheless, if a certain amount of bandwidth is guaranteed per connection, be it Expedited Forwarding (DiffServ)

or Guaranteed Service (IntServ), then no congestion occurs and no packets are lost as long as the traffic conforms to the negotiated traffic contract, e.g., if the offered load of the connection stays below its reserved bandwidth. The critical point is that the Slow Start and Congestion Avoidance algorithms are not aware of the guarantees given by the network to a flow.

There are two problems resulting from the slow-start and congestion-avoidance mechanisms:

- Because TCP does not know anything about the guaranteed rate available to a particular connection, it always exceeds this rate in the congestion avoidance phase when increasing the send rate linearly, i.e., it does not comply to the negotiated rate. As a consequence, packet loss occurs in a policing unit of the network such as the traffic shaper of a DiffServ router or a similar meter of a different network technology which enforces the compliance between the generated traffic on the one side and the traffic contract and the negotiated rate on the other side.
- After such a packet loss has occurred, TCP lowers its transmission rate for this connection, usually returning to the slow start phase. However, since a much higher constant rate has been reserved and guaranteed for this connection, the reduction of the transmission rate is basically just a waste of resources because the guaranteed rate is not fully exploited.

2.2 Window-Based Flow Control

In order to avoid buffer overflow at the receiver side, TCP implements a Flow Control mechanism. The TCP sender and receiver negotiate a maximum amount of packets which can be sent without being acknowledged. The original specification allows a window size of up to 64kbyte. The TCP modifications specified in RFC1323 allow for window sizes larger than this value. These modifications have been designed with the intention to better support so called 'long-fat pipes', i.e., networks with a large bandwidth-delay product. This value is an upper limit for the amount of bytes the sender is permitted to send without having to wait for acknowledgements from the receiver, provided the congestion window is open. In a network with a large bandwidth-delay product, the sender cannot exploit the full rate if the Flow Control Window is not large enough. The reason is that a fast sender has sent all packets it is allowed to transmit very soon and has to wait for the acknowledgement from the receiver. Especially the performance of short-lived TCP connections will be impacted, because these connections will perhaps never leave the slow-start phase. At least, the percentage of time a connection does not exploit the reserved bandwidth is extremely high. The option for large window sizes is already part of most modern TCP implementations, especially in a guaranteed rate environment the option should be activated and used.

In different TCP variants additional mechanisms and enhancements were added to the basic TCP. The mechanisms of slow start and congestion avoidance were introduced in TCP Tahoe [Jaco88]. In TCP Reno and New Reno

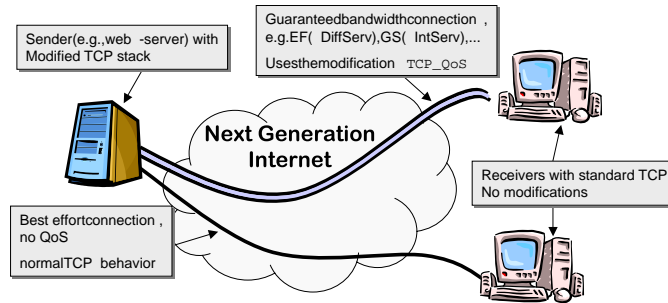


Fig. 2. Assumed scenario: TCP connections with QoS assurance can use modified TCP in the server. Normal connections are using standard TCP in the server (default). Clients always use standard TCP.

[FaF196], a modified Fast Retransmit [Jaco90] and a Fast Recovery algorithm were introduced, which avoid the slow start in some situations, when there is an obviously very short network congestion. However, during the slow start phase the available bandwidth is not used. TCP New Reno is part of the standard Linux operating system, which has been used for measurements in this paper. TCP Vegas [BrPe95] introduces ways to detect incipient stages of congestion before losses occur. These mechanisms can also be used in the slow start phase. Nevertheless, as there is no exact information about the available bandwidth, it takes time to adapt to the maximum rate. In a guaranteed rate environment a rate based flow control as proposed in this paper can skip this adaptation phase.

3 Scenario

Some assumptions concerning the QoS support and the overall scenario have been done in the presented work (cf. figure 2):

- The realization of QoS support in the network is not focused. Instead, it is assumed, that some kind of reservation is issued before a TCP connection is set up, and it is additionally assumed, that this reservation is strong and holds during the connection time.
- Reservation is done per flow, i.e. in the case of DiffServ each TCP connection is provided with an unique EF flow from end to end. Multiplexing of many TCP connections to one EF flow is not considered at the moment, dynamic adaptation of reservations neither.
- Best-effort and better services like EF in DiffServ or Guaranteed Service in IntServ co-exist in one network, but the bandwidth reserved per flow can not be borrowed to best-effort traffic if not exploited by the better services.

In this scenario, improving the performance of TCP is done in the context of the following objectives:

- Improving the performance of TCP means in the context of guaranteed bandwidth connections, that a TCP flow exploits the full rate which was reserved and is not hindered by internal mechanisms.
- Only *one* TCP stack in a peer-to-peer connection has to be modified. This is a prerequisite for an incremental deployment of the modified TCP in the future Internet. It is obvious that not each TCP stack can be modified at once. Client-server scenarios are the main paradigm nowadays, for example in the world wide web. Modifying the server's stack without a need for changes in all the client's stacks requires full interoperability of the improved TCP stack of the server and the legacy TCP stacks of the clients.
- The modifications should only be put into execution when the bandwidth is really guaranteed. Therefore the modifications should be activated only when the bandwidth has been previously reserved and confirmed.

4 Modifications of TCP in the server

As indicated in section 2, TCP is well designed for today's Internet with its best effort characteristic. But when guaranteed services should be used, the behavior of TCP hinders full exploitation of them. In order to achieve an improvement of performance in this environment, the TCP stack has been modified. The modification meets the objectives given in the previous section:

- *Modification of the slow start and congestion avoidance algorithms:* As described in section 2.1 the slow start and congestion avoidance algorithms prevent from the immediate exploitation of a guaranteed bandwidth. Due to service guarantees there will be no congestion during the connection. Consequently, an algorithm for congestion avoidance is not needed in this case. Therefore, a new TCP socket option `TCP_QoS` was implemented in the Linux kernel, by means of which the slow start algorithm in the sender is switched on or off. As can be seen in figure 4, described in the following section, this will avoid the slow increase of the TCP sending rate. Especially in the case of short-lived TCP connections the slow start algorithm results in the phenomenon, that these connections never reach the maximum rate or at least rest a high percentage of the connection life time in the slow start phase, far below the reserved rate. In the example shown in Fig. 1 the overall throughput for transmitting 100kbyte has been 37% with a round trip time of 500ms.
- *Rate based flow control:* To avoid bursty traffic leaving the TCP stack because of the window based flow control, the sending window was increased up to the path capacity and a rate based flow control algorithm was integrated, which sends out TCP segments in a smoothed way, conforming to the negotiated rate. Increasing the sending window to the product of bandwidth and round trip time effects that TCP can continuously send data with the negotiated rate without waiting for acknowledges from the receiver caused by higher round trip times (RTTs). In order to avoid exceeding the sending

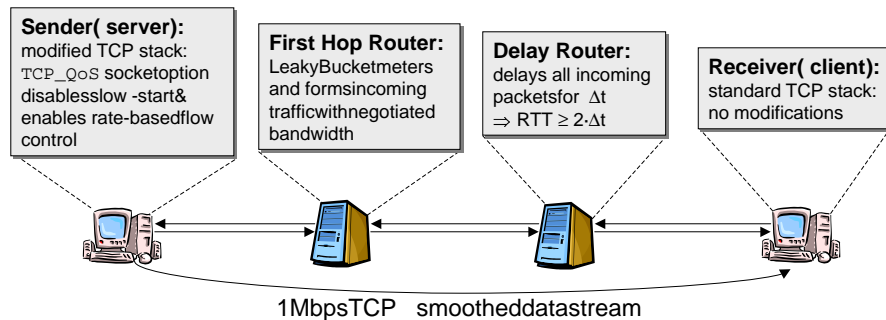


Fig. 3. Linux-Testbed with Differentiated Services- and Delay-Router

rate, the rate-based flow control is implemented by a token bucket model added to the TCP stack.

Supported by a fine-granular timer [RiWe00] the token bucket controls the outgoing rate in a byte-oriented way. The sending rate can be set as parameter of the socket option `TCP_QoS` mentioned above.

By implementing just these small modifications in one of the communication partners (in this case the server) the performance of the Transmission Control Protocol was improved up to nearly the maximum throughput. This has been proved by the measurements presented in the following section.

5 Evaluation of the performance improvement

The measurements and modifications of TCP have been done in a testbed environment illustrated in figure 3. Four standard PCs (Pentium III, 450MHz, 256 MB RAM) and full duplex 10BaseT network connections were used. TCP client and server are connected with a Differentiated Services router (using the KIDS implementation described in [BIWe99]) and another router realizing a delay queue. The DiffServ router implements a Traffic Shaper [RiWe00], based on a Leaky Bucket, as commonly used for limiting the bandwidth used by a flow. The delay queue in the second router just delays all incoming packets for a given amount of time in order to simulate different round trip times. The combination of DiffServ- and Delay-router therefore allows the testing of the behavior of TCP in networks with different bandwidth-delay products.

Several measurements were done with round trip times increasing from 10ms up to 500ms. In this paper only measurements with 500ms are presented, because they show most clearly the achieved performance improvement. The effects of smaller RTTs scales down linearly depending on the round trip time. In each measurement the server sent a smoothed TCP data stream with the negotiated rate to the receiver. In the QoS-Router an Expedited Forwarding [BHCD⁺97]

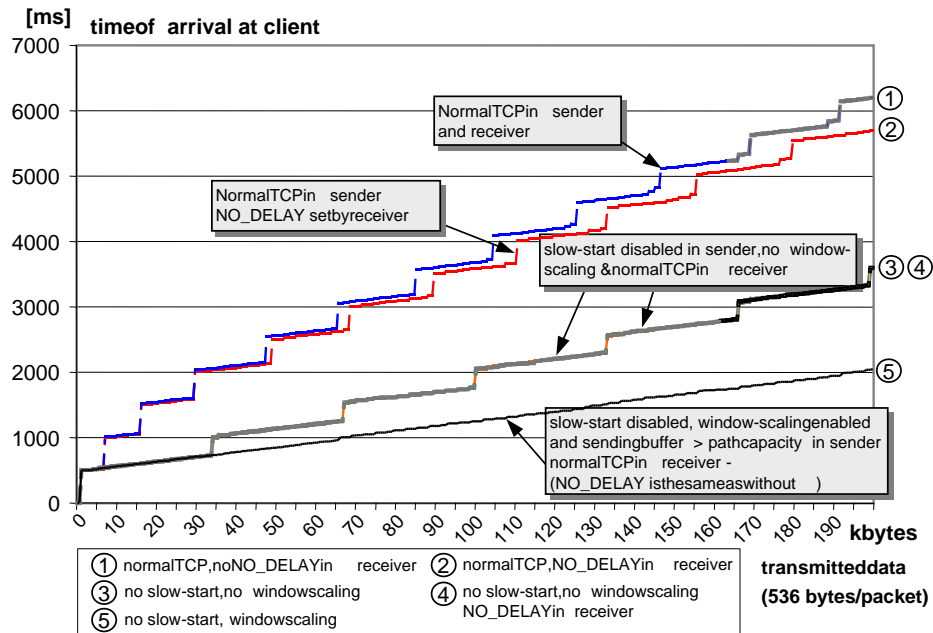


Fig. 4. Evaluation of modified TCP's performance with a round trip time of 500ms.

traffic profile with a reservation of 1,1 times of the sending rate has been installed with a burst size of 50.000 byte each.

Figure 4 shows different measurements done in this testbed with a RTT of 500ms. On the x-axis it shows the amount of data that has been sent downlink and received by the Host B. The y-axis shows the arrival times of the TCP segments (each about 536 bytes) of a TCP connection between client and server. The uppermost curve (curve 1) shows the normal behavior of a non-modified TCP. The typical behavior of TCP can be seen here: The fast arrival of a bunch of packets, followed by a time where no packets arrive due to the described slow start mechanism.

Curve 2 shows an also un-modified TCP where the socket option `NO_DELAY` was enabled in the client. Normally this delay allows the receiver to aggregate acknowledgements in order to prevent him from sending too much (small) acknowledge packets. This slows down the sender, but as it can be seen the throughput does not differ much. Curve 3 in figure 1 shows the behavior of a modified TCP, the slow start being disabled. Disabling the slow-start means here, that the congestion window of TCP is opened up to a threshold set by the new socket-option from the beginning of the connection. This socket option informs the TCP stack, that a guaranteed bandwidth exists for this socket, and that slow start is not needed. As can be seen, the throughput of this connection is higher due to

the fact that TCP starts with a higher sending rate. Interestingly, enabling the NO_DELAY-Option does not have a significant effect here.

Curve 5 finally traces a TCP connection, where the slow-start was disabled and the negotiation of a big window size was enabled. Enabling a window size that is bigger than 64kbyte is possible with the window-scale option defined in RFC 1323. The window size should be as large as the double of the bandwidth delay product (path capacity):

$$window\ size \geq 2 \cdot bandwidth_{negotiated} \cdot delay \quad (1)$$

In this case the biggest throughput is reached. In the measurement of Fig. 4 the TCP throughput is three times larger as with normal TCP and the negotiated bandwidth is exploited completely. As can be seen also, the rate based flow control hinders TCP from exceeding the negotiated bandwidth, although the sending windows allows more data to be sent.

The measurements in Fig. 4 have proven that the proposed solution of modifying only one peer in a TCP connection can increase the exploitation of guaranteed bandwidth up to the maximum. But this solution does not work always. It was assumed that the client in the described scenario has to be left unchanged. But when the path capacity exceeds the value of 64kbyte, the connection between the two peers should use the window-scale option to increase the window size. When the receiver is not accepting the window-scale option of RFC 1323 it can only be reached a throughput of

$$throughput_{effective} = \frac{negotiated\ bandwidth}{64\ kbyte} \quad (2)$$

In the example of Fig.4 the throughput of curve 3 is only $512\ kbps = \frac{1\ Mbps}{32\ kbyte}$ because the client has rejected the window-scale option and offered only a window-size of 32kbyte (this has been manually set to the maximum window-size to avoid the automatic acceptance of the window-scale option). In curve 5, where the window-scaling has been accepted the throughput reached the maximum possible amount of $1\ Mbps$, because the window size could have been increased up to the path capacity. Most commonly used operation systems support the window-scale option (as required by RFC 1323), so this problem can be neglected in the future Internet.

6 Integration of the Modifications in a Universal TCP

The measurements show that disabling slow-start and using big window sizes allows a throughput up to three times better than with an unmodified TCP (New Reno) implementation. As said before, the problem of unfairness is not relevant in a guaranteed rate environment, because the usage of the reserved rate does not influence other flows.

However, using this modified TCP in today's best-effort Internet would create a very aggressive behavior and unfairness of this TCP connection in relation to

connections of unmodified TCP implementations. Even in future QoS-supporting networks there will be a great amount of best-effort traffic. Thus, there is the need for a single TCP implementation which adapts its behavior, depending on the service guarantees the flow requires and the environment provides. The presented implementation realizes a single TCP stack, the modifications can be switched on or off at the socket interface, thus per flow, as described above. The only parameters needed for the modified TCP to work sufficiently are the reserved rate and the round trip time. Whereas the round trip time can be calculated by built-in TCP mechanisms at connection setup, the information about the reserved rate must be gathered from components outside of TCP.

7 Conclusion

This paper discussed the performance of TCP on guaranteed bandwidth connections provided by native IP mechanisms. The presented measurements illustrate that modifications of TCP are needed in order to make TCP more efficient in such an environment. The modifications discussed and proposed in this work consist of disabling the slow start and congestion avoidance algorithms for such connections and introducing a rate-based flow control.

For any modification to this dominant protocol it has to be considered how the changes affect the network in general and the deployed implementations. The mechanisms presented in this work must be integrated into a TCP entity which must be capable of supporting concurrently guaranteed bandwidth and regular best-effort network connections in an efficient manner. A basic requirement for the incremental deployment of such methods is the backward compatibility of the proposed modifications, i.e. that a server running a modified TCP stack can communicate with a client running a legacy stack. The mechanisms introduced in this work ensure both, the ease of deployment and the interoperability requirements because only one side of a connection must be changed, e.g., a web server, and the introduced changes are only local modifications, i.e., the peer is not affected.

The proposed modifications are especially well suited for web traffic where short-lived connections are prevalent and where the standard TCP mechanisms lower the reachable throughput. Not at least due to the increasing commercial and non-commercial interest in the Internet, the integration of QoS and web traffic is a very important issue. This integration requires further work such as to study how future QoS-architectures and existing components interoperate and can be better integrated. Elements in the end system which provide TCP with further information about the currently provided and needed QoS level will be necessary as well as information from the network about the load situation. Therefore, the presented modifications of TCP are currently integrated in an adaptive QoS architecture in the end system presented in [BeRS99].

References

- BaBa95. A. Bakre and B. Badrinath. I-TCP: Indirect TCP for mobile hosts. Proceedings of the 15th International Conference on Distributed Computing Systems (ICDCS), May 1995.
- BeRS99. M. Bechler, H. Ritter and J. Schiller. Integration of a Traffic Conditioner for Differentiated Services in End-systems via Feedback-loops. In *Proceedings of Broadband Communications*, Hong Kong, November 1999. IFIP.
- BHCD⁺97. F. Baker, J. Heinanen, M. Carlson, E. Davies, Z. Wang and W. Weiss. An Architecture for Differentiated Services. RFC 2475, IETF, June 1997.
- BlWe99. R. Bless and K. Wehrle. Evaluation of Differentiated Services using an implementation under Linux. In *Proceedings of the 7th IEEE/IFIP Workshop on Quality of Service (IWQoS99)*, London, October 1999. IEEE.
- Bona98. O. Bonaventure. A simulation study of TCP with the GFR service category. In *High-performance networks for multimedia applications*, Boston, October 1998. Kluwer Academic Publishers.
- BrCS94. R. Braden, D. Clark and S. Shenker. Integrated Services in the Internet Architecture: an overview. RFC 1633, IETF, June 1994.
- BrPe95. L. S. Brakmo and L. L. Peterson. TCP Vegas: End to End Congestion Avoidance on a Global Internet. *IEEE Journal on Selected Areas in Communication*, Vol. 13, No. 8, October 1995.
- BrSi97. K. Brown and S. Singh. mobileTCP: TCP for mobile cellular networks. *ACM Computer Communications Review* 27(5), May 1997.
- FaFl96. K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. In *Computer Communication Review V.26 (N. 3)*, New York, July 1996. ACM.
- Jaco88. V. Jacobson. Congestion avoidance and control. In *Proceedings of SIGCOMM 1988*, Stanford, August 1988. ACM.
- Jaco90. V. Jacobson. Modified TCP Congestion Avoidance Algorithm. In *Technical Report*, available at <ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt>, Berkeley, April 1990. LBNL.
- KJFG⁺96. S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, F. Lu and S. Srinidhi. Performance of TCP/IP over ABR. In *Proceedings of Globecom 1996*, London, November 1996. IEEE.
- NiJZ99. K. Nichols, V. Jacobson and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet. RFC 2638, IETF, July 1999.
- RiPW00. H. Ritter, T. Pastors and K. Wehrle. DiffServ in the Web: Different Approaches for Enabling better Services in the World Wide Web. In *Proceedings of Networking 2000 (joint conference of Broadband Communications, High Performance Networking and Performance of Communication Networks)*, Paris, May 2000. Springer.
- RiWe00. H. Ritter and K. Wehrle. Traffic Shaping in ATM and IP networks. In *Proceedings of Intern. Conference on ATM*, Heidelberg, June 2000.