Technical University Berlin

Telecommunication Networks Group

Identifying and Evaluating the Potential of Reusing Network Information from Different Flows

Michael Savorić

savoric@ee.tu-berlin.de

Berlin, December 2001

TKN Technical Report TKN-01-019

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

In an internet end system, many applications with various requirements on their data stream transmission run simultaneously. The management of these data streams in a sending end system, and particularly the flow and congestion control for the data streams (if they exist at all), is for each data stream separate and independent from all other data streams.

An interesting alternative to such a separate data stream management in a sending end system might be a common data stream management for all data streams in a sending end system. This common data stream management consists of two different aspects improving the overall performance, i.e., the throughput and the fairness, of the data streams of an end system. First, reuse of information about the current network conditions for an accurate adjustment of connection parameters of other data streams in an end system. This information can be implicitly obtained by some other data streams of the end system and/or explicitly obtained by a network performance probing approach from the end system. And second, a common congestion control between all data streams of an end system including TCP and UDP traffic.

In this work some existing approaches for network information reuse, network performance probing, and a common congestion control are regarded and the main ideas of these approaches are extracted to obtain the underlying principles of these approaches.

Based on this groundwork one of these approaches or a combination of some ideas of these approaches possibly combined with new ideas is selected as a case study to show the potential of a common data stream management compared to the standard data stream management of TCP and UDP data streams in an end system. This comparative investigation will be performed by simulations of an appropriate simulation model which must be developed and implemented by using a well-known simulation tool called network simulator version 2 (ns-2).

Contents

Chapter 1

Introduction

In an internet end system many applications with various requirements on their data stream transmission run simultaneously. The management of these data streams in a sending end system, and particularly the flow and congestion control for the data streams (if it exists at all), is for each data stream separate and independent from all other data streams.

An interesting alternative to such a separate data stream management might be a common data stream management for all data streams in a sending end system. Some scenarios where such a common data stream management with regard to the two existing internet transport layer protocols TCP and UDP might be advantageous are:

- During the lifetime of a TCP connection the TCP sender obtains useful information about the network conditions in the path to the receiver, e.g., the (smoothed) round trip time to the receiver or the current possible congestion window size. This information can be used for a new TCP connection from the same end system or an end system in the same subnet to the same receiver or to a receiver in the same subnet to calculate more appropriate values than the standard initial values of some of the TCP variables, e.g., the (smoothed) round trip time or the congestion window size. Thus the transient and inefficient start phase of the new TCP connection will be shortened and therefore the overall throughput of the TCP connection will be increased.

  If no TCP connections are yet established when a new TCP connection starts to send or the information obtained by previous TCP connections is out-dated then the end system can find out the current network conditions in the path to a receiver by using an active network performance probing approach.

- If two or more TCP connections are established to the same receiver or to receivers in the same subnet in parallel, then these TCP connections should have a common congestion control to reach a better fairness and to obtain a higher overall throughput. This common congestion control would also avoid the well-known situation where two or more parallel standard TCP connections in an end system behave substantially more aggressive in bandwidth consumption in the network than one TCP connection in an end system does.

- In contrast to TCP UDP has no built-in congestion or flow control. But due to fairness reasons every UDP sender should behave TCP-friendly, i.e., a UDP sender should not be more aggressive in bandwidth consumption than a TCP sender. One possibility to achieve this behavior of a UDP sender is to use the network feedback information obtained by established or recently closed TCP connections of the common data stream management to control the sending rate of a UDP sender. If no network information is available the sending rate of the UDP sender should be controlled by using other algorithms that guarantee the TCP-friendliness of a UDP sender.

Looking at these examples, two main issues in common data stream management systems are apparent:

- Reusing information about the current network conditions obtained from some existing or recently closed data streams and/or network performance probing approaches for a quicker or more accurate adaption of the initial values of the traffic or congestion control variables of other data streams.
- Providing a common flow and congestion control in an end system for all types of traffic. This includes the important task to make UDP streams TCP-friendly.

In the next four chapters for both of these issues some existing approaches are described. In Chapter 2, the network information reuse approach TCP control block interdependence (TCBI) is described. Chapter 3 considers the common congestion control approach ensemble-TCP (E-TCP). In Chapter 4, the passive network performance probing approach ”shared passive network performance discovery” (SPAND) is portrayed. And in Chapter 5 the main ideas of another common congestion control approach called congestion manager (CM) for data streams of different applications in an end system are described. A summary of the main features of all considered approaches can be found in Chapter 6.

Chapter 2

TCP Control Block Interdependence

2.1 Introduction

In the sender of a TCP connection the variables and parameters of the TCP connection are stored in a memory block called TCP control block. These variables and parameters, e.g., the congestion window size, the slow start threshold or the (smoothed) round trip time, are very important for the accurate and protocol-conform working of the TCP sender. Together the variables hold the information about the connection and network status obtained by the network so far on a per-connection basis (see Figure 2.1).

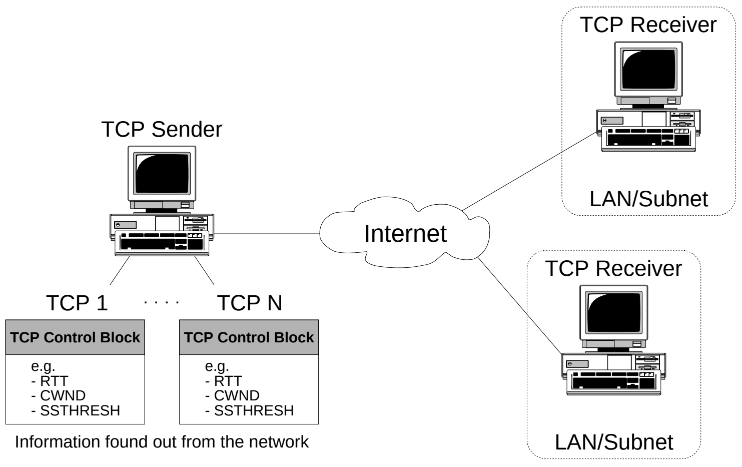

Figure 2.1: The TCP control blocks of a TCP sender

During the lifetime of a TCP connection this information is continously updated.

It might be — at least — an interesting idea to use the stored information of one TCP connection for other TCP connections in order to improve their performance, i.e., the fairness and throughput. This is the idea behind the TCP control block interdependence (TCBI) [5].

In this text a TCP sender with TCBI capabilities is called a TCBI TCP sender or, briefly, a TCBI sender. The TCBI approach is only reasonable between TCP connections of a TCBI sender which have the same TCP receiver or at least TCP receivers in the same subnet of the network. These TCP connections form a TCBI TCP connection set or, briefly, a TCBI connection set.

Which of the information of other TCP connections belongs to the same TCBI connection set and how this information is used for a new TCP connection of the TCBI connection set is dependent on the implementation of the TCBI controller in the TCBI sender.

The TCBI can be used not only for adjusting the connection variables of new TCP connections. In a scenario with mobile TCP receivers the TCBI can be used after a handover of a mobile TCP receiver for two reasons:
First and similar to the TCBI of a new TCP connection, to increase the throughput and fairness of the TCP connection by using a higher (initial) congestion window size after the handover if possible. And second, to reduce the probability of a congestion in the new subpath of the TCP connection between the handover switching node in the network and the new base station by using a smaller congestion window size than the current congestion window size if necessary. The latter case is the more important one from the network point of view, since a congestion negatively influences the throughput of all TCP connections over the congested path.

The next two sections summarize the potentialities and difficulties in using the TCBI approach in fixed and wireless networks.

2.2 TCP control block interdependence in fixed networks

In the following list some ideas about the feasibility and usability of the TCP control block interdependence in fixed networks are mentioned:

- If the TCP connections of a TCBI sender have different TCP receivers how fine should the granularity be chosen to determine that two or more of these TCP connections have TCP receivers in the same part of the network and can form a TCBI connection set?

  If the precise segmentation of a class A, B, or C network in subnetworks is not known, only the network part of a class A, B, or C network address can be used for that. But this might be too inaccurate, since, for example, a whole class B network usually consists of many LANs with different currently available bandwidths. The same problem occurs if some of the TCP receivers are located behind a router with network address translation (NAT) capabilities.

- The TCBI can be used not only between TCP connections which exist in parallel (ensemble TCBI), but also between a new TCP connection and one or more TCP connections which were recently closed (temporal TCBI). In some TCP implementations, e.g., in the Linux kernel, the control block of a terminated TCP connection further exists for a short period of time.

  But how long can the duration be between a closed TCP connection and a new TCP connection that the information obtained from the network so far is still meaningful and not totally out-of-date and even detrimental for the performance of the TCP connections?

  Measurements [1] have shown that with a proper observation method for the available bandwidth on a path seen by one TCP connection the value for the observed bandwidth can be used as a fairly good prediction for the future available bandwidth for other TCP connections on the same path for a period of time on the order of tens of minutes.

- The information exchange between TCP connections of a TCBI connection set can be done once, e.g., at the start of a new TCP connection, or dynamically during the whole lifetime of the TCP connections in the same TCBI connection set. This decision can depend on the TCP variables for which the TCBI is used, e.g., for all TCP connections of a TCBI connection set only one value for the (smoothed) round trip time or the derived timeout timer seems to be reasonable. Another possibility is to use an aggregate congestion window size and share it (dynamically) over the TCP connections of the TCBI connection set. A new part of the TCBI controller, potentially with different priority classes for the TCP connections of the TCBI connection set, is able to assign the aggregate congestion control parameters to each of the TCP connections in the TCBI connection set. This is similar to the ideas of the TCP implementation in the ensemble-TCP approach [6] which is described in detail in the next chapter.

- The TCBI might be used to shorten the transient start phase of a TCP connection with small congestion window sizes due to the TCP slow start algorithm. Particularly for short TCP connections with only a few segments to send the TCBI might be very useful, since for each of these TCP connections the transient start phase is a large part of the whole lifetime of this TCP connection and has a high and negative influence on the overall throughput of this TCP connection.

- Although the TCBI might be very useful for short TCP connections, for an accurate collection of information about the network the TCBI sender should have many TCP connections in parallel or one after another at short intervals and at least some of them should be longer TCP connections with dozens of segments to be transmitted. Only those longer TCP connections have a chance at all to increase their congestion window size to a value that represents the currently available bandwidth in the path to a receiver so that shorter TCP connections can benefit from it if they use the TCBI.

- Only those TCP connections can benefit from the TCBI which have a sufficient long round trip time between the TCBI sender and the TCP receiver and therefore can make use of a higher (initial) congestion window size. Other TCP connections with a short round trip time, e.g., intra-LAN TCP connections, cannot take advantage of a higher (initial) congestion window size, since the TCBI sender can send only one segment and a small fraction of another segment during this round trip time.

In general, one TCBI sender rarely has as many TCP connections that the TCBI can be used by expecting a noticeable increase of the mean throughput of all of its TCP connections. But, for example, a highly frequented WWW server or a proxy server with TCBI capabilities have a much higher probability to use the TCBI for increasing the mean throughput and the fairness of all their TCP connections.

2.3 TCP control block interdependence in wireless networks

In the following list some additional ideas about the usage of the TCP control block interdependence in networks with at least one wireless link, e.g., the last hop to TCP receivers in a wireless LAN, and with mobile TCP receivers are mentioned:

- For a standard TCP sender in a wired network an observed packet loss is an indication of congestion in the network. Therefore, it drastically reduces its sending rate by decreasing the congestion window size and uses the slow start algorithm to find out the current possible load for the network. This reaction of a TCP sender is not appropriate for a TCP connection over at least one wireless link, since in general most of the observed packet losses of such a TCP connection occur in the wireless link and are not caused by congestion in the network (in a wireless link the bit error rate (BER) is several orders of magnitude higher than in a wired link, i.e., in a wireless link a packet or acknowledgment loss occurs more often than in a wired link).

  If the TCP control block interdependence is used for a TCP connection over at least one wireless link, the described reaction of a standard TCP sender on packet losses in the wireless link can vitiate the whole advantage of the TCBI approach, since a higher congestion window size at the beginning of a TCP connection can be rapidly reduced to one (or two in current TCP implementations) after observing the first packet loss.

  For TCP connections in a wireless network the TCP control block interdependence should be combined with other mechanisms designed for a better performance for TCP over wireless links, e.g., the remote socket architecture (ReSoA) [4]). Another interesting idea is to enable the TCP sender to distinguish between packet losses caused by congestion in the network from packet losses in the wireless link(s). Then the TCP sender can adequately react on packet losses and no TCP throughput collapse will occur.

- In the context of this text the mobility of a TCP receiver means that the receiver is mobile during the lifetime of a TCP connection, i.e., the mobile receiver has to perform a handover from its current base station to its new base station. For a handover TCP connection the TCP control block interdependence should be used for two reasons. First and similar to the TCBI of a new TCP connection, to increase the throughput of the TCP connection by using a higher initial congestion window size if possible, and second, reducing the probability of congestion in the new subpath of the TCP connection by using a smaller congestion window size than the current congestion window size if necessary. The latter case is the more important one from the network point of view, since a congestion negatively influences the throughput of all TCP connections over the congested router or the congested (new) base station.

- The TCP control block interdependence can only be used if a TCP sender has the capability to detect a handover of a mobile TCP receiver, i.e., the handover must be non-transparent for the TCP sender. The handover detection can be done in an indirect way, e.g., by using the IP path discovery mechanism if there is enough time for it. But for some well-known mechanisms to achieve mobility in IP-based networks, e.g., mobile IP, this does not apply. Another reason against non-transparent handovers is that at any time the TCP sender has the knowledge of the current location of the mobile TCP receiver. But these are rather secondary aspects compared to the advantage a non-transparent handover has: Only a non-transparent handover can trigger a mechanism in the TCP sender which can avoid congestion in the new subpath of the TCP connection due to a handover of the mobile TCP receiver.

- A non-transparent handover can be interpreted as a two-step operation. In the first step the TCP connection to the old location of the mobile TCP receiver is closed. In the second step a new TCP connection to the new location of the mobile TCP receiver is opened. These two steps can be done in parallel to reduce the handover latency time.

For the new TCP connection of a non-transparent handover two scenarios are possible. In the first scenario the TCP sender starts sending segments following the last acknowledged segment of the old TCP connection before the handover has occurred. This additional feature requires some minor changes in the existing implementation of a TCP sender to exchange the state information of the old TCP connection with the new TCP connection. In the second scenario the TCP sender starts sending segments from the beginning of the old TCP connection. In this scenario also an application-controlled behavior of the TCP sender which avoids the sending of already acknowledged segments is conceivable.

If a TCP sender with TCBI capabilities has two or more TCP connections to (mobile) TCP receivers in adjacent cells of a wireless network than the TCBI can be used for a handover of one of these (mobile) TCP receivers into another cell of the wireless network where at least one of the (mobile) TCP receivers of the TCP sender is already located. This scenario is shown in Figure 2.2.

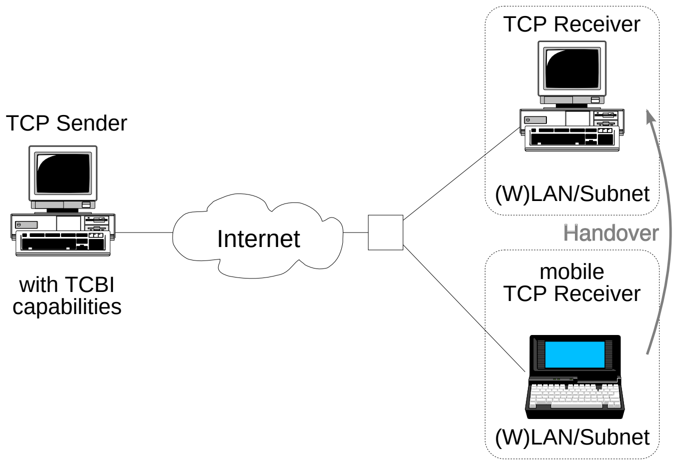

Figure 2.2: TCBI in a wireless network

Chapter 3

Ensemble-TCP

3.1 Introduction

The common congestion control of many TCP connections within an end system with the same receiver or two or more receivers in the same subnet is used to avoid the drawbacks of the following well-known behavior of TCP connections: N active TCP connections of an end system are roughly N times as aggressive to the network than one single TCP connection is. This behavior of many parallel TCP connections of an end system can lead to unfairness towards other TCP connections in the network. In addition, due to the aggressive behavior of parallel TCP connections the network can suddenly be highly congested. Then many packet losses occur, some or all of the parallel TCP connections have to undergo a slow start, and the overall efficiency of the data transmission will dramatically decrease.

One solution to that problem is the ensemble-TCP (E-TCP) [6] approach. In this common congestion control approach for TCP connections a group of parallel TCP connections with the same receiver or with receivers in the same subnet are combined to one TCP ensemble. Together with their connection status, i.e., their TCP parameters and variables, the TCP connections of a TCP ensemble have a common congestion control whose aggregated behavior and aggressiveness to the network is similar to the behavior and aggressiveness of a single TCP connection. Therefore, a single TCP connection of a TCP ensemble is less or at most as aggressive as a single standard TCP connection.

In addition to this ensemble TCP state sharing, the E-TCP approach provides a mechanism similar to the temporal TCBI where the network information obtained from previously closed TCP connections is reused for a new TCP connection to the same receiver or to a receiver in the same subnet. In this temporal TCP state sharing the E-TCP can be regarded as an implementation of the temporal TCBI.

Similar to the TCBI information reuse approach the same questions have to be answered for the ensemble-TCP approach:

- Which TCP connections form a TCP ensemble and share their TCP variables in order to form a common congestion control?
- Which TCP variables of the TCP connections of a TCP ensemble have to be shared and how should the TCP variables be updated and shared?
- If — similar to the temporal TCBI — also the cached last values of the variables of a previously

closed TCP connection are used for a better adjustment of the initial parameters of a new TCP connection, how should the cached values of the TCP variables be used that they are helpful in improving the performance of the new TCP connection?

Except for some mechanisms supporting T/TCP all changes in a given TCP implementation providing the E-TCP approach are located in the TCP sender. Therefore, it is worthwhile to investigate the E-TCP approach as an example-implementation of a common congestion control algorithm for TCP connections in a sending end system. UDP data streams are not supported by the E-TCP approach. But with some minor extensions the E-TCP approach should be amenable to support UDP data streams and implement a common congestion control for all data streams of a sending end system.

3.2 The E-TCP design

One or more TCP connections of a sending end system with the same receiver or with a receiver in the same subnet are combined to a TCP ensemble. For each TCP ensemble one ensemble control block (ECB) is created in which the necessary parameters and data structures for a common congestion control of the TCP connections in the TCP ensemble are stored. The ECB consists of

- several TCP parameters aggregated from the original TCP control blocks (TCBs) of the TCP connections in the TCP ensemble, e.g., the (smoothed) round trip time, the variance in round trip time, the congestion window size and the slow start threshold,
- some new data structures and parameters, e.g., the list of the TCBs of the TCP connections in a TCP ensemble, the maximum segment size (MSS) supported in the path between the pair of end systems of the TCP ensemble, and a bit that specifies if rate-based pacing is used or not (with that mechanism the performance of restarted idle TCP connections can be improved [7]),
- a time-sorted list of previously sent but not yet acknowledged TCP segments of the TCP connections in the TCP ensemble together with an associated skipped ACK counter for each TCP segment in the list (this list is used for an adaptation of a TCP segment retransmission and congestion decreasing scheme originally designed for TCP-INT [8]),
- some other parameters necessary only for T/TCP connections which are not in the main scope of this paper.

There are two types of ECBs: active ECBs and cached ECBs. An active ECB has at least one active TCP connection associated with it. A cached ECB is not associated with a currently active TCP connection. It consists of ECB parameters from a TCP ensemble where the last TCP connection of this TCP ensemble has been previously closed. A new TCP connection to the same receiver or to a receiver in the same subnet can profit from the network information stored in a cached ECB, e.g., to shorten the transient start phase with a lower performance by using more adequate initial values of some of the TCP variables. To avoid out-of-date network information stored in a cached ECB the lifetime of a cached ECB should be limited, e.g., by using an aging algorithm for the network information stored in a cached ECB.

In each TCP ensemble a scheduler with four priorities is used, e.g., to give retransmitted TCP segments a higher priority over other TCP segments or to have the ability to share the whole possible sending rate of a TCP ensemble in different proportions between the TCP connections of a TCP ensemble.

In the implementation of E-TCP described in [6] the TCP retransmission timer is not aggregated, i.e., each TCP connection of a TCP ensemble manages its own TCP retransmission timer.

Under a specific scenario with HTTP/1.0-traffic and no background traffic in the network the simulations show a 17 %–61 % higher throughput for ensemble TCP state sharing of E-TCP compared to standard TCP. For temporal TCP state sharing of E-TCP the throughput gain is about 10 %–27 % compared to standard TCP. The developers of E-TCP feel certain that the idea of ensemble or temporal TCP state sharing in the E-TCP approach is also useful under other traffic conditions in the network.

Chapter 4

Shared Passive Network Performance Discovery

4.1 Introduction

If it is possible to predict the network performance, e.g., the available bandwidth, the latency, or the packet loss probability, between a pair of end systems many applications could benefit from such an approach. For example, applications can choose the server with the best current network performance from a list of servers that offer the same service, e.g., FTP or WWW mirror sites. Each of these applications should be able to determine in advance the expected network performance between pairs of end systems. Previous work in this research area has been mainly focused on active network probing mechanisms from individual end systems. These network probes are developed to measure, for example, the round trip time and the available peak or fair share bandwidth between two end systems by sending a burst of packets or some special ICMP echo packets from one end system to the relevant end system from which they are sent back to the sender. Depending on the queueing implementation in the routers on the path between the end systems the time intervals between the received echoed packets can be used to calculate the whole bandwidth or the available fair share of the bandwidth in the bottleneck link in the path.

But the active network probing approaches have three drawbacks. These network probing approaches increase the network traffic. Sometimes this extra network traffic can become a measurable fraction of the whole traffic which must be handled by some of the servers. In addition, active network probing mechanisms require some time for their probes and the evaluation of the probe results. And network probing from a single end system results in less accurate network information and more redundant network probes from nearby end systems than a network probing mechanism for a set of end systems. The shared passive network performance discovery (SPAND) [11] approach eliminates these disadvantages by using network performance measurements from a collection of end systems combined with passive network performance prediction algorithms. This approach can be used only if the network performance between two end systems is predictable precisely enough from earlier network performance measurements done by nearby end systems and for a longer duration. A recent study [1] has shown that the network performance between two end systems, i.e., one client and one server, can be stable for many minutes and very similar to the network performance observed by other nearby clients.

The SPAND approach is mainly developed for end systems working as clients to choose the server with the best current network performance from a list of servers that offer the requested service. In this scenario the available network information is (re)used on the receiver side of the data streams. Although the original SPAND approach is not designed for a common data stream management in sending end systems, the main mechanisms in SPAND of collecting network information and reusing this network information for other data streams are nevertheless interesting and valuable for a common data stream management in sending end systems.

In the next three sections we take a look at passive network probing approaches and at the SPAND approach. This look is done with regard to a common data stream management in sending end systems.

4.2 Passive network performance probing

There are two main reasons for passive instead of active network performance probing mechanisms. First, active network probing mechanisms require additional network traffic for their network probing. For example, some of the existing active network probing approaches require tens of kilobytes of probe traffic for only one probe. This amount of probe traffic is a significant fraction of the mean transfer size for many WWW connections. And second, an active-probing-based decision which of the possible servers to use for the upcoming data transfer needs some time since the network paths to all possible servers have to be probed one after another. A passive network probing approach avoids these drawbacks.

But passive network probing mechanisms also have one main disadvantage. They can only collect network performance information for a remote end system if one of the end systems is connected to the remote end system. To avoid out-of-date network performance information or no network performance information at all for an end system about remote end systems the end system has to establish connections to the remote end systems often enough.

In passive network probing mechanisms the collection of network performance information should take into account whether the network performance measurements are done by TCP- or UDP-based data streams, since for different transport protocols the network performance measurements can vary and lead to inaccurate network performance predicitions for other transport protocols than those which were used for the measurements.

4.3 Network information sharing

In general, one single end system is not connected frequently enough to the remote end systems of interest to measure the current network performance in the path to the remote end systems often enough such that a passive network performance probing approach can be reasonably used to predict the future network performance in these paths sufficiently exactly. An alternative to that is the collection of the current network performance measurements of many end systems in a local domain. This can be done in a centralized way by using a network performance server which is regularly informed by the end systems about their current network performance measurements. Then, in the network performance server a passive network probing approach can be used to find the remote end systems with the best current network performance. The network information obtained by the passive network probing approach in the network performance server can be shared over the end systems in the local internet domain, for example, by using a network information request/response scenario between the end systems and the network performance server.

The network information sharing should not be applied for end systems with different network connectivity, e.g., between end systems which are connected via modems to the internet and end systems in an ethernet-based environment, or between end systems with different TCP implementations. In these cases the shared network information might be too inaccurate.

4.4 The SPAND prototype

The architecture of the current SPAND prototype is shown in the following Figure 4.1.

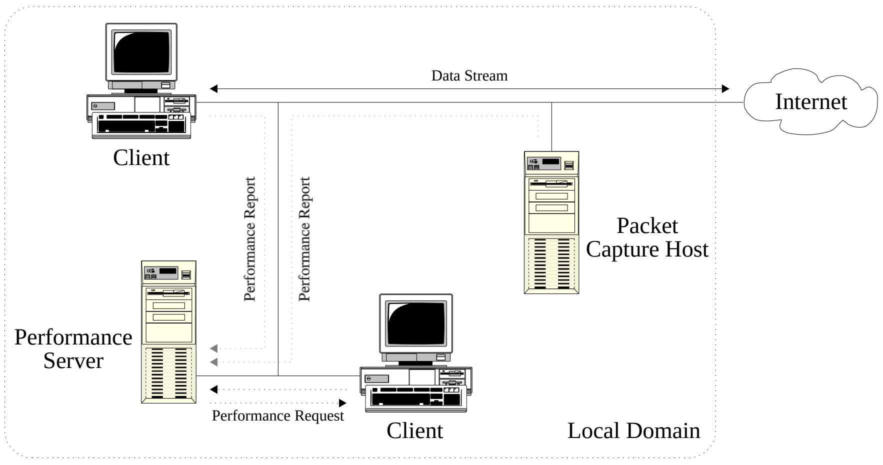

Figure 4.1: Architecture of the current SPAND prototype

It mainly consists of several clients with a modified network protocol stack and one performance server. The additional packet capture host is only used for network performance measurements of the data streams of those clients which are not able to do that on their own. The clients, the performance server, and the packet capture host form a local domain in the SPAND architecture.

All the network performance measurements of the clients and the packet capture host are sent in a performance report to the performance server of the local domain. In the performance server a passive probing mechanism is used to evaluate the performance reports and determine the current network performance in the network paths to the remote end systems.

When, for example, a client wants to use a service offered by two or more remote end systems it first contacts the performance server of the local domain using a performance request. If the performance server has network performance information about the remote end systems it informs the client accordingly. Now the client is able to choose the remote end system with the best current network performance. Otherwise, the client uses the default remote end system.

Depending on the number and frequency of performance reports received by the performance server, on the number of considered remote end systems, and on the runtime of the performance server, the fraction of answered performance requests might be high or low. Measurements in an industrial environment with the current SPAND prototype have shown that the performance server is able to answer 70 % of the performance requests within receiving the first 300 performance reports, and at around 10000 received performance reports the performance server reaches a steady state performance request answering rate of about 95 %. The measured accuracy of the answered performance requests shows that the predicted throughput of the performance server for a path to a remote end system is in 69 % of the answered performance requests within a factor of 2 and in 90 % of the answered performance requests within a factor of 4 of the real observed throughput (see [11]). The developers of SPAND hope that with some modifications in the performance capture host and in the performance server they will improve the accuracy of the SPAND performance estimations.

Chapter 5

Congestion Manager

5.1 Introduction

In a current internet end system each traffic flow is managed separately and independently from all other traffic flows of the end system. For efficiency reasons it might be a good idea to establish a common management framework for all traffic flows of an end system. Similar to the ensemble TCP approach for TCP connections the congestion manager (CM) [12],[13] provides a common management for all data streams of an end system to one receiver or to receivers in the same subnet from an end-to-end perspective. If in future internet infrastructures other service models than best effort are incorporated, e.g., differentiated/integrated services or other priority mechanisms for packets in the routers, a common congestion control of data streams of an end system based only on the receiver's IP address might be in general incorrect. In this case also the different a priori properties of the data streams have to be considered and a segregation of the different data streams of an end system must be performed.

The main design principles of the congestion manager are to put the applications in control and to support heterogenous data streams and applications of an end system. Many mechanisms of the CM need no additional information from the network beyond the existing mechanisms of the used transport protocols or the applications. But for some mechanisms of the CM an adaptation of the protocol stack in the receivers is necessary to support a new lightweight information feedback protocol.

Although the CM approach is neither transparent for the applications running on an end system and using the common management framework nor is it strictly located only on the sender side of the data streams, it is valuable to investigate the main ideas and mechanisms of the CM with regard to develop a transparent common congestion control for all data streams of a sending end system.

In the next section the architecture of the congestion manager is described. In addition, some performance results obtained by investigations of a CM prototype are stated.

5.2 The congestion manager framework

In the following Figure 5.1 the new protocol stack of a sender using the congestion manager is shown. The core of the congestion manager framework is the congestion manager itself. It collects network information and maintains network statistics which are used to control the data streams of an end system following robust control principles, e.g., the additive increase multiplicative decrease (AIMD)

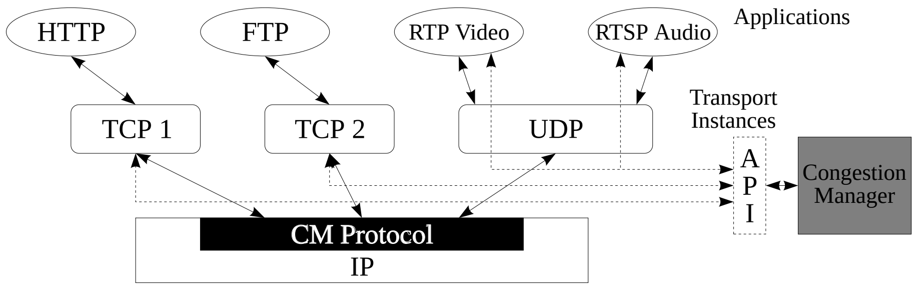

Figure 5.1: New protocol stack of a sender using the congestion manager

approach. The CM coordinates host- and domain-specific path information between the different data streams. The obtained path properties are shared between the data streams, since the transport protocols (TCP, UDP, and RTP) and some applications using the CM framework directly perform data transmissions only with the consent of the CM. Internally, the CM consists of the following parts:

- The congestion controller of the CM is implemented as a hybrid window- and rate-based scheme. A window-based AIMD mechanism similar to the existing TCP congestion control algorithm Newreno is used to control the overall data transmissions of an end system depending on the available bandwidth in the network. To reduce bursts of data transmissions of an end system a traffic shaper based on a rate estimation combining the current congestion window size and the smoothed round trip time is included in the congestion controller. If there is no feedback from the network, the congestion controller changes to an exponentially decaying rate-based scheme.

- The scheduler is used to distribute the available bandwidth to the existing data streams of an end system. With that approach priority mechanisms between data streams of an end system with different properties as well as fairness between data streams of an end system with equal properties can be obtained.

- Two forms of feedback from the receivers are used in the CM to adjust its congestion window size with response to packet losses due to congestion in the network: application notification (AN) and explicit feedback (EF).

  AN occurs when the applications or the transport protocols are able to send their own feedback information, e.g., packet losses or explicit congestion notification (ECN) from the routers, to the sender.

  EF is used for those applications or transport protocols which do not have an integrated feedback mechanism. In this case an explicit probing protocol from the CM to the receivers is used to obtain periodically feedback information, e.g., detect packet losses or other useful information, from the data streams of an end system on a per stream basis. This probing protocol should not require too much additional bandwidth in the network and should also be resilient to probing protocol message losses. The frequency of the probing protocol is determined to twice per round trip delay. Due to congestion in the network probing messages or responses can be lost, i.e., the sender has no acccurate estimation of the network state during times of congestion.

> In this case an exponential aging algorithm of the current network state is used that halves the sending rate of all data streams in an end system every so far measured minimum round trip time of the data streams during the duration of activity. This half-life period of the network state is chosen as a compromise between a too conservative and a too aggressive behavior to the network of the data streams of an end system.

The transport protocols and some applications access the features of the CM by using a new API incorporated with a new CM protocol located between the IP layer and the higher layers of the protocol stack in an end system supporting the CM.

The design of this new API is intended to enable the transport protocols and applications to adapt their sending rate to the current available bandwidth in the network paths to the receivers. Therefore, the API supports mechanisms to query path status, schedule data transmissions, and update variables depending on successful transmissions or congestion in the network. In addition, the API includes callback functionalities from the CM to the applications so that the applications can learn about the current network conditions and have the ability to decide which information to transmit and what relative fraction of the available bandwidth is used for a data stream. This feature, which is motivated by the end-to-end argument and the principle of application level framing (ALF), reduces the possibility to have congestion in the network paths to the related receivers. The CM can also learn from the applications, since the API provides a callback mechanism from the applications to the CM to update the network status parameters in the CM.

The new CM protocol is designed to take advantage of all features of the CM, e.g., the probing protocol to periodically obtain feedback information from the receivers. However, a CM sender is able to communicate with a receiver that does not support the new CM protocol and the CM. In this case, the CM in the sender works well only for those transport protocols and applications which provide feedback information to the sender. Whether a receiver supports the CM protocol and the CM or not can be determined by the CM sender with a two-way handshake protocol.

If the receiver supports the CM protocol and the CM, then every packet of the data streams of the sender is encapsulated in a new CM protocol data unit with a CM protocol header. This CM protocol header consists of a protocol field which contains the original protocol type identifier of the encapsulated packet. With the type field in the CM protocol header the different types of CM protocol messages, e.g., a probe or a probe response message, can be chosen. Each CM protocol header is marked by a flow ID that uniquely identifies the flow for which the CM protocol message is sent and a sequence number that reflects the incremental sequence number of the encapsulated packet in the flow.

An experiment that compares the throughput of four independent TCP Newreno connections with the throughput of four coupled TCP/CM connections shows the expected result that the overall throughput of the single TCP Newreno connections is 15 % higher than the overall throughput of the TCP/CM connections. But while the throughputs of TCP Newreno connections vary by a factor of 2.7 all TCP/CM connections have nearly the same throughput. Therefore, for TCP connections the CM improves the fairness and the performance consistency between TCP connections that the predictability of the throughput of a TCP/CM connection is much better than for a TCP Newreno connection.

Another experiment with one TCP/CM connection and an audio data stream shows that the audio application is able to react to changing network conditions by choosing encoding algorithms with diffferent coding rates that match the current throughput of the TCP/CM connection. The overall throughput of the TCP/CM connection and the audio data stream is equal to the throughput of a Newreno TCP connection.

Chapter 6

Summary

In the previous chapters the main ideas of four common congestion control and/or network information reuse approaches are described. The following Table 6.1 summarizes the functionalities and properties of these approaches. For comparison reasons, it is assumed that the functionalities of the original receiver-side located SPAND approach are adaptated to a sender-side located network information reuse approach similar to the three other approaches.

Table 6.1: Overview of the four network information reuse approaches

	TCBI	E-TCP	SPAND	CM
network information reuse	+	+	+	+
common congestion control	−	+	−	+
support for TCP/UDP	+/−	+/−	+/+	+/+
located only on the sender side	+	+	(+)	−
transparent for applications	+	+	+	−
complexity of the implementation	low	medium	high	high
no additional network traffic (e.g. probing)	+	+	−	−
performance gain (expected)	+	++	+	++

The overview shows that the E-TCP approach has some advantages over the other three approaches. The CM provides much more functionalities than E-TCP, e.g., the mechanism that applications can adapt their sending rate on the current network conditions, but at the cost of a much higher complexity of the implementation and a new API which changes the standard transport layer protocol interfaces.

Bibliography

[1] Paxson, V.: End-to-End Internet Packet Dynamics. IEEE/ACM Transactions on Networking, Vol.7, No.3 (1999) 277–292

[2] Floyd, S., Handley, M., Padhye, J., Widmer, J.: Equation-Based Congestion Control for Unicast Applications. ACM SIGCOMM 2000 (2000)

[3] Braden, R.: T/TCP – TCP Extentions for Transactions. RFC 1644 (1994)

[4] Schläger, M., Rathke, B., Bodenstein, S., Wolisz, A.: Advocating a Remote Socket Architecture for Internet Access using Wireless LANs. Journal of Mobile Networks and Applications (2001) 23-42

[5] Touch, J.: TCP Control Block Interdependence. RFC 2140 (1997)

[6] Eggert, L., Heidemann, J., Touch, J.: Effects of Ensemble TCP. ACM SIGCOMM Computer Communication Review, Vol. 30, No. 1 (2000) 15–29

[7] Visweswaraiah, V., Heidemann, J.: Improving Restart of Idle TCP Connections. Technical Report 97-661, University of California (1997)

[8] Balakrishnan, H., Padmanabhan, V., Seshan, S., Stemm, M., Katz, R.: TCP Behavior of a busy Internet Server: Analysis and Improvements. Proceedings IEEE Infocom'98 (1998)

[9] Savorić, M.: The TCP Control Block Interdependence: Some Performance Results. TKN Research Report Series (2001)

[10] Daniel, B., Nkweti, P., Wepiwe, G.: Implementing TCP Control Block Interdependence (TCBI) in Linux. TKN Research Report Series (2001)

[11] Seshan, S., Stemm, M., Katz, R.: SPAND: Shared Passive Network Performance Discovery. Proceedings USITS'97 (1997)

[12] Balakrishnan, H., Rahul, H., Seshan, S.: An Integrated Congestion Management Architecture for Internet Hosts. Proceedings ACM SIGCOMM'99 (1999)

[13] Balakrishnan, H., Seshan, S.: The Congestion Manager. Internet Draft draft-ietf-ecm-cm-02.txt, Work-in-Progress (2000)

[14] Reyes-Lecuona, A., González-Parada, E., Casilari, E., Casasola, J.C., Diaz-Estrella, A.: A page-oriented WWW traffic model for wireless system simulations. Proceedings ITC 16 (1999) 1271–1280

[15] Fall, K., Varadhan, K. (Editors): The ns Manual. VINT Project (2001)

[16] Jain, R.: The Art of Computer Systems Performance Analysis. Wiley & Sons (1991)