Congestion Control Mechanisms of Transport Protocols

Jiawei Chen
Helsinki University of Technology
jichen@cc.hut.fi

Abstract

Existing network transport protocols such as TCP and UDP have limitations when fitting into new technologies, includes, e.g., increased heterogeneity and mobility. In order to solve this problem, our attempt is to design a self-adaptive transport protocol, which can fit into the network condition dynamically according to the parameters given by upper layer and the network layer.

This paper analyzes the congestion control mechanisms of nowadays transmission protocols, and does comparison among them in many different aspect. I first describe the congestion control mechanism of traditional TCP, and then give out an overview and list the congestion function of nowadays transport protocols (such as TCP, UDP, DCCP, SCTP, CUBIC-TCP and CTCP). In the end, I do a simple comparison among all the protocols above, and point out the possible advantages and limitations of these protocols under different transport performances (RTT Fairness, TCP Fairness, Utilization Ratio, Packet Loss Rate, Smoothness of throughput, Convergence Time, etc.),

Future work will focus on applying our theoretical assumption into experimentation. We will decide the self-adaptive parameters of the protocol as well as further performance optimizations.

KEYWORDS: DCCP, SCTP, CUBIC-TCP, CTCP, Congestion Algorithm, RTT Fairness, TCP Fairness

1 Introduction

The traditional transport protocols such as Transmission Control Protocol (TCP) have served the Internet well for decades. Though it still works fine nowadays and probably will carry on in near future,it appears more strained to fit into the new environment [3]. For example, the congestion control mechanisms in TCP work well in wired networks but often over-react in wireless networks where packets can be lost due to factors other than congestion. Another example is in multimedia applications sharing networks, we need congestion control without ordered reliable delivery, which is not implemented by TCP or User Datagram Protocol (UDP). Such defects appear more and more obviously since new environment coming and being used, includes, e.g., increased heterogeneity and mobility [5][2]. The lack of appropriate guarantees or specific features has led to the widespread development of specialized protocols used in conjunction with or instead of standard transport protocols.

Traditional transport protocols that operate over unreliable battlefield networks provide an "all-or-nothing" choice for transport Quality of Service (QoS); either total order and reliability (e.g., TCP) or no guarantees at all (e.g., UDP) [10]. One attempt to overcome this is to combine some features of the two protocols together, for example, the Datagram Congestion Control Protocol (DCCP). DCCP is an unicast transport protocol that provides built-in congestion control. It is basically based on UDP but it offers applications a choice of modular congestion control mechanisms among a set of standardized algorithms, which orchestrates both TCP and UDP mechanisms. DCCP satisfies the real-time needs of various multimedia applications like video conferencing, IP telephony, audio and video streaming modifies the Internet network foundation.

Traditional TCP works well on the commodity Internet, but has been found to be inefficient and unfair to concurrent flows as bandwidth and delay increase[7]. Its congestion control algorithm needs a very long time to probe the bandwidth and recover from loss in high bandwidth delay product (BDP) links. Moreover, the existence of random loss on the physical link, the possible lack of a buffer on routers, and the existence of concurrent bursting flows all prevent TCP from utilizing high bandwidth. Furthermore, it exhibits a fairness problem for concurrent flows with different round trip times (RTT Fairness). Researchers have proposed a number of solutions to remedy the aforementioned problem. The most common way is to change its congestion window function, for example, Stream Control Transmission Protocol (SCTP), CUBIC Transmission Control Protocol (CUBIC TCP), and Compound Transmission Control Protocol (CTCP). These new solutions promise to improve TCP performance on high-speed networks significantly and are hence usually called high speed TCP variants.

In this project, our attempt to improve the performance of transport protocols is to turn it into an adaptive service, which can dynamically configure itself according to the policies given by the upper application layer and the context retrieved from the underlying network layer [3]. However,before being able to orchestrate such transport service, the components (quality attributes such as reliability, flow control, congestion control, etc.) to be orchestrated need to be understand first. Among them, congestion control algorithm seems to be the most important factor. This survey paper aims at doing a pre-analysis of the exiting transport protocols and finding out the effects that different congestion algorithms have on the performance issues.

In this paper, our main objective is to identify the different congestion control mechanisms of the existing protocols and finding out how they affect the performance in different aspects. In section 2, the traditional transmission control protocol (TCP) would be presented as an example to explain how congestion control mechanism works. Section 3 to section 6 list some other transport protocols used nowadays and describes their main features,including the motivation, congestion algorithm, advantages and limitations. As I had qualitative analysis and novel insights into the mechanisms utilized by those protocols, the comparison of these transport protocols would be discussed in Section 7. The future research and conclusion is given in Section 8.

2 Congestion Control Mechanism of TCP

In this section, the congestion control mechanism of traditional transmission control protocol would be discussed, which helps us to have a better understanding over other protocols.

First, we define transport layer flow control as any scheme by which the transport sender limits the rate at which data is sent over the network [10]. The goals of flow control may include one or both of the following:

(1) preventing a transport sender from sending data for which there is no available buffer space at the transport receiver, or

(2) preventing too much traffic in the underlying network.

Flow control for (2) also is called congestion control, or congestion avoidance. Congestion is essentially a network layer problem, and dozens of schemes are discussed and classified.

In TCP, there is a congestion window ($cwnd$) which determines the number of bytes that can be sent out at any time. This is used to prevent the physical link between two end points from getting overloaded with too much traffic. The sending rate should always be under the size of the congestion window.

TCP uses a slow-start mechanism at the beginning when connection established. For initializing the connection, every Route Trip Time, $cwnd = cwnd + 1$, which is exponential growth. Suppose the bandwidth is W, then it just takes $RTT * \log_2 W$ to fully occupy the bandwidth. Although it's called slow-start, actually it is not slow anymore. As we can see, through slow-start, the congestion window can reach to a large value quickly, but obviously it cannot keep growing. TCP set a threshold for the slow-start session. When the size of congestion window has reached the value, it comes into the congestion control session. In this session, if all segments are received and the acknowledgments reach the sender on time, the window size will increase steadily but slowly. The window keeps growing linearly until a timeout occurs or the receiver reaches its limit. If a timeout occurs, the window size will drop dramatically. For congestion control algorithm, TCP use:

$$Ack : cwnd = cwnd + \frac{a}{cwnd}$$

$$Loss : cwnd = cwnd - b \times cwnd$$

among them, $a = 1$; $b = 0.5$. It is additive increase and multiplicative decrease (AIMD) algorithm [1]. This congestion control algorithm works well in low-RTT network. But with the development of long distance network, if RTT is high, the utilize of the whole network is quite low. Another important requirement is RTT fairness. It means, in a certain bandwidth network environment, different connections with different RTT should share approximately the same bandwidth utilization ratio, but TCP is not a good solution. Nowadays, people are already trying to find a solution to improve TCP.

3 Datagram Congestion Control Protocol (DCCP)

Many other transport protocols appear in order to fit into different usages and varies of network physical conditions, especially for the high bandwidth delay product network. In the following sections I list and analyze some of the most well-known protocols so as to do the comparison in the discussion part.

3.1 Motivation

Fast-growing Internet applications including streaming media, telephony and interactive games need new requirements of network protocols. Most of them prefer timeliness to reliability. One special requirement of those applications is that they are extremely sensitive to delay and quality fluctuation. On the other hand, losing a certain number of packages would not affect they quality of service. This special characteristic of real-time application decides that TCP is not suitable for them, because TCP rather focuses on ensuring data transmission. In this case, retransmission of data packets is not need, and so does the order of packets' arrival. Most of these applications currently use UDP. Through the analysis of UDP traffic, UDP's lack of explicit connection setup and teardown presents unpleasant difficulties to network address translators and firewalls. Furthermore, because of UDP's lacking of congestion control, competing traffic problem would be caused. In this circumstance, Datagram Congestion Control Protocol appears.

3.2 Overview

DCCP is a unicast, connection-oriented transport protocol with bidirectional data flow. It provides built-in congestion control, including Explicit Congestion Notification (ECN) support [12]. DCCP offers a choice of modular congestion control mechanisms among a set of standardized algorithms for real-time applications. For the moment, two mechanisms are currently specified, TCP-like and TCP-Friendly Rate Control (TFRC) congestion control. These algorithms aim different applications. For instance, on-line games which want to make quick use of any available bandwidth might use TCP-Like; while streaming media applications trade off this responsiveness for a steadier, less bursty rate might use TFRC.

Different from UDP, DCCP connections start and end with three-way handshakes, and the 16-byte generic header which

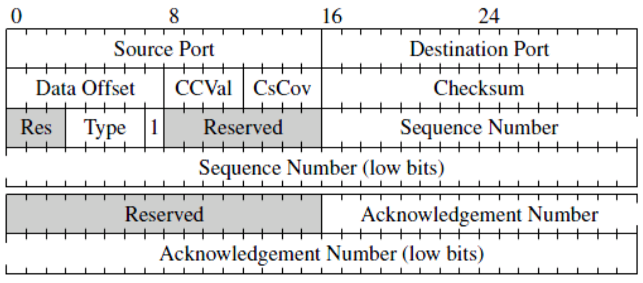

Figure 1: DCCP Header

datagrams begin with is shown in Figure 1. The Type field gives the type of packet. Even the acknowledgement number is optional, potentially reducing header overhead for unidirectional flows of data. Normally sequence and acknowledgement numbers are 48 bits long, but end points can set it to 24 bits while in the session negotiation period.

3.3 congestion Control algorithm

For congestion control algorithm, DCCP gives the application some choices of congestion control mechanisms. The choice is made via Congestion Control IDs (CCIDs). A connection's CCIDs can be negotiated when establishing the connection.

DCCP's CCID2 provides a TCP-like congestion control mechanism. Its congestion control algorithms are quite similar with TCP: a congestion window $cwnd$, a slow-start threshold, and an estimate of the number of data packets outstanding [15]. To reduce Ack load, it is set to at least two for a congestion window of four or more packets. However, to ensure that feedback is sufficiently timely, it is capped at $cwnd/2$, rounded up. Within these constraints, the sender changes Ack Ratio as follows. Let R equals the current Ack Ratio.

(1) For each congestion window of data where at least one of the corresponding Acks was lost or marked, R is doubled;

(2) For each $cwnd/(R^2 - R)$ consecutive congestion windows of data whose Acks were not lost or marked, R is decreased by 1.

The second formula is used to increase the number of Acks per congestion window, namely $cwnd/R$, by one for every congestion-free window that passes. However, since R is an integer, we instead find a k so that, after k congestion-free windows, $cwnd/R + k = cwnd/(R - 1)$.

TFRC congestion control in DCCP's CCID3 uses a different approach. Instead of a congestion window, a TFRC sender uses a sending rate. The receiver sends feedback to the sender roughly once per round trip time (RTT) reporting the loss event rate. The sender uses this loss event rate to determine its sending rate; if no feedback is received for several round-trip times, the sender halves its rate.

Regulate Sending rate is set by a Markov Model. The model is described as:

X is the transmit rate in bytes/second s is the packet size in bytes p is the loss event rate T_0 is the TCP retransmission time in seconds.

$$X(p) = \frac{s}{RTT\sqrt{2bp/3} + T_0(3\sqrt{3bp/8})p(1+32p^2)}$$

This is reasonably straightforward, and does not require reliable delivery of feedback packets, as long as the sender trusts the receiver's reports of the loss event rate. However, a mere loss event rate is ripe for abuse by misbehaving receivers.

3.4 Advantages

DCCP is a nice solution for real-time application. It avoids Internet congestion caused by package loss like UDP usually do. With the unreliable feature of UDP, it also wastes the traffic of networks. DCCP is a low-expense, unreliable congestion control protocol. In all the control packets, DCCP can send data messages simultaneously. The header of DCCP is changeable, the most common one only use 12 bytes. DCCP have two kinds of congestion control mechanism, and may adding new ones now. DCCP have 9 kinds of packets, more than TCP and UDP. This increases its flexibility and expand ability. Such as the DCP-Mov packets helps it to be adapted to mobile devices [4].

3.5 Limitations

Until now, DCCP also has some problems. DCCP should support both IPv4 and IPv6 at the same time. Whether DCCP is secure enough is still under consideration. Furthermore, applications generally do not want to implement TCP-friendly congestion control themselves. This is not only because congestion control can constrain performance, but also because properly implementing congestion control is very hard, as the long history of buggy TCP implementations makes clear [4]. Applications might be willing to subject themselves to congestion control, not least for the good of the network, as long as it was easy to use and met their needs. A modular congestion control framework would also make it easier to develop new applications, and to deploy congestion control advances across many applications at once.

4 Stream Control Transmission Protocol (SCTP)

4.1 Motivation

Telephony, video conferences and many other telecommunication applications appears in the modern Internet. It is necessary to transfer signalling messages over it. But TCP and UDP is not a good solution for those telecommunication network applications. SCTP is a general purpose unicast transport protocol for IP network data communications, which has been recently standardized by the IETF [16]. It was initially introduced as a means to transport telephony signaling messages in commercial systems, but has since evolved for more general use to satisfy the needs of applications that require a message-oriented protocol with all the necessary TCP-like mechanisms. Traditional TCP protocol has the problems of Head Of Line blocking, bad real time support, vulnerable to Denial of Service attack and so on [14]. SCTP is a better solution under this circumstance.

4.2 Overview

SCTP provides sequencing, flow control, reliability and full-duplex data transfer like TCP. However, it also enhances a set of capabilities not in TCP that make applications less susceptible to loss. Like UDP, SCTP is message-oriented and supports the framing of application data. Meanwhile like TCP, SCTP is session-oriented and communicates by establishing an association between two endpoints. Different with TCP, in SCTP, it is possible to have multiple logical streams within an association where each is an independent stream of messages and delivered in-order. Some of the important feature of SCTP is: Multi-homing, Multi-streaming, Initiation protection, Message framing, Configurable unordered delivery and Graceful shutdown. Normally, the upper layer user of SCTP would be Switched Circuit Network (SCN) signalling adaptable module, and the lower layer would be IP network.

4.3 Major Algorithm

Comparing with traditional transport protocol, the most important feature of SCTP is multi-homing and multi-streaming.

One of the most important change of SCTP is its support of multi-homed nodes, which means a server can be reached by several diffrent IP addresses. If packages from one node to another travels on physically different paths, and also different destination IP address are used, the connection becomes tolerant against physical network failures and other problems of that kind. As shown in figure 2, server has two IP addresses which are available for both ethernet and wireless network connection. Client can connect to server by multi-homing, therefore if one connection break, client can still maintain data exchanging with server by another connection.

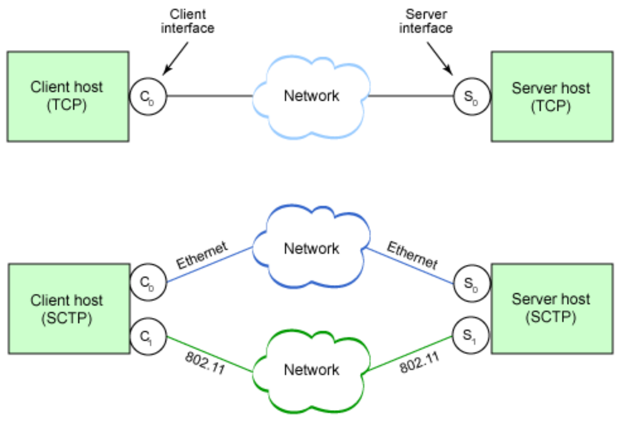

Figure 2: Multi-Homing Mechanism

Another important feature of SCTP is, it supports multiple streams within an association. In SCTP, each stream represents a sequence of messages within a single association, and they use their own seq number just like different TCP sessions. Both stream identifiers and sequence numbers are included in the data package [11]. This means that there would be no unnecessary head-of-line blocking between independent streams of messages in case of loss in one stream. All the streams within an association are independent but related to the association as shown in figure 3.

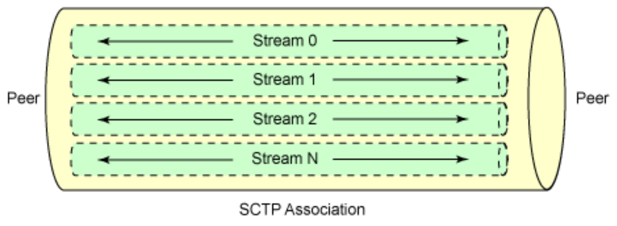

Figure 3: Multi-Streaming Mechanism

The congestion control mechanism of SCTP is quite similar with TCP. But for the congestion control function, STCP uses a more aggressive method.

$$Ack : cwnd = cwnd + a$$

$$Loss : cwnd = cwnd - b \times cwnd$$

Among them, a and b are set as 0.01 and 0.125. In this case, every time congestion window decrease, only a certain number of RTT is needed for recovering the original size of the congestion window, which is not relevant with the size of window now. This means, SCTP uses multiplicative increase and multiplicative decrease algorithm for congestion control. As TCP uses AIMD algorithm, this will cause problems with RTT fairness and TCP friendly. SCTP's congestion window grows much faster than TCP's, and occupies almost all the bandwidth.

4.4 Advantages

Though SCTP has TCP-like congestion and flow control mechanisms targeted for bulk data transfer, we argue that SCTP's feature-set makes it a better web transport than TCP. Performance-wise, SCTP's multistreaming avoids TCP's HOL blocking problem when transferring independent web objects, and facilitates aggregate congestion control and loss recovery [11]. Functionality-wise, SCTP's multihoming provides fault-tolerance and scope for load balancing, and a built-in cookie mechanism in SCTP's association establishment phase provides protection against SYN attacks.

4.5 Limitations

The major limitation of SCTP might be RTT fairness and TCP friendly. As SCTP use a quite aggressive congestion control function, it may occupy most of the bandwidth while working with other TCP connection. Furthermore, the aggressive algorithm would surely cause high package loss and lead to the multiplicative decreasing of TCP connections.

Some other limitations of SCTP are that SCTP is quite complex and need extra-supports, such as it needs 4 way handshake, and also multi-homing server.

5 CUBIC Transmission Control Protocol (CUBIC TCP)

5.1 Motivation

The low utilization problem of TCP in fast long-distance networks is well documented by S. Floyd in High Speed TCP for Large Congestion Windows [6]. This problem came from the slow increase of congestion window following a congestion event in a network with a large bandwidth delay product (BDP). Many experience indicates that this problem is frequently observed especially under a network path with over 100ms round-trip times (RTTs). This problem is equally applicable to all Reno style TCP standards and their variants which use the same linear increase function for window growth. CUBIC is designed to solve this problem. It offers a new congestion control function which is not relied on RTTs.

5.2 Overview

CUBIC is a high speed variant of standard TCP. To solve the problem of low utilization which TCP has, it uses a cubic function instead of a linear window increase for congestion control mechanism in order to improve scalability and stability under fast and long distance networks. The main feature of CUBIC is that its window growth function is defined in real-time so that its growth will be independent of RTT. It enhances the fairness properties of BIC while retaining its scalability and stability. In CUBIC, we try to find the balance between the congestion window size before windows reduction and after windows reduction. Despite this, although the real-time increase of the window enormously enhances the TCP friendliness of the protocol, in short RTT network, CUBIC's window growth is slower than TCP. So in order to keep the growth rate the same as TCP, CUBIC uses a new TCP mode to help change this situation. In CUBIC's TCP mode, it uses the same congestion control mechanism as TCP while the RTT is short.

5.3 Congestion Control Algorithm

Cubic's congestion function is:

$$W_{cubic} = C(t-K)^3 + W_{max}$$

where C is a scaling factor, t is the elapsed time from the last window reduction, W_{max} is the window size just before the last window reduction, and $K = \sqrt[3]{W_{max}\beta/C}$, where β is a constant multiplication decrease factor applied for window reduction at the time of loss event (i.e., the window reduces to βW_{max} at the time of the last reduction). Figure 4 shows the growth function of CUBIC.

The cubic function ensures the intra-protocol fairness among the competing flows of the same protocol [13]. Suppose that two flows are competing on the same end-to-end physical link, the two flows converge to a fair share since they drop by the same multiplicative factor β, so a flow with larger W_{max} will reduce more, and the growth function allows the flow with larger W_{max} will increase more slowly, since K is larger as W_{max} is larger. Thus, the two flows eventually converge to the same window size. This function also offers a good RTT fairness property because the window growth rate is dominated by t, the elapsed time. Since any competing flows with different RTT will have the same t after a synchronized packet loss, CUBIC also ensures linear RTT fairness while TCP offer square RTT fairness in terms of throughput ratio.

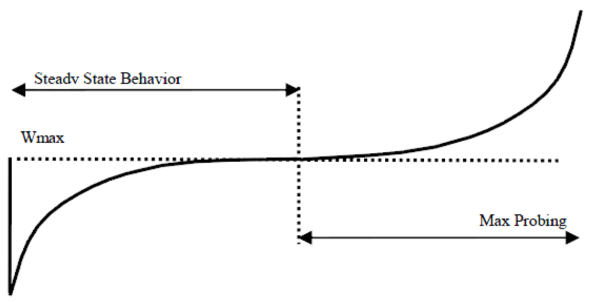

Figure 4: The Window Growth Function of CUBIC [9]

5.4 Advantages

CUBIC TCP ia a nice solution for BDP network. With the development of Internet, the route trip times usually become very high (around 100 to 200ms). In this case, standard TCP has a low utilize radio. Nowadays many High Speed TCP variants came out and tried to fix this problem. But the difficulties are not only raising the efficiency, but also maintain the RTT fairness and TCP friendly. CUBIC works much more better than those previous high speed TCP variants. And now, CUBIC TCP is implemented and used by default in Linux kernels 2.6.19 and above.

5.5 Limitations

Although CUBIC TCP seems a nice solution for high speed transport layer, there are some limitations of it. CUBIC TCP suffers from slow convergence-yields poor network responsiveness, prolonged unfairness between flows, increases unfairness between long and short lived flows. CUBIC TCP is a good solution for standard Internet environment, but it is not aimed for special usage, such as satellite network or mobility usage.

6 Compound Transmission Control Protocol (CTCP)

6.1 Motivation

As we discussed before, the protocol design requirements for high speed TCP variants are mainly two things: Efficiency and TCP fairness. Efficiency means to effectively utilize high-speed link even with large delay. TCP fairness means that the protocol should be able to be progressively deployed. It is easy to meet efficiency requirement, but it is difficult to be both efficient and TCP fairness. One core idea of delay based congestion control is that the increase of RTT is considered as early congestion, and the sending rate is reduced to avoid self-induced buffer overflow. So in CTCP, we try to find a synergy of both delay-based and loss-based approach.

6.2 Overview

CTCP is a synergy of both delay-based approach and loss-based approach [17]. It has two components for congestion control.First is the delay-based component like the algorithm in FAST TCP. We notice that delay-based approaches already have this nice property of adjusting its aggressiveness based on the link utilization [18]. Another component is loss-based component like what BIC TCP and SCTP do. For a high-speed and long delay network, it will take standard TCP an unreasonably long time to recover the sending rate after a single loss event. And using MIMD congestion control windows can effectively help fixing the problem. So in Compound TCP, we aim at combining this two features together. CTCP keeps the same Slow-Start behavior of regular TCP at the start-up of a new connection.

6.3 Congestion Control Algorithm

In CTCP, there are two window state variables: $cwnd$ and $dwnd$. $cwnd$ represents the congest window while $dwnd$ is the delay window. Specifically, the TCP sending window is now calculated as follows:

$$win = min(cwnd + dwnd, awnd),$$

where $awnd$ is the advertised window from the receiver.

For the $cwnd$, it updates as standard TCP,

$$Ack : cwnd = cwnd + 1/win$$

$$Loss : cwnd = cwnd/2$$

For $dwnd$ component calculation, we set:

$$Expected = win/baseRTT$$

$$Actual = win/RTT$$

$$Diff = (Expected - Actual) \cdot baseRTT$$

And the control functions is:

$$dwnd(t+1) = \begin{cases} dwnd(t) + (\alpha \cdot win(t)^k - 1)^+, & \text{(1a)} \\ (dwnd(t) - \zeta \cdot diff)^+, & \text{(1b)} \\ (win(t) \cdot (1-\beta) - cwnd/2)^+, & \text{(1c)} \end{cases}$$

$$ifdiff < \gamma,$$

$$ifdiff \geqslant \gamma,$$

$$iflossisdetected.$$

where $(.)^+$ is defined as $max(.,0)$.

If a retransmission timeout occurs, $dwnd$ should be reset to zero and the delay-based component is disabled. Also, following the common practice of high-speed protocols, CTCP also reverts to standard TCP behavior when the window is small. Delay-based component only kicks in when win is larger than some threshold W_{low}.

6.4 Advantages

With this combination delay-based component and loss-based component, Compound TCP can satisfy all three requirements pretty well. First, CTCP can efficiently use the network resource and achieve high link utilization. In theory, CTCP can be very fast to obtain free network bandwidth, by adopting a rapid increase rule in the delay-based component. Secondly, CTCP has similar or even improved RTT fairness compared to regular TCP. This is due to the delay-based component employed in the CTCP congestion control algorithm. Thirdly, CTCP has good TCP-fairness. By employing the delay-based component, CTCP can gracefully reduce the sending rate when the link is fully utilized.

7 Summary of Transport Protocols

This section aims at comparing the performance of the protocols above in detail in different aspects, and show the result in a table. A good protocol should work well in real network environment, and keep fairness while being efficient.

Network environment is almost the most important influential factor [8]. Even a little bit change may cause totally different performance for those transport layer protocols. I conclude those factors as: bandwidths (from 1Mb/s to 1Gb/s), RTT (from 16ms to 200 ms), range of queue size (from 2% to 100% BDP), and range of background traffic levels (number of sessions, distribution of connection sizes and network topology). Background traffic is quite important while simulating these protocols as you can see from the experimental results.

For the performances analysis of those protocol, I mostly focus on the following feature: RTT Fairness, TCP Fairness, Utilization Ratio, Packet Loss Rate, Smoothness of throughput and Convergence Time. Namely, the most important feature are:

[Efficiency] It must improve the throughput of the connection to efficiently use the high-speed network link, including Utilization Ratio and Smoothness of throughput.

[RTT fairness] It must also have good intra-protocol fairness, especially when the competing flows have different RTTs.

[TCP fairness] It must not reduce the performance of other regular TCP flows competing on the same path. This means that the high-speed protocols should only make better use of free available bandwidth, but not steal bandwidth from other flows.

Thanks for the experimental work done by [15] [8], I could give out a basic comparison among those transport protocols. All the works blow are based on these experimental results.

For the convergence time of protocols, only high speed TCP variants are compared. We observed that STCP significantly improved its convergence time with background traf-

Transport Protocol	Protocol Type	Congestion Control Algorithm	Congestion Basement	RTT Fairness
TCP	Standard TCP	$Ack : cwnd = cwnd + \frac{a}{cwnd}$	Loss Based	Quite Bad, only keeps fairness when RTT is short
UDP	Standard UDP	No Congestion Control	No Congestion Control	Do not have Ack, so independent of RTT
DCCP	UDP Variant	$cwnd/R + k = cwnd/(R-1)$	Loss Based	Same as TCP, depends heavily on RTT length
SCTP	High Speed TCP Variant	$Ack : cwnd = cwnd + a$	Loss Based	Becomes better when RTT increase
CUBIC UDP	High Speed TCP Variant	$W_{cubic} = C(t-K)^3 + W_{max}$	Loss Based	Quite good, designed for solving this problem
Compound TCP	High Speed TCP Variant	$win = min(cwnd + dwnd, awnd),$	Loss and Delay Based	Quite good, even excellent

Table 1: Comparison of Transport Protocols: Part 1

Transport Protocol	TCP Fairness	Utilization Ratio	Packet Loss Rate	Smoothness of throughput	Convergence Time
TCP	Medium	Bad, caused by RTT fairness problem, the congestion window cannot reach to maximum size	Good due to the low utilization ratio, normally depend on the condition of link	Good	Medium
UDP	Quite bad, since UDP does not have congestion control, it won't care about TCP fairness	Good.	Bad, but it is not concerned for UDP	Not tested here	Not tested here
DCCP	Quite bad, DCCP use quite aggressive congestion control.	Good, similar as UDP	Bad, but packet loss is acceptable for DCCP	Not tested here	Not tested here
SCTP	Very bad, even more aggressive congestion control than DCCP	Good, but at the price of low TCP fairness and complex multi-streaming technology	Bad, quite high loss rate	Bad	Bad, but better with background traffic
CUBIC UDP	Good, use TCP mode and cubic function to solve the problem	Good	Good performance	Good	Good
Compound TCP	Good,Using loss-based mechanism to ensure it	Good	Good	Good, much better than SCTP	Good

Table 2: Comparison of Transport Protocols: Part 2

fic. When RTT is short, the convergence time of STCP seem to be much higher than others.

For RTT fairness, it is interesting to note that all protocols obtained a better RTT fairness index with background traffic as RTT increased, but Compound TCP and CUBIC TCP have a better RTT fairness. They dramatically changed the bad situation of TCP.

For TCP fairness, CUBIC TCP and Compound TCP have better performance. Under low RTT, STCP did not show much fairness to TCP. As the delay increases, all the high speed protocols show less fairness to TCP (especially so for STCP). If the background data stream is DCCP, it appears similar situation.

For utilization Ratio, when the propagation delays are small, all protocols are able to obtain high link utilization. When round-trip times increases, link utilization for SCTP and DCCP degrades.

For packet loss rate, We observed that among all the high TCP variants, STCP resulted in the highest packet loss rates. This result indicates that it is more aggressive than other protocols in achieving a high transmission rate. This also explains why STCP obtained the lowest indexes for TCP friendliness among all protocols.

For smooth of throughput, we use coefficient of variation (CoV) of throughput to measure the smoothness and sometimes stability of a protocol. We observe that Compound TCP achieves small CoV values in short RTT. We note that the CoV values for STCP are among the highest, regardless whether background traffic is used or not.

Table 1 and 2 shows the comparison of these 6 transport protocols in different aspects.

8 Conclusion

In this paper, I did a survey on the most popular Transport protocols nowadays. TCP, DCCP, SCTP, CUBIC TCP and CTCP are listed and analyzed. Especially, I focussed on the congestion avoidance control algorithm of them. During the analysis, I find the most important feature of TCP nowadays is to balance efficiency with fairness. In different network conditions, fairness, convergence time, packet loss rates, link utilization, RTT fairness, TCP friendliness, and stability of throughput are considered to evaluate these protocols. Congestion control algorithms might directly decide the performance of these protocols. DCCP is based on UDP and added congestion control for real time application. SCTP use multi-homing and multi-streaming to increase the utilization ratio. CUBIC use New TCP mode to maintain fairness and MIMD algorithm to improve efficiency. CTCP combines loss-based and delay-based windows control. I do not declare any winner in our evaluation but simply show contrasting difference between them. And my analysis is just used as a reference and of course needs experimental proof.

References

[1] M. Allman, V. Paxson, and W. Stevens. Tcp congestion control, rfc2581.

[2] G. Anastasi, E. Ancillotti, M. Conti, and A. Passarella. Experimental analysis of a transport protocol for ad hoc networks (tpa). In *PE-WASUN '06: Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, pages 9–16, New York, NY, USA, 2006. ACM.

[3] P. G. Bridges, G. T. Wong, M. Hiltunen, R. D. Schlichting, and M. J. Barrick. A configurable and extensible transport protocol. *IEEE/ACM Trans. Netw.*, 15(6):1254–1265, 2007.

[4] L. M. de Sales, H. O. Almeida, and A. Perkusich. On the performance of tcp, udp and dccp over 802.11 g networks. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 2074–2078, New York, NY, USA, 2008. ACM.

[5] H. Elaarag. Improving tcp performance over mobile networks. *ACM Comput. Surv.*, 34(3):357–374, 2002.

[6] S. Floyd. High speed tcp for large congestion windows, rfc3649. 2003.

[7] Y. Gu, X. Hong, and R. L. Grossman. Experiences in design and implementation of a high performance transport protocol. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 22, Washington, DC, USA, 2004. IEEE Computer Society.

[8] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu. A step toward realistic performance evaluation of high-speed tcp variants. In *Fourth International Workshop on Protocols for Fast Long-Distance Networks*, 2006.

[9] S. Ha, I. Rhee, and L. Xu. Cubic: a new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, 2008.

[10] S. Iren, P. D. Amer, and P. T. Conrad. The transport layer: tutorial and survey. *ACM Comput. Surv.*, 31(4):360–404, 1999.

[11] H. Kamal, B. Penoff, and A. Wagner. Sctp versus tcp for mpi. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 30, Washington, DC, USA, 2005. IEEE Computer Society.

[12] E. Kohler, M. Handley, and S. Floyd. Designing dccp: congestion control without reliability. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–38, New York, NY, USA, 2006. ACM.

[13] D. Leith, R. Shorten, and G. McCullagh. Experimental evaluation of cubic tcp. In *In Proc. Workshop on Protocols for Fast Long Distance Networks*, 2007.

[14] P. Natarajan, J. R. Iyengar, P. D. Amer, and R. Stewart. Sctp: an innovative transport layer protocol for the web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 615–624, New York, NY, USA, 2006. ACM.

[15] F. Nivor. Experimental study of dccp for multimedia applications. In *CoNEXT '05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 272–273, New York, NY, USA, 2005. ACM.

[16] L. Ong and J. Yoakum. An introduction to the stream control transmission protocol. 2002.

[17] K. Tan, J. Song, Q. Zhang, and M. Sridharan. Compound tcp: A scalable and tcp-friendly congestion control for high-speed networks. In *In Proc. of PFLDnet*, 2006.

[18] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound tcp approach for high-speed and long distance networks. pages 1–12, April 2006.