Dual-Priority Congestion Control Mechanism for Video Services
Real Network Tests of CVIHIS

Juha Vihervaara, Pekka Loula and Teemu Alapaholuoma
Pori Campus, Tampere University of Technology, Pohjoisranta 11, 28100, Pori, Finland

Keywords: Congestion Control, Video Traffic, Priority.

Abstract: Information can be shared effectively by using of videos. Therefore, it is no wonder that videos form most of the Internet traffic. For the efficient operation of the Internet, it is essential that these videos are shared with a proper and effective way. In our previous paper, we have already presented a congestion control mechanism for the proper transmission of videos. This mechanism can offer two priority levels for video. There is a low priority service where the bandwidth is given away to other connections after the load level of a network exceeds a certain level. Instead, the other priority level, a real-time mode, always wants its fair share of the bandwidth. In this study, we have tested the operation of this mechanism by doing real network tests.

1 INTRODUCTION

Computer and communication technology offers many kinds of possibilities for information sharing. Some of these methods use video-based approaches for information sharing. These video-based approaches have many advantages and positive effects. For example, related to the education of technical skills, Donkor (2010) found that the users of video-based instructional materials demonstrated superior levels of craftsmanship compared to the users of print-based instructional materials. Concerning the health care field, Pandey (2010) presents that YouTube can be considered an important educative tool that can play a significant role in the event of global disease outbreaks.

So, it is no wonder that the transfer of video has increased dramatically on the Internet over the past decade. Cisco (2015) shows that video traffic also plays a big role in Internet traffic in the future. It predicts that consumer Internet video traffic will be 80 percent of all consumer Internet traffic in 2019. Every second, nearly a million minutes of video content will cross the network by 2019.

There are two different kinds of transfer modes needed among video services. The first one is a backward loading mode where delay and bandwidth demands are moderate. For example, the whole video can be loaded to the user device before it is watched. This mode is also useful when the information producer loads videos to proxy servers. This mode works like a low-priority service where the bandwidth is given away to other connections when the load level of the network is high enough. The second mode is a real-time mode where delay and bandwidth demands are important. For example, a live broadcast is watched. This mode always wants its fair share of the bandwidth. It is also possible that a case can be some kind of intermediate form. At first, the video can be transferred by using high speed. When there is enough data in the receive buffer, the transfer mode can be changed to the backward loading type.

We have developed an algorithm that supports both these video transfer modes. This algorithm is based on network congestion control. This congestion control mechanism was named Congestion control for VIdeo to Home Internet Service, or CVIHIS. The algorithm has been presented by the study Vihervaara and Loula (2015). In this our previous study, we analyzed the performance of CVIHIS by simulations. In this study, we will test the operation of the algorithm in real network environments.

It is important to test our mechanism in real network environments. Simulations are very useful due to their simplicity and flexibility. Unfortunately, simulations don’t always reflect totally real-world situations. For example, the paper Jansen and McGregor (2006) presents that simulation tools may

Vihervaara, J., Loula, P. and Alapaholuoma, T.
Dual-Priority Congestion Control Mechanism for Video Services - Real Network Tests of CVIHIS.
DOI: 10.5220/0006049800510059
In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016) - Volume 3: KMIS*, pages 51-59
ISBN: 978-989-758-203-5

use simplified models of TCP which don't always correspond to real word implementations in all situations. The paper Floyd and Kohler (2003) presents that scenarios used in simulations often include certain assumptions. For example, the flows of the congested link share a small range of round-trip times. All assumptions affect simulation results and evaluations. Some divergences from the reality are unimportant if they don't affect the validity of simulation results. However, we don't yet understand which aspects of models affect in a fundamental way to the system behaviour and which aspects can be ignored safely.

This study also includes extensive tests, in which the mechanism is tested against itself. In the simulation phase, the mechanism was tested against itself only in a preliminary way. Testing a mechanism against itself in a comprehensive way is important, since it is desirable that a control system like the Internet congestion control is not too heterogenous. It can be seen, also on the basis of the results of this study, that it can be used only a few congestion control mechanisms on the Internet, if we want to guarantee the desired operation of the system.

The paper is organized as follows: Section 2 gives the backgrounds for network congestion control; Section 3 describes our dual-mode congestion control algorithm; Section 4 shows the experimental results and Section 5 concludes the paper.

2 BACKGROUNDS

The principles of Internet congestion control are introduced in this part. Congestion control is an extensive research area and things only relevant for this study are presented here.

2.1 Congestion Control

The background of congestion control is in queuing theory (Lakshmi and Bindu, 2011). In a packet-switched network, packets move into queues and out of queues when these packets traverse through a network. Because of that, packet-switched networks are often said to be as the networks of queues. The goal of congestion control is to avoid a congestion situation in network elements. So, congestion control has to adapt the sending rates of senders to match to the available end-to-end network capacity.

There can be many reasons why networks become congested. Singh et al. (2008) lists such kind of reasons: limited memory space, limited channel bandwidth, limited capacity of the routers, load of the network, link failure, heterogeneous bandwidths. TCP/IP networks, such as the Internet, are especially open to congestions because of their basic connectionless nature on the network layer. IP routers don't support resource reservation in a normal situation. Instead, packets of variable sizes are sent by any host at any time. This makes it difficult to predict traffic patterns and resource needs. While connectionless networks have their advantages, robustness against congestion is not one of them.

The consequences of bad congestion control are described in Kurose and Ross (2012). One cost of a congested network is that large queuing delays are experienced. Queueing delays increase the response times of web services. For example, Fabian et al. (2010) presents that the fast download speed of a website increases visitor loyalty and user satisfaction. Therefore, if we think a network user's point of view, congestion is not a desired situation. Because of that, the network with user's friendly properties must implement some kind of congestion control.

2.2 Congestion Control Mechanisms for Video Services

The TCP protocol implements congestion control. Although some video services use TCP to implement their transport services in a manner that actually works, TCP is not an ideal protocol for the use of all video applications. Especially, real-time video applications are often loss-tolerant and they do not want to use retransmissions offered by TCP. In the case of TCP, the bytes after the missing ones can not be delivered to the application before the retransmissions of the missing ones. For these reasons, UDP is often more suitable for the use of real-time video applications. Unfortunately, UDP does not implement congestion control.

There are many congestion control algorithms that are suitable for the use of video services. LEDDBAT (Shalunov et al., 2012) provides low priority services by using one-way delay measurements to estimate the amount of queued data on the routing path. LEDBAT connections withdraw from using the bandwidth after the queuing delay exceeds the predefined target. The most known proposal for real-time streaming media is TCP Friendly Rate Control version of DCCP (Floyd et al., 2008). TFRC is designed for applications that need a smooth rate. Google Congestion Control for

Real-Time Communication on the World Wide Web (Lundin et al., 2012) is a relatively new proposal in this area.

2.3 TCP Friendliness

The real-time mode of CVIHIS aims to share the bandwidth of transmission links in an equitable manner. This equal allocation of bandwidth is called friendliness. We often talk about the TCP friendliness, because TCP was the first protocol which implemented congestion control on the Internet. So, there has often been an effort to make a new congestion control mechanism to work in a TCP friendly way.

Unfortunately, TCP fairness is a complicated thing. Even a TCP flow itself is not always friendly against another TCP flow. As just told in the previous subsection, there are several different versions for the TCP protocol. These all versions can not be totally identical with their behaviours. TCP's throughput also degrades through higher RTTs (Widmer et al., 2001). Because of that, TCP has a bias against high round-trip time connections, and even identical TCP implementations are not equal. In addition, there is not an exact definition for the concept of TCP friendliness. When a new mechanism is developed and compared against the TCP protocol, there is always some room for personal opinions.

3 DUAL-MODE CONGESTION CONTROL MECHANISM CVIHIS

In this section, the algorithm of CVIHIS is presented shortly. The study Vihervaara and Loula (2015) gives the more comprehensive and justified presentation. This section also briefly describes the implementation principles of CVIHIS for real network environment.

3.1 Algorithm of CVIHIS

CVIHIS is a rate-based congestion control approach. It uses the Exponentially Weighted Moving Average (EWMA) filter to smooth sending rates because the sending rates of video applications should usually vary in a smooth way. When congestion control is carried out, sending rates can be adjusted by observing packet losses or delays. Both indicators are utilized by both modes of CVIHIS. The backward loading mode is a more delay-based than loss-based approach. Instead, the real-time mode has been shifted more loss-based direction. CVIHIS is an end-to-end mechanism so that routers can continue to work in a normal way without complex issues.

3.1.1 Backward Loading Mode

The rate adaptation schema of CVIHIS is presented in Figure 1. Two delay values are picked on the basis of round-trip time measurements. The minDelay value is the shortest delay value experienced during the lifetime of the connection. The minDelay value presents a situation in which the queues of a routing path are empty. The maxDelay point is updated every time when a packet drop occurs using the delay value of the last received packet before the dropped one. The maxDelay value includes queening delay components. It corresponds to a situation in which the buffer of a router overflows.

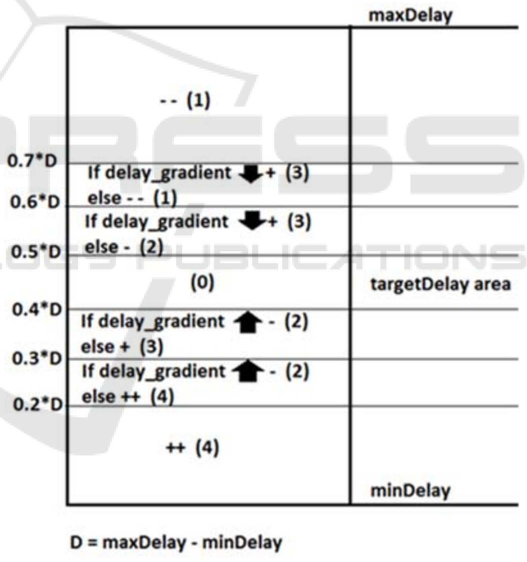


Figure 1: Rate adaptation schema of CVIHIS.

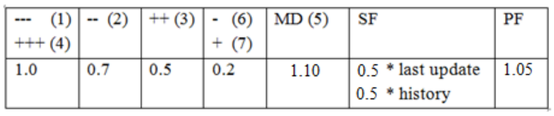
With the help of these two delay values, the delay space is divided into seven rate adaptation areas. We could also say that the queue of a router is divided into parts. The idea is that CVIHIS tries to keep the queue level at the level of the targetDelay area. If the queue level is over the target, sending rates are decreased. If the queue level is below the target, sending rates are increased. The positioning factors for each delay area are presented on the left side of Figure 1. The target delay area is not put in the middle of the delay space so that queuing delays can be kept short.

There is a black arrow inside some delay areas. This arrow represents the value of delay gradient. If the arrow indicates upwards, based on the delay measurements of two consecutive packets, delays are increasing and the queue is filling up. So, the sending rates are over the capacity of the outgoing link and, therefore, the rates must be decreased. If the arrow is downwards, the queue is empting and the sending rates can be increase. If there is a conflict of interests between the delay area and delay gradient adaptation, the gradient adaptation is chosen.

So, CVIHIS adjusts its sending rate through seven adjustment steps. Six of these steps are presented in Figure 1 by +++, ++, +, -, --, ---. There are three different level steps for both increasing and decreasing the sending rate. Bigger steps are used when the queue level is far away from the target. When the rate adaptation is based on delay gradients, the steps are shortest. The seventh adjustment step is a multiplicative decrease step which is used after a packet drop. In its additive increase phase, the TCP protocol increases its sending rate by one segment for each round-trip time interval. In its basic form, CVIHIS increases or decreases its sending rate by one packet for each square root of round-trip time interval. By using square root, CVIHIS alleviates the favoring behavior of short distance connections.

Table 1 presents the factors for the rate adjustment. Some of these factors differ somewhat from the study Vihervaara and Loula (2015), because the used TCP version is now NewReno. NewReno is somewhat more aggressive than Reno, which was the used TCP version in the simulation test. The four left most columns present the cases where the rate adjustment is based on the square-root of round-trip time. The factor indicates how many more or less packets will be send during the next square-root of round-trip time than just before. These rate adjustment steps presented in Figure 1 are presented in the first row of this table. MD is a multiplicative decrease factor which is used after packet drops to increase the sending gap of packets. SF is a smoothing factor used for the EWMA filter. PF is a pushing factor used only by the real-time mode.

Table 1: Adjustment parameters of CVIHIS.



--- (1) +++ (4)	-- (2)	++ (3)	- (6) + (7)	MD (5)	SF	PF
1.0	0.7	0.5	0.2	1.10	0.5 * last update 0.5 * history	1.05

All conclusion procedures presented in Figure 1 are implemented on the receiver side. Only the rate adaptation commands are transmitted to the sender. The values of the integer-based rate adaptation commands are presented inside parentheses in Figure 1 and Table 1.

3.1.2 Real-time Mode

For the real-time mode, we must modify the algorithm so that it can behave in a more aggressive way so that there is not anymore back off behaviour. We have taken an approach in which the delay areas are pushed upwards. The minimum delay value is pushed upwards by the pushing factor so that the current value is multiplied by this factor. It was found that the pushing factor of 1.05 is the suitable value.

The target delay may be over the delay demand of an application. In such a case, this delay demand has to be satisfied by Quality-of-Service mechanisms (Meddeb, 2010), because CVIHIS is a pure congestion control mechanism.

3.2 Software Implementation for Real Network Tests

For testing CVIHIS in real network environment, the CVIHIS code implementation must be placed somewhere to the protocol stack. There are some possibilities for this placing. For example, an open source operating system could be utilized so that its kernel implementation could be modified. Through this modification, the UDP implementation of this operating system could be changed to correspond to the algorithm of CVIHIS. Instead of using this elaborate and efficient solution, we chose the easiest implementation option. CVIHIS was implemented as a normal socket program on the top of the UDP protocol. This solution is possible because UDP protocol doesn’t offer any special transport services which could disturb the operations of CVIHIS. This option offers some advantages. UDP port numbers can be used in such a way that the same computer can run a few traffic sources at the same time. This reduces the number of terminals needed for the test network. In addition, if we do large scale real network testing in the future, transferring CVIHIS communication inside the UDP protocol will offer resistance against firewall blocking.

The natural choice was to use the programming language C for this implementation, because in that way we could take advantage of the CVIHIS implementation of the NS-2 simulation program

(NS-2, 2011). If we compare this real implementation to the simulation one, it can be said that the receiving-end solutions are both pretty similar. Instead, the transmitting-end solutions deviate more from each other because the real network version can not utilize easy to use timers and even handlers offered by the simulation environment. In addition, the real network version should take into account that, unlike in simulation environment, the clocks of transmitting and receiving ends have not been synchronized with each other.

4 TEST RESULTS

This section describes the structure of the test network. It also presents the most relevant results of the measurements. It is worth up to bring out one thing. The test results of CVIHIS show that there is room for improvement in some cases. In these cases, there can be present connections with very different round-trip times. As it is known, the TCP protocol favours short round-trip time connections. Therefore, it is also necessary to make the algorithm of CVIHIS to favour short round-trip times. However, CVIHIS does this favouring in a more moderate way than TCP does. In fact, favouring the connections of shorter round-trip times is not only a bad thing. Shorter connections consume less network resources. By favouring connections of shorter round-trip times, network operators can maximize the total traffic volume in their networks. The question is that how strong this favouring can be so that the objective of fairness does not suffer too much.

4.1 Test Network

The structure of the test network is presented in Figure 2. There are four end nodes and two routers. The link between the routers is the bottleneck link. With this simple test network structure, and with the help of tc-program, we can easily emulate different types of network structures. Tc (traffic control)(tc, 2016) is the user-space utility program used to configure the Linux kernel packet scheduler.

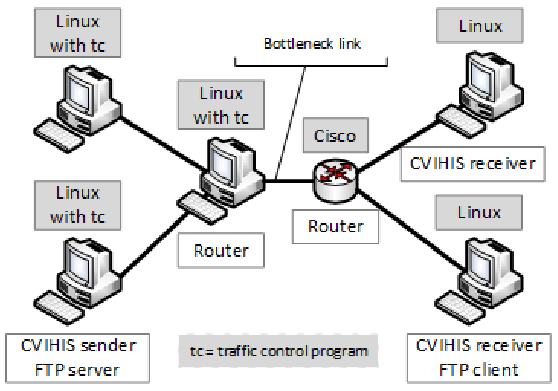


Figure 2: Structure of the test network.

In this study, tc is used in two ways. Tc is used for traffic shaping in the Linux router. In this way, we can vary the capacity of the bottleneck link and the transmission queue size of this link. When traffic is shaped, the transmission rate of that link is under control. Typically this means that we are lowering the available bandwidth. Traffic shaping can also be used to smooth the burstsness of incoming links by defining the queue size of the link. If this queue size is exceeded, the incoming packets are dropped. In the traffic source nodes, we use tc to define the delay characteristics the outgoing links. With this way, it is possible to emulate different round-trip times of connection paths.

4.2 Backward Loading Mode

The backward loading mode has to reach the constant sending rate if there is no need for back off operation. This means that the sending rate does not oscillate. This was ensured by doing twenty simulations. In these simulation cases, the queue size of the bottleneck link varied between 40-60 packets. The capacity of the bottleneck link varied between 2-4.5 Mbps and the round-trip time varied between 10-250 milliseconds. The sending rate stabilized in all cases.

The second objective for this mode is the proper back off behaviour. The back off behaviour was tested against one TCP NewReno connection by using the same kinds of test setups as in the case of the stability check. The proper back off behaviour realized in all cases. This was not a surprise because the used NewReno version is more aggressive TCP version than the Reno version used in the simulation phase.

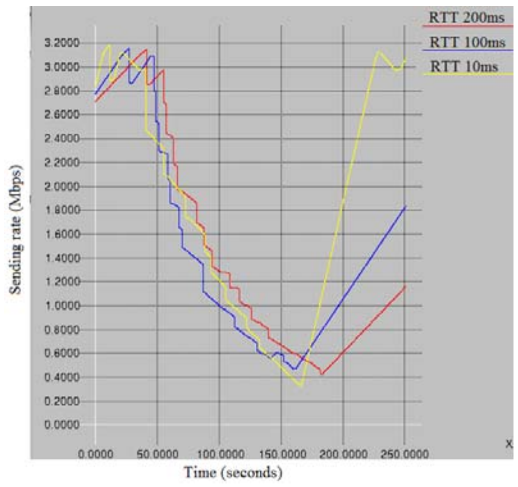


Figure 3: Simulation results of the backward loading.

Figure 3 presents the sending rates of the CVIHIS connections in the tests, in which the back off behaviour was tested by using different round-trip times (10, 100, 200ms) for CVIHIS, when TCP used the round-trip time of 200 milliseconds. The capacity of the bottleneck link was 3 Mbps. The TCP connections were active between the test time of about 50-170 seconds. Of course, based on the algorithm, CVIHIS can especially increase its sending rate faster in the cases of short round-trip times.

4.3 Real-time Mode

The TCP-friendliness of CVIHIS was tested so that the tests were made against the TCP NewReno version. CVIHIS was tested against one TCP connection by doing twenty six simulations. The queue size of the bottleneck link was 50 packets. The capacity of the bottleneck link varied between 2-4.5 Mbps. Four different round-trip times (20, 80, 140, 200 milliseconds) were used. There were also differences between the starting rates of the connections.

These test results are presented in Figure 4. The actual measurements are presented by using the points. The lines between the points only visualize the trend of the sending rates. As it can be seen, the phase effect affects so that individual measurements can depart from the trend. The figure presents how many percent CVIHIS gets from the capacity of the bottleneck link. This figure shows the acceptable level of averaged fairness. In the worst cases, the connection of higher bandwidth gets about 1.6 times as much bandwidth as the slower connection.

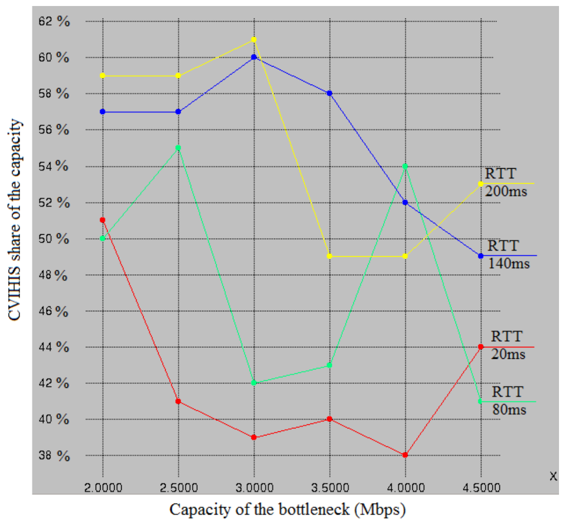


Figure 4: Real-time mode against one TCP connection.

As it is said, TCP favors the connections of short round-trip times and CVIHIS does this favoring in a more modest way. The results confirm this. Round-trip times affect less CVIHIS than TCP. CVIHIS manages somewhat modest way when round-trip times are short. When round-trip times are long, CVIHIS manages some better than TCP.

The used Linux version was also supported by another TCP's congestion control mechanism. This version was TCP CUBIC (Ha et al., 2008). Actually, CUBIC is the current default TCP algorithm in Linux. So, we made some tests where CVIHIS was tested against the CUBIC version. The preliminary results show that CUBIC behaves some more aggressively than NewReno. So, if it is wanted that CVIHIS manages in a friendly way with the CUBIC version, the rate adjustment parameters of CVIHIS have to be adjusted slightly so that CVIHIS would behave more aggressively.

4.4 CVIHIS against Itself

This section presents the results of tests in which CVIHIS was tested against itself. The results of the previous subsection and the study Vihervaara and Loula (2015) show that it is very difficult to achieve the acceptable level of fairness in heterogeneous network environments. So, the implementation of a workable solution for network congestion control might require that there are only a few different kinds of congestion control mechanisms on the Internet. Thus, it is important that CVIHIS behaves against itself as desired.

The real-time mode of CVIHS was tested against itself by doing 30 tests. In these cases, the queue size of the bottleneck link varied between 40-60 packets. The capacity of the bottleneck link varied between 2-6 Mbps. The round-trip times varied between 10-240 milliseconds. The tests were also made so that the connections used different round-trip times. There were also differences between the starting rates of the connections. Based on these test, the sending rates indicated the good level of fairness. In most cases, transmission rates differed less than 10 percent. Only when the round-trip times were substantially different, the rate differences were larger than this. If the worst cases are examined, the connection of higher bandwidth got about 1.7 times as much bandwidth as the slower connection. In this case, the faster connection had the round-trip time of 10 milliseconds and the slower connection 180 milliseconds.

One of these tests is presented in Figure 5. The capacity of the bottleneck link is 4 Mbps. The round-trip times of these CVIHIS connection are 60 (red) and 180 (blue) milliseconds. In this case, the average sending rates are 2085 Mbps and 1935 Mbps. The first 50 seconds are not used as part of these rate calculations.

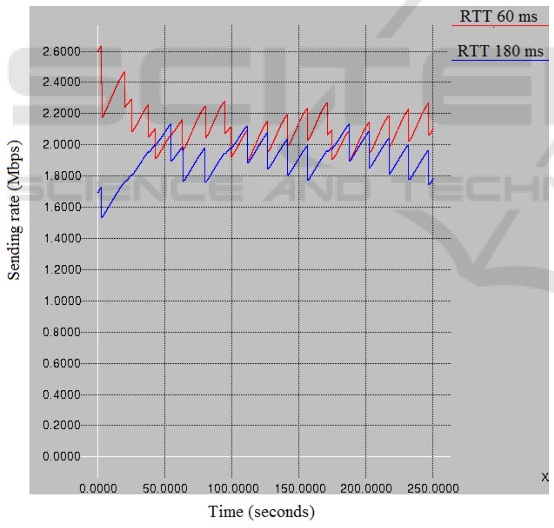


Figure 5: CVIHIS real-time mode against itself.

Some tests were also made so that there were four active CVIHIS connections in the network at the same time. CVIHIS managed in an acceptable manner in these tests although it took more time to balance the sending rates in the cases of long round-trip times. The result of one such test is presented in Figure 6. In this case, the capacity of the bottleneck link is 10 Mbps and the round-trip time is 30ms. The queue size of the bottleneck link is 60 packets. The number of active connections is presented at the bottom part of this figure.

It was also tested in what way the withdraw behavior of the backward loading mode realizes when there is a real-time connection on the connection path at the same time. We made 50 tests so that the capacity of the bottleneck link was 2 or 4 Mbps. The queue size of the bottleneck link was 60 packets. The round-trip times varied from 10 to 200 milliseconds.

When both modes used the same round-trip time, the back off behavior was proper. The tests were also made so that the modes used different round-trip times. In these cases, the back off actions happened slowly if the round-trip time of the real-time mode was much longer than the round-trip time of the backward loading mode. If the round-trip times differed considerably, for example, ten times, backward action did not take place at all. This behavior is presented in Figure 7. The figure shows the sending rates of the backward loading mode in three separate cases. In these cases, the round-trip times of the real-time mode are 10, 150, and 200 milliseconds. In all these cases, the round-trip time of the backward loading mode is 50 milliseconds. It is easy to moderate this phenomenon. The backward loading mode could use less aggressive rate adjustment parameters than the real-time mode. In this study, both modes used the same parameter set.

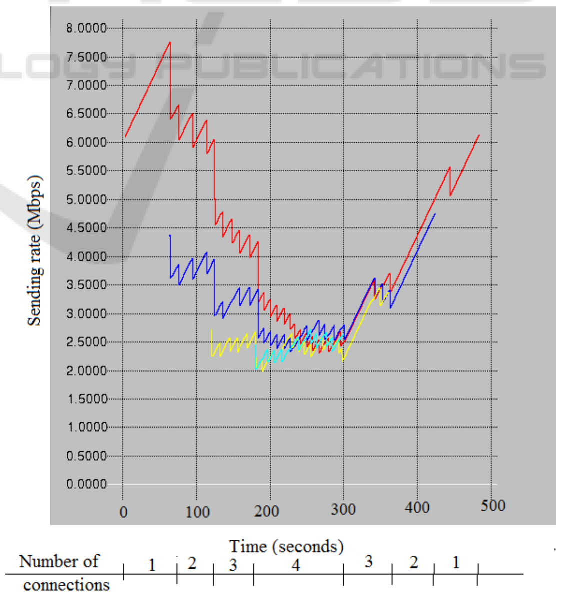


Figure 6: Four real-time mode connections.

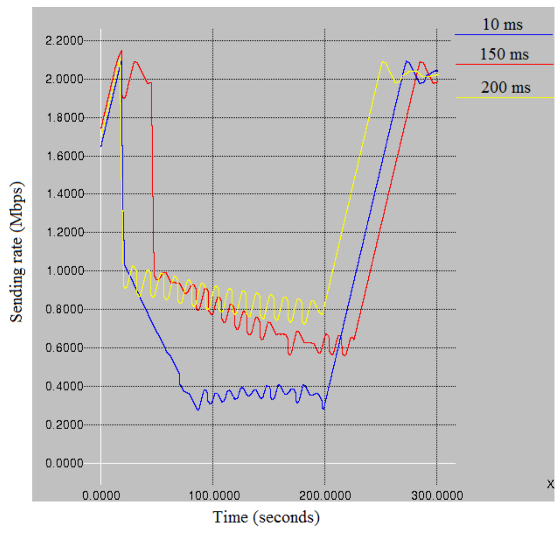


Figure 7: Back off behavior the backward loading mode.

5 CONCLUSIONS

The simulations and real network tests have shown that our end-to-end type dual-mode congestion control mechanism operates as desired. It makes possible to prioritize information distribution because non-delay sensitive data flows can be transferred with high speed only when the network load level is small enough. Of course, this kind of traffic prioritization could be implemented better by genuine Quality-of-Service mechanisms (Meddeb, 2010). However, at the moment, it seems that there is no willingness to implement the required quality-of service functionalities in routers.

Especially when testing the mechanism against itself, the results have been good. TCP friendliness is also at an acceptable level. However, these interpretations are based on a personal opinion because there are not any official criterions for fairness. But, for example, Floyd et al. (2008) defines that flows are "reasonably fair" if their sending rates are within the factor of two from one another. Instead of specifying a complete protocol, only the basic mechanism was defined in this study. Therefore, all kinds of special cases, such as the start phase of the connection, need extra study in the future.

As it usually does, the current state of the Internet set limitations for the perfect operation of our congestion control mechanism. First of all, existing networks are heterogeneous. The round-trip times of connections vary greatly. There are several TCP versions which differ at least to some extent. Our study has shown that relatively small differences between the versions can set big challenges for TCP friendliness. In addition, TCP's congestion control strongly favors connections of short round-trip times. This also set challenges for TCP-friendliness. Our understanding of the Internet congestion control and research results give an insight that congestion control can be implemented in a good working way if there are only a few compatible congestion control mechanisms on the Internet. As long as we have this heterogeneous environment, the only good working congestion control solution seems to be the currently widely used network overprovisioning. However, the question might arise if overprovisioning is too simple solution in the current technological world.

REFERENCES

Cisco. 2015. *Cisco visual networking index: forecast and methodology*, 2014-2019, Retrieved 06.04.2016 from http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html.

Donkor F. 2010. *The comparative instructional effectiveness of print-based instructional and video-based materials for teaching practical skills at a distance.* International Review of Research in Open and Distance Learning, 11(1). Pages 96-115.

Floyd S, Kohler E. 2003. *Internet Research Needs Better Models.* ACM SIGCOMM Computer Communication Review, Vol. 33, Issue 1. Pages 29-34.

Floyd S, Handley M, Padhye J, Widmer J. 2008. *"TCP Friendly Rate Control (TFRC): protocol specification"*. IETF RFC5348. Retrieved 06.06.2016 from https://www.ietf.org/rfc/rfc5348.txt.

Fabian B, Goertz F, Kunz S, Muller S, Nitzsche M. 2010. *Privately waiting–a usability analysis of the tor anonymity network.* Sustainable e-Business Management. Pages 63–75.

Ha S, Rhee I, Xu L. 2008. *CUBIC: a new TCP-friendly high-speed TCP variant.* ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel. Vol. 42 Issue. Pages 64-74.

Jansen S, McGregor A. 2006. *Performance, validation and testing with the Network Simulation Cradle.* MASCOTS 2006, 14th IEEE International Symposium on Modeling, Analysis, and Simulation. Pages 355-362.

Kurose J, Ross K. 2012. *Computer Networking: A Top-Down Approach.* 6th Edition, Pearson International Edition, 888 pages.

Lakshmi G, Bindu C. 2011. *A Queuing Model for Congestion Control and Reliable Data Transfer in Cable Access Networks.* IJCSIT, International Journal of Computer Science and Information Technologies, Vol. 2, Issue 4. Pages 1427-1433.

Lundin H, Holmer S, Alvestrand H. 2012. *“A Google congestion control algorithm for real-time communication on the World Wide Web”*. IETF informational Internet draft version 3. Retrieved 06.05.2016 https://tools.ietf.org/html/draft-alvestrand-rtcweb-congestion-03.

Meddeb A. 2010. "*Internet QoS: Pieces of the Puzzle*". IEEE Communication. Magazine, Vol. 48, Issue 2. Pages 86-94.

NS-2. 2011. A collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. 2013. *NS-2 Manual*. Retrieved 06.10.2016 http://www.isi.edu/nsnam/ns/doc/index.html.

Pandey A, Patni N, Singh M, Sood A, Singhd G . 2010. *YouTube As a Source of Information on the H1N1 Influenza Pandemic.* American Journal of Preventive Medicine, Vol 38, Issue 3. Pages 1–3.

Shalunov S, Hazel G, Iyengar J, Kuehlewind M. 2012. *“Low Extra Delay Background Transport (LEDBAT)”*. IETF RFC6817. Retrieved 06.06.2016 from https://tools.ietf.org/html/rfc6817.

Singh K, Yadav R, Manjul M, Dhir R. 2008. *Bandwidth Delay Quality Parameter Based Multicast Congestion Control.* IEEE ADCOM 2008, 16th International Conference on Advanced Computing and Communications. Pages 399-405.

tc. 2016. *tc(8) - Linux manual page.* Retrieved 06.06.2016 from http://man7.org/linux/man-pages/man8/tc.8.html.

Vihervaara J, Loula P. 2015. *Dual-Mode Congestion Control Mechanism for Video Service.* ICIMT 2015, 7th International Conference on Information and Multimedia Technology. Pages 50-56.

Widmer H, Denda R, Mauve M. 2001. *A Survey on TCP-Friendly Congestion Control.* IEEE Network, Vol. 15, Issue 3. Pages 28-37.