Congestion Control Supported Dual-mode Video Transfer

Juha Vihervaara, Teemu Alapaholuoma, Tarmo Lipping and Pekka Loula

Pori Campus, Tampere University of Technology, Pori, Finland
juha.vihervaara@tut.fi

Abstract. Transfer of videos over the Internet has increased considerably during the past decade and recent studies indicate that video services represent over half of the Internet traffic, with a growing trend. For the user-friendly operation of the Internet, it is important to distribute these videos in a proper and efficient way. However, no congestion control mechanism suitable and widely used for all kinds of video services is available. We have developed a congestion control mechanism, which is particularly suitable for long-living video traffic. The advantage of the proposed mechanism is its dual-priority nature. There is a mode for low priority traffic where the bandwidth is given away to other connections after the load level of a network exceeds a certain level. On the other hand, the real-time mode of the mechanism acquires its fair share of the network capacity. The real network tests of this study verify the proper operation of our congestion control mechanism.

Keywords: Congestion control · Video transfer

1 Introduction

Transfer of videos over the Internet has increased considerably during the past decade. Cisco forecasts that Internet video traffic play a big role also in the future [1]. It predicts that video traffic will form 82 percent of all consumer Internet traffic by 2020. Internet video to TV will continue to grow at a rapid pace, increasing 3.6-fold by 2020. Virtual reality based applications will also increase the video type traffic of the Internet. Videos are widely used because video-based solutions offer advantages and possibilities for many application areas. For example, in education, the use of video-based instructional materials often produce better learning results compared to the traditional print-based materials [2]. In addition, YouTube can be considered as an important tool for education [3].

Due to high popularity of video traffic, it is also an important cause of network congestions. Network operators have largely relied on overprovisioning and TCP congestion control to avoid congestions in their networks. However, unnecessarily high overprovisioning with high power consumption does not promote green Internet ideology [4]. Although some video services use TCP to implement their transport services in a manner that actually works, TCP's transport service is not suitable for all video applications. By implementing retransmissions, TCP offers reliable transport services to applications. Normally, a real-time video application does not require retransmissions because this type of applications are often loss-tolerant. Occasional packet drops do not degrade the quality of service experienced by the users of these applications. These packet drops can be alleviated by using the error correction properties of the applications. If the application is working in a real-time repeat mode, the order delivery property of TCP may cause problems. Due to the head-of-the-line blocking problem, the bytes following the missing ones cannot be delivered to the application. TCP's bursty-like transmission also causes delay jitters and sudden quality degradations because there can be abrupt and deep sending rate reductions. For these reasons, real-time video applications often prefer to use the unreliable UDP protocol. Unfortunately, UDP does not implement congestion control.

The approach of using congestion control only with TCP traffic has been appropriate in the past because TCP has represented major proportion of network traffic. However, nowadays UDP based long living communication events are common due to the popularity of various video services among consumers. It makes sense to equip these communication events with congestion control. This may offer new opportunities for old and new congestion control mechanisms to become deployed.

There are different kinds of ways to use video over the Internet. With live broadcasting, only a moderate buffering can be used at the receiver side due to the real time requirements. Therefore, delay requirements and bandwidth demands are important. On the other hand, in non-real-time applications where extensive buffering can be utilized at the receiver side and, therefore, delay and bandwidth demands are not important, some kind of background loading may be preferred. For example, the service provider can download content to proxy servers by using backward loading. The case can also be some kind of intermediate form. At first, the video can be transferred with the high speed. When there is enough data in the receive buffer, the transfer mode can be switched to the backward loading type.

So, two different kinds of transfer modes are needed in modern video services: a backward loading mode where delay and bandwidth demands are moderate and a real-time mode where delay and bandwidth demands are of high priority. Based on these different kinds of demands, the two modes also need different kinds of congestion control mechanisms. The backward loading mode has to work like a low-priority service in which the bandwidth is given away to other connections when the load level of the network is high enough. In contrast, the real-time mode always wants its fair share of the bandwidth.

Many congestion control mechanisms have been developed to be used either with low priority or with real-time services. However, little research effort has been put into developing a mechanism suitable for both modes. We recently developed this kind of integrated mechanism that supports both of these transfer modes. This mechanism was named Congestion control for VIdeo to Home Internet Service (CVIHIS). The algorithm was presented in the paper [5], where CVIHIS's performance was analyzed by extensive simulations. In the paper [6], we tested the operation of CVIHIS in real network environments. In these real network tests, CVIHIS was tested against itself more comprehensively than in the simulations. This paper re-presents and refines the results of the paper [6]. This study takes into account the situations where two congested routers simultaneously occur on the transmission path. The paper also analyzes how CVIHIS will work with the common problems of delay-based congestion control mechanisms

The paper is organized as follows: Section 2 outlines congestion control backgrounds; Section 3 introduces our dual-mode congestion control algorithm fir video services; Section 4 presents the test results and Section 5 concludes the paper.

2 Congestion Control

Congestion control principles of the Internet are presented in this section. Congestion control is a wide research area and only issues relevant for this study are introduced here.

2.1 Importance of Congestion Control

In its basic form, the Internet is built upon an assumption of best effort service. This means that the network does its best to deliver data packets to receivers as quickly as possible. On the other hand, the best effort principle also means that the network does not guarantee anything. It is not against Internet's laws that packets are queued or dropped inside the network. Congestion situations handled by queuing and dropping of packets are therefore fully acceptable. Of course, from a network user’s point of view, congestion is not a desired situation. Therefore, a network claiming to operate in a user friendly manner must implement some kind of congestion control.

If congestion control is implemented in an inoperative way, serious troubles may occur. When some part of the network is in a congested state, it queues traffic and packets may be dropped. Therefore, receivers do not receive the expected packets in time and senders cannot get acknowledgements inside the time limits. After that, senders, which are implementing reliable communication, will start to resend packets causing further congestion. This can lead to congestion collapse in which case only little useful communication is happening through the network [7].
Many reasons can lead to a congested network. The paper [8] specifies such kind of reasons: limited capacity of the routers, load of the network, link failures, heterogeneous bandwidths. The consequences of inoperative congestion control are discussed in [9]. They point out that in a congested network large queuing delays are experienced, which increases the response times of web services.

The background of congestion control is in queuing theory [10] as packets move into and out of queues when they pass through a network. Therefore, packet-switched networks can be considered as networks of queues. However, it is good to remember that extensive queuing inside the network is not a desired operation. Queue lengths should not reflect the steady condition we want to maintain in the network. Instead, they should reflect the size of bursts we need to absorb [11]. The goal of congestion control is to avoid a congestion situation in network elements. By another, more sophisticated definition, the target of congestion control is to adapt the sending rates of senders to match the available end-to-end network capacity. This definition emphasizes the fact that network-wide approaches must be used to implement congestion control. Otherwise, congestion is only shifted from one node to another. Therefore, in theory, we should monitor traffic in the whole network.

2.2 Congestion Control Mechanisms for Video Services

Because the TCP and UDP protocols are not completely suitable for video services, there is the need for a protocol that takes into account the requirements of video traffic. In this section, some congestion control algorithms, suitable for video traffic, are presented. Each of these algorithms is suitable for either a low priority or a real-time service. None of them has been developed with both these service types in mind.

LEDBAT [12] is designed for low priority applications. It has been used in some background bulk-transfer applications such as BitTorrent, for example. It provides low priority services by using one-way delay measurements to estimate the amount of queued data on the data path. When the estimated queuing delay is less than the predetermined target, LEDBAT concludes that the network is not yet congested and it increases its sending rate to utilize the free capacity of the network. When the estimated queuing delay becomes larger than the predetermined target, LEDBAT decreases its sending rate as a response to the potential congestion. The sending rate is increased and decreased more aggressively if the queuing delay is far from the target. TCP-LP [13] is another delay-based congestion control protocol for low priority services.

The next two algorithms are suitable for the real-time mode as they want their fair share of the bandwidth. The best known proposal for video services is DCCP [14] and its TCP Friendly Rate Control version [15], abbreviated as TFRC. DCCP offers congestion control for UDP-like unreliable applications. DCCP can be briefly described as TCP without byte-stream semantics and reliability, or as UDP with added congestion control, handshakes and acknowledgments for congestion feedbacks. The main issue with DCCP's congestion control is that the congestion control is not a part of DCCP itself but DCCP allows applications to choose from a set of congestion control mechanisms. Therefore, different kinds of congestion control mechanisms can be used with DCCP, TFRC being one of them. TFRC uses a throughput equation to calculate the allowed sending rate. Because DCCP tries to be fair against TCP, it is natural that TFRC uses the TCP throughput equation. TFRC is designed for applications that require smooth rate. Therefore, TFRC responds to the changes of the available bandwidth more slowly than TCP.

Google Congestion Control for Real-Time Communication [16] is a new proposal in this area. It defines two congestion control methods: one for the sender side and another for the receiver side. Either both or only one of these methods can be used. The receiver side uses delay gradients in a sophisticated way to detect congestions. The sender side method is based on information about round-trip times and packet losses.

One possibility to achieve a dual-mode congestion control mechanism, such as the one presented in this study, would be to put together the best low-priority and real-time congestion control algorithms. However, this kind of implementation would be ungainly, especially when the mode has to be changed on the fly. The real dual-mode mechanism presented in this study allows the change between the modes in a seamless way.

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. Often the term TCP friendliness is used as in the past years most of the traffic flows were TCP flows and the TCP protocol has traditionally been responsible for the congestion control of the Internet. Therefore, it is a natural choice to compare a new mechanism against the TCP protocol. The basic idea is to protect existing TCP flows from the flows that use too aggressive congestion control mechanisms.

Unfortunately, TCP friendliness is a complicated concept. Even a TCP flow itself is not always friendly against another TCP flow. Several versions of the TCP protocol exist and these versions are not completely identical in their behaviours. In addition, TCP's throughput degrades in case of higher round-trip times (RTT) [17]. Therefore, TCP has a bias against high-RTT connections giving preference to the users with short RTTs. Several improvements such as the Delayed ACK mechanism [18], for example, have been suggested to make TCP congestion control work in a better way. Unfortunately, only some TCP implementations have adopted these improvements and, therefore, different code implementations behave in different ways. Due to this, even identical TCP implementations are not equal. Another problem is that there is no 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

The algorithm of CVIHIS is introduced in this section with brief description of the implementation principles of CVIHIS for real network tests.

3.1 Basic Properties of the Algorithm

CVIHIS is a receiver-based mechanism so that most of the processing can be done at the receiver side instead of the heavy loaded server side. It complies with the end-to-end approach, which states that complex issues should not reside in routers. Because the sending rates of video applications should usually vary in a smooth way, CVIHIS is a rate-based congestion control approach. The window-based control is seldom suitable for continuous multimedia streaming because it tends to produce bursty-like traffic behavior [19]. Exponentially Weighted Moving Average (EWMA) is also used by CVIHIS to filter out quick rate changes.

If the network does not deliver explicit congestion feedbacks, the sending rate adjustment can only be based on packet losses or delays. Both indicators are utilized by

CVIHIS, but the algorithm is somewhat more delay-based than loss-based. The reason for emphasizing the delay-based approach is that it generates suitable conditions for implementing the low-priority behavior [20]. CVIHIS uses one-way delays in delay measurements so that the necessary conclusions can be made at the receiver side. However, using one-way delays also has other benefits that are explained by the paper [21].

3.2 Backward Loading Mode

Fig. 1 presents the rate adaptation schema of the backward loading mode. The algorithm of CVIHIS keeps track of two delay values, minDelay and maxDelay, based on one-way delay measurements. The minDelay value corresponds to the situation when the queues of the routers are empty on the connection path. The minDelay value includes only propagation delay components, not queuing delays. The minDelay value is the shortest delay value experienced during the lifetime of the connection. Instead, the maxDelay value includes the queuing delay component. It corresponds to the situation in which the buffer of a router overflows. Therefore, maxDelay is updated every time when a packet drop occurs. CVIHIS uses the delay value of the last received packet prior to the dropped packet for the maxDelay value.

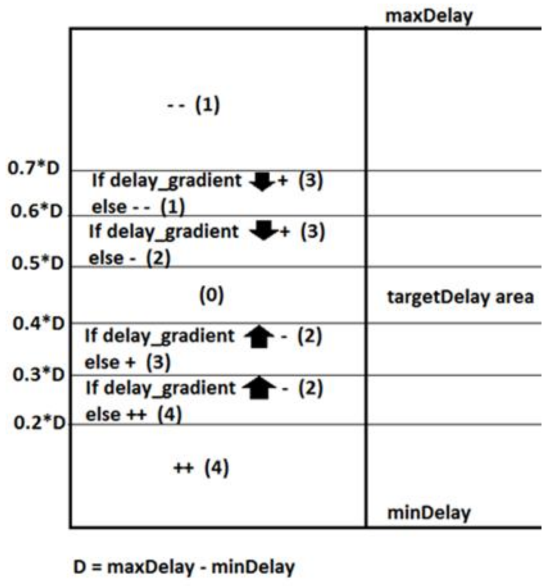

Fig. 1. Rate adaptation schema of CVIHIS (Source: [5])

With the help of these two delay values, the delay space is divided into seven rate adaptation areas. It could also be said that the queue of the router is divided into several corresponding parts. The seven rate adaptation areas are used so that sufficiently accurate information about the state of the network can be provided to end-hosts. CVIHIS' objective is that it tries to keep the queue at the level of the target delay area. When operating in the upper delay areas, CVIHIS decreases its sending rate and, when operating in the lower delay areas, CVIHIS increases its sending rate. The positioning factors for each delay area are presented on the left side of Fig. 1. The targetDelay area is not placed in the middle of the delay space but is shifted somewhat downwards so that queues can be kept short.

The black arrow inside some of the delay areas in Fig. 1 represents the delay gradient obtained by comparing the delay values of two consecutive packets. If the arrow points upwards, delays are increasing, delay gradient is positive, and the queue is filling. If the arrow points downwards, delays are decreasing, delay gradient is negative, and the queue is emptying. Inside the four delay areas with the arrows, the rate adaptation command is based on the actual delay value and the value of the delay gradient. The rate adaptation scheme tries to achieve two targets: it tries to drive the queue level to the target delay area by measuring the actual delay value and, on the other hand, it tries to adapt the sending rate according to the bottleneck capacity. This is done by means of the delay gradient. If there is a conflict between the delay area and delay gradient adaptation, the gradient adaptation is chosen. The two extreme delay areas do not use delay gradients for rate adaptation decisions because these areas are far away from the targetDelay area.

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 a round-trip time interval. By using square root, CVIHIS alleviates the favoring behavior of short distance connections.

CVIHIS adjusts its sending rate through seven adjustment steps. Six of these steps are presented in Fig. 1. Bigger steps are used when the queue level is further away from the target. In Fig. 1, the step sizes are denoted by different number of + or – marks. If there are three marks, CVIHIS increases or decreases its transmission rate by one packet for each square root of the round-trip time. If there are two marks, the adjustment steps are smaller. The smallest steps are indicated by one + or – mark. To enter the targetDelay area in a smooth way, CVIHIS uses short steps in the delay areas just beside the targetDelay area (rate adaptation feedbacks 2 and 3). The adjustment steps related to the delay gradients (rate adaptation feedbacks 6 and 7) are the shortest ones. The seventh adjustment step is a multiplicative decrease step taken after a packet drop. The multiplicative decrease step is taken only once per a round-trip time cycle.

Table 1 presents the rate adjustment factors of CVIHIS. These factors are set so that CVIHIS can compete in a fair manner with the TCP NewReno version. The integer values in brackets refer to the rate adjustment commands of CVIHIS presented in Fig. 1. All decision procedures related to Fig. 1 are implemented at the receiver side. Only the rate adaptation commands are transmitted to the sender. In the case of four leftmost columns, the rate adjustment is based on the square root of the round-trip time. The factor expresses how many more or less packets will be sent during the next square root of the round-trip time than just before. MD is a multiplicative decrease factor used after packet drops to increase the sending gap of packets. SF is a smoothing factor used for the EWMA filter to filter out quick rate changes. PF is a pushing factor used only by the real-time mode.

Table 1. Adjustment parameters of CVIHIS (Source: [6])

--- (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

3.3 Real-Time Mode

The backward loading mode backs off when it competes with TCP. In order to be suitable for the real time mode, the implementation code has to be modified so that it will behave in a more aggressive way. On the other hand, it is desirable that the code implementation of the backward loading mode is modified as little as possible. Both of these goals can be achieved in a simple way by using an approach in which the minimum delay value is pushed upwards in a continuous manner. This means that the delay areas of Fig. 1 are also pushed upwards and, therefore, CVIHIS behaves in a more aggressive way. Shifting the delay areas upwards is only done when competing behavior is actually needed. If the last measured delay value is smaller than the pushed minimum delay value, the minimum delay value is set to the value of the last measured delay.

This kind of minimum delay pushing means that the real-time mode of CVIHIS is not a delay-based congestion control solution any more. The pushing operation shifts this version towards loss-based congestion control. Therefore, the real-time mode of CVIHIS is a kind of hybrid solution, a delay-loss-based solution. The minimum delay value is pushed upwards in a multiplicative way. It was found that the pushing factor of 1.05 is suitable.

It is worth noting that CVIHIS is a pure congestion control mechanism. If the application is delay sensitive, delay requirements must be satisfied by Quality-of-Service mechanisms [22]. The dual mode mechanism presented in this study can also be achieved using Quality-of-Service techniques. In this case, the mechanism could be called a dual priority mechanism. However, this kind of mechanism can not be a pure end-to-end mechanism as the implementation would require network support at least to some extent.

3.4 Software Implementation

The code implementation of CVIHIS should be placed somewhere in the protocol stack to enable real network tests. There are several possibilities for this placing. For example, the kernel implementation of an open source operating system could be modified. In this way, the UDP implementation of the operating system could be adjusted to correspond to CVIHIS' algorithm. Instead of using this kind of elaborate solution, an easier implementation option was chosen. CVIHIS was implemented through a normal socket program on top of the UDP protocol. This solution is possible because the UDP protocol does not offer any special transport services, which could disturb the operation of CVIHIS. Unlike in simulation environment, the real network implementation should take into account that the clocks of the source and receiving ends have not been synchronized with each other. Therefore, the real network version has to obtain round-trip times by actual measurements. When a packet is transmitted, the sender side stores the transmission timestamp. When the corresponding acknowledgment arrives, the round-trip time is calculated.

In addition, this implementation option provides further advantages. Same computer can be used to run a few traffic sources in parallel because these traffic sources can be separated from each other by means of UDP port numbers. This reduces the number of computers needed for the test network. The solution also offers resistance against firewall blocking if large scale real network testing is done in the future.

4 Test Results

In this section, the structure of the test network is described and the most relevant results of the network tests are presented. This study presents the results related to the paper [6] in an extended way. The test results of CVIHIS shows that there is room for improvements in some cases. In these cases, there can be present connections which posses very different round-trip times. As it is mentioned, TCP favors the connections of short round-trip times, therefore, it is necessary to adjust the algorithm of CVIHIS to favor short round-trip times as well. However, CVIHIS performs this favoring in a more moderate way than TCP. In fact, it is not necessarily a bad idea to favor the connections of shot round-trip times. Often these short connections consume less network resources than long way connections. By favoring connections of short round-trip times, network operators can maximize the total traffic volume in their networks. The challenge is to find a suitable balance so that fairness does not suffer too much.

4.1 Test Network

Fig. 2 presents the structure of the test network. There are four end nodes and three routers. The links between the routers form the bottleneck links of the connection path. With this simple test network structure, and with the help of the tc (traffic control) program, it is possible to emulate different types of networks. Tc [23] is the Linux utility program used to configure the kernel packet scheduler.

Tc is utilized in two ways to vary the configuration of the test network. Tc is used to control the traffic in the Linux routers. In this way, the capacity of the bottleneck link and the queue size of the link can be varied. When traffic is controlled, the transmission rate of the link is under control. Typically, this means that the available bandwidth is decreased. Traffic control can also be used to smooth the burstness of incoming links by defining the queue size of the link. If the queue size is exceeded, the incoming packets are dropped. At the end nodes, tc is used to define the delay characteristics of the outgoing links. In this way, it is possible to emulate different round-trip times of connection paths.

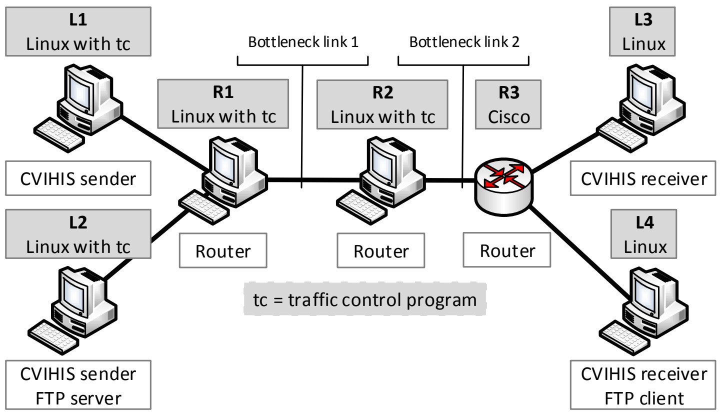

Fig. 2. Structure of the test network

4.2 Backward Loading Mode

The goal of the backward loading mode is to achieve stable sending rate if there is no need for the backoff function. This was ensured by performing twenty tests in the test network. In the test 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.

Another objective of this mode is proper backoff behavior. This was verified against one TCP NewReno connection by using the same kinds of test setups as in the case of the stability check. The proper backoff behavior was observed in all cases. In Fig. 3, the sending rates of the CVIHIS connections are presented in test cases, where the backoff behavior was verified by using three different round-trip times (10, 100 and 200 ms) for CVIHIS. TCP used the round-trip time of 200 milliseconds in all these cases. The capacity of the bottleneck link was 3 Mbps. The TCP connections were active between the test time of about 50-170 seconds. As it can be seen, CVIHIS increases its sending rate faster if the round-trip times are short.

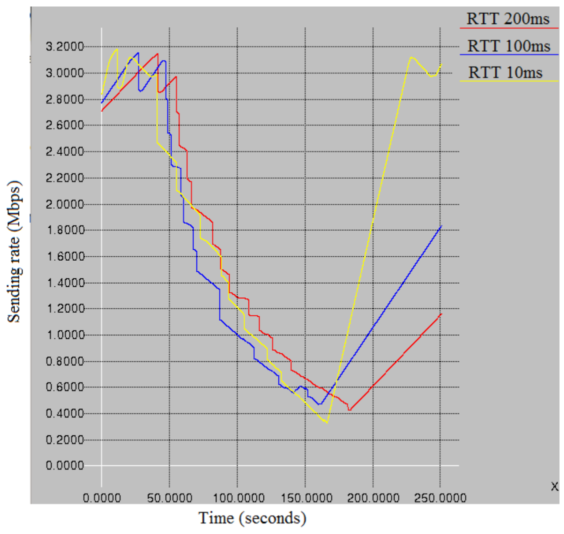

Fig. 3. Simulation results of the backward loading (Source: [6])

4.3 Real-Time Mode

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

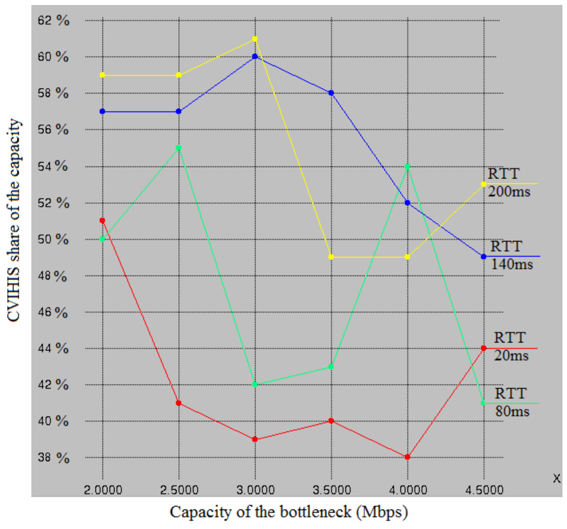

Fig. 4. Real-time mode against one TCP connection (Source: [6])

Fig. 4 presents the test results. As it can be seen, individual measurements depart from the trend due to the phase effect. The figure presents the proportion of the CVIHIS connection from the capacity of the bottleneck link. The figure indicates acceptable level of averaged fairness. In the worst case, the connection of higher bandwidth gets about 1.6 times as much bandwidth as the slower connection.

As mentioned earlier, TCP favors the connections of short round-trip times while CVIHIS does this in a more modest way. The above results confirm this. Round-trip times affect CVIHIS less than TCP. CVIHIS manages relatively modestly when round-trip times are short. When round-trip times are long, CVIHIS manages somewhat better than TCP.

The Linux version used in the real network tests also supports another TCP congestion control mechanism. This version is CUBIC TCP [24]. In fact, CUBIC is the current default TCP algorithm of Linux. Therefore, CVIHIS was also tested against the CUBIC version. The preliminary results show that CUBIC behaves somewhat more aggressively than NewReno. If CVIHIS is desired to manage in a friendly way against 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

The results of the previous subsection and the paper [5] show that it is challenging to attain acceptable level of fairness in heterogeneous network environments. Hence, implementing a well-performing solution for network congestion control might require that there are only a few kinds of congestion control mechanisms on the Internet. Thus, it is important that CVIHIS behaves in a fair manner also against itself.

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

Fig. 5 presents the result of one of the tests. The capacity of the bottleneck link is 4 Mbps and the round-trip times of the connections are 60 (red) and 180 (blue) milliseconds. In this case, the average sending rates are 2.085 Mbps and 1.935 Mbps. The first 50 seconds where omitted when calculating averaged rates.

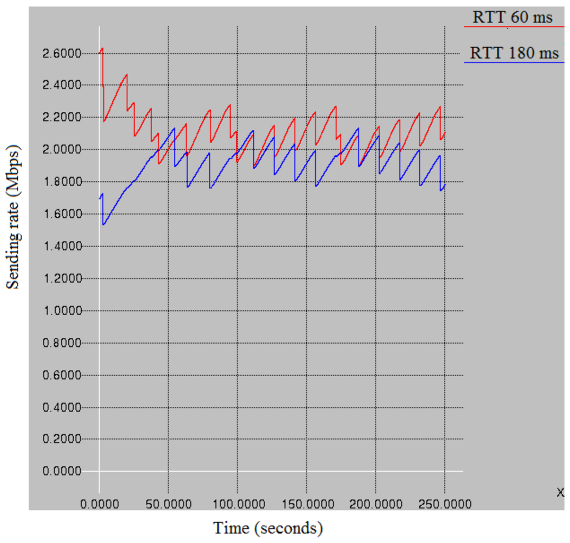

Fig. 5. CVIHIS real-time mode against itself (Source: [6])

Some tests were made involving four CVIHIS connections in the active state at the same time. CVIHIS performed in an acceptable way in these tests although it took more time to balance the sending rates when round-trip times were long. Fig. 6 presents the result of one such test. In this case, the capacity of the bottleneck link was 10 Mbps and the queue size of the bottleneck link was 60 packets. The round-trip time was 30ms. The number of connections in the active state is shown at the bottom part of this figure.

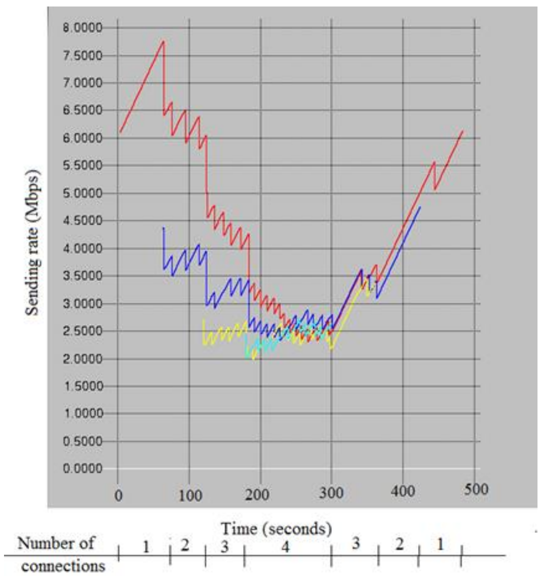

Fig. 6. Four real-time mode connections (Source: [6])

The backoff behavior of CVIHIS in the backward loading mode was also tested when there was a real-time connection on the connection path at the same time. Fifty tests were made so that the capacity of the bottleneck link was 2 or 4 Mbps and the queue size of the bottleneck link was 60 packets. The round-trip times varied from 10 to 200 milliseconds.

When both modes had the same round-trip time, the backoff action was as expected. Tests were also performed using different round-trip times for the modes. In these tests, the backoff action occurred slowly if the real-time mode connection had significantly longer round-trip time than the backward loading mode connection. When the round-trip times differed considerably, ten times, for example, backoff action did not take place at all. Fig. 7 illustrates the above-mentioned behavior. In this figure, the sending rate of the backward loading mode is presented in three separate cases. The round-trip times of the real-time mode connection are 10, 150, and 200 milliseconds in these cases. In all these cases, the round-trip time of the backward loading mode connection is 50 milliseconds. It is fairly easy to moderate this phenomenon by adjusting the parameters of the backward loading mode so that it would behave less aggressively than the real-time mode. In this study, both modes shared the same parameter set.

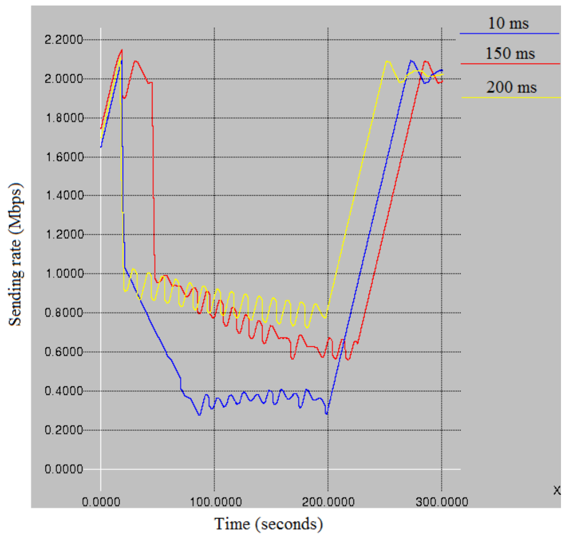

Fig. 7. Backoff behavior of the backward loading mode (Source: [6])

4.5 Case of Two Queues

There has been only one non-empty queue on the connection path in the previous test cases. As far as this single non-empty queue condition is met, the behaviour of our test network structure is compatible with that of more complicated network structures. The location of this non-empty queue can change if the size of the queue and the bandwidth of the out-going link remain similar. The more complicated network structures have been taken into account by using different one-way propagation delay values in the previous test cases. However, in real networks, it can happen that there are several non-empty queues on the connection path at a certain moment. In this subsection, the case in which there are two non-empty queues on the connection path is tested. The case of two non-empty queues affects especially the maximum delay value so that it is not static any more.

Now there are three end nodes that send traffic by using the real-time mode of CVIHIS. The third source is located in Linux Router 2. The receiver of this third connection is located in the receiving hosts L4. This host runs two receiving processes at the same time. There are now two bottleneck links which reside in the Linux routers. The connection R2-L4 has only one bottleneck while the other two connections have two bottlenecks. We want to test the case in which there are occasionally two non-empty queues on the connection path. So, the capacity of the second bottleneck link should be 1.5 times as much as the capacity of the first bottleneck link, or a little bit more. This is because the first link has two connections and the second link three connections.

Six tests were made to study CVIHIS' fairness against itself. The queue sizes of the bottleneck links were 40 packets and one-way propagation delays were 50 ms in all the cases. The results of these tests are presented in Table 2. The second and third columns present the capacity of the bottleneck links. The actual test results are presented in the last three columns. These columns present the sending rates of the connections and the standard deviations of CVIHIS' sending rates. The standard deviations are presented inside the parentheses. Based on these results, it can be said that the sending rates indicate good level of fairness. We also carried out tests, in which one of these three connections owned the one-way propagation delay value of 150 ms. These tests also indicated good level of fairness. In the worst cases, the faster comparable connection gets about 1.15 times as much bandwidth as the slower connection.

Table 2. Ten tests for testing CVIHIS' friendliness with two queues

	Capacity of link R1-R2 [kbps]	Capacity of link R2-R3 [kbps]	CVIHIS 1 L1-L3 [kbps]	CVIHIS 2 L2-L4 [kbps]	CVIHIS 3 R2-L4 [kbps]
1	4000	6000	2008 (33)	2050 (32)	-
2	4000	6000	2005 (47)	2045(42)	2040 (53)
3	4000	6500	1929 (76)	2005 (86)	2646 (75)
4	6000	9000	3049 (58)	3053 (57)	-
5	6000	9000	3047 (82)	3042 (87)	3048 (60)
6	6000	10000	3081 (83)	3014 (84)	3968 (168)

The results indicate that CVIHIS' sending rate varies more in the case of two queues than in the case of one queue. This can be seen when comparing the results of the rows 1 and 2 to each other. The same is also true for the rows 4 and 5. This is because the maximum delay value related to the packet drop situations is not static any more. The maximum delay value varies according to the level of the non-full queue. This CVIHIS' sending rate fluctuation can also be seen in Fig. 8. In this figure, the third sending node is sending between 70 and 170 seconds. As can be seen, the sending rates of the two other connections vary more in the middle phase of the test when there are three connections and two bottleneck links in the active state.

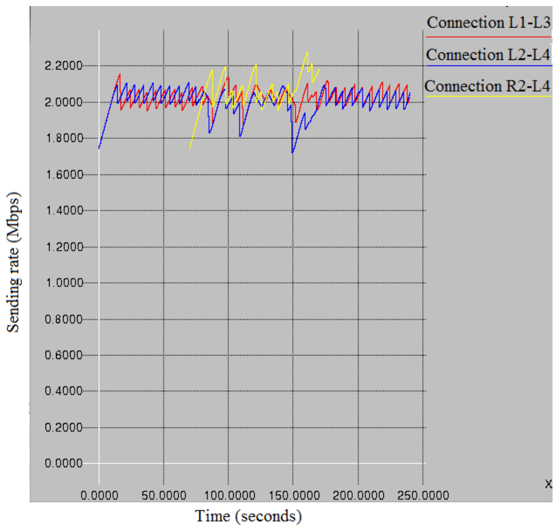

Fig. 8. Test case considering two bottleneck links

4.6 Advantages of the Minimum Delay Value Pushing

Delay-based congestion control mechanisms have some well-known problems which can affect their performance. The papers [25] and [26] list and analyze these problems. Based on these papers, the common problems of delay-based congestion control mechanisms are:

- inability to compete fairly against loss-based congestion control protocols
- persistent congestion
- clock synchronization problem if one-way delay measurements are used
- rerouting problem.

In this subsection, CVIHIS' capability to cope with these problems is explained, although CVIHIS' real-time mode is not a pure delay-based mechanism. It is important to note that these problems can be solved with the help of the minimum delay value pushing in CVIHIS.

Competing against loss-based congestion control protocols is not a big problem for the real-time mode of CVIHIS. Here, CVIHIS was tested against the loss-based TCP protocol. It was observed that the real-time mode of CVIHIS is actually capable in competing against TCP. This mode can compete against loss-based algorithms because CVIHIS shifts its target delay area upwards when competing behavior is needed.

In persistent congestion, the queue of the router is occupied all the time. As a result, delay-based congestion control mechanisms cannot obtain proper value of the minimum delay. The paper [25] suggest that shifting of the minimum delay value alleviates the persistent congestion problem. CVIHIS pushes its delay areas upwards by shifting the minimum delay value. This increases the congestion level of the network leading finally to packet drops. Many connections back off after these packet drops. They reduce their sending rates in a multiplicative manner and the congestion level of the network alleviates. This allows CVIHIS to estimate the correct value of the minimum delay.

The problem with measuring one-way delays is that the clocks of the devices are typically not synchronized accurately in the Internet. Therefore, the one-way delay measurement includes the corresponding one-way delay and the clock offset between the nodes. Even if initially accurately synchronized, two clocks will differ after some time due to clock drift. Due to clock offset and clock drift, one-way delay measurements are challenging.

For CVIHIS, clock offset is not a problem as CVIHIS probes two delay values, minDelay and maxDelay (see Fig. 1) and divides the delay space between these values into several delay areas. CVIHIS can do this correctly if maxDelay is greater than minDelay even if these delay values are negative due to the clock offset. The actual one-way delay measurement related to a certain packet includes the same clock offset and, therefore, the calculated delay is within the minDelay-maxDelay area.

Clock drift, however, can cause problems for CVIHIS. If the measured delay value is increasing due to clock drift, the minDelay value will become outdated. After a certain period of time, the minDelay value does not correspond to the actual propagation delay of the connection path any more. In an extreme situation, the measured delay value including only the propagation delay component may reside closer to maxDelay than minDelay. This means that the connection makes a conclusion that there is an incipient congestion in the network although the queues of the routers are completely empty. This problem can be solved by updating the minDelay value from time to time. In this way, pushing the minimum delay value upwards helps to cope with the clock drift problem. The maxDelay value will be updated after every packet drop, therefore, the clock drift is not critical for the maxDelay. The real network tests of CVIHIS indicated that the clock synchronization problem is not harmful for CVIHIS because the tests were carried out without the synchronization of clocks.

If the route of a connection is changed without an explicit signal from the network, the end host cannot detect it. If the new route has a shorter propagation delay, this does not cause any serious problem for CVIHIS as some packets will probably experience shorter one-way delay values and the minimum delay value will be updated. The maximum delay value will also be updated after the next packet drop. On the other hand, if the new route has a longer propagation delay than the original one, it can pose a problem to CVIHIS. The connection cannot know whether the increase in the delay is due to a congestion in the network or change of the route. Without this knowledge, the end host will interpret the increased delay as a signal of a congestion and the host will decrease its sending rate.

In the following, the rerouting properties of CVIHIS are tested using a simulation. This test uses the real-time mode version of CVIHIS. The ability of CVIHIS to discover rerouting is based on pushing of the minimum delay value and updating the maximum delay value. The maximum delay value is updated after every packet drop. In the simulation, there are two possible routes between the end nodes. There is a direct default route, which is switched off two times during the simulation. So, the traffic has to be switched to the backup route, which has longer propagation delay than the default route. The default route is switched off between seconds 80-150 and 220-320. The capacity of the default route is 700 kbps and the capacity of the backup route is 400 kbps. The simulation result is presented in Fig. 9. As can be seen, CVIHIS can observe the route changes and it can accommodate its sending rate according to the new route.

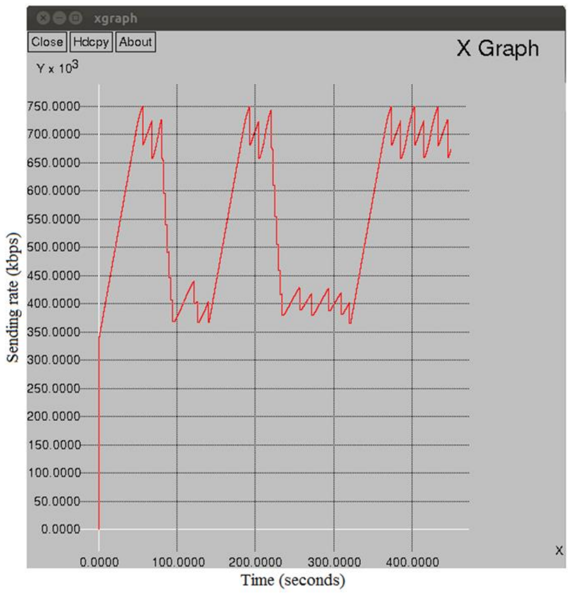

Fig. 9. Rerouting test of the real-time mode

The problems of delay-based congestion control mechanisms suggest that it is perhaps useful to update the minimum delay value from time to time also in the backward loading mode. Continuous shifting of the minimum delay value could be used also with the backward loading mode. Of course, the shifting factor of this mode should be only slightly over one as we do not want to compete against other connections with this mode. The paper [25] introduce another kind of solution for updating the minimum delay value. After receiving a certain number of data packets, the receiver can check the smallest delay value among these packets. If the difference between the smallest delay value and the current minimum delay value is larger than a certain threshold for a certain number of consecutive times, the receiver interprets this as a change of the propagation delay.

5 Conclusions

During the last decade, video-type data services in their various forms have become increasingly common. In certain parts of the network, this type of data transmission generates considerably more than half of the total network traffic. We have developed a congestion control mechanism, which is particularly suitable for long-living video transfer. This mechanism includes two modes, the backward loading mode and the real-time mode.

The main objective of the backward loading mode is to back off when there are bandwidth demands from other connections. Based on the test cases, we can conclude that the backward loading mode operates primarily as expected. This mode gives bandwidth away to other connections when the load level of the network is high enough. The main objective of the real-time mode is that it should be TCP-friendly. At the same time, however, it is desirable that the sending rate of this mode would vary in a much smoother way than TCP's sending rate. Based on the tests, it can be said that these objectives are met, however, as usually with this kind of solutions, not in a perfect way. The developed mechanism could manage better regarding TCP friendliness when short or long round-trip times are considered. On the other hand, it is a deliberate decision to change the sending rate of CVIHIS based on the square root of the round-trip time instead of using TCP's round-trip time approach.

The current state of the Internet presents challenges related to the proper operation of Internet congestion control. The Internet is a very heterogeneous environment with different types of applications sending different amounts of data through different kinds of network paths. Based on our research results, relatively small differences between the TCP versions can challenge TCP friendliness. For this kind of heterogeneous environment, the only well-functioning congestion control solution seems to be the currently widely used network overprovisioning. Based on our research we suggest that Internet congestion control can be put into practice in a well-functioning way if there are only a few compatible congestion control mechanisms present. Thus, it is important that a congestion control mechanism behaves against itself in a fair manner. The results of this study show that the CVIHIS connections are able to share the network capacity with each other in a fair manner.

Several research directions can be mentioned for future work. In all the test cases of this study, fixed packet size was used. There is a need to take into consideration heterogenous packet sizes. For example, rate adaptation could be based on the number of bytes rather than the number of packets. There is also a need for research collaboration with specialists of other research areas. One such kind of problem is how the rate adaptation behavior of the application could be integrated with the mechanism of CVIHIS. It is an application specific issue if the real-time mode is smooth enough for the needs of a particular application, because the sending rate of this mode oscillates slightly due to the pushing of the minimum delay value.

References

1. Cisco: Cisco Visual Networking Index: Forecast and Methodology, 2015-2020. Retrieved 02.02.2016 from http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html. (2016).
2. Donkor, F.: 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. (2010).
3. Pandey, A., Patni, N., Singh, M., Sood, A. and Singhd, G.: YouTube As a Source of Information on the H1N1 Influenza Pandemic. American Journal of Preventive Medicine, Vol 38, Issue 3, Pages 1-3. (2010).
4. Bianzino, M. Chaudet, C., Rossi D. and Rougier J.: "A survey of green networking research," IEEE Commun. Surveys & Tutorials, Vol. 14, Issue 1, Pages 3-20. (2013)
5. Vihervaara, J. and Loula, P.: Dual-Mode Congestion Control Mechanism for Video Service. ICIMT 2015, 7th International Conference on Information and Multimedia Technology, Pages 50-56. (2015).
6. Vihervaara, J., Alapaholuoma, T. and Loula, P.: Dual-Priority Congestion Control Mechanism for Video Services, Real Network Tests of CVIHIS. Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Vol. 3: KMIS. Pages 51-59. (2016).
7. Floyd, S. and Fall, K.: Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transactions on Networking, Vol. 7, No. 4, pp. 458-472. (1999).
8. Singh, K., Yadav, R., Manjul, M. and Dhir, R.: Bandwidth Delay Quality Parameter Based Multicast Congestion Control. IEEE ADCOM 2008, 16th International Conference on Advanced Computing and Communications, Pages 399-405. (2008).
9. Kurose J, Ross K. 2012. Computer Networking: A Top-Down Approach. 6th Edition, Pearson International Edition, 888 pages. (2006).
10. Lakshmi, G. and Bindu, C.: 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. (2011).
11. Braden, R., Clark, D., Crowcroft, J., David, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Petterson, L., Ramakrisman, K., Shenker, S., Wroclawski, J. and Zhang, L.: Recommendations on Queue Management and Congestion Avoidance in the Internet. IETF RFC 2309, Informational, 17 pages. (1998).
12. Shalunov, S., Hazel, G., Iyengar, J. and Kuehlewind, M.: "Low Extra Delay Background Transport (LEDBAT)". IETF RFC6817. Retrieved 06.06.2016 from https://tools.ietf.org/html/rfc6817. (2012).
13. Kuzmanovic, A. and Knightly, E.: "TCP-LP: low-priority service via end-point congestion control," IEEE/ACM Transactions on Networking, vol. 14, pp. 739-752. (2006).
14. Kohler, E., Handley, M. and Floyd, S.: "Datagram Congestion Control Protocol (DCCP)," IETF RFC4340, Available: https://www.ietf.org/rfc/rfc4340.txt. (2006).
15. Floyd, S., Handley, M., Padhye, J. and Widmer, J.: "TCP Friendly Rate Control (TFRC): protocol specification". IETF RFC5348. Retrieved 06.06.2016 from https://www.ietf.org/rfc/rfc5348.txt. (2008).
16. Holmer, S., Lundin, H., Carlucci, G., De Cicco, L. and Mascolo, S.: "A Google congestion control algorithm for real-time communication on the World Wide Web". IETF informational Internet draft, Retrieved 02.02.2017, https://tools.ietf.org/html/draft-alvestrand-rmcat-congestion-03. (2015).

17. Widmer, H., Denda, R. and Mauve, M.: A Survey on TCP-Friendly Congestion Control. IEEE Network, Vol. 15, Issue 3, Pages 28-37. (2001).
18. Braden, R.: Requirements for Internet Hosts -- Communication Layers. IETF RFC 1122, 116 pages. (1989).
19. Akan, Ö.: On the throughput analysis of rate-based and window-based congestion control schemes. Computer Networks, vol. 44, Pages 701-711. (2004).
20. Ros, D. and Welzl, M.: Less-than-best-effort service: A survey of end-to-end approaches. IEEE Communications Surveys & Tutorials, Vol. 15, No. 2, Pages 898- 908. (2013).
21. Almes, G., Kalidindi, S. and Zekauskas, M.: A One-way Delay Metric for IPPM. IETF RFC 2679, 19 pages. (1999).
22. Meddeb, A.: "Internet QoS: Pieces of the Puzzle". IEEE Communication. Magazine, Vol. 48, Issue 2, Pages 86-94. (2010).
23. tc: tc(8) - Linux manual page. Retrieved 06.06.2016 from http://man7.org/linux/man-pages/man8/tc.8.html. (2016).
24. Ha, S., Rhee, I. and Xu, L.: 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. (2008).
25. La, R., Walrand, J. and Anantharam, V.: Issues in TCP Vegas. Retrieved 02.03.2017 from http://www.eecs.berkeley.edu/~ananth/1999-2001/Richard/IssuesInTCPVegas.pdf, 17 pages. (1999).
26. Rodríguez-Pérez, M., Herrería-Alonso, S., Fernández-Veiga, M. and López-García, C.: Common problems in delay-based congestion control algorithms: a gallery of solutions. European Transactions on Telecommunications, Vol. 22, Issue 4, Pages 168-178. (2011).