

# **PERFORMANCE COMPARISON OF THE DIFFERENT STREAMS IN A TCP BOTTLENECK LINK IN THE PRESENCE OF BACKGROUND TRAFFIC**

**Igli Tafa<sup>1</sup>, Elior Vila<sup>2</sup>, Ezmerina Kotobelli<sup>1</sup>, Alda Stefani<sup>1</sup>, Jona Dushi<sup>1</sup>**

<sup>1</sup> *Polytechnic University of Tirana, Information Technology Faculty*

<sup>2</sup> *University of Elbasan, Information Technology Faculty*

*e-mails: itafaj@gmail.com; elior.vila@uniel.edu.al; ekotobelli@fti.edu.al;  
aldastefani92@gmail.com; jona.dushi@fti.edu.al*

*Albania*

**Abstract:** The purpose of this work consists in the evaluation of the TCP streams performance, when the link that causes bottleneck is also shared by the background traffic. Standard TCP is the most used protocol which sustains the majority of Internet traffic. Nevertheless TCP manifests some problems when using almost all of the available bandwidth. Based on this problems are realized different versions of TCP. The aim of this paper is to test the conflict between TCP flows bandwidth allocation. We have implemented a network which includes FTP traffic and background traffic. TCP flows are simulated so that they begin transmitting at different time. Background traffic is added besides TCP flows so that we approach a real network model. Besides studying how TCP flows compete with each other, we will evaluate if the background traffic has an impact on the behaviour of TCP flows and if it influences how the available bandwidth is shared equally among flows. We compare standard TCP Tahoe and TCP Reno, which do not differ much between each-other, but they use different algorithms so through simulation we will evaluate the changes between the bandwidth they use, even though we expect TCP flows to be more aggressive in getting the bandwidth from other TCP flows.

**Key words:** TCP Stream, FTP Traffic, TCP Tahoe, TCP Reno.

## **1. INTRODUCTION**

TCP and UDP are the two protocols of the transport layer used in Internet network. TCP is a reliable connection-oriented protocol, which guarantees the sending of the packets from source to destination in the order they were sent. UDP is an unreliable, connectionless protocol so a best effort protocol which use no mechanism

to handle packet loss or managing out of order received packets. TCP is a congestion control protocol which controls the rate at which packets are transmitted between sender and receiver (slow down packet sending speed when detects that the network is overloaded). The purpose of UDP protocol is to send data as fast as possible. Many implementation of TCP are realized and for each of this implementations are done different researches and studies over their impact on network utilization. Based on this two protocols we will study the behaviour of TCP flows which shows aggressive approach in getting the bandwidth and UDP which transmit with a constant rate regardless of other traffics on the network. We have used NS2 simulator to evaluate the differences between TCP flows, also to study the impact of the background traffic. In this paper we will look through providing Network Random Input, the analysing of Initial Conditions, the Mathematical Model (used algorithms), the realizations of some Simulations, the Graphic Representation of Flow Comparison, the Calculation of the Confidence Interval and at the Conclusions from finished simulations. At the end of the paper are included the references that we have looked during our work.

## **2. NETWORK RANDOM INPUT**

The network model we have used is composed from 10 nodes, build based on Dumbo-Bell typology that consists of only one bottleneck link which is shared between some streams. In the network we have included two TCP flows, one UDP flow with CBR traffic and another UDP flow with VBR traffic where both UDP streams are in the background. To test the network behaviour so that it approaches the real network where the traffic changes over time we have included the elements: first, we have involved background traffic; second, we have included VBR traffic which changes over time. We also have used the exponential method of generating the traffic, through which we do not generate traffic all the time but there are periods of time when we generate traffic (burst time) and periods of time when we do not generate traffic (idle time); third, to get different results each time we simulate the network we use the random number generator method which provides us random input for each simulation we do.

## **3. INITIAL CONDITION**

Based on our network typology we distinguish four streams which share the bottleneck link between the nodes 4 and 5. The bottleneck link is 10Mbps full-duplex and with a queue size of 10 packets while the other links have a queue size of 100 packets.

- We create traffic between the nodes 0 and 6 (TCP Reno). For the TCP link we use the maximum size of 100 packets for the congestion window and we take 512 byte the packet size, as we have an Ethernet link and that's the packet size which can be transmitted in the network based on RFC 895 and it

is also the standard packet size used by UDP. We use the minimum timeout of 0.2 seconds and we set the first UDP flow's starting time at 5.0 second and the ending time at 90.0 second.

- We create FTP traffic between the nodes 3 and 9 (TCP Tahoe). We use the same parameters for this flow as the one above and in this way we evaluate the flows under the same conditions. This flow begin to transmit later than the first one, starts at 10.0 second and ends at 90.0 second.
- We create CBR traffic between the nodes 1 and 7. The CBR traffic generate packets at the speed of 100 packets/sec. The CBR traffic source starts at 15.0 second and ends at 60.0 second. CBR traffic do not transmit actual data but informs UDP agent that has data to transmit and the agent creates packets and sends the data. We use a simplex link as UDP does not have a congestion control mechanism.
- We create 1 Mbps VBR traffic between the nodes 2 and 8. We use the exponential on/off method to generate traffic and we set the On period to 150 ms and the Off period to 100 ms. The used link is simplex one. The packet size is the same as used in TCP and CBR. VBR traffic starts at 20.0 second and ends at 70.0 second.

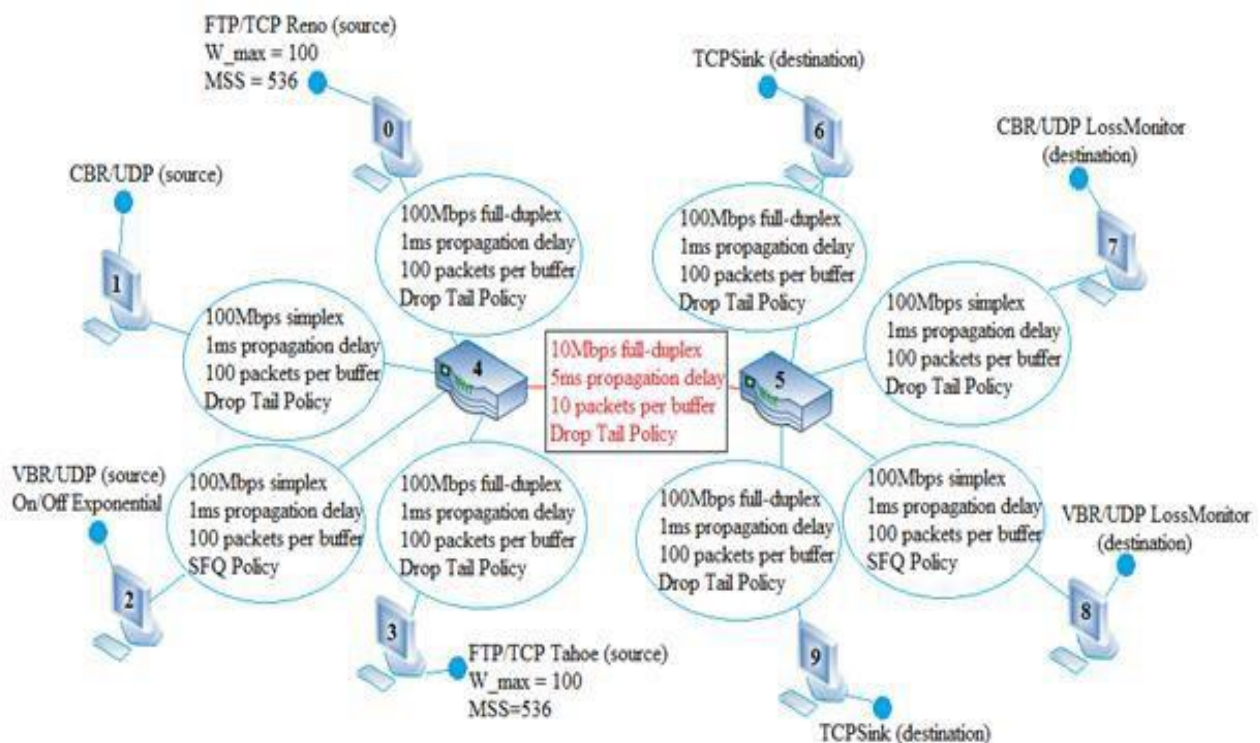


Fig. 1. Dumb-Bell typology with a bottleneck link

### 3.1. Network topology with 10 nodes

- The nodes 4 and 5 create the bottleneck link
- The nodes 0 and 3 generate FTP traffic (source nodes)
- The nodes 1 and 2 generate CBR and VBR traffic (source nodes)
- The nodes 6, 7, 8, 9 are destination nodes

We set the bandwidth to 10 Mbps full- duplex for the bottleneck link to simulate a 10BaseT Ethernet link. The queue size is 10 packets smaller than the queue size of other links to create in this way the bottleneck and the buffer should have an appropriate size as the congestion avoidance mechanism brings throughput oscillation. Over buffered routers increase latency when we have congestion, whereas under buffered routers bring more problems because packet forwarding generate throughput oscillation. For all the other network links we set the capacity to 100 Mbps full- duplex to simulate a 100BaseTX Ethernet link and we set the queue size to 100 packets. For most of the links we use the DropTail algorithm for the queue (FIFO) whereas for the n2-n4 and n5- n8 links we use the SFQ algorithm to provide various network conditions. In the simulation we will perform is used Dumb- Bell typology with 2 TCP sessions which share only a bottleneck link between two routers. Dumb- Bell typology with a congested link allows as to study the network traffic and it simulates a bottleneck link where TCP sessions are transmitted in the internet. Along with TCP streams we have UDP streams which are placed in the background and cause congestion and TCP packet loss.

#### 4. MATHEMATIC MODEL

UDP protocol do not implement any congestion avoidance mechanism contrary to TCP protocol which has congestion control mechanism. Standard TCP Tahoe is one of the earliest implementations which uses the go-back-n model to control network congestion.

TCP Tahoe is based on the algorithms:

- 1) Slow Start
- 2) Congestion Avoidance
- 3) Fast Retransmit

TCP Reno is similar to TCP Tahoe, but they differ because TCP Reno has an additional algorithm which is Fast Recovery. During slow start phase congestion window increases with one for each acknowledge (ACK) message received, so the congestion window grows exponentially. Slow start phase continues until the congestion window is equal to or greater than threshold. Once this condition is true begins the next phase which is congestion avoidance. During the congestion avoidance phase instead of increasing the window size with one each time we receive a ACK message, congestion avoidance algorithm increases the window size with one for each RTT (Round-Trip delay Time) so in this way the congestion window size has a linear growth. In the fast retransmit phase when we receive a certain number of duplicated ACK for the same packet, the sender retransmit the packet without waiting for the timer to expire. Fast recovery works during those cases when a certain number of duplicated ACK (threshold usually is set to 3) is received. Just as during fast retransmit, the sender retransmit lost packets but instead of slow start the congestion window is decreased by half and then it counts the duplicated ACK to determine when it should send packets again. The window used by the sender:

$\min(\text{awin}, \text{cwin} + \text{ndup})$

- **awin:** receiver window
- **cwin;** sender congestion window
- **ndup:** the number of duplicated ACK (stays at 0 until the number of duplicated ACK reaches threshold)

Referring RFC 2581 related to TCP Congestion Control we could explain the mathematical model based on mentioned algorithms. Specifications:

- **SEGMENT:** each TC/IP data or ACK packet
- **SENDER MAXIMUM SEGMENT SIZE (SMSS):** the maximum size of the segment that the sender can transmit
- **RECEIVER MAXIMUM SEGMENT SIZE (RMSS):** the maximum size of the segment that the receiver can accept
- **FULL-SIZED SEGMENT:** the segment which has the maximum number of data bytes that is allowed
- **RECEIVER WINDOW (rwnd):** the window that the receiver uses most
- **CONGESTION WINDOW (cwnd):** The state variables that limit the amount of data that a TCP sender can transmit. At certain time, a TCP sender may not send data with a greater sequential number than the sequential number confirmed and the minimum of cwnd and rwnd
- **INITIAL WINDOW (IW):** the size of cwnd after three-way-handshake has finished
- **LOSS WINDOW (LW):** the size of cwnd after the TCP sender detects loss of packets using retransmission timer
- **RESTART WINDOW (RW):** the size of cwnd after the TCP sender begins again the transmission after an idle time (if slow start algorithm is used)
- **FLIGHT SIZE:** the number of data which are send but not acknowledge yet.

The cwnd variable is a limit at the sender side, in the number of data the sender can transmit before getting the ACK. The rwnd variable is a limit at the receiver side, in the delayed number of data. Minimum size of cwnd and rwnd manage the transmission of the data Slow-start algorithm is used before the transmission or after the recovery of lost packets from retransmission timer, to inspect the network in determining the available bandwidth. IW, cwnd initial size, it should be smaller than or equal to  $2 * \text{SMSS}$  byte and it should not be greater than 2 segments.

The initial value of ssthresh (slow-start threshold) can be a random value and it can be reduced as a response to congestion. The slow-start algorithm is used when  $\text{cwnd} \leq \text{ssthresh}$ , and the congestion avoidance algorithm is used when  $\text{cwnd} > \text{ssthresh}$ . The sender can use each of them when cwnd is equal to ssthresh. During slow-start, TCP increments cwnd with at least SMSS byte for each received ACK that confirms the data send. During congestion avoidance, cwnd is incremented with 1 for each RTT. Slow-start ends when cwnd exceeds ssthresh or when congestion is detected. Congestion avoidance ends when congestion is detected.

$$\text{cwnd} += \text{SMSS} * \text{SMSS}/\text{cwnd} \quad (1)$$

When an TCP sender detects the segments loss using retransmission timer, the value of ssthresh should be set to:

$$\text{ssthresh} = \max(\text{FlightSize}/2, 2 * \text{SMSS}) \quad (2)$$

TCP sender should use fast retransmit algorithm to detect and repair the loss, based on ACK duplicated packets. After receiving 3 duplicated ACK, TCP performs a retransmission for the missing segment, without waiting for the retransmission timer to expire. Fast retransmit and fast recovery algorithms can be implemented together:

- 1) When the third duplicated ACK is received, ssthresh is set to the value given by equation (2)
- 2) The lost segment is retransmitted and cwnd is set to ssthresh + 3\*SMSS. This affects cwnd in three segments which are away from the network and are in the receiver buffer.
- 3) For each duplicated ACK received, cwnd increments from SMSS. This artificially increments cwnd.
- 4) A segment is transmitted if it is allowed from the new value of cwnd used by the receiver.
- 5) When another ACK is received to confirm new data, cwnd is set to ssthresh. This is called window deflation.

## 5. EXPERIMENTAL STAGE

To reach conclusions about how TCP streams compete with each other in utilization of bandwidth that is being shared also from traffic background we realize a total of 10 simulations. Given that the results obtained from each simulation differ, then for all the performed simulations we realize superposition of graphs to see if they fall almost in the same footprint and to confirm with certainty the received outcome. By all performed simulations we will evaluate how much and what proportion of the available bandwidth occupies each stream and then we will give an overall average of bandwidth utilization from each stream. Below we provide graphical presentations obtained by NS2 to show different results of realized simulations.

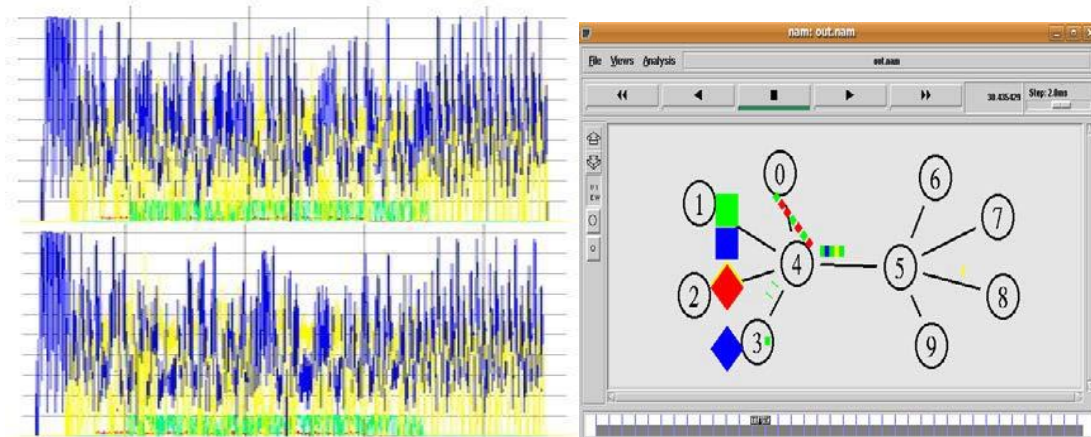


Fig. 2. Overview of NAM during the simulation and illustration of 2 different results taken from simulations

### 5.1. Representation of streams comparison

The purpose of the work consists in measuring and analysing the performance of TCP Reno and TCP Tahoe. Their study is done in the presence of background traffic. We will make the comparison of TCP streams by evaluating the bandwidth that they allocate (also considering other streams). Regarding the graphics taken from simulations we see that streams which compete aggressively among themselves to exploit as much bandwidth are TCP streams. TCP Reno stream provides more bandwidth than standard TCP Tahoe stream. Meanwhile in cases when in network appear all types of streams- TCP streams and those in the background as VBR or CBR, the bandwidth used by each TCP stream decreases. To give a clearer idea of graphical presentations we initially represent streams separately and then give a graphical representation of all streams that are on the network and compete for bandwidth, evaluating the bandwidth size that each stream occupies.

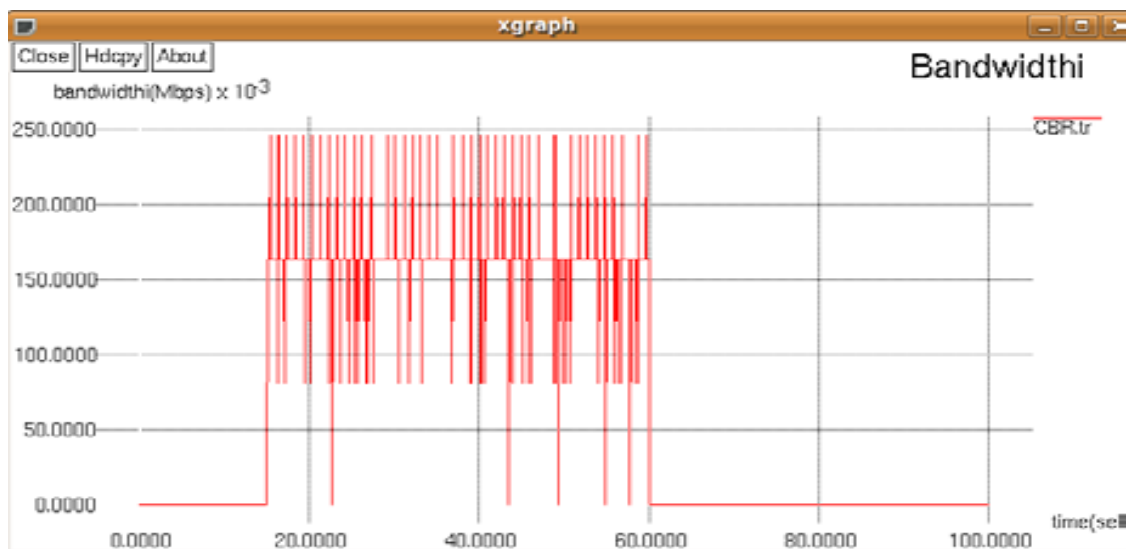


Fig. 3. Graphic presentation for the bandwidth used by CBR (in the presence of 4 streams on network)

So, as we see from the graph for CBR, the bandwidth that CBR use is very low compared with the size of the link. So, as we see from the graph for CBR, the bandwidth that CBR. The maximum value that can achieve CBR in the utilization of the link's bandwidth is 250 kbps and the minimum value is 90 kbps. The nominal bandwidth is 170 kbps, and this relates to the fact that CBR transmits packages with constant speed and does not show any aggressiveness in obtaining bandwidth. On the other hand this is related to the fact that UDP does not use any mechanism for congestion.

Compared with the case of CBR, VBR represents more fluctuations in bandwidth and this is related to the fact that VBR transmits the packages with different speed. The size of the bandwidth that VBR allocates reaches the maximum value at 1 Mbps. Unlike CBR where we used mechanism Drop Tail, in the case of VBR we use GFS mechanism.

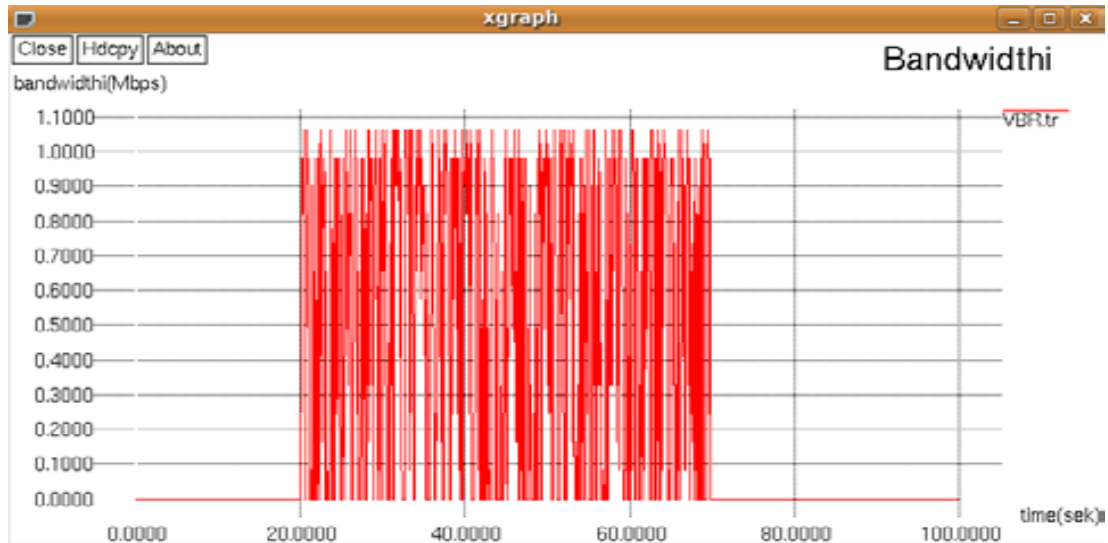


Fig. 4. Graphic presentation for the bandwidth used by VBR (in the presence of 4 streams on network)

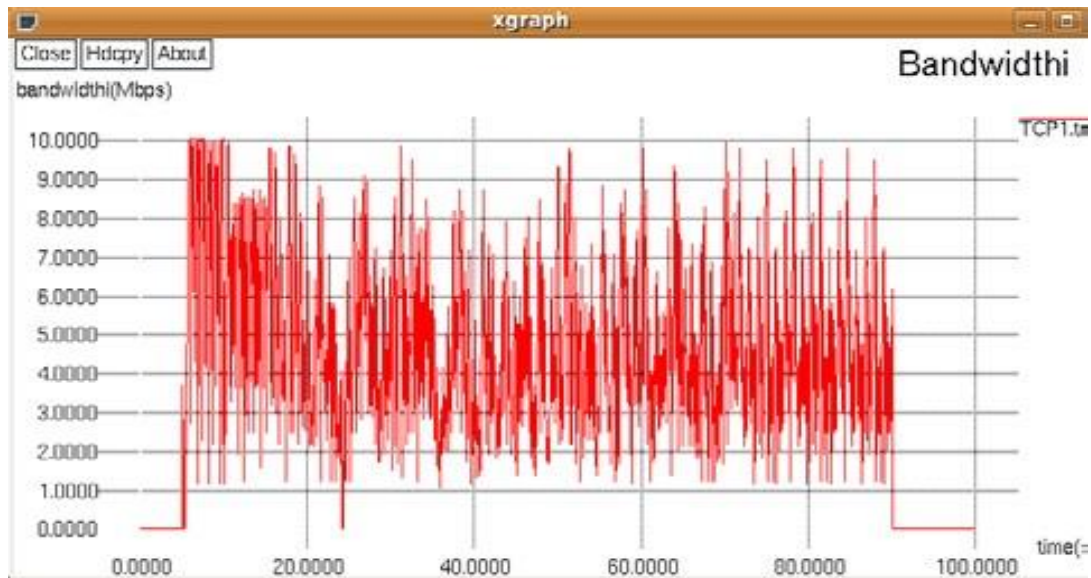


Fig. 5. Graphic presentation of bandwidth used by TCP Reno (in presence of background traffic)

From the graph obtained for the bandwidth used by TCP Reno we see that at the very beginning of the simulation when on the network none of the traffic sources had not started any transmission, TCP Reno uses all the potential capacity of 10Mbps bottleneck link. While the emergence of other traffics on network, the bandwidth used by TCP Reno decreases, as TCP Reno should share the link - although at certain moments when it can find space in the link tends to use it aggressively. It is noted that the average of the values connected with the bandwidth used by TCP Reno is roughly 4-5Mbps.



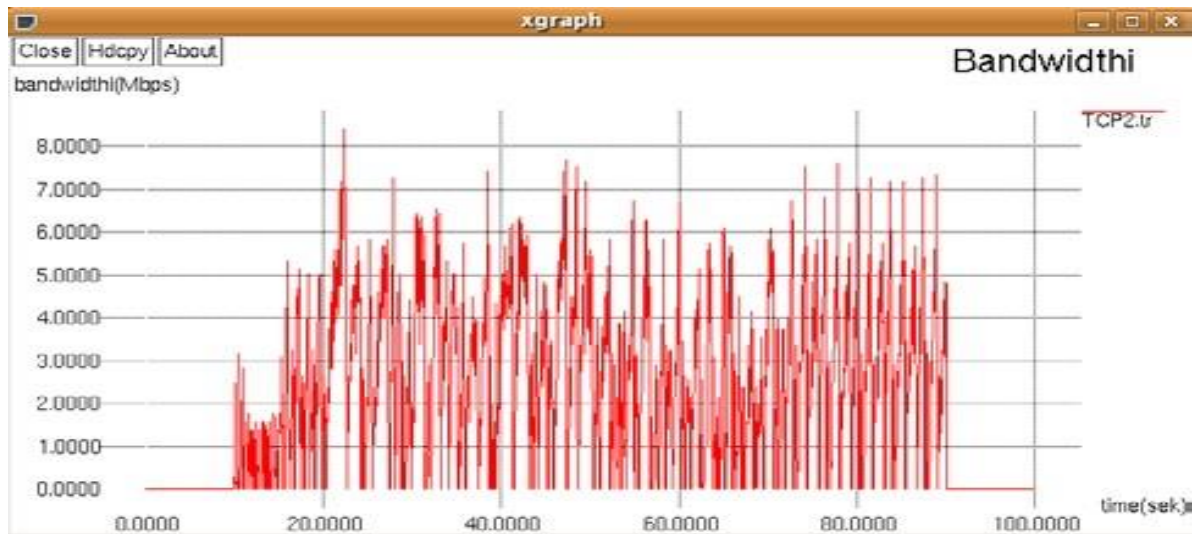


Fig. 6. Graphic presentation of bandwidth used by TCP Tahoe (in presence of background traffic)

Just as we affirm theoretically that TCP Reno was an improvement of TCP Tahoe and it should be expected to have a low utilization of bandwidth-it, from the graph given above we confirm such a thing. TCP Tahoe does not show the same aggressiveness as TCP Reno, therefore never fails to utilize the full capacity of maximum bottleneck link. The maximum value of bandwidth that TCP Tahoe can achieve to utilize is 8 Mbps. Meanwhile the approximate average value of the average bandwidth that is being used, it is 2-3Mbps resulting to be lower than the bandwidth used by TCP Reno. At the following charts we will provide graphical presentations of TCP streams in the presence of background traffics. What it is normally expected as a result is that in the presence of background traffics the used bandwidths by each TCP stream are expected to decrease, because the bottleneck link is shared by 4 streams. Normally TCP will remain aggressive in the link not only against the background traffic, but also against each other.

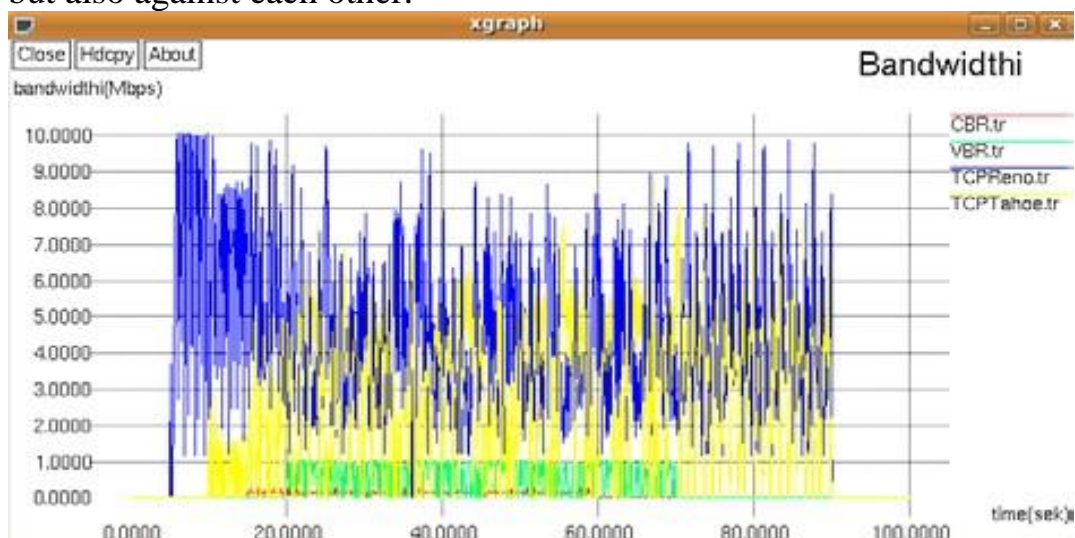


Fig. 7. Graphic presentation for sharing the bandwidth of bottleneck link among 4 streams

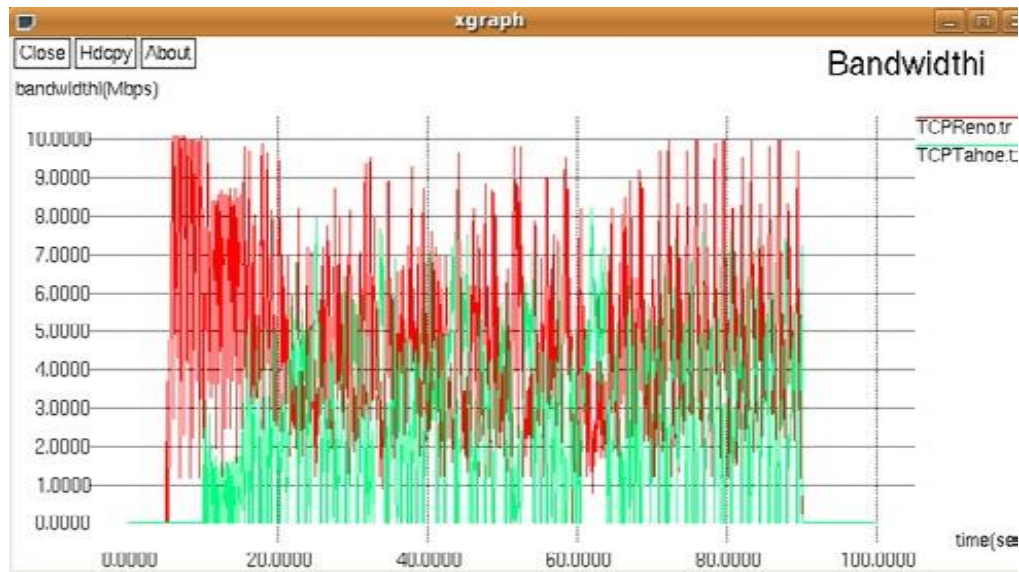


Fig. 8. Graphic presentation for sharing the bandwidth of bottleneck link among 2 TCP streams

In the graphical representation of figure 8 we note that in the presence of background traffic, the sharing of the bandwidth between streams starts to become fairer. This is due to the fact that with the increase of RTT, it also grows the delay of ranks. With the increase of RTT, we see that it is also decreased the aggressiveness in getting the bandwidth for TCP Reno stream, so TCP Tahoe has more opportunity to provide bandwidth even though the TCP Reno stream dominates on network.<sup>1</sup>

## 5.2. Calculation of the confidence interval

The results of simulation wouldn't had any value if we wouldn't have any idea about their accuracy, knowing that from every simulation it is received a different result. To evaluate the data from every simulator, we must built the interval of confidence. To calculate the interval of confidence we proceed as shown below:

- We simulate and obtain an evaluation X1 for the measurement that we are interested
- We repeat the simulation M times and get X2...XM, all different from each other.
- We evaluate the average of M champions

---

<sup>1</sup> (Reminder: In telecommunications, the round-trip delay time (RTD) or **round-trip time (RTT)** is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received. This time delay therefore consists of the propagation times between the two points of a signal).

$$\bar{X} = \frac{\sum_{j=1}^M X_j}{M} \quad \sigma_X^2 = \frac{1}{M-1} \sum_{m=1}^M (X_m - \bar{X})^2 \quad (1)$$

Knowing that we cannot find a right evaluation with only one value, we consider an interval [c1, c2]. Based at the theorem of certified limits, we assume that 95% of the values are in the same interval. For Gaussian distribution, the interval of confidence evaluates with a derivation  $\pm 1.96$ . The interval of confidence with a 95% For M=10 we get  $\bar{X} = 2.17$  and  $\sigma_X^2 = 2$

$$\bar{X} \pm \frac{1.96\sigma_X}{\sqrt{M}} \quad (2)$$

Then we will calculate the interval of confidence for the bandwidth in the CBR stream. For M =10 we get  $\bar{X} = 0.073$  and  $\sigma_X = 0.00005$ . The interval of confidence is [0.069; 0.075] calculate the Interval of confidence for the bandwidth for VBR stream. For M=10 we get  $\bar{X} = 0.29$  and  $\sigma_X^2 = 0.0004$ . The interval of confidence: [0.278; 0.302]

- We've calculating the interval of confidence for the bandwidth in the TCP Reno stream: For M=10 we get  $\bar{X} = 4.14$  and  $\sigma_X^2 = 2$

The interval of confidence: [2.081; 2.259] (as we stated above from the graphical appearances, the majority of the values are between 2-3 Mbps)

## 6. CONCLUSIONS

In this project we treated how TCP streams (TCP Reno and TCP Tahoe) compete with each other in allocation of the bandwidth in a bottleneck link and also how they compete with background traffics (CBR and VBR). Traffic in the background ensures more approaching in a real network and it is noticed that with his presence on the network, TCP streams become more fair in relation to each other by reducing aggression. What was observed in the graphic presentations was that TCP Reno dominated in its transmission on network compared to standard TCP Tahoe. This refers to the fact that TCP Reno is an improvement of the standard TCP Tahoe because besides the algorithms that it uses - slow start, congestion avoidance, fast retransmit, it also uses and fast recovery algorithm. Fast Recovery acts in those cases when a certain number of duplicate ACK-s is taken (threshold generally is set at 3). The sender retransmits the lost packets, but rather the slow start the congestion window is halved and then counts the duplicate ACK-s for defining when to send packages. The interval of confidence: [4.053 ; 4.227] (as we stated above from the graphical appearance, the majority of the values are between 4-5 Mbps). We calculate the interval of confidence for the bandwidth for the TCP Tahoe stream: Starting from the graphical presentations we affirm that TCP Reno uses on the average 62% of the link bandwidth while TCP

Tahoe 32%. The rest of the link is shared between the CBR and VBR that use on the average 1% and 5% of the available bandwidth of the bottleneck link.

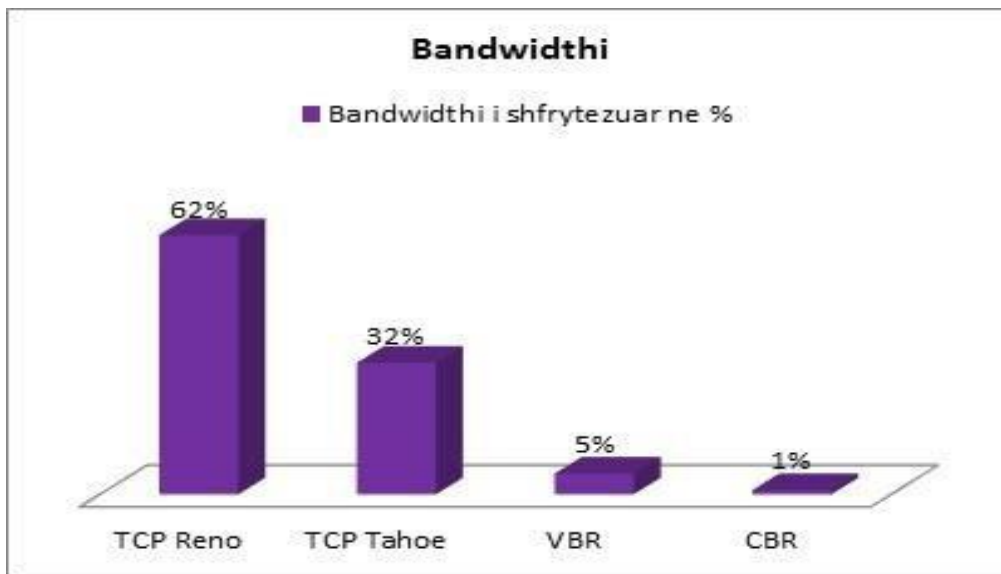


Fig. 9. The average bandwidth used by each of the streams in the bottleneck link

TCP Reno performs better when the losses of packages are small. In cases where losses are great in a window, then Reno is not performing well and its performance is almost the same as TCP Tahoe. Another problem is that if the window is very small when losses occur, then we will never get duplicate ACK for fast retransmit and we must wait for timeouts. This method does not detect effectively the losses of packets.

## REFERENCES

- [1] Teerawat Issariyakul, Ekram Hossain (2011), "Introduction to Network Simulator NS2"
- [2] Jeonghoon Mo, Richard J.La, Venkat Amatharam, Jean Walrand (1998) "Analysis and Comparison of TCP Reno and TCP Vegas", Department of Electrical Engineering and Computer Sciences, University of California Berkeley
- [3] Navraj Chohan (2012) "An Analysis of TCP through Simulation",
- [4] Yuvaraju B. N, Niranjana N Chiplunkar (2015) "Scenario Based Performance Analysis of Variants of TCP using NS2-Simulator", Department of Computer Science and Engineering NMAM Institute of Technology, Nitte - 574110, Karnataka, INDIA"
- [5] Sangtae Ha, Yusung Kim, Long Le, Injong Rhee (2014) "A Step toward Realistic Performance Evaluation of High-Speed TCP Variants", Department of Computer Science, North Carolina State University, Lisong Xu, Department of Computer Science and Engineering,
- [6] Soohyun Cho, Texas A&M University, (2006) "Congestion Control Schemes For Single and Parallel TCP Flows in High Bandwidth-Delay Product Networks".