Performance Comparison of different Congestion Control Protocols in Hybrid Network

Mehta Ishani
PG Student
Gardi Vidyapith
Rajkot

Udit Narayan Kar
Research Scholar
Saurastra University
Rajkot

Atul Gonsai, PhD
Associate Professor
Saurastra University
Rajkot

ABSTRACT
Widely implemented topology in Internet is Heterogeneous networks which are a combination of wired and wireless networks. Due to high bandwidth availability at wired links and high delayed products at wireless links, bottleneck situation is generated at Access Point and it has to face severe Congestion consequences. Some of the most known and recent protocols developed to provide faster and lighter congestion control are TCP (Transmission Control Protocol) and the Rate Control Protocol (RCP). This paper provides performance comparison of TCP, RCP+ and our proposed algorithm RCP++ in hybrid network.

General Terms
TCP- Transmission Control Protocol, RCP – Rate Control Protocol,

Keywords
TCP, RCP, RCP+, RCP++, Congestion. Hybrid

1. INTRODUCTION
TCP is the most widely used congestion control protocol in the today's Internet scenario [1]. However, TCP has limitations such as not providing good bandwidth utilizations especially in high bandwidth-delay product networks, introducing high network load and excessive overhead over the network, whereas, RCP is not flexible enough to co-exist with dynamic hybrid networking scenario where multiple protocols co-exist [8]. RCP involves complex computations at routers and have risk of buffer overflow [5, 6, and 7]. In this experiment we have analyzed the performance of TCP, RCP+ and our proposed model RCP++. This experiment aims at studying the effects of hybrid (wired and wireless) networks on TCP, RCP+ and RCP++ by the researchers of this experiment in presence of heterogeneous network configurations. The simulation results obtained of these protocols in heterogeneous scenario; show that RCP++ is able to significantly increase the efficiency of congestion control mechanisms. This experiment covers maximum of the performance parameters like Throughput, Dropped Packet, End to End Delay, Packet Delivery Ratio and Network Routing Load. The study is made in dynamic networking scenario so that bandwidth capacity evaluation is near to practical approach in hybrid networks.

2. LITERATURE SURVEY
2.1 TCP
Congestion control mechanism given in [1, 5] is shown in Fig 1. Initial congestion window size for each TCP connection is 4 Maximum Segment Size (MSS). It uses slow start procedure to acquire the available bandwidth. The Slow-Start procedure keep a new sender from overflowing network buffers, while at the same time increase the congestion window fast enough and avoid performance loss while the connection is operating with a small window. Slow-Start additively increases the congestion window by one MSS for each new acknowledgment received, which results in the window doubling after each window's worth of data is acknowledged.

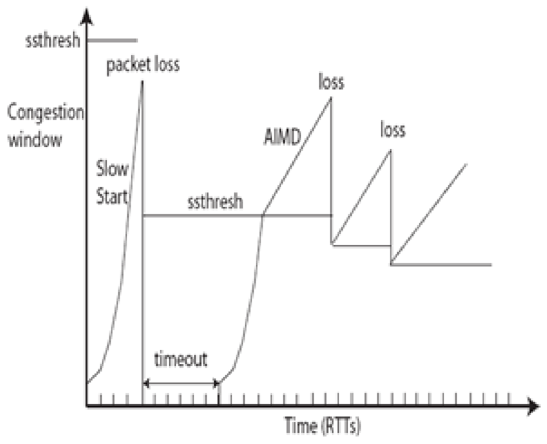


Figure 1: TCP's congestion control mechanism [5]

A connection enters Slow-Start at initial or on experiencing a packet retransmission timeout, and exits Slow-Start when it detects a packet loss or when the congestion window has reached a dynamically computed threshold, ssthresh. More specifically, ssthresh is set to half of the current congestion window when packet loss was detected. TCP exits Slow-Start and enter the Congestion Avoidance phase, where it continues to probe for available bandwidth, but more cautiously than in Slow-Start. During periods when no packet losses are observed, TCP performs an Additive Increase of the window size, by 1 MSS on receiving each acknowledgment packet. The equation can be given as:

$$Cwnd = cwnd + 1 \quad (1)$$

And when Congestion occurs it decreases the window size by half as given in equation below:

$$Cwnd = Cwnd * \frac{1}{2} \quad (2)$$

Thus this way TCP's additive increase multiplicative decrease algorithm works [1].

2.2 RCP

Researchers [5, 6, 7, and 11] have proposed Rate Control Protocol (RCP). In RCP as the name suggest, a router assigns a single rate, R (t), for flow transmission. The basic idea is: If there is spare capacity available, then share it equally among all flows. If the link capacity is insufficient then a queue started building up, then and the flow rate is decreased evenly. The solution is, tried to minimize Flow completion time (FCT). FCT is time from when the first packet of a flow is sent (in TCP, this is the SYN packet) until the last packet is received [6]. To minimize FCT for each router a well-known method is to use processor-sharing (PS) i.e. a router divides outgoing link bandwidth equally among ongoing flows [7]. An equation to calculate rate is given below:

$$R\,(t) = R\,(t - d0) + \frac{\left[\alpha(C - y(t)) - \beta\frac{q(t)}{d0}\right]}{N(t)} \qquad (3)$$

Where,
d0 = a moving average of the RTT measured across all flows,
R (t−d0) = last updated rate,
C = link capacity,
y(t) = measured input traffic rate during the last update interval (d0 in this case),
q(t) = instantaneous queue size,
N(t) = router’s estimate of the number of ongoing flows (i.e., number of flows actively sending traffic) at time t
α, β = parameters chosen for stability and performance.

Simulation results show that RCP performs better than TCP in terms of FCT and link utilization in wired network [9]. It is also taken account that RCP possess routing overhead on network also due to having complex computation and involving routers. Researchers of paper [8] have introduced two limitations of RCP: (1) RCP will need to operate alongside existing non-RCP traffic, such as TCP and UDP, without adversely affecting or being affected by the other traffic; and (2) RCP will need to operate in a network where some routers are not RCP-enabled.

2.3 RCP+

Authors [2, 3, and 4] described a simple congestion control algorithm called RCP+ which reduces flow completion time for diverse flow types to large extend for broad range of traffic conditions and network situations. In RCP+, unlike XCP, RCP they are not following feedback mechanism but they are adapting the ancient congestion window based congestion control mechanism. The reason behind sticking to the congestion window for congestion control is that, while each second new flow are entering and moving out of network, it is tough to obtain exact number of flows at particular RTT. This inspired us to stay with congestion window based mechanism instead of feedback based mechanism [3]. The equation of RCP+ can be given as:

$$N(t) * R(t) = (\alpha * C - \alpha * y(t) – (\beta * q(t)/d)) \qquad (4)$$

Where, d is moving average of RTT per interval, R(t) is last updated rate, y(t) is existing traffic observed in network, q(t) is the instantaneous queue size, C is link capacity and N(t) is number of flows. α (alpha) is a stability parameter and β (beta) is a performance parameter to make the rate stable and not aggressive. Thus, the equation gives us the desired aggregate rate change in presence of traffic in the next interval. Here, Rate is kept same for each flow. “Cwnd” congestion window value is having proportionate relationship with the rate and so, “cwnd” value is set on the bases of the Rate computed. The fundamental mechanism of congestion control in TCP is slow start and congestion avoidance. Based on that, authors [2] have developed mechanism that supports slow start and congestion avoidance. Connection generally initiates with congestion window set to MSS. Now once, each segment is acknowledged successfully, the congestion window is increased by one MSS.

$$Cwnd = cwnd + MSS \qquad (5)$$

Now, what is the maximum number of segments sent by RCP+ in one RTT?

$$Cwnd = cwnd + MSS * (cwnd/MSS) \qquad (6)$$

$$\text{I.e. } cwnd = 2*cwnd \qquad (7)$$

That means, for every RTT, there the congestion window increments by doubling the previous cwnd value. The congestion window keeps incrementing in multiple of two until it reaches to its maximum threshold, i.e. ssthresh. This threshold value is maintained based on window advertisement and updated based on response to congestion. RCP+ also uses congestion avoidance. How?

$$Cwnd = cwnd + MSS * (MSS/cwnd) \qquad (8)$$

How does RCP+ perform loss detection and recovery functionality? All the packets that are not acknowledged are to be retransmitted by the RCP+ once the timer times out. This timeout event is also known as retransmitting time out (RTO). RCP+ considers that all the packet loss is due to congestion and no other reason. Because of this, TCP sender reduces its transmission rate with guilt that reason is himself. Means, once the retransmission time out occurs; TCP sender resets its ssthresh value according to the equation:

$$Ssthresh = Max\ (cwnd / 2,\ 2 * MSS) \qquad (9)$$

After retransmitting the lost packet, RCP+ sender used slow start and goes for rate calculation again based on equation (4) of RCP+ and the calculated rate is given to initial packet so that initial window is provided to the flow is better than the one applied in TCP. When the receiver gets the packet that is not the original but the transmitted, it is obviously not ordered. In this situation, receiver provided with the ACK that identifies the first octet of missing data. After 3 retransmission, and receiving 3 duplicate acknowledgements, without waiting for next timer to timeout, follows the mechanism of Fast Retransmission

3. PROPOSED ALGORITHM – RCP++

Our proposed approach RCP++ is based Improved AIMD and RCP+ algorithm. In our proposed approach we use congestion window mechanism of Improved AIMD algorithm to use the spare capacity of congestion window after occurrence of congestion event.

The limitation of improved AIMD is that the algorithm does not include the different arrival time of flows [1]. So we have used modified equation of RCP+ algorithm.

RCP+ algorithm is implemented based on the theory of RCP. RCP+ is having the added advantage of coexistence with other wired and wireless TCP, XCP, RCP and DCCP protocols. RCP+ is flexible like TCP and so is expected to have wide implementation over current demands of Internet. [2]

In the theory of RCP there is a concept of queue. RCP was originally implemented in wired network [5]. In wireless network the essential point is traffic is in bursty nature. Hence it is difficult to use queue concept while going to practical simulation. So we modified the equation of rate change.

Here we come up with a new proposal of congestion control scheme enhanced RCP. Here we use Improved AIMD mechanism as well as modified rate change equation of RCP+. Our aim is to gain benefits of both schemes by eliminating each other's demerits.

Initially we defined congestion window size 4 MSS and start data transfer. If acknowledgment is received means no congestion and increase congestion windows size by +1 and transfer data

If acknowledgment is not received then congestion would likely to be occurred then decrease congestion window size by equation:

$$\text{Cwnd} = (\text{cwnd} * 1/2) + K \qquad (10)$$

Where K is the increment in congestion window size (w) in t cycles or epoch. Now calculate the rate by equation:

$$\text{Rate} = (\alpha * C - \alpha * y(t))/N(t) \qquad (11)$$

Again checks if rate is greater than congestion window size then increase congestion window size otherwise transfer data.

Below figure shows the flow chart for the same.

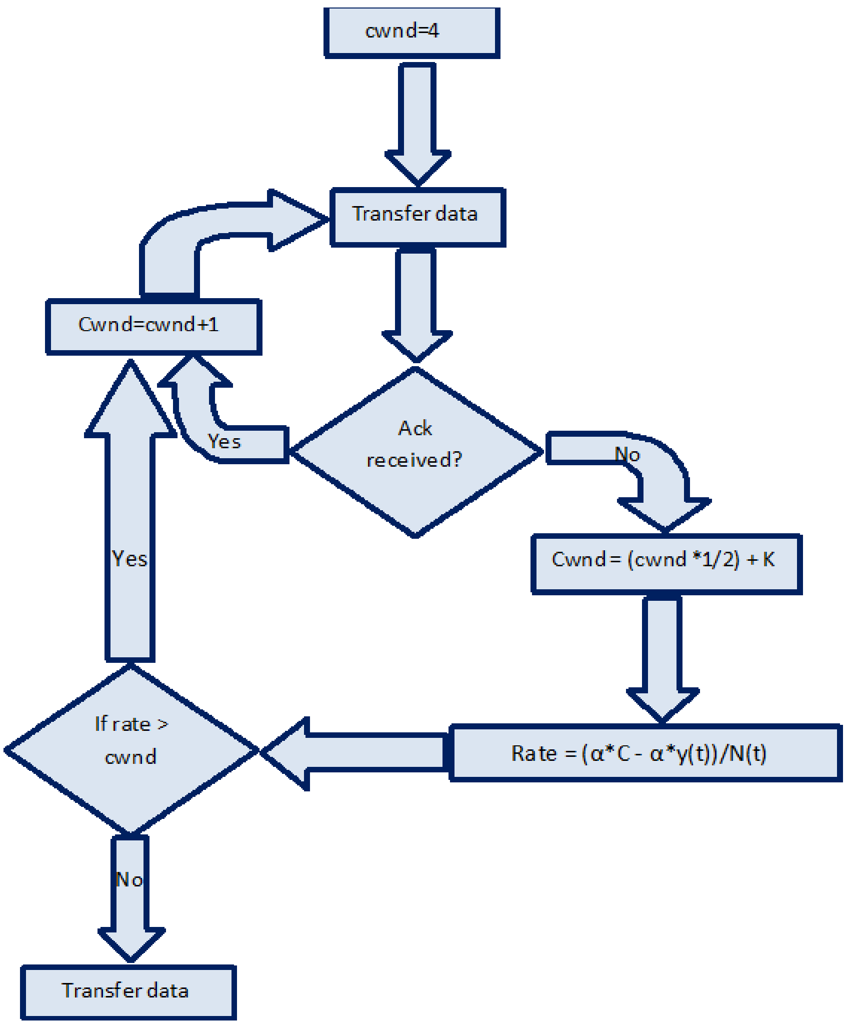


Figure 2 Flow chart for RCP++

Algorithm

1. Initially set congestion window size 4 MSS
2. Data transfer
3. If acknowledgement is received increase congestion window size by cwnd = cwnd +1 and repeat step 2
4. Otherwise decrease congestion window size by equation Cwnd = (cwnd *1/2) + K
5. Calculate the rate by equation
   Rate = (α*C - α*y(t))/N(t)
6. If rate > cwnd then go to step 2
7. Otherwise transfer data

4. IMPLEMENTATION DETAILS

We have implemented RCP++ in NS-2 simulator and version is 2.35.

4.1 Implementation Code

Following are the steps for modification we have done in C++ file of TCP to implement RCP++

Step 1: Go to the location where tcp.cc is located. There, under the class definition of TCP agent, bind two variables for its usability under the procedure of bind.

```
Bind ("bw_",& bw_);
bind("flow_",&flow_);
```

Step 2: Modify TcpAgent::window procedure

```
If (frto_ == 2)
{
Return (force_wnd(2) < wnd_?
  force_wnd(2) : (int)wnd_+4);
}
```

Step 3: Modify TcpAgent::slowdown procedure

```
Double win, halfwin, decreasewin, k;
k = (windowd() / 2) -1;
If (cwnd_ < ssthresh_)
             slowstart = 1;
     If (precision_reduce_)
             {
             halfwin = (windowd() / 2)+k;
             }
             Else
             {
             Int temp;
             temp = (int)((window() / 2)+k);
             halfwin = (double) temp;
            }
```

Step 4: Modify TcpAgent::processQuickStart procedure

```
Define following variables
Int app_rate, bw, flow;
Float alpha,  yt;
```

Step 5: Add following lines to the procedure of processQuickStart

```
qs_requested_ = 0;
qs_approved_ = 0;

If (qsh->flag() == QS_RESPONSE && qsh->ttl() == ttl_diff_
&& qsh->rate() > 0)
{
app_rate = (int) ((alpha * TcpAgent::bw_ - alpha * yt) /
(TcpAgent::flow));
}
```

Step 6: Give the computed rate to qs_cwnd defined under the procedure of processQuickStart

```
If (app_rate > initial_window())
{
qs_cwnd_ = app_rate;
qs_approved_ = 1;
}
Else
{ // Quick Start rejected
}
```

Step 7: Once the changes in tcp.cc are made, save and exit the editor. Next step is to open the terminal and under super user, type following commands one by one.

```
./configure
Make clean
Make
Make Install
```

Once you have done this your TCP agent will become RCP enabled now run the TCL scripts and obtained the results.

4.2 Hybrid Scenario

We have created a scenario having 2 nodes with wired connection and connected them to wireless domain having 3 nodes via a base station as shown in figure 3

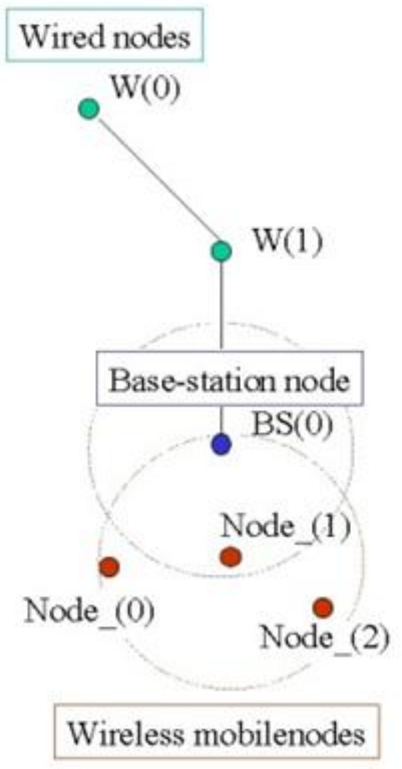


Figure 3 Hybrid Scenario

Following table 1 present the configuration details of the simulation.

Table 1. Configuration for RCP++

Layer	Parameter	Values
Application	FTP	FTP over TCP agent
Configuration	No of nodes	2 wired and 3 wireless
Mobility	Simulation time	250 s
Traffic	Type	TCP/CBR
Routing	Protocol	DSDV
MAC	MAC	802_11
PHY	Propagation model	Two ray ground
	Antenna	Omni
System	OS	Ubuntu 12.04
	Processor	Intel (R) core(TM) i5

4.3 Results and Graphs

To measure the actual strength of the algorithm, different parameters considered here are Throughput, Packet Delivery Ratio and Delay. These are the key parameters that would help to evaluate TCP, RCP+ and RCP++ [12].

Throughput: It is the rate of successful message delivery over a communication channel. Throughput is measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot.

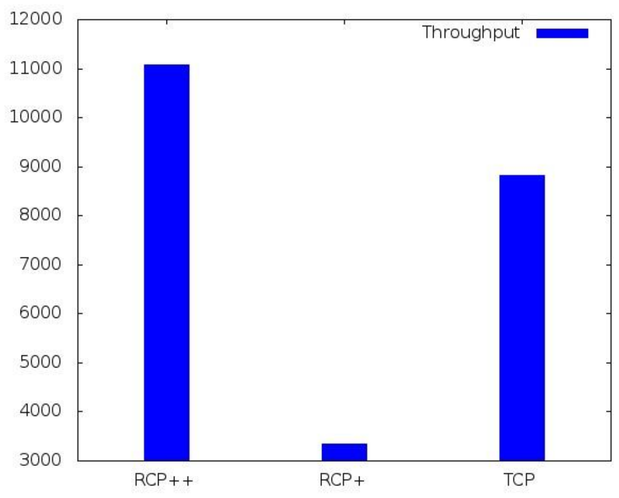
Packet Delivery Ratio: It is ratio of the number of delivered data packet to the destination. ∑ Number of packet receive / ∑ Number of packet send.

End-to-end Delay: It is the average time taken by a data packet to arrive in the destination. It also includes the delay caused by route discovery process and the queue in data packet transmission. Only the data packets that successfully delivered to destinations that counted. ∑ (arrive time – send time) / ∑ Number of connections. The lower value of end to end delay means the better performance of the protocol.

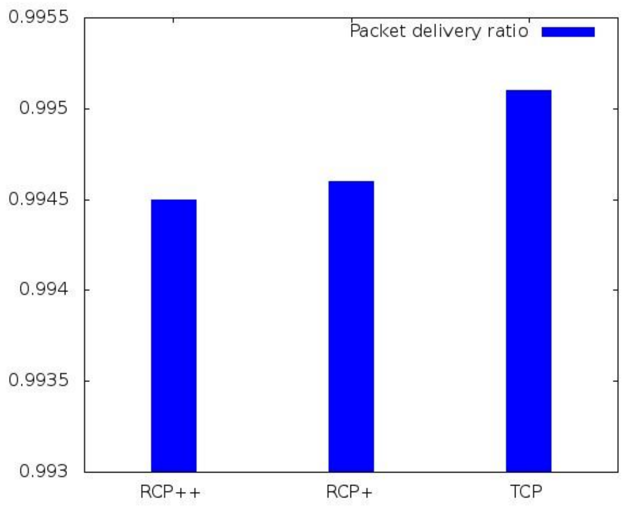
The values and graphs of packet delivery ratio, throughput and average delay for RCP++ RCP+ and TCP are given in following figures

Table 2 Statistics for RCP++, RCP+ and TCP

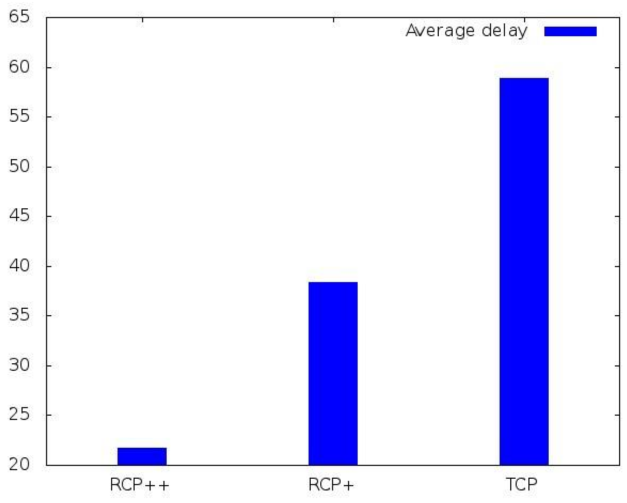
Protocol	Packet delivery ratio	Throughput	Delay
RCP++	0.9945	11075.05	21.6781
RCP+	0.9946	3348.71	38.3672
TCP	0.9951	8831.66	58.9255



Graph 1 Throughput graph



Graph 2 Packet delivery ratio graph



Graph 3 End to end delay graph

Again RCP++ wins in hybrid scenario also

5. CONCLUSION

Newly proposed protocol of RCP++ is tested on wide range of situations but, all upon simulation environment. The only limitation of this protocol observed on simulation environment is that it works for smaller no of nodes. If the number of nodes increases, its performance degrades to great extent. Thus, to make it work for more no of nodes, further improvements are necessary. Again, to increase the acceptance of the novel approach of RCP++, most of the simulation results must match with real time implementations. For this, we must test it on emulation or real time situations. Without testing the protocol on live environment by implementing the code of RCP++ on real routers, popularity and acceptance of the protocol would remain limited.

If deployed on live networks and in the real routers, RCP++ is likely to have a tremendous impact on applications including large number of users browsing web through 802.11 standards, long file transfers like videos, gaming and at the end improving users' experience. Now, when the impact is so huge, why isn't it then already widely prevalent in today's network? The biggest issue is implementation in real networks. Today's corporate scenario does not allow new network technologies to be easily integrated and instantly implemented especially in live networks. The biggest challenge is not the algorithm testing but the deployment of algorithm over the real network routers and end host, which still remains as an open area for the researchers.

6. ACKNOWLEDGMENTS

We would like to thank University of Grant Commission (UGC) for providing us the infrastructure on which we implemented our experimental study.

7. REFERENCES

[1] S. Floyd, "High-speed TCP for Large Congestion Windows," RFC 3649, http://www.icir.org/floyd/hstcp.html, December 2003.

[2] Dr. Atul Gonsai, Bhargavi Goswami & Uditnarayan Kar, "*Newly developed Algorithm RCP+ for Congestion Control on large scale Wireless Networks*", *International Journal of Innovations & Advancement in Computer Science IJIACS* ISSN 2347 – 8616 *Volume 3, Issue 2* April 2014

[3] Dr. Atul Gonsai, Bhargavi Goswami, Uditnarayan Kar, "*Design of Congestion Control Protocol for Wireless Networks with Small Flow Completion Time*" Proceedings on National Conference on Emerging Trends in Information & Communication Technology. 2013

[4] Dr. Atul Gonsai, Bhargavi Goswami, Uditnarayan Kar, "*Experimental Based Performance Testing of Different TCP Protocol Variants in comparison of RCP+ over Hybrid Network Scenario*" International Journal of Innovations & Advancement in Computer Science IJIACS ISSN 2347 – 8616 Volume 3, Issue 2 April 2014

[5] Nandita Dukkipati, "*Rate Control Protocol (RCP): Congestion Control to make Flows Complete Quickly*", Ph.D. Thesis, *Stanford University*, Stanford, California. 2007

[6] Nandita Dukkipati & Nick McKeown, "*Why Flow-Completion Time is the Right Metric for Congestion Control and why this means we need New Algorithms", ACM SIGCOMM Computer Communication Review, Volume 36,* 2006

[7] Nandita Dukkipati & Nick McKeown, "*Processor Sharing Flows in the Internet", Stanford HPNG Technical Report,* Stanford University, Stanford, California. 2007.

[8] Chia-Hui Tai, Jiang Zhu, Nandita Dukkipati, *"Making large scale deployment of RCP practical for real networks" IEEE INFOCOM,* 2008

[9] Mehta Ishani, Uditnarayan Kar, Dr. Atul Gonsai, "*Why RCP is better than TCP" International Journal for Scientific Research & Development (IJSRD), Vol. 2, Issue 12,* 2015

[10] Book "*Introduction to Network Simulator NS2*" springer, 2009

[11] Rate Control Protocol (RCP) Home Page, URL: *http://yuba.stanford.edu/rcp*

[12] Throughput, Packet delivery ratio, End to end delay, *http://harrismare.net/2011/07/14/packet-delivery-ratio-packet-lost-end-to-end-delay/*