What if the end systems knew the bandwidth available in the network?

Paulo Loureiro[1], Edmundo Monteiro[2]

[1] Polytechnic Institute of Leiria, Leiria, Portugal, e-mail: loureiro@estg.ipleiria.pt
[2]University of Coimbra, Coimbra, Portugal, e-mail: edmundo@dei.uc.pt

In this paper we propose Open Box Transport Protocol (OBP), a mechanism that provides information to the end systems about the current state of the network path. With this information (essentially the available bandwidth and the most restricted link capacity) the end systems can efficiently use the network resources. OBP is easy to implement because the intermediate nodes only have to provide information about the current network state to the end systems, and don't have to estimate the transmission rate of the flows. The coexistence of flows from OBP and other transport protocols in the same link does not introduce additional problems because the transmission rate of OBP flows is estimated at the sender end system.

In this paper we present the new algorithm to estimate the transmission rate of each flow at the end systems. Moreover, we present the evaluation results in fat networks based on NS2 simulations. The results show that OBP can outperform current transport protocols and achieve results as good as XCP (Explicit Congestion Notification Protocol), using a simpler model.

***Index Terms*—Network Protocols, Transport Protocols, Congestion Control, Active Queue Management, Routing and Forwarding.**

I. Introduction

Beyond the traditional applications, the Internet is being used by a set of new applications, for example the multimedia applications VoIP and IPTV. These kinds of applications have changed the characteristics of the Internet traffic. Therefore, the congestion control actions must be adjusted because the new types of traffic have high requirements in terms of bandwidth and delay. Besides that, wireless networks are in expansion. For this kind of networks, the congestion control decisions may not be the best if the transport protocols only use the detection of packets losses as the criterion to identify the congestion inside the network. In wireless networks the corruption in packets is more frequent than in wired networks. Moreover, the current Internet capacity is continuously increasing.

The Additive-Increase-Multiplicative-Decrease (AIMD) [1] congestion control algorithm used by TCP [2, 3] shows poor performance in high Bandwidth-Delay Product (BDP) networks. If we intend to efficiently use the network capacity an alternative to TCP has become important.

Explicit Control Protocol (XCP) [4] is a congestion control solution based on intermediate nodes collaboration. In XCP, the intermediate nodes have to estimate a fair rate, to be used by the flows, and send this rate back to the sender end systems. Thus, this scheme implies great processing in the intermediate nodes, and the coexistence of the XCP packets and packets from other transport protocols needs additional queue management schemes.

Open Box Transport Protocol (OBP) [5] is a congestion control algorithm based on collaboration among routers and end systems. In contrast to XCP, OBP estimates the transmission rate at the sender end systems. This solution implies a small processing at the intermediate nodes and the output interfaces queue can simultaneously support packets from other transport protocols.

Digital Object Identifier inserted by IEEE

Using NS2 [6] simulations, we show that the performance of OBP is similar to XCP, achieving high utilization, negligible packet loss rate and low persistent queue length. The remainder of this paper is organized as follows: Section 2 provides the related work on congestion control mechanisms based on routers collaboration; Section 3 summarizes the characteristics and the design of the OBP scheme; results are presented in Section 4; Section 5 presents the conclusions and some directions for future work.

II. Background and Related Work

Over the past few years, several solutions have been proposed to give TCP [1, 2, 3, 7, 8, 9, 10] better and more network feedback, beyond packet loss information and propagation delay variation. In addition, the research community has been specifying alternative solutions to the TCP architecture. As OBP, some of these solutions are classified in the category of "modification of the network infrastructure", and are briefly explained as follows.

Explicit Control Protocol (XCP) [4] is a congestion control approach, which outperforms TCP in the traditional network environments, and is efficient, fair, scalable, and stable in the high BDP networks. This traffic control protocol generalizes the use of the Explicit Congestion Notification proposal (ECN) [13].

Variable-structure congestion Control Protocol (VCP) [11] is like a "two bit" version of XCP. This solution has lower performance than XCP [11].

QuickStart [12] is a TCP congestion control extension. With Quick-Start, the initial congestion window can be potentially large, avoiding the time-consuming slow-start.

Explicit Congestion Notification (ECN) [13] provides notification mechanisms, used by the intermediate nodes to notify the end systems about imminent network congestion. The benefit of this solution is the reduction of the delay and the minimization of the packets loss.

The OBP model assumes that congestion control decisions are made at the end systems. The intermediate nodes, along

the network path, only have to provide the information about the network state to the end systems. OBP is computationally simpler than XCP, since the intermediate nodes do not have to make decisions about congestion control and only have to inform the end systems about the current network state. Moreover, OBP is more flexible than other solutions because the intelligence of the model is at the end systems. So, future adaptations of the OBP algorithm will be easier.

III. Open Box Transport Protocol

In this section we briefly describe OBP developed in our previous work [5] and present an improved version of the algorithm to estimate the *transmission rate* (*TR*), based on the current network state.

The OBP algorithm enables the end systems to define their *TR* based on a set of variables, which represent the state of the network path, between two end systems. The end systems calculate the *TR* with the objective of changing the current network state to a new state. This new state must avoid congestion and, at the same time, maximize the use of the network capacity.

A network path is a set of links and routers, interconnecting two end systems. Each link has a maximum capacity, an available capacity, a propagation delay and waiting queues. OBP defines the state of the network path by two variables: *narrow link* – the smaller link capacity amoung all the links in a network path; and *tight link* – the smaller available bandwidth among all the links of the network path. These two variables should be in each packet, in IP header, more precisely in the IP options field. Using these two variables, the OBP algorithm brings the information about the current state of the network path to the end systems. The network path can change during the communication process. However, to OBP, this fact is not important because OBP only needs to know the network state information provided by the bottleneck router. It is not relevant if the bottleneck router is always the same or changes during the communication process. With this information OBP calculates the *TR* to achieve the maximum network utilization and, simultaneously, avoid the network congestion.

OBP works as follows: the sender end system sends packets for the network at the rate *TR*. At the first node, the node updates the two OBP variables: *narrow link* and *tight link*. At the second node, if the *narrow link* variable, inside the packet, has a value greater than the current output link capacity, OBP replaces the value of the *narrow link* by with the output link capacity. The current value of the *tight link* variable is replaced if the available bandwidth of the output link is smaller than the current value of the *tight link*. The values of these two variables will arrive at the receiver end system. This end system sends back the ACK packet, which also contains the OBP variables. By this way, the information of the network state will arrive at the sender end system. Using this information the sender end system updates the *TR*.

Contrasting with XCP, in which the network nodes define the *TR* of all flows, OBP collects the state of the network path and uses it to continually keep the *TR* updated. This means that the end systems manage the *TR* and the network nodes are free to perform other functions.

A. OBP design

In OBP, the sender end systems define the *TR*. To do this, the sender end systems use the information about the current state of the network path received in the ACK packets. This state information allows a thin adaptation between the *TR* and the current network state. OBP uses a *Control Module* to efficiently use the network capacity and, simultaneously, to avoid congestion. It also uses a *Fairness Module* to provide fairness for all flows.

1) OBP Control Module

The objective of the OBP *Control Module* is to perform good network utilization and to avoid the congestion inside the network. To achieve these objectives, the sender end systems use two variables: *transmission rate* (*TR*) and *equilibrium point* (*EP*). When a new ACK packet is received, the end systems update the *TR*. The *TR* depends on the *EP* and the *available bandwidth* (*AB*). Therefore, OBP defines the *TR* using a *multiplicative-increase* (*MI*) algorithm. In this case, the *multiplicative-increase* is *conditioned* (*MIC*) by the network state. The *MIC* allows a quick increase of the *TR* and the flows can quickly use all network capacity, without producing congestion in the network.

Fig. 1 shows the *control factor* used to calculate the *TR*, based on the *AB*. If the network informs the end systems that *AB* near zero, the *control factor* is near zero (this means that the *TR* is equal to *EP*). If the network informs the end systems of an *AB* around 50%, the *control factor* is set to a value near 0.5 (this means that the *TR* is equal to *EP* plus *EP* multiplied by 0.5).

Whenever the end systems produce more load than the network capacity, the network informs them of a negative *AB*. In this case, the flows should reduce the *TR*. Therefore, the flows reduce the *TR* and this new *TR* will be equal to the *EP* minus the *EP* multiplied by a *control factor*. For example, if *AB* is near the double of the bottleneck capacity, this means that the flows have a *TR* near the double. So, the flows reduce to half of the current *TR*. This reduction is obtained by a *multiplicative-decrease* algorithm, *conditioned* by the state of the network path (*MDC*). In summary, OBP uses the *multiplicative-increase-conditioned-multiplicative-decrease-condictioned* (*MICMDC*) algorithm to lead to the convergence between the *TR* and the network capacity. The functions used by OBP will be presented in the next section.

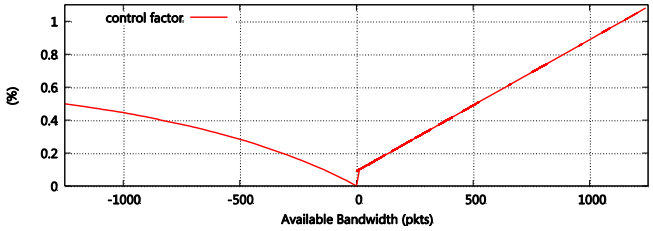


Fig. 1 – *Control factor* based on the available bandwidth. In this example the maximum AB is 1250 packets.

2) OBP Fairness Module

The objective of the *OBP Fairness Module* is to assure that

all flows can use a fair network bandwidth slice. This means that the flows with a higher *TR* should reduce their *TR* and the flows with a lower *TR* must be able to increase their *TR*.

OBP has a *fairness factor*. This factor is proportional to the *TR* of each flow. This *fairness factor* influences the calculation of the *TR*, which also depends on the *EP* and the *AB*. For instance, if one flow has a large *TR*, the *fairness factor* reduces the *TR* given by the *factor control*. As a result, the *fairness factor* helps the flows with a lower *TR* and penalizes the flows with higher *TR*.

Fig. 2 shows the *fairness factor* line and how this factor influences the *factor control* (the *increase factor*). So the *increase factor* represents the real increase and is based on the *control factor* and on the current *TR*. Fig. 2 shows that when the *TR* is near zero the *increase factor* is near 1. When the *TR* is near the maximum the *increase factor* is small. The OBP functions used to plot these lines will be presented in the next section.

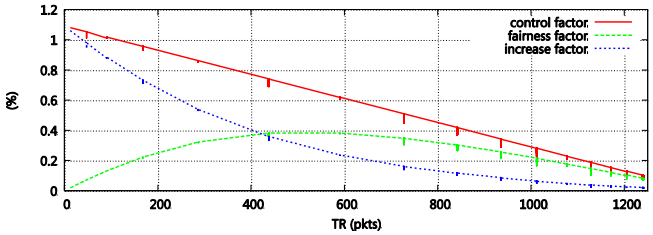


Fig. 2 – The *control factor*, the *fairness factor* and the *increase factor* based on the transmission rate of a flow. In this example the maximum *TR* is 1250 packets.

B. OBP implementation

OBP uses the support of network nodes to provide the transport service. As a result, OPB implementation is done at the end systems and at the intermediate nodes. The sender end systems do the most critical task. They have to estimate the adequate *TR* to efficiently use the network capacity, avoid congestion and guarantee fairness. The receiver end systems only have to receive the information of the network state in the data packets and send this information back to the source end-systems, through the ACK packets. The intermediate nodes only have to insert the information of its state inside the data packets.

1) End systems

The sender end systems use the *multiplicative-increase-conditioned-multiplicative-decrease-condictioned* (*MICMDC*) algorithm to calculate the *TR* of the flows. In terms of implementation, each flow has an *equilibrium point* (*EP*). This *EP* is updated once per *RTT*. When an end system receives an ACK packet, the end system looks for the information of the network state and uses it to calculate the new *TR* value. After this update, the end system puts the packets in network at the rate *TR*.

Expression (1) shows how the end systems calculate the *TR*. This calculation is used when *AB* is positive. The component (2) of (1) is the *control factor* and the component (3) is the *fairness factor*.

$$TR = EP + EP * \left(\frac{AB+\gamma*Max.TR}{Max.TR}\right) * \left(1 - \delta * \frac{EP}{\sqrt{(EP)^2+\left(\frac{Max.TR}{2}\right)^2}}\right) \quad (1)$$

$$EP * \left(\frac{AB+\gamma*Max.TR}{Max.TR}\right) \quad (2)$$

$$EP * \left(\frac{AB+\gamma*Max.TR}{Max.TR}\right) * \delta * \frac{EP}{\sqrt{(EP)^2+\left(\frac{Max.TR}{2}\right)^2}} \quad (3)$$

The network can also inform the end systems about a negative *AB*. This means that the *TR* is higher than the network capacity. When the *AB* is negative, the end systems decrease the *TR* using expression (4).

$$TR = \left(EP + EP * \left(\frac{AB}{Max.TR-AB}\right)\right) \quad (4)$$

Once per *RTT* the value of the *EP* is updated. The *EP* is equal to the sum of all the *TRs* calculated in the previous *RTT* period. The *EP* is updated through expression (5).

$$EP = \frac{\sum TR}{n} \quad (5)$$

The OBP constants have the following configuration: $\delta = 0.9$; $\gamma = 0.09$. The variable $Max.TR$ is calculated using the data received in the *narrow link* field.

2) Intermediate nodes

Each intermediate node knows its state. The node knows its interface capacity and its available bandwidth.

When a packet leaves the first node, the node updates the two OBP fields: the *narrow link* and the *tight link*. At the second node, the same packet is again updated if the variables *narrow link* and *tight link* have lower values than the values at the current node.

At intermediate nodes, the available bandwidth (*AB*) is maintained by expression (6). This expression uses the interface capacity (*IC*), the quantity of bytes received (*BR*) at this interface (this quantity is updated in short periods of time) and the amount of bytes inside (*BI*) the output interface queue when the last period of time started.

$$AB = IC - BR - BI \quad (6)$$

Fig. 3 shows how the OBP algorithm works. This picture was obtained by simulation. This figure shows the *transmission rate* (*TR*), the *equilibrium point* (*EP*) and the *available bandwidth* (*AB*). In this example, at the beginning, the network informs that there is 100 Mb/s of *AB*. This information is used to calculate the *TR* using expression (1). Since the *AB* is positive the *EP* and the *TR* grow quickly using the increase *MIC*. When the *TR* reaches the maximum and the *AB* falls to the minimum, both variables stabilize. The *TR* has a small oscillation at 100 Mb/s and around the value of *EP*. This oscillation is important because it allows the new flows to have *AB* for increasing its *TR*.

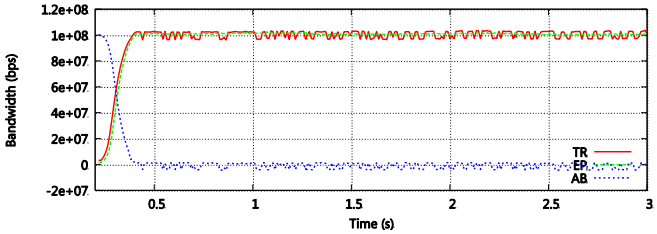


Fig. 3 – The evolution of three variables: *transmission rate*, *equilibrium point* and *available bandwidth*.

IV. EVALUATION

In this section, we discuss the evaluation results obtained with the use the NS2 simulator (Network Simulator version2) to evaluate the performance of OBP for several network scenarios. In the simulation scenarios the link capacity varies from 10 Mb/s to 2488 Mb/s; the round-trip propagation delay from 10 ms to 1 s; the number of long-lived (FTP) flows from 1 to 1000; and arrival rates of short-lived (web-like) flows of 100 per second. The simulation experiments included voice calls and streaming traffic. We generated traffic in two directions, forward path and reverse path. The bottleneck buffer size was set to one bandwidth-delay product. The data packet size was 1000 bytes, while the ACK packet was 40 bytes. The simulation time was 100 seconds. For comparison purpose, we also ran simulations for other schemes including the following protocols: TCP Reno [2], SACK [14], and XCP [4], under the same network and traffic settings. With the TCP Reno and the SACK we also enabled, in routers, the RED [15] and the ECN [13]. The settings of the TCP Reno, SACK and XCP were those recommended by their authors and presented in NS2 implementations. These three protocols were chosen because the TCP Reno and SACK are the transport protocols most used in the Internet, and XCP is the protocol with router support that has the best results.

A comparative evaluation between the four transport protocols was made. The scenarios used were the single bottleneck link and the multiple bottleneck links. We studied the effect of varying the link capacity, the round-trip propagation delay, and the amount of FTP flows. The simulation results demonstrate that, for a wide range of scenarios, OBP achieves comparable performance to XCP. For example, high convergence, high bottleneck utilization, negligible packet drop rate and low persistent queue. Both schemes, OBP and XCP, outperform the other schemes.

The OBP capacity, for distributing the network resources between all the flows, was assessed by the fairness testes. In these tests, the flows were started at spaced instants. From the results, it is possible to evaluate the OBP capacity to redistribute the network resources among all active flows. The Jain's Fairness Index [16] was also used to evaluate the OBP's fairness. The tests used the scenario with one bottleneck link and the RTT varied between 10 ms and 1 second.

A. *Scenario with one bottleneck link*

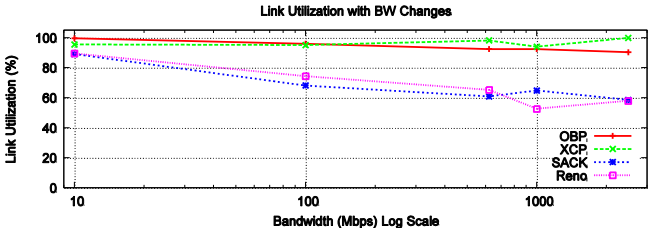
This section evaluates the OBP functionality in a scenario with one bottleneck link. This evaluation was done for several configurations: varying the bottleneck link capacity and varying the RTT. The OBP's fairness was also evaluated for several RTT configurations.

1) *Varying the bottleneck link capacity*

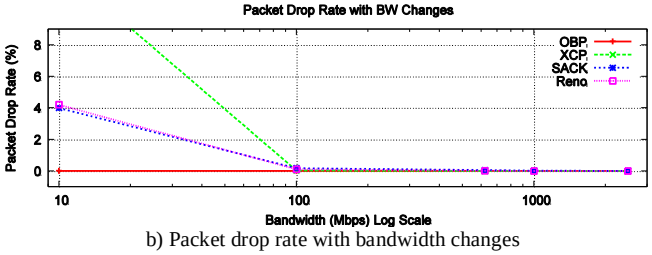
These experiments used the following settings: the round-trip propagation delay was 100 ms; 20 FTP flows in the forward path and 20 more in the reverse; the rate creation of the web-based flows was 100 per second; 10 voice flows and 10 streaming flows in the forward direction and 10 more in the reverse. The bottleneck capacity varied from 10 Mb/s to 2488 Mb/s.

As shown in Fig. 4, OBP and XCP achieve similar results. The bottleneck capacity has a high utilization with both mechanisms. No packet drop is detected for bottleneck capacities higher than 100 Mb/s. For the bottleneck capacity equal to 10 Mb/s, OBP achieves the correct *TR* to use the network capacity without drops. In all of the tests, OBP is the only scheme without drops. This fact is related with the OBP algorithm, which always defines the *TR* based on the available bandwidth.

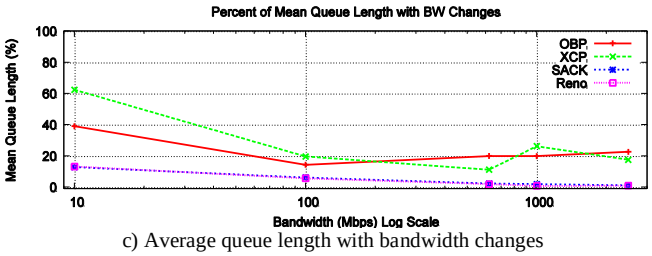
OBP and XCP maintain a persistent bottleneck queue length near 20%. The other two schemes, above 100 Mb/s, show no skills to use all the available bandwidth.



a) Link utilization with bandwidth changes



b) Packet drop rate with bandwidth changes



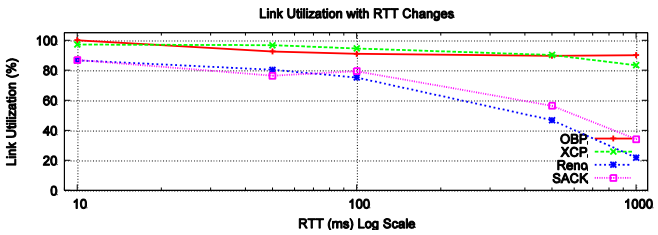
c) Average queue length with bandwidth changes

Fig. 4 - One bottleneck with the link capacity varying from 10 Mb/s to 2488 Mb/s

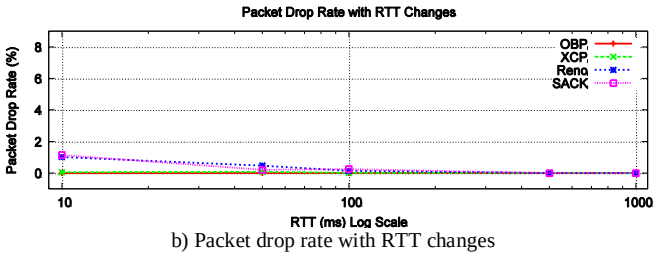
2) *Varying the Feedback Delay*

These experiments used the following settings: the bottleneck link capacity was 1 Gb/s; 100 FTP flows in the forward direction and 100 in the reverse direction; 100 new web-based flows per second; 50 voice flows and 50 streaming flows in the forward direction and 50 in the reverse. The RTT varied between 10 ms and 1000 ms.

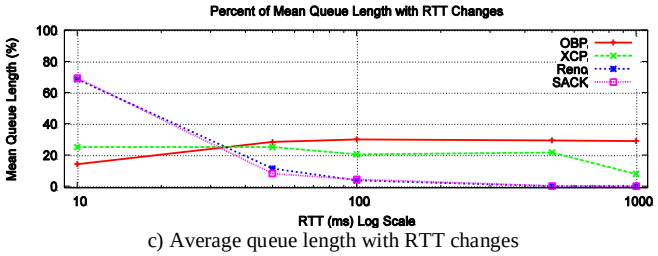
As shown in Fig. 5, OBP has high bottleneck capacity utilization, higher than 95%; the average bottleneck queue utilization is less than 20%; for high RTTs, OBP has higher bottleneck utilization than XCP; OBP is not sensitive to RTT variations. The results of OBP are similar in all tests. In all cases of wide RTT variation, no drops are detected. Comparing to the other schemes, the OBP performance is comparable to or better than XCP. OBP has better performance than the other TCP variants.



a) Link utilization with RTT changes



b) Packet drop rate with RTT changes



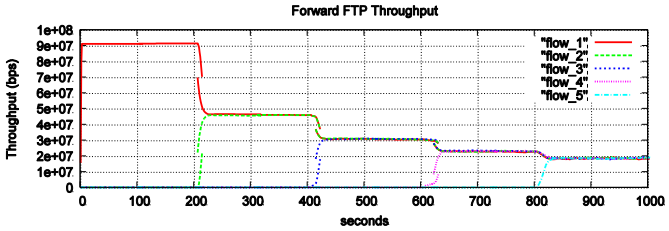
c) Average queue length with RTT changes

Fig. 5 - One bottleneck with the round-trip propagation delay varying from 10 ms to 1000 ms.

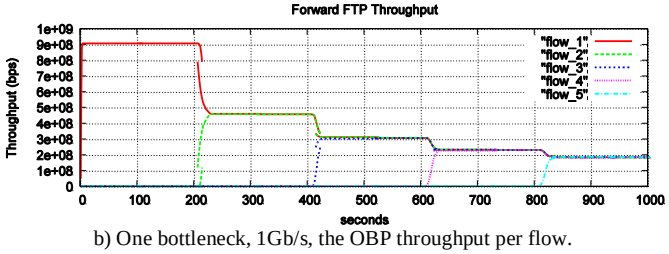
3) Fairness

This section evaluates the OBP's capacity to distribute the network capacity evenly among all the active flows. Fig. 6 shows the results of two experiments, differentiated by the bottleneck capacity, which varied from 100 Mb/s to 1 Gb/s. In these tests one new flow was started at each 200 seconds.

The results show OBP capacity to allow the increase of the *TR* of the new flows and, simultaneously, to force the older flows to decrease the *TR*. These results prove that the OBP's *fairness factor* can equally distribute the network capacity by all the active flows.



a) One bottleneck, 100Mb/s, the OBP throughput per flow.



b) One bottleneck, 1Gb/s, the OBP throughput per flow.

Fig. 6 - One bottleneck with one new flow started every 200 seconds.

The next experiments used the Jain's Fairness Index to evaluate the OBP competence to distribute the network resources among several flows, started at the same time. The scenario used 1 Gb/s bottleneck link capacity. The traffic included 50 flows in the forward direction and 100 new web-based flows per second. The RTT varied from 10 ms to 1 second.

Fig. 7 shows the results obtained. The protocols Reno and SACK achieved good results for low RTTs. For high RTTs there is a large degradation. XCP achieves good results for all the RTT configurations, with the exception of the RTT equal to 1 second, where there is a small degradation. The results of OBP are all near 1, which means that OBP can equally distribute the network capacity by all the active flows.

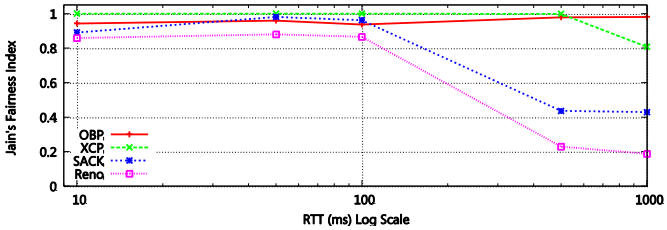


Fig. 7 - Jain's Fairness Index under scenarios of one bottleneck link.

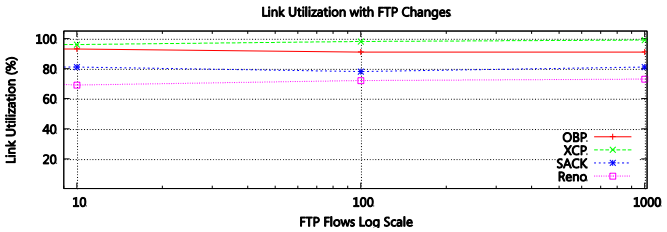
B. Parking-lot scenario with seven bottlenecks

This section evaluates the OBP capacity in a scenario with seven bottlenecks. This evaluation was done for several configurations: varying the number of long flows, varying the RTT. The results contain the average of all the results obtained in all the bottleneck links, in the forward and the reverse directions.

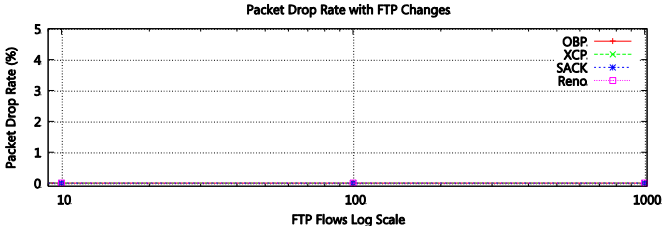
1) Varying the number of FTP flows

These experiments used the following settings: the seven bottleneck links capacity was 1 Gb/s; the RTT was 80 ms; 100 new web-based flows per second; 50 voice flows and 50 streaming flows in the forward direction and 50 in the reverse direction. The number of FTP flows in the forward direction varied between 1 and 1000 and another 100 flows in the reverse direction. Another 5 cross flows were used, in each bottleneck link, in forward direction, with 5 ms of propagation delay.

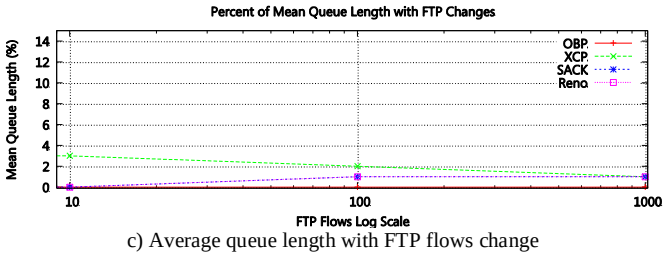
As shown in Fig. 8, OBP has high bottleneck utilization, always higher than 90%. The average bottleneck queue size is minimum. OBP and XCP have similar results. The results of OBP are similar in all settings.



a) Link utilization with FTP flows change



b) Packet drop rate with FTP flows change



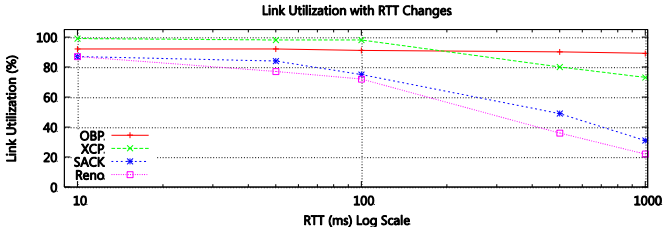
c) Average queue length with FTP flows change

Fig. 8 - Seven bottlenecks with the number of long-lived flows increasing from 1 to 1000.

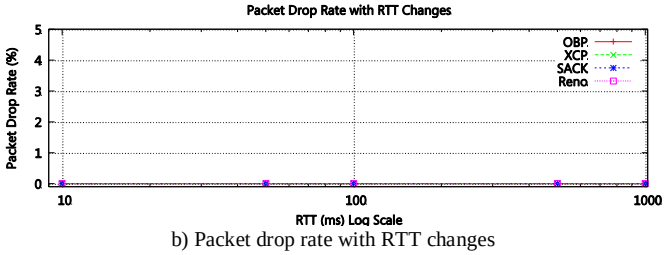
2) Varying the feedback delay

These experiments used the following settings: the bottleneck links capacity was 1 Gb/s; 100 FTP flows in the forward direction, 100 in the reverse direction and 5 cross flows in each bottleneck link with 5 ms of propagation delay; 100 new web-based flows per second; 50 voice flows and 50 streaming flows in the forward direction and 50 in the reverse. The RTT varied between 10 ms and 1 second.

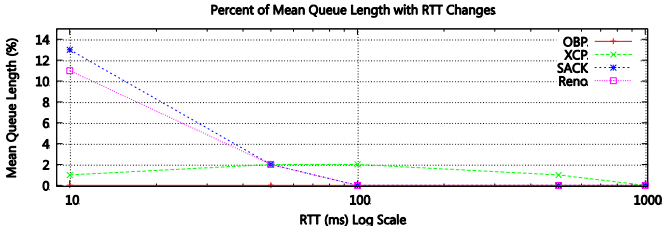
As shown in Fig. 9, OBP achieved high bottleneck utilization; the average bottleneck queue utilization is small; for high RTTs, OBP has higher bottleneck capacity utilization than CP; OBP is not sensitive to RTT variations; the results of OBP are similar in all tests; for all wide RTT variation, we do not observe any packet drops. Comparing to the other schemes, the OBP performance is comparable or better than XCP. OBP has better performance than other TCP variants.



a) Link utilization with RTT changes



b) Packet drop rate with RTT changes



c) Average queue length with RTT changes

Fig. 9 - Seven bottlenecks with the round-trip propagation delay varying from 10 ms to 1000 ms.

V. Conclusions and Future Work

In this paper we presented an evaluation of Open Box Transport Protocol (OBP). OBP is a congestion control protocol, which can be used in large BDP networks. OBP is an explicit congestion control approach, where the senders decide the congestion control actions based on information about the network state. In OBP, the senders have skills to look inside the network and to make congestion control decisions, based on the most restricted interface capacity and the available bandwidth.

Using the NS2 simulator, we show that OBP achieves high bottleneck utilization, negligible packet loss, low persistent bottleneck queue, convergence, good fairness, and it outperforms XCP in some conditions. OBP achieves these results and maintains the congestion control decisions at the sender end systems.

As part of our future work, we plan to evaluate OBP in wireless networks and to implement OBP in the Linux operating system.

References

[1] D. Chiu and R. Jain. "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance", in Computer Networks. J. of Computer Networks and ISDN, 17(1):1-14, June 1989.

[2] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control", IETF RFC 2581, April 1999.

[3] V. Jacobson, "Congestion Avoidance and Control", SIGCOMM'88, August 1988.

[4] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", SIGCOMM'02, August 2002.

[5] Paulo Loureiro, Saverio Mascolo, and Edmundo Monteiro, "Open Box Protocol (OBP)", in proceedings of the High Performance Computing and Communications (HPCC) 2007, Houston, USA, September 2007.

[6] Network Simulator NS2. Http://www.isi.edu/nsnam/ns/.

[7] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", IETF RFC 2582, April 1999.

[8] Cheng Jin, David X. Wei, Steven H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance", in Proceedings of IEEE Infocom 2004, 2004.

[9] Floyd, Sally. s.l. "HighSpeed TCP for Large Congestion Windows", RFC 3649, December 2003.

[10] Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M., Wang, "TCP Westwood: End-to-End Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks", in Proceedings of ACM Mobicom 2001.

[11] Yong Xia, Lakshminarayanan Subramanian, Ion Stoica, Shivkumar Kalyanaraman, "One More Bit Is Enough", in Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications 2005, Philadelphia, Pennsylvania, USA August 22 - 26, 2005.

[12] A. Jain and S. Floyd, "Quick-Start for TCP and IP", IETF Internet Draft draft-amit-quick-start-02.txt, October 2002.

[13] K. K. Ramakrishnan and S. Floyd, "The Addition of Explicit Congestion Notification (ECN) to IP", IETF RFC 3168, September 2001.

[14] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options", IETF RFC 2018, October 1996.

[15] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Trans. Networking, 1(4):397-413, August 1993.

[16] R. Jain, "The art of computer systems performance analysis: techniques for experimental design, measurement, simulation and modelling". New York: John Wiley & Sons, 1991.