Performance Analysis of TCP Variants

Harshad Sathe, Kaustubh S Pande
Northeastern University

1. ABSTRACT

Data transmission between hosts is mostly dictated by TCP variants today. There are many factors that affect the flow of data across these connected entities. Imbalance in the flow of data often leads to network Congestion. Each TCP variant deals with these situations with various Congestion avoidance and detection techniques. The goal of this paper is to study the performance of TCP variants under Congestion.

2. INTRODUCTION

Despite acute differences in features and implementation, Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) still remain two of the widely used members of Internet Protocol suite. While TCP provides connection oriented, full-duplex and reliable packet transmission across the heterogeneous processes on terminals over an Internet, UDP is the connection-less, non-reliable protocol. Both, however, are placed on the fourth layer, i.e. the Network Layer of the OSI model.

TCP has been extensively used across applications on every operating system platform. Its point-to-point connection scheme, synchronized start-up between endpoints by means of a Three-way handshake and a graceful termination of connection by guaranteeing delivery of data to the end points are some of the features that made it so successful. However, not all devices and networking requirements were alike. Diverse networking conditions gave rise to many variants of TCP, though underlying TCP working principle remained the same.

For this study, we are considering four commonly used TCP variants, TCP Tahoe, TCP Reno, TCP New Reno and TCP Vegas. TCP Tahoe is the modification of default TCP, proposed by Vin Cerf and Robert Kahn, in that it has new algorithms like Slow Start and Congestion Avoidance implemented. Congestion Avoidance, in this case, is achieved by 'Additive Increase Multiplicative Decrease' principle. The inability of TCP Tahoe to perform better with network congestion gave rise to another variant, TCP Reno, which along with preserving Tahoe's features, supports Fast Recovery. TCP Reno, however, waits before retransmitting when multiple packets are lost from window. This is overcome by the new variant TCP New Reno, which gives high retransmission time. TCP Vegas, on the other hand, follows packet delay based Congestion Avoidance scheme as against packet loss based scheme of other variants. And, the delay of packets determines the rate at which packets are to be sent.

We have performed three experiments. The first experiment observes the performance of above listed variants under Congestion. In similar conditions of Congestion, TCP Vegas yields the maximum throughput, drops the least and has the smallest latency. So, Vegas clearly outperforms other variants. We will analyze details in the subsequent section. The second experiment is to observe how fair TCP variants are to each other when they are sharing the bandwidth. It was observed that similar variants treat each other reasonably fairly, but non-similar ones compete and if one of them is New Reno, it almost always wins. We will elucidate that in the respective section. The third experiment is to observe influence of queuing over the pair of variants. Here, RED emerges as a winner over DropTail to provide fair bandwidth between SACK and Reno. We explain why in the corresponding section.

Section 3 describes the methodology adopted for all of these experiments. We deal with details of all three experiments in the subsequent section 4. This section has graphs with Standard Deviation plotted on them using Error Bars. Section 5 goes on to conclude the paper summarizing the results of the experiments. Section 6 is about some online and other references we used to analyze obtained results.

3. METHODOLOGY

There are many network simulators available in the market. NS2, NS3, QualNet, GloMoSim, NetSim, OMNeT++, OPNET are just some of the examples. NS2 is one of the most popular simulators due to its flexibility, modular behavior and open source availability. It is discrete event driven simulator that supports various routing algorithms, queuing algorithms, etc. over wired and wireless networks. Given the fact that it is written in C++ and Object Oriented Tool Command Language (OTcl), it is more suitable for users to control simulation through Tcl scripts. Also, NS2 simulates majority of TCP variants. Its support for NAM and X-Graph to animate and plot the simulation results makes it really helpful for users to interpret results. NS2 also has vey large and active user community. These factors led us to choose NS2 over other tools.

Fig 1: Network Topology

Figure 1 shows the Network topology we have used in all the experiments. All the links between nodes are 10 Mbps full duplex links and follow DropTail Queuing algorithm for all experiments except the third one. We will explain the reason exclusively there. Node N1 is the TCP source attached to FTP application. Node N2 is the CBR source which is unresponsive UDP flow. Node N3, in this case, is the sink of the CBR flow, while N4 is the sink of TCP flow that is sourced at N1. Similarly, N5 and N6 are the TCP source and sink respectively which are idle for the first experiment.

Tcl scripts yield their simulation results in trace files (with .tr extension) and we have written Python scripts to analyze results to calculate throughput, latency and the drop rate for each of the network settings. We have also performed T-Test and Standard Deviation test to analyze if the differences observed are statistically significant. Standard Deviation is plotted on each of the graphs by means of Error Bars.

For the first experiment, the CBR source at N2 introduces the congestion on the link N2-N3. We are observing performance of all four variants in the load conditions. This was primarily observed for increasing CBR bandwidth from 1Mbps to 10 Mbps gradually. We tried below variations for this experiment:

- Time Difference: Altering when TCP and CBR sources start changes the congestion situation across N2-N3.
- Change in packet size: We tried increasing and decreasing packet sizes of both TCP and CBR flows to observe the effect on parameters.

For the second experiment, though topology remains the same, we varied the number of TCP sources to two. Now, both N1 and N5 are in use, while N2 remains the source of CBR flow. We observed the fairness among Reno, New Reno and Vegas. This was primarily observed for increasing CBR rate. We will vary the time at which each variant starts and time taken by each variant to converge.

Finally, the third experiment is about choosing right queuing algorithm among Drop Tail and Random Early Drop (RED). TCP flows exist between Nodes N1-N4 and N5-N6. CBR flow is on N2-N3 and TCP variant on each of these source nodes is different. The performance of TCP and CBR is observed over time for the above mentioned queuing algorithms.

4. RESULT ANALYSIS

4.1 TCP Performance Under Congestion

The objective of this experiment is to analyze performance of TCP variants, with the linear increase in the CBR rate. The bandwidth of each of the links in the topology in Figure 1 is set to 10Mbps. CBR rate is increased linearly by 1 Mbps for each run until it reaches 10Mbps. Network parameters are recorded at each of those rates.

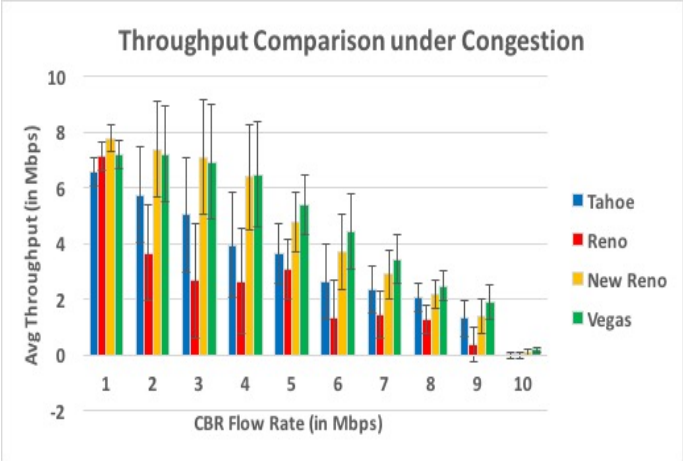


Fig 2. Throughput comparison of TCP Variants

Above **Fig 2** shows the graphical representation of Throughputs over increasing CBR rate. Analyzing this and further graphs will help us know which among the four TCP variants in question yields better throughput, lesser drop rate and less latency.

At the low CBR rate, as we can see above, New Reno is a clear winner. But as the traffic increases, New Reno's throughput decreases linearly. For the higher CBR rates, Vegas consistently performs better. Second to yield maximum throughput in this case, is New Reno. New Reno's better performance at the lower CBR rates and overall better performance than Tahoe and Reno can be attributed to its implementation of Fast Transmission and Fast Recovery. These two features give New Reno the ability to recover quickly from packet drop in case of Congestion. Tahoe's poor performance in the high Congestion is due to the Slow Start mechanism it implements. Packets are transmitted slowly when congestion is encountered and this reduces the overall throughput of Tahoe. Reno waits before retransmitting packets in situations of Congestion though it recovers quickly. This is the reason throughput value does not have linear pattern. In fact, it is swinging around optimal value. Congestion control mechanism of Vegas is delay based. Therefore, it is slow in retransmitting packets at the lower CBR rates as compared to New Reno. However, it picks up on the higher CBR rates and retains the lead.

We performed unequal Variance T-test for throughput values of Vegas and New Reno, Tahoe and Reno. The resultant Value of p (probability) is greater than the set

value of 0.5, which means values are statistically insignificant.

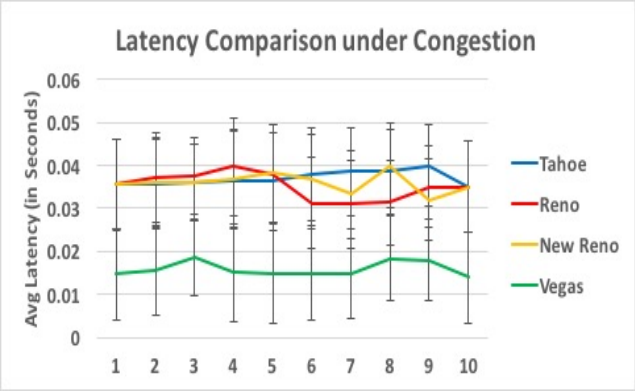


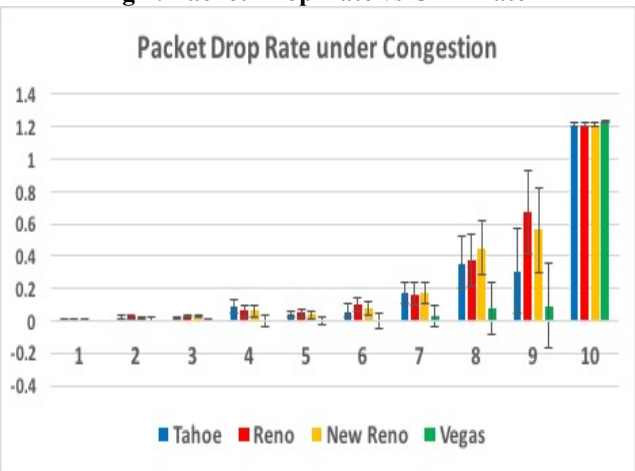
Fig. 3: Average Latency vs CBR Rate

As we can see in above **Fig 3**, in case of latency, the Vegas variant clearly emerges as the winner over all the other three variants. Right from low CBR till the maximum rate, Vegas has the least Latency. This is primarily because of delay based Congestion Control mechanism Vegas has adopted. Other packet loss based TCP variants have almost similar values at given CBR rate. Vegas' excellent performance can be attributed to the fact that it closely monitors the Round Trip Time (RTT).

Above results are confirmed by the T-test as well. When we compared latencies each of Tahoe, Reno and New Reno with Vegas, assuming unequal variance and tail value as 2 for stricter tests, the resulting p value is less than set value of 0.5, which means these values differ significantly. And, other variants when compared among themselves, have no significant statistical differences.

Fig 4 shows the graphical representation of TCP variants in case of packet drop rate. As we can see, Vegas (in blue) clearly outperforms other variants. Vegas has very minor or in fact zero packet loss until 6 Mbps. Then, packets start getting dropped. Packet drops for Tahoe, Reno and New Reno increases steadily with increase in CBR Rate. Finally, packet drop rate of Vegas becomes equivalent to other variants in case of 10Mbps CBR rate.

Fig 4: Packet Drop Rate vs CBR Rate



These results are also confirmed by unequal variance T-test. In this T-test, when compared values of Tahoe and Reno, Reno and New Reno, we observed that probability value is greater than already set value of 0.5, which means there is not much significant difference in these values. However, when compared with Vegas, p value is less than 0.5 helping us infer Vegas packet drop differs significantly.

Thus, after experimenting with above settings and trying several other settings described in Methodology section, we can safely infer that though Vegas and New Reno are close in terms of throughput, when compared with other two parameters, Vegas clearly has an upper hand over all the three other variants in case of Congestion.

4.2 Fairness Between TCP Variants

Out on the internet there are diverse operating systems, each of which may use different TCP variant for communication. The objective of this experiment is to analyze the fairness of TCP variants when they share the network medium with other TCP variants. As in the experiment 1 we recorded throughput, latency and drop rate values for each of the TCP variant pair Reno/Reno, NewReno/Reno, Vegas/Vegas and NewReno/Vegas. Following graphs would help analyze the fairness between the TCP variants.

In this test, although we expected both TCP variants should be fair to each other, initially the TCP Reno source at N5 had more throughput than the TCP Reno source at N1. The difference in the throughput is only visible in case of low CBR bandwidth (**Fig 5**). In order to transmit in a less congested network, the TCP Reno drops packets and enters fast recovery, which increases latency and decreases its throughput. The two TCP variant experience similar drop rates and have fair throughputs as the congestion gradually increases and the CBR at N2-N3 starts consuming more bandwidth.

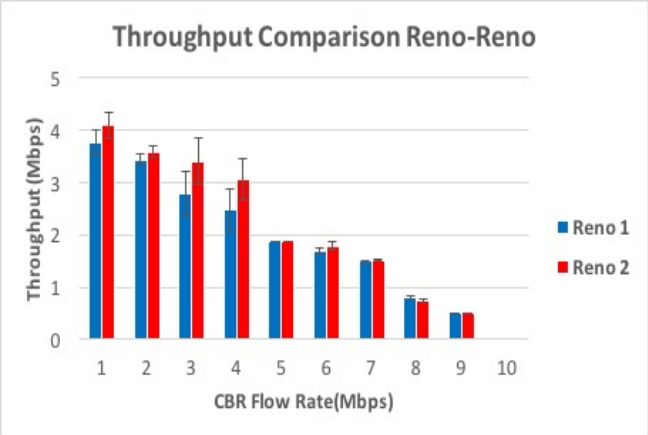


Fig 5. Reno-Reno Throughput vs CBR Rate

The above results are confirmed by T-test analysis. By assuming unequal variance and tail value as 2 for stricter tests, the resulting p value is greater than set value of 0.5, which means these values do not differ significantly.

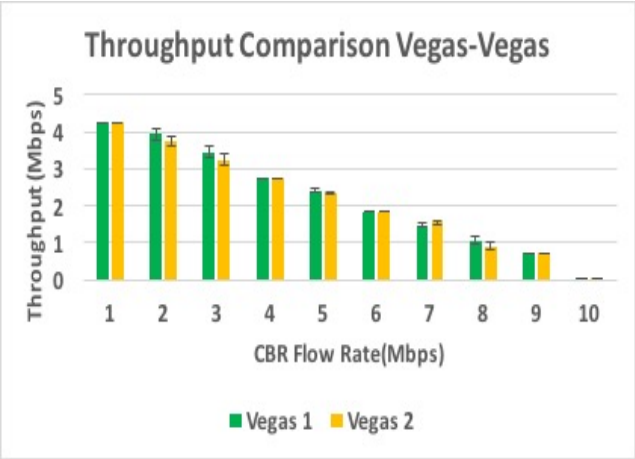


Fig 6. Vegas-Vegas Throughput vs CBR Rate

In case of Vegas/Vegas TCP Variants in **Fig 6,** we can observe that the two variants are fair to each other and have almost similar throughput even when the congestion increases. The reason for this behavior is that they have same congestion control mechanism implemented.

Above results are confirmed by T-test analysis. By assuming unequal variance and tail value as 2 for stricter tests, the resulting p value is greater than set value of 0.5, which means these values do not differ significantly.

The next TCP pair which we considered for the simulation test was TCP NewReno/Reno in the same network configuration and resulted in unfair throughput obtained for TCP Reno as seen in **Fig 7.** The variance in throughput is due to the different congestion control algorithms implemented by each of the two variants. New Reno achieves high throughput because of its property to retransmit immediately after one consecutive acknowledgement. However, Reno's performance is aggressively affected as it retransmits the lost packet only after waiting for three successive acknowledgements.

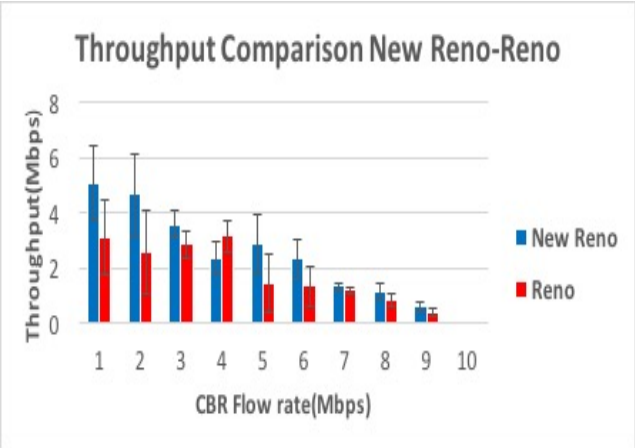


Fig 7. New Reno-Reno Throughput vs CBR Rate

The above results are confirmed by T-test analysis. By assuming unequal variance and tail value as 2 for stricter tests, the resulting p value is less than set value of 0.5, which means these values differ significantly.

In our fourth simulation test for TCP variants we used NewReno/Vegas. Again, we can verify from the graphs in **Fig 8,** that the two variants are unfair to each other. As an indication to determine rate at which needs to be sent, Vegas emphasizes on packet delay rather than packet loss. Thus when the congestion is increased due to CBR flow and other TCP variant New Reno, Vegas decreases its throughput.

The above results are confirmed by T-test analysis for New Reno-Vegas. By assuming unequal variance and tail value as 2 for stricter tests, the resulting p value is less than set value of 0.5, which means these values differ significantly.

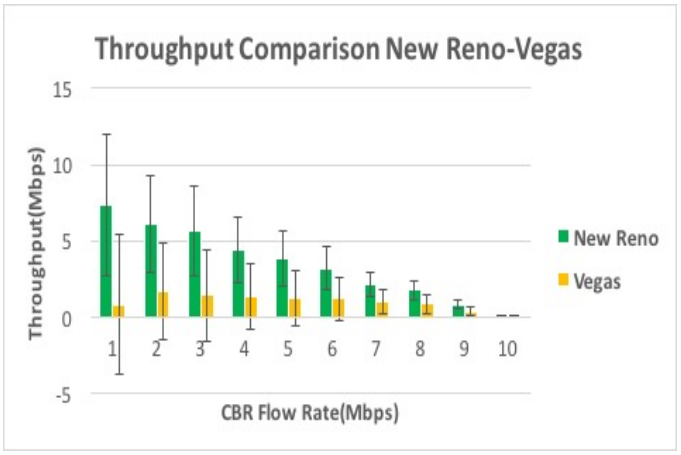


Fig 8. New Reno-Vegas Throughput vs CBR Rate

4.3 Influence of Queuing

The objective of this experiment is to study the influence of queuing discipline on the overall throughput and latency of flows. We have used the same topology as Experiment 1. In this experiment, we start the TCP flow first, once it is stabilized, the CBR rate of 6Mbps is started. We then compare the throughput and latency of TCP variants Reno, SACK. This is measured against increasing time. And the queuing algorithms used are DropTail and RED.

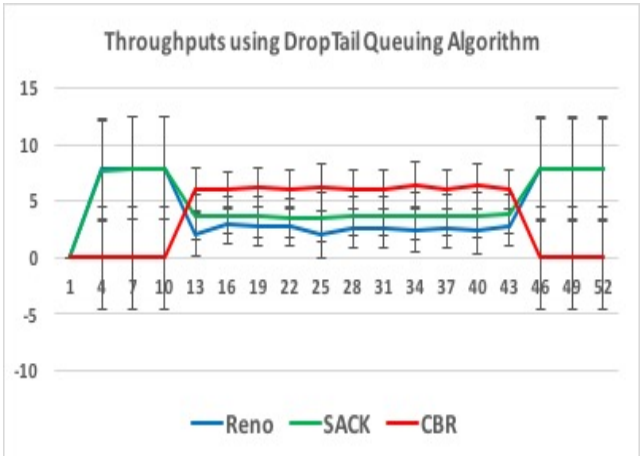


Fig 9. Throughput (in Mbps) vs Time (in Seconds)

In case of DropTail algorithm **Fig 9**, we can observe that SACK and Reno achieve maximum throughput in the first 10 seconds before CBR flow begins. As soon as the CBR flow starts, the TCP throughput is reduced to half instantly and keeps oscillating until the CBR is stopped at 44 secs. After that, both TCP variants attain their maximum throughput back. As shown in **Fig 9**. And **Fig 10** we can also notice that Reno utilizes lower bandwidth when the DropTail queuing algorithm is implemented at that node. Also, the average throughput difference between the TCP variants SACK and Reno is less for RED algorithm, indicating that DropTail is unfair to the different TCP variants. The primary reason for this conclusion is, the fact that RED algorithm drops packets on basis of probabilistic analysis. It allows fair share to TCP flows in its queue. However, in case of DropTail, the packets are dropped as soon as the queue becomes full. We can also understand from these graphs that along with time, RED algorithm is fair to share the bandwidth for Reno and SACK.

Red along with SACK is a good idea, for the fact that SACK is aggressive to deal with multiple packet drops from a window of data RED analyses network congestion and uses probabilistic approach to drop packets as soon as the buffer reaches its capacity rather than handling surge traffic. Thus, we can say that SACK along with RED would achieve stable and high throughput under congestion.

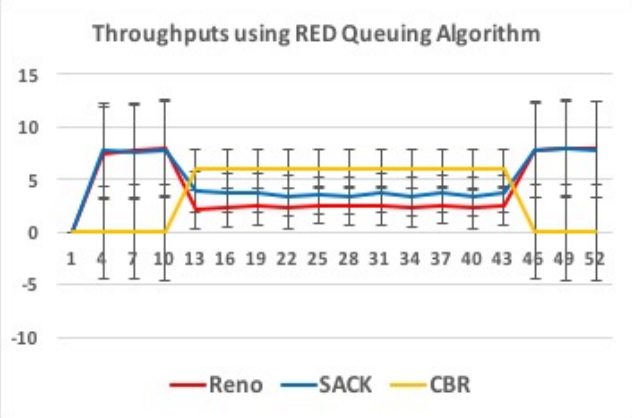


Fig 10. Throughput (in Mbps) vs Time (in Seconds)

In order to determine end-to-end latency for the flows which differ between RED and DropTail, we measure performance for TCP Reno. From **Fig 11** we can understand that the latency for RED queuing algorithm reduces as we increase the Queue size. We can also observe that the values for RED are more stable under congestion as compared to DropTail. Based on this experiment we can conclude that fair bandwidth utilization and average latency is better for RED queuing algorithm.

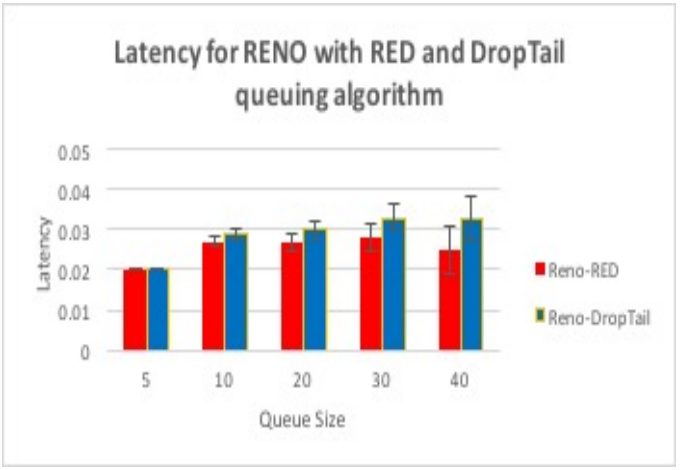


Fig 11. Latency comparison for RENO

5. CONCLUSION

The objective of this project was to compare TCP variants under Congestion. In the first experiment, we compared Tahoe, Reno, New Reno and Vegas based on congestion. Our results have proved that considering collective results of throughput, latency and packet drop rate, Vegas is the best variant. T-test and Standard Deviation gave a statistical backing to results. The second experiment is more real life. There are multiple situations where different types of TCP variants have to share bandwidths with each other. We concluded through our results that TCP variants are unfair to each other, much against our initial belief. New Reno is especially dominant, mostly due to its Fast Retransmission algorithm. Whenever New Reno is paired with any one of the other variants, it dominates. It was reasonably fair with Reno but with Vegas, its much more aggressive. Similar variants, however, seems to have reasonably fair distribution of bandwidth among themselves. Third experiment assessed the impact of queuing algorithms RED and DropTail on TCP flow. Lower latency rates comparison displayed that RED is a better choice under congestion.

Thus, Vegas emerged to be the best variant among the four, and RED proved to be better quieting algorithm, however, analyzing few more TCP variants will give us the conclusive evidence about the overall best variant.

6. REFERENCES

1. http://www.academia.edu/9202646/Detail_Comparison_of_Network_Simulators
2. 'Simulation Based Comparisons of Tahoe, Reno and SACK TCP' by Kevin Fall and Sally Floyd.
3. 'Computer Networks, a Systems Approach', 4th Edition by Larry L. Paterson and Bruce S. Davie
4. 'Computer Networks, by Andrew S Tannenbaum