Performance measurement of computer networks to improve congestion control

Ms. Banshri Raichana and Mrs. Sangeeta Kulkarni

Abstract—In communication networks, congestion occurs when a link or node is carrying data beyond its capacity limits and hence its quality of service (QoS) deteriorates. Typical effects of congestion include queuing delay, packet loss (loss of information) or the blocking of new connections. Congestion needs to be controlled to ensure the QoS and also to increase efficiency of the system in order to survive the ever growing competition of service providers in market today. In order to evaluate the QoS it is essential to measure the performance of network in terms of throughput, efficiency, delay, packet dropping probability etc. These parameters are influenced by variations in the network parameters like packet size, packet interval and link capacity. To implement congestion control early detection of packet drop or monitoring link buffer would help avoid congestion. The concentration of this paper is around network parameters, how they affect system performance and how these results can help avoid congestion scenario.

Index Terms—Network congestion, Packet delay, Quality of Service (QoS), Throughput.

I. INTRODUCTION

Even before we implement any network security protocol, congestion control protocol or routing protocol it is very essential to understand various network parameters and the effect of varying these parameters on the performance of the network. It is observed that when a network link fails, the packets travelling through this link either need to be routed or buffered to avoid data loss. In order to avoid such scenarios early detection of congestion needs to be done.

The network designing and simulation is accomplished using NS2 (network simulator version 2) [1][2]. NS2 is an event-driven network simulator embedded into the Tool Command Language (Tcl). An extensible simulation engine which is implemented in C++ and is configured and controlled via a Tcl interface. Using the 'ns' command, a network topology is defined, traffic sources and sinks are configured, statistics are collected, and then simulation is invoked.

In general, C++ is used for implementing protocols and extending the ns-2 library[1]. OTcl is used to create and control the simulation environment and also for selection of output data. Simulation is run at the packet level, allowing visualization of detailed results.

This network has been designed to implement bottleneck and packet drop scenario. Using the available tools of NS2 and by varying the network parameters like link capacity, packet size and packet interval we can observe the effect on network performance and can implement the network congestion avoidance algorithm Random Early Detection (RED) [4].

Random early detection (RED) is an active queue management algorithm[4]. It is also a congestion avoidance algorithm. In the conventional tail drop[5] algorithm, a router or other network component buffers as many packets as it can, and simply drops the ones it cannot buffer. If buffers are constantly full, the network is congested. Tail drop distributes buffer space unfairly among traffic flows. Tail drop can also lead to TCP global synchronization[5] as all TCP connections "hold back" simultaneously, and then "step forward" simultaneously. Networks become under-utilized and flooded by turns. RED addresses these issues.

RED monitors the average queue size and packets based on statistical probabilities. If the buffer is almost empty, all incoming packets are accepted. As the queue grows, the probability for dropping an incoming packet grows too. When the buffer is full, the probability has reached 1 and all incoming packets are dropped. Thus RED buffer mechanism with constant bit rate traffic can be used at an initial stage to understand the effect of change of network parameters over system performance.

II. DESIGNING THE NETWORK

There are four basic steps to design and simulate any network. First step is to develop a model (implementation of a protocol); second step is to create a simulation scenario (designing a network topology and traffic scenario); third step is to choose and collection of statistics, and finally fourth step is
to visualize and analyze simulation results which may be carried out after the simulation execution.

As shown in Fig.1, a network with four nodes (Node0, Node1, Node2, Node3) has been designed. A link with appropriate link type, link capacity, buffer mechanism is to be defined between these nodes. In this network a bottleneck between node2 and node3 is defined so that we can simulate packet loss and measure overall network performance.

![Network simulation window in NS2](image)

**Fig 1. Network simulation window in NS2**

### III. NETWORK PARAMETERS

In communication networks throughput[6] is the average rate of successful message delivery over a communication channel. The throughput is usually measured in bits per second (bit/s or bps). The average throughput of the network was calculated by calculating total number of packets that were transmitted, received and or dropped. The output file generated after simulation of the network is used along with an AWK program to calculate throughput of the network. AWK is a language for processing text files. An AWK program is a sequence of pattern-action statements. AWK reads the input which in our case is the output file generated after simulation by NS2. A line is scanned for each pattern in the program, and for each pattern that matches, the associated action is executed.

The delay[7] of a network specifies how long it takes for a bit of data to travel across the network from one node or endpoint to another. It is typically measured in multiples or fractions of seconds. For calculating packet delay the packet ID ($12), source ($2) and destination node ($3) need to be matched correctly. Where $ indicates the column in the output file (.tr or trace file) generated after simulation by ns2. Likewise for throughput aggregate value of $6 is taken and total number of received bits are calculated to get the overall throughput. Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination.

### IV. RESULT ANALYSIS

#### Analysis by varying the link capacity

Keeping the packet interval constant (0.005ms) and packet size constant (500bytes) it is observed from Fig 2. that as we go on increasing bottleneck link capacity(bandwidth) the efficiency goes on increasing since the number of dropped packets reduces and thereby the throughput of overall network increases.

![Varying Bandwidth v/s Efficiency](image)

**Fig 2. Varying Bandwidth v/s Efficiency**

#### Analysis by varying packet size

In this analysis, keeping the packet interval constant (0.005ms) and link capacity of each link constant (1MB) we can observe from Fig 3. that as we go on increasing the packet size(bytes), number of dropped packets goes on increasing and received packets goes
on reducing proportionally, thereof both the efficiency and throughput reduce.

![Fig 3. Varying Packet size v/s Efficiency](image)

**Analysis by varying packet interval**

We can see from Fig 4. that if the inter packet arrival delay is increased, throughput goes on reducing since the number of dropped bits goes on reducing and approaches to zero i.e all the transmitted bits are being received. Efficiency thereby goes on increasing and attains saturation value of 100%. For this analysis we have kept the packet size constant (500 Bytes) and link capacity of each link constant (1MB).

![Fig 4. Varying Packet interval v/s throughput](image)

V. CONCLUSION

The network parameters were varied and its effects on performance of the network was monitored. These analysis not only help us to design better networks with less delay and maximum throughput but it will also help us avoid congestion or control congestion. A congestion control algorithm RED is used to improve the performance of network by reducing packet dropping probability. This buffer handling mechanism can also be used to prioritize packet drop and further improve QoS and system performance.

VI. REFERENCES