Implementation of stop-and-wait protocol on a network

Omkar Bhalekar, Abhishek Sakpal, Rohankumar Vashi

Principles of Telecommunication Networking TCET (710)

College of Applied Science & Technology

Rochester Institute of Technology

1, Lomb Memorial Drive, Rochester, NY, 14623

***Abstract*—The stop-and-wait protocol is a protocol used in networking for successful transmission of packets. This protocol is used for ensuring the packets reach their desired destination and the destination node send an acknowledgement of the received packet. Throughput is an important part of a network which convey the information about number of packets transmitted per unit time. We have implemented this using python programming language to analyze the working of protocol. We have assigned the port numbers to edge nodes and switches. Also, defined connection list which are specified by port numbers of edge node and switch node. As the probability of transmission changes, the throughput of network changes. In this network, some of the edge nodes are transmitting packets while some are receiving packets and sending an acknowledgement via switch. Finally, we have calculated the throughput of a network.**

I. Introduction

In data communication domain, conveyance of information amongst smaller networks is easy to understand and debug. But larger networks are highly susceptible to errors especially when the probability of the all the nodes involved in a network is same, then the network condition deteriorates giving rise to congestion which can either lead to discarded packets or denial of service. This directly affects the overall throughput i.e. The amount of data in our case packets that traverse through a given network per unit time.

To handle the errors in a network, one of the extensively used techniques based on reliability is using error detection followed by re-transmission on request. One of the schemes used here is Stop-and-Wait. In Stop-and-Wait technique, the source node sends one packet at a time and waits for the destination to reciprocate until it transmits the next packet. Once the destination sends an acknowledgement that it has received the packet successfully only then will the source node transmit another packet. One more parameter that comes into picture is the Time to Live(TTL) for each packet. TTL limits the lifespan of the packets and the packets are discarded once the designated TTL expires. In this way the source does not have to wait forever to receive the acknowledgement and then it can re-transmit the same packet again. To make things simpler for the receiver i.e. reassuring that the packets received are unaltered, the source introduces a redundancy check for each packet it transmits in a sequential fashion. With the help of this redundancy check, the receiver or the destination node can easily make out the reception of erroneous packets thereby discarding them and abstaining itself from sending an acknowledgement to the transmitter. Based on the successful transmission of the packets, we can easily calculate the throughput of the network.

In this paper, we analyse the throughput of a network under variety of traffic conditions. For that we have proposed a nexus of six interconnected switches connected to six edge nodes each. Three out of six edge nodes are configured in such a way that they follow the Stop-and-Wait scheme and the remaining three nodes are configured just to receive acknowledgments and do not transmit any packets. The six switches can be configured using manual routing tables. So, each switch will have its own routing table specifying shortest path the packet takes to arrive at the destination. To assess all possible combinations for transmission, we have considered the likelihood of transmission of all the nodes as identical. So, for this we consider test cases for three probabilities'. 0.01, 0.15, 0.2. All nodes transmit in three different time frames of 10000, 20000, 30000-time steps. As each node will transmit the packet simultaneously, the packet will be stored in switch connected to it. The switch does a work of forwarding and not anything else. The switch maintains its routing table and update after each time steps. The routing table contains the information of port number, edge node and route to reach each node.

II. Solution

The network topology consists of six edge nodes and six switch nodes. Each edge node is connected to separate switch node. And, switch node is connected to other two switch nodes. This means, when e1 wants to transmit packet to e3, then it will take path of e1-s1-s2-e3. We have assigned the port number to port of switches and edge nodes which are mentioned in edge node and switch node initiation functions. As per the condition, three edge nodes are programmed to transmit and remaining

three nodes are receiving. These three remaining nodes must send acknowledgments to transmitter.

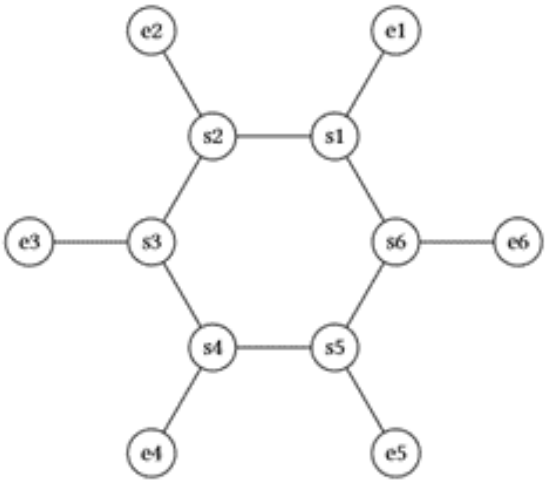

Fig 1. Network Topology

e1, e2, e3, e4, e5 e6: Edge Nodes

s1, s2, s3. s4, s5, s6: Switch Nodes

We are simulating our program for 10000, 20000 and 30000-time steps with transmitting probabilities of 0.01, 0.15 and 0.2.

The following table will throw some light on throughput values for different scenarios.

	Probability 1	Probability 2	Probability 3
Time Steps	0.01	0.15	0.2
10000	0.0285	0.2873	0.3243
20000	0.0321	0.2887	0.32205
30000	0.0299	0.2852	0.3230333

Table 1: Throughput values for different probabilities with 10000, 20000 and 30000 time steps

We configured the switch nodes as per the above mentioned table. This helped us in implementing the stop-and-wait protocol very easily. We improved the code for receiving acknowledgement from receiving edge node to transmitting node. The program shows the source edge node and destination edge node and the time step at which the packet is transmitted. The acknowledgements are also determined in same way as packet. When destination node sends an acknowledgement that time the message pop up saying 'ACK RECEIVED AT [DESTINATION NODE] FROM [SOURCE NODE]'.

We are assuming following port number of switch nodes connected to respective switch nodes.

Switch #	Port #1	Port #2	Port #3
S1	E1	S6	S2
S2	S1	E2	S3
S3	S2	E3	S4
S4	S3	E4	S5
S5	S4	E5	S6
S6	E6	S5	S1

Table 2: Connection List of ports of switches and connected nodes

As the transmitter send the packet that packet is sent through connected switch node. This connected switch node will forward packet and packet will reach its desired destination. After receiving packet, the edge node will send an acknowledgement to the transmitting edge node. When transmitting node receive an acknowledgement, it is ready to transmit. This process continues for maximum number of iterations. We have introduced the flag which will have value True and False. When flag is having true value, the process of switch is ongoing. Its process will be unaltered. When the flag holds false value, the switch is ready to forward the packet. By this means, the congestion in network will be less and this is an approach towards redundant network in turn network will become low traffic network.

Throughput of a network is calculated by the simple formula.

$$\text{Throughput} = \frac{\text{No. of successful transmitted packets}}{\text{Time Steps}}$$

We have calculated 9 throughputs for different iterations. Also, mentioned the maximum throughput in all scenarios.

III. Conclusions

The network topology which we implemented has six edge nodes and six switch nodes. We investigated after changing the probability of transmission, the throughput of the network changes. The throughput of a network is maximum when no. of iterations is maximum. In the end, stop-and-wait protocol is successfully implemented using Python programming language.

IV. References

[1] An introduction to computer networks
By PL Dordal

[2] Foundation of Python Network Programming
By John Goerzen