SFU Communication Networks

ENSC 427

SPRING 2011

FINAL PROJECT

EXPLORING TRAFFIC FOR P2P FILE SHARING PROTOCOL USING OPNET

GROUP 01

Abhishek Dubey – ada4@sfu.ca
Ashkan Mirnabavvi – amirnaba@sfu.ca
Vikas Yadav – vya3@sfu.ca

Abstract

P2P file sharing protocol is widely used by most torrent engines available for downloading data. Most torrent engines are based on the following phenomenon: more seeds (peers) there are, faster is the response time for the download for the client (end user). The whole idea of using P2P network is instead of downloading a file from a single source, the end user can download it from multiple sources. In this project, we will be exploring the performance based on the number of peers. We will be observing variables of interest such as Throughput and Link utilization on various networks.

Introduction

The reputation of file sharing peer-to-peer (P2P) networks became more prominent when Napster was invented in 1999. P2P networks have many advantages over the standard client-server networks. In fact, many experts predict that the standard client-server networks might be substituted by P2P networks in the near future. In a P2P network the client and server are essentially the same (the client itself is the server), while in a client server network, there is a dedicated server where all clients file requests are processed there. After each file request is processed, the correct file is then sent back to the requesting clients. As the number of client requests increases, the server can face a major dilemma of becoming overloaded, thus reducing the speed of downloads. Fortunately this problem is nonexistent in P2P networks, since as mentioned earlier the client and server are the same implying that each client has access to any type of data in the network and may set up, demand, as well as transmit this data. As a result, P2P networks have substantially lower operational costs compared to the standard client-server networks. Additionally, because of the bandwidth of data transmission in P2P networks, the likelihood of successful connection increases as more clients join and connect with the network. The Gnutella protocol (KaZaa, LimeWire) and the BitTorrent protocol (BitTorrent, Azureus) are the most popular P2P protocols on the internet. BitTorrent is a newer P2P protocol compared to Gnutella and the figure below gives a quick demonstration of how Peers and server are interconnected to each other.

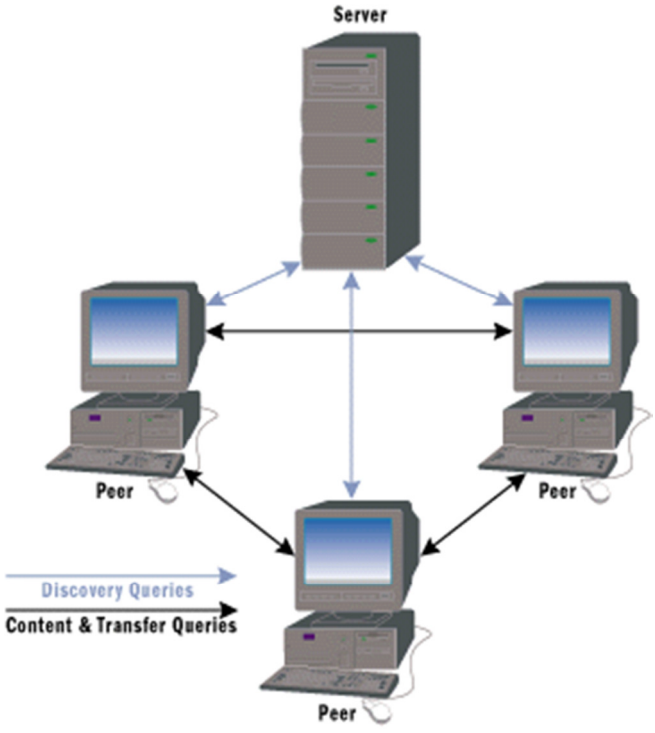

Figure 1 P2P Network

OPNET Simulation of P2P

The network topology we used to simulate the baseline scenario of P2P network is shown below. FTP Server, router, peers, seeds mainly constitute the network connected via duplex link. Seeds and peers together generate Bittorrent and FTP traffic in form of generating packets at a constant rate throughout the network. Towards the end we observe the throughput at the central router, i.e. number of packets that pass through the router. We also observe the link utilization between peer and router, and seed and router.

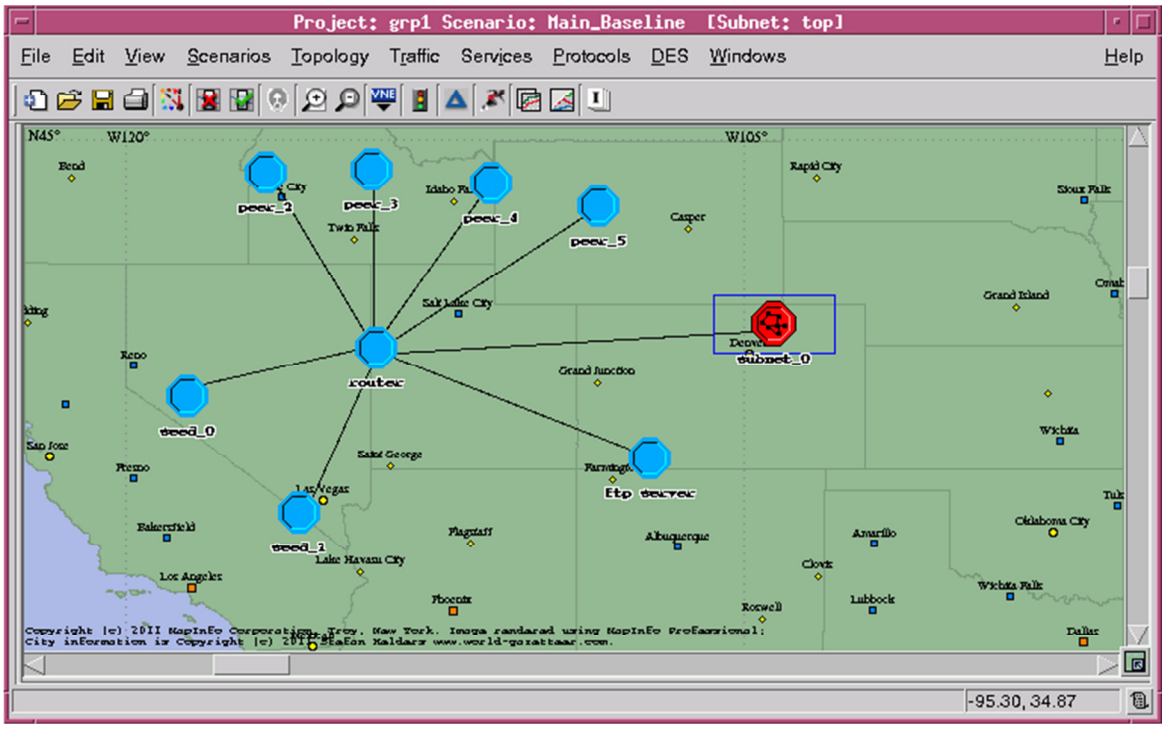

Figure 2 Network topology for baseline scenario for a P2P network

Our second scenario consists of the same topology, but with 6 peers instead of 8. This way we will be able to observe the differences in throughput and link utilization for both networks. We expect that the more the number of peers are, better throughput we will have.

Packet Formats

In this project we create four types of packets with two different formats: Request packets vs. Reply packets. Request packet format basically consists of 3 fields: protocol (2 bits), source address (4 bits), and destination address (4 bits). On the other hand, reply packet format consists of same three fields with an additional field of 32 bits for the data. We chose 2 bits for the protocol because we have four different types of packets and 2 bits are sufficient as 00,

01, 10, and 11. We keep the packet size of request packets to 24 bytes, whereas we set the size for the reply packets to 1024 bytes. We do this when we create node models and assign packet formats. Below are the two different packet formats:

Figure 3 Request packet format

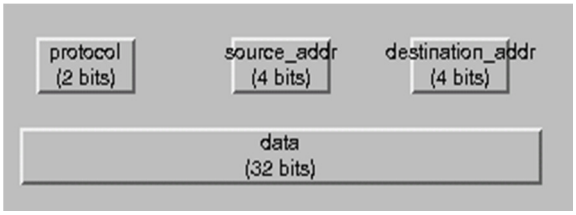

Figure 4 Reply packet format

The four types of packets are: Bittorrent request packets, FTP request packets, Bittorrent reply packets, and FTP reply packets. Bittorrent request packets are generated by peer nodes for seeds and other peers, while Bittorrent reply packets are generated by seeds to reply to peers. FTP Server request packets are generated by all nodes to request data from the server, while FTP reply packets are generated by FTP server in order to respond to requests generated by peers and seeds.

Node Models

Next we created four different types of node models for the OPNET simulation: Seed node model, Peer node model, Server node model, and Router node model. Each node model is created based on the description in the previous section. First, we created a node model for a seed as shown below:

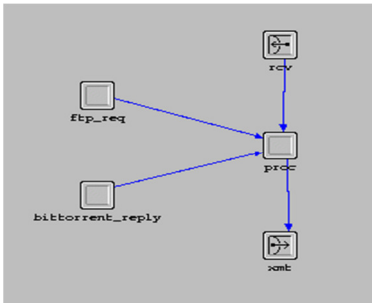

Figure 5 Seed node

We created a central processor whose process model we will define later in this project, which is connected by two simple sources ftp_req and bittorrent_reply, which basically generate FTP request packets and Bittorrent reply packets respectively.

Second node model we created was for a peer node, which performs both functions that a seed not would do. In addition, it also generates Bittorrent request packets for seeds and other peers.

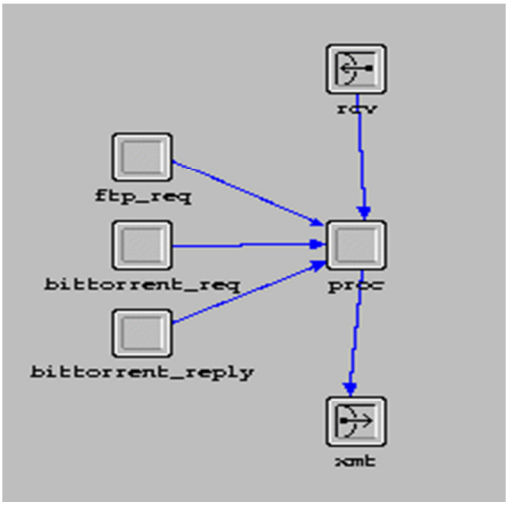

Figure 6 Peer node

We will define a process model later in this project for the central processor. Rest are all simple sources generating packets as their names describe. One key point to note is that we set different interarrival times for packet for seeds and peers to make it as close to a real life scenario as possible. Seeds generate packets at a faster rate than peers.

Third node model we created was for the server, which consists of a central processor, whose process model is described later in the project and a simple source, which generates FTP reply packets.

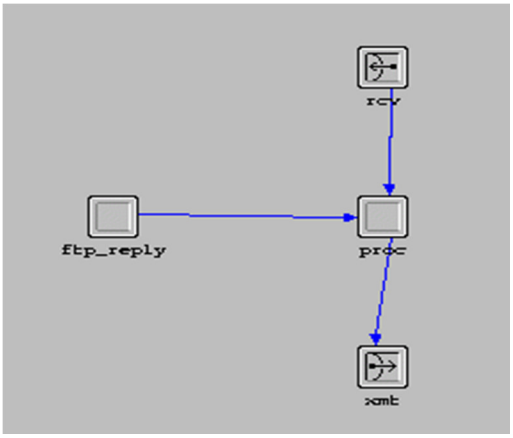

Figure 7 FTP server node

The final node model we created was for the router, whose main job is to send a packet to whatever destination address is assigned to the packet. It does not generate any packets of its own. This router basically interconnects all peers, seeds, and server with each other as they should be in a P2P network.

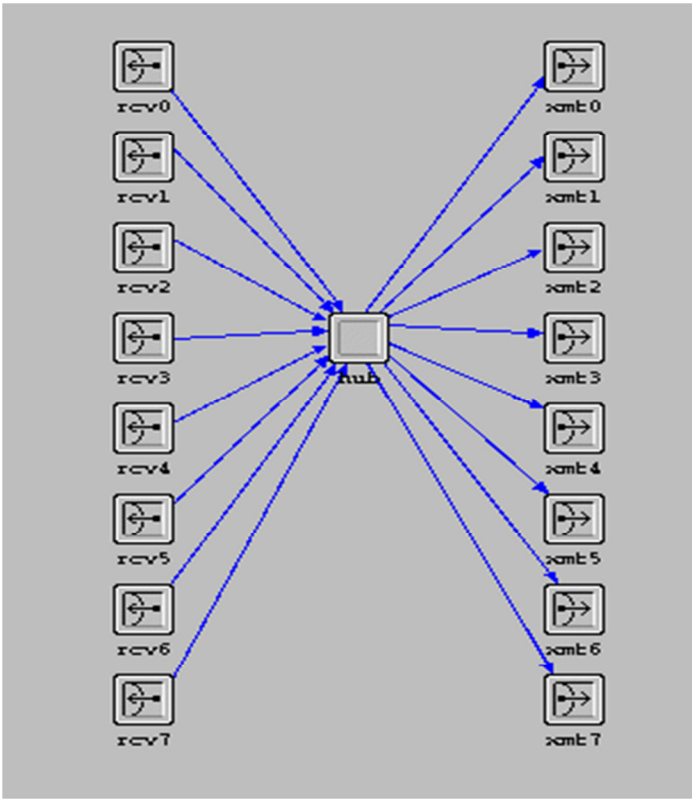

Figure 8 Router node model

Process Models

For all the above node models, we defined a process moder for the central processor, which defines what it should do when a packet arrives at the receiver end before it is send to the transmitter. The first process model we created was used by both seeds and peers as they share a common structure and perform almost the same functions. The model consists of an initial state and an idle state. There are 5 transitions for the idle state:

- Default: advised by the packet switching tutorial
- Source arrival: this state deals with the requests that come for FTP packets
- Source arrival 2: this state deals with the request that come for Bittorrent packets
- Source arrival 3: this state deals with Bittorrent reply packets
- Recive arrival: this deals with when packets are being received at the receiver end

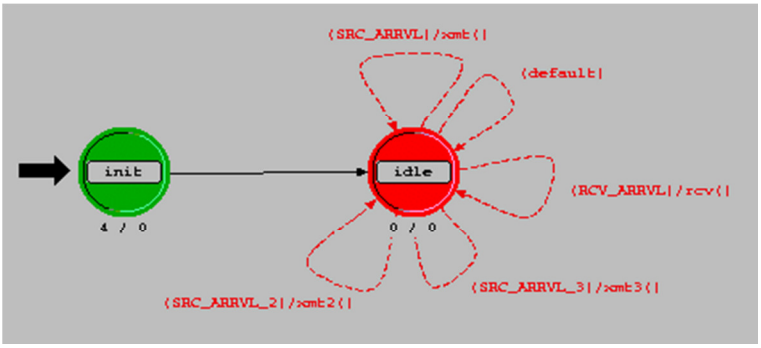

Figure 9 Process model used by seeds and peers

We defined the following state variables which would assign destination addresses based on the uniform distribution, which were defined in the enter execs for the initial state. The two images below demostrate both respectively:

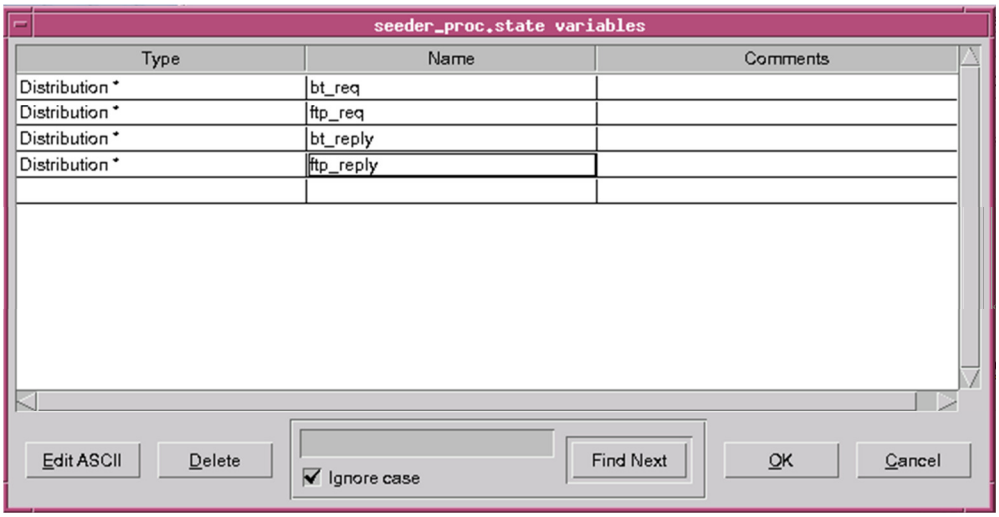

Figure 10 State variables

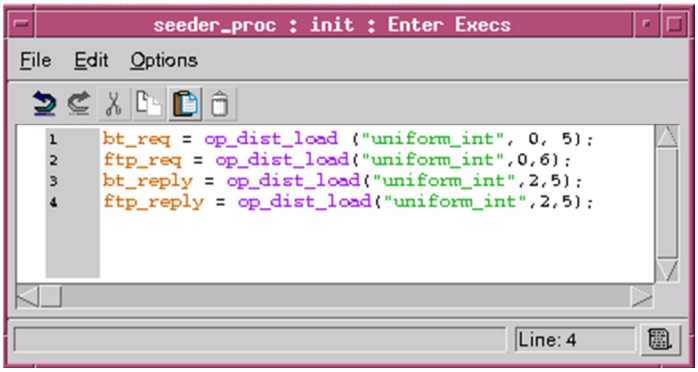

Figure 11 Enter execs for the initial state

Finally, in the functional block, we defined a function associated with each of the states. Because of the lengthy code, a screenshot is not attached here. Please refer to the project zip file for functional block code.

Next process model we defined was for the FTP server. This process model is quite simple and straightforward with an initial state and an idle state. Idle state has 3 transitions:

- Source arrival: for transmitting FTP packets
- Receive arrival: for incoming FTP requests
- Default: as advised by the packet switching tutorial in OPNET

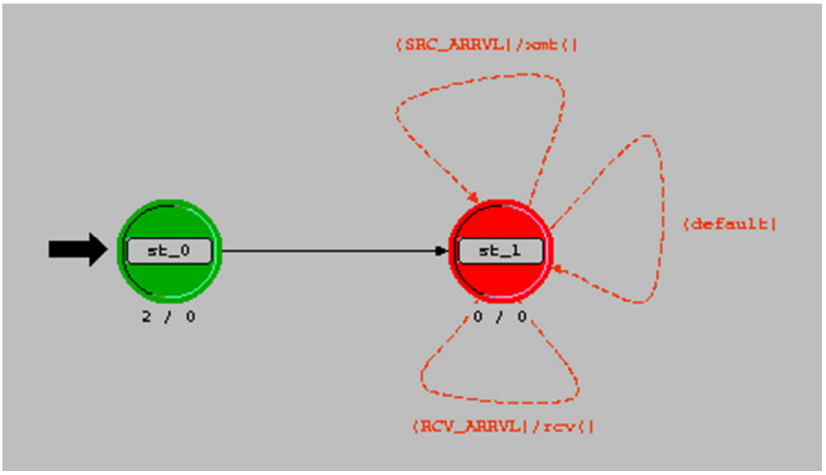

Figure 12 FTP Server process model to be used by server node

There is only one state variable in this code: ftp_reply. Detailed code for functional block may be found in the project files.

Lastly, we created a process model for our router, which is even simpler than the previous one. It only has two transition states: default and packet arrival because there is always some packet coming in at the router and router's job is to send it to whatever destination address is defined in the packet.

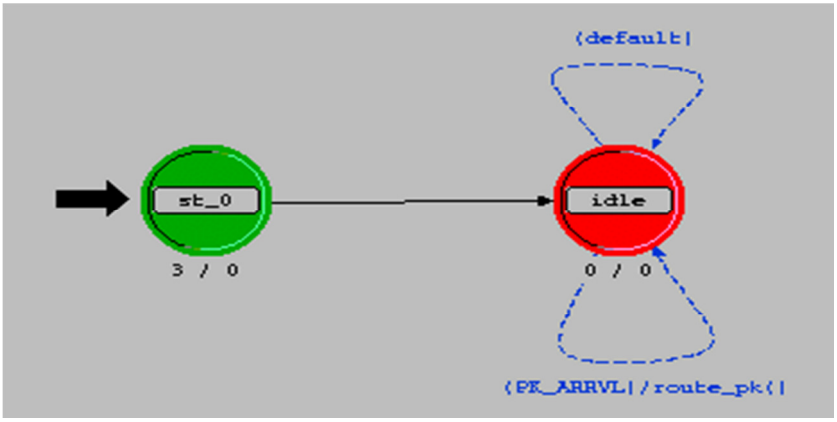

Figure 13 Router process model

There is a global statistic we defined in this process model, THROUGHPUT, which records the number of packets that go through the router. Enter execs for the the initial state are shown below:

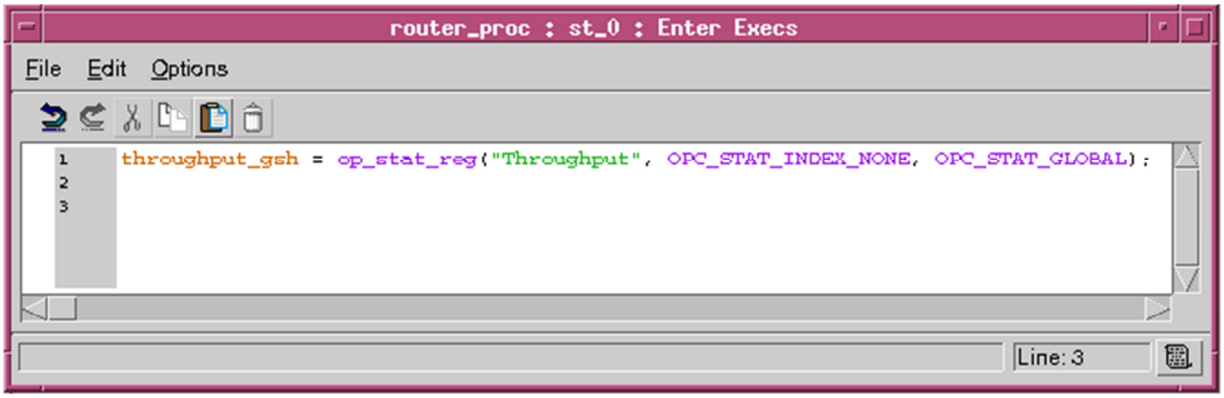

Figure 14 Enter execs for the initial state of router

Results & Discussion

Once all the node models and process models were in place, we built our network with 4 seeds, 8 peers, 2 routers, and 1 server, out of which we placed 2 seeds, 4 peers and one router in a subnet. We called this our baseline scenario as demonstrated below:

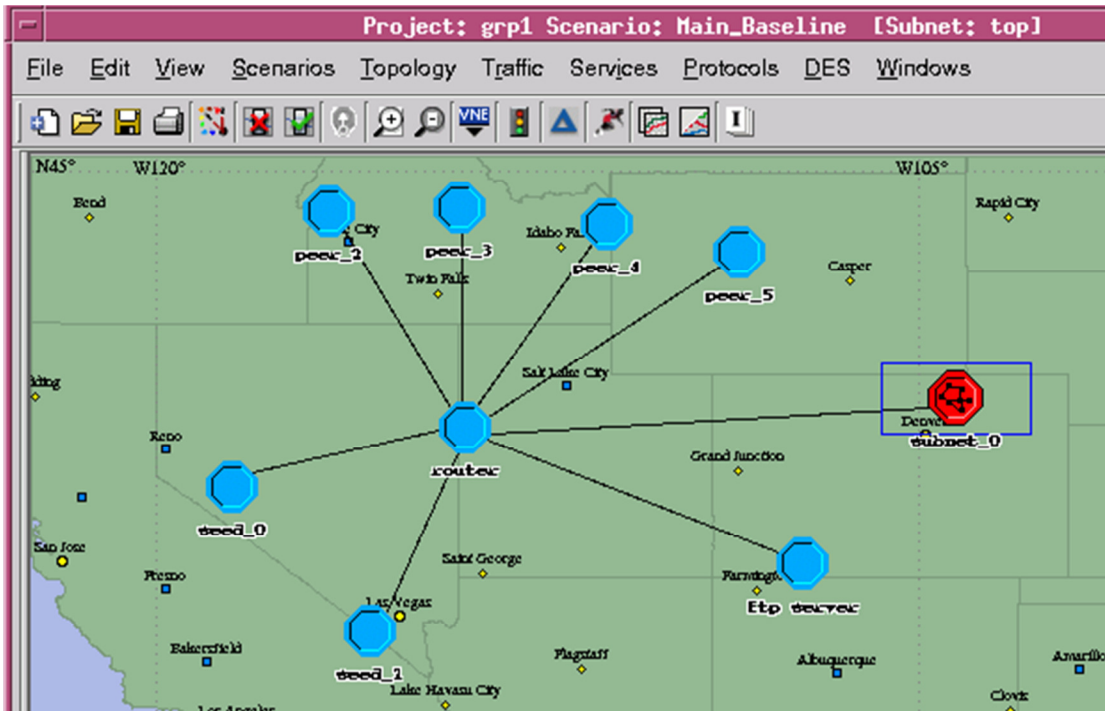

Figure 15 Baseline Scenario

We created one more scenario with 4 seeds, 6 peers, 2 routers and 1 server. The idea was to keep everything constant and only change the number of peers from 8 to 6 because we want to observer how peers help in a P2P network. The network topology for this scenario is shown below:

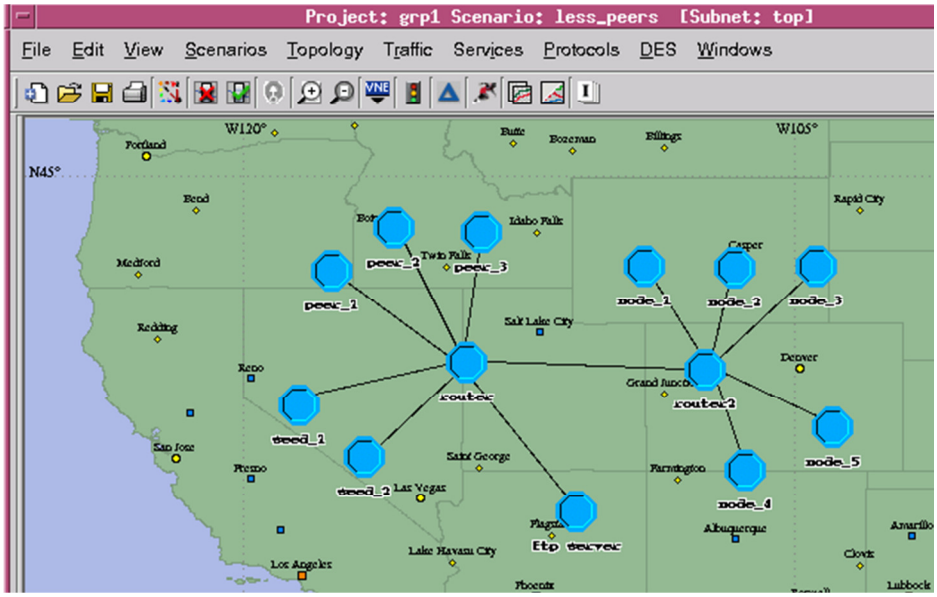

Figure 16 Less peers scenario

We simulate both network for 1000 seconds (approx 16.5 mins) and observed the throughput at the router and link utilization between 'seed and router' and 'peer and router'.

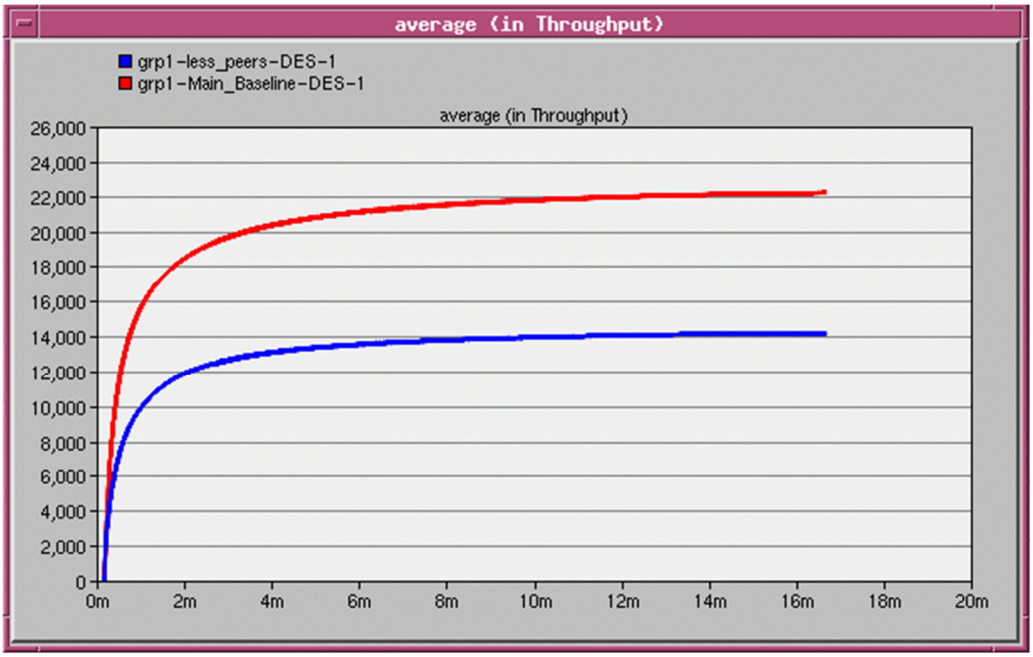

Figure 17 Average throughput for both scenarios

As we can see that the throughput is higher for baseline scenario and rightly so. Because in baseline scenario there are more peers and hence more number of packets are being generated before it stabilizes.

Next we look at the link utilization between peer # 3 and router:

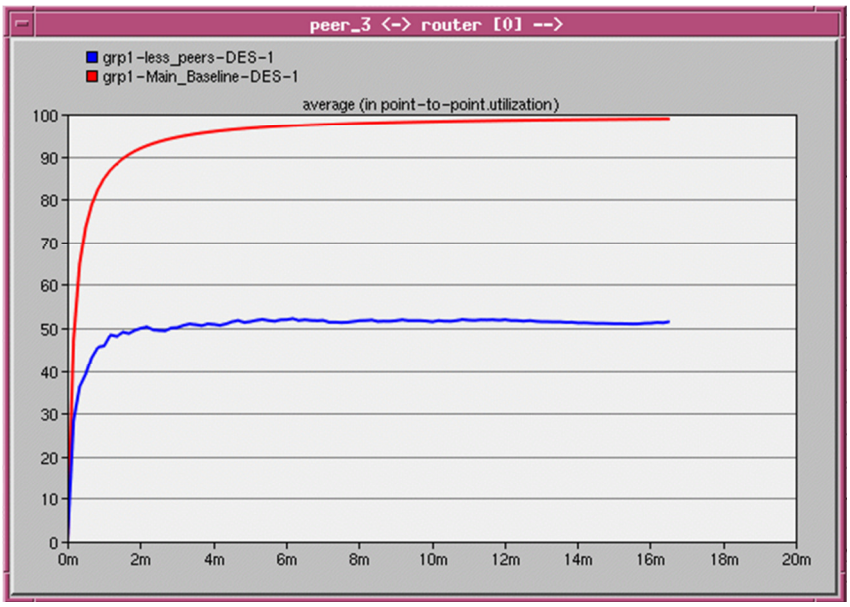

Figure 18 Link utilization between a peer and router

We observe that the link utilization is higher for baseline scenario, which is what we expected. Since baseline scenario has more peers, each peer will have to help other peers by generating more packets until they are stabilize, which is precisely why we see high link utilization for the baseline scenario.

Next we look at the link utilization between a seed and router:

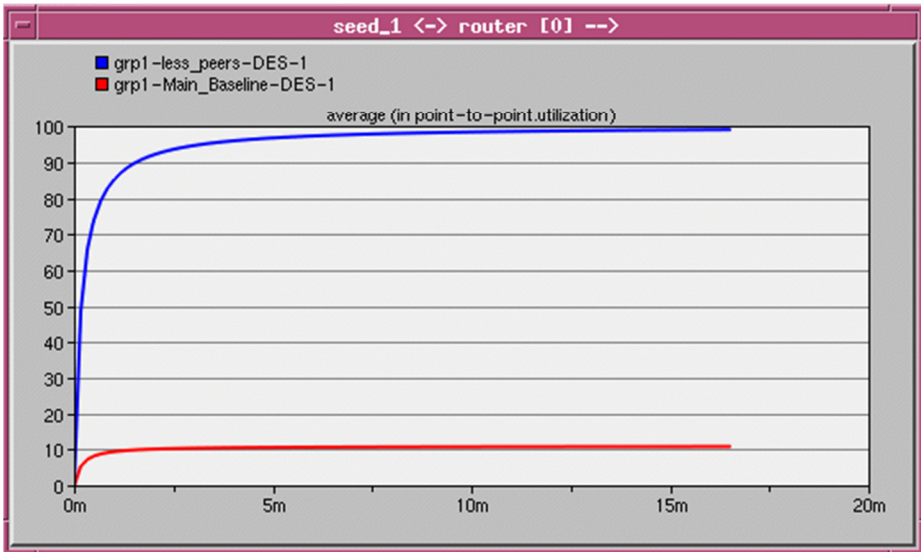

Figure 19 Link utilization between a seed and router

We see that the graphs are reversed here, i.e. the link utilization for baseline scenario is lower. This is what we expected, because if there is less number of peers, the other peers will rely more on the seeds for their packets, which causes the link utilization between seed and router to increase.

Few problems we encountered during this project were:

- Unable to use the basic sink model in the OPNET library because of some disruptions caused in the lab. We overcame this difficulty by referring to the packet switching tutorial, which later provided us the basic framework for the whole project.
- Once the whole network was build, it was fairly time consuming to debug the code and see where a couple of errors were resulting, mostly the "out of range" error. Later we found that this error taking place in the function block of the process model for seeds and peers. We were missing and else statement, which wasn't taking care of the third protocol, which was meant for the FTP server.
- We ran into some issue of running OPNET remotely with two of our accounts. It would time out after 20-30 minutes or so. Luckily for one of our team members, it wasn't timing out, so we did the whole project through his account.

Conclusion

Overall, we observed the importance of number of peers in a P2P network. We observed various results like Throughput and Link utilization to get to a conclusion that more the number of peers are, better throughput we will get in the network and as a result faster download time in real life. In future, we can improve this project by adding more variety to the ways the packets were being generated. In this case we had different interarrival times for seeds and peers so that seeds generate packets at a faster rate. In future, we can have different types of seeds and peers with different interarrival times as there would be in real life.

References

[1] "BitTorrent Inc. Open Sources New P2P Protocol - The H Open Source: News and Features." The H: Security News and Open Source Developments. Web. 13 Feb. 2011. <http://www.h-online.com/open/news/item/BitTorrent-Inc-open-sources-new-P2P-protocol-1007814.html>.

[2] "BitTorrent Protocol Encryption." Wikipedia, the Free Encyclopedia. Web. 13 Feb. 2011. <http://en.wikipedia.org/wiki/BitTorrent_protocol_encryption>.

[3] Katzela, Irene. Modeling and Simulating Communication Networks: a Hands-on Approach Using OPNET. Upper Saddle River, NJ: Prentice Hall, 1999. Print.

[4] Leuf, Bo. Peer to Peer: Collaboration and Sharing over the Internet. Boston: Addison-Wesley, 2002. Print.

[5] "Peer-to-peer." Wikipedia, the Free Encyclopedia. Web. 13 Feb. 2011. <http://en.wikipedia.org/wiki/Peer-to-peer>.