ENSC 427: COMMUNICATION NETWORKS

The Effects of Network Awareness in P2P Protocols on Network and Protocol Performance

Spring 2009

FINAL PROJECT

Cristian Panaitiu (cap5@sfu.ca)
Steven Verner (sverner@sfu.ca)

http://cris.panaitiu.googlepages.com/ensc427project

Abstract

In recent years, P2P applications have become increasingly popular among the new generation of web-savvy users. Applications like Kazza and Bit torrent are being used more and more to exchange large amounts of data.

With the increase in data transfer comes increased network congestion; in turn, this reveals that a fundamental weakness of the current generation of P2P: peer search algorithms. P2P protocols like Bit torrent search for data by locating a peer that is geographically close, or the fastest peer etc... These algorithms, on the other hand do not yet have any criteria for topological closeness of the destination peer and the geographically close peer is not necessarily the fewest number of router hops away.

The ensuing congestion from P2P networks has led many ISPs to throttle P2P traffic to be able to provide higher quality of service (QoS) to regular web users at the expense of P2P users. Thus we propose implementing network awareness in the P2P protocol itself for a twofold benefit; the P2P downloads will be faster, and the QoS to regular web users will be maintained.

Introduction

Increasingly popular P2P protocols such as Bit torrent and Limewire are virtual networks that function over existing internet connections. These protocols are also responsible for up to 70% of internet traffic in terms of bytes [3]. Thus, the inefficiency in the peer search protocol causes many large pieces of data to be transferred over unnecessarily many router hops; in particular, the backbone links will tend to become flooded with P2P traffic unnecessarily being sent over many hops through the internet. This situation causes the ISPs to throttle P2P traffic in an attempt to avoid over-congesting their networks and to provide acceptable QoS to web (non-P2P) users.

P2P users can take a number of steps to avoid ISP throttling, such as encrypting packet headers etc… but implementing network awareness is a better solution that provides a twofold benefit to the P2P user. First, by requesting desired packets from peers that are as close as possible on the network, the transferred packets take fewer router hops, increasing performance. Second, decreasing the number of packet requests sent over long distances on the network decreases the P2P traffic on busy international connections; thus, in the situation where the packet cannot be found nearby, a user can request it from far away and the chance of the packet being dropped is low anyways, as less backbone link bandwidth will be used by P2P traffic. In other words, long distance queries for a file will only happen to seed the local networks with that file for other users on the local network, solving the problem of backbone link congestion.

Current Research

Current research is currently being conducted in the effect of network awareness on the performance of the network. [1] and [2] propose geographic awareness as a way of decreasing congestion on the network, however, this does not take into account that topological closeness to another node is seldom related to geographic closeness. Instead, [6] suggests that a centralized, ISP side system can provide congestion parameters to P2P

users. This enables the P2P client to make least cost connection choices that help the ISP, as well as the users and avoids the need for throttling and associated reactions such as packet encryption [7].

Approach

The approach taken in this project towards determining whether there exists benefit in implementing network awareness is indirect; a network with P2P, web users and web servers will be set up and each user will be allowed to send requests and receive packets. The packet traffic on the network will be analyzed and key parameters will be collected as follows to analyze the network load and QoS the users receive.

1) First, links will be investigated; it will be determined whether backbone link congestion is significantly alleviated by ISP throttling. The cases of a main backbone link, as well as a lightly used backbone link will be compared with and without throttling to see if there is significant reduction in the time for which the link throughput is 100%. The significance of this result is evident in that at times when a link is fully used, packets have to queue, thus delaying delivery and lowering QoS.

2) The most important parameters related to the performance of P2P download are the end to end (E2E) delay and P2P packet distance traveled; the higher these values, the longer the time taken to download a file by the end user. Packet loss for P2P will also be examined as another important indicator of the QoS P2P users receive with and without throttling. In reality, cumulative E2E delay will be examined.

3) Finally, the QoS for the web user will be measured by similar parameters as the P2P user: E2E delay and average throughput.

The average packet throughput and E2E delay will be measured for individual types of users; packet loss is only measured as an average for P2P users. Backbone link utilization will be measured for one busy link (with many incoming P2P connections) and one relatively unused link.

Expected Results

Link utilization is expected to drop significantly with the introduction of throttling for very busy links. Busy links, which carry very many P2P packets, will experience the greatest benefit from throttling, because most of the throttled packets will likely be those that would have passed through these links. For the purpose of this project, local links are made just as fast as backbone links because they are assumed to be relatively cheap to provision and the entire purpose of throttling is to prevent the constant congestion of backbone links with P2P packets.

P2P traffic is expected to experience a noticeable increase in packet loss as throttling is enabled. Overall, traffic is expected to have shorter E2E delay on account of

all the dropped P2P packets. Throughput of the web users is expected to increase due to the lower congestion on the links.

Implementation

The Network

Implementation of the network is shown in figure 1. Each blue node on the network is an ISP, handling international packet routing. Each red node is a local subnet, shown in figure 2; these subnets are star topologies around a central hub, each containing some P2P users, some web users and some web servers. The intention behind this model is that each user connects to a corresponding user and requests a packet. The packet will be sent to the requester, thus simulating traffic flow on a network. The network will be modeled and simulated with opnet 14.0 and a throttling algorithm will be developed to dynamically determine P2P packet traffic quotas for the entire network. The topology of this network is

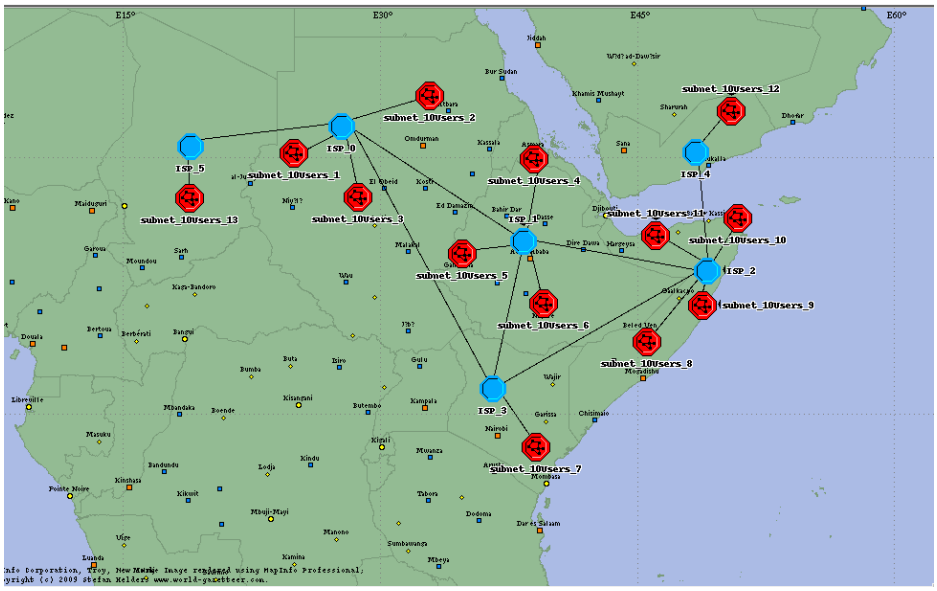


Figure 1: The top layer view of the network: Red nodes are star topologies with P2P, web clients and web servers connected to a simple hub.
Blue nodes represent the ISPs that handle traffic control.

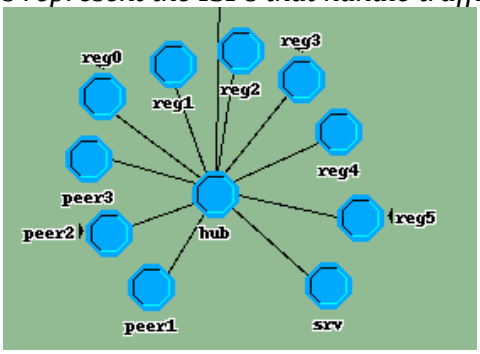


Figure 2: Nodes making up each subnet include mostly regular users, some P2P users and a few web servers, all connected to a central hub that is in turn connected to the ISP hubs

ISPs

The blue nodes in figure 1 illustrate the ISP hubs that connect together to form the backbone of the simulated network. Each ISP hub has a unique ISP ID and a fairly complex routing table to know towards which port a packet is destined. The throttling is also implemented in the ISP hubs (not the smaller subnet hubs). Node model for the ISP router is shown in figure 3, with the corresponding process model. There is a slight difference in that each of the ISP node models differ in the number of input and output ports. Their process models also

Regional Subnets

The Regional Subnets consist of some P2P, some web server and some web client users, each of which is generating packets according to a pre-determined behavior. The generated traffic may be sent to another subnet, or may be destined to a user in the same subnet. The regional subnets use links that are of a very high throughput capability relative to the traffic flowing through them. The logic behind this is that throttling effects on backbone links are being investigated, not the effect on local subnet links. Local links are cheaper to provision anyways than backbone ones so minimizing their cost is lower priority than minimizing the cost of backbone connections.

Routing Implementation

The implementation of routing was done by hand. Each subnet is assigned a unique subnet ID and each node on these subnets has a unique destination ID (discussed in packet formats) and a protocol ID. The subnet ID is used by the routing tables to find the correct router, and the destination address is used to locate the destination node in the subnet. This way, packets are routed continuously. Essentially, the routing tables are ‘if’ statements in the process model of the hubs. Appendix A contains details of the code implementing these routing tables.

User Types and Behavior

There were three different types of users modeled with this project: web user, P2P user and web server. The behavior of each one is relatively easy to understand; the P2P user sends out request packets and P2P packets with constant interarrival times, whereas those sent by the web clients and servers had exponentially distributed interarrival times.

Figure 5 shows the typical node model and process model used for all three types of users - both being similar to those in the packet switching tutorials, but with modified code and attributes. The node model for all three users looks identical, with the only difference being the attributes of the source, sending different types of packets, and the distribution of the interarrival time. The process model is also very similar to that of packet switch tutorial except that it keeps track of the global variable for counting packets sent.

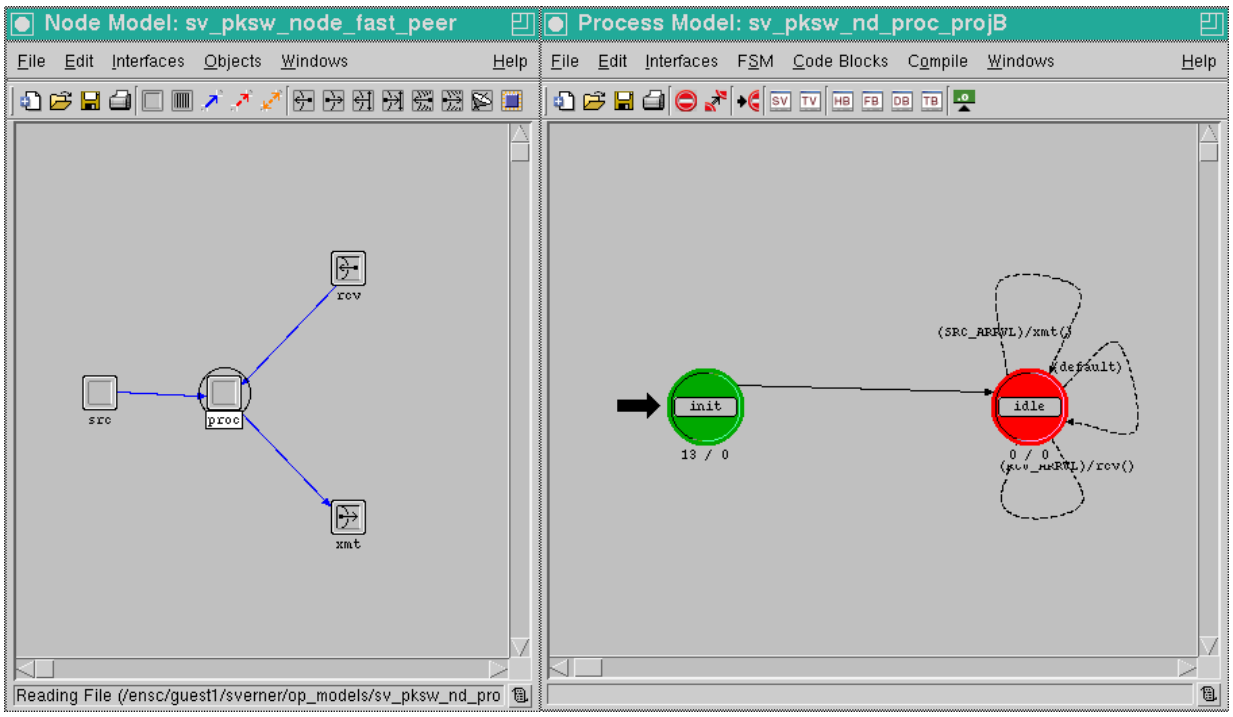


Figure 5: the node model (left) and process model (right) of the three different types of users: web, server and P2P.

Packet Formats

The packets being sent were of different types: namely P2P packets, web (HTTP) packets and request packets. Each had a destination address (16 bits), a destination subnet address (4 bits) followed by an origin ID (4 bits), which specifies the origin subnet ID.

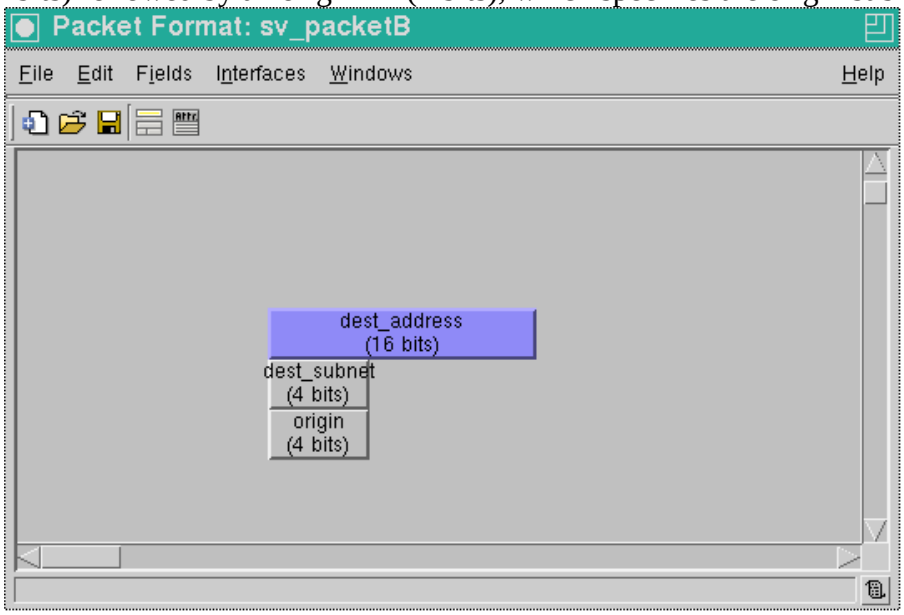


Figure : The packet format used for each of the three user types.

Implementation of Request and Data packets

The packets sent and received in this simulation were of a random nature in that a request packet would be sent to an appropriate destination node by either a P2P or web client user. The users would also send random P2P and web packets respectively, without taking into account from which destination the request came from. While not a realistic model, this nonetheless serves to send out packets into the network with the same rate as would be expected from a P2P and web network.

Implementation of Throttling Algorithms

Throttling was the most important achievement of this project; the difference in traffic parameters for the throttled and un-throttled scenarios were the subject of intense scrutiny. In this project, throttling was implemented in terms of a bandwidth quota; namely, 30% (number that represents the percentage of P2P users in the network) of all packets being generated could be P2P packets, otherwise they would be destroyed upon being received at the router. If there was bandwidth available, then the quota for P2P bandwidth would be made higher. The way this is implemented is that the three second moving average of traffic coming out of an ISP router is less than around two thirds of the total outgoing link capacity, then P2P packets will be routed until the link becomes too congested.

See conclusion for discussion of possible improvements and future work

Results

Backbone Link Utilization - Lightly used link

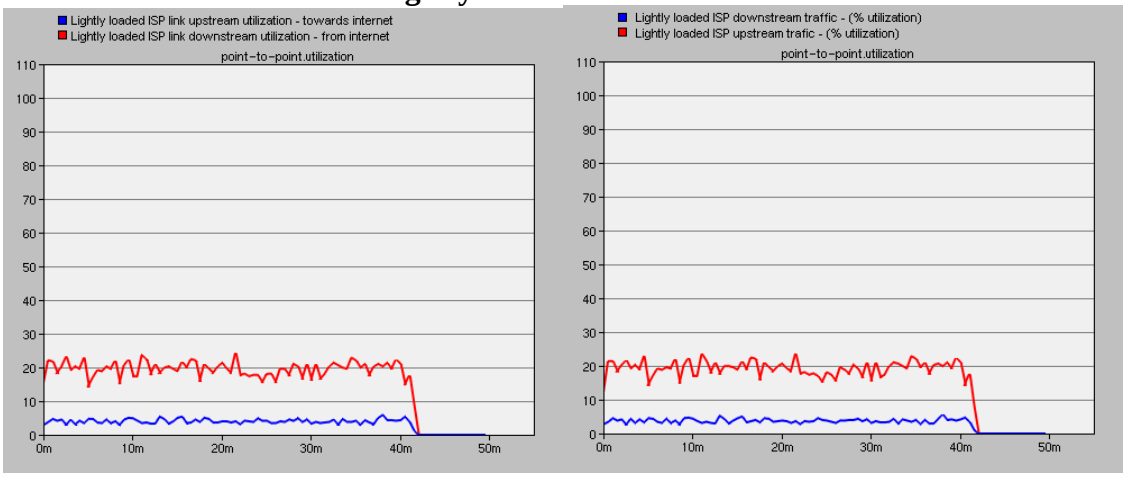


Figure : Upstream and downstream link utilization for a lightly loaded ISP with no throttling (left) and throttling (right). Notice that there is little difference in the traffic pattern because the P2P traffic is allowed to occupy the empty bandwidth in the link as per the bandwidth allocation algorithm.
The top (red) graph is the outgoing internet traffic, whereas the blue one is the incoming internet traffic.

Backbone Link Utilization - Heavily used link

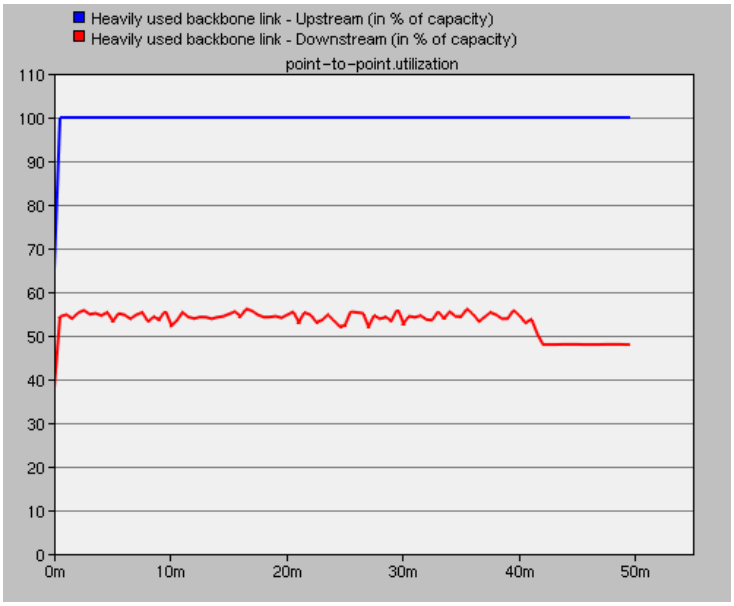


Figure : Here, the same case can be observed as below; the heavily used network has more upstream than downstream. This is the case without throttling, in which theP2P traffic is dropped, as is the web traffic

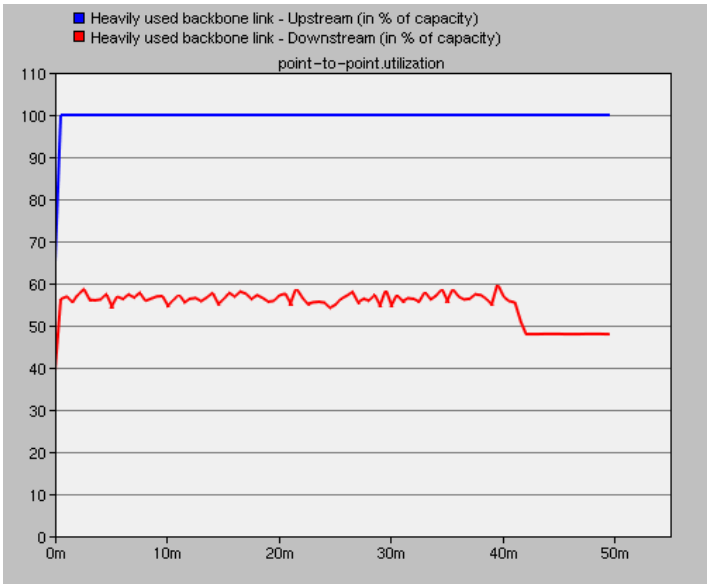


Figure : The heavily used link from ISP router 1 to 0 sends many more packets than it receives. This is the scenario without throttling, in which the web traffic occupies all of the outgoing (blue) stream

End to End delay

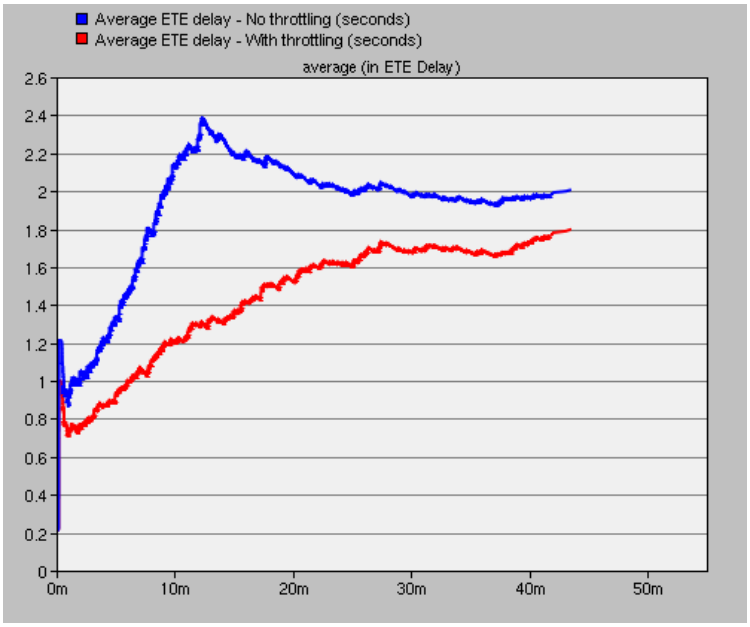


Figure : The ETE delay is decreased by implementing network awareness: packets traverse fewer hops and wait less in queues, thus making the delay from end to end of their route shorter overall.

Peer Packet Loss

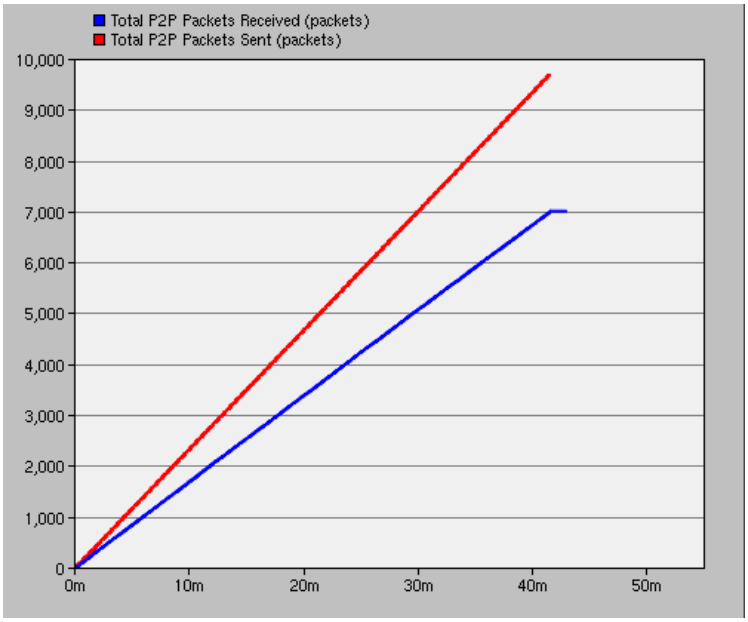


Figure : In the case of no throttling, the router buffers of 1000 packets at each input port overflow under the massive burden of the P2P packet streams. Loss is estimated at just under 30%.

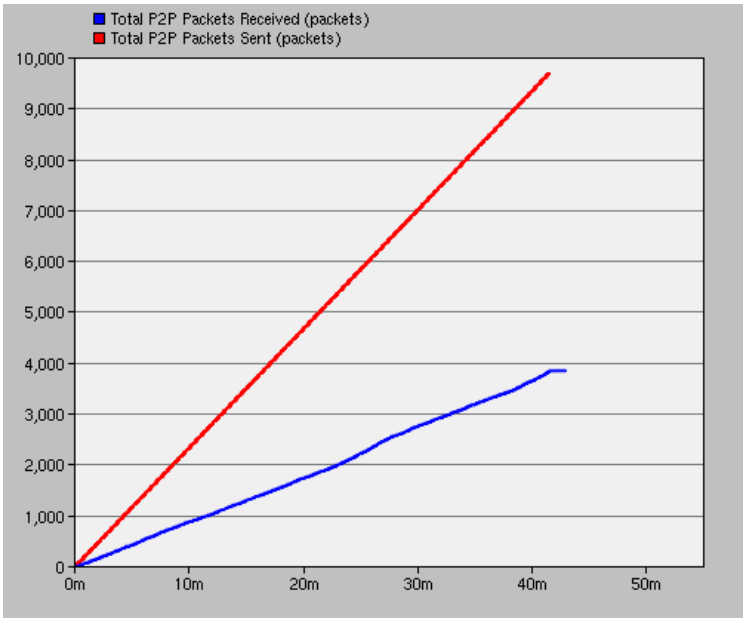


Figure : In the case of throttled internet, the router will drop many of the packets, leading to a loss figure of around 60%

Packet distance traveled

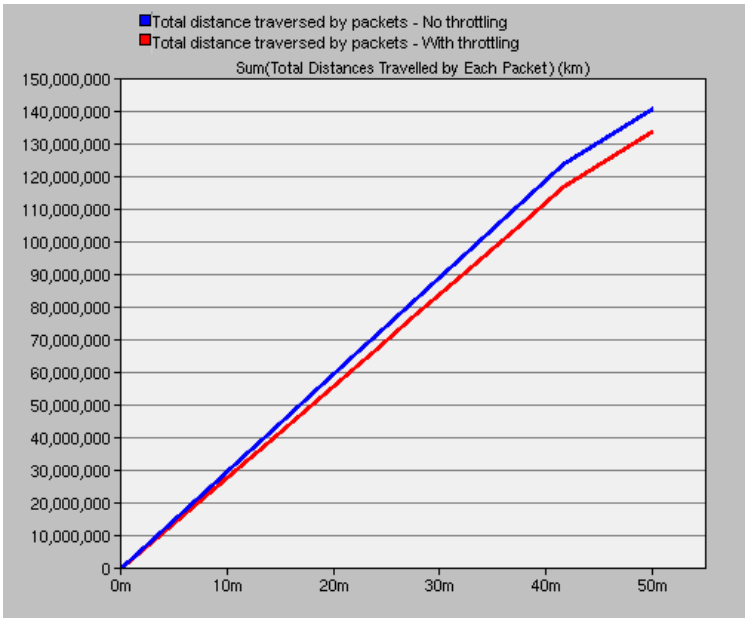


Figure : The total distance traveled by all of the packets on the network noticeably decreases with the introduction of throttling (red line). The change is around 5%.

Regular web user average success packet delivery

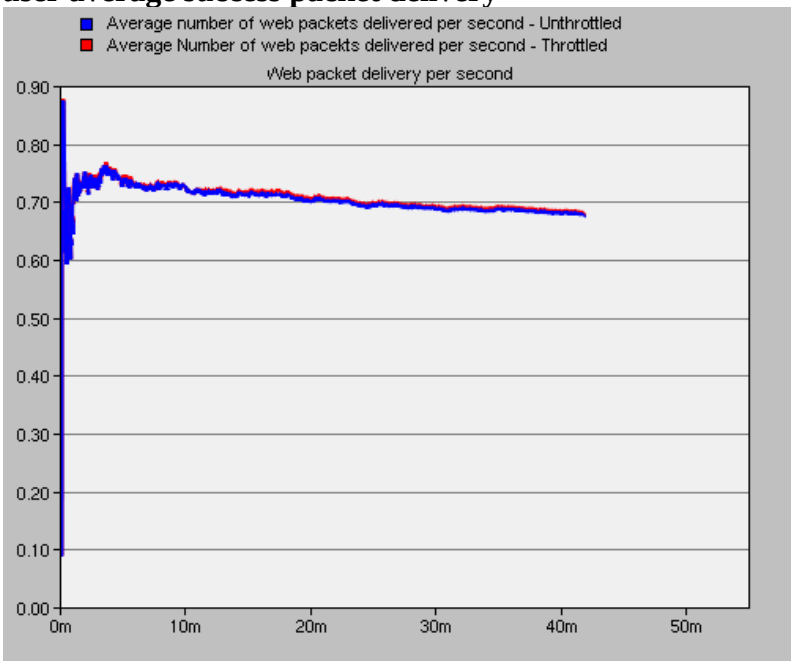


Figure : Average packets delivered to web users per second increase only slightly with throttling because very few web packets are dropped with or without throttling due to large buffer and limited congestion.

Discussion and Conclusion

This project chose to attack the problem of Internet Service Providers (ISPs) throttling of peer-to-peer (P2P) traffic. The topic is currently a hot topic with ISPs facing legal action over the practice [8] as well as coping with increasingly congested internet infrastructure due to P2P traffic. The topic has become even more relevant to recent research as P2P traffic is routed inefficiently over networks. Packets can be downloaded from any of a great number of other peers, however current P2P protocols do not attempt to download from the topologically closest peer (through network awareness). This practice could drastically decrease internet congestion, increasing performance for web users and decreasing costs for ISPs. For this reason we chose to investigate network the impact of network awareness on P2P.

This project had several components. First step was to model a network with peer-to-peer users, normal web users and web site server nodes. The network was built mainly from scratch with some inspiration taken from tutorials. We also continuously worked on the model to make it more realistic. The model provided an approximation to the packet traffic expected in such a network which worked well for our simulation, but the model could have be improved in several ways in the future:

- The users do not distinguish between packets and their content; any packet received from a fellow peer or from a server is a good packet
- Routing had room for improvement by implementing congested route sensing and avoidance. A very complex project would warrant such effort.
- The user models could have included more detailed implementation of the protocol and behavior (leeches, heavy P2P, light P2P etc…)
- More diverse distribution and size of subnets and user activity.

The second and most important aspect of this project was designing and implementing our throttling/network awareness scheme on our network model. This required extensive effort seen in the ISP Router nodes as this was a unique approach which we didn't see in any of our sources, so it was necessary to design and code our ISP routers from scratch. The ISP Routers kept track of what type of packet was being routed, congestion over each link and the percentage of P2P packets passed over each link. Throttling was implemented when the link wasn't congested, when the packet was P2P, and when the percentage of P2P traffic of that link was greater than 30%. Several improvements could be made to our solution including:

- Throttling could be implemented at different percentages P2P over different links (experimentation could lead to a more optimal solution)
- Throttling could be implemented such that P2P packets are transferred more easily to central subnets (in the case where packets are differentiated from each other) to decrease average transmission distance
- Congestion detection over links could have been improved as this wasn't implemented perfectly
- Other approaches to network and location awareness could be investigated such as altering the user selected protocol

The results were somewhat surprising; throttling decreased E2E delay and distance traveled for the packets, but it caused no perceived benefit to the normal web traffic. Link congestion could also not be solved by making the links faster. Throttling also increased the P2P packet loss to unacceptable levels.

The congestion could be explained by poor provisioning of the network at the planning stage. In reality, the congestion is really redistributed from bandwidth allocated to P2P to bandwidth allocated to web users; the throttling algorithm makes sure of this.

In the end, the main objective of this endeavor is the maximization of user QoS such that the user can download content faster. Average distance traveled by P2P packets was decreased leading to an overall decrease in E2E delay. The E2E delay of packets is the most important parameter concerning download speed and has improved by up to 20%. In overview, it can be seen that by adopting network awareness, the P2P user can gain a two fold benefit; namely that he will download from closer peers and that will not be subject to throttling efforts.

References

1. "Study of the location awareness in bit torrent like networks"
   URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4127056
   Location awareness as solution to P2P peer search inefficiency

2. "BitTorrent Location-aware Protocol 1.0 Specification"
   URL: http://wiki.theory.org/BitTorrent_Location-aware_Protocol_1.0_Specification
   Specification for a location aware BT protocol

3. "Impact of P2P traffic to the IP communication network performances"
   URL: http://www.sparc.uni-mb.si/OPNET/PDF/ImpactOfP2P.pdf
   Paper examining the effect of P2P applications on IP network

4. "L-CAN: Locality aware structured overlay for P2P live streaming"
   URL: http://150.140.187.130/getfile.php?fid=36
   Paper outlining the effect of location awareness in BT protocol

5. "The BitTorrent Protocol Specification"
   URL: http://bittorrent.org/beps/bep_0003.html
   Specification of bit torrent protocol by Bram Cohen

6. "P4P: Explicit Communication for Cooperative Control Between P2P and Network Providers"
   URL: http://www.dcia.info/documents/P4P_Overview.pdf
   Centralized network routing algorithm proposed to introduce network awareness to P2P applications by providing congestion information. This information is

provided by the ISP because the network is difficult to probe by individual ISP users.

7. “Throttling P2P traffic is short-sighted, experts say”: IT Business news article URL: www.itbusiness.ca/it/client/en/home/News.asp?id=49626&PageMem=1 Article referring to [6] as a solution to legal problems faced by Bell over packet throttling practices

8. “Internet Multicast Backbone Topology Image” by Elan Amir of UC Berkley URL: http://a.parsons.edu/~limam240/visualcomplexity/images/116_big03.jpg

9. Sizing Router Buffers” by G. Appenzeller, I Keslassy, and N. McKeown URL: http://yuba.stanford.edu/~nickm/papers/sigcomm2004.pdf

Further Acknowledgements

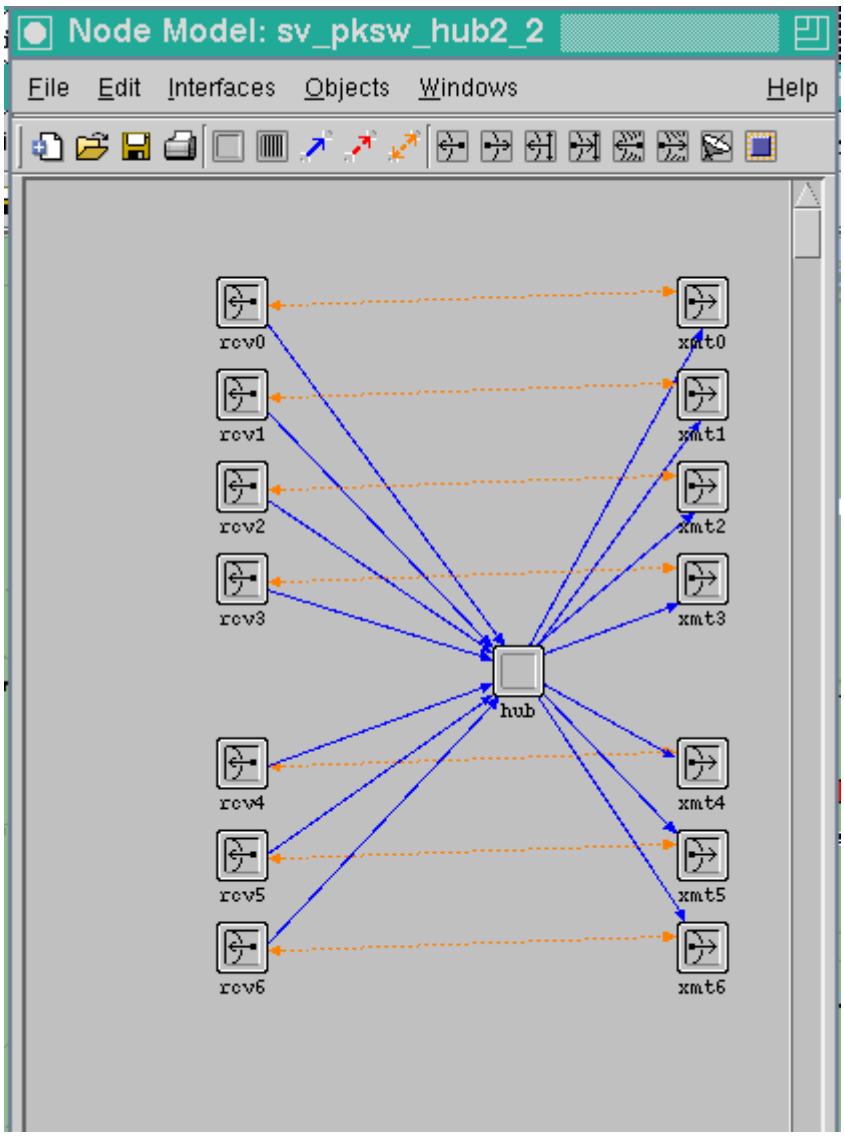
The node , packet formats s-and process models, as well as associated code for the hubs and the users were inspired from the two packet switching tutorials from opnet.

The topology of this network was inspired partially by lecture materials presented in ensc 427 in spring of 2009 and partially from online examples of ISP topology such as [8].

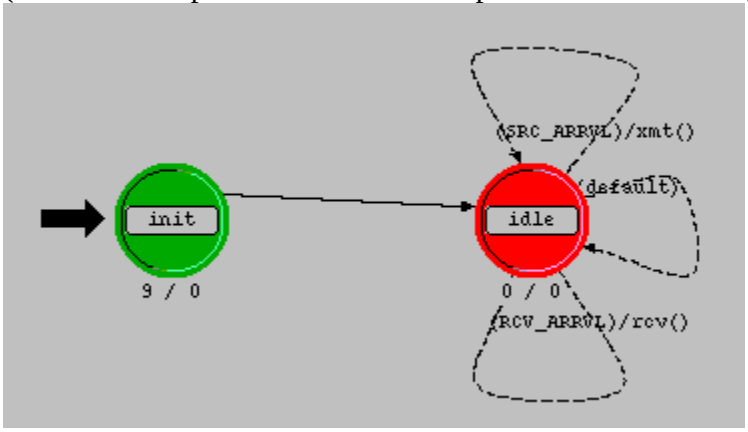
Appendix A:

ISP Router Node:

Node Model:
(This is ISP_2, a template for all ISP hubs, the number of transmitter/receiver pairs is varied)
(based on the hub node from Opnet Packet Switching Tutorial 2)


Node Model: sv_pksw_hub2_2
File
Edit
Interfaces
Objects
Windows
Help
rcv0
rcv1
rcv2
rcv3
rcv4
rcv5
rcv6
hub
xmt0
xmt1
xmt2
xmt3
xmt4
xmt5
xmt6

ISP Router Proc Process Model:
(based on hub process model from Opnet Packet Switching Tutorial 2)



ISP Router State Variables:
(same for ISP 0-5)

sv_pksw_hub2_proc_2.state variables

Type	Name	Comments
int	subnet_id	
Stathandle	distance_packet_speed	
double	total_norm_sent	
double	total_peer_sent	
int	total_sent1	
int	total_sent2	
int	total_sent3	
int	time_slot	

Edit ASCII | Delete | Ignore case | Find Next | OK | Cancel

ISP Router Global Interfaces:
(Same for ISP 0-5)

Declare Global Statistics: sv_pksw_hub2_proc_2

Stat Name	Mode	Count	Description	Group	Capture Mode	Draw Style	Low Bound	High Bound
Sum(Total Distances Travelled by Each Packet) (km)	Single	N/A					0.0	disabled

ISP_5 Router Header Block:
(Code is written from scratch)

```
#define PK_ARRVL (op_intrpt_type () == OPC_INTRPT_STRM)
#define OTHER_SUBNET_STRM 10
double total_distance_sent = 0;        //initializes global variable
int filter_flag = 0;                   //initializes globabl variable (turns filtering on/off)
```

ISP_0 through ISP_4 Router Init Enter Execs:
(Code is written from scratch)

```
#define PK_ARRVL (op_intrpt_type () == OPC_INTRPT_STRM)
#define OTHER_SUBNET_STRM 10
extern double total_distance_sent = 0;        //refers to global variable (total distance sent
for all packets)
extern int filter_flag = 0;                   //refers to global variable (filtering on/off)
```

ISP_2 Function Block:
(Code is written from scratch)

(representative of all ISP function blocks. The only difference is the routing tables, distance tracking tables and origin-to-subnet distance tracking tables are different for each ISP Router. See comments in code)

```
static void route_pk(void)
{

double distance_packet_speed_temp;
int dest_subnet;
int dest_address;
int origin_subnet;
Packet *pkptr;
FIN (route_pk());

pkptr = op_pk_get (op_intrpt_strm ());
op_pk_nfd_get_int32 (pkptr, "dest_subnet", &dest_subnet);
op_pk_nfd_get_int32 (pkptr, "origin", &origin_subnet);
op_pk_nfd_get_int32 (pkptr, "dest_address", &dest_address);

if((int)op_sim_time()%3 < 1)   //keeps track of the bits transmitted over each route in the
last 3 seconds
//this is used to determine if the route is congested and should throttle p2p packets or if
the route is not
//congested in which case all p2p packets should be allowed to pass through unthrottled
{
if(time_slot !=1)
```

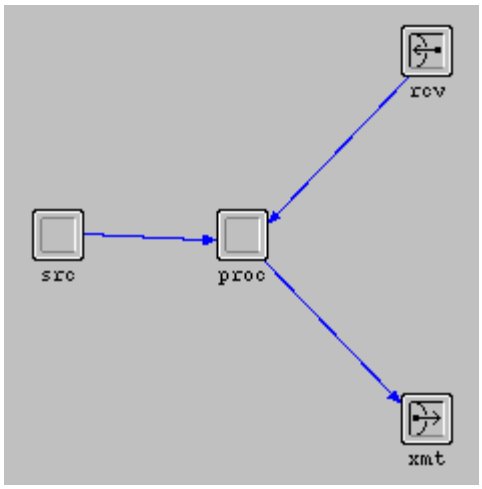
```
{
total_sent1 = 0;
time_slot = 1;
}
total_sent1+= 1024;
}
else if((int)op_sim_time()%3 < 2)
{
if(time_slot !=2)
{
total_sent2 = 0;
time_slot = 2;
}
total_sent2+= 1024;
}
else
{
if(time_slot !=3)
{
total_sent3 = 0;
time_slot = 3;
}
total_sent3+= 1024;
}

if(dest_address <=5)  //counts web packets sent
total_norm_sent++;
else if(dest_address >6)  //counts p2p packets sent
total_peer_sent++;

//this if statement determines what packets to throttle (drop in this case.  A future
implementation would //simply move a throttled packet to the end of the send queue,
delaying it).  The statement checks if the packet //is a p2p packet, if the route is congested
and if over 30% of sent traffic is p2p packets.  If all these cases //return true the packet is
throttled:
if(((dest_address <=6 || total_peer_sent/(total_norm_sent+total_peer_sent)<0.30)
||(total_sent1 +total_sent2 + total_sent3) < 20000*2)
||filter_flag == 0)
{

//this section determines how a packet is to be routed once it is decided that the packet
should be sent (and not //throttled).  This is basically a routing table. This section also
keeps track of distance a packet has travelled in //kilometers  This section is different for
the 6 different ISP Routers.
if (dest_subnet == 10)
{
```

```
op_pk_send (pkptr, 2);
total_distance_sent+= 202*1.8;
}
else if (dest_subnet == 11)
{
op_pk_send (pkptr, 1);
total_distance_sent+= 202*1.9;
}
else if (dest_subnet == 8)
{
op_pk_send (pkptr, 4);
total_distance_sent+= 202*2.8;
}
else if (dest_subnet == 9)
{
op_pk_send (pkptr, 3);
total_distance_sent+= 202*1;
}
else if (dest_subnet == 12)
{
op_pk_send (pkptr, 0);
total_distance_sent+= 202*8.2;
}
else if (dest_subnet == 7)
{
op_pk_send (pkptr, 5);
total_distance_sent+= 202*7.7;
}
else
{
op_pk_send (pkptr, 6);
total_distance_sent+= 202*5.8;
}

//this sections determines the distance from the origin subnet and adds it to the packet's
total distance //travelled if the packet wasn't received from another ISP Router.  This
section is also different for each
//ISP Router and is just a table with different node values and corresponding distances
if(origin_subnet == 10)
{
total_distance_sent+= 202*1.8;
}
else if(origin_subnet == 11)
{
total_distance_sent+= 202*1.9;
}
```

```
else if(origin_subnet == 9)
{
total_distance_sent+= 202*1;
}
else if(origin_subnet == 8)
{
total_distance_sent+= 202*2.8;
}
else;
distance_packet_speed_temp = total_distance_sent;

op_stat_write (distance_packet_speed, distance_packet_speed_temp);
}

FOUT;
}

//hub init enter execs
Objid parent_subnet;
parent_subnet = op_topo_parent (op_topo_parent(op_id_self ()));
op_ima_obj_attr_get_int32 (parent_subnet, "user id", &subnet_id);
distance_packet_speed = op_stat_reg ("Sum(Total Distances Travelled by Each Packet)
(km)", OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);

total_norm_sent =0;
total_peer_sent = 0;
total_sent1 =0;
total_sent2 =0;
total_sent3 =0;
time_slot =1;
```

Web (Normal) User Node:

Node Model:
(Based on Peripheral Node Node model from Opnet Packet Switching Tutorial #1)



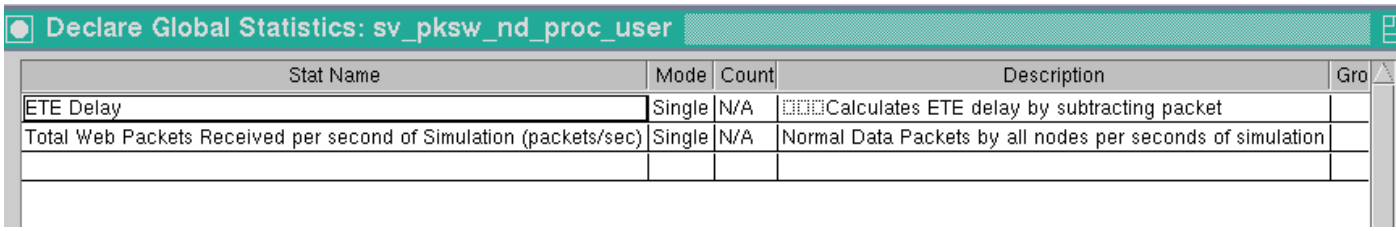
Proc Process Model: Same as ISP Router Proc Process Model

Web User State Variable:

sv_pksw_nd_proc_user.state variables

Type	Name	Comments
Distribution *	address_dist	
Stathandle	ete_gsh	
int	server_rcv_counter	
Stathandle	norm_speed	
int	subnet_id	

Web User Global Variables Interface:



Web User Header:
(based on code from Peripheral Node model from Opnet Packet Switching 2 Tutorial)

```
Objid parent_subnet;
parent_subnet = op_topo_parent (op_topo_parent(op_id_self ()));
op_ima_obj_attr_get_int32 (parent_subnet, "user id", &subnet_id);
distance_packet_speed = op_stat_reg ("Sum(Total Distances Travelled by Each Packet)
(km)", OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);

total_norm_sent =0;
total_peer_sent = 0;
total_sent1 =0;
total_sent2 =0;
total_sent3 =0;
time_slot =1;
```

Web User Init Execs:
(partially based on code from Peripheral Node model from Opnet Packet Switching 1 and 2 Tutorial)

```
Objid parent_subnet;
parent_subnet = op_topo_parent (op_topo_parent(op_id_self ()));
op_ima_obj_attr_get_int32 (parent_subnet, "user id", &subnet_id);

address_dist = op_dist_load ("uniform_int", 6, 6);
ete_gsh = op_stat_reg ("ETE Delay", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
norm_speed = op_stat_reg ("Total Web Packets Received per second of Simulation
(packets/sec)", OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
```

Web User Function Block:
(mostly written from scratch, partially based on code from Peripheral Node model from Opnet Packet Switching 1 and 2 Tutorial)

```
static void xmt(void)
{
Packet * pkptr;
FIN(xmt());
```

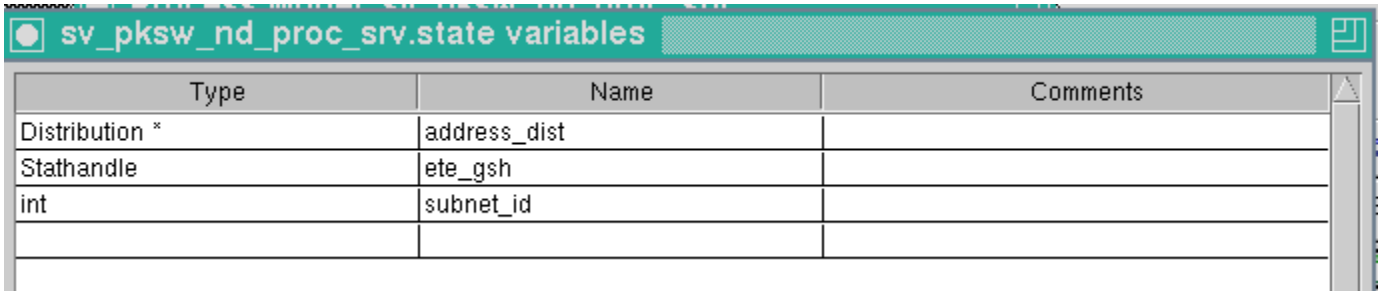
```
pkptr = op_pk_get (SRC_IN_STRM);

op_pk_nfd_set_int32 (pkptr, "dest_subnet", (int)op_dist_uniform (13.0) + 1);
op_pk_nfd_set_int32 (pkptr, "origin", subnet_id );
op_pk_nfd_set_int32 (pkptr, "dest_address",(int)op_dist_outcome (address_dist));
op_pk_send (pkptr, XMT_OUT_STRM);
FOUT;
}

static void rcv(void)
{
Packet *pkptr;
double ete_delay;
double norm_speed_temp;
FIN (rcv());
pkptr = op_pk_get (RCV_IN_STRM);
server_rcv_counter++;
ete_delay = op_sim_time () - op_pk_creation_time_get (pkptr);
normal_packets_total++; //variable counts the total data packets
//received by normal users (non-P2P)
norm_speed_temp = normal_packets_total/op_sim_time ();
op_stat_write (ete_gsh, ete_delay);
op_stat_write (norm_speed, norm_speed_temp);
op_pk_destroy (pkptr);
FOUT;
}
```

Server Node:

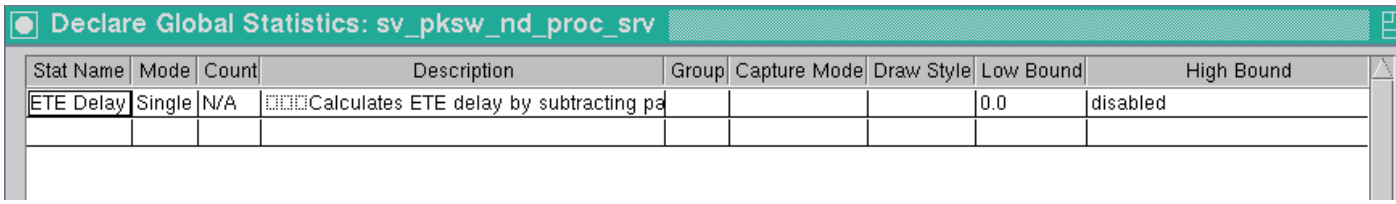
Node Model: Same as Web User

Proc Process Model: Same as ISP Proc Process Model

Server State Variables:



Server Global Variables:



Server Header:
(from Opnet Packet Switching Tutorial #1)

```
/* packet stream definitions */
#define RCV_IN_STRM 0
#define SRC_IN_STRM 1
#define XMT_OUT_STRM 0
/* transition macros */
#define SRC_ARRVL (op_intrpt_type () == OPC_INTRPT_STRM && op_intrpt_strm
() == SRC_IN_STRM)

#define RCV_ARRVL (op_intrpt_type () == OPC_INTRPT_STRM && op_intrpt_strm
() == RCV_IN_STRM)
```

Server Init Enter Execs:
(partially based on code from Peripheral Node model from Opnet Packet Switching 1 and 2 Tutorial)

```
Objid parent_subnet;
parent_subnet = op_topo_parent (op_topo_parent(op_id_self ()));
op_ima_obj_attr_get_int32 (parent_subnet, "user id", &subnet_id);
address_dist = op_dist_load ("uniform_int", 0, 5);
ete_gsh = op_stat_reg ("ETE Delay", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
```

Server Function Block:
(partially based on code from Peripheral Node model from Opnet Packet Switching 1 and 2 Tutorial)

```
static void xmt(void)
{
Packet * pkptr;
FIN(xmt());
pkptr = op_pk_get (SRC_IN_STRM);


//address_dist = op_dist_load ("uniform_int", 0, 1);
op_pk_nfd_set_int32 (pkptr, "dest_subnet", (int)op_dist_uniform (13.0) + 1);
op_pk_nfd_set_int32 (pkptr, "origin", subnet_id );
op_pk_nfd_set_int32 (pkptr, "dest_address",(int)op_dist_outcome (address_dist));
op_pk_send (pkptr, XMT_OUT_STRM);
FOUT;
```

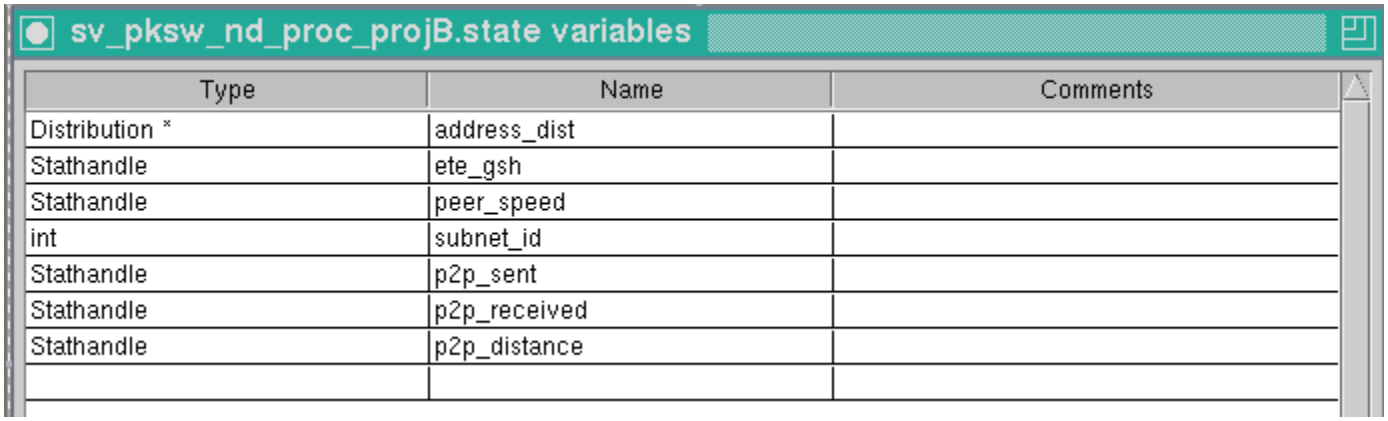
```
}

static void rcv(void)
{
Packet *pkptr;
double ete_delay;
FIN (rcv());
pkptr = op_pk_get (RCV_IN_STRM);

//ete_gsh = op_stat_reg ("ETE Delay", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
ete_delay = op_sim_time () - op_pk_creation_time_get (pkptr);

op_stat_write (ete_gsh, ete_delay);
op_pk_destroy (pkptr);
FOUT;
}
```

P2P Node:

Node Model: Same as Web User

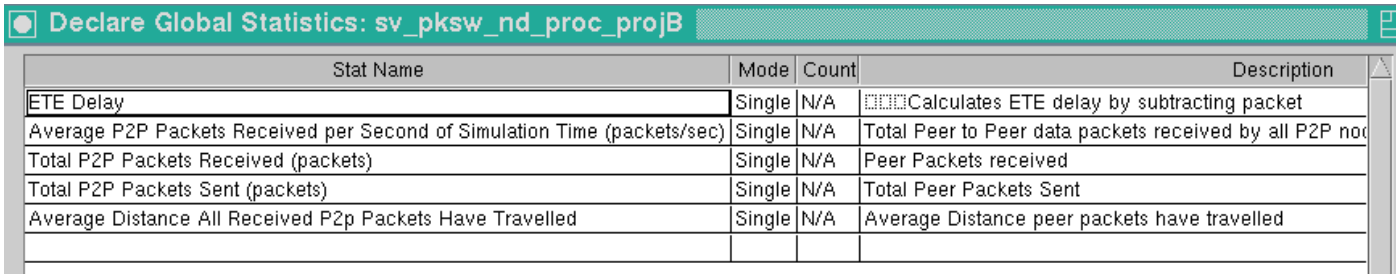
Server Proc Process Model: Same as ISP Proc Process Model

P2P State Variables:



Type	Name	Comments
Distribution *	address_dist	
Stathandle	ete_gsh	
Stathandle	peer_speed	
int	subnet_id	
Stathandle	p2p_sent	
Stathandle	p2p_received	
Stathandle	p2p_distance	

P2P Global Variables Interface:



Stat Name	Mode	Count	Description
ETE Delay	Single	N/A	Calculates ETE delay by subtracting packet
Average P2P Packets Received per Second of Simulation Time (packets/sec)	Single	N/A	Total Peer to Peer data packets received by all P2P no
Total P2P Packets Received (packets)	Single	N/A	Peer Packets received
Total P2P Packets Sent (packets)	Single	N/A	Total Peer Packets Sent
Average Distance All Received P2p Packets Have Travelled	Single	N/A	Average Distance peer packets have travelled

P2P Header Block:
(based on code from Peripheral Node model from Opnet Packet Switching 1 and 2 Tutorial)

```
/* packet stream definitions */
#define RCV_IN_STRM 0
#define SRC_IN_STRM 1
#define XMT_OUT_STRM 0
/* transition macros */
#define SRC_ARRVL (op_intrpt_type () == OPC_INTRPT_STRM && op_intrpt_strm
() == SRC_IN_STRM)

#define RCV_ARRVL (op_intrpt_type () == OPC_INTRPT_STRM && op_intrpt_strm
() == RCV_IN_STRM)

#define START (intrpt_code == SSC_START)
double peer_packets_total = 0;

int peer_packets_sent =0;
```

P2P Init Enter Execs:
(partially based on code from Peripheral Node model from Opnet Packet Switching 1 and 2 Tutorial)

```
Objid parent_subnet;
parent_subnet = op_topo_parent (op_topo_parent(op_id_self ()));
op_ima_obj_attr_get_int32 (parent_subnet, "user id", &subnet_id);

address_dist = op_dist_load ("uniform_int", 7, 9);
ete_gsh = op_stat_reg ("ETE Delay", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
peer_speed = op_stat_reg ("Average P2P Packets Received per Second of Simulation
Time (packets/sec)", OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);


p2p_sent = op_stat_reg ("Total P2P Packets Sent (packets)",
OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
p2p_received = op_stat_reg ("Total P2P Packets Received (packets)",
OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);

//p2p_distance = op_stat_reg ("Sum of Distance Travelled by all Packets/Sum of Packets
Received", OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
```

P2P Function Block:
(partially based on code from Peripheral Node model from Opnet Packet Switching 1 and 2 Tutorial)

```
static void xmt(void)
        {
        Packet * pkptr;
        FIN(xmt());
        pkptr = op_pk_get (SRC_IN_STRM);


        //address_dist = op_dist_load ("uniform_int", 2, 3);

        op_pk_nfd_set_int32 (pkptr, "dest_subnet", (int)op_dist_uniform (13.0) + 1);
        op_pk_nfd_set_int32 (pkptr, "origin", subnet_id);

        op_pk_nfd_set_int32 (pkptr, "dest_address",(int)op_dist_outcome
(address_dist));
        op_pk_send (pkptr, XMT_OUT_STRM);
        peer_packets_sent++;
        op_stat_write (p2p_sent , peer_packets_sent);
        FOUT;
        }
```

```
static void rcv(void)
        {
        Packet *pkptr;
        double ete_delay;
        double peer_speed_temp;

        FIN (rcv());
        pkptr = op_pk_get (RCV_IN_STRM);

        //ete_gsh = op_stat_reg ("ETE Delay", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
        ete_delay = op_sim_time () - op_pk_creation_time_get (pkptr);
        peer_packets_total++;
        peer_speed_temp = peer_packets_total/op_sim_time();
        op_stat_write (ete_gsh, ete_delay);
        op_stat_write (peer_speed, peer_speed_temp);
        op_stat_write (p2p_received , peer_packets_total);
        op_pk_destroy (pkptr);
        FOUT;
        }
```