ENSC 427: COMMUNICATION NETWORKS
FINAL PROJECT PRESENTATIONS
Spring 2010

Implementation of the Gnutella Protocol

Group #7

Zhiyu Hu

Yuyuan Liu

Email: yla41@sfu.ca zyh@sfu.ca
Webpage: http://www.sfu.ca/~yla41/

Outline

- Introduction and Motivation
- Scope of the project
- Implementation details of Gnutella node
  -Ping, pong, query, query hit.
- Scenarios and Simulation results
- Conclusion
- References

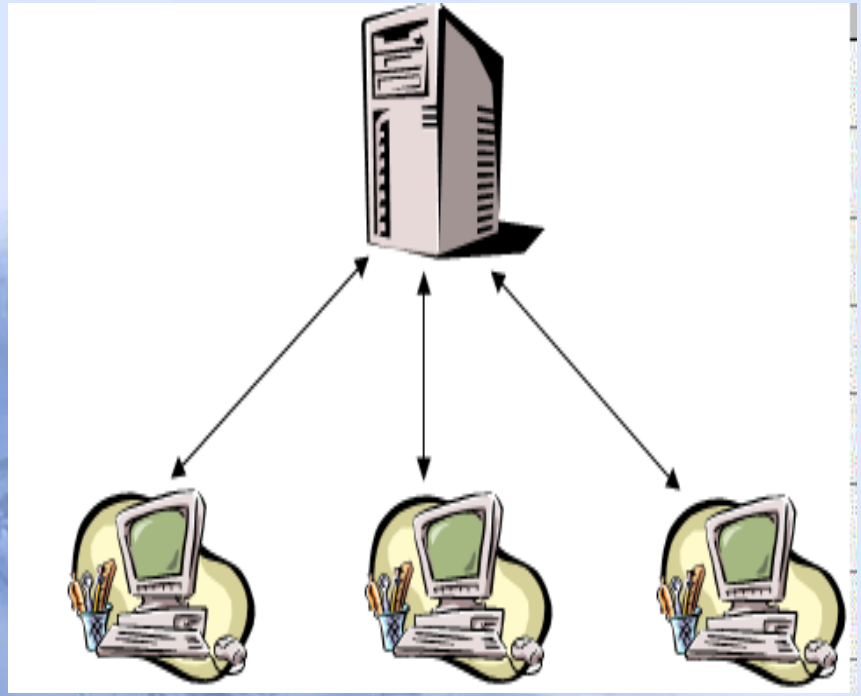
What is P2P?

- Is a technology which “enables any network-aware device to provide services to another network-aware device”

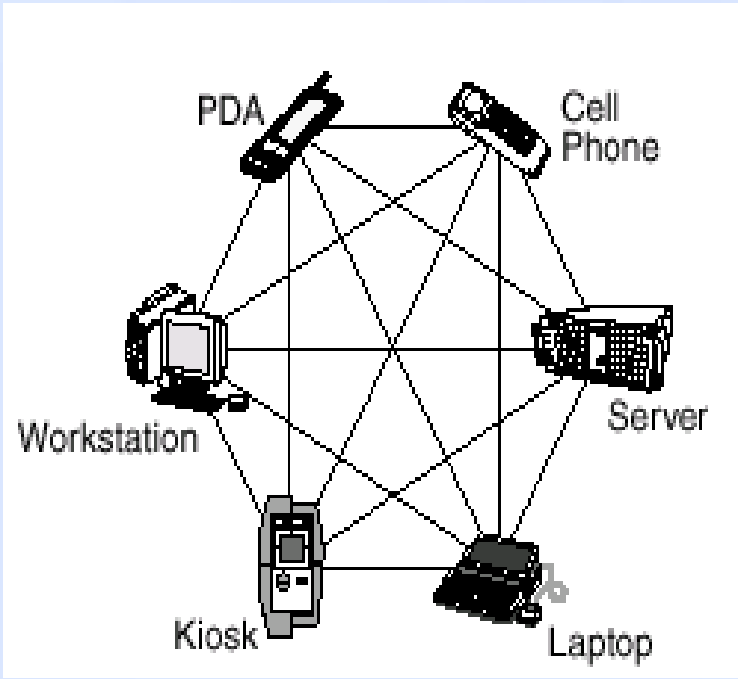
- A peer in P2P network acts as both a client and a server in traditional client/server architecture

What is P2P?

Not p2p



P2P



Why P2P?

- Harness lots of spare capacity
  - 1 Big Fast Server: 1Gbit/s, $10k/month++
  - 2,000 cable modems: 1Gbit/s, $??
  - 1 000 000 end hosts: Uh wow
- Build self-managing systems that deal with huge scale
  - Same techniques attractive for both companies /servers / P2P

    E.g., Akamai's 14,000 nodes

    Google's 100,000+ nodes

Overview of related work

P2P file-sharing

- Quickly grown in popularity

1. Dozens or hundreds of file sharing applications
2. 35 million American adults use P2P networks 29% of all Internet users in US!
3. Audio/Video transfer now dominates traffic on the Internet

Overview of related work

Gnutella:

- In 2000, J. Frankel and T. Pepper from Nullsoft released Gnutella
- Soon many other clients: Bearshare Bearshare, Morpheus, LimeWire, etc.
- In 2001, many protocol enhancements including “ultrapeers”

Scope of the project

- Establish the Gnutella node to simulate the behaviours of ping, pong, query and query hit.
    1. build packet format, process model and node model.
    2. by combining above three, we build Gnutella node.

Scope of the project

- Simulations in different topologies

  Hexagon, Tree, Line,etc.

Implementation details of Gnutella node

- How ping and pong work
- How query and query hit work
- Packet format
- Node model
- Process model
- Algorithm in Proc state

Implementation details of Gnutella node

- How ping and pong work
- How query and query hit work
- Packet format
- Node model
- Process model
- Algorithm in Proc state

How Ping and Pong work

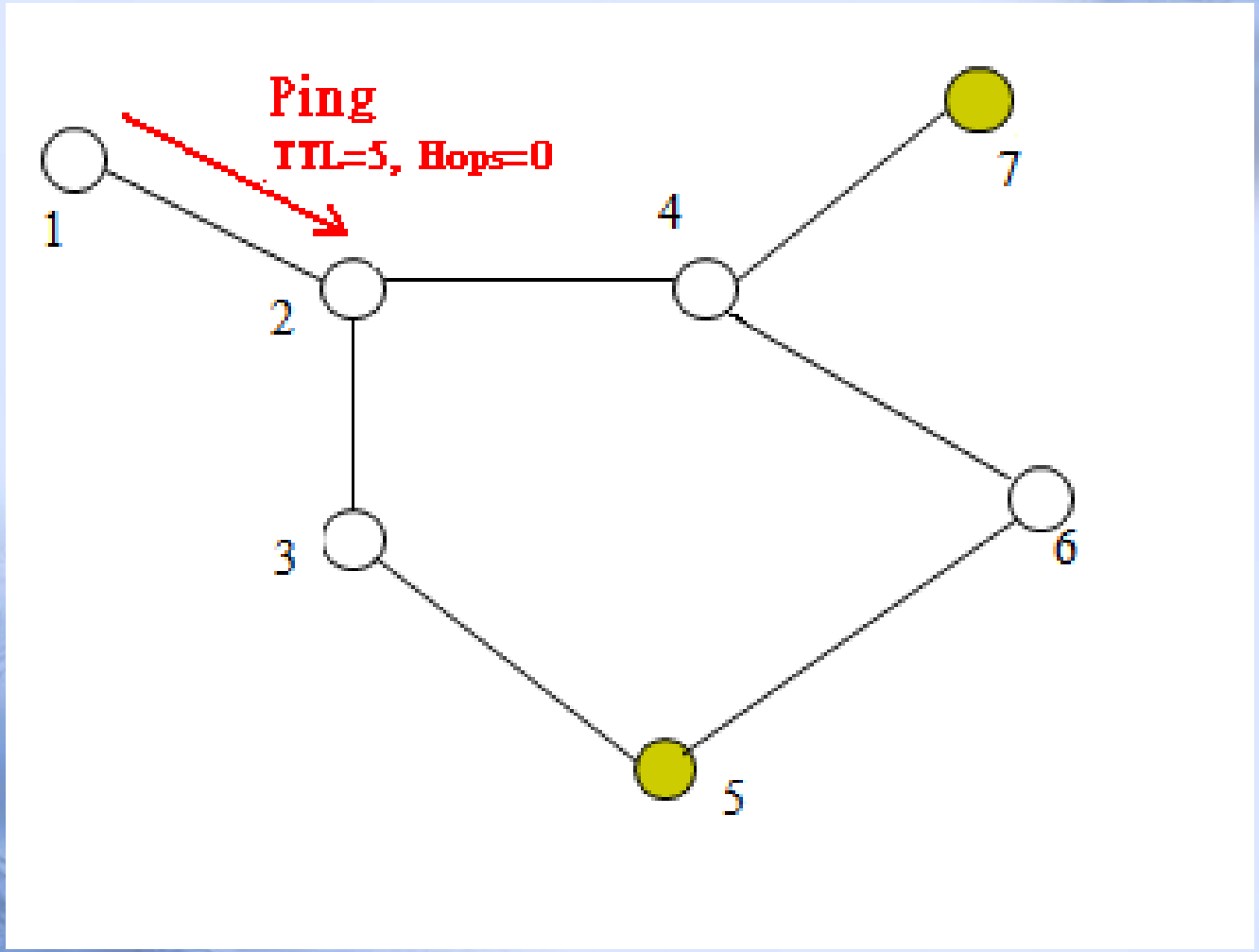


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How Ping and Pong work

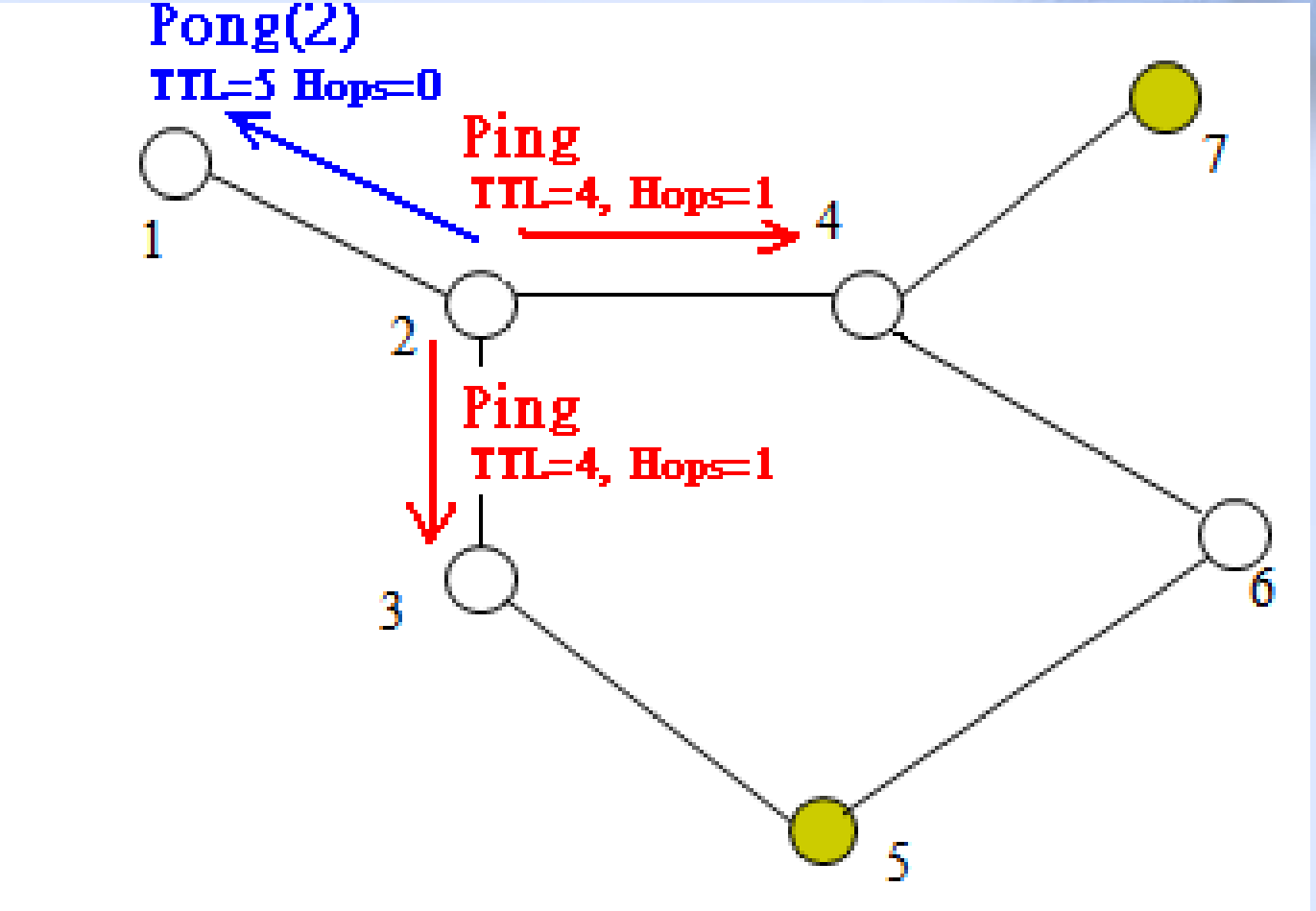


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How Ping and Pong work

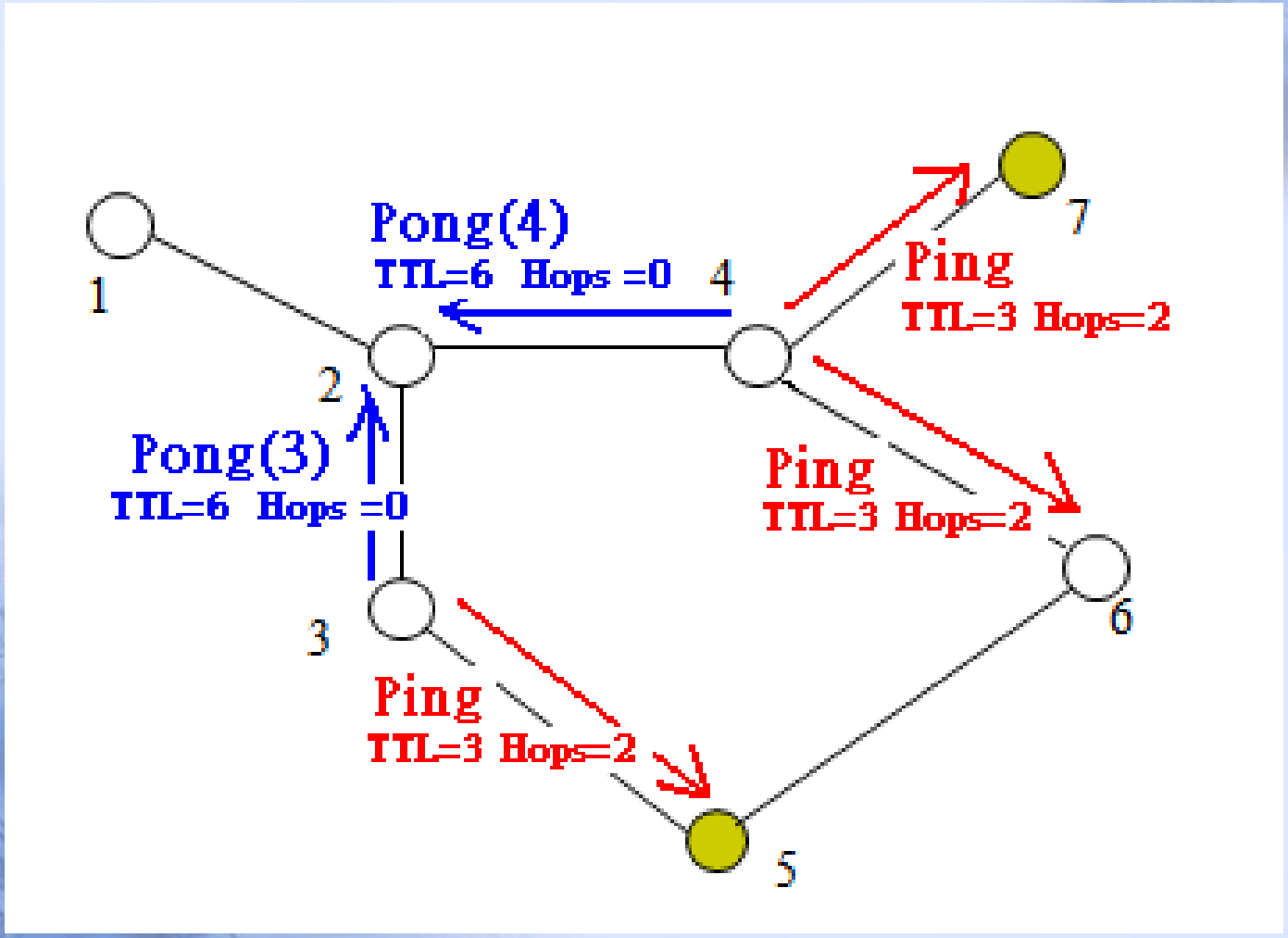


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How Ping and Pong work

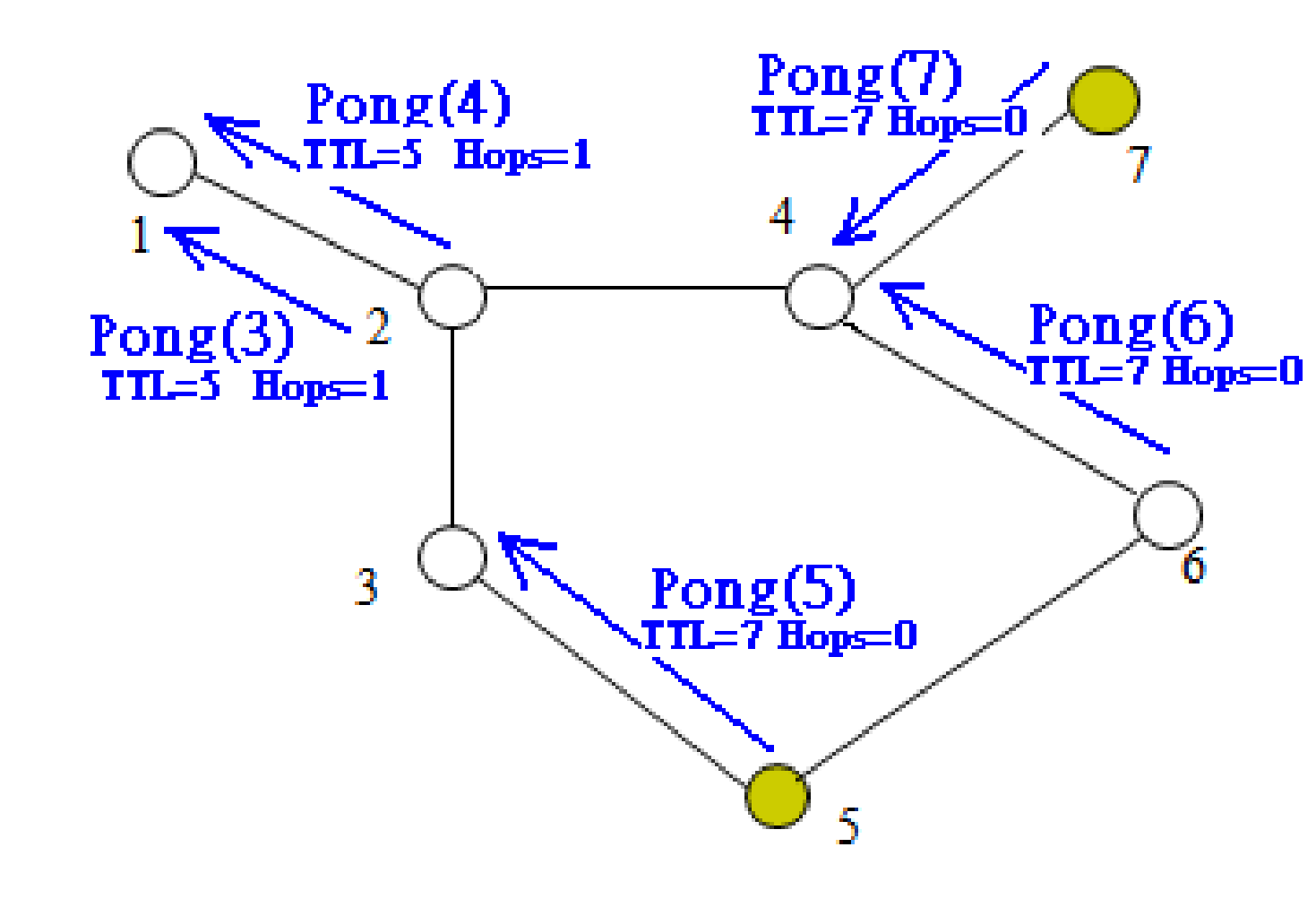


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How is pong routed

- Pong(7) will reach node(1) along the reversed direction of ping which is sent by node 1 and flooded by other nodes.

- This routing rule also applies to QueryHit. Query Hit will reach the source node of Query along the reversed direction of Query which is sent by node 1 and flooded by other nodes.

How ping and pong work

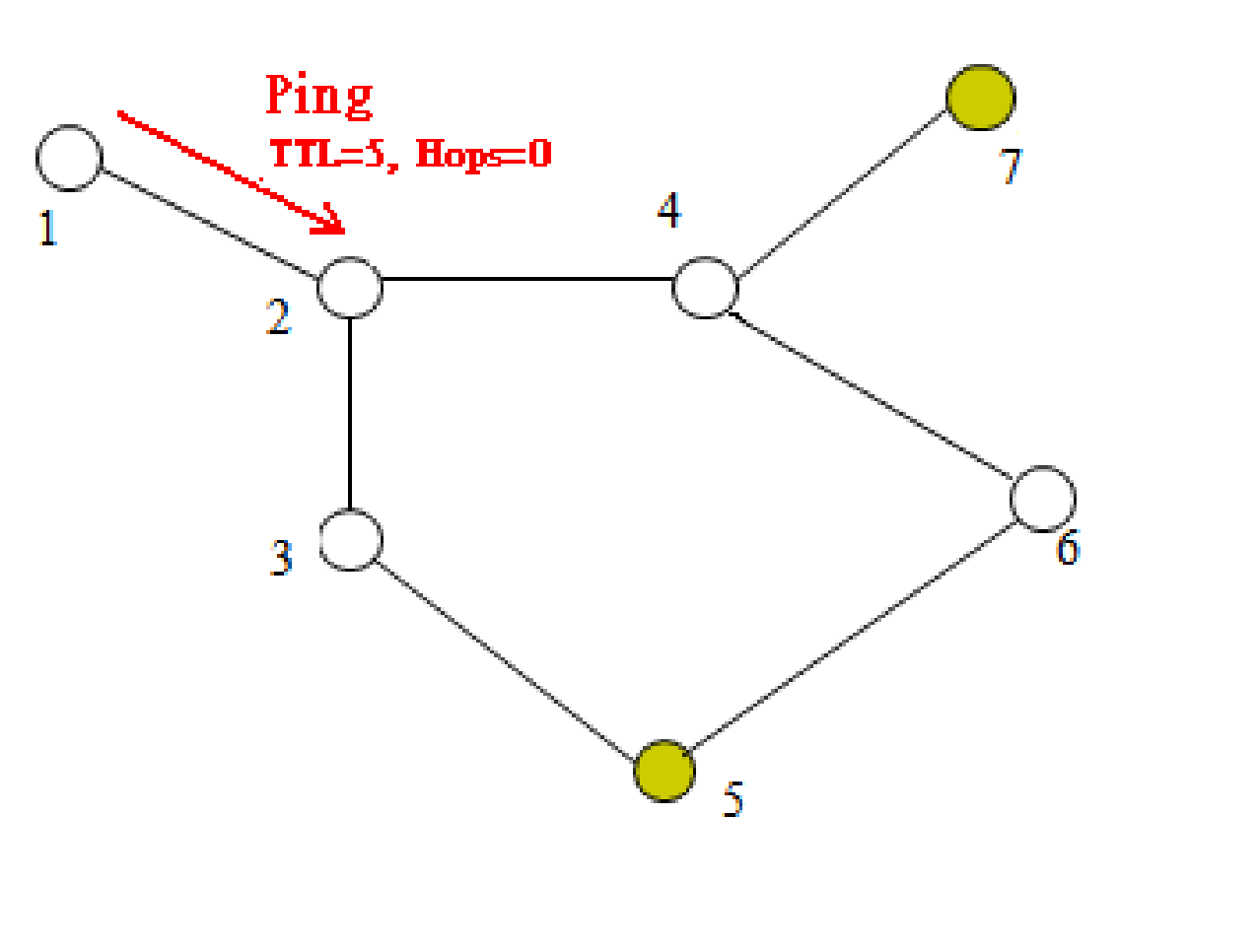


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How ping and pong work

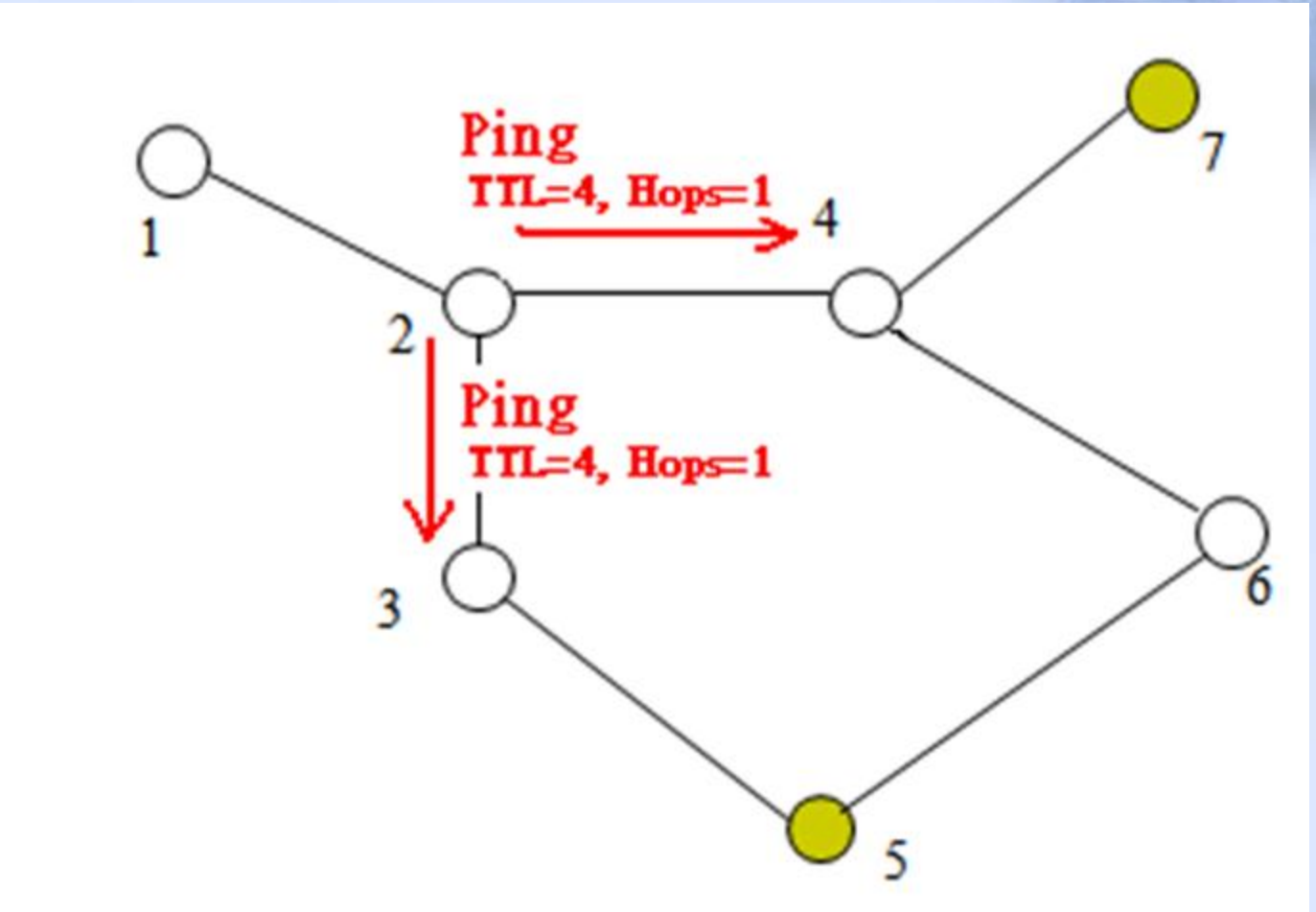


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How ping and pong work

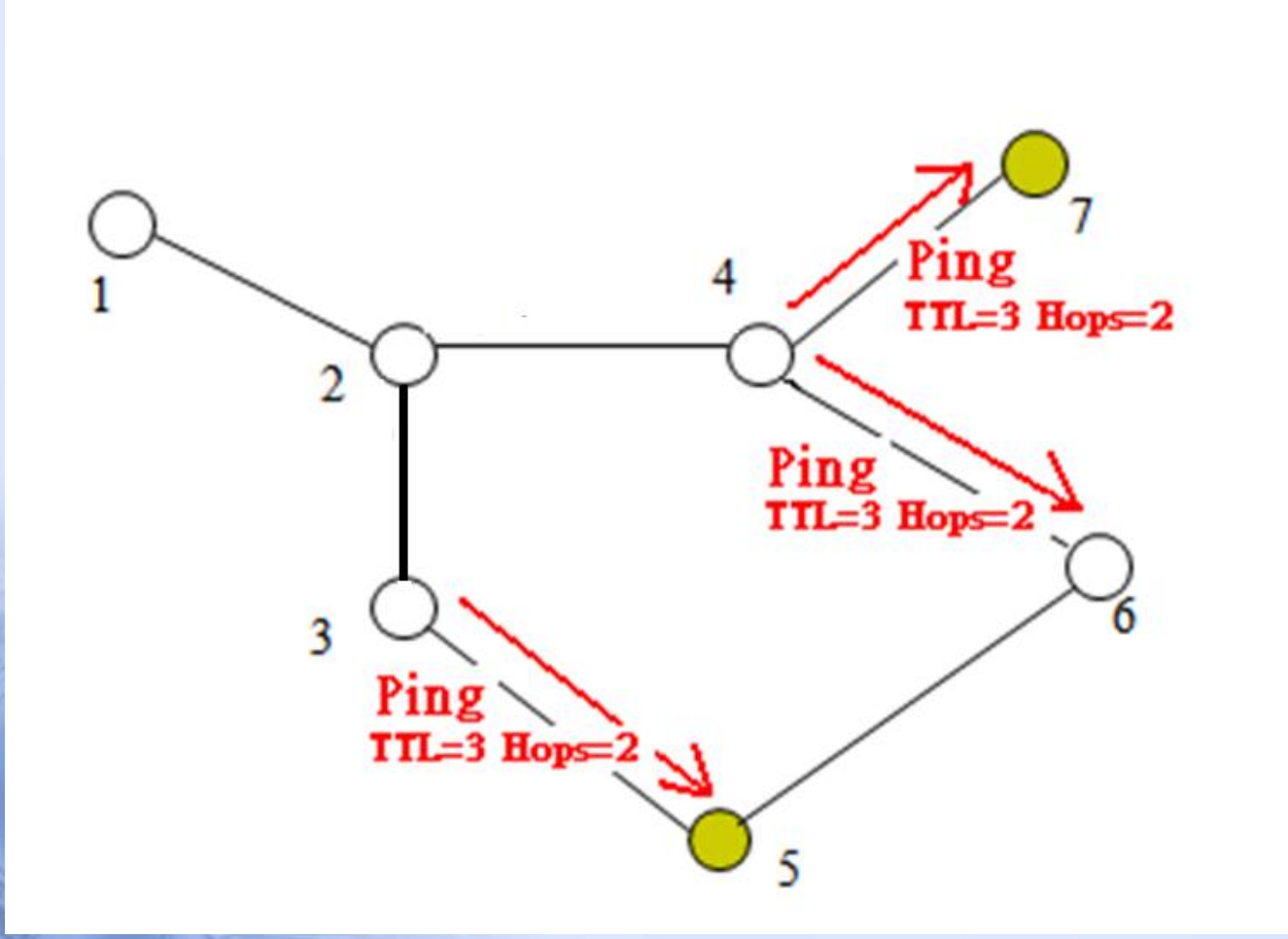


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How ping and pong work

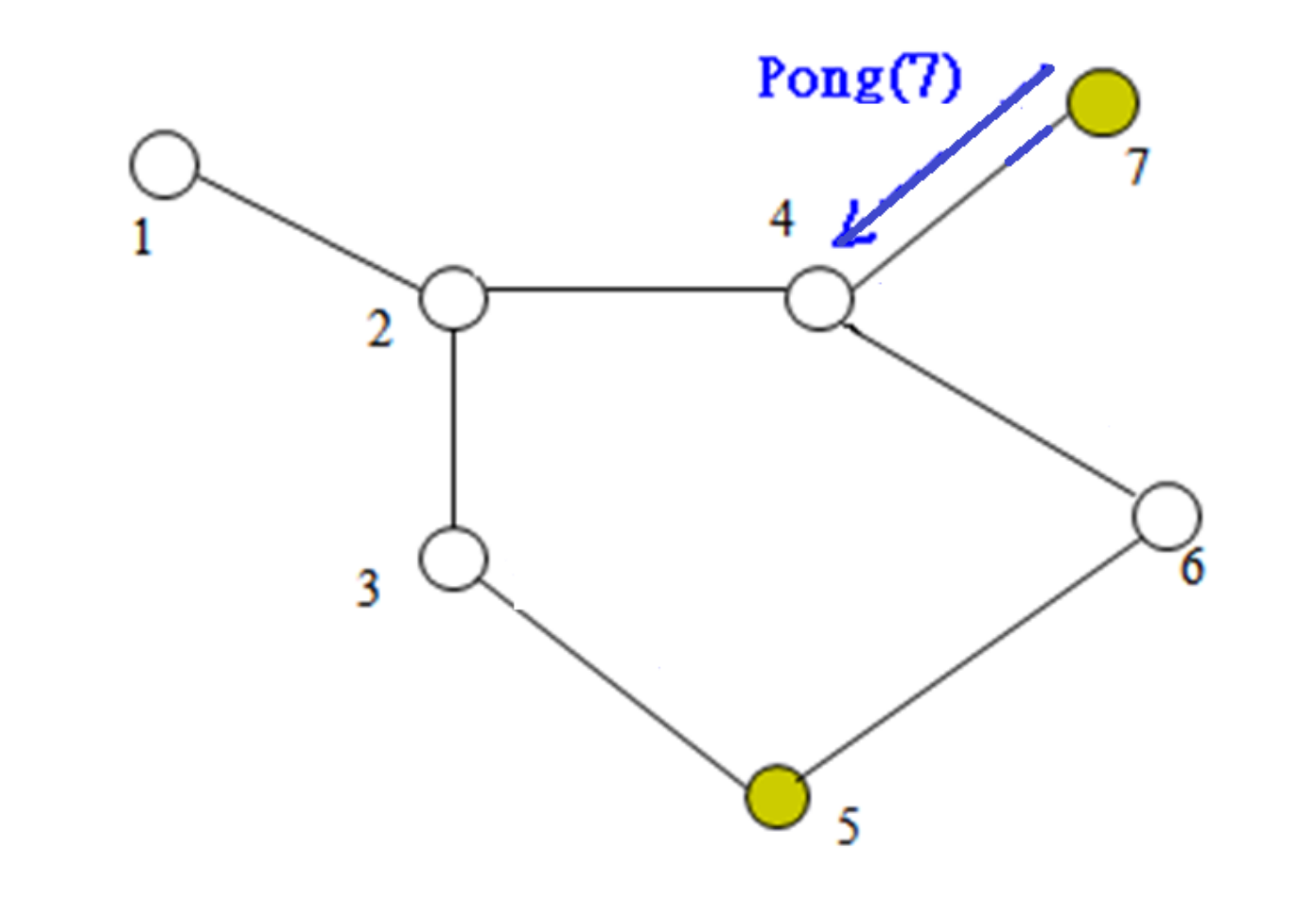


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How ping and pong work

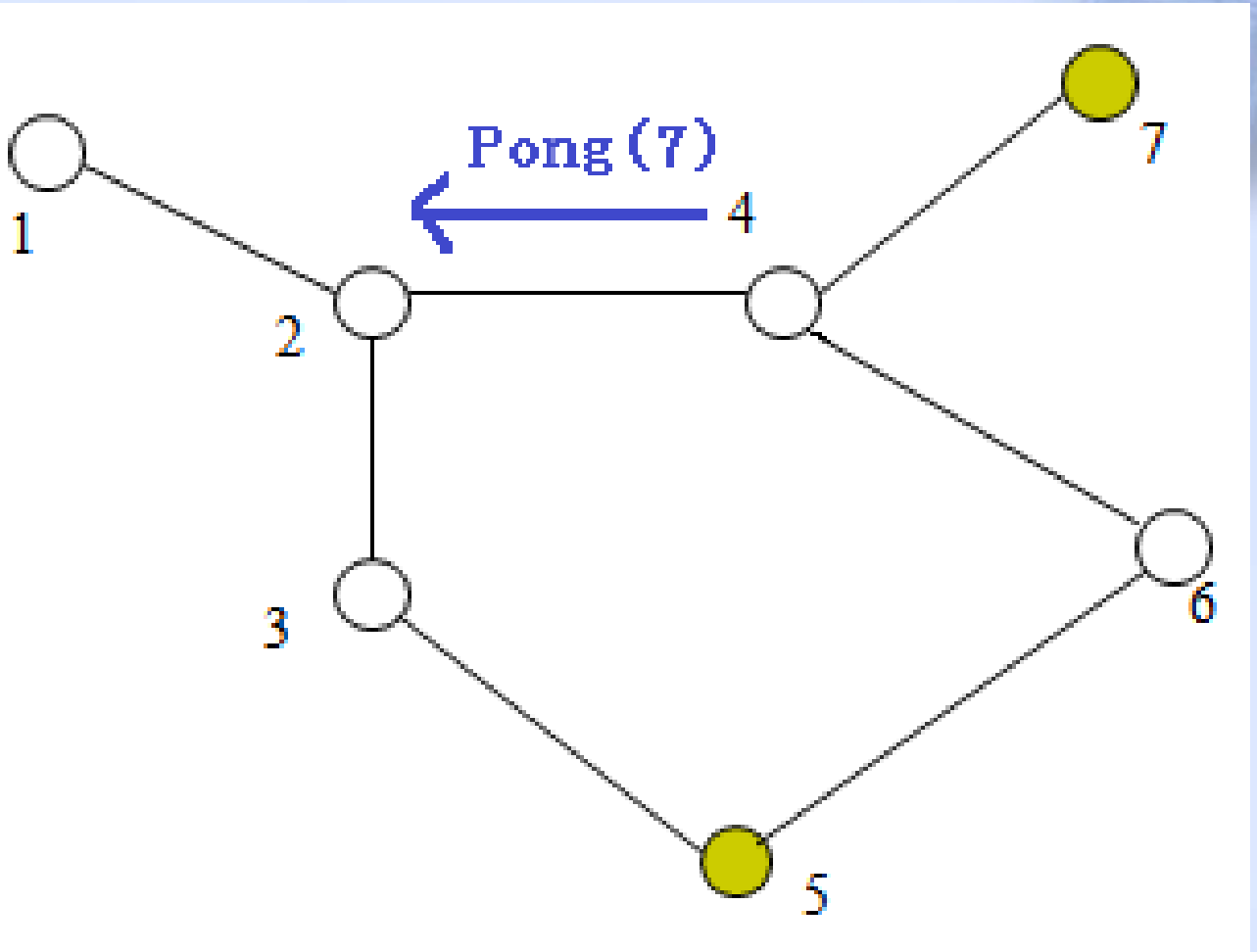


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How ping and pong work

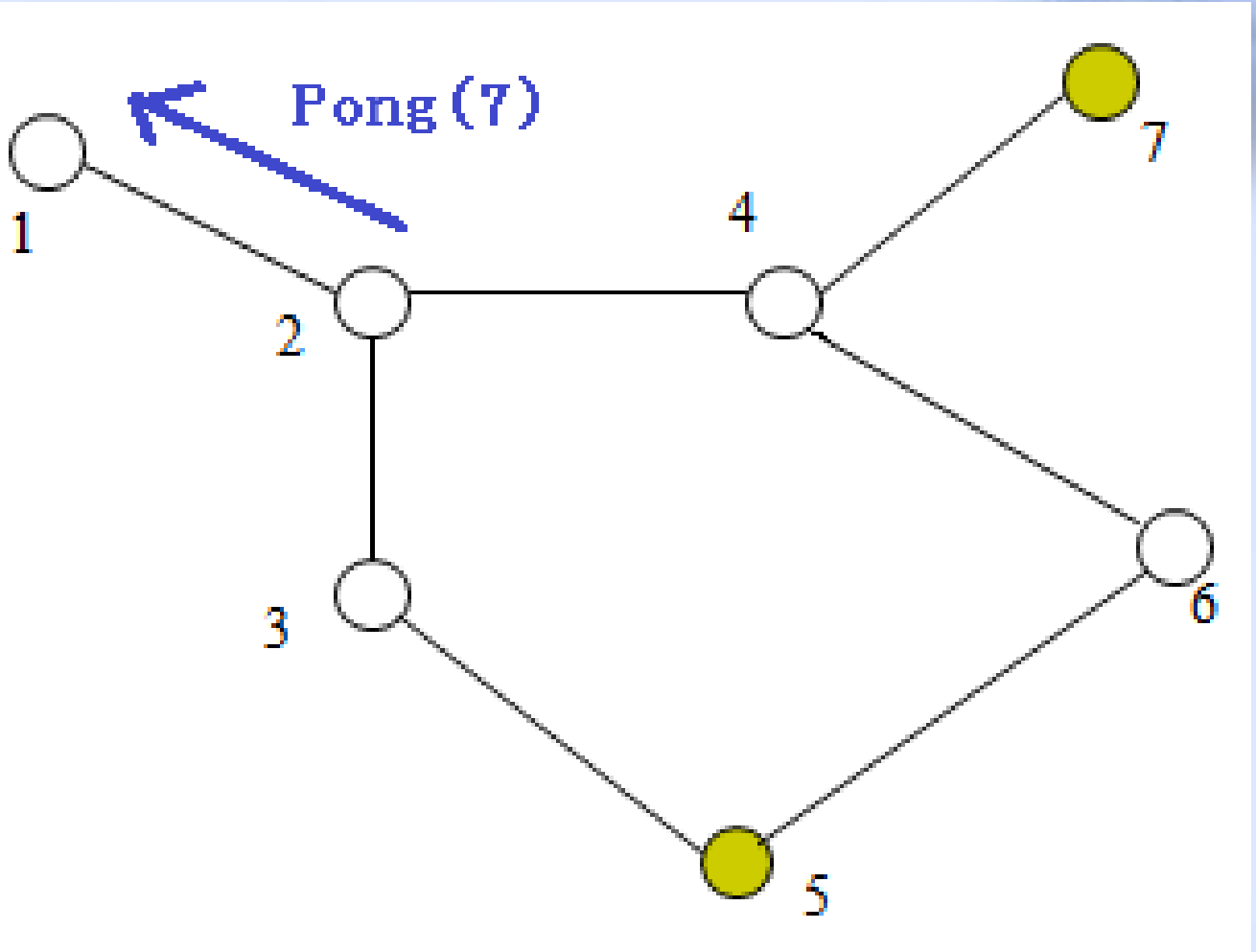


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

Implementation details of Gnutella node

- How ping and pong work
- How query and query hit work
- Packet format
- Node model
- Process model
- Algorithm in Proc state

How Query, Query Hit work

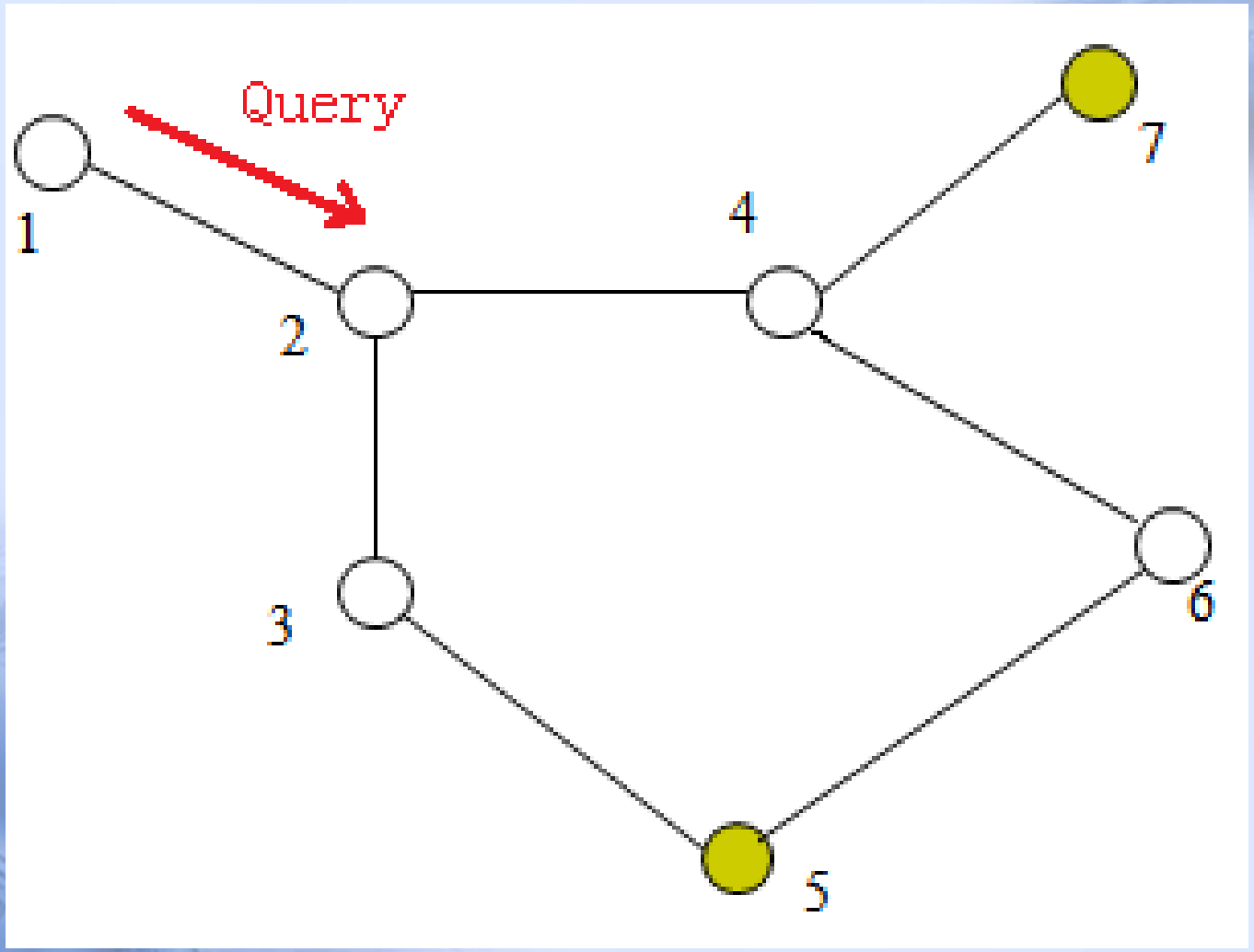


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How Query, Query Hit work

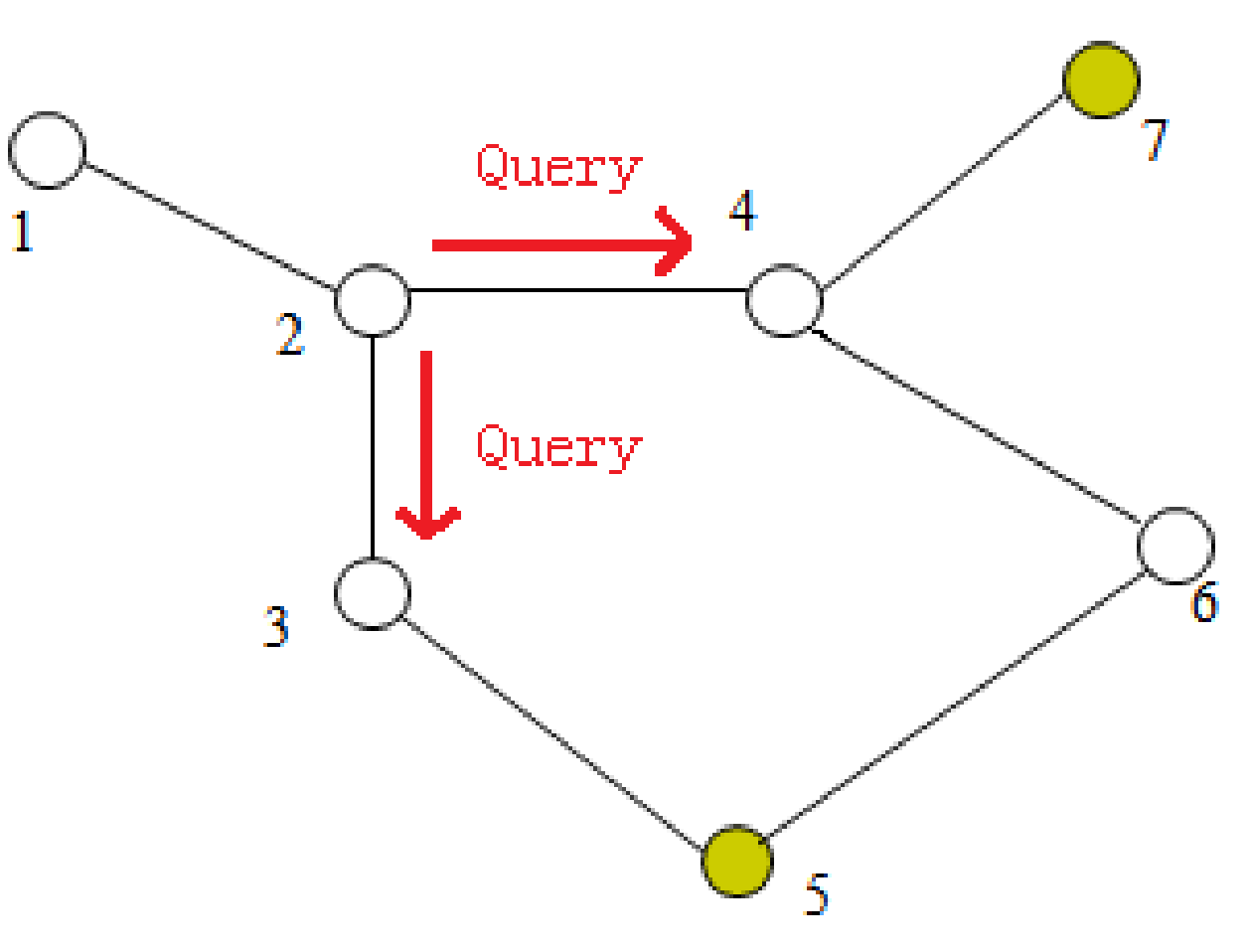


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How Query, Query Hit work

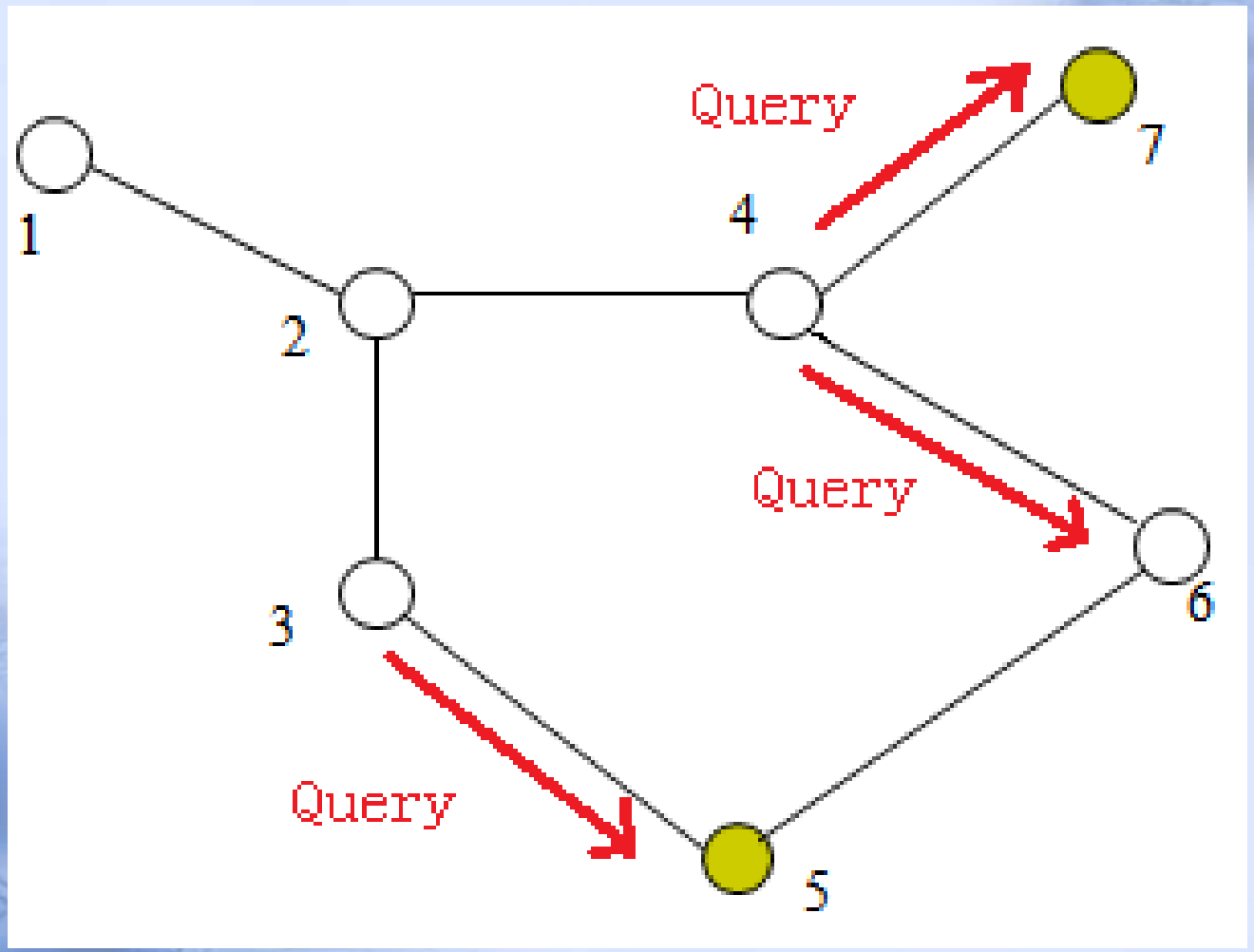


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How Query, Query Hit work

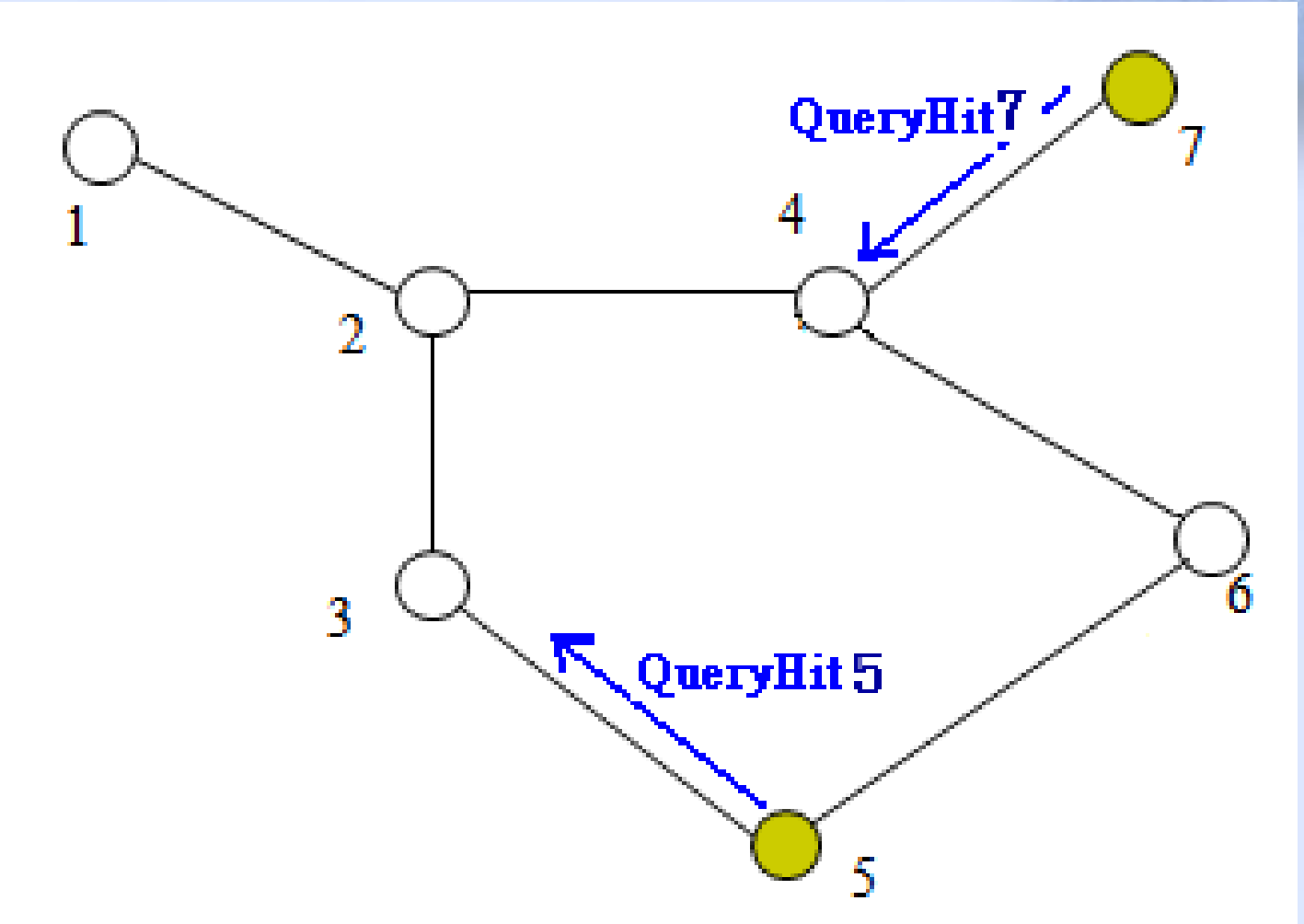


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How is QueryHit routed

- Pong(7) will reach node(1) along the reversed direction of ping which is sent by node 1 and flooded by other nodes.

- This routing rule also applies to QueryHit. QueryHit(7) will reach node 1 along the reversed direction of Query which is sent by node 1 and flooded by other nodes.

How Query, Query Hit work

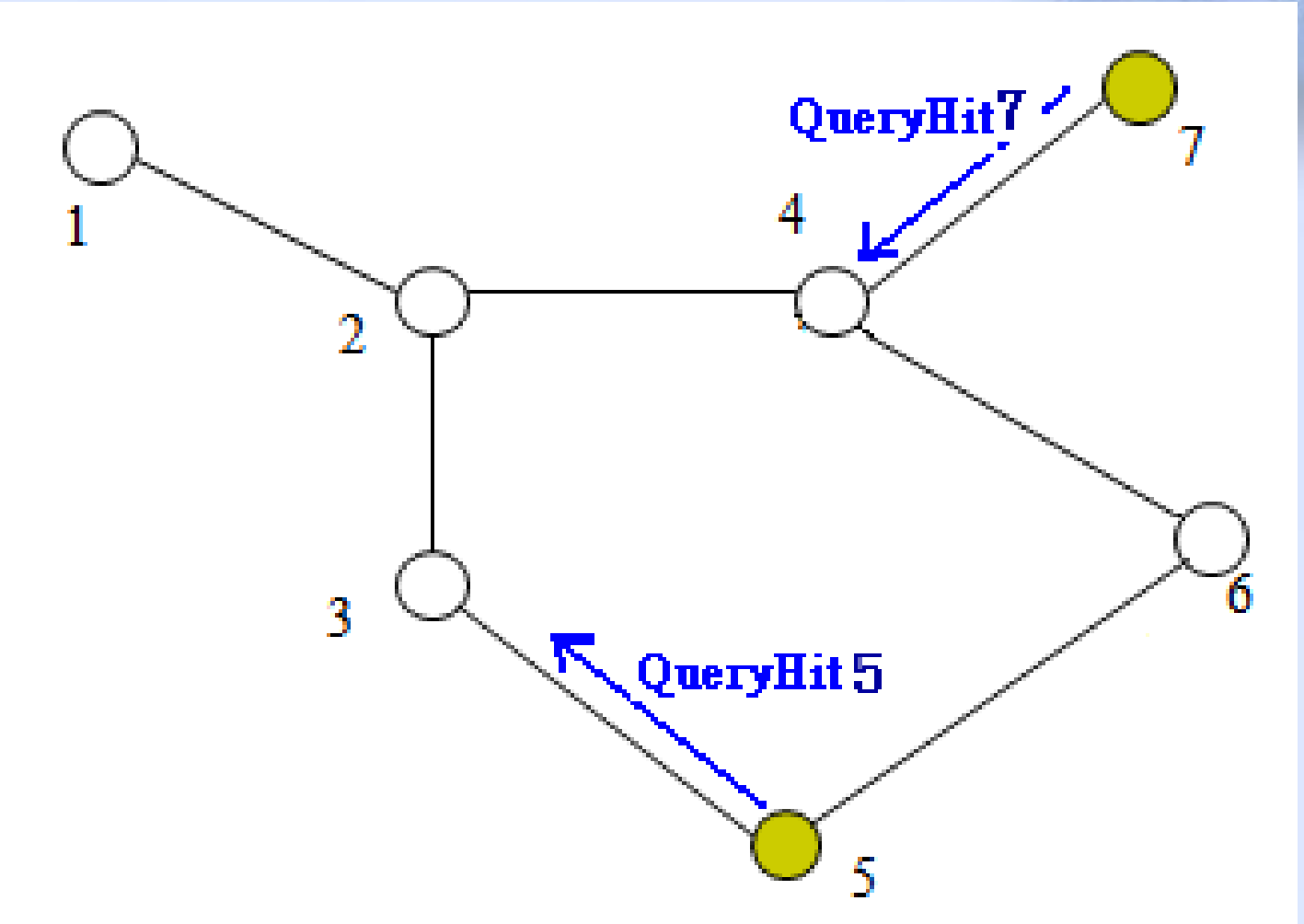


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How Query, Query Hit work

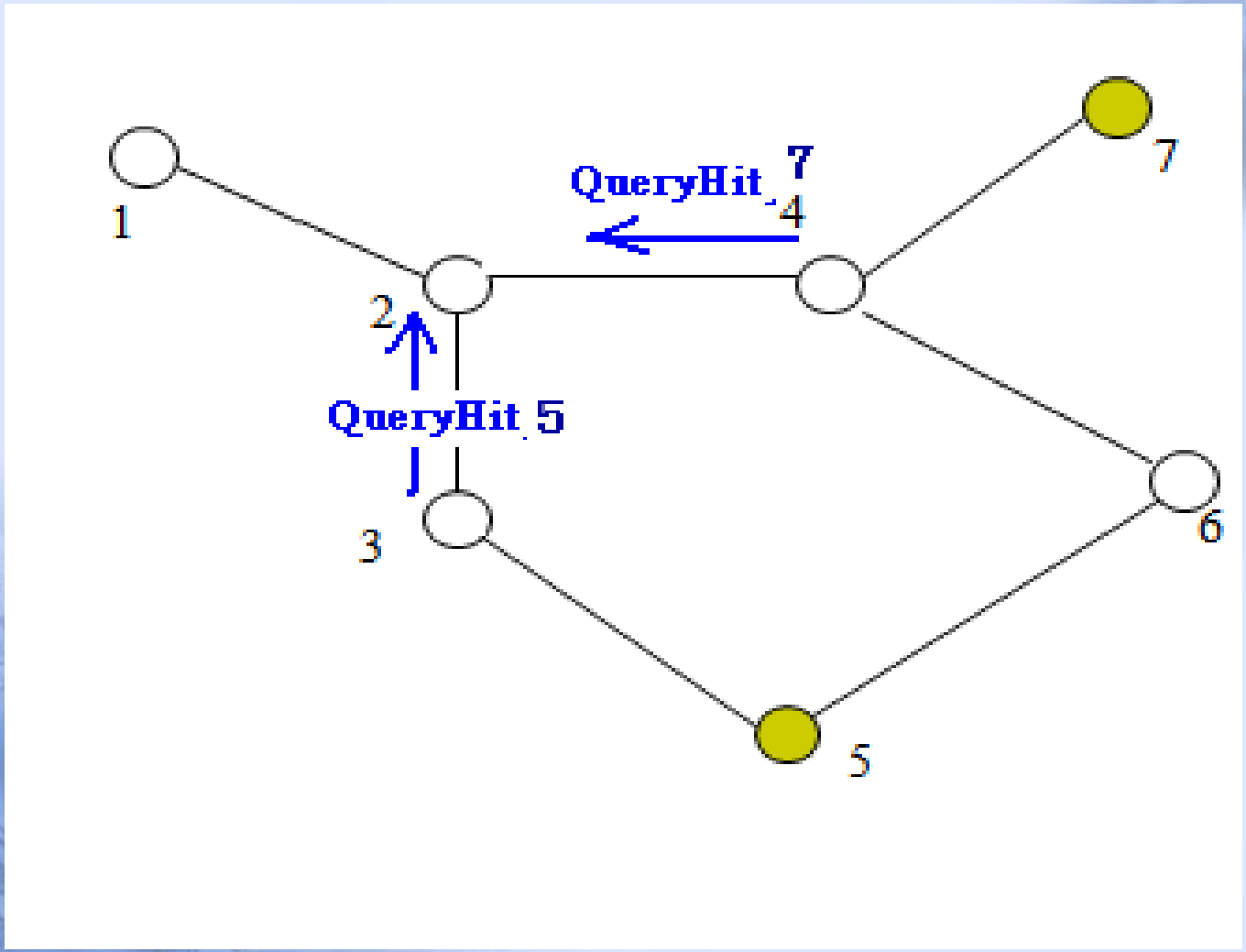


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

How Query, Query Hit work

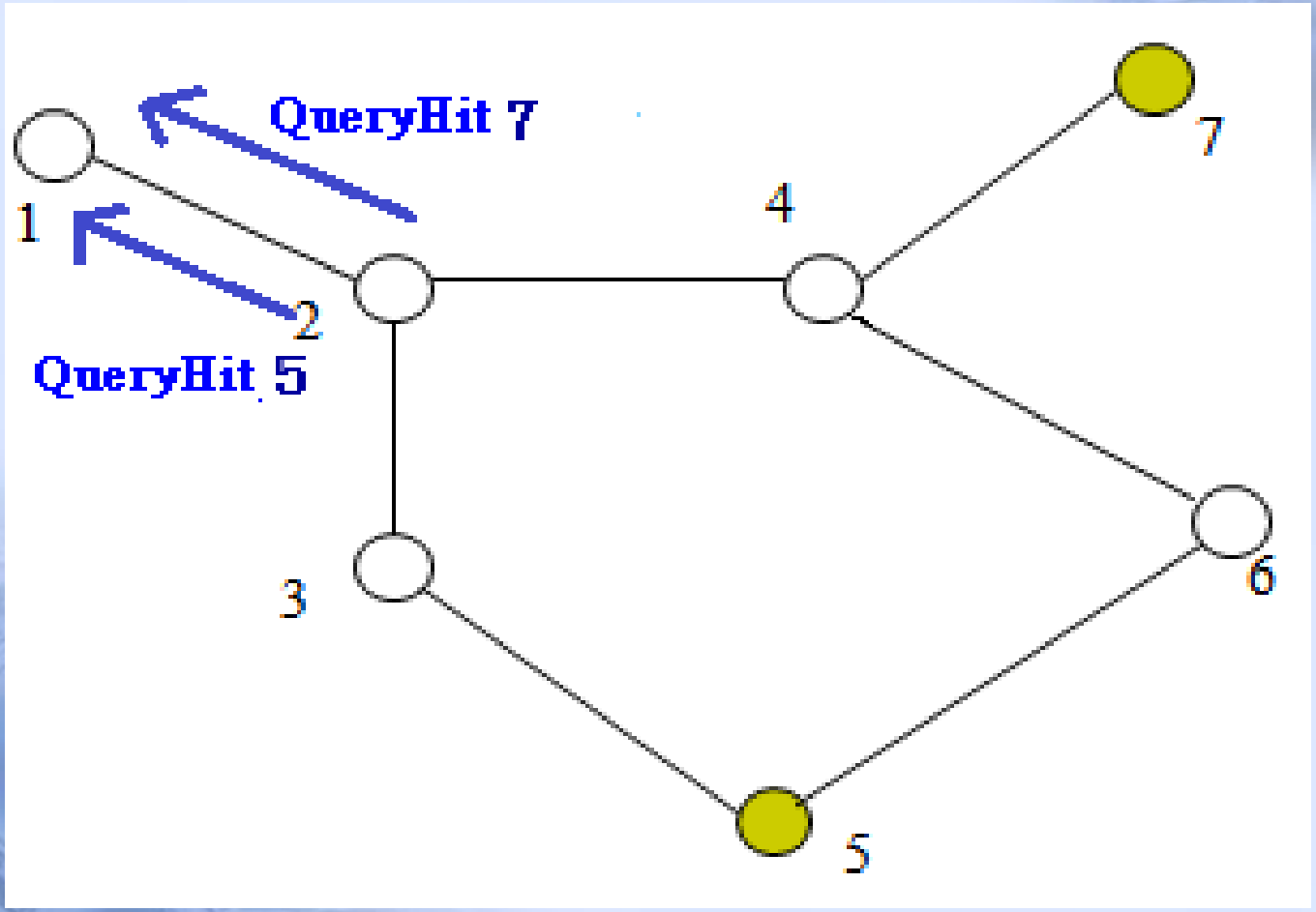


Image source: H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report.

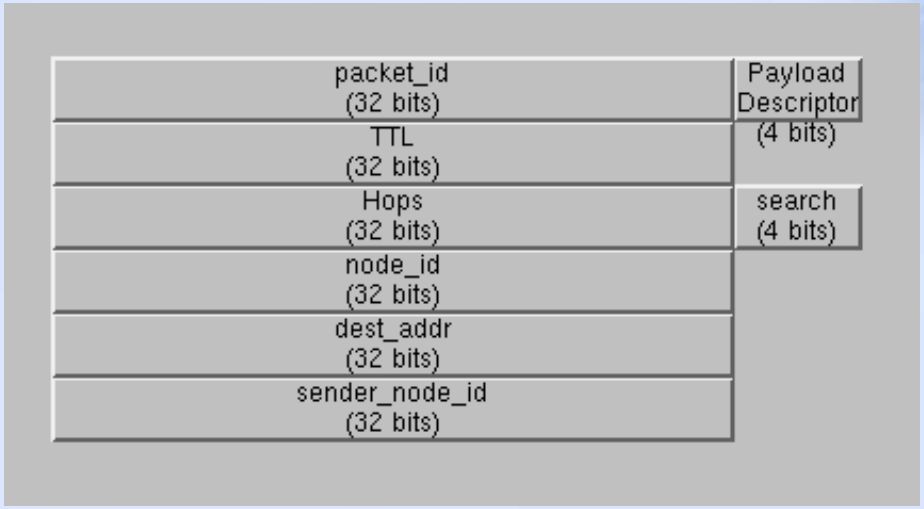
Implementation details of Gnutella node

- Our Gnutella node can implement ping, pong, query and query hit correctly.

Implementation details of Gnutella node

- How ping and pong work
- How query and query hit work
- Packet format
- Node model
- Process model
- Algorithm in Proc state

Packet format

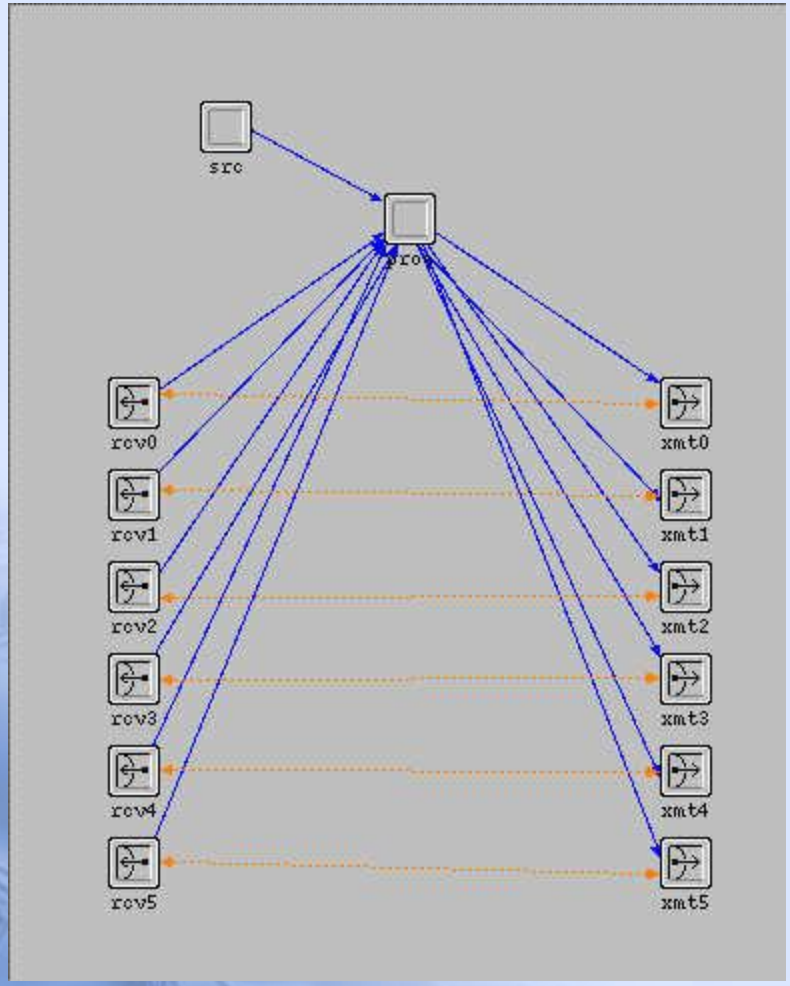


- Payload Descriptor: used to indicate packet type.
  ping 1, pong 2, query 4, query hit 8.
- TTL, Hops: control the total traffic.
- Dest_addr: used for pong and queryhit routing.
- Search: the content to be searched, used in Query.

Implementation details of Gnutella node

- How ping and pong work
- How query and query hit work
- Packet format
- Node model
- Process model
- Algorithm in Proc state

Node model

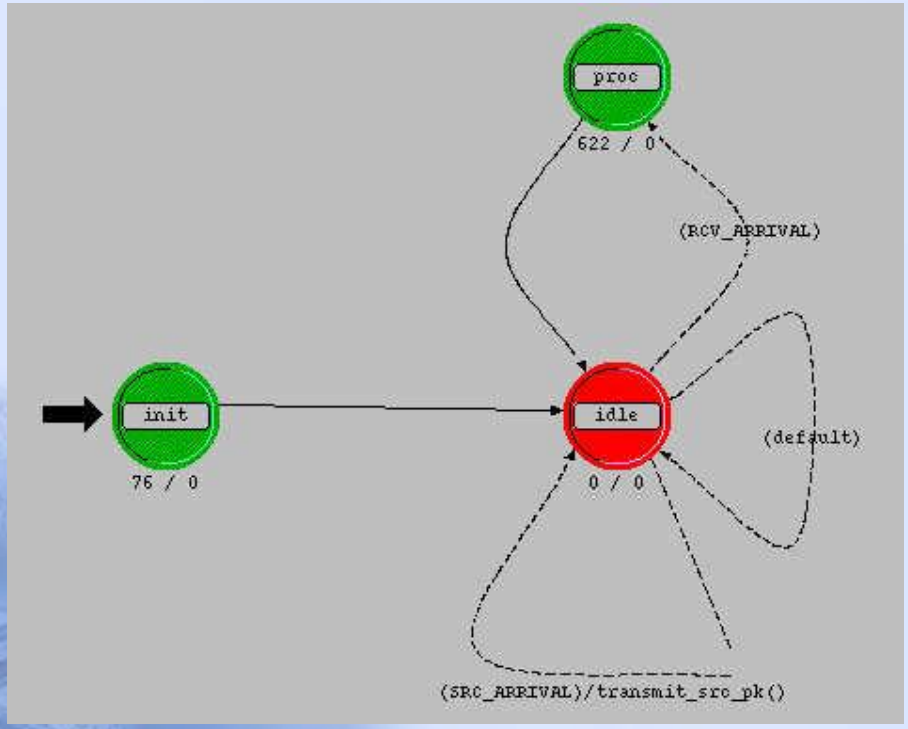


- Src: send ping every second(ping source).
- Proc: manipulate every received packet (packet processor).
- Rcv: receivers.
- Xmt: transmitters.

Implementation details of Gnutella node

- How ping and pong work
- How query and query hit work
- Packet format
- Node model
- Process model
- Algorithm in Proc state

Process model



- If the packet received is from src, then
  - Assign proper value to each field
  - Copy this packet 5 times
  - Send these packets through xmt(0:6)
  - Go back to idle

- If the packet is from one of six rcvs then
  - Processing the packet according to a specific algorithm.
  - Go back to idle

Implementation details of Gnutella node

- How ping and pong work
- How query and query hit work
- Packet format
- Node model
- Process model
- Algorithm in Proc state

Algorithm in proc state

- If the packet is ping
  check if it is a duplicated ping.
  if yes-> destroy ping.
  if not-> 1. save this new ping to cache
  2. generate pong and send it back

  3. if ttl >0
  forward ping to other 5 xmts.
  else destroy ping.

Algorithm in proc state

- If the packet is pong

  check if it is a duplicated pong.

  if yes-> 1. destroy pong.

  if not-> 2. save this new pong to cache

  3. check if it is due to the ping generated by this node.

  if yes-> destroy pong. Generate and send query

  if not-> 1. decode dest_addr to get the tranmitter# where the pong will be forwarded.

  2. update dest_addr.

  3. forward pong through that xmt.

Algorithm in proc state

- If the packet is query

  check if it is a duplicated query.

  if yes-> destroy query.

  if not-> 1. save this new query to cache.

  2. check whether the node's data pool has the desired data.

  if yes-> Generate and send query hit back.

  if not-> check if ttl>0

  if yes-> 1. update TTL and Hops fields in the packet.

  2. copy this packet four times.

  3. forward these five query packets.

Algorithm in proc state

- If the packet is query hit

  check whether it is due to the query generated by this node.

  if yes-> 1. destroy this packet.

  if not-> 1. decode dest_addr to get the tranmitter#
  where the query hit will be forwarded.

  2. update dest_addr.

  3. forward query hit through that xmt.

Two snapshots of the code

/ensc/guest1/yla41/op_models/liu_testgnutella_node_proc.pr.c

File Edit Options

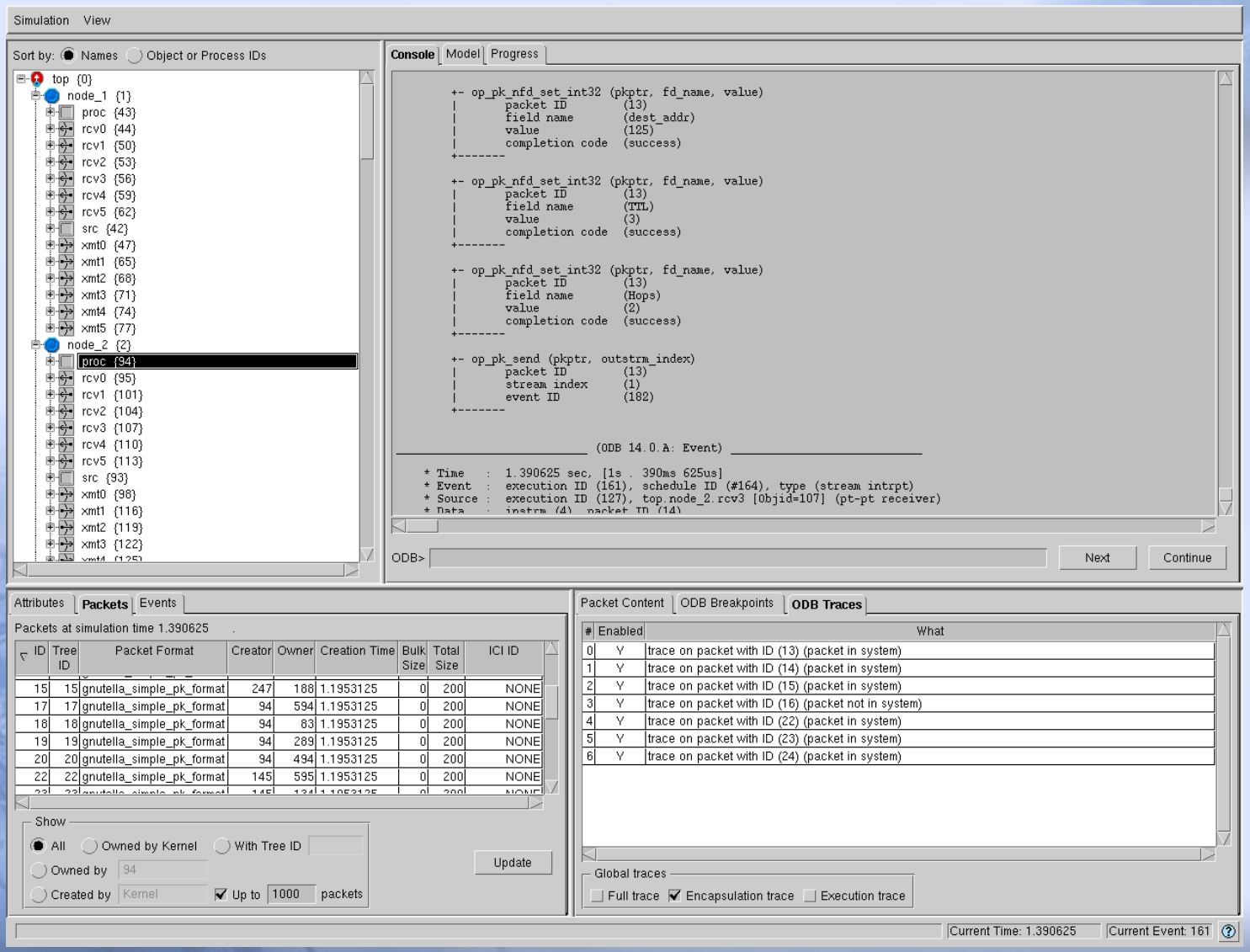
```
/* Process model C form file: liu_testgnutella_node_proc.pr.c */
/* Portions of this file copyright 1992-2007 by OPNET Technologies, Inc. */



/* This variable carries the header into the object file */
const char liu_testgnutella_node_proc_pr_c [] = "MIL_3_Tfile_Hdr_ 140A 30A opnet 7 4BC276F3 4BC276F3 1 cepheus yla41 0 0 none
#include <string.h>



/* OPNET system definitions */
#include <opnet.h>



/* Header Block */

#include <math.h>
/*packet stream definition*/
//in stream
#define src_in 0
#define rcv0_in 1
#define rcv1_in 2
#define rcv2_in 3
#define rcv3_in 4
#define rcv4_in 5
#define rcv5_in 6
//out stream
#define xmt0_out 0
#define xmt1_out 1
#define xmt2_out 2
#define xmt3_out 3
#define xmt4_out 4
#define xmt5_out 5

/*define message ids*/
//payload descriptor value
#define PING 1
#define PONG 2
#define Query 4
#define Query_hit 8

//TTL init
#define TTL_INIT 5

//ping pong cache size
#define CACHE_SIZE 1000
//data cache size
#define DATA_CACHE 1

//packet size
#define PK_SIZE 200.
//define cache structure
typedef struct
	{
		int packet_id;
		//int TTL;
		int Hops;
		int node_id;
		int dest_addr;
		int Payload_des;
		int search;
```

File Edit Options

```
		pong_o_pk_id=op_pk_id(pong_pkptr);

		//op_pk_total_size_set (pong_pkptr,PK_SIZE);
		op_pk_nfd_set_pkid (pong_pkptr,"packet_id", pong_o_pk_id);

		op_prg_odb_bkpt ("1");

		op_pk_nfd_set_int32 (pong_pkptr,"Payload Descriptor", PONG);
		op_pk_nfd_set_int32 (pong_pkptr,"TTL", TTL_INIT); //TTL=HOPS+1
		op_pk_nfd_set_int32 (pong_pkptr,"Hops", 0);
		op_pk_nfd_set_int32 (pong_pkptr,"node_id", node_id);//node_objid);//////
		op_pk_nfd_set_pkid (pong_pkptr,"pong_qh_parentID", o_pk_id);

		op_prg_odb_bkpt ("2");

		op_pk_nfd_set_int32(pong_pkptr,"dest_addr",dest_addr);/*------------------*/
		op_pk_nfd_set_int32 (pong_pkptr,"search", 0);

		//reply ping by using pong
		if(input_rcv_num==rcv0_in)
		    {op_pk_send(pong_pkptr,xmt0_out);}
		else if(input_rcv_num==rcv1_in)
		    {op_pk_send(pong_pkptr,xmt1_out);}
		else if(input_rcv_num==rcv2_in)
		    {op_pk_send(pong_pkptr,xmt2_out);}
		else if(input_rcv_num==rcv3_in)
		    {op_pk_send(pong_pkptr,xmt3_out);}
		else if(input_rcv_num==rcv4_in)
		    {op_pk_send(pong_pkptr,xmt4_out);}
		else if(input_rcv_num==rcv5_in)
		    {op_pk_send(pong_pkptr,xmt5_out);}
		pk_pong_out++;
		pk_total_out++;
		//save new ping
		ping_cache[ping_ptr].packet_id=(int)o_pk_id;
		ping_ptr++;
		if(ping_ptr>=CACHE_SIZE)
		    {
		        ping_ptr=0;
		    }
		//if ttl is ok, update ttl, hops and fwd ping
		if(ttl>0)
		    {
		        int new_dest_addr;
		        //int input;
		        //input=input_rcv_num;
		        new_dest_addr=dest_addr*10+input_rcv_num;//dest_addr+1+1+1+input_rcv_num;
		        op_pk_nfd_set_int32 (pkptr, "dest_addr", new_dest_addr);
		        op_pk_nfd_set_int32(pkptr,"TTL",ttl-1);
		        op_pk_nfd_set_int32(pkptr,"Hops",hops+1);
		        //copy packet 4 times
		        pkptr_1=op_pk_create_fmt("gnutella_simple_pk_format");

		        op_pk_nfd_set_pkid (pkptr_1, "packet_id", o_pk_id);
		        op_prg_odb_bkpt ("3");

		        op_pk_nfd_set_int32 (pkptr_1, "Payload Descriptor", PING);
		        op_pk_nfd_set_int32 (pkptr_1, "TTL", ttl-1);
		        op_pk_nfd_set_int32 (pkptr_1, "Hops", hops+1);
		        op_pk_nfd_set_int32 (pkptr_1,"node_id", node_id);//node_objid);//////////////
		        op_pk_nfd_set_int32 (pkptr_1, "dest_addr", new_dest_addr);//////////////////
		        op_pk_nfd_set_int32 (pkptr_1,"search",search);//////////
		        op_pk_nfd_set_pkid (pkptr_1,"pong_qh_parentID",pong_qh_parentID);

		        op_prg_odb_bkpt ("4");

		        pkptr_2=op_pk_create_fmt("gnutella_simple_pk_format");
		        op_pk_nfd_set_pkid (pkptr_2, "packet_id", o_pk_id);
		        op_pk_nfd_set_int32 (pkptr_2, "Payload Descriptor", PING);
```

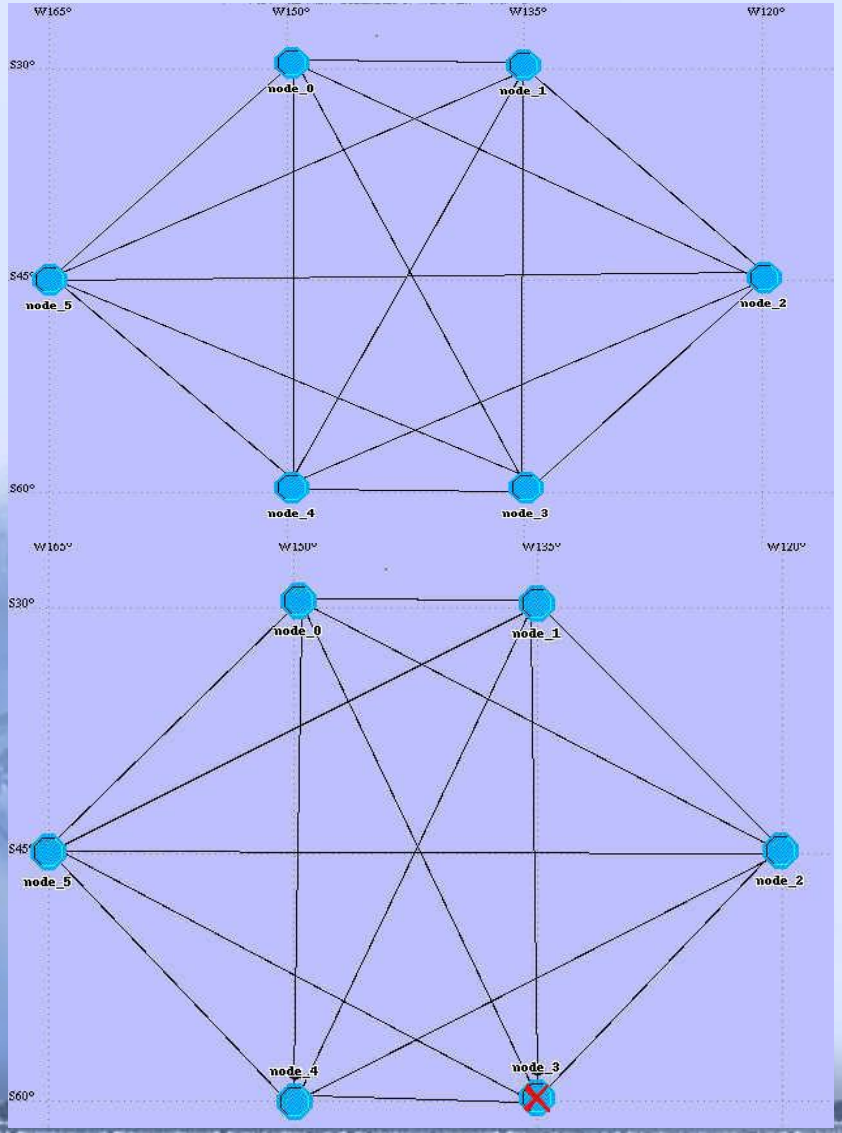
Line: 1

Debugging mode of simulation



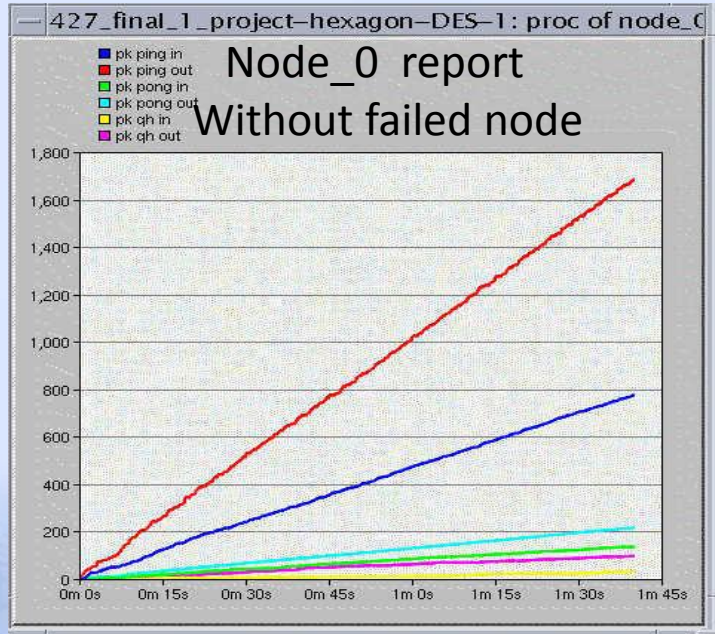
Scenarios and simulation results

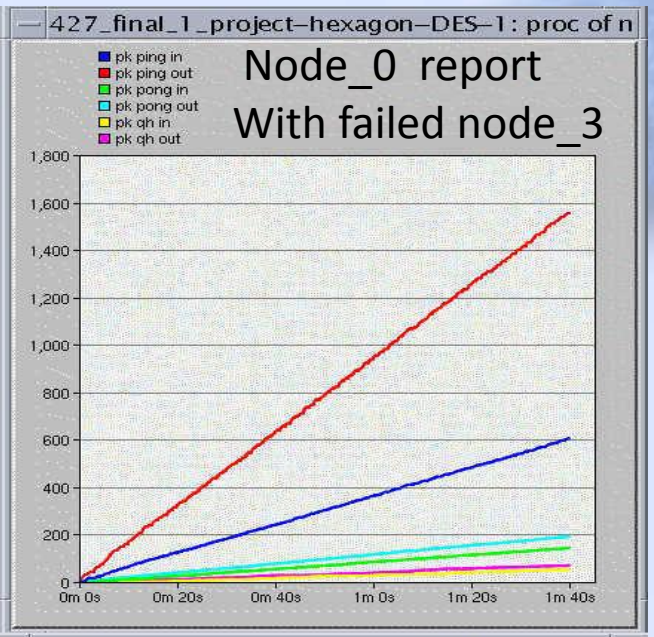
Hexagon Topology

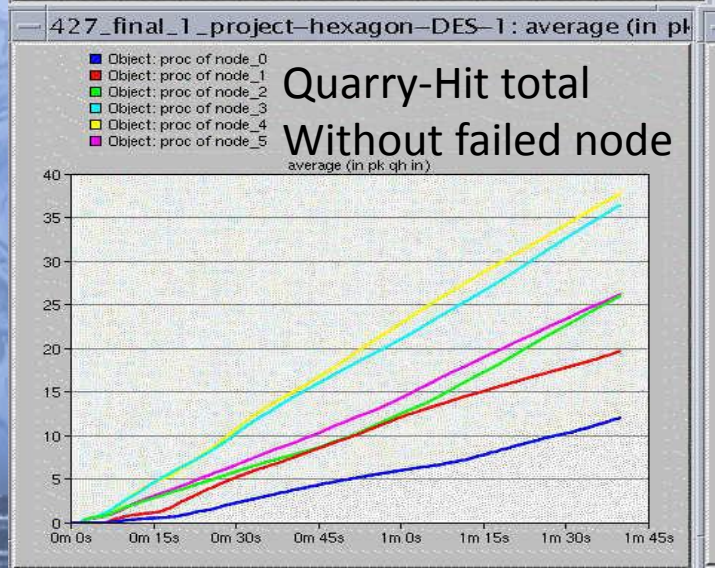


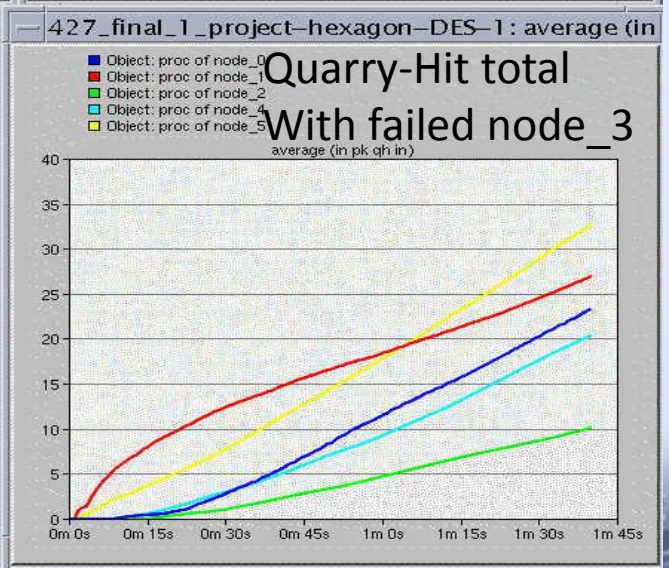
- Basic P2P Topology type
- Every node have the same configuration
- Every node generate its own “PING”
- Failed node does not effect function of other nodes

Hexagon Topology Simulation Results

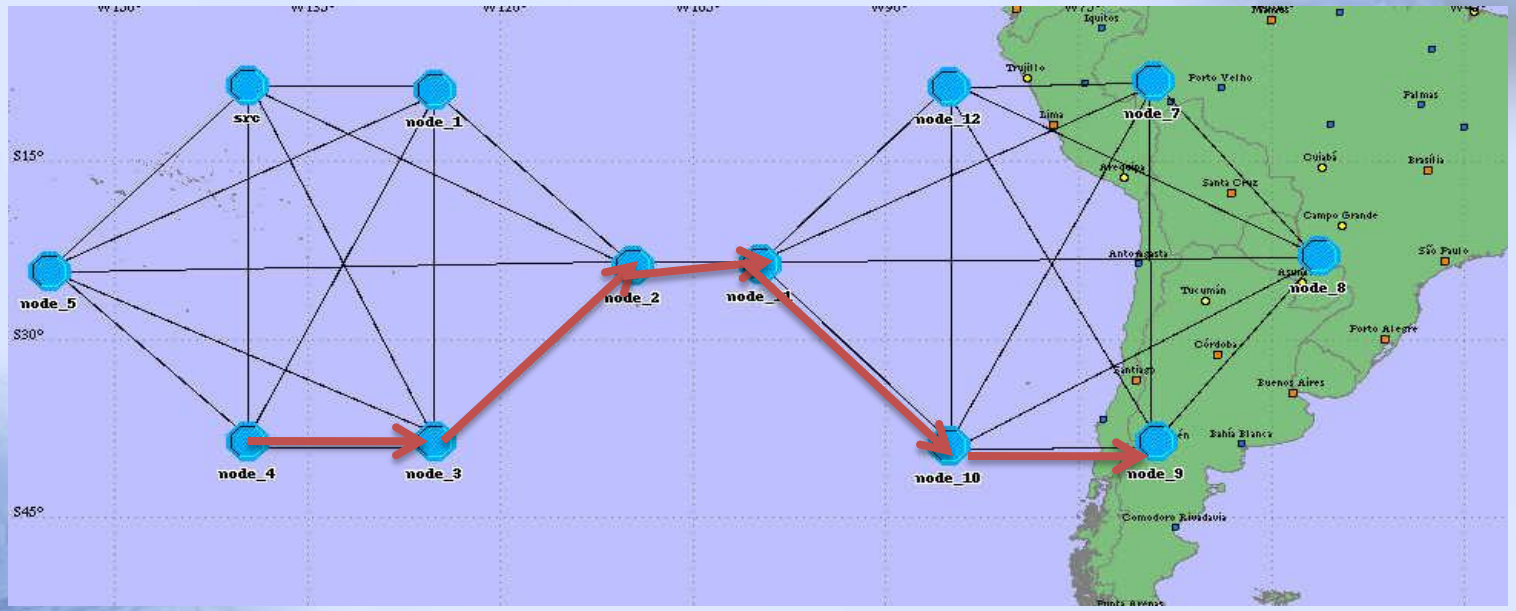






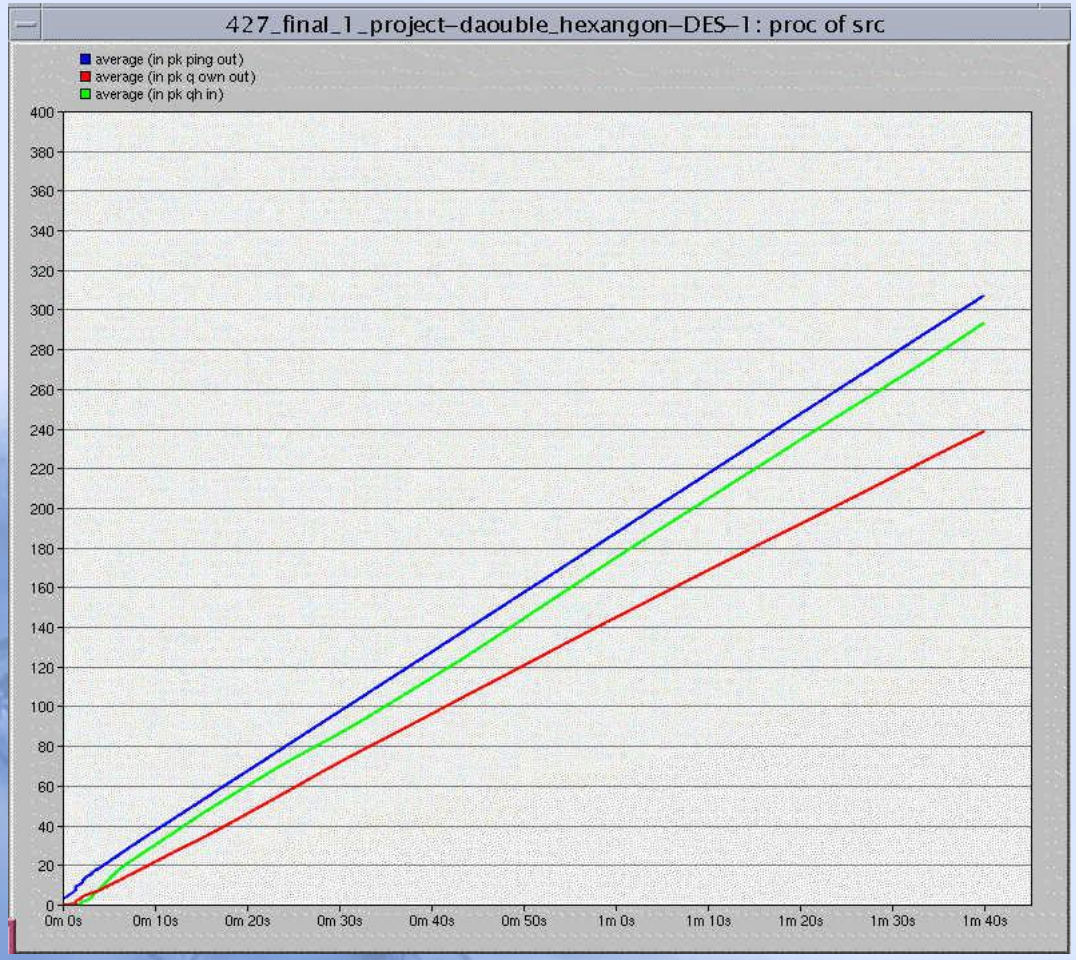


Duplicated Hexagon



- Each nodes can be reach by another one within 5 steps
- Can be viewed as two sub nets
- Only SRC_node generate ping
- Test SRC_node have data access from sub-net B

Duplicated Hexagon results

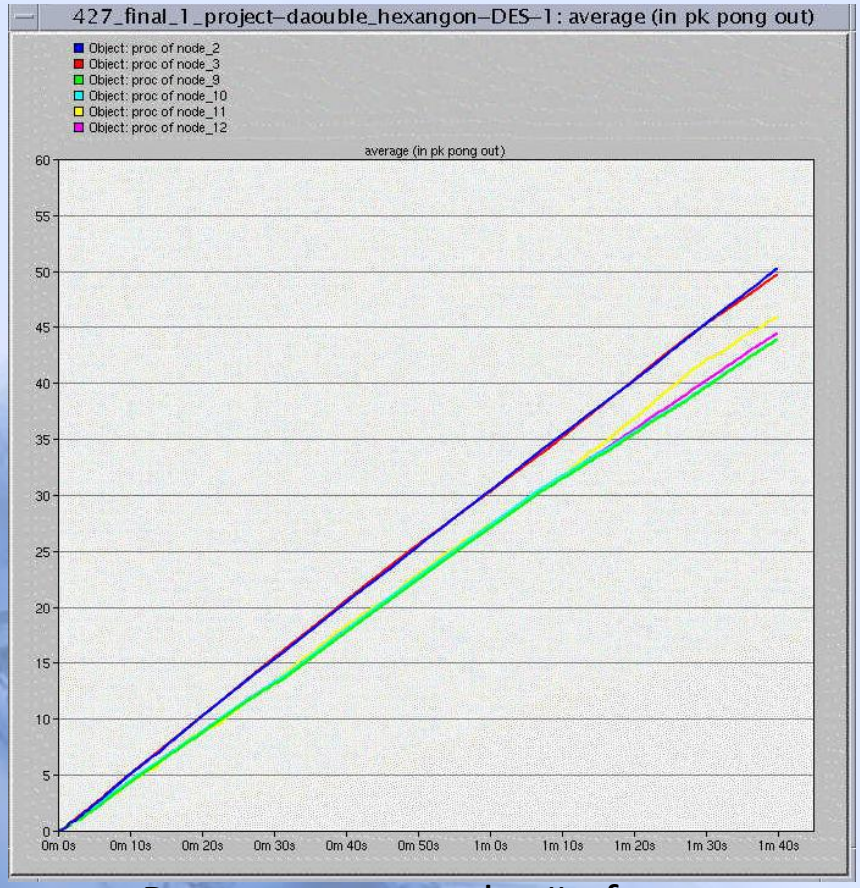


SRC_node results

1. Src ping out (blue)
2. **q_own_out** (red)is the output number of **query** that response to **pong** come in, not include the forwarded **query** packets
3. **qh** (green) is the input number of **quarry hit**

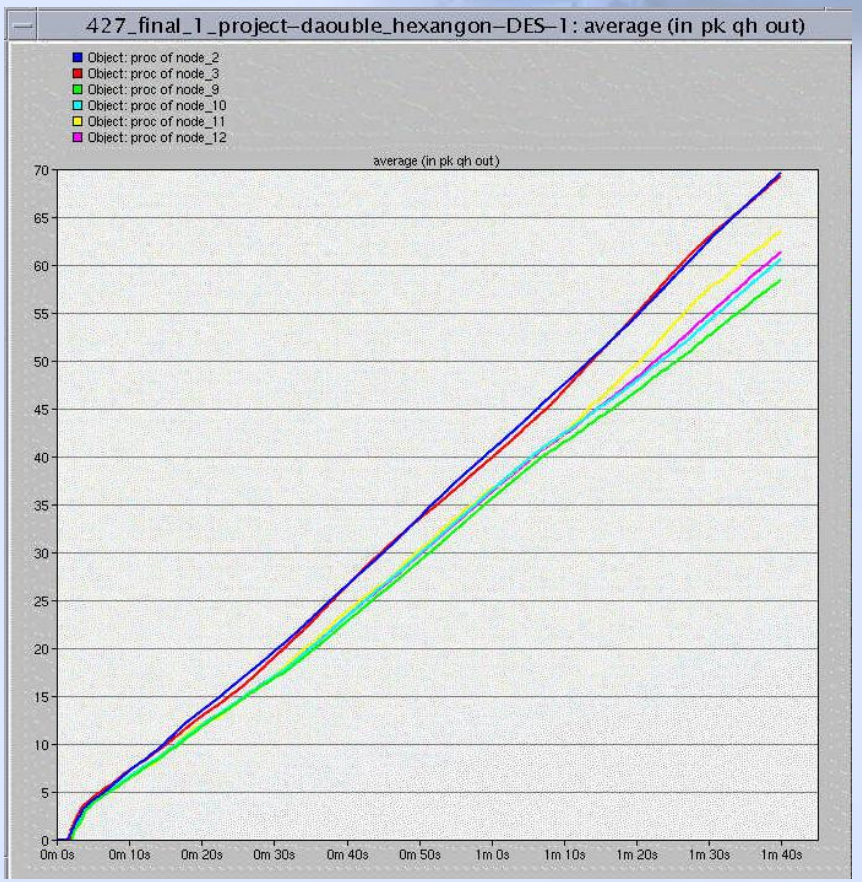
Duplicated Hexagon results

Pong out put numbers



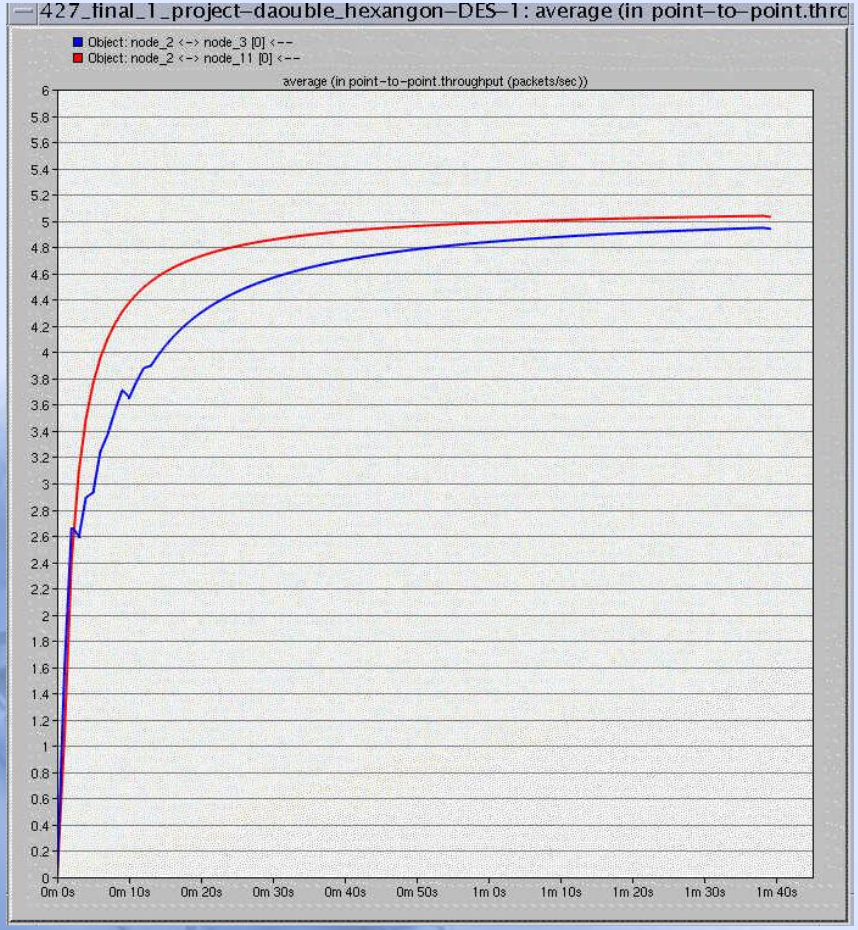
Pong out means the # of possible connections

Quarry hit out put numbers



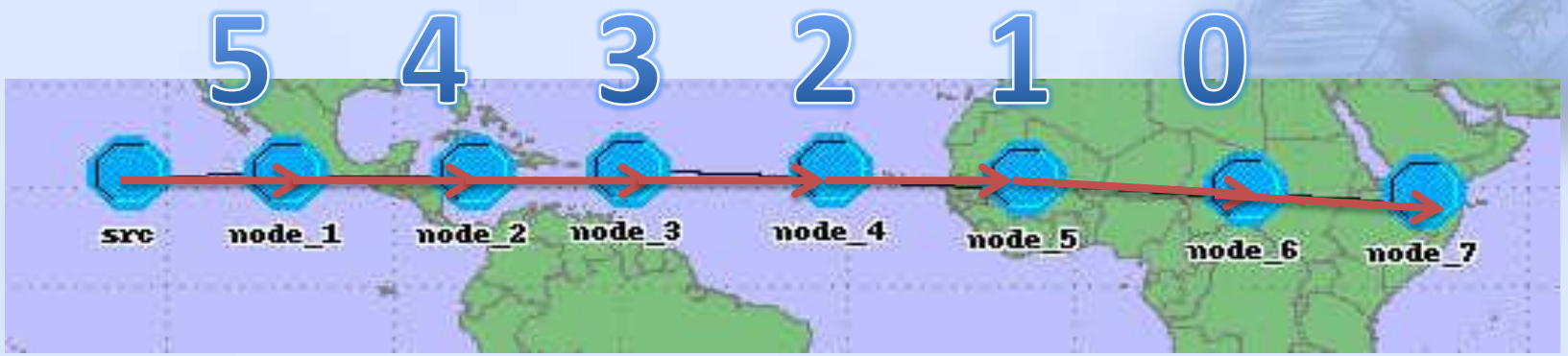
Quarry hit out means the # of required data are available

Duplicated Hexagon results



- Node 2 <-> node 3 (blue) are link with in a same sub net
- Node 2<-> node 11(red) are the link connect two subnet together
- Expect higher throughput for link connect two subnet together
- coincide with simulation

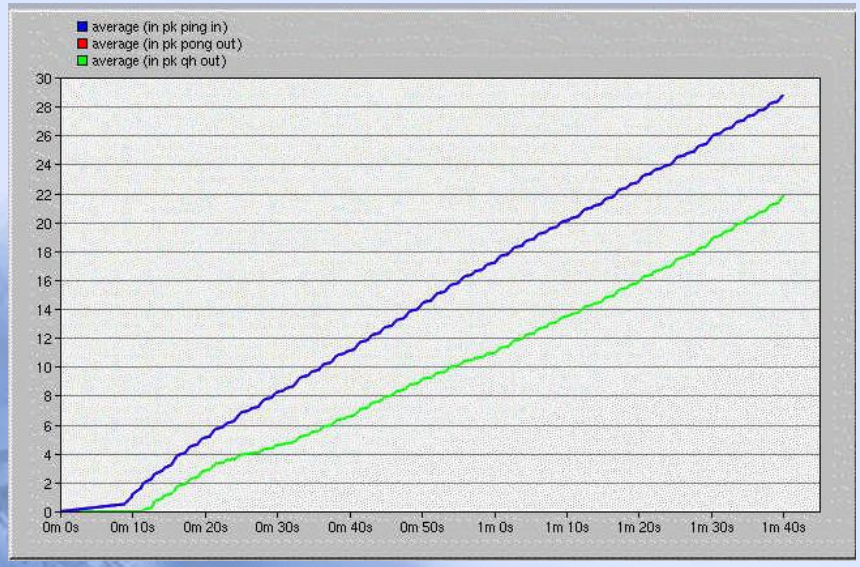
Line Topology



- Single line connect for 8 nodes
- Single SRC node in the beginning (only this node **Ping** out)
- Used to test TTL (Time-to-Live)
- Two different TTL are simulated (5 vs 50)
- Expect no packets received or transmitted for node_7 when TTL = 5

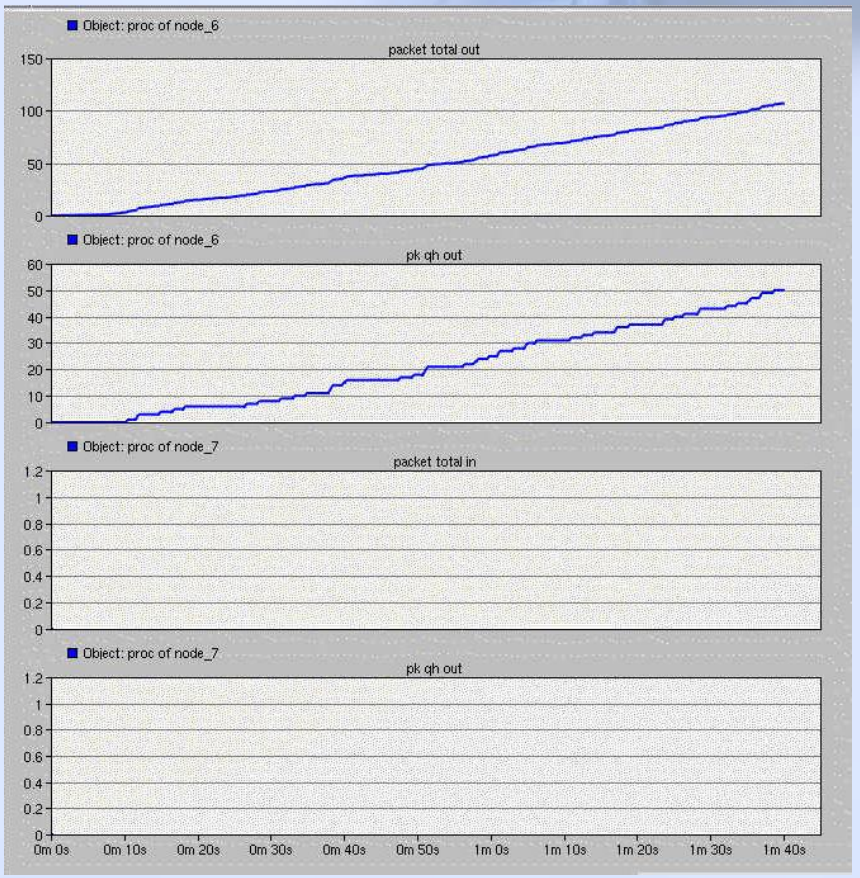
Line Topology simulation results

TTL = 50 node_7

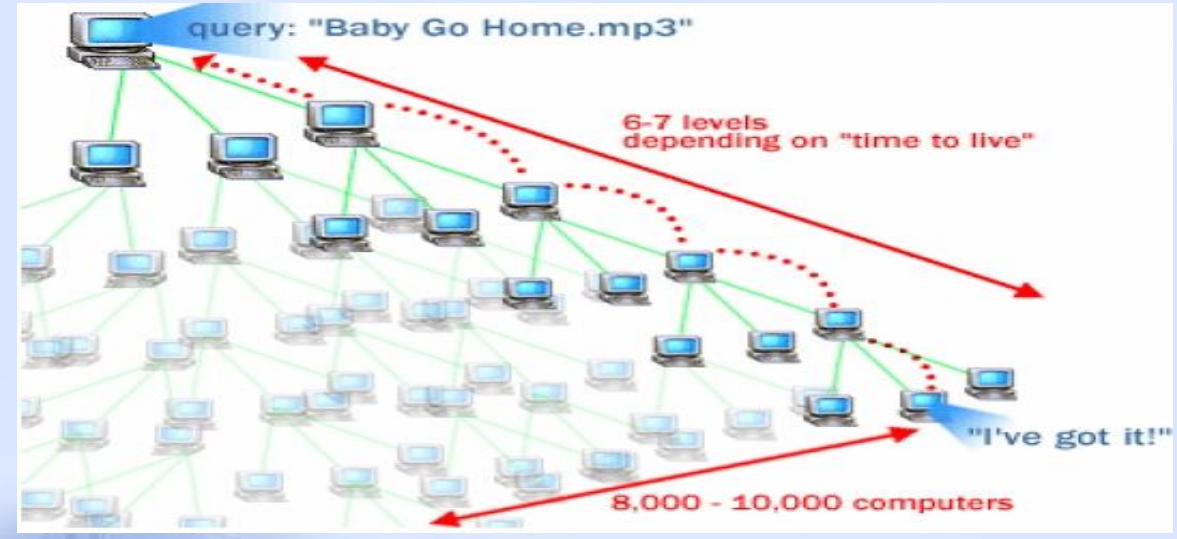


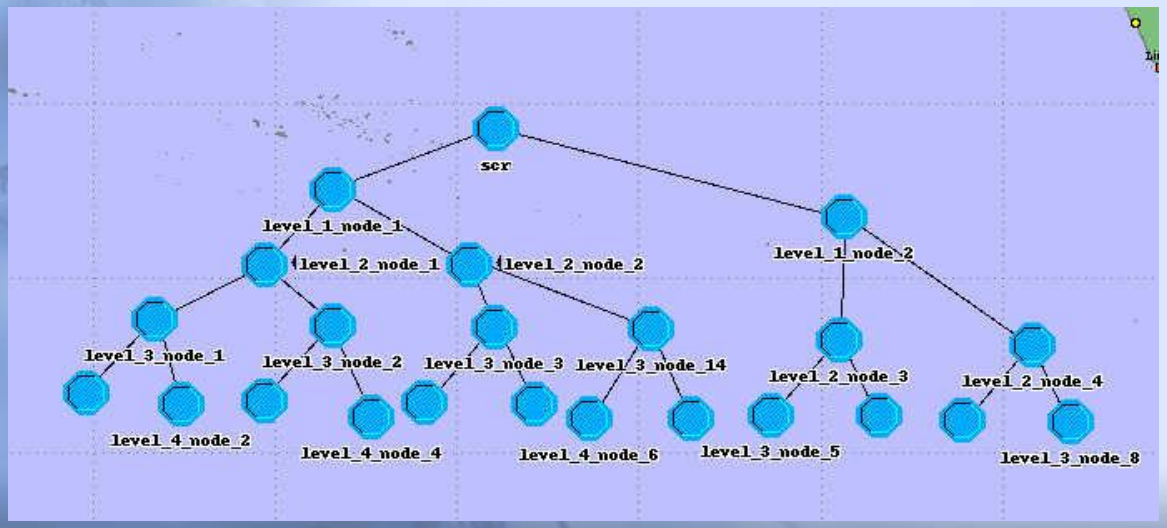
# of Ping (in) = # of Ping(out)

TTL = 5 node_7 and node_6



Tree Topology

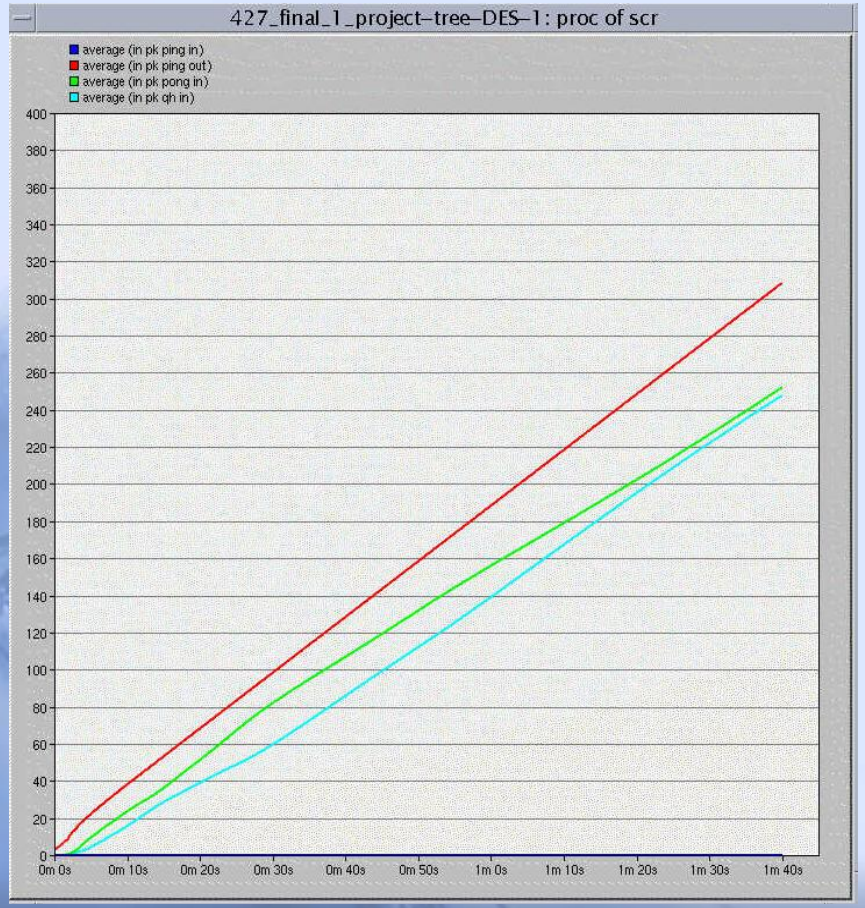




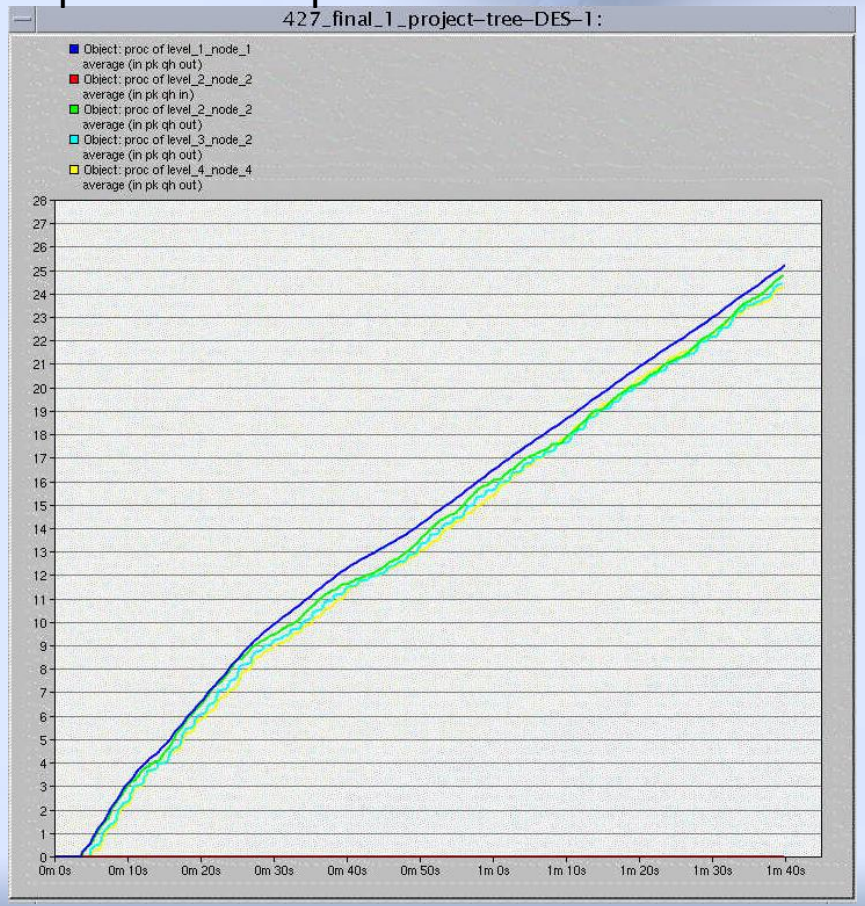
- More realistic, more closer to open source file sharing network
- Use to test successfulness of Flooding search method
- Only top SRC node generate PING packets
- 4 level setup with each node derived out two nodes down

Tree Topology Simulation Results

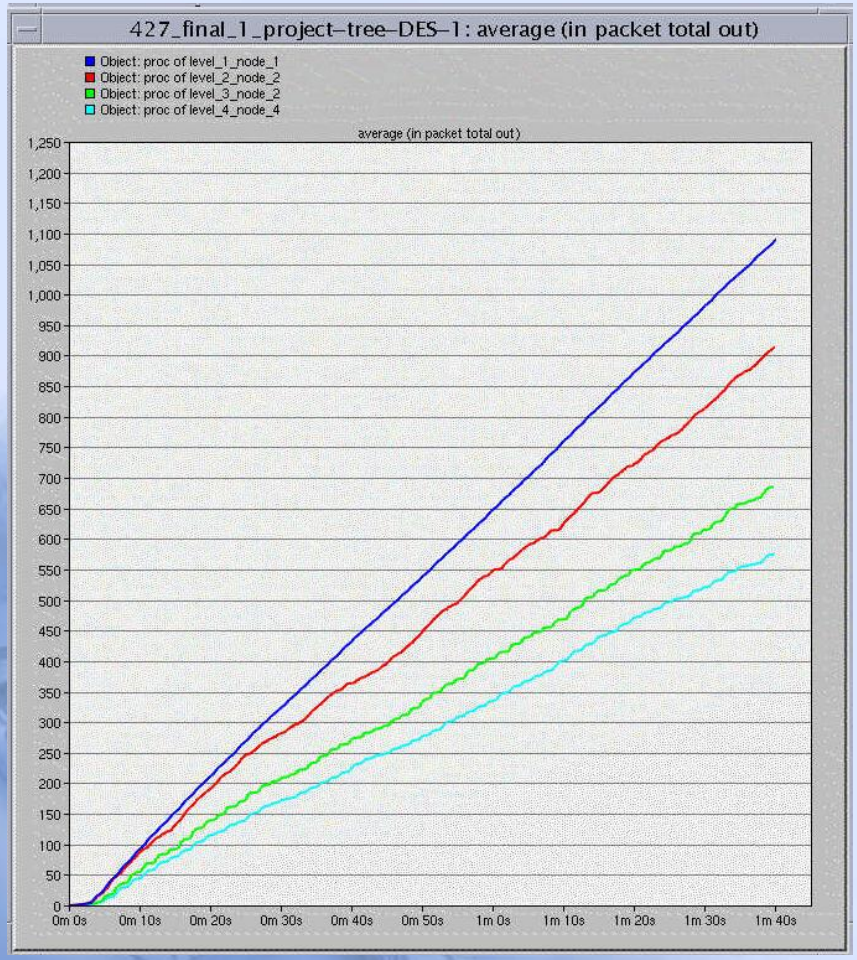
SCR_Node results



Level 1 to level 4 signle_node QH packets compare



Tree Topology Simulation Results



- The pk_total_out records the total number for all types of packets output form a single node
- Indicate the level traffic for each node
- During 100s, only 100 ping packets goes out from SRC node, but results at least 600 packets output from each node

Conclusion

- Gnutella is practical for small networks with few requests
- A larger network would generate far more traffic per node than a smaller one, making it inherently unscalable

Future work

- Scalable solution
- Dynamic simulation
- Add Push descriptors in model

References

- [1] R.Schollmeier, "*A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Application"s*, Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE (2002).
- [2] E.Bangeman, "*Ars Technica Study: BitTorrent sees big growth*" [Online]. Available: http://arstechnica.com/old/content/2008/04/study-bittorren-sees-big-growth-limewire-still-1-p2p-app.ars [Accessed: March. 15, 2010].
- [3] T. Mennecke *"Slyck News- eDonkey 2000 Nearly Double the Size of FastTrack"* [Online]. Available: http://www.slyck.com/news.php?story=814 [Accessed: March 15th, 2010].
- [4] A.Rasti, D.Stutzbach and R.Rejaie "*On the Long-term Evolution of the Two-Tier Gnutella Overlay*" University of Oregon. P.2.
- [5] J. Cardoso and M. Lytras, *Semantic Web engineering in the knowledge society,* Hershey, PA ,2009.
- [6] L. Alfred and W. Sing, *Peer-to-peer computing : building supercomputers with Web technologies / Alfred Wai-Sing Loo.* Springer : London, 2007.
- [7] A . Dufour, *Improving the performance of the Gnutella network.* Simon Fraser University: Burnaby B.C 2006.
- [8] "The Gnutella specification v0.4". www9.limewire.com/developer/gnutella_protocol_0.4.pdf. [Accessed: March. 15,2010].
- [9] S. Osama, *Modeling and caching of peer-to-peer traffic.* Burnaby B.C: Simon Fraser University, 2006.

References

- [10] D. Worthington; M. Nate (25 May 2005). "BitTorrent Creator Opens Online Search". BetaNews [Online]. Available: http://www.betanews.com/article/BitTorrent_Creator_Opens_Online_Search/1117065427. [Accessed: March. 05,2010].
- [11] H.Su and K.Wu, “Gnutella Network Robustness,” *ensc.sfu.ca*, report. Spring 2009. [Online]. Available: http://www.ensc.sfu.ca/~ljilja/ENSC427/Spring09/Projects/team12/Gnutella_Network_Robustness_Final_Version.pdf [Accessed: March. 05, 2010].

Thank you!

- Questions?