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, 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

How Ping and Pong work

How Ping and Pong work

How Ping and Pong work

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

How ping and pong work

How ping and pong work

How ping and pong work

How ping and pong work

How ping and pong work

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

How Query, Query Hit work

How Query, Query Hit work

How Query, Query Hit work

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

How Query, Query Hit work

How Query, Query Hit work

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 transmitter#
          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 transmitter#
           where the query hit will be forwarded.
           2. update dest_addr.
           3. forward query hit through that xmt.
Two snapshots of the code
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

Node_0 report
Without failed node

Quarry-Hit total
Without failed node

Node_0 report
With failed node_3

Quarry-Hit total
With failed node_3
Each node can be reached 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

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

Quarry hit out put numbers

Pong out means the # of possible connections

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

TTL = 5 node_7 and node_6

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


References


Thank you!

• Questions?