Improving Gnutella

Willy Henrique Säuberli
Seminar in Distributed Computing, 16. November 2005

Papers:

I. Making Gnutella-like P2P Systems Scalable; SIGCOMM 2003
II. Peer-to-Peer Overlays: Structured, Unstructured, or Both? MSR-TR-2004-73 2004
III. Should We Build Gnutella on a Structured Overlay? HotNets-II 2004

Motivation

In the **spring of 2000**, when **Gnutella was a hot topic** on everyone's mind, a concerned few of us in the open-source community just sat back and shook our heads. Something just wasn't right. **Any competent network engineer that observed** a running gnutella application would tell you, through simple empirical observation alone, that **the application was an incredible burden on modern networks and would probably never scale**. I myself was just stupefied at the gross abuse of my limited bandwidth,

Jordan Ritter - *Why Gnutella Can't Scale. No, Really.*

Overview

- Systems
  - Gnutella 0.4
  - Gnutella 0.6
  - Pastry/DHT (Distributed Hash Table)
- Gia
  - Topology adaptation
  - Flow Control
  - One-hop Replication
  - Search Protocol
  - Evaluation
- Structural Gnutella
  - Overhead of maintaining structured/unstructured overlay
  - Overhead of queries in structured/unstructured overlay
- Conclusions

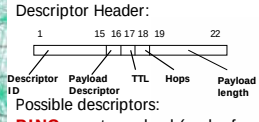
Gnutella 0.4

Original Gnutella Specification:

- Acquisition of addresses is not part of the protocol
  -> Host cache services predominant way
- TCP/IP connection to servant and ASCII string sent:
  `GNUTELLA CONNECT/<protocol version string>\n\n`
- Servant response
  `GNUTELLA OK\n\n` (anything else interpreted as rejection)
- Sending of any of Gnutella protocol descriptors
- -> file requests done over http requests

Gnutella 0.4

Gnutella Protocol descriptors:
Descriptor Header:



Possible descriptors:

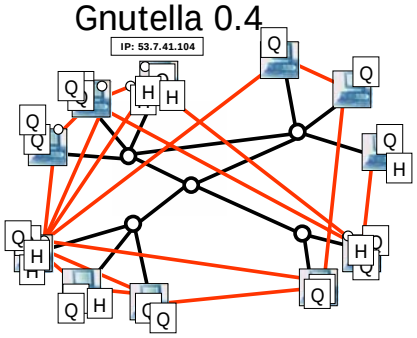
PING: empty payload (probe for servants)
PONG: port, IP,#files,#KB (response to PING)
QUERY: minimum speed, search criteria
QUERYHIT: #hits, port, IP, speed, result set, servant identifier
PUSH: servant identifier, file index, port, IP (if firewalled)

Gnutella 0.4

Descriptor Routing

- PONG carried along same path like PING
- QueryHit carried along same path like Query
- PUSH carried along same path like QueryHit
- PING and Query forwarded to all connected servants, except the one that sent
- Servant decrements TTL and increments Hops field
- Servants avoids forwarding descriptors with ID already seen.

Gnutella 0.4



7

Gnutella 0.4

Problems

1. **Flooding** -> queries received several times
2. **Churn** -> high rate of joining and leaving
3. **Node Overloading** -> to much connections
4. **No bootstrapping** in protocol (mostly done central)
5. **No load balancing** -> queries, downloads

8

Overview

- Systems
  - Gnutella 0.4
  - Gnutella 0.6
  - Pastry/DHT (Distributed Hash Table)
- Gia
  - Topology adaptation
  - Flow Control
  - One-hop Replication
  - Search Protocol
  - Evaluation
- Structural Gnutella
  - Overhead of maintaining structured/unstructured overlay
  - Overhead of queries in structured/unstructured overlay
- Conclusions

9

Gnutella 0.6

- The Ultra peer system has been found effective for this purpose. It is a scheme to have a hierarchical Gnutella network by categorizing the nodes on the network as leaves and ultra peers. A leaf keeps only a small number of connections open, and that is to ultra peers. An ultra peer acts as a proxy to the Gnutella network for the leaves connected to it. This has an effect of making the Gnutella network scale, by reducing the number of nodes on the network involved in message handling and routing, as well as reducing the actual traffic among them.

RFC-Gnutella 0.6 - Chapter 2.3, *Leaf Mode and Ultrapeer Mode*

10

Gnutella 0.6

Improvements:

- GWebCache for addresses
- X-Try header (for rejected connection)
- host addresses stored in pong messages
- store addresses from QueryHit in local cache
- Nodes classified as Peers and Leaves

11

Gnutella 0.6

requirements for Ultrapeers:

- no firewall
- suitable operating system
- sufficient bandwidth
- sufficient uptime
- sufficient RAM and CPU

12

Overview

- Systems
  - Gnutella 0.4
  - Gnutella 0.6
  - Pastry/DHT (Distributed Hash Table)
- Gia
  - Topology adaptation
  - Flow Control
  - One-hop Replication
  - Search Protocol
  - Evaluation
- Structural Gnutella
  - Overhead of maintaining structured/unstructured overlay
  - Overhead of queries in structured/unstructured overlay
- Conclusions

13

Pastry/DHT

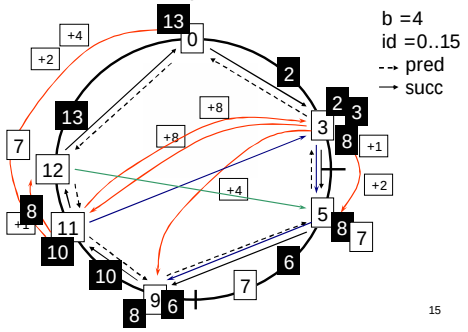
- peers distributed on Ring structure
- peers id computed with hash function of IP
- successor: next peer in id space
- predecessor: last peer in id space
- files matched to nodes with hash function

Chord:

- id space of 2^b, e.g. b=128
- additional pointer to all peers with address id+2^i, i=0..b-1

14

Pastry/DHT

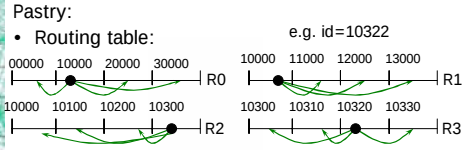


15

Pastry/DHT

Pastry:

- Routing table:



- Joining of node n:
  - join over node s
  - copy of s routing table
  - copy of i-th row of node n to message to nodes in row i
- Leaving: failure detection, copy value of neighbour

16

Pastry/DHT

Problem of DHT:

- failure causes loss of items and disconnection in ring
  -> each peer keeps list of $\log_2(N)$ next nodes
  ->files replicated in successors
- not designed for heterogeneous network
  ->files distribution independent of capacity
- designed for exact word queries

17

Overview

- Systems
  - Gnutella 0.4
  - Gnutella 0.6
  - Pastry/DHT (Distributed Hash Table)
- Gia
  - Topology adaptation
  - Flow Control
  - One-hop Replication
  - Search Protocol
  - Evaluation
- Structural Gnutella
  - Overhead of maintaining structured/unstructured overlay
  - Overhead of queries in structured/unstructured overlay
- Conclusions

18

Gia Design

Design:

- dynamic topology adaptation:
  Most nodes within short range of high capacity node
- active flow control
  avoid overloaded hot-spots
- one-hop replication
  all nodes maintain pointers to content of neighbours
- search protocol
  biased random walks directed to high-capacity nodes

19

Gia – Topology Adaptation

Topology adaptation

- High capacity <-> high degree (~supernodes)
  - level of satisfaction:
    Minimum/maximum number of connections
    prefer neighbours with higher capacity and lower degree
    drop neighbours with highest degree

20

Gia – Topology Adaptation

Let C_i represent capacity of node i
if $num\_nbrs_X + 1 \leq max\_nbrs$ **then** {we have room}
  ACCEPT Y; return

{we need to drop a neighbor}
$subset \leftarrow i \ \forall i \in nbrs_X$ such that $C_i \leq C_Y$
if no such neighbors exist **then**
  REJECT Y; return
candidate $Z \leftarrow$ highest-degree neighbor from $subset$

if $(C_Y > max(C_i \forall i \in nbrs_X))$ {Y has higher capacity}
or $(num\_nbrs_Z > num\_nbrs_Y + H)$ {Y has fewer nbrs}
then
  DROP Z; ACCEPT Y
else
  REJECT Y

Algorithm 1: *pick_neighbor_to_drop*(X, Y):
When node X tries to add Y as a new neighbor, determine whether there is room for Y. If not, pick one of X's existing neighbors to drop and replace it with Y. (In the algorithm, H represents a hysteresis factor.)

21

Gia - Flow control

Flow Control

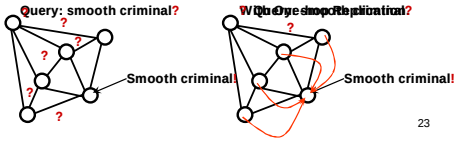
- peers periodically assign tokens to neighbours
  - queries only forwarded if token received
    -> overloaded nodes stop receiving queries
  - token proportionally to capacity
    -> more capacity, more queries can be sent
    -> more queries from nodes with high capacity
  - peers not using tokens are marked as inactive
    -> get less tokens

22

Gia – One-hop Replication

One-hop Replication

- peers keep index of files at neighbours
  -> response to queries includes files at neighbour
- peers keep copy of files at neighbours
  -> paper tried to improve network structure and network querying. Copy of file would improve availability



23

Gia Search Protocol

Search Protocol

- Random walk instead of flooding
- Query forwarded to neighbour with highest capacity
- Book-keeping of queries to avoid redundant paths
  - node remembers paths used
  - query only forwarded if MAX_RESPONSES not reached
  - addresses of nodes already mentioned in Query Hit attached to query

24

Evaluation Gia

Reference Systems:

- FLOOD: search flooding network
- RWRT: Random Walk over Random Topology
- SUPER: nodes classified as normal or supernode

25

Evaluation Gia

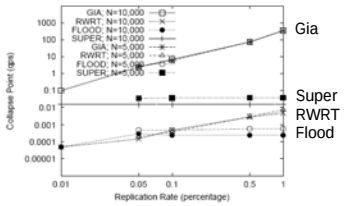


Figure 3: Comparison of collapse point for the different algorithms at varying replication rates and different system sizes.

26

Evaluation Gia

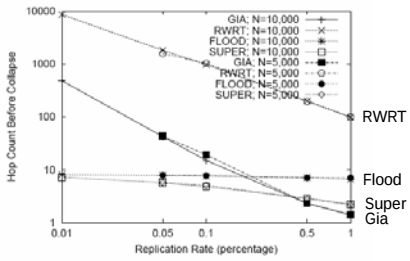


Figure 4: Hop-count before collapse.

27

Evaluation Gia

- RWRT better than FLOOD, specially high replication factor
- Extremely low hop-counts at higher replication rate
- Performance of FLOOD decreases with system size

28

Evaluation Gia

How to handle churn

- Failure in network may lead to loss of query
  - Keep-alive messages
  - query reissued if no keep alive-messages received
  - to avoid loss of queries do to adaptation, paths are kept for a while, to reroute queryHits

29

Gia Network is unstructured

Why not DHTS/keep network unstructured?

1. P2P clients are extremely transient (ø 60 min.)
2. Keyword search more often than exact-match
3. Designed to improve query performance, but most queries are for hay not needle
4. DHT maps files to users (not a user decision)
5. Don‘t support complex queries
6. Don‘t cope with churn (high overhead for leaving)

30

Overview

- Systems
  - Gnutella 0.4
  - Gnutella 0.6
  - Pastry/DHT (Distributed Hash Table)
- Gia
  - Topology adaptation
  - Flow Control
  - One-hop Replication
  - Search Protocol
  - Evaluation
- Structural Gnutella
  - Overhead of maintaining structured/unstructured overlay
  - Overhead of queries in structured/unstructured overlay
- Conclusions

31

Structured overlay

Gnutella 0.4 improved with Pastry network structure

- up to 32 peers in network table
- Boostrapping like in Pastry
- I'm alive for failure

Results

- Pastry maintains more neighbours
- overhead between 0.4(4) and 0.4(8)
- overhead grows with network size, but slowly
- overhead negligible for all systems

32

Structella - Maintenance

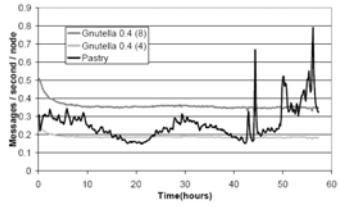


Figure 1: Maintenance overhead in messages per second per node over time for the Gnutella 0.4 and Pastry graphs.

33

Structured overlay

Gnutella 0.6 improved with Pastry network structure

- supernodes implemented in network
  - supernodes organized in pastry network
  - normal nodes attached randomly to supernodes

Gia improved with Pastry network structure

- Builds network with pastry structure based on gia neighbour selection principles (satisfaction)

34

Superpasty - Maintenance

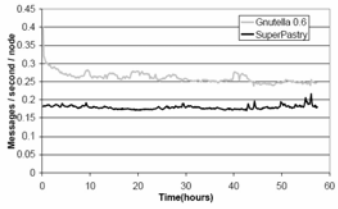


Figure 2: Maintenance overhead in messages per second per node over time for the two graphs using super-peers.

35

HeteroPastry -Maintenance

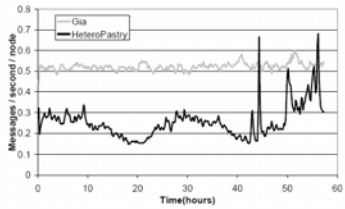


Figure 3: Maintenance overhead in messages per second per node over time for Gia and HeteroPastry.

36

Overview

- Systems
  - Gnutella 0.4
  - Gnutella 0.6
  - Pastry/DHT (Distributed Hash Table)
- Gia
  - Topology adaptation
  - Flow Control
  - One-hop Replication
  - Search Protocol
  - Evaluation
- Structural Gnutella
  - Overhead of maintaining structured/unstructured overlay
  - Overhead of queries in structured/unstructured overlay
- Conclusions

37

Structured overlay

Results presented only considered **overhead for maintain structure**.

Explore advantages of structured overlays using querying advantages of Gia network

- structure helps avoiding that queries visit nodes several times
- route queries to nodes with higher capacity

38

Pastry – Query overhead

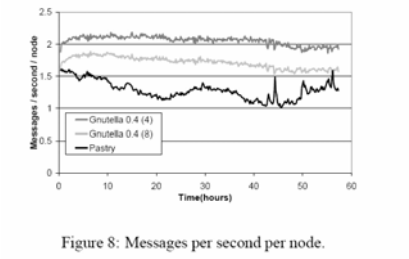


Figure 8: Messages per second per node.

39

Pastry – Success rate

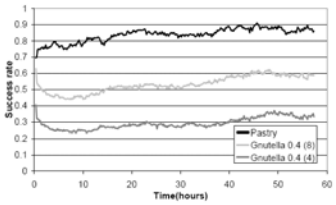


Figure 9: Query success rate.

40

Structured overlay

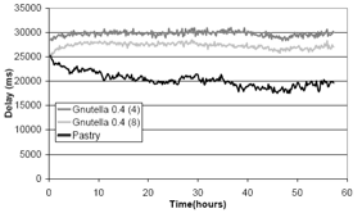


Figure 10: Query delay for successful queries.

41

HP/SP - success rate

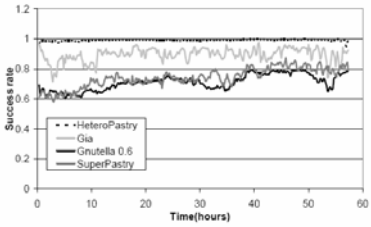


Figure 11: Query success rate.

42

HP/SP – Query delay

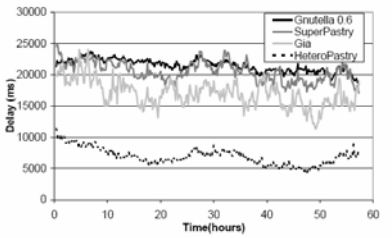


Figure 12: Query delay for successful queries.

43

HP/SP – Query overhead

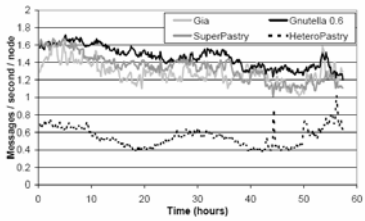


Figure 13: Messages per second per node.

44

Conclusions

- **Most work experimental**
  - Gia introduces several techniques that help efficiency
- **Problems to deal:**
  - High rate of churn
  - High heterogeneity of nodes in bandwidth, query rate, CPU, RAM, availability
  - different configurations lead to different solutions
- **Structures**
  - not a solution, but may help improve efficiency
- **Implementation for results on real network:**
  - legal issues
  - highly distributed system
  - no control of single peers in real environment

45

Sources

- Original Gnutella 0.4 specification:
  http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf
- RFC-Gnutella 0.6
  http://rfc-gnutella.sourceforge.net/developer/testing/index.html
- Pastry/DHT
  Jie Wu; Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, Chapter 39
- Papers:
  - Making Gnutella-like P2P Systems Scalable.
    Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker
  - Peer-to-Peer Overlays: Structured, Unstructured, or Both?
    Miguel Castro, Manuel Costa and Antony Rowstron
  - Should We Build Gnutella on a Structured Overlay?
    M. Castro, M. Costa, A. Rowstron
  - Why Gnutella Can't Scale. No, Really.
    Jordan Ritter

46