600.413 Topics in P2P Network Systems

Andreas Terzis

terzis@cs.jhu.edu

With lots of help from Jim Kurose, Brian Levine, Jon Crowcroft

Logistics

- Short seminar
  - Class meets Tues-Thurs 4:00-5:15
  - 8 Lectures
- Website http://www.cs.jhu.edu/~terzis/600.413/
  - Announcements go there
  - You are required to check daily
- Grading
  - 1-page summary of assigned papers before the lecture
    - No late assignments!
  - In-class final on 4/24

Outline

- What we are going to cover
  - What P2P systems are and how are they different from client-server apps
  - Evolution of P2P designs
  - Applications built on top of P2P systems
  - Implications of P2P systems
  - Alternative architectures

Background

- "Traditional" Internet applications are based on the Client-Server model
  - World Wide Web/HTTP
  - Usenet News/NNTP
  - FTP
- Architecture
  - Dedicated servers
    - Server farms used for scalability
  - Clients run different software from servers

What is a P2P system?

- A distributed system architecture:
  - No centralized control
  - Nodes are symmetric in function
- Large number of unreliable nodes
- Enabled by technology improvements
- However, this means they need distributed algorithms for
  - Service discovery
  - Neighbor status tracking
  - Application layer routing
  - Resilience, handling link and node failures

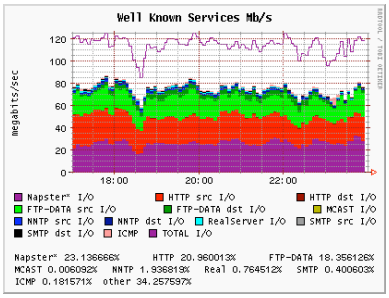
(Potential) Advantages

- Harness the resources of millions of PCs
  - Many disks, CPUs, network connections
- Easier to deploy
  - Incentive to deploy app
- Easier to administer
- More robust
  - Many replicas
  - Geographic distribution

Napster

- Program for sharing files over the Internet
- A "disruptive" application/technology?
- History:
  - 5/99: Shawn Fanning (freshman, Northeasten U.) founds Napster Online music service
  - 12/99: first lawsuit
  - 3/00: 25% UWisc traffic Napster
  - 2000: est. 60M users
  - 2/01: US Circuit Court of Appeals: Napster knew users violating copyright laws
  - 7/01: # simultaneous online users:

    Napster 160K, Gnutella: 40K, Morpheus: 300K

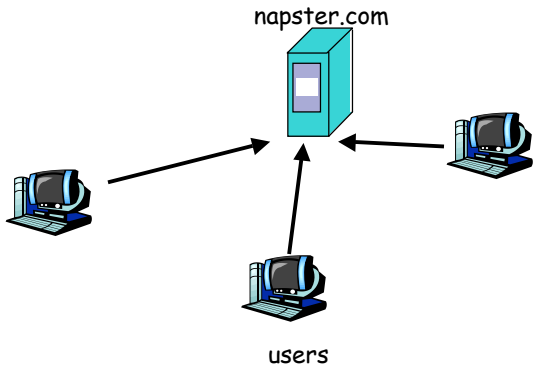


Napster Description

- Application level, client-server protocol over point-to-point TCP
- Four steps
  - Connect to Napster server
  - Upload your list of files (push) to server.
  - Give server keywords to search the full list with.
  - Select "best" of correct answers. (pings)

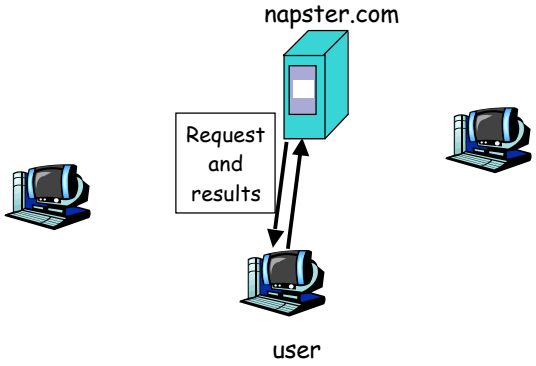
Napster Description

1. File list is uploaded



Napster Description (2)

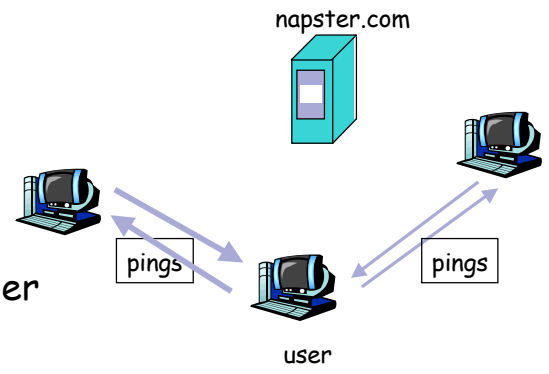
2. User requests search at server.



Napster Description (3)

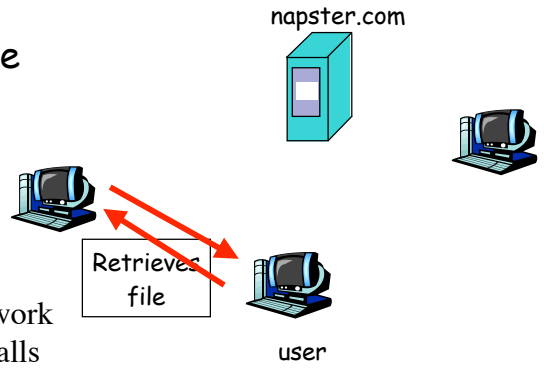
3. User pings hosts that apparently have data.

Looks for best transfer rate.



Napster Description (4)

4. User retrieves file



Retrieval has variations to work through firewalls

Napster Messages

General Packet Format

[length] [type] [data...]

Length:

Intel-endian 16-bit integer

size of [data...] in bytes

Type: (hex)

Intel-endian 16-bit integer.

00 - error
02 - login requested
03 - login accepted
05 - auto upgrade
64 - notification of shared file
66 - remove file
C8- search request

C9 - search response
203 - get request
204 - download ack
25B - whois query
25C - whois result
269 - list all channels
.....

Napster requesting a file

1. Search Request C1->S (after logging in to server)
FILENAME CONTAINS "Sneaker Pimps" MAX_RESULTS 75 FILENAME CONTAINS "tesko suicide" BITRATE "AT LEAST" "128" MAX_RESULTS 100 FILENAME CONTAINS "Ventolin" LINESPEED "EQUAL TO" 10

2. Search Response S->C1
"random band - random song.mp3" MD5 filesize 128 44100 159 lefty IP 4

3. Download request C1->S
203 <nick> <filename>

4. Download ack

5. Connection C1->IP:port (C2)

6. C2->C1 "1"

7. C1->C2 GET

8. C1->C2 <mynick> <filename> <offset>

9. C2->C1 <filesize><data>

Napster Evaluation

- Centralized Server
  - Single point of failure
  - can load balance among servers using DNS rotation
  - potential for congestion
  - Napster "in control" (freedom is an illusion)
- No security:
  - passwords in plain text
  - no authentication
  - no anonymity

Napster Evaluation (2)

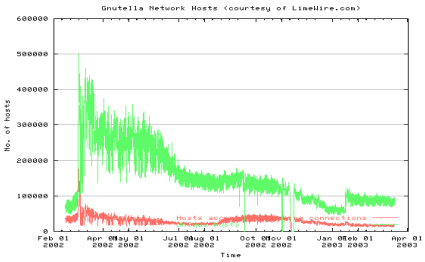
- Performance
  - Downloading from "best" peer?
  - Single peer download
- Availability
- Rich queries

Gnutella

- Peer-to-peer networking: applications connect to peer applications
- Focus: decentralized method of searching for files
  - Contrast with Napster's centralized index server(s)
- Each application instance serves to:
  - Store selected files
  - Route queries (file searches) from and to its neighboring peers
  - Respond to queries (serve file) if file stored locally
  - Server + client = servent

Gnutella History

- 3/14/00: release by AOL, almost immediately withdrawn
- Too late: 88K users on Limewire at 3/16/03
- Many iterations to fix poor initial design (poor design turned many people off)
  - Clients: Limewire, Bearshare, OpenCola, etc.



Gnutella Message Format

- Message ID: 16 bytes
- FunctionID: 1 byte indicating
  - 00 ping: used to probe gnutella network for hosts
  - 01 pong: used to reply to ping, return # files shared
  - 80 query: search string, and desired minimum bandwidth
  - 81: query hit: indicating matches to 80:query, my IP address/port, available bandwidth
- RemainingTTL: decremented at each peer to prevent TTL-scoped flooding
- HopsTaken: number of peer visited so far by this message
- DataLength: length of data field

How it works

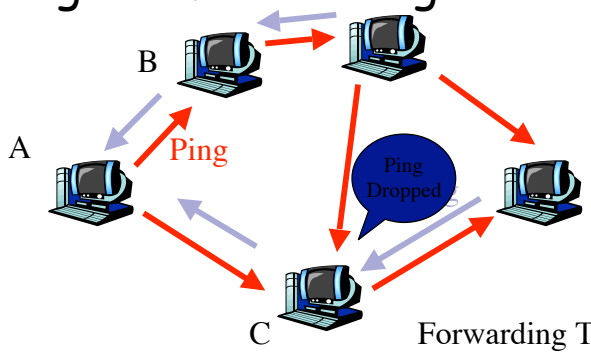
- First find a network node to connect to
  - How do you find where to connect? Limewire Gateway (pong server)
- Ping
  - Find more servents to connect
  - Discover the *horizon size* (amount of data shared)
  - Other servents reply with pong messages
    - IP Address of servent, number of files shared, number of kilobytes shared
    - Why report the files, kilobytes?

How it works (2)

Once connected, searching by flooding:

- If you don't have the file you want, query your neighbors.
- If they don't have it, they contact their neighbors, for a maximum hop count of 10.
- Requests are flooded, but there is no tree structure.
- No looping but packets may be received twice.
- Reverse path forwarding

Flooding and Forwarding



Forwarding Table

Message ID	Neighbor	Message Type
3	A	Ping

Queries and Data Transfer

- Client sends a *Query* message
  - Contains minimum speed, search criteria (string)
  - Query is flooded until TTL expires
- Node that matches criteria returns a *QueryHit* message
  - Contains port, IP Address, Speed
  - Result set (file index, file size, file name)
  - Servent Identifier

Queries and Data Transfer (2)

- QueryHit response reaches the requestor
- Requestor contacts the servent *directly*
  - At the IP address+port supplied
  - Using HTTP request
    - GET /get/<file index>/<file name>/ HTTP/1.0\r\n
      Connection: Keep-Alive\r\n
      Range: bytes=0-r\r\n
      User-Agent: Gnutella\r\n
      \r\n
  - Server sends back the data
    - HTTP 200 OK\r\n
      Server: Gnutella\r\n
      Content-type: application/binary\r\n
      Content-length: 4242342\r\n
      \r\n
  - Why HTTP?

Firewall Traversal

- If server is behind firewall, does not accept incoming connections
- Requestor sends *Push* message with Servent ID contained in *QueryHit*
  - Contains: Servernt ID, File Index, IP Addr, Port
  - Server opens a TCP connection to IPAddr+port
    - GIV HTTP request (non standard one)
    - Requester replies with GET HTTP request

Gnutella Evaluation

- What we care about:
  - How much traffic does one query generate?
  - How many hosts can the network support at once?
  - What is the latency associated with querying?
  - Is there a bottleneck?

Gnutella: Initial Problems and Fixes

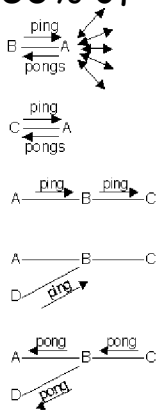
- Freeloading: WWW sites offering search/retrieval from Gnutella network without providing file sharing or query routing.
  - Block file-serving to browser-based non-file-sharing users (How?)
- Prematurely terminated downloads:
  - Software bugs
  - long download times over modems
  - modem users run gnutella peer only briefly (Napster problem also!) or any users becomes overloaded
  - fix: peer can reply "I have it, but I am busy. Try again later"

Gnutella: Initial Problems and Fixes (2)

- 2000: avg size of reachable network ony 400-800 hosts. Why so smalll?
  - modem users: not enough bandwidth to provide search routing capabilities: routing black holes
- Fix: create peer hierarchy based on capabilities
  - previously: all peers identical, most modem blackholes
  - connection preferencing:
    - favors routing to well-connected peers
    - favors reply to clients that themselves serve large number of files: prevent freeloading
  - Limewire gateway functions as Napster-like central server on behalf of other peers (for searching purposes)

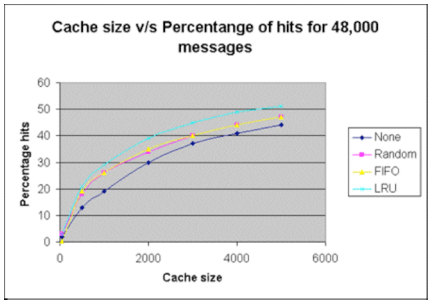
Gnutella Enhancements

- Pings/Pongs can consume up to 50% of bandwidth
- Solutions:
  - Pong Limiting
  - Pong Caching
  - Ping Multiplexing
    http://www.limewire.com/index.jsp/pingpong



Gnutella Enhancements (2)

- Cache query responses
- Results
- Evolving Protocol
  - Gnutella Developer Forum
    - UltraPeers
    - Alternative query routing algs



Can Heterogeneity Make Gnutella Scale?

- Ideas
  - Replace query flooding with multiple random walks
  - Proactive replication (#replicas proportional to sqrt(request rate))
  - Result: Two orders of magnitude improvement in terms of query-time, per node load and message traffic

Can Heterogeneity Make Gnutella Scale? (2)

- Gnutella assumption:
  - All peers are equal
  - Not true! Heterogeneity among P2P peers (dial-up users vs. college users)
  - Evolve topology to match node capacities
  - Use random walks over this topology

Can Heterogeneity make Gnutella scale? (3)

- Solution outline
  - C_i, node capacity in[j,i] messages from j->i, out[i,j] messages i->j
  - Init in[i,j]=out[i,j]=0, OutMax[i,j]=c_i/d_I
  - Update according the messages received/sent
  - Check if overloaded
    - If so redirect high-input neighbor to neighbor with high OutMax (spare capacity)
    - Intuitively, take yourself *out of the loop*
    - If node cannot be found ask neighbor to *throttle back*
- Result: Average query length reduces from 70 to 2-9 hops (depending on topology)

Gnutella Evaluation (2)

- Architectural lessons learned?
  - ..
  - ..
  - ..
  - ..
- Anonymity and security?
- Other?
- Good source for technical info/open questions:
  - http://www.limewire.com/index.jsp/tech_papers

FreeNet History

- Final Year project Ian Clarke , Edinburgh University, Scotland, June, 1999
- Sourceforge Project
- V.0.1 (released March 2000)
- Latest version(March, 2003): 0.5.1

Design goals and non-goals

- Goals
  - Anonymity for producers and consumers
  - Deniability for information storers
  - Resistance to attempts by third parties to deny access to information
  - Efficient dynamic storage and routing of information
  - Decentralization of all network functions
- Non-Goals
  - Permanent storage guarantee(!)
  - General network usage anonymity

Freenet: How it works

- Data structure
- Key Management
- Problems
  - How can one node know about others
  - How can it get data from remote nodes
  - How to add new nodes to Freenet
  - How does Freenet manage its data

Data structure

- Routing Table
  - Pair: node address: ip, tcp; corresponding key value
- Data Store
  - Requirement:
    - rapidly find the document given a certain key
    - rapidly find the closest key to a given key
    - keep track the popularity of documents and know which document to delete when under pressure

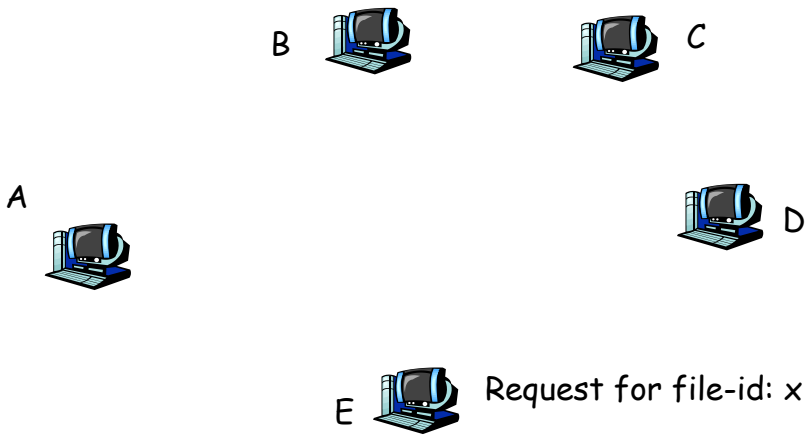
Key Management(1)

- A way to locate a document anywhere
- Keys are used to form a URI
- Two similar keys don't mean the subjects of the file are similar!
- Keyword-signed Key*(KSK)*
  - *Based on a short descriptive string, usually a set of keywords that can describe the document*
  - *Example: University/hopkins/cs/andreas/lecture1.ppt*
  - *Uniquely identify a document*
  - *Potential problem – global namespace*

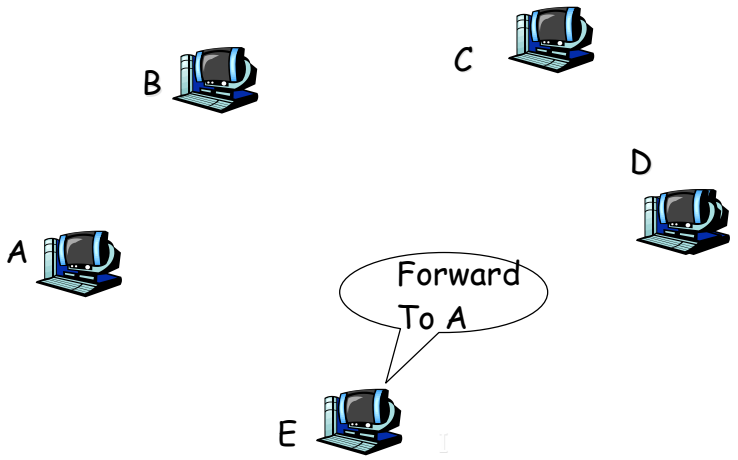
Key Management (2)

- Signed-subspace Key*(SSK)*
  - *Add sender information to avoid namespace conflict*
  - *Private key to sign/ public key to verify*
- Content-hash Key*(CHK)*
  - *Message digest algorithm, Basically a hash of the document*

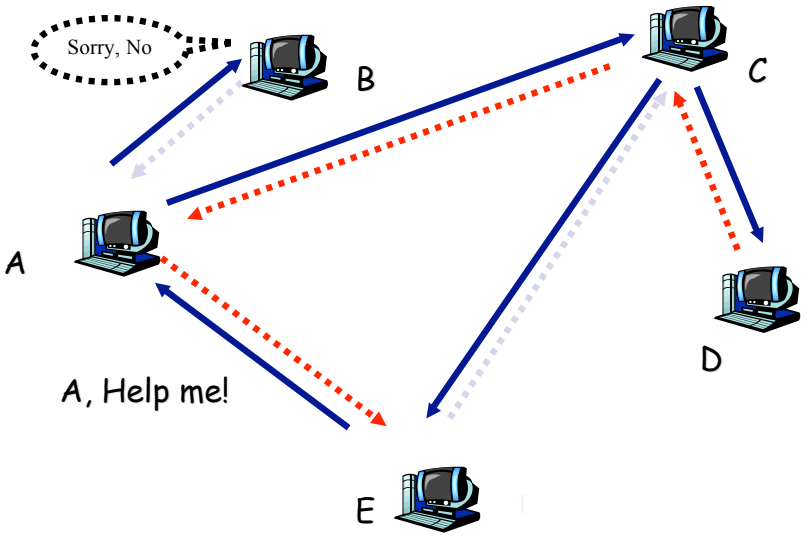
Routing Algorithm



Routing Algorithm (2)



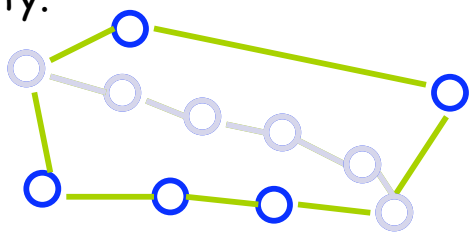
Routing Algorithm (3)



Strength of routing algorithm(1)

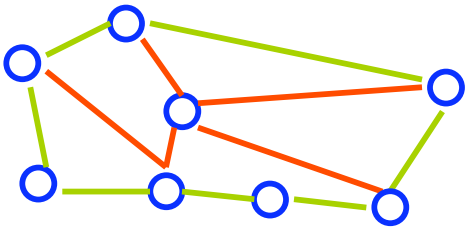
- Replication of Data Clustering (1)

  (Note: Not subject-clustering but key-clustering!)
- Reasonable Redundancy: improve data availability.



Strength of routing algorithm(2)

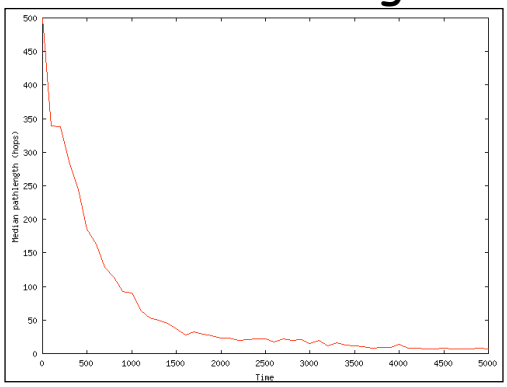
- New Entry in the Routing Table: the graph will be more and more connected. --- Node discovery



Security and authentication issues

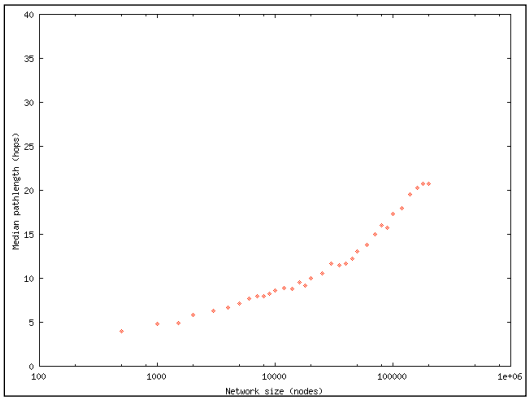
- How to ensure anonymity:
  - Nodes can lie randomly about the requests and claim to be the origin or the destination of a request
  - Hop-To-Live values are fuzzy
  - Then it's impossible to trace back a document to its original node
  - Similarly, it's impossible to discover which node inserted a given document.

Network convergence



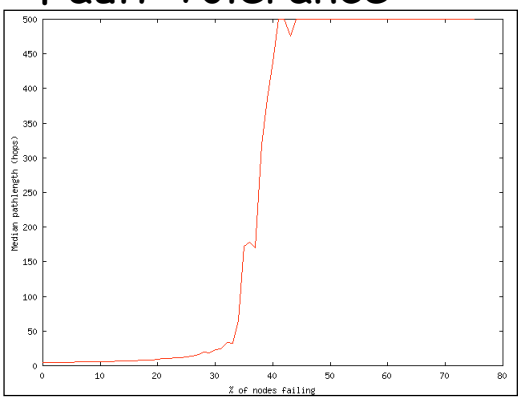
- X-axis: time
- Y-axis: # of pathlength
- 1000 Nodes, 50 items datastore, 250 entries routing table
- the routing tables were initialized to ring-lattice topology
- Pathlength: the number of hops actually taken before finding the data.

Scalability



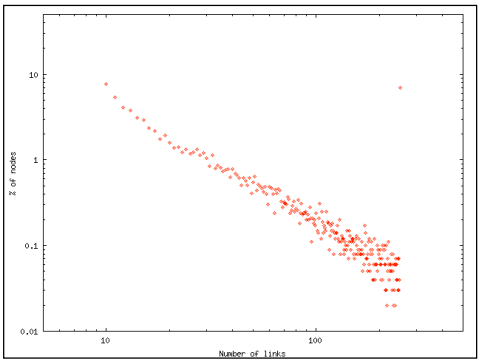
- X-axis: # of nodes
- Y-axis: # of pathlength
- The relation between network size and average pathlenth.
- Initially, 20 nodes. Add nodes regularly.

Fault Tolerance



- X-axis: # of nodes failing
- Y-axis: # of pathlength
- The median pathlength remains below 20 even when up to 30% nodes fails.

Small world Model



- X-axis: # of links
- Y-axis: # % of nodes
- Most of nodes have only few connections while a small number of news have large set of connections.
- The distribution follows a power law.

Freenet Evaluation

- Keep anonymity
- Distributed model; data available
- Converge fast
- Adaptive

Protocol Details

- Header information
  - DataReply
    UniqueID=C24300FB7BEA06E3
    Depth=a
    * HopsToLive=2c
    Source=tcp/127.0.0.1:2386
    DataLength=131
    Data 'Twas brillig, and the slithy toves Did gyre
    and gimble in the wabe: All mimsy were
    the borogoves And the mome raths outgrabe