



Vivek Rai

John Lopez

25 March, 2004

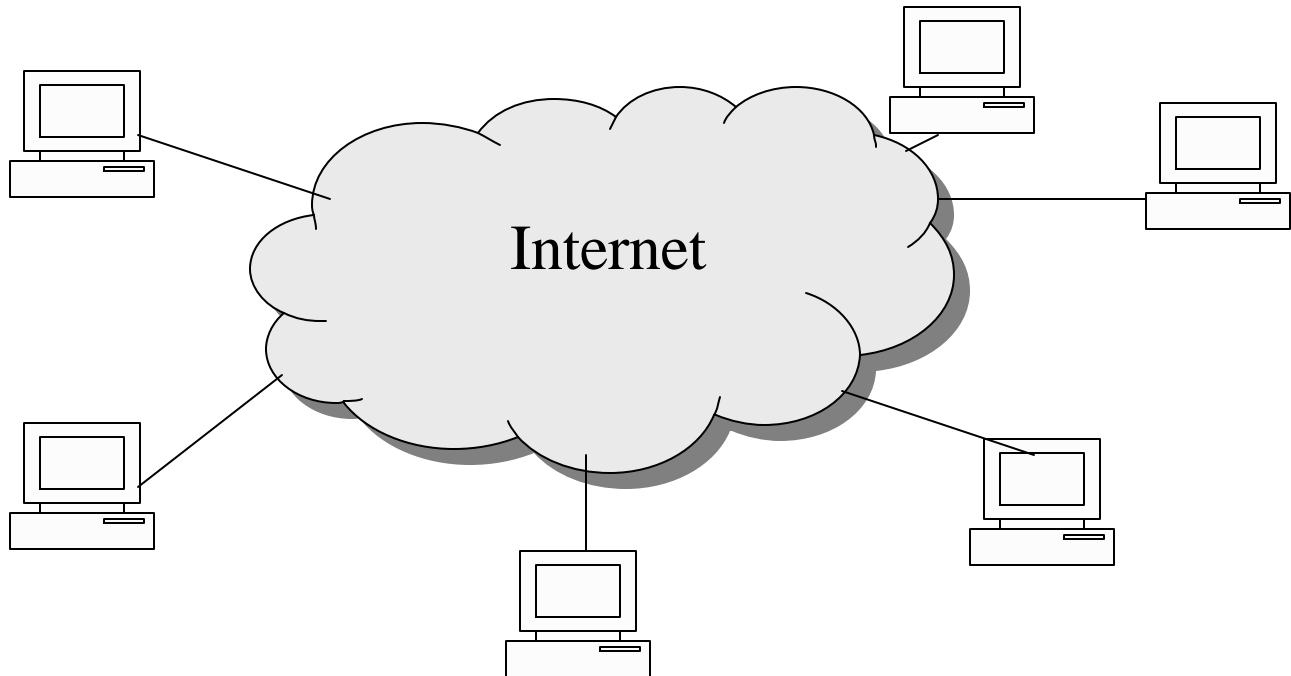


Overview

- ◆ Background: Peer-to-Peer Networks (P2P)
- ◆ *Chord* Protocol
- ◆ *Chord* vulnerabilities
- ◆ Security Concerns: Structured Overlays
- ◆ Security Analysis
- ◆ Conclusions

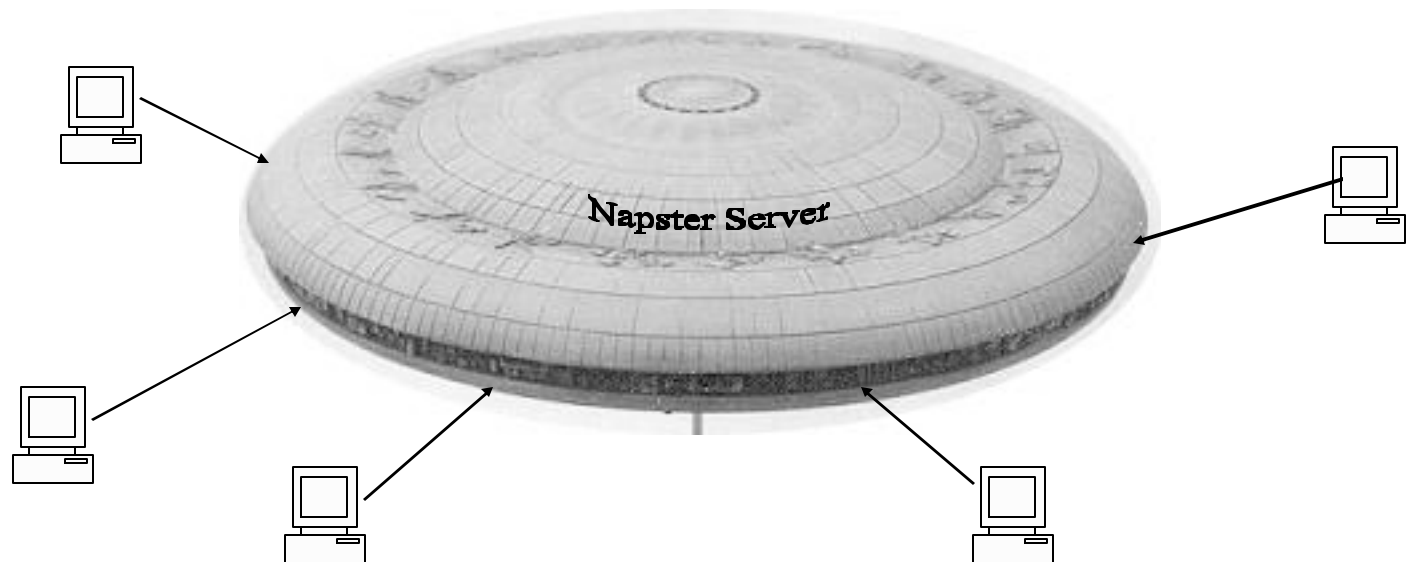
What is P2P technology?

- ◆ A distributed system architecture:
 - No centralized control
 - No hierarchical organization
 - Each node is equivalent in functionality (“Peers”)
- ◆ Nodes share bandwidth, storage, computation



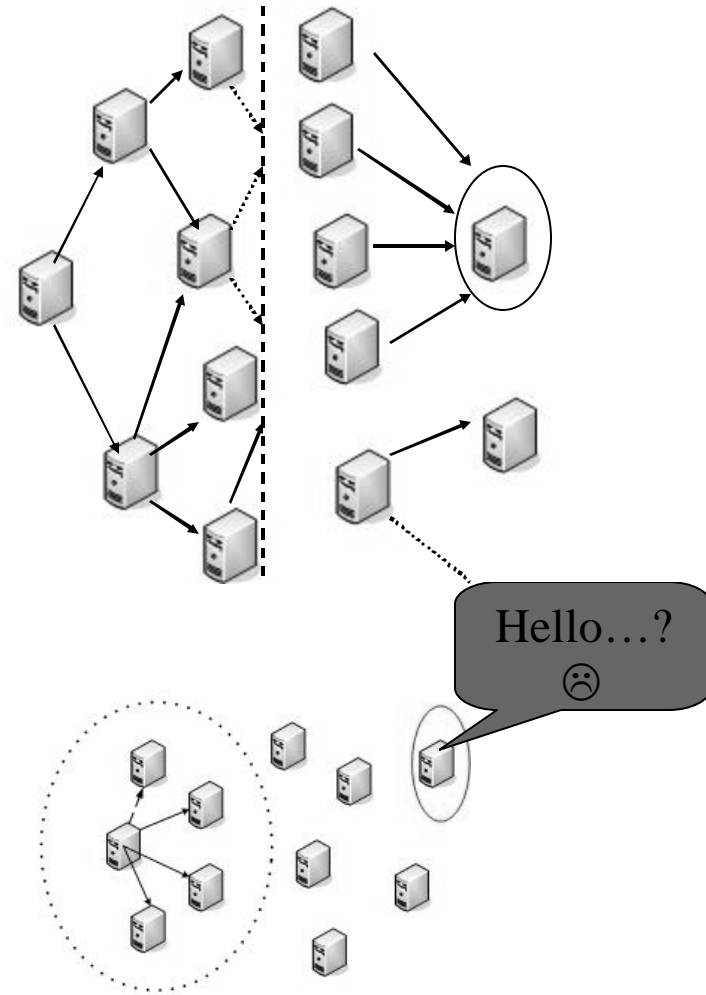
Centralized, Unstructured (Napster)

- ◆ Object indices stored at a central server
(the Mothership)
- ◆ Queries sent to central server →
- ◆ ← Object index (location) returned to node
- ◆ Disadvantage: Single point of failure (not practical)!



Distributed, Unstructured Overlays (KaZaa, Gnutella, Free net)

- ◆ Organize nodes in a random graph
- ◆ “Flood” graph with messages in attempt to query content stored by nodes
- ◆ Massive overhead with respect to messages transmitted; not scalable
- ◆ Possible Solution: Constrained flooding
- ◆ *New Problem*: unreachable content!





Structured P2P Overlays

- ◆ Conform to specific topology
- ◆ Provide reliable, scalable searches
- ◆ Employ Distributed Hash Tables
- ◆ Dynamically map objects to *live* nodes



Distributed Hash Tables (DHTs)

- ◆ DHTs comprised of a lookup protocol
- ◆ Basic components of the look-up protocol:
 - Key identifier space
 - Node identifier space
 - Rules for linking keys to nodes
 - Routing tables, per node, referring to other nodes
 - Rules for updating routing tables as nodes join/fail
- ◆ Nodes are hashed based on IP address



The *Chord*^[1] protocol

- ◆ Structured overlay which specifies:
 - how to find locations of keys (objects)
 - how new nodes join the system
 - how to recover from the failure (or departure) of existing nodes
- ◆ *Chord* system model addresses:
 - Load balance: distributed hash function spreads keys evenly over the nodes
 - Decentralization: *Chord* is fully distributed
 - Availability: adjusts its internal tables automatically; node responsible for key can always be found



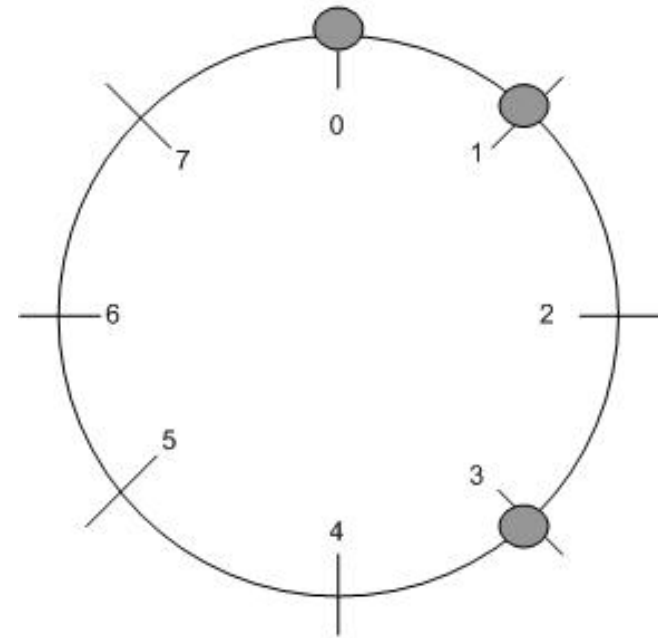
Chord^[1] (2)

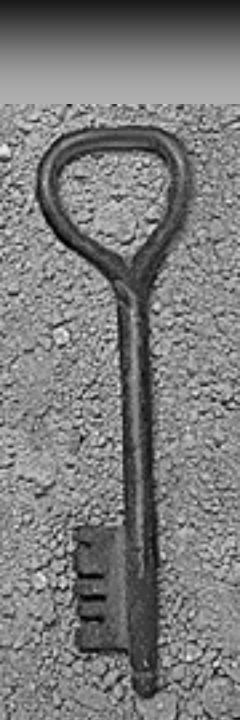
- ◆ Employs consistent hashing whose properties include:
 - Load balancing with high probability
(all nodes receive \sim same number of keys)
 - When Nth node joins/leaves network, only fraction of keys ($O(1/N)$) are moved to a different location
- ◆ Does not require that every node know about every other node
- ◆ Each node maintains information about only $O(\log N)$ other nodes; lookup requires $O(\log N)$ messages



Chord^[1] (3): Hashing

- ◆ Hash function assigns each node and key an m -bit “identifier”
- ◆ Identifiers are ordered in “identifier circle” modulo 2^m
- ◆ Key, k , is assigned to “matching” node identifier
- ◆ Responsibility of successor





Chord^[1] (3): Routing

- ◆ Node required to only be aware of successor
- ◆ Each node maintains a routing table of, at most, m entries (*finger table*)
- ◆ The i^{th} entry of n 's table contains identity of the first node that succeeds n by at least 2^{i-1} on identifier circle

Note: Table maintains successors

Node ID	Offset	Successor
n_a	0	n_a
n_a	1	$\text{succ}(n_a+1)$
n_a	2	$\text{succ}(n_a+2)$
\cdot \cdot \cdot	\cdot \cdot \cdot	\cdot \cdot \cdot
n_a	2^{i-1}	$\text{succ}(n_a + 2^{i-1})$

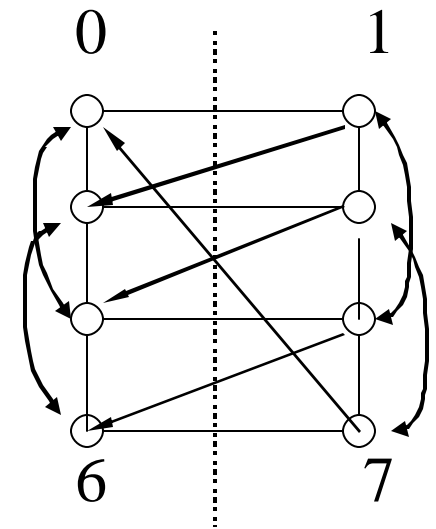


Chord^[1] (4): Node Joins/Failures

- ◆ Nodes may join/leave arbitrarily
- ◆ Finger tables of existing nodes are updated
- ◆ Keys are transferred to responsible node

Adversarial Link Failure

- ◆ Adversary attempts to corrupt routing table entries (exploits trust relationship)
- ◆ Corrupted table entries can segment the identifier circle (two disparate graphs)
- ◆ Minimum number of links required to segment graph: Bisection width
- ◆ Bisection width of *Chord* is N
- ◆ Loguinov *et al.* [5] study bisection width of popular DHTs






Adversarial Node Failure

- ◆ An adversary can acquire $O(\log N)$ node-ids to “expel” a target node from the network (Sybil attack)
- ◆ If multiple node-ids can be easily acquired, then this attack can be carried out on a large scale
- ◆ To combat such an attack, node-id certification must be employed[3]



Chord in the real world

- ◆ Typically, the size of the network is much smaller than the size of the node-identifier space
- ◆ Therefore, each node is responsible for a zone of the network
- ◆ Zone sizes, in practice, are random
- ◆ As a result of random node-join, maximum zone size is $O(\log N)$




DOS attack-rapid node join/leave[2]

- ◆ Attack: Rapid node joins and leaves .
- ◆ Analysis:
 - At the time of node-join, it requires $O(\log_2 N)$ messages to find a new neighbor within the Chord.
 - Thus, the total number of messages to construct a neighbor table in Chord is $(O(\log_2 N))^2$.
 - New neighbor table requires establishing $O(\log_2 N)$ new network-layer connections.
 - Moreover, a node-join requires transfer of keys, which is also a message overhead.



DOS attack-frequent node join/leave 2

- ◆ Example: $N = 2^{30}$, frequency of node join = 100/second, fraction of faulty nodes = 10%, size of a routing-message = 1KB.
 - Traffic due to one node-join = 50 MB/sec.
 - Total traffic generated due to frequent node join/leave of all faulty nodes = 10^{10} MB/sec.
- ◆ Next, we discuss how to stage a DOS attack over a single link.



DOS attack – Single Node

- ◆ Attack: A set of faulty nodes forward all their routing-traffic to a single node.
- ◆ Analysis:
 - Amount of routing-queries forwarded to a node is proportional to the number of incoming links.
 - Under uniform zone size-distribution, average number of incoming links is equal to the average number of outgoing links.
 - f number of faulty nodes, can create a total amount of $O(\log_2 N) * f$ traffic at a single link.
- ◆ Example: $N = 2^{30}$, Fraction of faulty nodes = 1%, message/sec = 1KB/sec.
 - Traffic at the attacked node = $15 * 10^3$ MB/sec.



Secure Message Forwarding[4]: Attack

- ◆ Every routing-query traverses fixed number of hops.
 - Average number of routing hops depend on the routing geometry.
 - It is $\log_2(N)/2$ in Chord, $(d/4)(N^{1/d})$ in CAN, and $\log_2^b(N)$ in Pastry and Tapestry.
- ◆ What if one of the routing hop is malicious?
 - Query gets dropped.
 - Queries gets forwarded to a malicious node.



Secure Message

Forwarding: Analysis[4]

- ◆ Let f be the fraction of malicious nodes in the network.
- ◆ Fraction of successful queries is approximately equal to $(1-f)^{E[h]}$, where $E[h]$ is the expected routing length.
- ◆ Thus, in a Chord network of size 2^{30} , with fraction of malicious node equal to 5%, fraction of successful queries is only 46%.
- ◆ More detailed analysis is in Appendix[1].



Secure Routing Primitive [4]

- ◆ Objective: Takes a message and a destination key and ensures that with very high probability at least one copy of the message reaches a correct root node.
- ◆ Approach: Applies a *failure test* to determine if routing worked. If the failure test returns positive, a more expensive routing is used.



Routing Failure Test

- ◆ Objective: How to check the authenticity of a node in the P2P network?
- ◆ Basic Idea:
 - Every node can sample its neighbor set to estimate the node-density in the network.
 - If a node's routing table conforms to the linkage rules of the network, it can estimate the node-density with minimum error.
 - If not, there will be a significant estimation error.



Routing Failure Test 2

- ◆ Details:
 - Every correct node estimates the node density in the P2P network, using it's neighbor table.
 - Node-density estimation procedure requires careful study of the numerical differences between the neighbor identifiers.
 - Pastry has a symmetric neighbor-id distribution, it estimates node-density using the average routing-distance in it's neighbor table.
 - Since linkage rules for different DHTs differ significantly, the estimation algorithm is also different.
 - For example, Chord implies only one way routing. Thus, Pastry's estimation algorithm may not work for Chord, due to the asymmetric neighbor-id distribution in Chord.



Routing Failure Test 3: Pastry[4].

◆ Test:

- All the node-ids in the root node's neighbor set have valid node certificates.
- The average numerical distance μ_{rn} between consecutive nodeIds in rn satisfies $\mu_{rn} < \mu_p^{*?}$, where $?$ is a threshold constant.

◆ Result:

- The probability of false positive:
 - $\alpha = \exp\{-k[(r+1)\log((r+?)/(r+1))-\log ?]\}$.
- The probability of false negative:
 - $\beta = \exp\{-k[(r+1)\log((r+?*f)/(r+1))-\log (*f)]\}$, where f is the fraction of faulty nodes.
- $r = n/k$, where n is the size of the sender node's neighbor set and k is the size of the root node's neighbor set or the replica root set.



Routing Failure Test 4

- ◆ Usage:
 - Authenticity of the root-node is first validated before storing a key on that node.
 - If the root-node is not valid, the key is stored at a carefully chosen neighbor set of the root-node. Why?
 - These carefully chosen nodes from the neighbor set are known as *replica roots*.
 - When some another node searches for a key it first validates the authenticity of the corresponding root-node.
 - If invalid root-node, the searching node resorts to redundant routing.



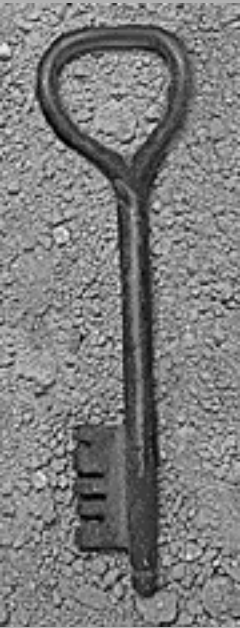
Redundant Routing

- ◆ Basic Idea:
 - Route copies of the message over multiple routes toward each of the destination key's replica roots.
 - Insure that the routes are diverse.
 - The technique is sufficient in overlays that distribute the replica nodes uniformly over the id space(e.g. CAN and Tapestry).
 - Not sufficient for the overlays like Chord and Pastry, where the paths to the replica nodes converge to the root node (path overlap).
- ◆ A technique called *neighbor set anycast* is developed in [4], which reduces the path overlap by purposely sending the queries over the less favored paths.
- ◆ Read more on path-overlap in P2P networks from [5].



Iterative Routing[2]

- ◆ Sender maintains history of the previously queried node.
 - Suppose node n is queried at the t^{th} step.
 - The node n forwards the query to the node n' at $(t+1)^{\text{th}}$ step.
 - If n' fails to provide a good route, the query backtracks and starts from n again, on an alternative path.
 - Routing is typically augmented with the *hop tests* to check whether the route provided by a node is good.
- ◆ Simulation results in [3] show that, iterative routing improves the probability of routing successfully as compared to the redundant routing but simultaneously increases the hop count.



Conclusion

- ◆ DHTs are definitely more scalable than the unstructured or centralized P2Ps but are they secure?
- ◆ DHT research is still in its infancy, we need better models to study their security aspect.



References

1. I. Stoica, R. Morris, D.L. Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, “*Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications*,”. *IEEE/ACM Transactions on Networking*.
2. E.Sit, and R.Morris, “Security Considerations for Peer-to-Peer Distributed Hash Tables,”. IPTPS’02.
3. J.R.Douceur, “The Sybil Attack,” IPTPS’02.
4. M.Castro, P.Druschel, A.Ganesh, and D.S.Wallach, “Secure routing for structured peer-to-peer overlay networks,” *ACM OSDI’02*.
5. D.Loguinov, A.Kumar, V.Rai, and S.Ganesh, “Graph-Theoretic Analysis of Structured Peer-to-Peer Systems: Routing Distances and Fault Resilience,” *ACM SIGCOMM’03*.
6. http://www.umiacs.umd.edu/docs/Presentation_Nov10.pdf



Appendix[1]: Extended Analysis (Chord).

- ◆ Let, f be the fraction of malicious nodes.
- ◆ Probability of a successful query over the path of random length h , is equal to $(1-f)^h$.
- ◆ Remember, h lies between 0 to $\log_2(N)$.



Appendix[1]: Extended Analysis (Chord) 2

- ◆ Let $g(h)$ be the PDF of the routing length.
- ◆ Thus, Probability of a Successful query is equal to $\sum_{h=0}^{\log_2 N} g(h)(1-f)^h$.
- ◆ h is binomially distributed , see Loguinov *et al.*[].
- ◆ Probability of successful query in chord is equal to $(1-f/2)^{\log_2 N}$.