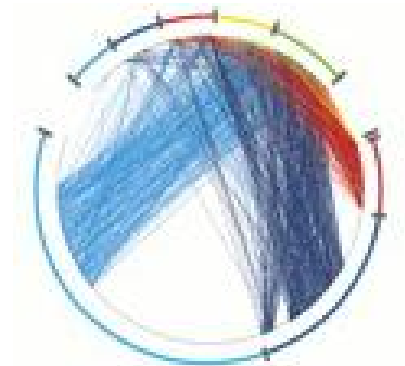
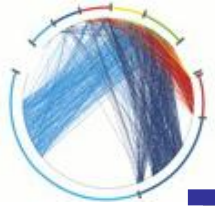


Models and Algorithms for Complex Networks

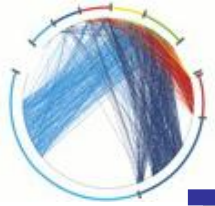
Searching in P2P Networks





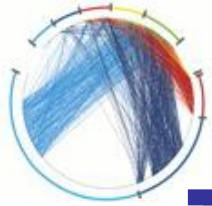
What is P2P?

§ “the **sharing** of computer resources and services by **direct** exchange of information”



What is P2P?

§ “P2P is a class of applications that **take advantage of resources** – storage, cycles, content, human presence – available at the edges of the Internet. Because accessing these **decentralized** resources means operating in an environment of unstable and unpredictable IP addresses P2P nodes must operate outside the DNS system and have significant, or **total autonomy from central servers**”



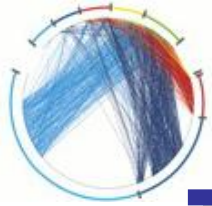
What is P2P?

§ “A **distributed** network architecture may be called a P2P network if the participants share a part of their own resources. These **shared resources** are necessary to provide the service offered by the network. The participants of such a network are both resource providers and resource consumers”



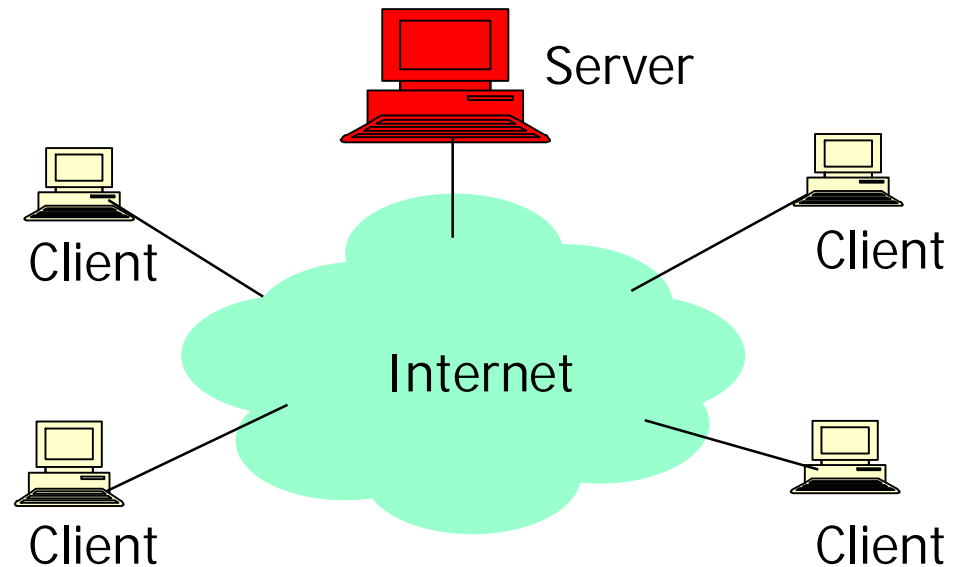
What is P2P?

- § Various definitions seem to agree on
 - § sharing of resources
 - § direct communication between equals (peers)
 - § no centralized control



Client/Server Architecture

- § Well known, powerful, reliable server is a data source
- § Clients request data from server
- § Very successful model
 - § WWW (HTTP), FTP, Web services, etc.



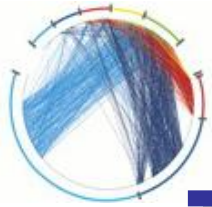
* Figure from <http://project-iris.net/talks/dht-toronto-03.ppt>



Client/Server Limitations

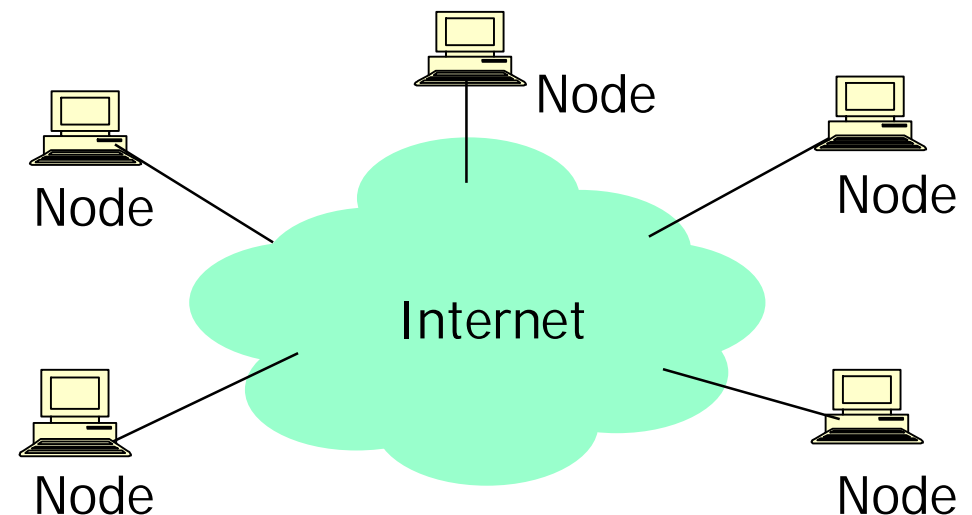
- § Scalability is hard to achieve
- § Presents a single point of failure
- § Requires administration
- § Unused resources at the network edge

- § P2P systems try to address these limitations

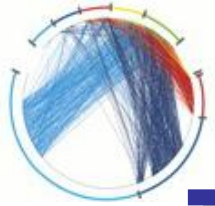


P2P Architecture

- § Clients are also servers and routers
 - § Nodes contribute content, storage, memory, CPU
- § Nodes are autonomous (no administrative authority)
- § Network is dynamic: nodes enter and leave the network "frequently"
- § Nodes collaborate directly with each other (not through well-known servers)
- § Nodes have widely varying capabilities

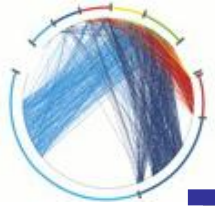


* Content from <http://project-iris.net/talks/dht-toronto-03.ppt>



P2P Goals and Benefits

- § Efficient use of resources
 - § Unused bandwidth, storage, processing power at the “edge of the network”
- § Scalability
 - § No central information, communication and computation bottleneck
 - § Aggregate resources grow naturally with utilization
- § Reliability
 - § Replicas
 - § Geographic distribution
 - § No single point of failure
- § Ease of administration
 - § Nodes self-organize
 - § Built-in fault tolerance, replication, and load balancing
 - § Increased autonomy
- § Anonymity – Privacy
 - § not easy in a centralized system
- § Dynamism
 - § highly dynamic environment
 - § ad-hoc communication and collaboration



P2P Applications

§ Are these P2P systems?

§ File sharing (Napster, Gnutella, Kazaa)

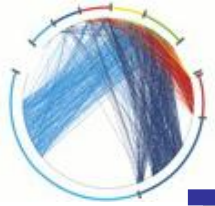
§ Multiplayer games (Unreal Tournament, DOOM)

§ Collaborative applications (ICQ, shared whiteboard)

§ Distributed computation (Seti@home)

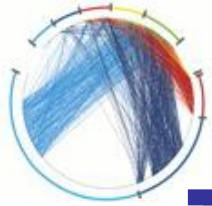
§ Ad-hoc networks

§ We will focus on **information sharing P2P systems**



Information sharing P2P systems

- § The resource to be shared is information (e.g. files)
- § The participants create an **overlay network** over a physical network (e.g. the Internet)
- § **P2P search problem**: locate the requested information in the overlay network efficiently
 - § small number of messages and hops
 - § low latency
 - § load balance
 - § easy to update in a highly dynamic setting



Popular file sharing P2P Systems

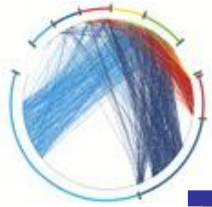
§ Napster, Gnutella, Kazaa, Freenet

§ Large scale sharing of files.

§ User A makes files (music, video, etc.) on their computer available to others

§ User B connects to the network, searches for files and downloads files *directly* from user A

§ Issues of copyright infringement



Napster

§ program for sharing files over the Internet

§ a “disruptive” application/technology?

§ history:

§ 5/99: Shawn Fanning (freshman, Northeastern U.) founds Napster Online music service

§ 12/99: first lawsuit

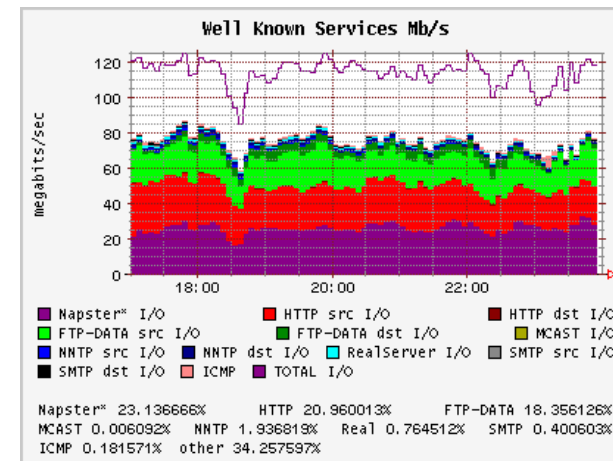
§ 3/00: 25% UWisc traffic Napster

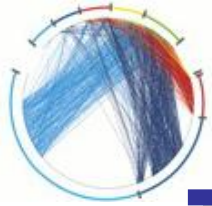
§ 2000: est. 60M users

§ 2/01: US Circuit Court of Appeals: Napster knew users violating copyright laws

§ 7/01: # simultaneous online users:

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



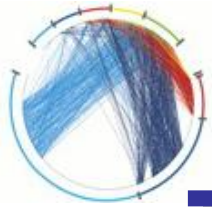


Napster: how does it work

Application-level, client-server protocol over point-to-point TCP

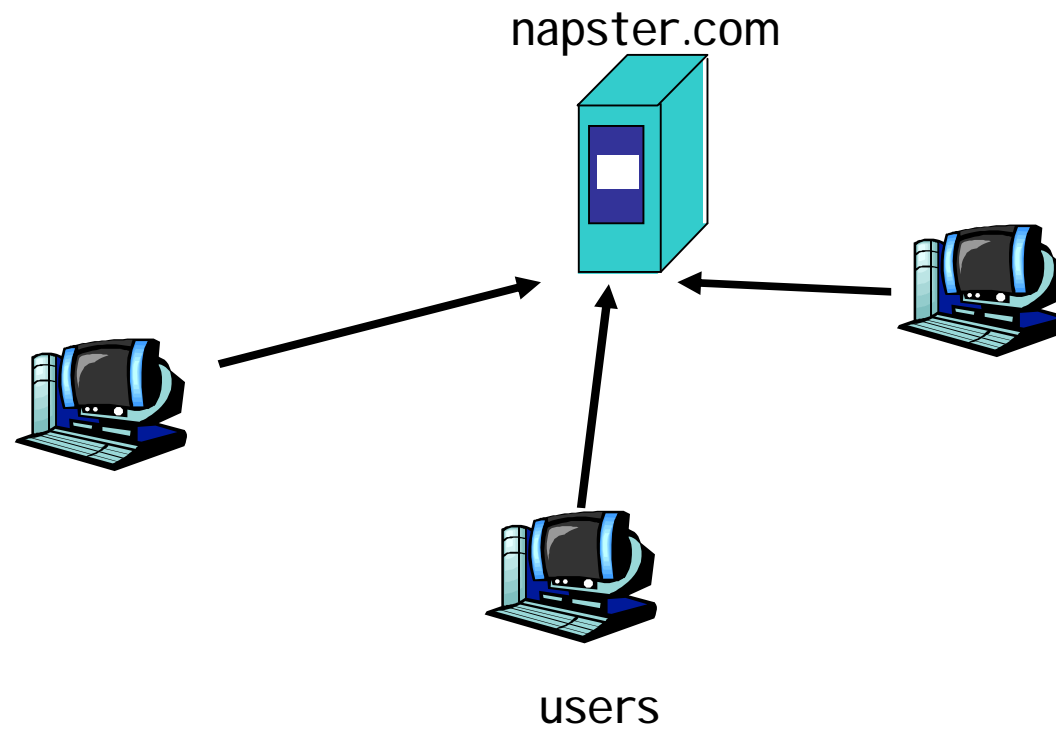
Four steps:

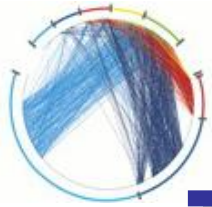
- § Connect to Napster server
- § Upload your list of files (push) to server.
- § Give server keywords to search the full list with.
- § Select “best” of correct answers. (pings)



Napster

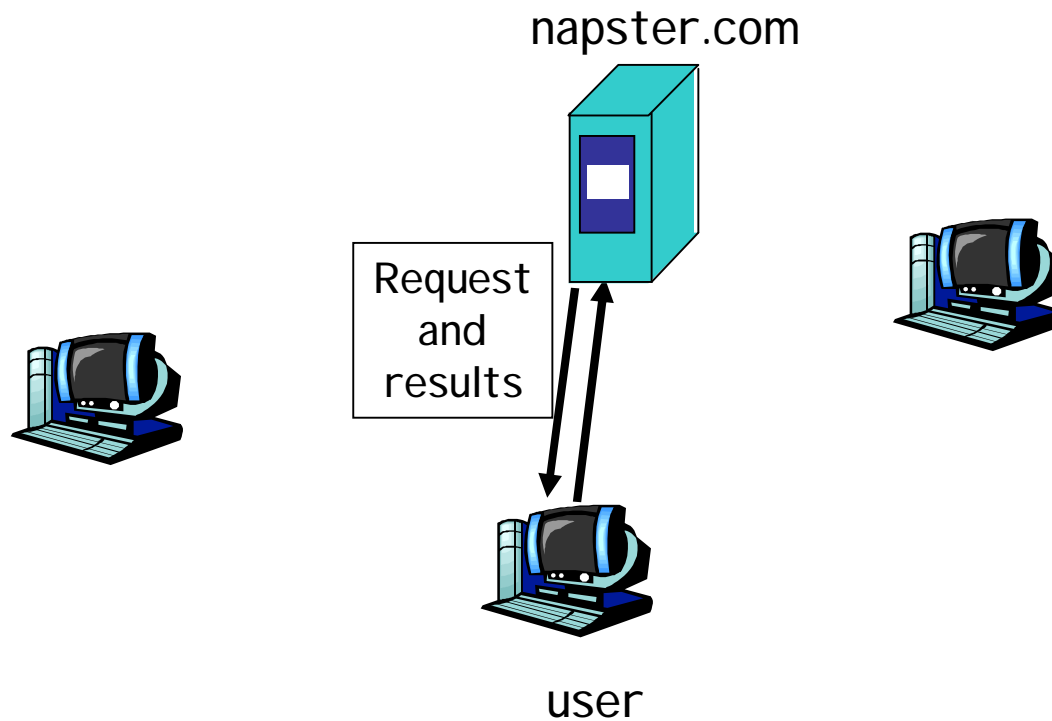
1. File list is uploaded

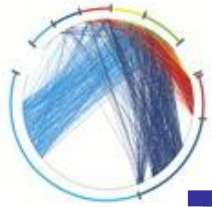




Napster

2. User requests search at server.

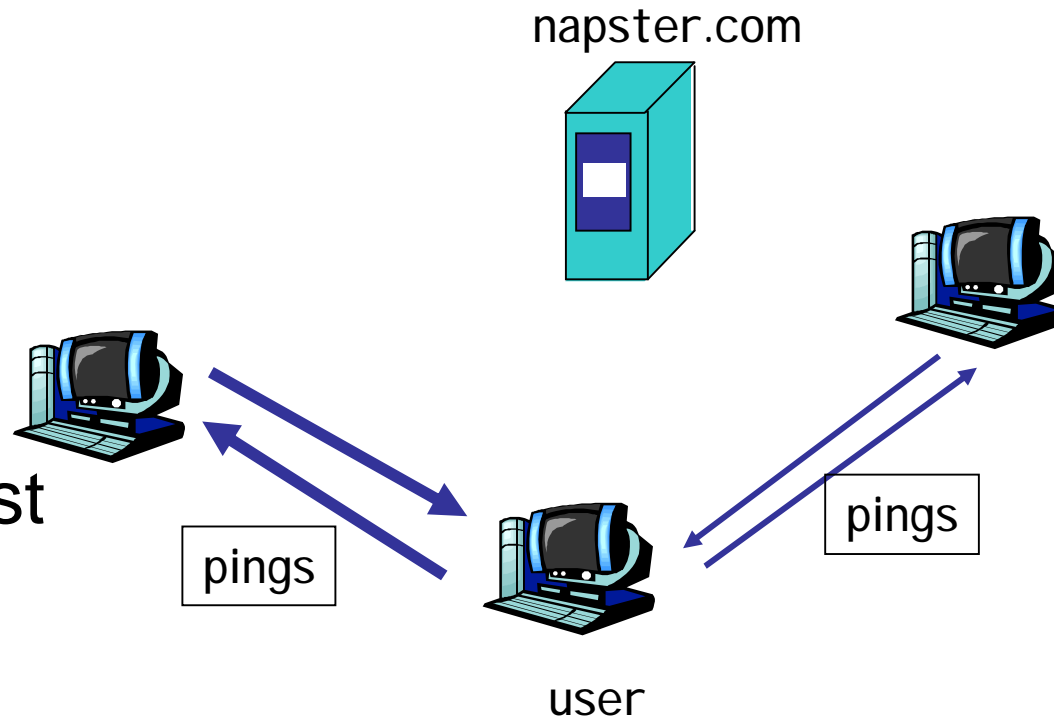


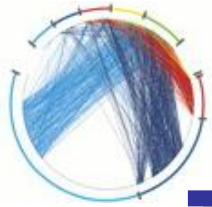


Napster

3. User pings hosts that apparently have data.

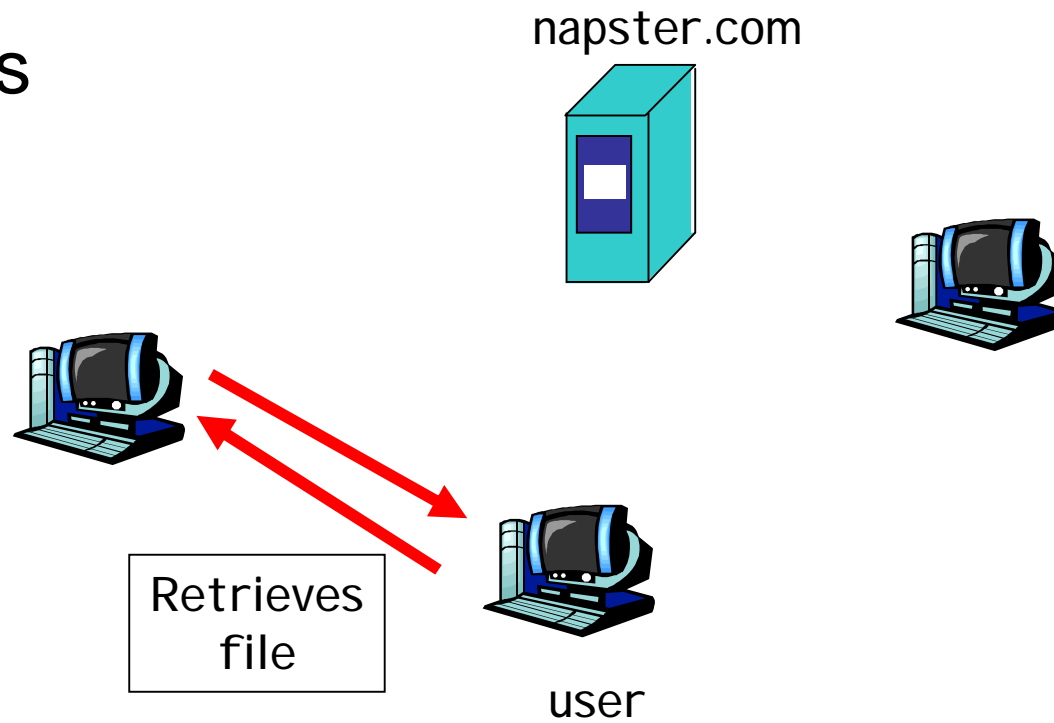
Looks for best transfer rate.

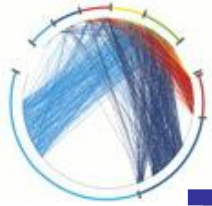




Napster

4. User retrieves file





Napster

§ Central Napster server

- ↳ Can ensure correct results
- ↳ Fast search

↳ Bottleneck for scalability

↳ Single point of failure

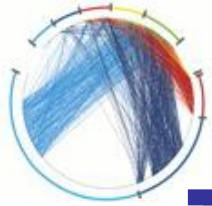
↳ Susceptible to denial of service

- Malicious users
- Lawsuits, legislation

§ Hybrid P2P system – “all peers are equal but some are more equal than others”

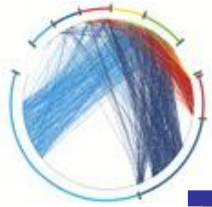
§ Search is centralized

§ File transfer is direct (peer-to-peer)



Gnutella

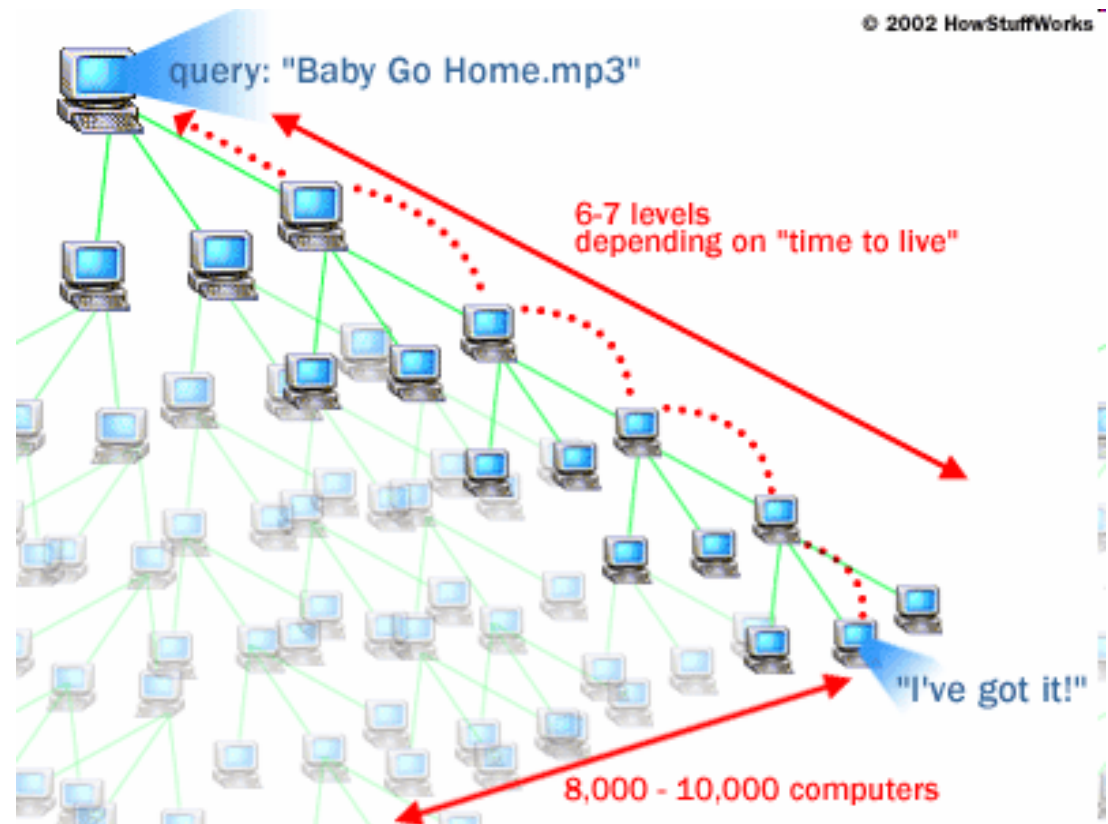
- § Share any type of files (not just music)
- § Completely decentralized method of searching for files
 - § applications connect to peer applications
- § each application instance serves to:
 - § store selected files
 - § route queries (file searches) from and to its neighboring peers
 - § respond to queries (serve file) if file stored locally
- § Gnutella history:
 - § 3/14/00: release by AOL, almost immediately withdrawn
 - § too late: 23K users on Gnutella at 8 am this AM
 - § reverse engineered. many iterations to fix poor initial design



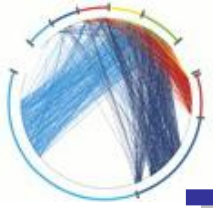
Gnutella

Searching by **flooding**:

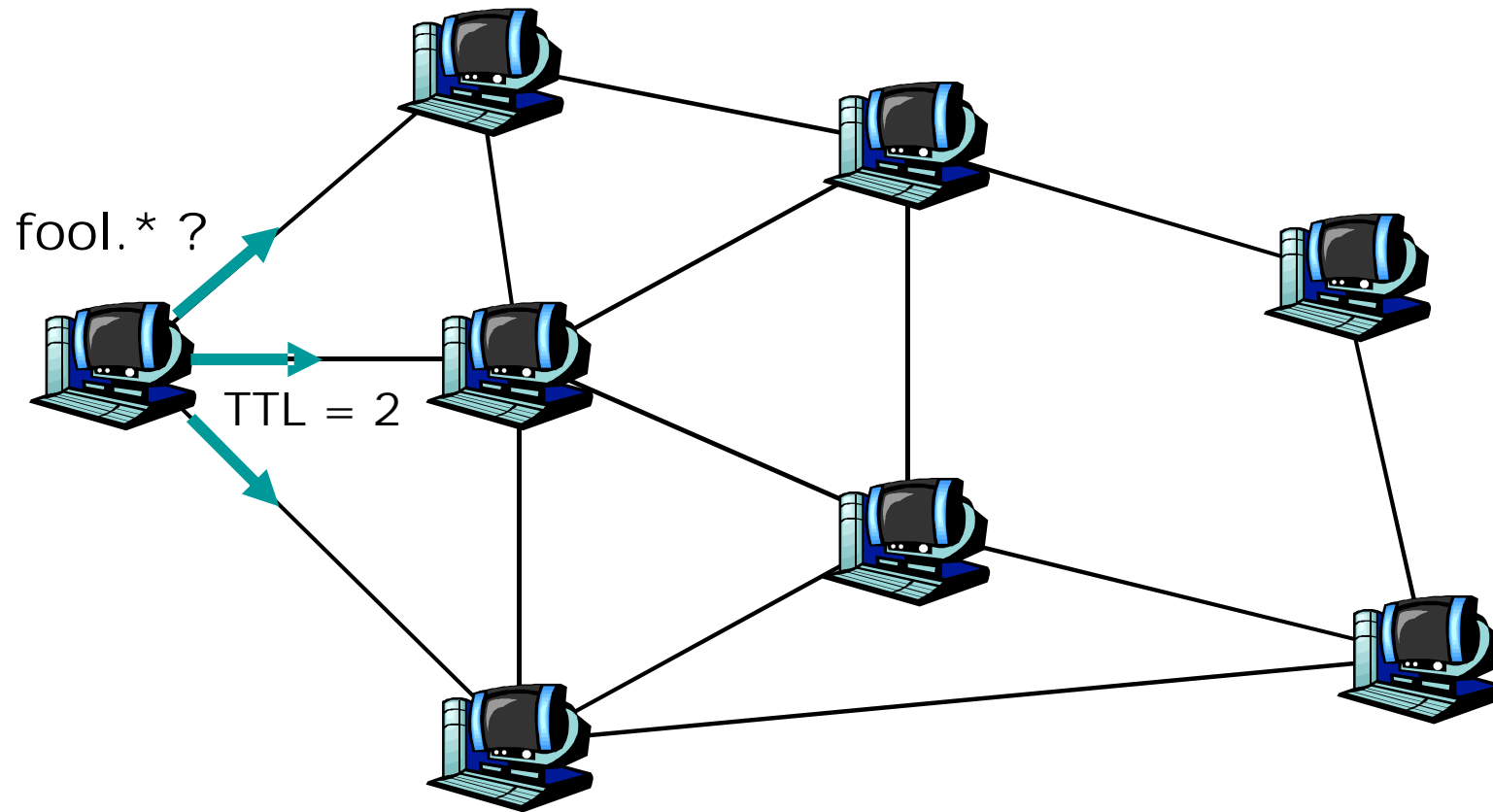
- § If you don't have the file you want, query 7 of your neighbors.
- § If they don't have it, they contact 7 of their neighbors, for a maximum hop count of 10.
- § Requests are flooded, but there is no tree structure.
- § No looping but packets may be received twice.
- § Reverse path forwarding

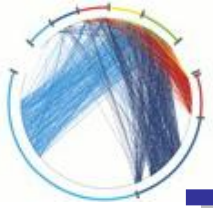


* Figure from <http://computer.howstuffworks.com/file-sharing.htm>

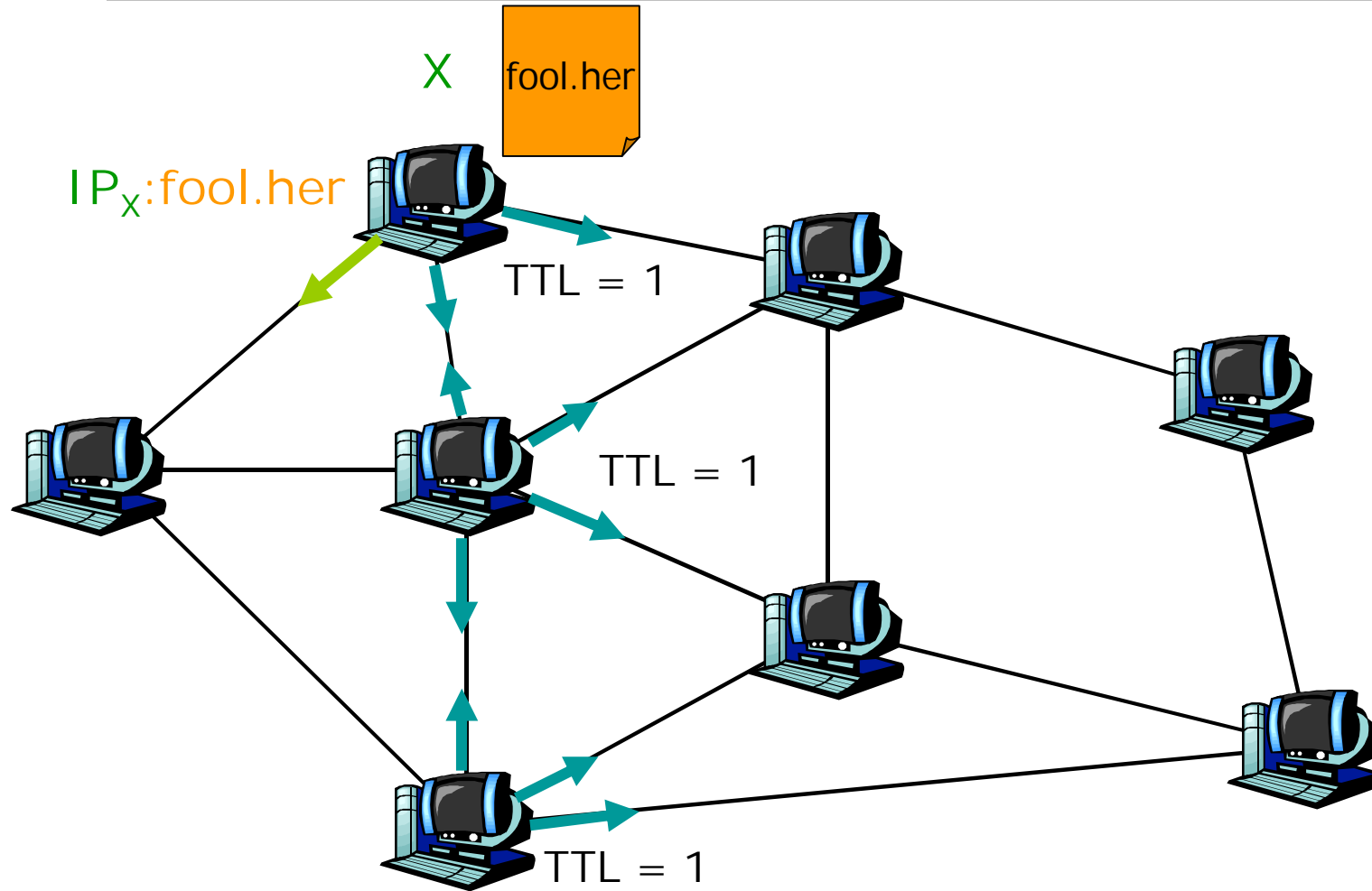


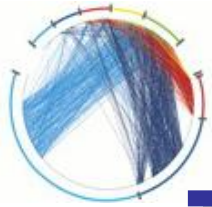
Gnutella



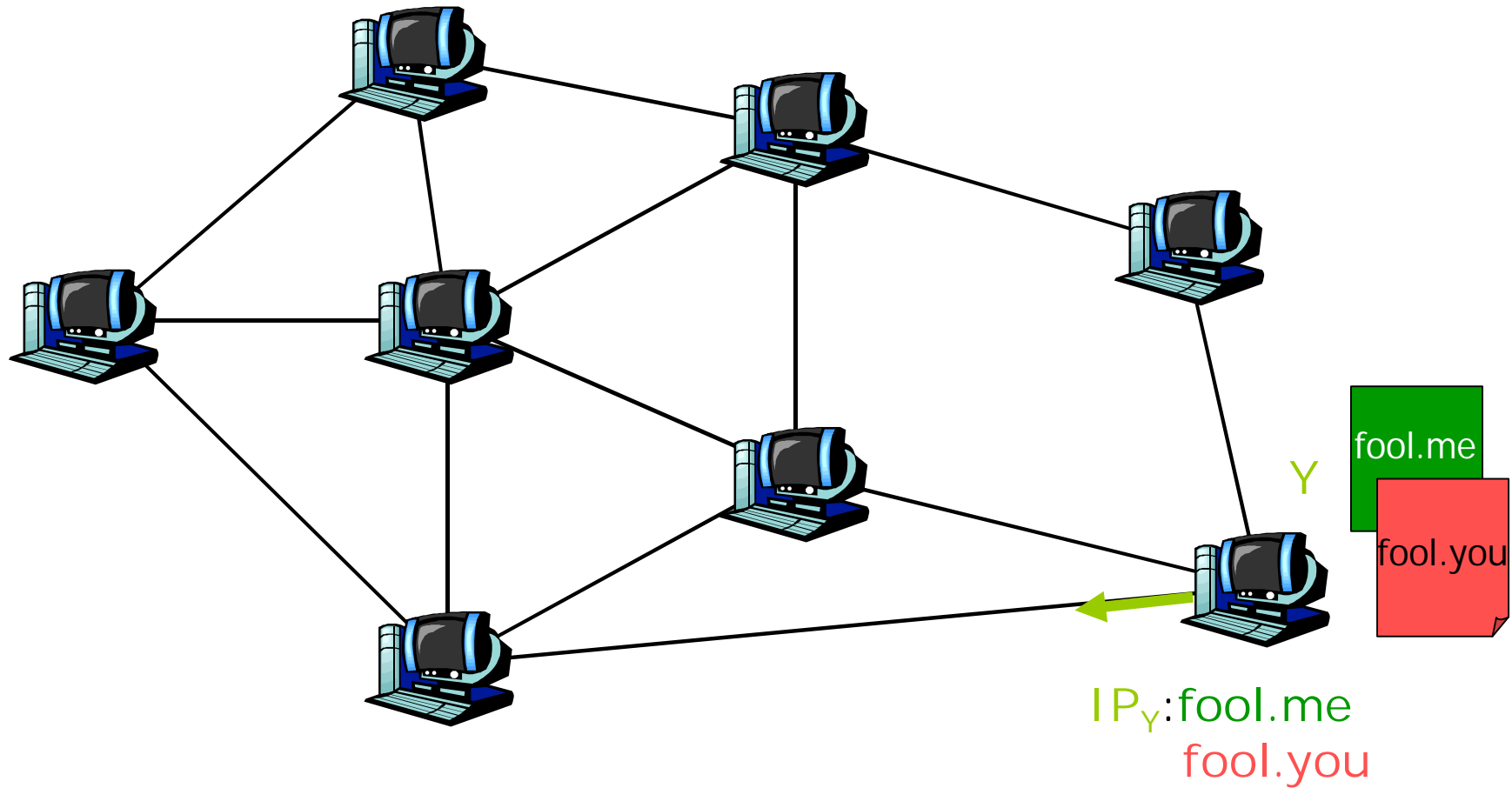


Gnutella



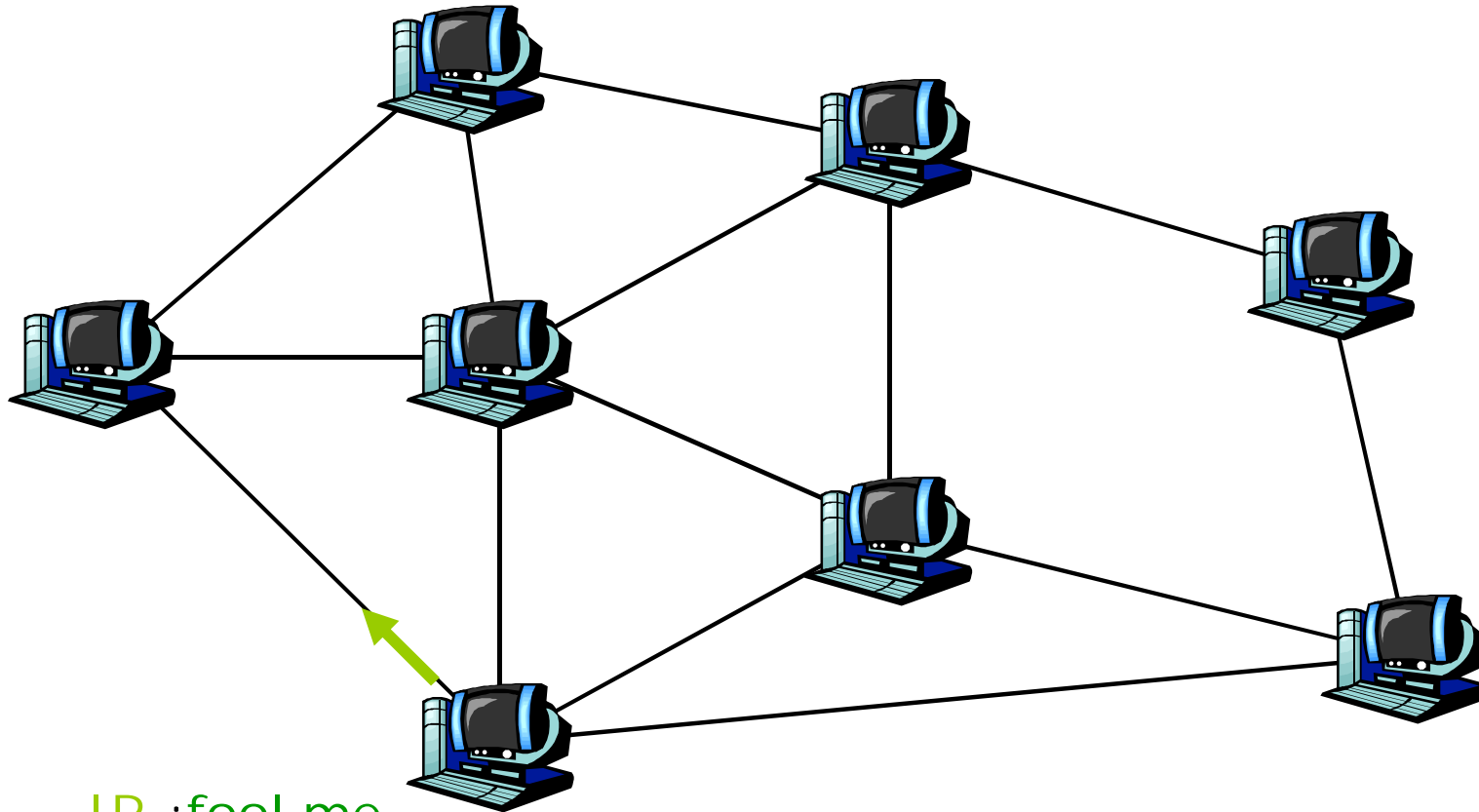


Gnutella

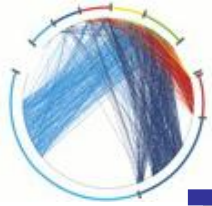




Gnutella



IP_y:fool.me
fool.you



Gnutella: strengths and weaknesses

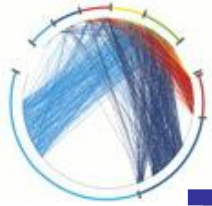
§ pros:

- flexibility in query processing
- complete decentralization
- simplicity
- fault tolerance/self-organization

§ cons:

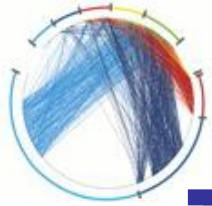
- ý severe scalability problems
- ý susceptible to attacks

§ Pure P2P system



Gnutella: initial problems and fixes

- § 2000: avg size of reachable network only 400-800 hosts. Why so small?
 - § **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 black holes
 - § preferential connection:
 - favors routing to well-connected peers
 - favors replies to clients that themselves serve large number of files: prevent freeloading

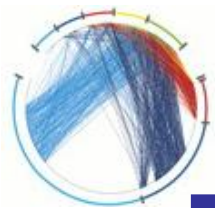


Kazaa (Fasttrack network)

- § Hybrid of centralized Napster and decentralized Gnutella
 - § hybrid P2P system

- § Super-peers act as local search hubs
 - § Each super-peer is similar to a Napster server for a small portion of the network
 - § Super-peers are automatically chosen by the system based on their capacities (storage, bandwidth, etc.) and availability (connection time)

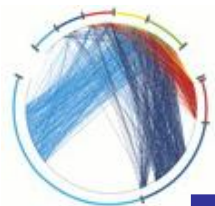
- § Users upload their list of files to a super-peer
- § Super-peers periodically exchange file lists
- § You send queries to a super-peer for files of interest



Anonymity

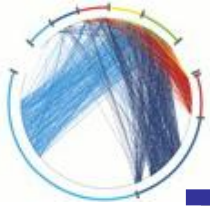
- § Napster, Gnutella, Kazaa don't provide anonymity
 - § Users know who they are downloading from
 - § Others know who sent a query

- § Freenet
 - § Designed to provide anonymity among other features

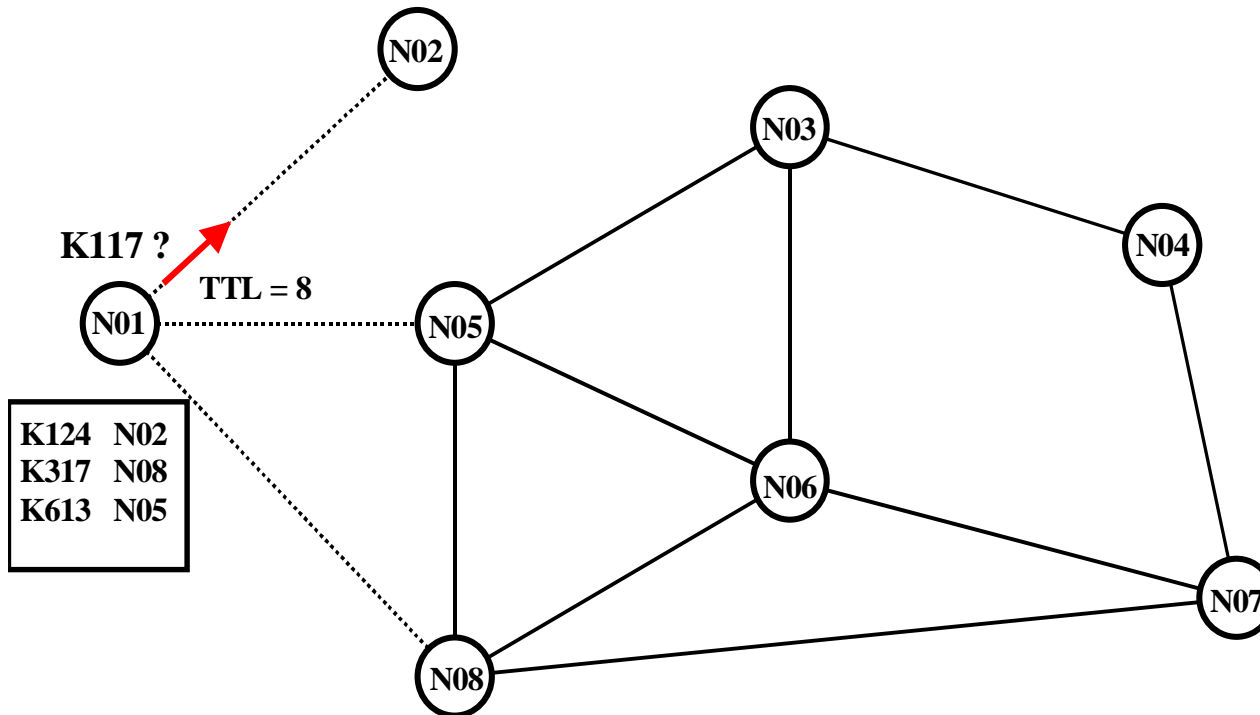


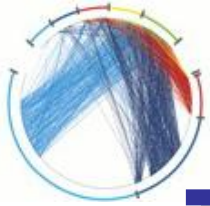
Freenet

- § Keys are mapped to IDs
- § Each node stores a **cache** of keys, and a **routing table** for **some** keys
 - § routing table defines the **overlay** network
- § Queries are routed to the node with the most **similar** key
- § Data flows in reverse path of query
 - § Impossible to know if a user is initiating or forwarding a query
 - § Impossible to know if a user is consuming or forwarding data
 - § Keys replicated in (some) of the nodes along the path

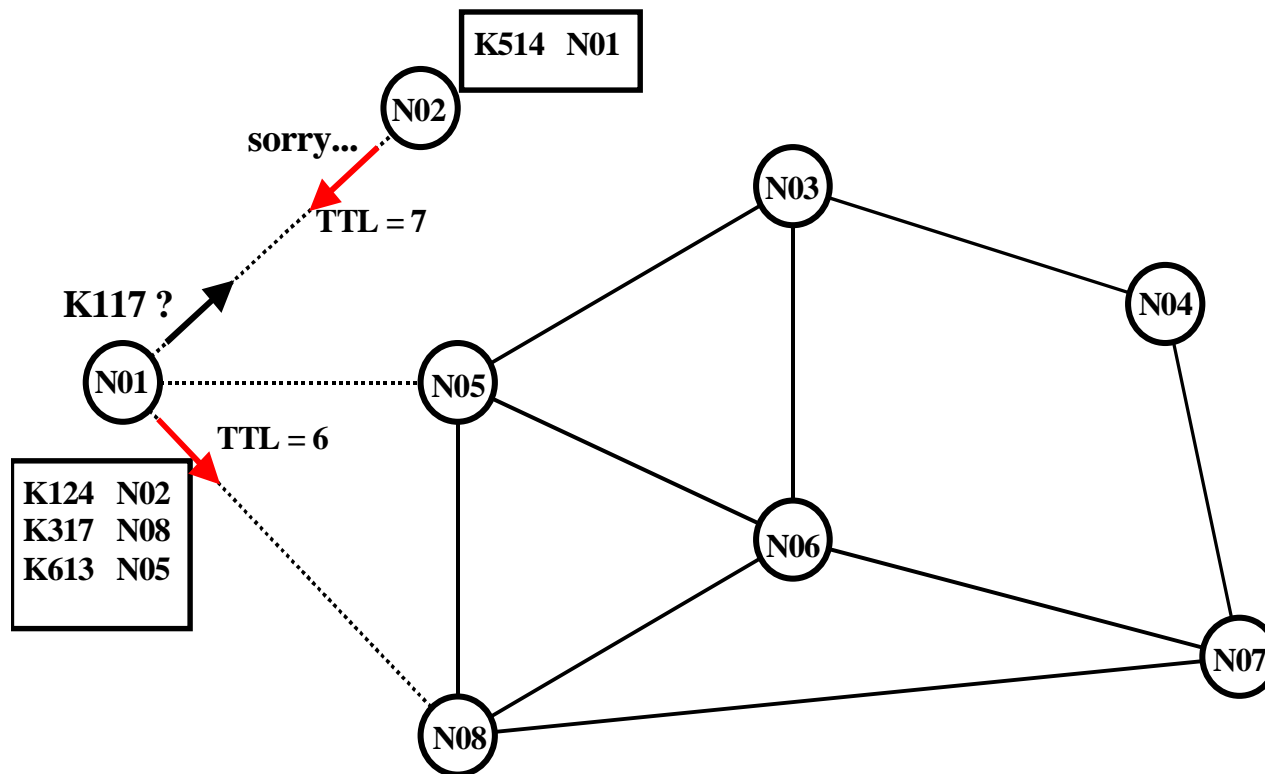


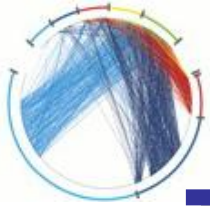
Freenet routing



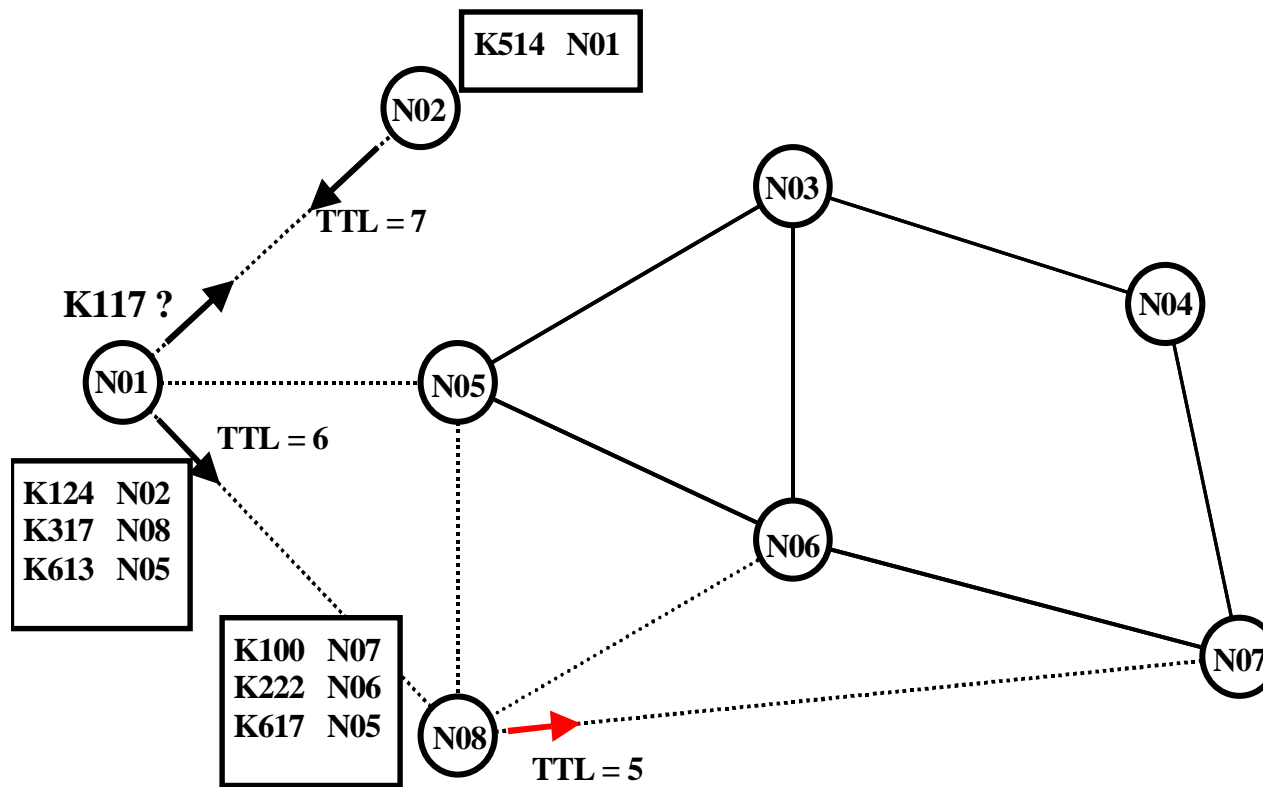


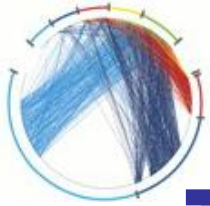
Freenet routing



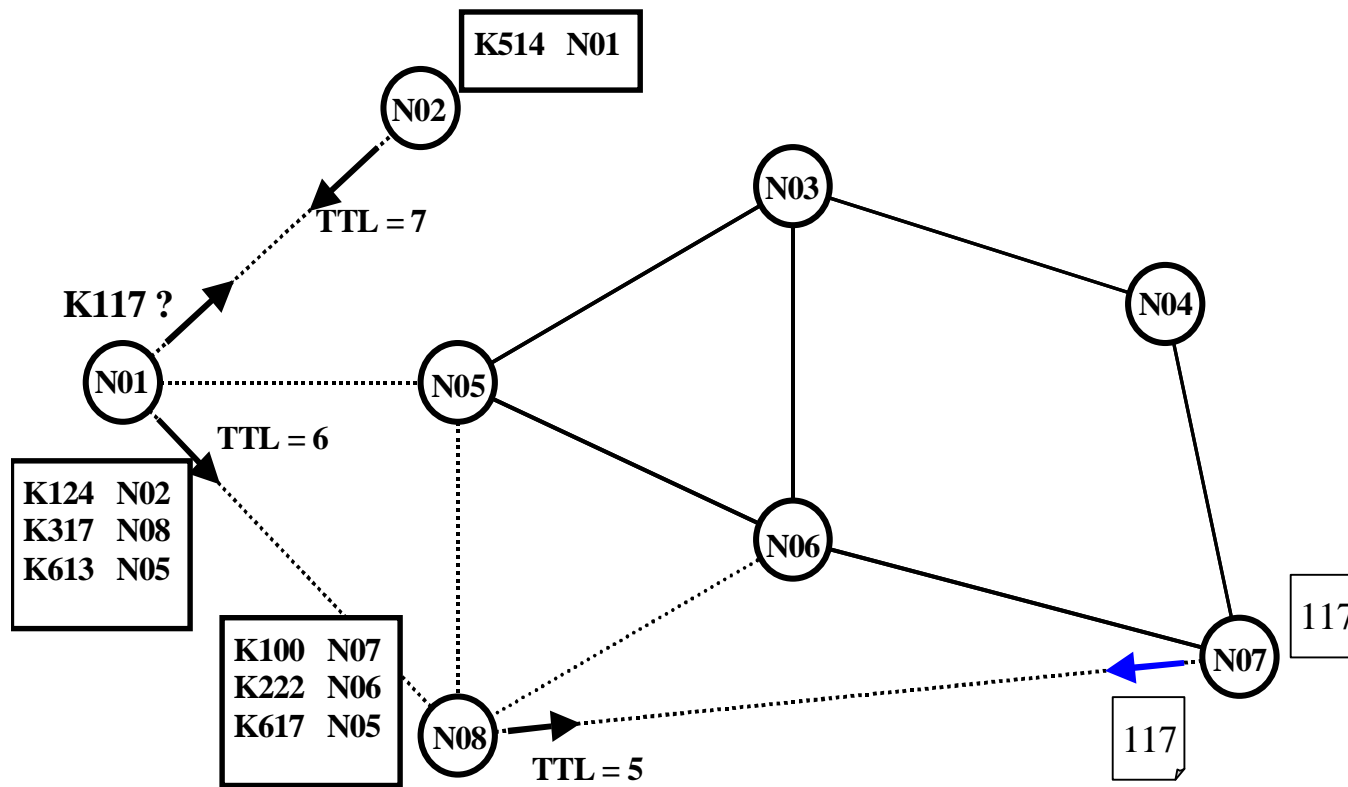


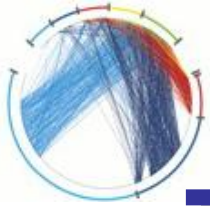
Freenet routing



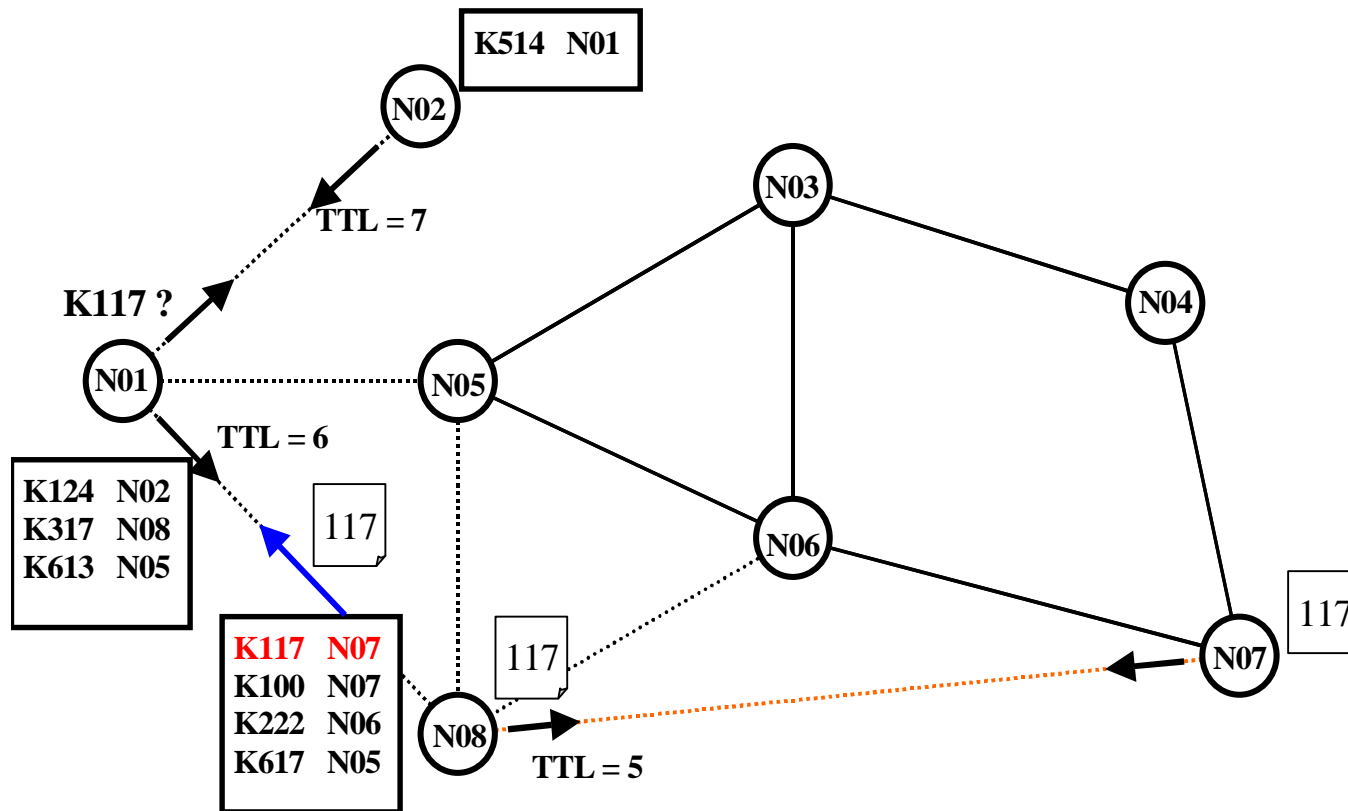


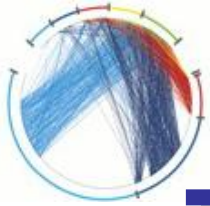
Freenet routing



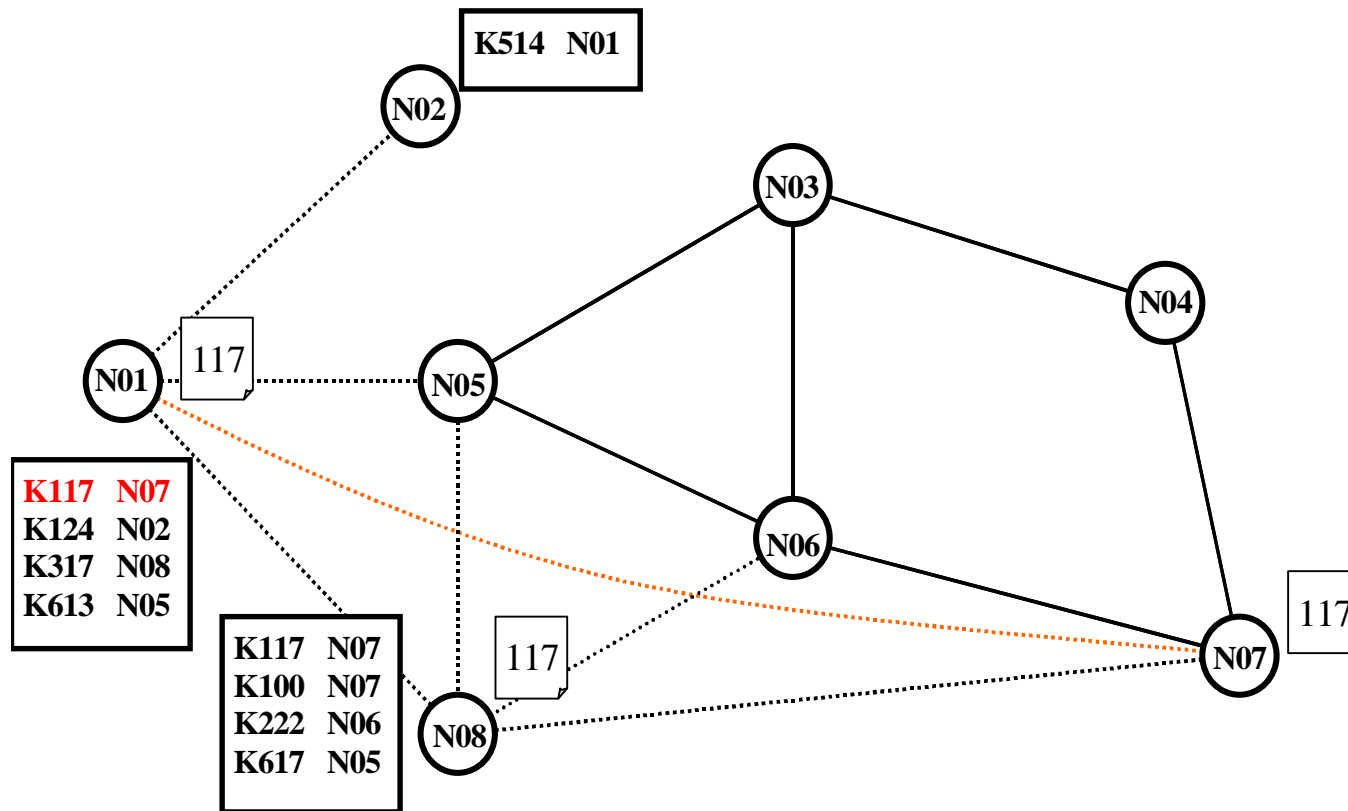


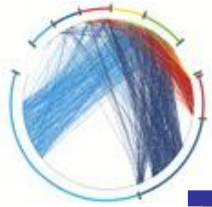
Freenet routing



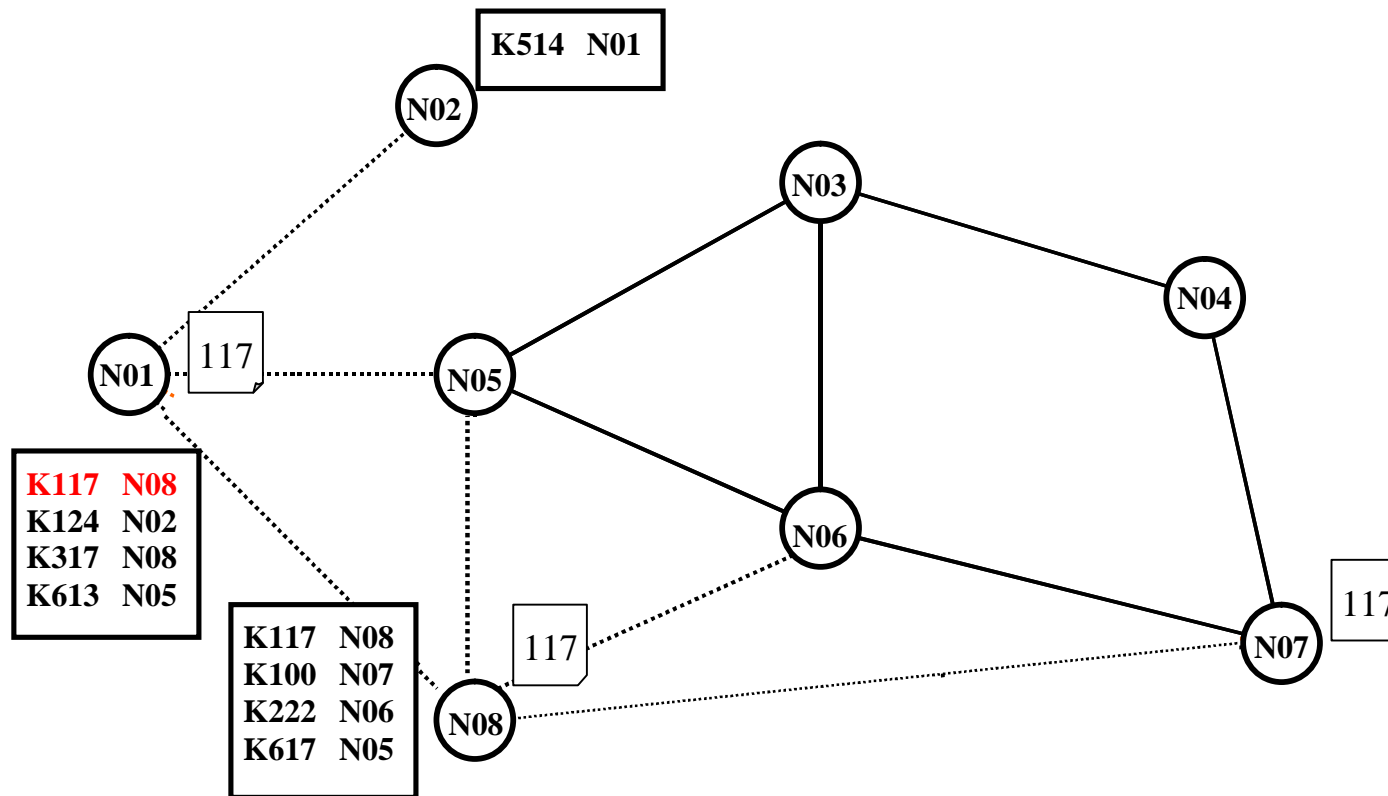


Freenet routing

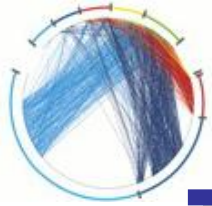




Freenet routing



Inserts are performed similarly – they are unsuccessful queries



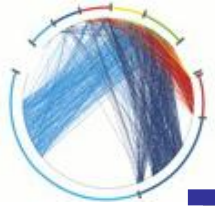
Freenet strengths and weaknesses

§ pros:

- ↳ complete decentralization
- ↳ fault tolerance/self-organization
- ↳ anonymity
- ↳ scalability (to some degree)

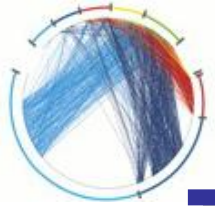
§ cons:

- ŷ questionable efficiency & performance
- ŷ rare keys disappear from the system
- ŷ improvement of performance requires high overhead (maintenance of additional information for routing)



Unstructured vs Structured P2P

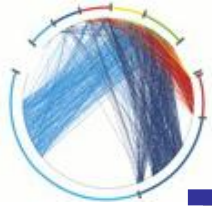
- § The systems we described do not offer any guarantees about their performance (or even correctness)
- § **Structured P2P**
 - § Scalable guarantees on numbers of hops to answer a query
 - § Maintain all other P2P properties (load balance, self-organization, dynamic nature)
- § Approach: **Distributed Hash Tables (DHT)**



Distributed Hash Tables (DHT)

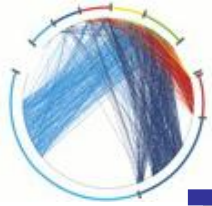
- § Distributed version of a hash table data structure
- § Stores (**key**, **value**) pairs
 - § The **key** is like a filename
 - § The **value** can be file contents, or pointer to location
- § **Goal**: Efficiently insert/lookup/delete (**key**, **value**) pairs

- § Each peer stores a subset of (**key**, **value**) pairs in the system
- § Core operation: Find node responsible for a **key**
 - § Map **key** to node
 - § Efficiently route insert/lookup/delete request to this node
- § **Allow** for frequent node arrivals/departures



DHT Desirable Properties

- § Keys should be mapped evenly to all nodes in the network (load balance)
- § Each node should maintain information about only a few other nodes (scalability, low update cost)
- § Messages should be routed to a node efficiently (small number of hops)
- § Node arrival/departures should only affect a few nodes



DHT Routing Protocols

§ DHT is a generic **interface**

§ There are several **implementations** of this interface

§ Chord [MIT]

§ Pastry [Microsoft Research UK, Rice University]

§ Tapestry [UC Berkeley]

§ Content Addressable Network (CAN) [UC Berkeley]

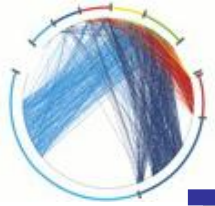
§ Viceroy [Israel, UC Berkeley]

§ SkipNet [Microsoft Research US, Univ. of Washington]

§ Kademlia [New York University]

§ P-Grid [EPFL Switzerland]

§ Freenet [Ian Clarke]



Basic Approach

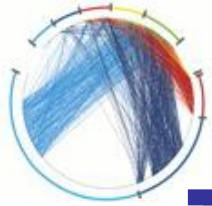
In all approaches:

- § **keys** are associated with globally unique **IDs**
 - § integers of size **m** (for large **m**)

- § **key ID** space (search space) is uniformly populated - mapping of keys to **IDs** using (**consistent**) hashing

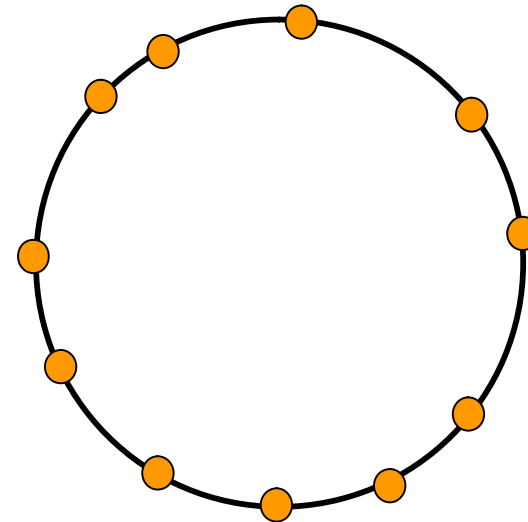
- § a node is responsible for indexing all the keys in a certain subspace (zone) of the **ID space**

- § nodes have only partial knowledge of other node's responsibilities



Consistent Hashing

§ Example: map the **keys** to the ring

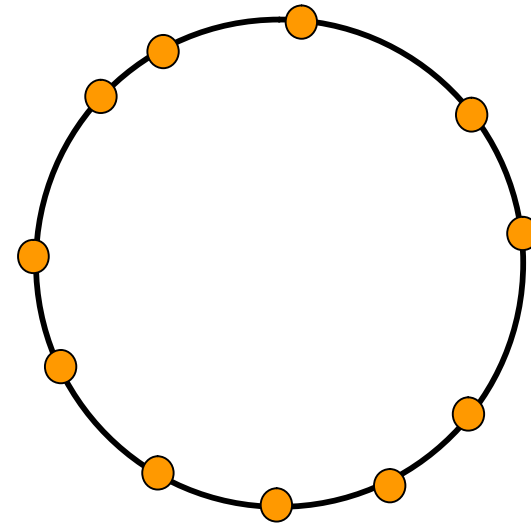


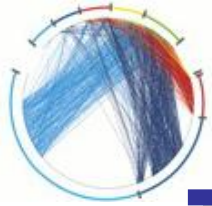
The ring is just a possibility.
Any metric space will do



Consistent Hashing

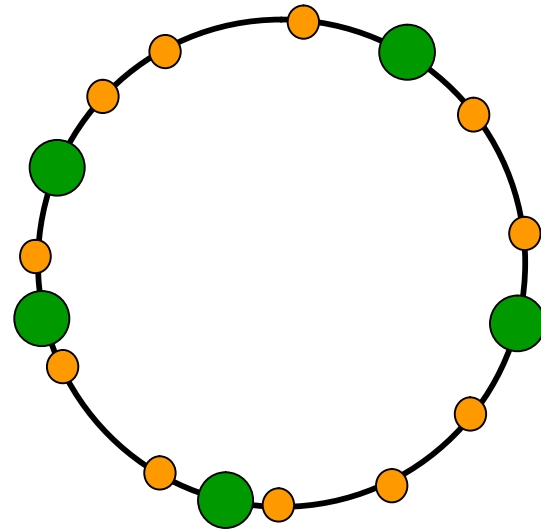
§ How do we distribute the keys uniformly to the nodes?

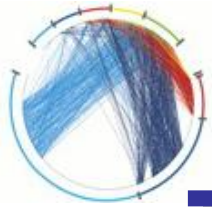




Consistent Hashing

§ The main idea: map **both keys and nodes** (node IPs) to the same (metric) ID space





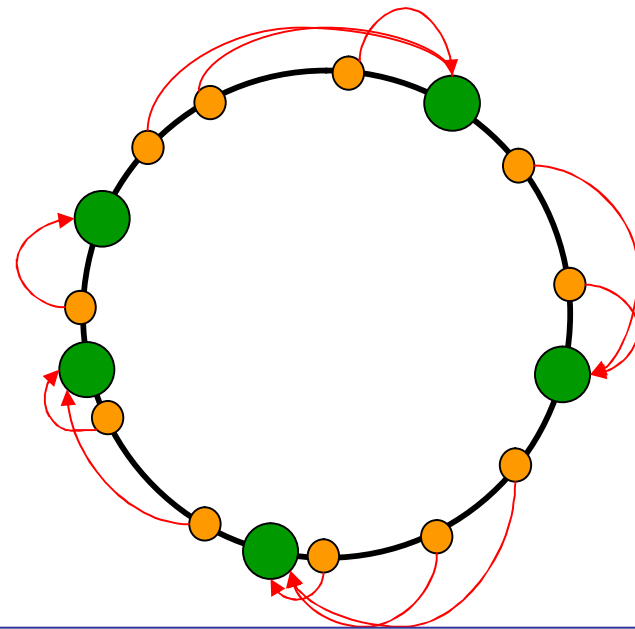
Consistent Hashing

§ The main idea: map both **keys** and **nodes** (node **IPs**) to the same (metric) **ID space**

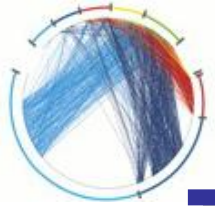
§ Each **key** is assigned to the **node** with **ID** clockwise closest to the key **ID**

§ uniformly distributed

§ at most logarithmic number of keys assigned to each node



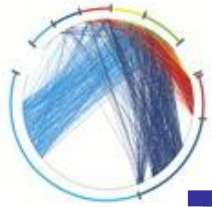
Problem: Starting from a node, how do we locate the node responsible for a key, while maintaining as little information about other nodes as possible



Basic Approach Differences

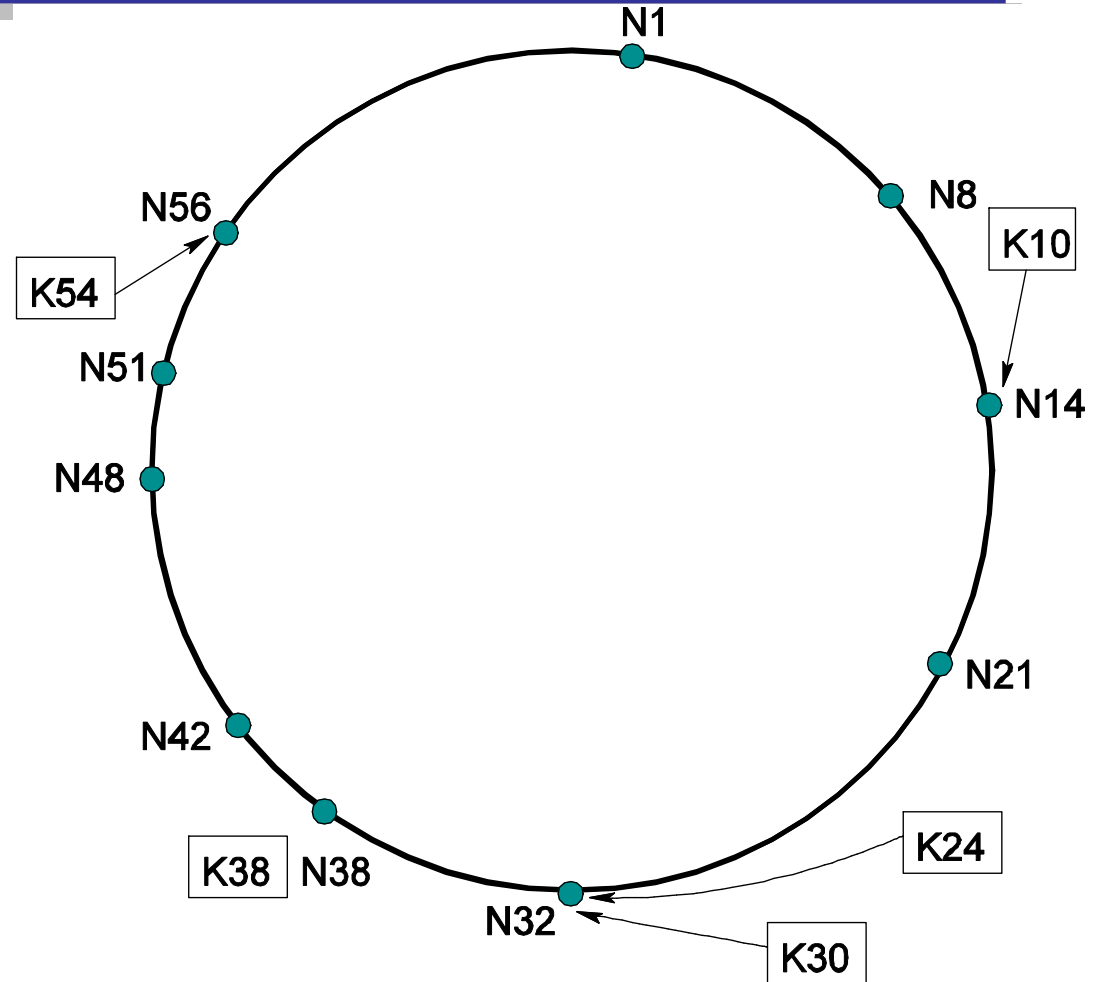
- § Different P2P systems differ in:
 - § the choice of the ID space
 - § the structure of their network of nodes (i.e. how each node chooses its neighbors)

- § The goals are
 - § search in logarithmic time
 - § insertions and deletions in logarithmic time
 - § maintain good load balance

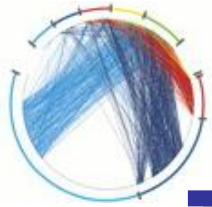


Chord

- § Nodes organized in an **identifier circle** based on node identifiers
- § Keys assigned to their **successor** node in the identifier circle
- § Hash function ensures even distribution of nodes and keys on the circle



* All Chord figures from "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications", Ion Stoica et al., IEEE/ACM Transactions on Networking, Feb. 2003.

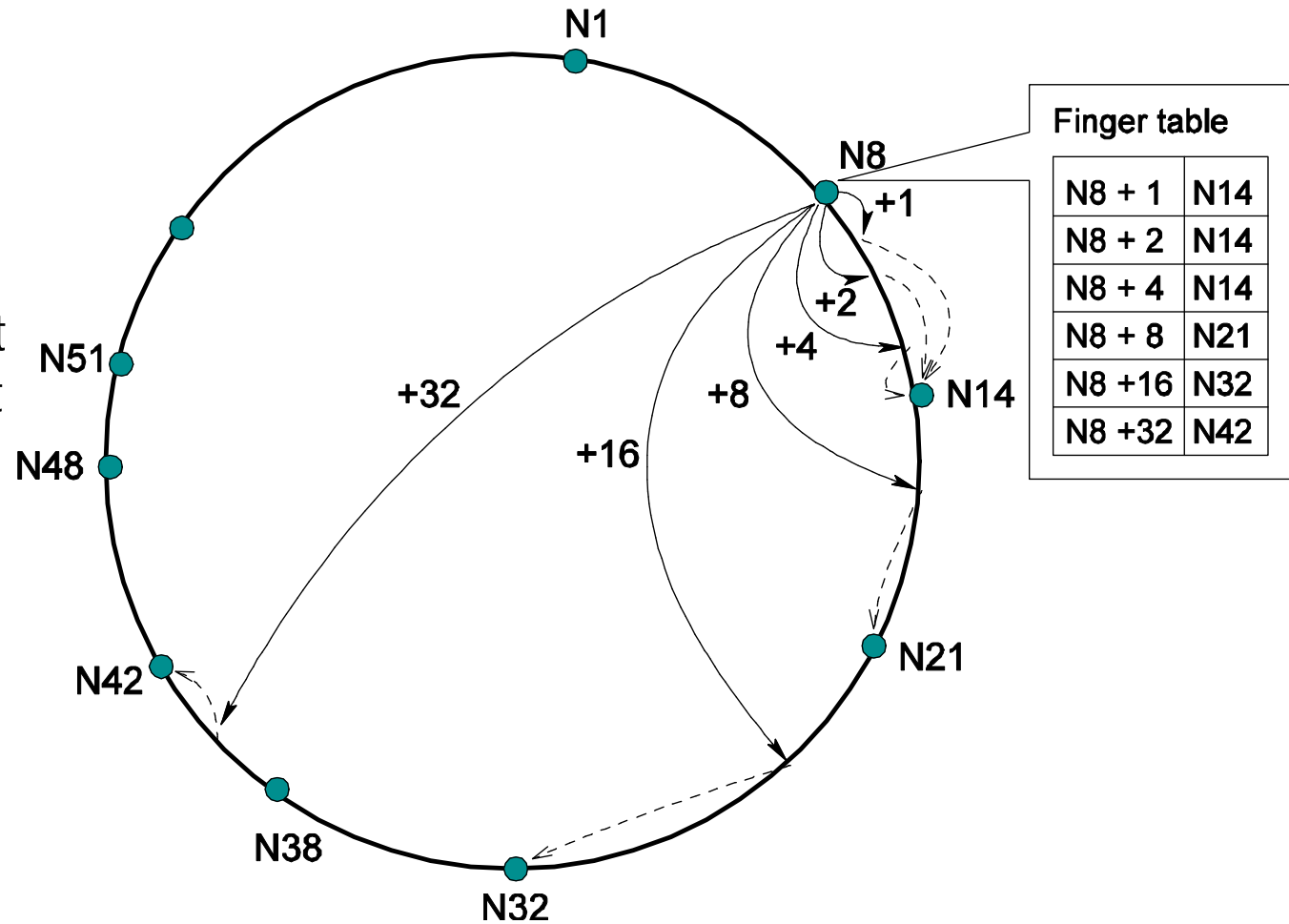


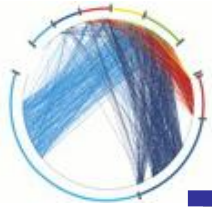
Chord Finger Table

§ $O(\log N)$ table size

§ i^{th} finger points to first node that succeeds n by at least 2^{i-1}

§ maintain also pointers to predecessors (for correctness)

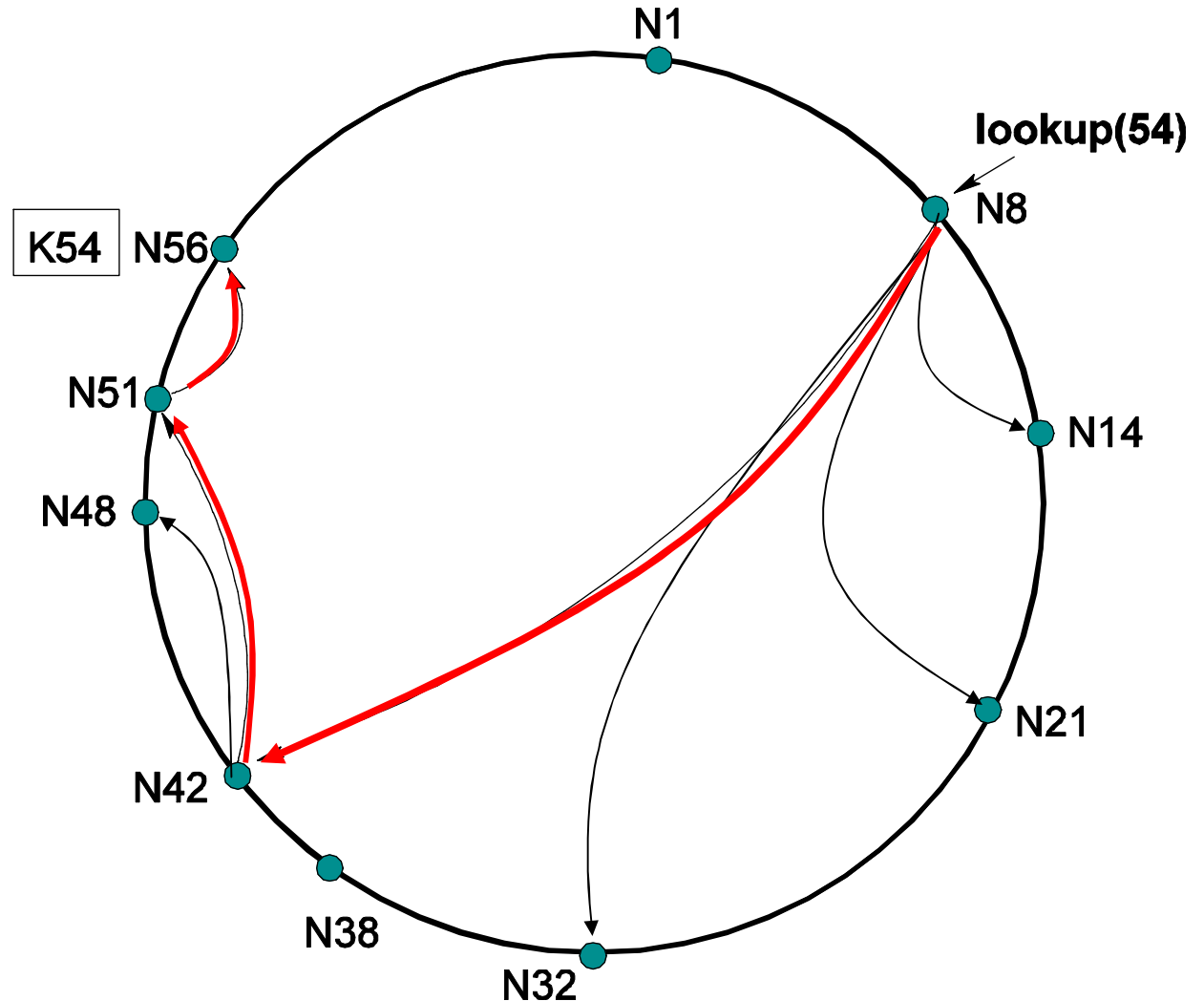


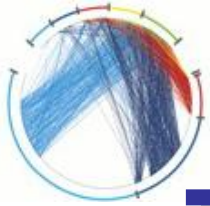


Chord Key Location

§ Lookup in finger table the furthest node that precedes key

§ Query homes in on target in $O(\log N)$ hops





Chord node insertion

Insert node **N40**:

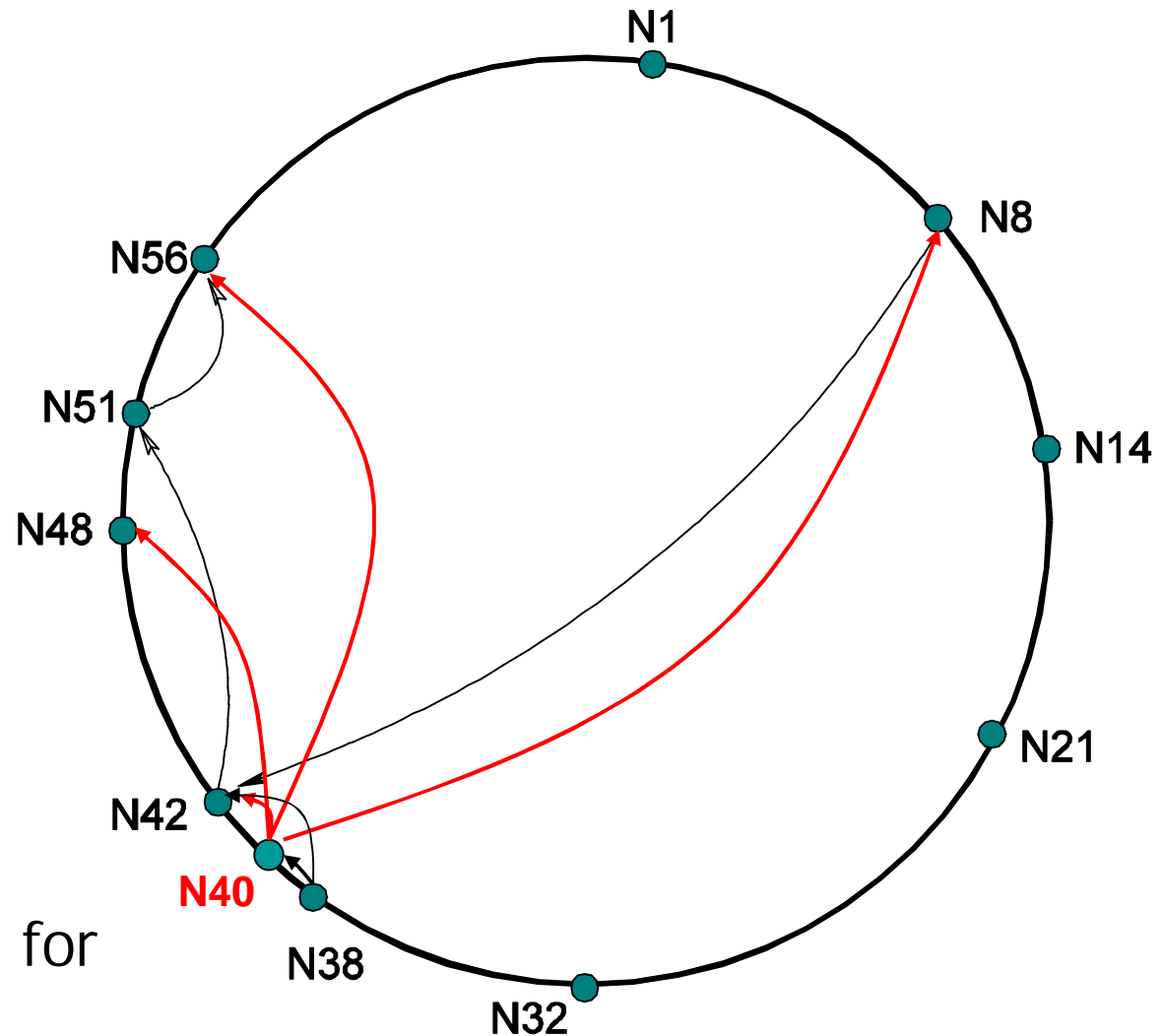
Locate node

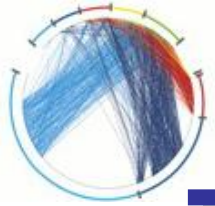
Add fingers

Update successor
pointers and other
node's fingers
(max in-degree
 $O(\log^2 n)$ whp)

Time $O(\log^2 n)$

Stabilization protocol for
refreshing links

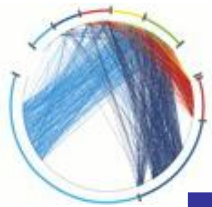




Chord Properties

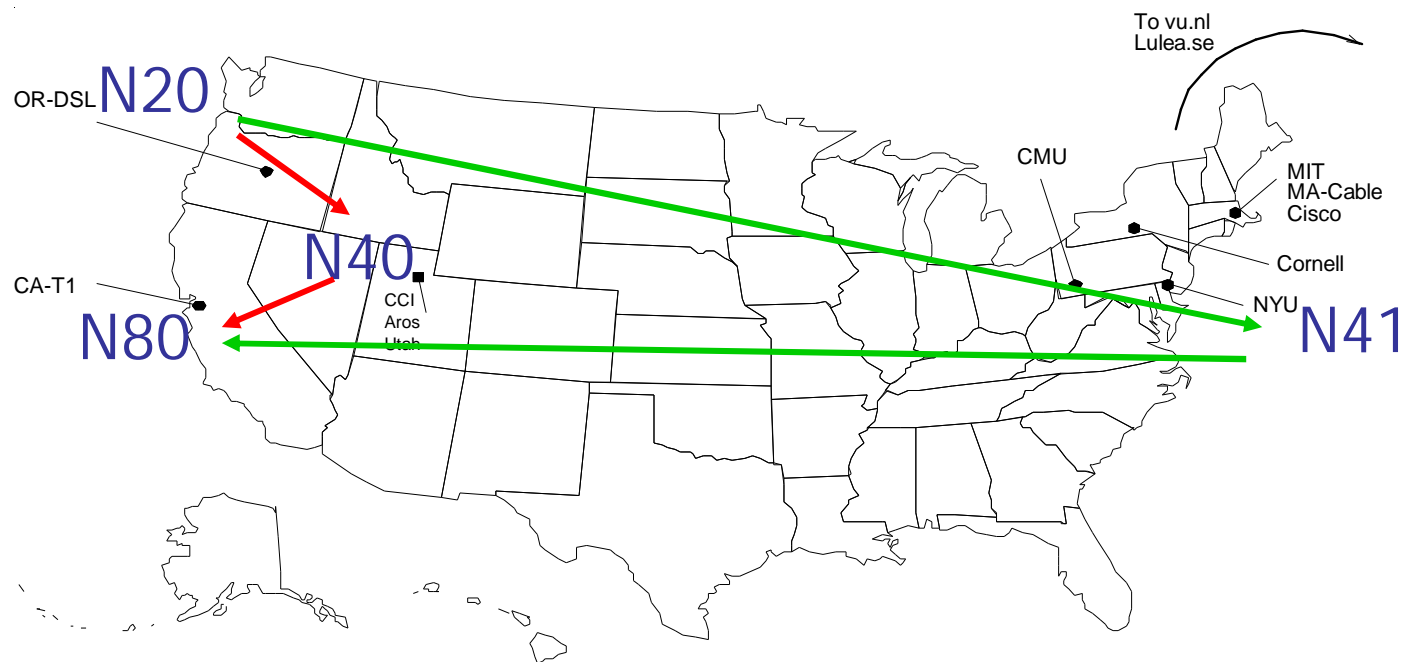
- § In a system with N nodes and K keys, with **high probability**...
 - § each node receives at most K/N keys
 - § each node maintains info. about $O(\log N)$ other nodes
 - § lookups resolved with $O(\log N)$ hops
 - § Insertions $O(\log^2 N)$

- § In practice never stabilizes
- § No consistency among replicas
- § Hops have poor network locality

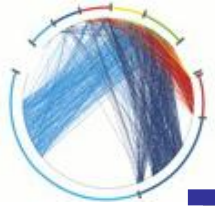


Network locality

§ Nodes close on ring can be far in the network.



* Figure from <http://project-iris.net/talks/dht-toronto-03.ppt>



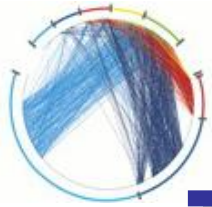
Plaxton's Mesh

§ map the nodes and keys to b -ary numbers of m digits

§ assign each key to the node with which it shares the largest prefix

§ e.g. $b = 4$ and $m = 6$

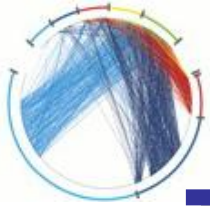
321302 → 321333
321002



Plaxton's Mesh – Routing Table

§ for $b = 4$, $m = 6$, $\text{nodeID} = 110223$; routing table:

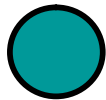
	$d = 0$	$d = 1$	$d = 2$	$d = 3$
$p = 0$	032130	1	210231	303213
$p = 1$	103002	1	123011	133233
$p = 2$	0	111210	112301	113331
$p = 3$	110031	110122	2	110310
$p = 4$	110200	110212	2	110232
$p = 5$	110220	110221	110222	3



Plaxton algorithm: routing

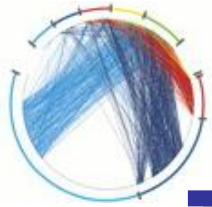
Move closer to the target one digit at the time

locate
322210



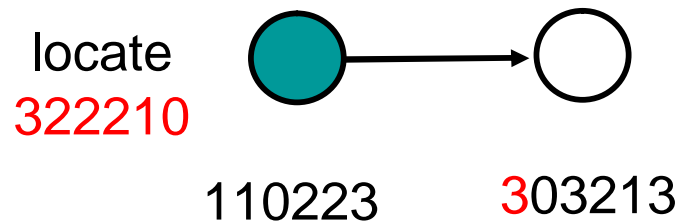
110223

	d = 0	d = 1	d = 2	d = 3
p = 0	032130	1	210231	303213
p = 1	103002	1	123011	133233
p = 2	0	111210	112301	113331
p = 3	110031	110122	2	110310
p = 4	110200	110212	2	110232
p = 5	110220	110221	110222	3

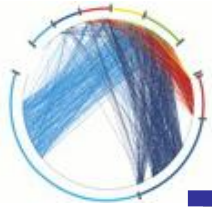


Plaxton algorithm: routing

Move closer to the target one digit at the time

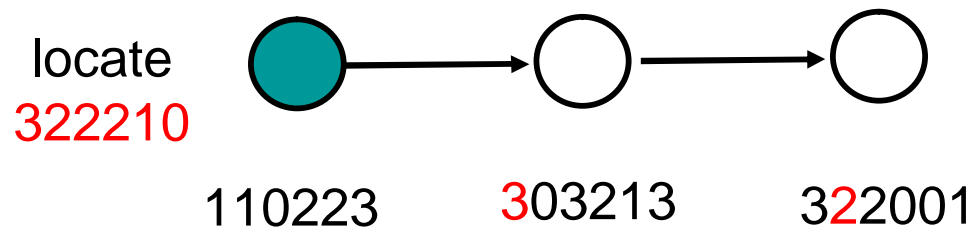


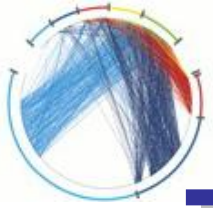
	d = 0	d = 1	d = 2	d = 3
p = 0	032130	1	210231	303213
p = 1	103002	1	123011	133233
p = 2	0	111210	112301	113331
p = 3	110031	110122	2	110310
p = 4	110200	110212	2	110232
p = 5	110220	110221	110222	3



Plaxton algorithm: routing

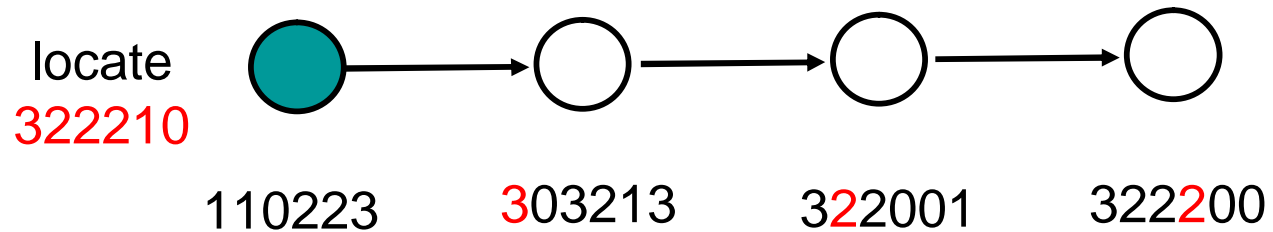
Move closer to the target one digit at the time





Plaxton algorithm: routing

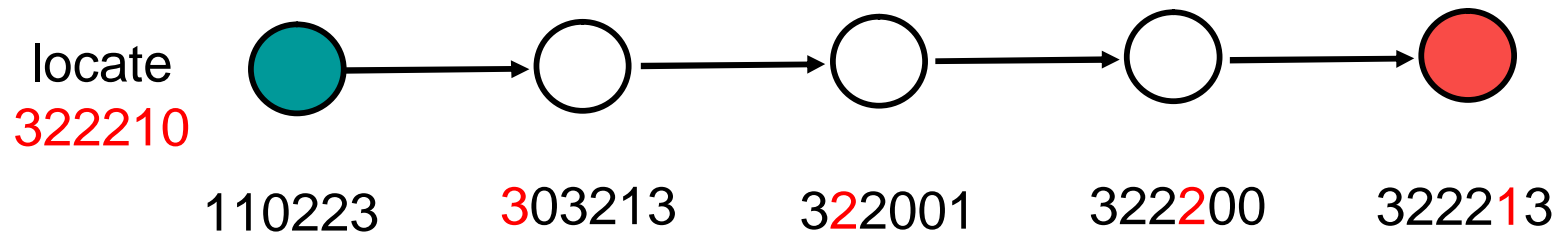
Move closer to the target one digit at the time

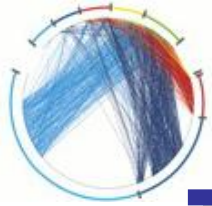




Plaxton algorithm: routing

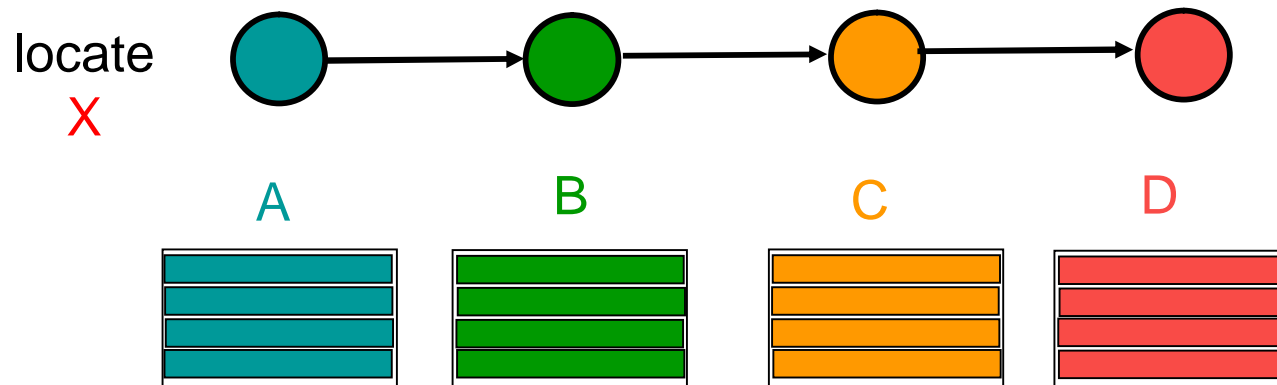
Move closer to the target one digit at the time





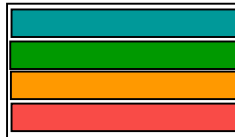
Pastry: Node Joins

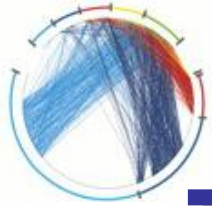
§ Node **X** finds the closest (in network proximity) node and makes a query with its own ID



§ Routing table of **X**

§ the *i*-th row of the routing table is the *i*-th row of the *i*-th node along the search path for **X**

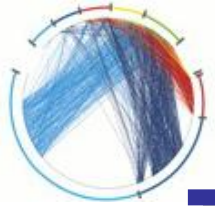




Enforcing Network Locality - Pastry

§ For the (i,j) entry of the table select the node that is geographically closer to the current node.

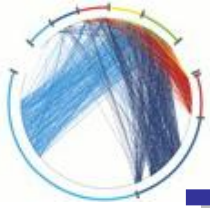
110223	$d = 0$	$d = 1$	$d = 2$	$d = 3$
$p = 0$	032130	1	210231	303213
$p = 1$	103002	1	123011	133233
$p = 2$	0	111210	112301	113331
$p = 3$	110031	110122	2	110310
$p = 4$	110200	110212	2	110232
$p = 5$	110220	110221	110222	3



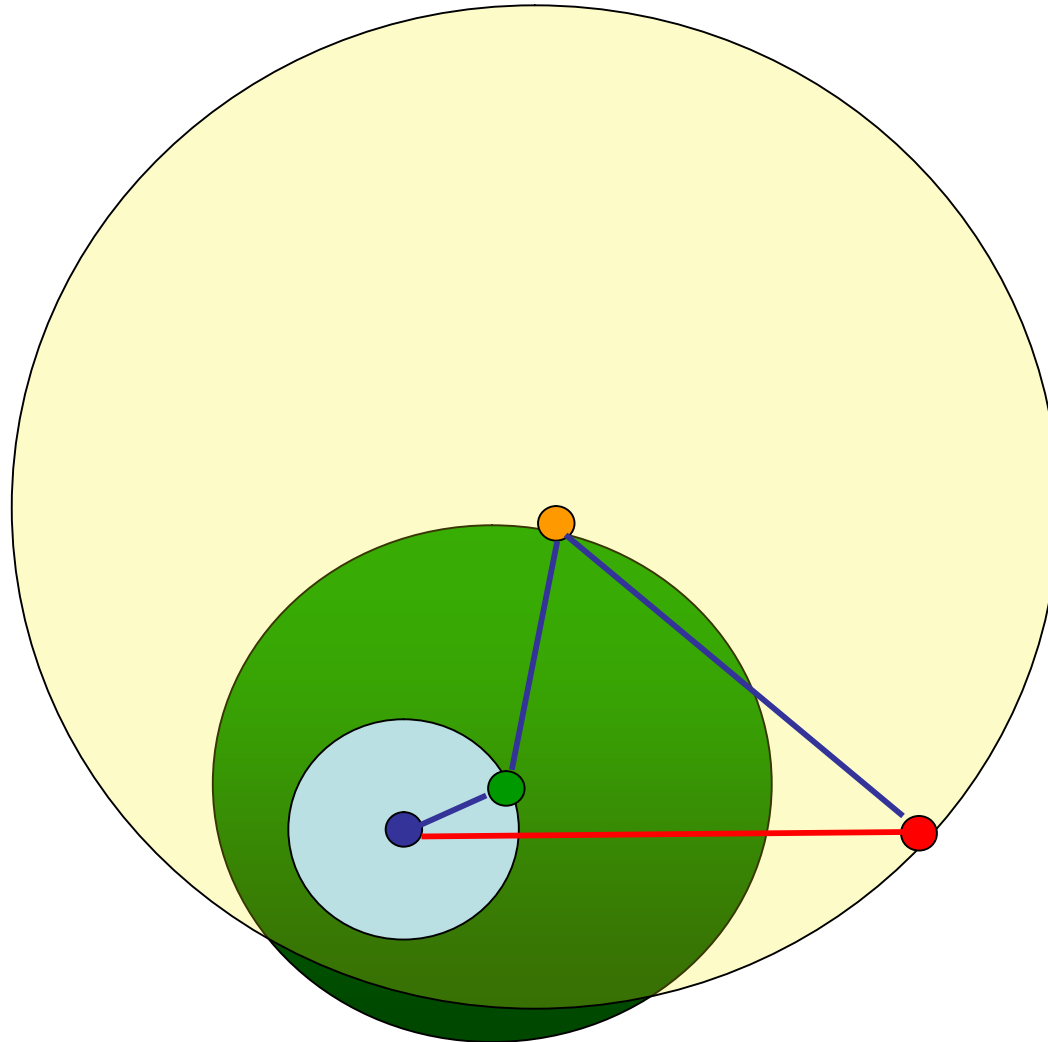
Enforcing Network Locality -Pastry

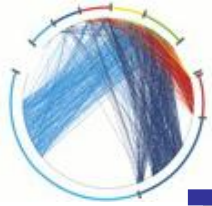
§ Critical property

- § for larger row numbers the number of possible choices decreases exponentially
 - in row $i+1$ we have $1/b$ the choices we had in row i
- § for larger row numbers the expected distance to the nearest neighbor increases exponentially
- § the distance of the source to the target is approximately equal to the distance in the last step – as a result it is well approximated



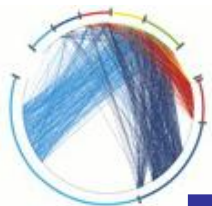
Enforcing Network Locality - Pastry





Network Proximity

- § The starting node **A** is the closest one to node **X**, so by triangular inequality the neighbors in first row of the starting node **A** will also be close to **X**
- § For the remaining entries of the table the same argument applies: the distance of the intermediate node **Y** to its neighbors dominates the distance from **X** to the intermediate node **Y**



CAN

- § Search space: d-dimensional coordinate space (on a **d-torus**)
- § Each node owns a distinct **zone** in the space
- § Each node keeps links to the nodes responsible for zones adjacent to its zone (in the search space) – $\sim 2d$ on avg
- § Each key hashes to a **point** in the space

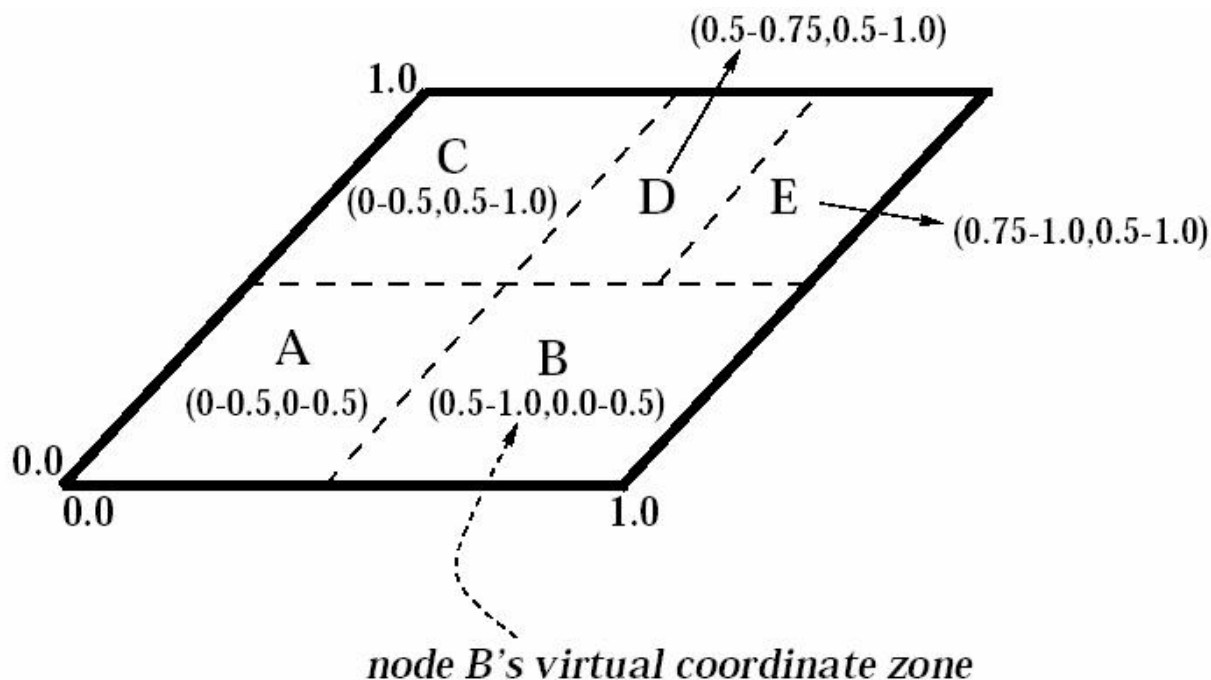
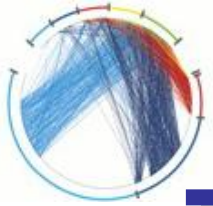


Figure 1: Example 2-d space with 5 nodes

* Figure from "A Scalable Content-Addressable Network", S. Ratnasamy et al., In Proceedings of ACM SIGCOMM 2001.



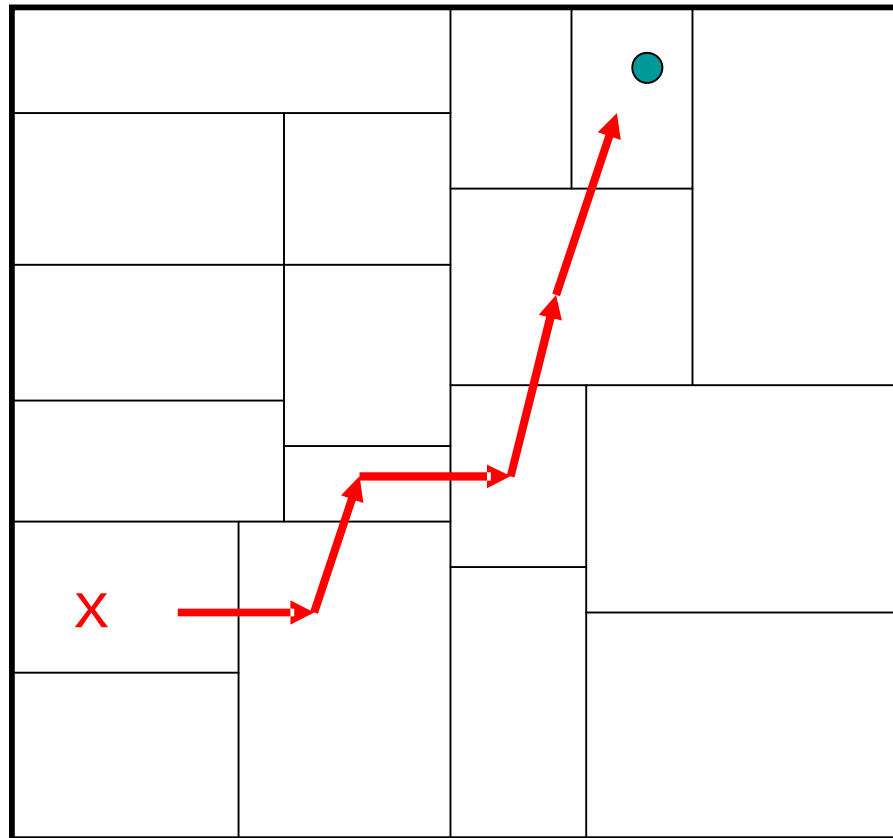
CAN Lookup

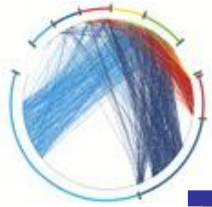
Node x wants to
lookup key K

$K \rightarrow (a, b)$

Move along neighbors
to the zone of the key
each time moving
closer to the key

expected time $O(dn^{1/d})$
can we do it in $O(\log n)$?



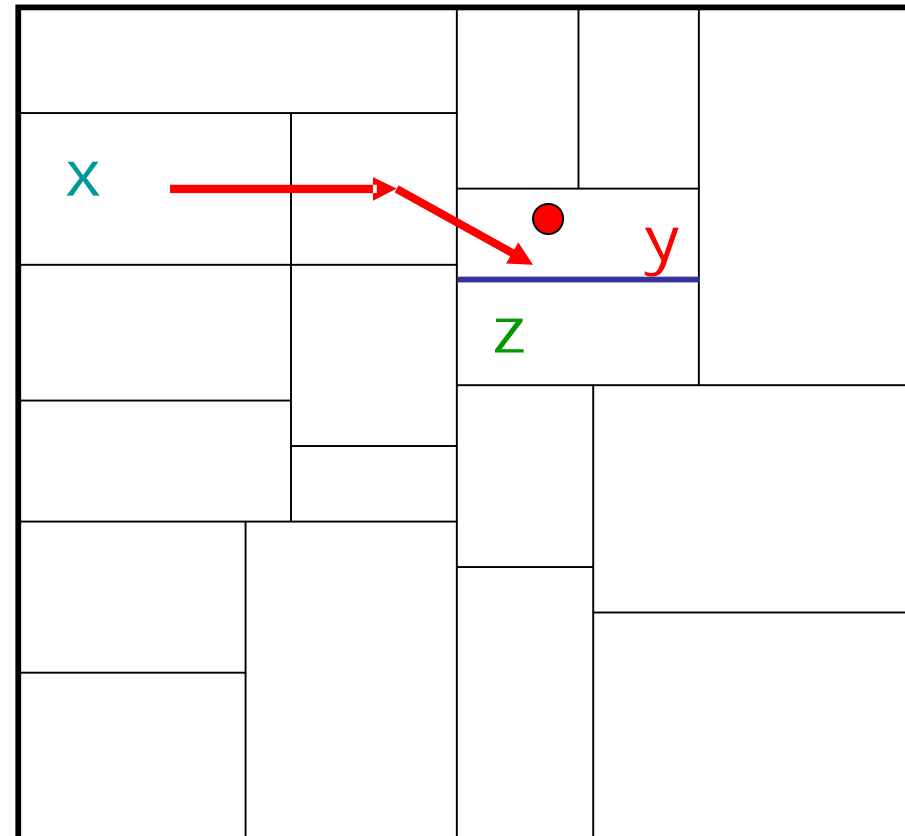


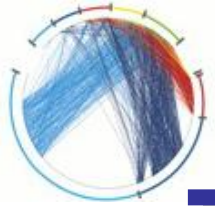
CAN node insertion

Node y needs to be inserted
It has knowledge of node x

IP of $y \rightarrow (c,d)$
zone belongs to z

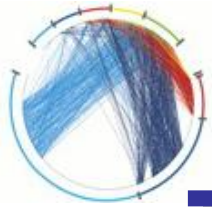
Split z 's zone





Kademlia

§ The routing idea is similar to Plaxton's mesh: improve closeness one bit at the time.



Kademlia – Hashing and distance

- § Nodes and Keys are mapped to m -bit **binary** strings
- § Distance between two identifiers: the XOR string, as a binary number

$$x = 010110$$

$$y = 011011$$

$$x \oplus y = 001101$$

$$d(x,y) = 13$$

XOR: 1 if two bits are different
0 if two bits are the same

- § If x and y agree in the first i digits and disagree in the $(i+1)$ then $2^i \leq d(x,y) \leq 2^{i+1}-1$

$$x = 010110$$

$$y = 011110$$

$$x \oplus y = 001000$$

$$d(x,y) = 8$$

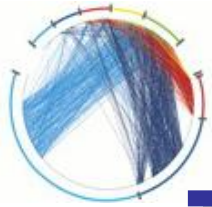
$$x = 010110$$

$$y = 011001$$

$$x \oplus y = 001111$$

$$d(x,y) = 15$$

symmetric
unidirectional



Kademlia – Routing table

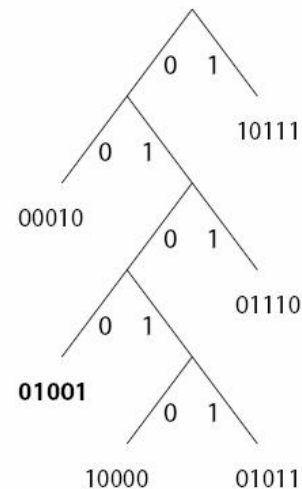
§ Each node with ID x , stores m k -buckets

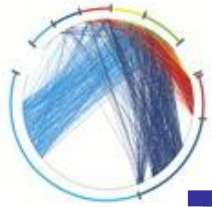
§ a k -bucket stores k nodes that are at distance $[2^i, 2^{i+1})$

- empty bucket if no nodes are known

Routing table for node 01001

$k = 1$





Kademlia – Updating buckets

- § Whenever a node receives any message, it updates the appropriate k-bucket
- § If the bucket is full the least-recently node is removed **if** it is not live

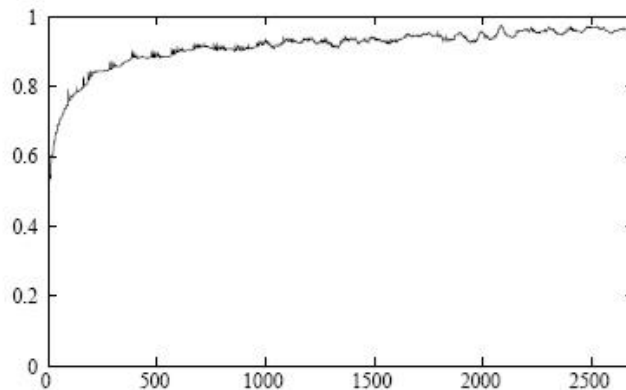
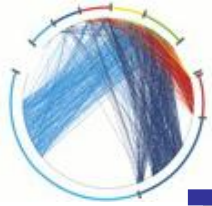
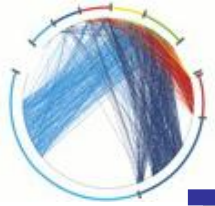


Figure 1: Probability of remaining online another hour as a function of uptime. The x axis represents minutes. The y axis shows the the fraction of nodes that stayed online at least x minutes that also stayed online at least $x + 60$ minutes.



Kademlia – Node lookup

- § The lookup process is **iterative**: everything is controlled by the initiator node
 1. query in **parallel** the α nodes closest to the query ID
 2. nodes return the k nodes closest to the query ID
 3. go back to step 1, and select the α nodes from the new set of nodes
 4. Terminate when you have the k closest nodes
- § Key lookups are done in a similar fashion, but they terminate when the key is found
- § the requesting node caches the key locally.



Kademlia – Other operations

§ Refresh

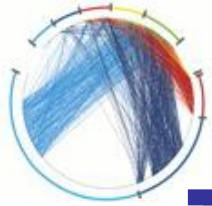
§ periodically, all k-buckets are refreshed by making a query for a value within the bucket

§ Node Joins

§ contact a participating node and insert it in the appropriate bucket

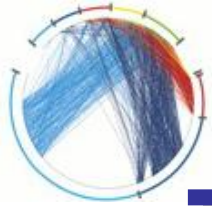
§ perform a query for your own ID

§ refresh all buckets



Kademlia – Lookups

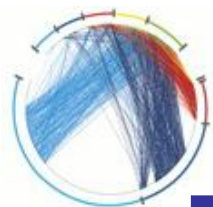
- § **Invariant:** If there exists some node with ID within a specific range then the k-bucket is not empty
 - § if the invariant is true, then the time is logarithmic
 - we move one bit closer each time
- § Due to refreshes the invariant holds with high probability



Kademlia - Properties

- § Easy table maintenance. Tables are updated when lookups are performed
 - § due to XOR symmetry a node receives lookups from the nodes that are in its own table

- § Fast lookup by making parallel searches
 - § at the expense of increased traffic.



e-mule snapshots

eMule v0.47a

Desconectar | Kad | Servidores | Tráfico | Buscar | Compartidos | Mensajes | IRC | Estadísticas | Preferencias | Herramientas... | Ayuda

Lista de servidores (301)

Nombre del servidor	IP	Descripción	Ping	Usuarios	Número máxi...	Archivos	Preferen...	Error	Servi...	Límit...	Límit...	Versión	ID baja
Razorback 2.0	195.245.244.243 : 4661	www.razorback2.com - dual opteron ...	969	1.10 M	1.10 M	146.23 M	Alta	0	No	3.50 K	20.00 K	17.9 ...	274....
DonkeyServer No1	62.241.53.2 : 4242	www.First-Load.de / dual opteron 16...	282	364.00 K	950.00 K	53.90 M	Normal	0	No	20.00 K	20.00 K	17.9	143....
Razorback 2.4	64.34.161.177 : 5661	www.razorback2.com - dual opteron ...	250	564.47 K	1.20 M	44.59 M	Normal	0	No	4.00 K	20.00 K	17.7	225....
Byte Devils	64.34.165.203 : 5306	http://www.bytedevils.com/	937	532.24 K	500.00 K	42.06 M	Normal	0	No	5.00 K	20.00 K	17.7	212....
Razorback 2.3	64.34.162.148 : 5661	www.razorback2.com - dual opteron ...	110	464.62 K	1.20 M	36.71 M	Normal	0	No	4.00 K	20.00 K	17.7	185....
Razorback 2.2	64.34.162.138 : 5661	www.razorback2.com - dual opteron ...	609	364.90 K	1.20 M	28.83 M	Normal	0	No	4.00 K	20.00 K	17.7	145....
Byte Devils	64.34.165.204 : 5306	http://www.bytedevils.com/	297	320.35 K	500.00 K	25.31 M	Normal	0	No	5.00 K	20.00 K	17.7	128....
DonkeyServer No2	62.241.53.16 : 4242	www.First-Load.de	203	106.88 K	400.00 K	17.90 M	Normal	0	No	10.00 K	10.00 K	17.9	49.10 K
BIG BanG 9	80.239.200.108 : 3000	www.BIGBanG.to/bb9	78	140.98 K	500.00 K	12.41 M	Normal	0	No	5.00 K	40.00 K	17.9	84.86 K
DonkeyServer No3	62.241.53.17 : 4242	www.First-Load.de	62	86.34 K	400.00 K	11.96 M	Normal	0	No	10.00 K	10.00 K	17.9	42.69 K
www.razorback2.com/...	195.245.244.244 : 3000	www.first-load.info - dual opteron 2...	390	70.09 K	1.15 M	11.87 M	Normal	0	No	20.00 K	40.00 K	17.9	22.07 K
DonkeyServer No6	62.241.53.15 : 4242	www.First-Load.de	187	59.18 K	220.00 K	10.41 M	Normal	0	No	10.00 K	10.00 K	17.9	29.58 K
Pirate's Lair 5	64.34.176.139 : 5821	Loot, Pillage, and Burn !	406	121.88 K	1.20 M	9.63 M	Normal	0	No	4.00 K	20.00 K	17.7	48.75 K
Sonny Boy 6	64.34.161.180 : 3921	Athlon 2600 1GB	234	121.68 K	800.00 K	9.61 M	Normal	0	No	20.00 K	30.00 K	17.7	48.67 K
Em Server No.1	193.138.230.251 : 4242	Em Server No.1	140	71.73 K	300.00 K	9.30 M	Normal	0	No	6.00 K	10.00 K	17.7	32.78 K
!-= www.FreeSexBay....	83.149.117.56 : 4321	www.FreeSexBay.com - Your Daily P...	125	42.18 K	1.00 M	9.16 M	Normal	0	No	10.00 K	10.00 K	17.4	18.74 K
ChezToff (Serveur Fr)	213.186.60.106 : 4661	http://www.cheztoff.com	204	100.21 K	150.00 K	8.28 M	Normal	0	No	1.00 K	25.00 K	17.9	42.17 K
www.UseNeXT.to	212.112.243.146 : 4661	www.UseNeXT.to	157	59.46 K	400.00 K	8.13 M	Normal	0	No	10.00 K	10.00 K	17.7	30.88 K
Byte Devils	194.213.0.20 : 3306	http://www.bytedevils.com/	219	63.47 K	900.00 K	7.36 M	Normal	0	No	5.00 K	20.00 K	17.9	29.81 K
Byte Devils	64.34.164.183 : 5306	http://www.bytedevils.com/	343	91.36 K	500.00 K	7.22 M	Normal	0	No	5.00 K	20.00 K	17.7	36.55 K
BIG BanG 6	80.239.200.105 : 3000	www.BIGBanG.to/bb6	93	42.63 K	300.00 K	7.22 M	Normal	0	No	5.00 K	20.00 K	17.9	19.83 K
Sonny Boy 7	64.34.161.240 : 5221	Athlon 2600 1GB	250	90.97 K	800.00 K	7.19 M	Normal	0	No	20.00 K	30.00 K	17.7	36.39 K
BIG BanG 11	80.239.200.110 : 3000	www.BIGBanG.to/bb11	94	70.00 K	300.00 K	6.94 M	Normal	0	No	1.00 K	30.00 K	17.9	32.44 K
www.UseNeXT.info	80.190.240.125 : 4661	www.UseNeXT.info	218	77.52 K	400.00 K	6.59 M	Normal	0	No	4.00 K	4.00 K	17.7	36.54 K
L337 Kr3w 1	66.172.60.141 : 4661	www.netstats.cs_source.ru	421	81.30 K	100.00 K	6.55 M	Normal	1	No	2.00 K	5.00 K	17.4	13.01 K
<<< Saugcenter 1 >>>	80.190.233.144 : 6565	http://www.saugcenter.net	125	56.55 K	160.00 K	6.39 M	Normal	0	No	2.50 K	2.50 K	17.9 ...	21.26 K
! !****ifreesex.net >>...	194.30.160.81 : 4661	www.ifreesex.net	140	19.55 K	100.00 K	6.39 M	Normal	0	No	10.00 K	10.00 K	16.46	8.35 K
DonkeyServer No5	62.241.53.4 : 4242	www.First-Load.de	140	31.13 K	240.00 K	6.02 M	Normal	0	No	10.00 K	10.00 K	17.8	14.32 K
Asyrix 2.1	80.190.251.50 : 4321	www.nice-xxx.com	125	47.42 K	350.00 K	5.84 M	Normal	0	No	4.00 K	4.00 K	17.7	14.95 K
Sonny Boy 8	64.34.161.241 : 4761	Athlon 2600 1GB	281	71.86 K	800.00 K	5.69 M	Normal	0	No	20.00 K	30.00 K	17.7	28.75 K
Breizh Digitalus	213.186.47.84 : 4661	http://www.nana-sexy.com	141	53.81 K	155.00 K	5.49 M	Normal	0	No	2.00 K	25.00 K	17.9	26.00 K
!! Saugstube !!	193.138.221.214 : 4242	www.saugstube.de	281	54.18 K	200.00 K	5.44 M	Normal	0	No	2.00 K	10.00 K	17.7	25.66 K
!-= www.FreeSexBay....	83.149.101.91 : 4321	www.FreeSexBay.com - Your Daily P...	141	28.82 K	1.00 M	5.38 M	Normal	0	No	10.00 K	10.00 K	17.5	15.30 K

Información de servidores | Registro | Limpiar

100 cd 52x for only 13.29 !!
<http://dvd.razorback2.com/>

— SEX SEX - ADULT + 18 I - <http://www.click-for-sex.biz> —
Very hot girls getting ultra nasty by swapping and swallowing big loads of cum.
Downloadable high quality movies, click here now:
-> <http://www.click-for-sex.biz> <-

Desconectar

Nuevo servidor

IP o dirección: Puerto: : 4661

Nombre:

Añadir a la lista

Actualizar server.met desde URL

Actualizar

Información propia

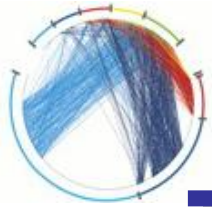
eD2K Red
Estado: Conectado
IP: Puerto: 80.221.16.4:4427
ID: 68214096
ID alta

eD2K Servidor
Nombre: Razorback 2.0
Descripción: www.razorback2.com - du...
IP: Puerto: 195.245.244.243:4661
Versión: 17.9 (lugdunum)
Usuarios: 1,097,804
Archivos: 146,227,576

Kad Red
Estado: Conectado

Servidor Web
Estado: Desactivado

Conexión establecida con: Razorback 2.0 | Usuarios: 11.6 M(374) | Archivos: 844.2 M(51) | Subida: 1.6 | Descarga: 0.0 | eD2K: Conectado | Kad: Conectado

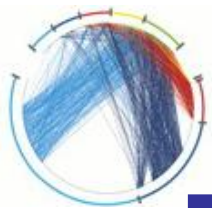


e-mule snapshots

The screenshot displays the eMule v0.47a interface. The top menu bar includes icons for Desconectar, Kad, Servidores, Tráfico, Buscar, Compartidos, Mensajes, IRC, Estadísticas, Preferencias, Herramientas, and Ayuda. The main window is titled 'Contactos (218)' and shows a list of contacts with columns for ID, Tipo, and Distancia. Below this, there is a 'Búsquedas actuales (4)' section with a table of active searches.

ID	Tipo	Distancia
FCEDD43D68B3434FD1219613DC4A9E8C	3(0)	0000101001010101100110011111010001010000111000110010001000001100001100011001010100001000110101100100110111011100110010101
FCE328228E259DAD6F4455929A8899DA	3(0)	000010100101010110110010111101011011110111011100001010011000011011001000010110110111101000100001111110101011111011000011
FCC57CC1CAAB1934967AC06C34023B29	3(0)	0000101001111101001100010000100000010010100000100100111110000100011000101001111110100101100101000101111110110000110000
FC9E058F30C3038AFF9FC8A14F093B80	3(0)	000010100010011001001000010001101111001100001000100001000011011000100010100110011100101001000111001010011010010110010011001
FC7B0ACD5386A48DE4A0A557E785AF9	3(0)	0000101011000011010001110000010010010000111000001011110100001001110011110011100111101100111111011000010110111011100000
FC27206EC969C10C686ADBAC5A5CD...	3(0)	0000101010011111011011011010011100001010101000100100101010100000010110101001110011100001100111001101111001000010011001100010101
FC20F1EE508BC7265DA658588FD3D686	3(0)	000010101001100010111000010011110010011011100000100110011101010100000001110101010001001101000000010101011100011000110011111
FC20E192341AECC1920578EB16EA7681	3(0)	0000101010011000101011000101101111101111010001011001010000110101001110101010100010011011011100100110010111001000110011000
FC02C1A51F92847E70E417C1339453E0	3(0)	00001010101110101000110001101100101110001011001000011111010010101010101100101101111000101101101110100110101001111001
FC00D2B22A9A1DA4EA7803B683237B9A	3(0)	0000101011100010011111011110111101001010000110010110011010000011011100101010011100110000110000011001011101001110010000011
FBE49A53C69DB858B67324C2097F6FD0	3(0)	00001101010111001101011110011010000010101010100011000010010111010101010010000000111011100101000110000000101000100011001001
FB943FD2003608E4089CC73731AD0D7C	3(0)	000010100101100011100100001101111000011111110110000011001010001101010111001111111101000001110110100110100001110101001100101
FB64F71951F390A92290318BDFE886E7	3(0)	0000110111011001011101011010000100100100011100000010110110010111111111000011000010111011010101010100101010000111111110
FB046C698853B4D3C08374FF651E0407	2(0)	000011011011110000100001101000000100100010010000011111100011110001110111010000010011101001111110000001100011110001100011110
FAD4850F2EF5F1E8A1F48CB6E7C9E9DE	3(0)	0000110001101100110010001100011011101100101111001110100010010001111001010011110110110100001100110001010110100000011101000111
FACB8465A96D8315C24B7504CA7B7AA2	3(0)	00001100011100111100100110101000110101010100110000010001101100100011110001100001001111001101000100111100001101001110110111011
FAS5FB3040CC033E5EE6F3C7BC3809FC7	3(0)	0000110011100111111111011001101110001111000010111011100000101001001100110011110000000110010010110100011011111010111100001101110
FA22B6492913A1EB95D19271C3B1BF74	3(0)	00001100100110101111011100000001101010101000001010100010011101001000100000101010100001000001010001101001100010110000110101
FA0B186A121C0BFDCCA349A6FE85F034	3(0)	00001100101100110101010101000111010111100000000100010001000111100000110011100101001110111111000000101110010101
FA0736B89A68ABD7D2D4CBE02C1D177	3(0)	00001100101111110111010111000101010100101010100000001011000110100000011111001110110100011010000111011110000110100110110
F57CFB8A6A381B7ED7D50264097240AE	3(0)	00000011110001001010100110001010100111100111001000010100100001010100001100011100001010100100011000000111101001111010111
F57BCB5C96880B5DC6245AEF17718F0F	3(0)	0000001111000011100001101001010101010101010000111000000010010001000110110111011101100000110111110010010000011000110100000010110
F56A0D5199618B7C2525094C7A83A173	3(0)	00000011110100100100000010011000010110101010101000000001010000111100001110100011001101111001111111111111100100011001101010
FC50F02445F4513D000C4469232021	3(0)	000000111101000010010001100100011001000110010001100100011001000110010001100100011001000110010001100100011001000110010001

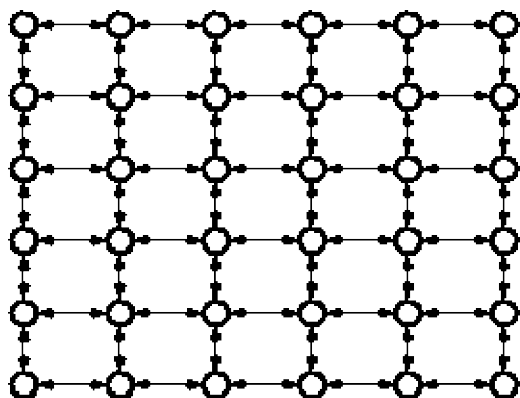
N...	Clave	Tipo	Nombre	Estado	Carga	Paquetes envia...	Respuestas
1	442A...	Buscar Fuentes		Finalizando	0 (0/0)	19/8	0
-1	F462...	Búsqueda de N...		Finalizando	0 (0/0)	4/0	1
-1	FC5A...	Búsqueda de N...		Finalizando	0 (0/0)	4/0	1
-1	F1A6...	Búsqueda de N...		Activo	0 (0/0)	1/0	0



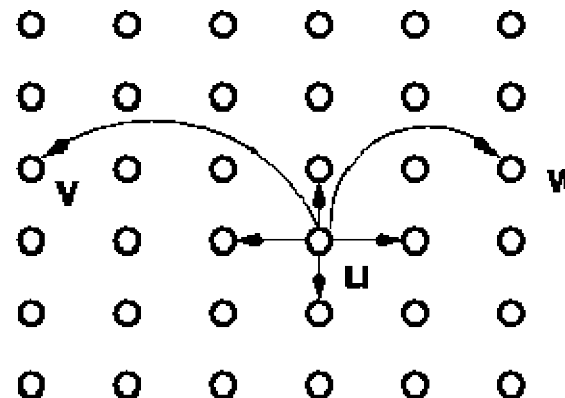
Kleinberg's small world

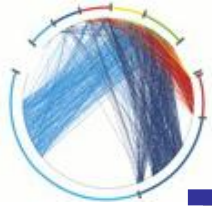
- § Consider a 2-dimensional grid
- § For each node u add edge (u,v) to a vertex v selected with pb proportional to $[d(u,v)]^{-r}$
- § Simple Greedy routing
 - § If $r=2$, expected lookup time is $O(\log^2 n)$
 - § If $r \neq 2$, expected lookup time is $\Omega(n^\epsilon)$, ϵ depends on r
- § The theorem generalizes in d -dimensions for $r=d$

A)



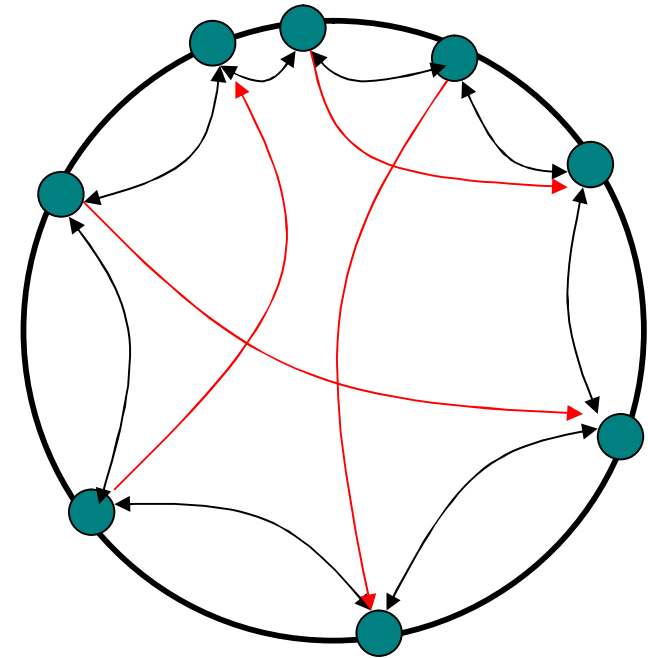
B)





Symphony

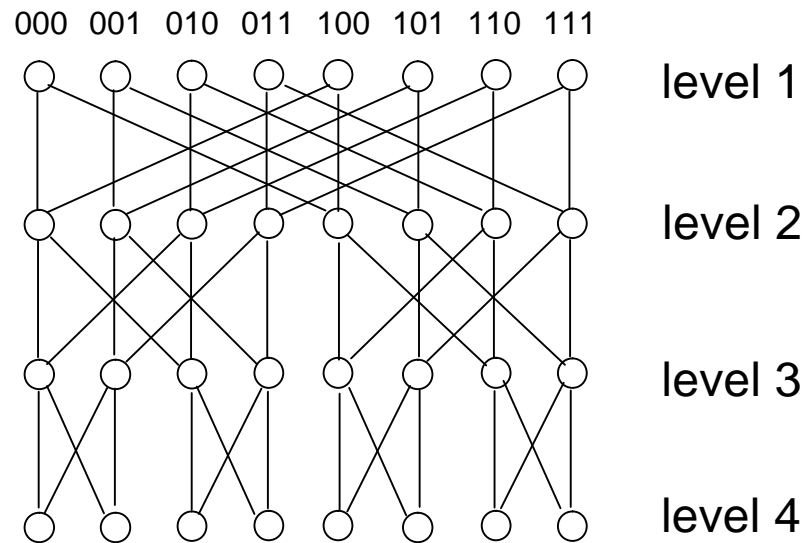
- § Map the nodes and keys to the ring
- § Link every node with its successor and predecessor
- § Add k random links with probability proportional to $1/(d \log n)$, where d is the distance on the ring
- § Lookup time $O(\log^2 n)$
- § If $k = \log n$ lookup time $O(\log n)$
- § Easy to insert and remove nodes (perform periodical refreshes for the links)

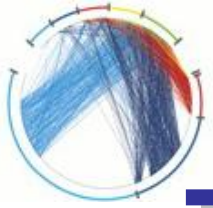




Viceroy

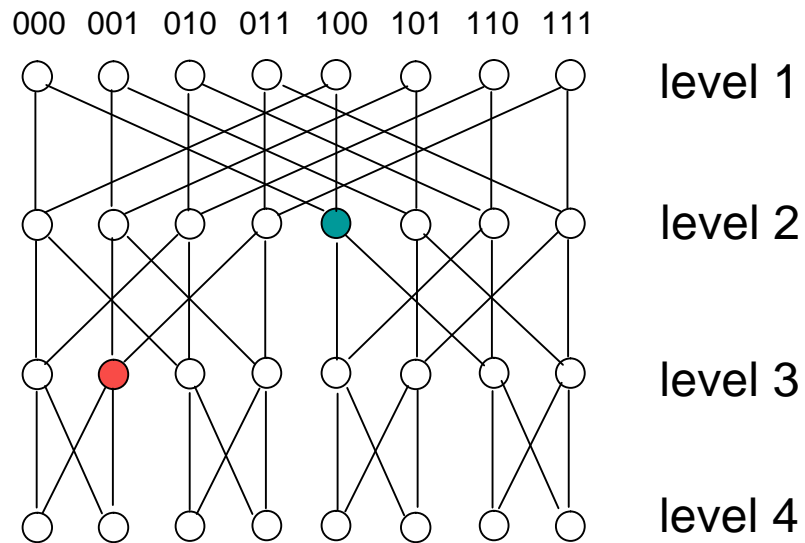
§ Emulating the butterfly network

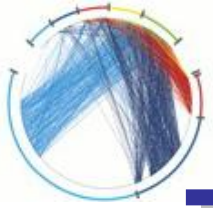




Viceroy

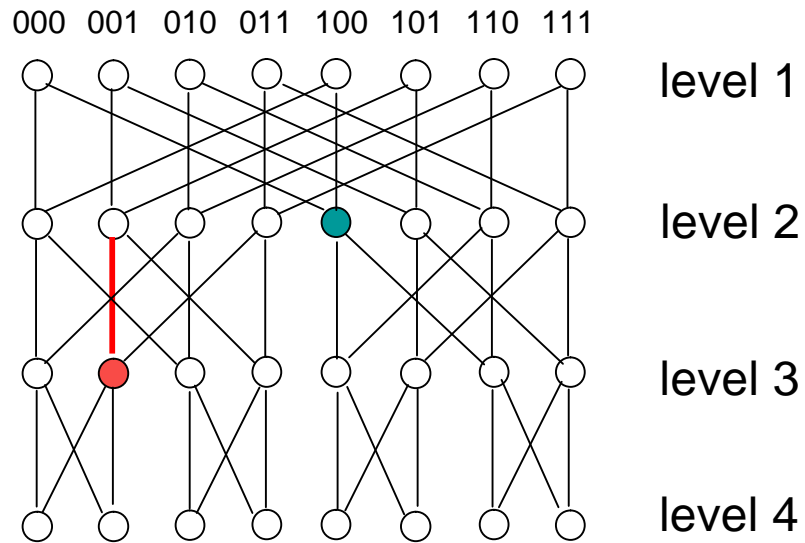
§ Emulating the butterfly network





Viceroy

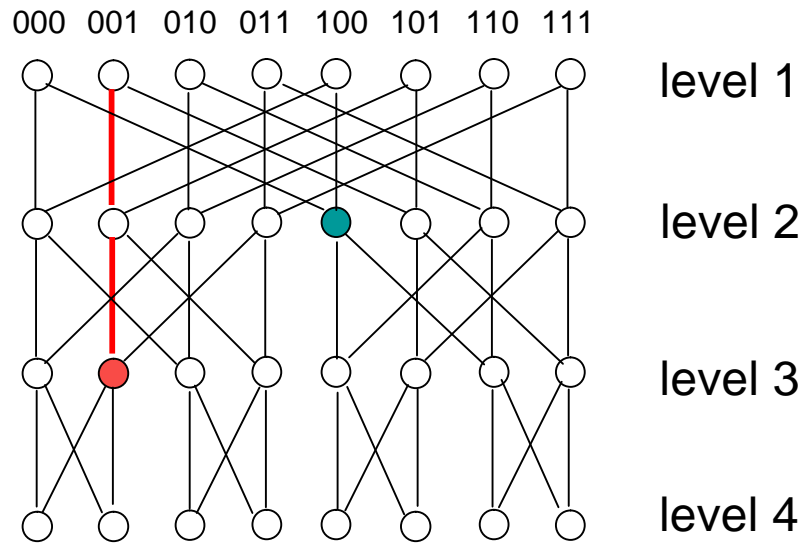
§ Emulating the butterfly network

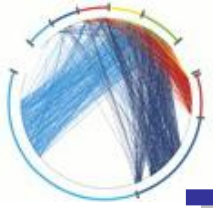




Viceroy

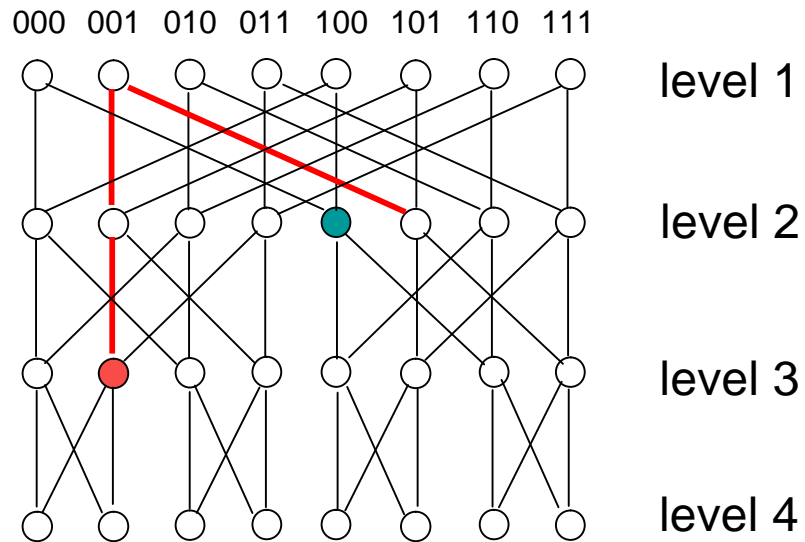
§ Emulating the butterfly network

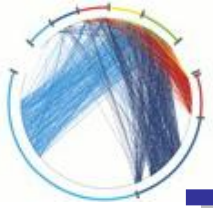




Viceroy

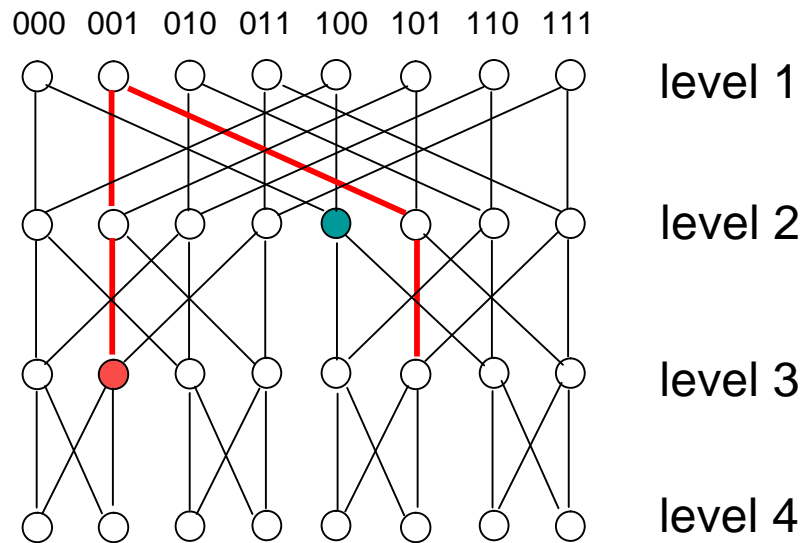
§ Emulating the butterfly network





Viceroy

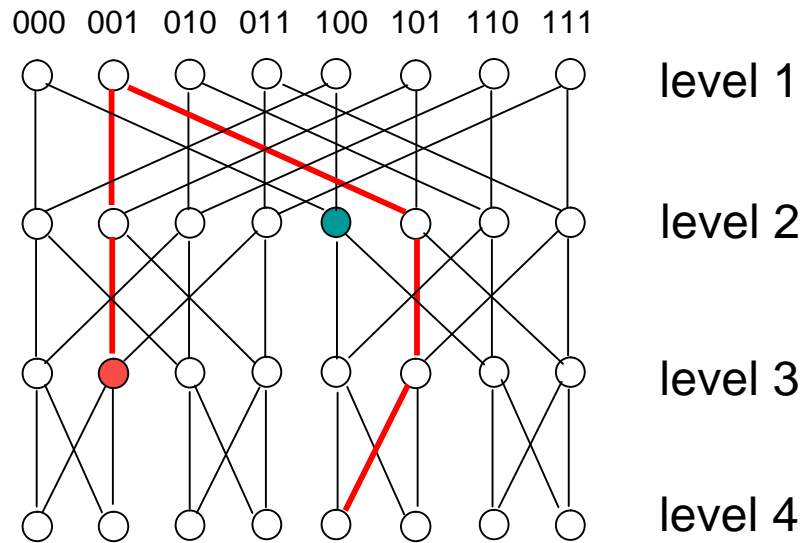
§ Emulating the butterfly network

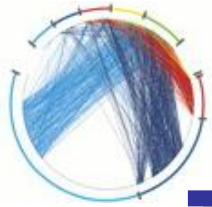




Viceroy

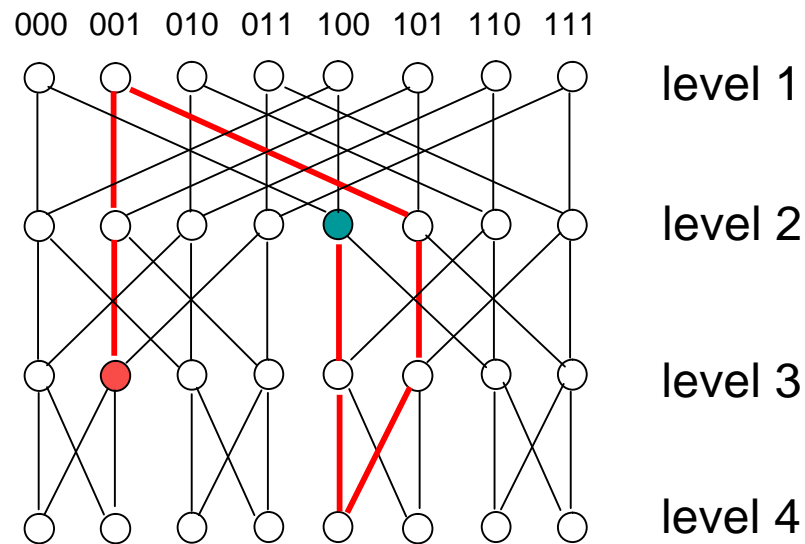
§ Emulating the butterfly network





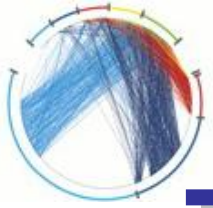
Viceroy

§ Emulating the butterfly network



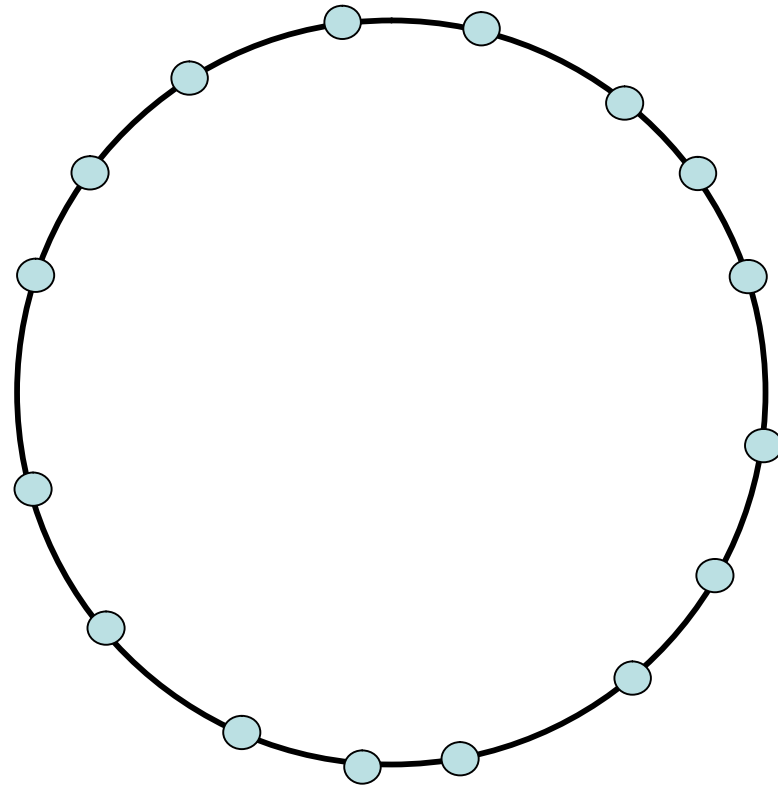
§ Logarithmic path lengths between any two nodes in the network

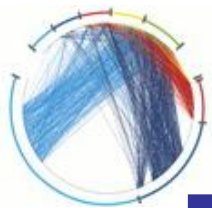
§ Constant degree per node



Viceroy network

§ Arrange nodes and keys on a ring, like in Chord.



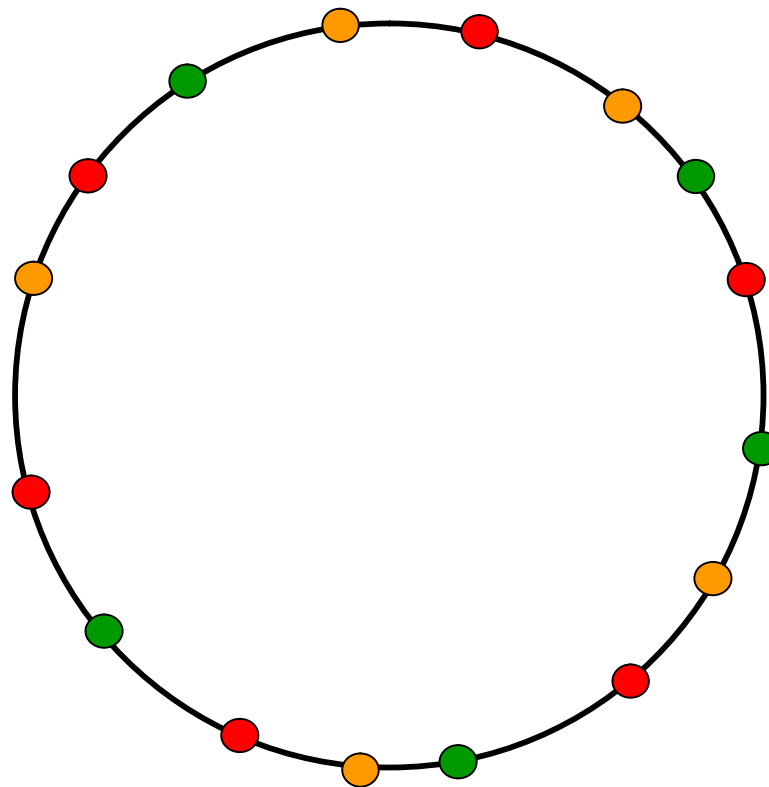


Viceroy network

§ Assign to each node a level value, chosen uniformly from the set $\{1, \dots, \log n\}$

§ estimate n by taking the inverse of the distance of the node with its successor

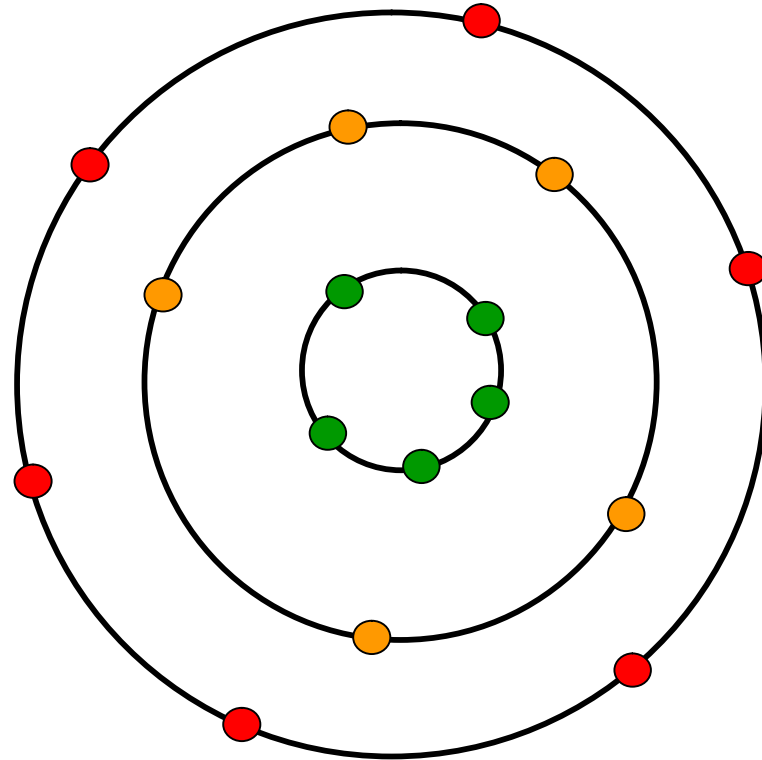
§ easy to update

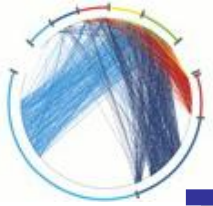




Viceroy network

§ Create a ring of nodes within the same level





Butterfly links

- § Each node x at level i has two **downward** links to level $i+1$
 - § a **left link** to the first node of level $i+1$ after position x on the ring
 - § a **right link** to the first node of level $i+1$ after position $x + (\frac{1}{2})^i$

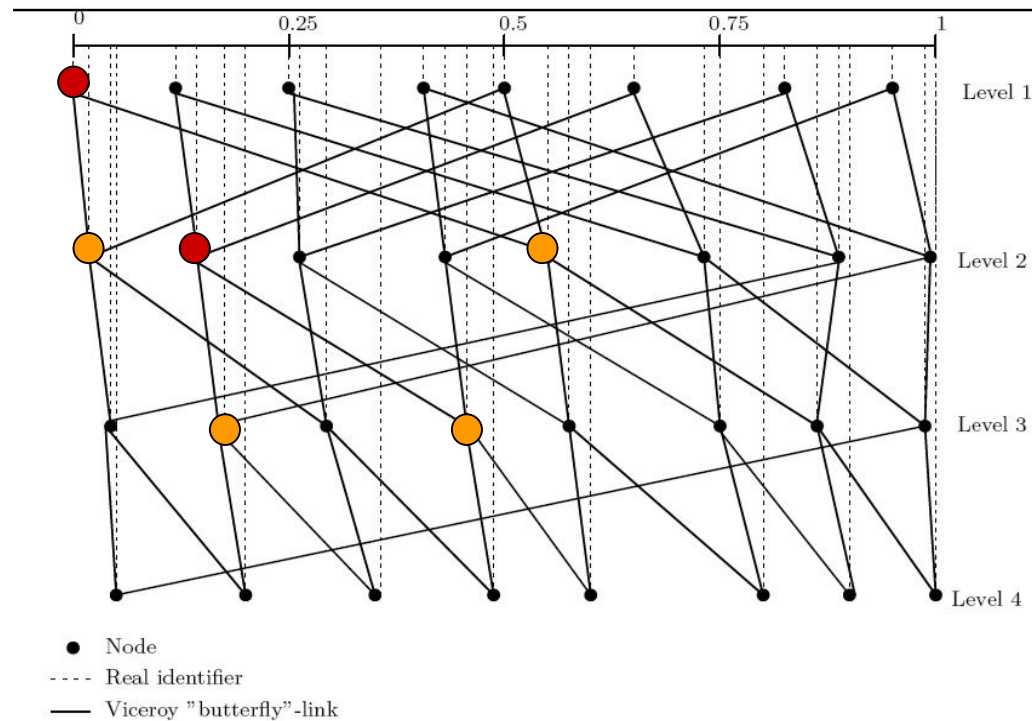
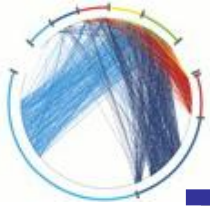
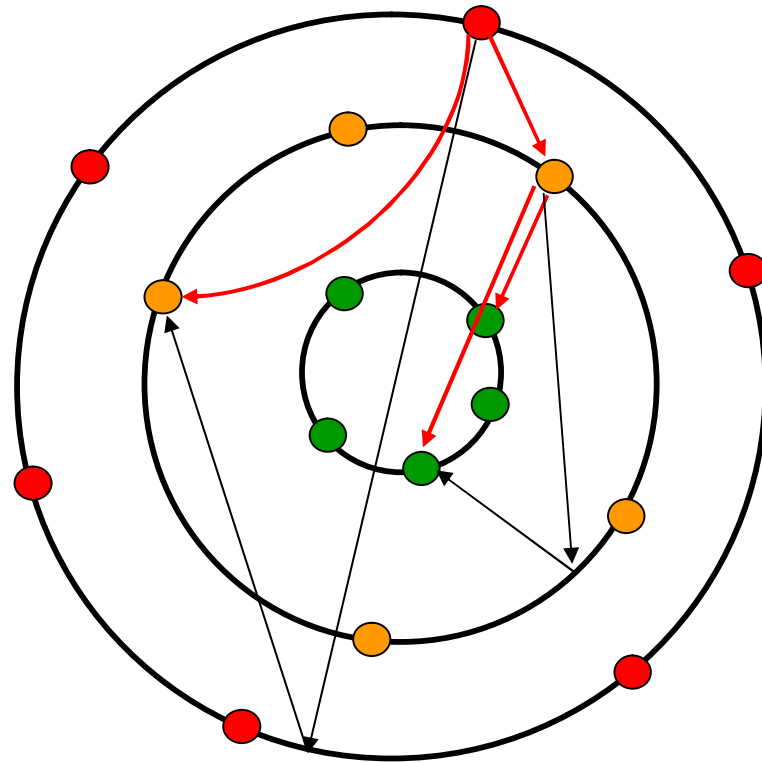
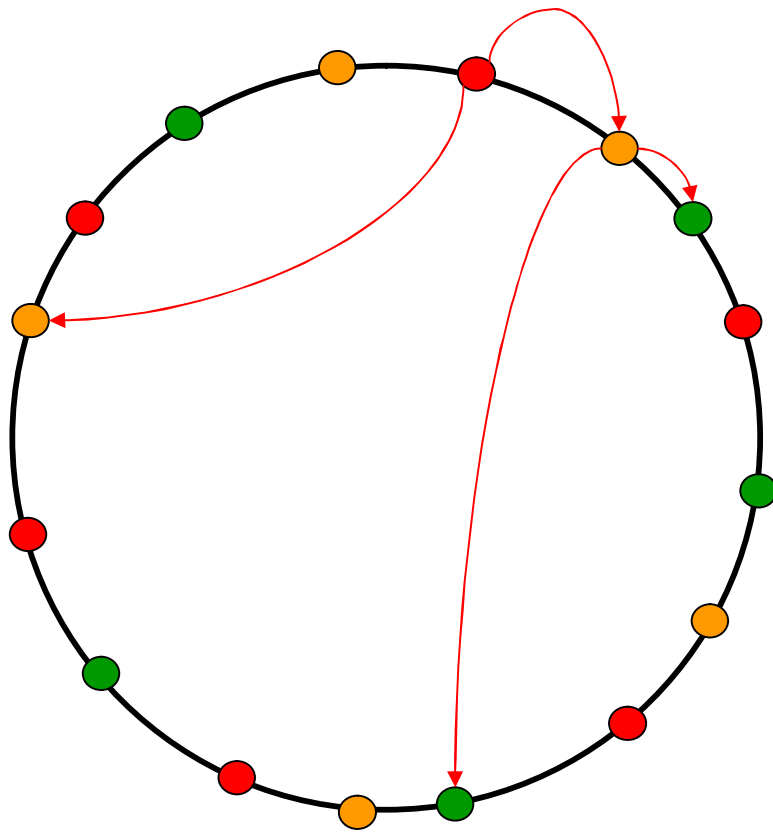


Figure 1: An ideal Viceroy network. Up and ring links are omitted for simplicity.



Downward links



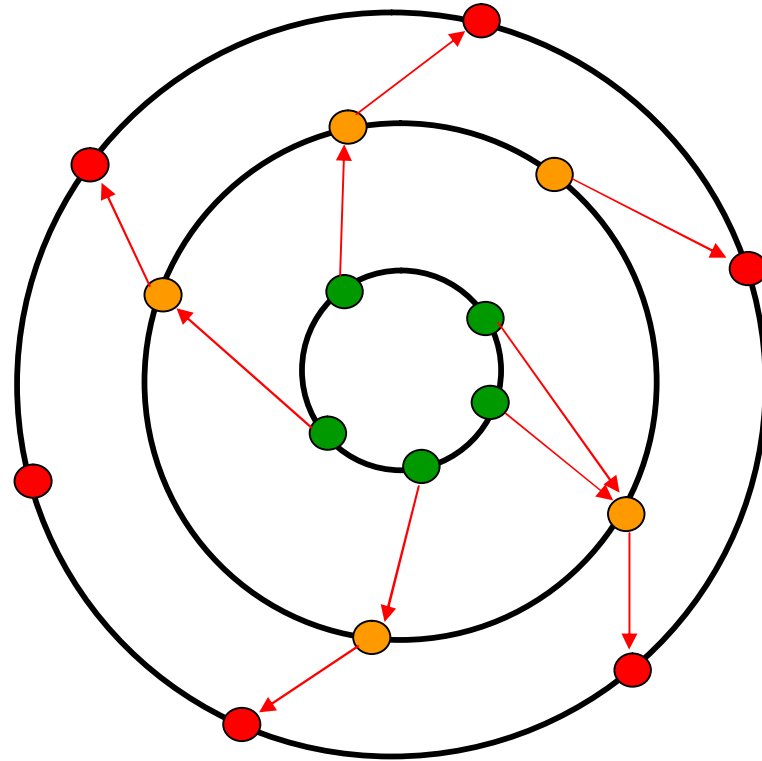
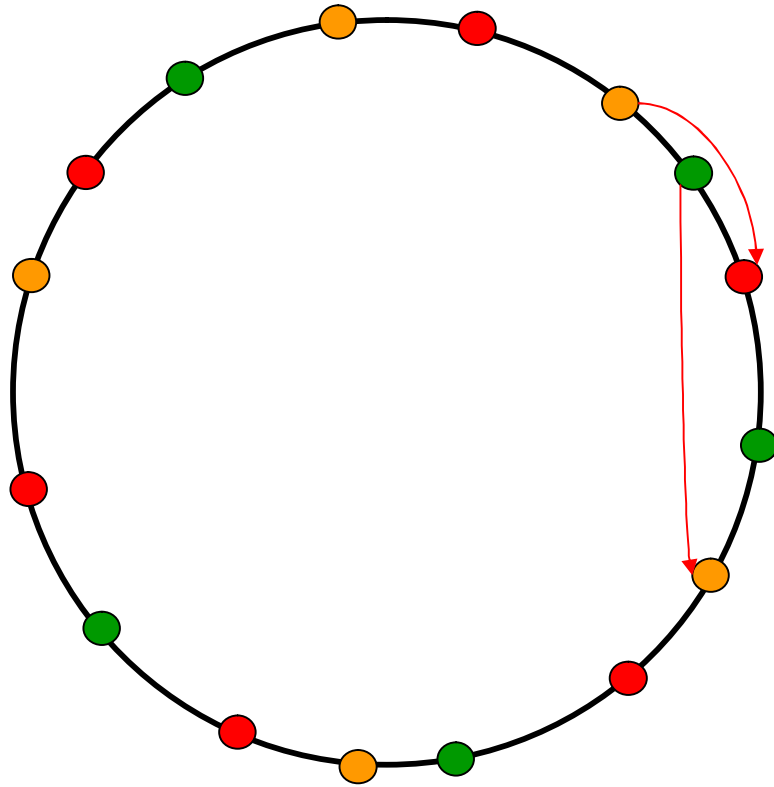


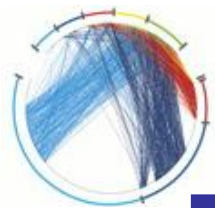
Upward links

§ Each node x at level i has an **upward** link to the next node on the ring at level $i-1$



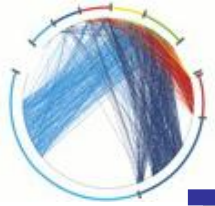
Upward links





Lookup

- § Lookup is performed in a similar fashion like the butterfly
 - § expected time $O(\log n)$
- § Viceroy was the first network with constant number of links and logarithmic lookup time



P2P Review

§ Two key functions of P2P systems

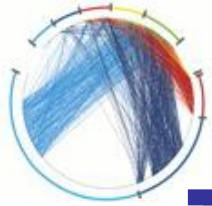
- § Sharing content
- § Finding content

§ Sharing content

- § Direct transfer between peers
 - All systems do this
- § Structured vs. unstructured placement of data
- § Automatic replication of data

§ Finding content

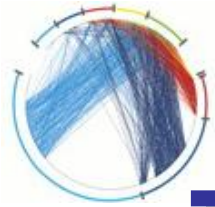
- § Centralized (Napster)
- § Decentralized (Gnutella)
- § Probabilistic guarantees (DHTs)



Distributed Hash Tables

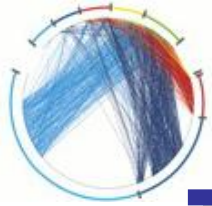
- § Consistent Hashing: map both keys and nodes to the same space
 - § guarantees good load balancing properties

- § Routing (overlay) networks
 - § degree d per node (usually $O(\log n)$)
 - § number of hops $O(\log n / \log d)$ (or $O(\log n)$)
 - § can it be made to be fault tolerant?



Acknowledgements

§ Thanks to Vinod Muthusamy, George Giakkoupis, Jim Kurose, Brian, Levine, Don Towsley



References

- § G. Giakkoupis, [Routing algorithms for Distributed Hash Tables](#), Technical Report, University of Toronto, 2003
- § Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong, "[Freenet: A Distributed Anonymous Information Storage and Retrieval System](#)," in Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009
- § S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker. [A Scalable Content-Addressable Network](#). ACM SIGCOMM, 2001
- § I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan. [Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications](#). ACM SIGCOMM, 2001.
- § A. Rowstron, P. Druschel. [Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems](#). 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001).
- § P. Maymounkov and D. Mazieres. [Kademlia: A peer-to-peer information system based on the XOR metric](#). In Proceedings of IPTPS02,
- § B. Y. Zhao, J. D. Kubiatowicz, A. D. Joseph, [Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing](#). UC Berkeley Computer Science Division, Report No. UCB/CSD 01/1141, April 2001.