An Introduction to Peer-to-Peer Networks

Presentation for
MIE456 - Information Systems Infrastructure II

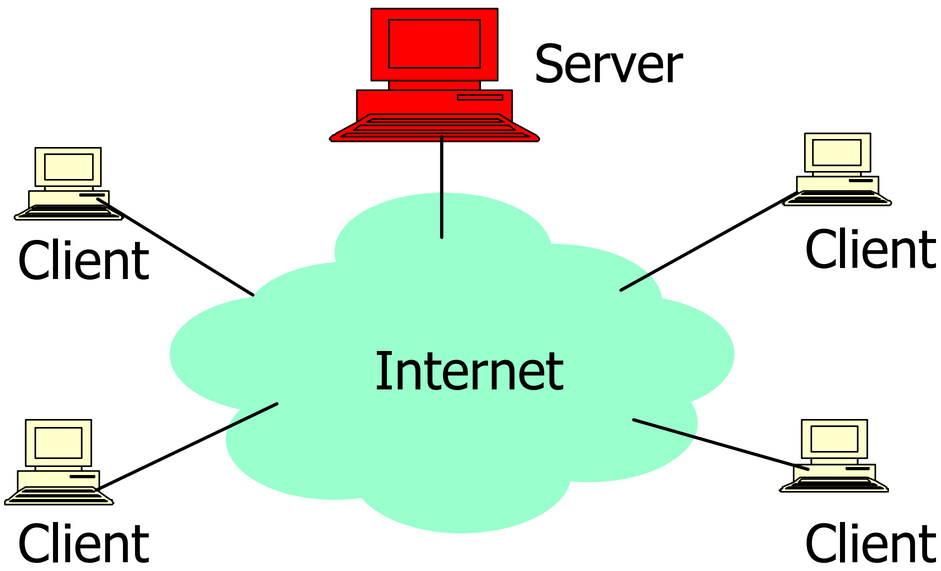
Vinod Muthusamy
October 30, 2003

Agenda

- Overview of P2P
  - Characteristics
  - Benefits
- Unstructured P2P systems
  - Napster (Centralized)
  - Gnutella (Distributed)
  - Kazaa/Fasttrack (Super-peers)
- Structured P2P systems (DHTs)
  - Chord
  - Pastry
  - CAN
- Conclusions

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

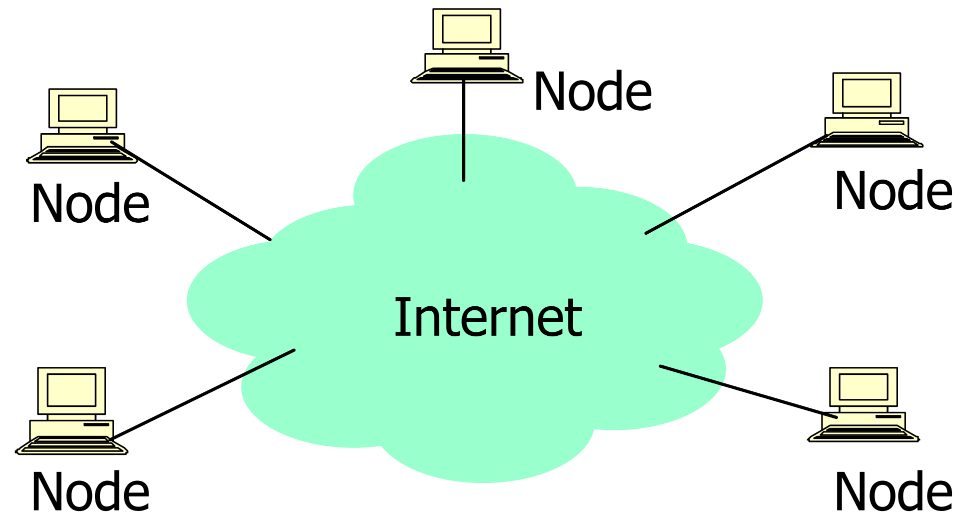
P2P Computing*

- P2P computing is the sharing of computer resources and services by direct exchange between systems.
- These resources and services include the exchange of information, processing cycles, cache storage, and disk storage for files.
- P2P computing takes advantage of existing computing power, computer storage and networking connectivity, allowing users to leverage their collective power to the 'benefit' of all.

* From http://www-sop.inria.fr/mistral/personnel/Robin.Groenevelt/Publications/Peer-to-Peer_Introduction_Feb.ppt

P2P Architecture

- All nodes are both clients and servers
  - Provide and consume data
  - Any node can initiate a connection
- No centralized data source
  - "The ultimate form of democracy on the Internet"
  - "The ultimate threat to copy-right protection on the Internet"



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

P2P Network Characteristics

- 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

P2P Benefits

- Efficient use of resources
  - Unused bandwidth, storage, processing power at the edge of the network

- Scalability
  - Consumers of resources also donate resources
  - Aggregate resources grow naturally with utilization

- Reliability
  - Replicas
  - Geographic distribution
  - No single point of failure

- Ease of administration
  - Nodes self organize
  - No need to deploy servers to satisfy demand (c.f. scalability)
  - Built-in fault tolerance, replication, and load balancing

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

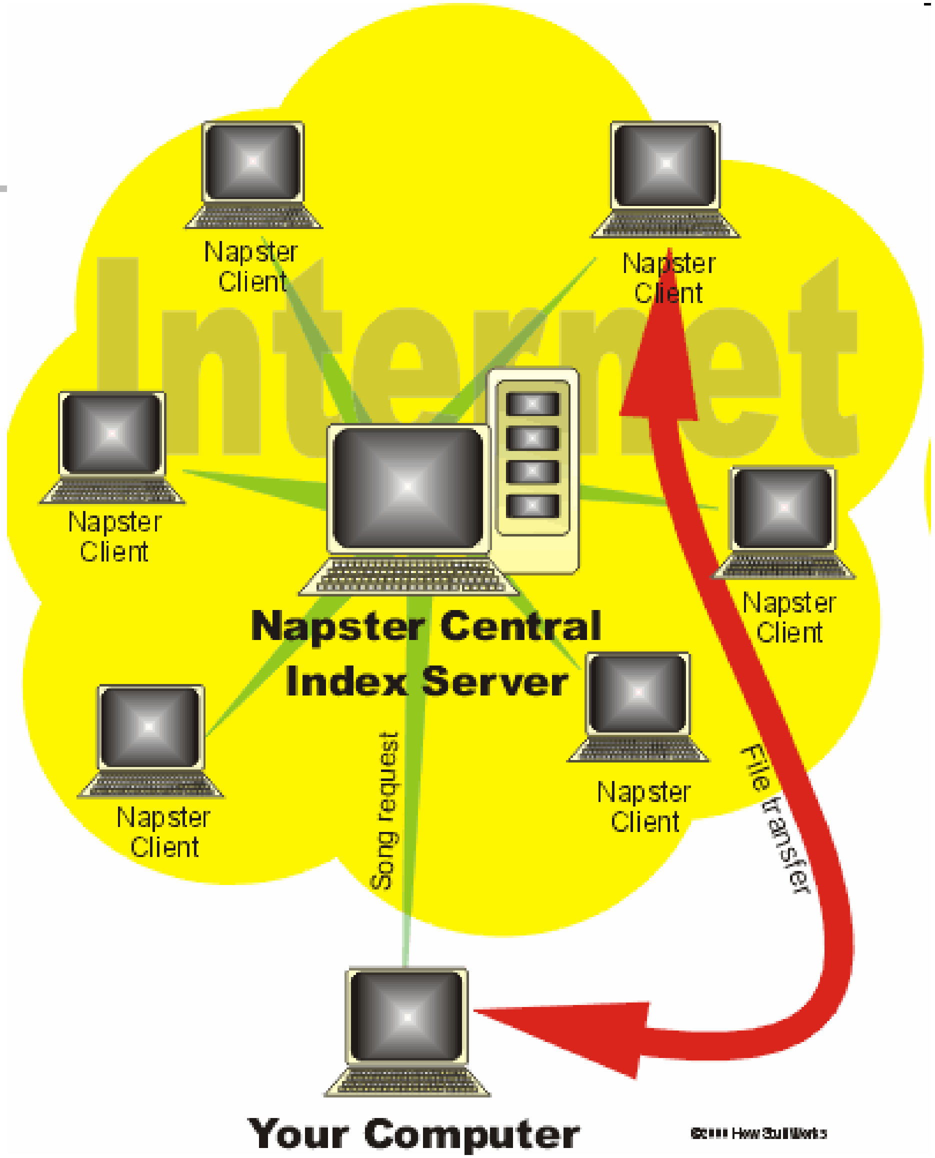
Popular P2P Systems

- 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

- A way to share music files with others

- Users upload their list of files to Napster server
- You send queries to Napster server for files of interest
  - Keyword search (artist, song, album, bitrate, etc.)
- Napster server replies with IP address of users with matching files
- You connect directly to user A to download file



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

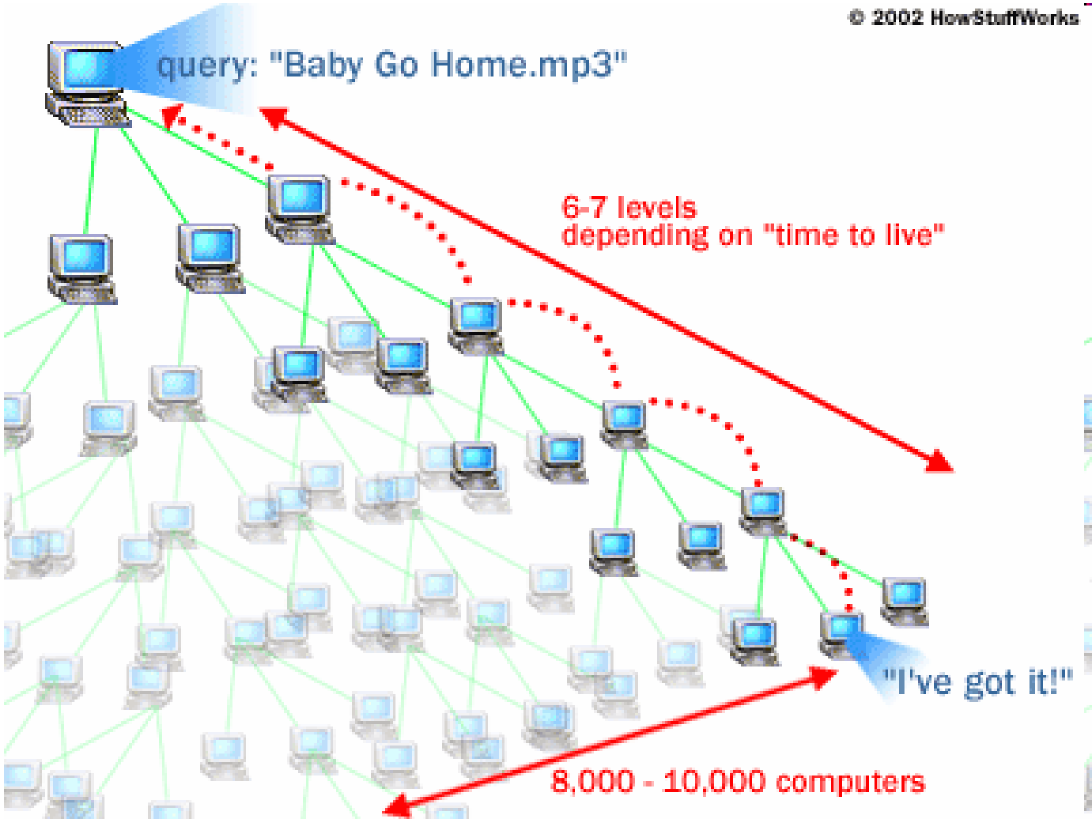
Napster

- Central Napster server
  - Can ensure correct results
  - Bottleneck for scalability
  - Single point of failure
  - Susceptible to denial of service
    - Malicious users
    - Lawsuits, legislation

- Search is centralized
- File transfer is direct (peer-to-peer)

Gnutella

- Share any type of files (not just music)
- Decentralized search unlike Napster
- You ask your neighbours for files of interest
- Neighbours ask their neighbours, and so on
  - TTL field quenches messages after a number of hops
- Users with matching files reply to you



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

Gnutella

- Decentralized
  - No single point of failure
  - Not as susceptible to denial of service
  - Cannot ensure correct results
- Flooding queries
  - Search is now distributed but still not scalable

Kazaa (Fasttrack network)

- Hybrid of centralized Napster and decentralized Gnutella

- 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

Free riding*

- File sharing networks rely on users sharing data
- Two types of free riding
  - Downloading but not sharing any data
  - Not sharing any interesting data
- On Gnutella
  - 15% of users contribute 94% of content
  - 63% of users never responded to a query
    - Didn’t have “interesting” data

* Data from E. Adar and B.A. Huberman (2000), “Free Riding on Gnutella”

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

- 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
- "Smart" queries
  - Requests get routed to correct peer by incremental discovery

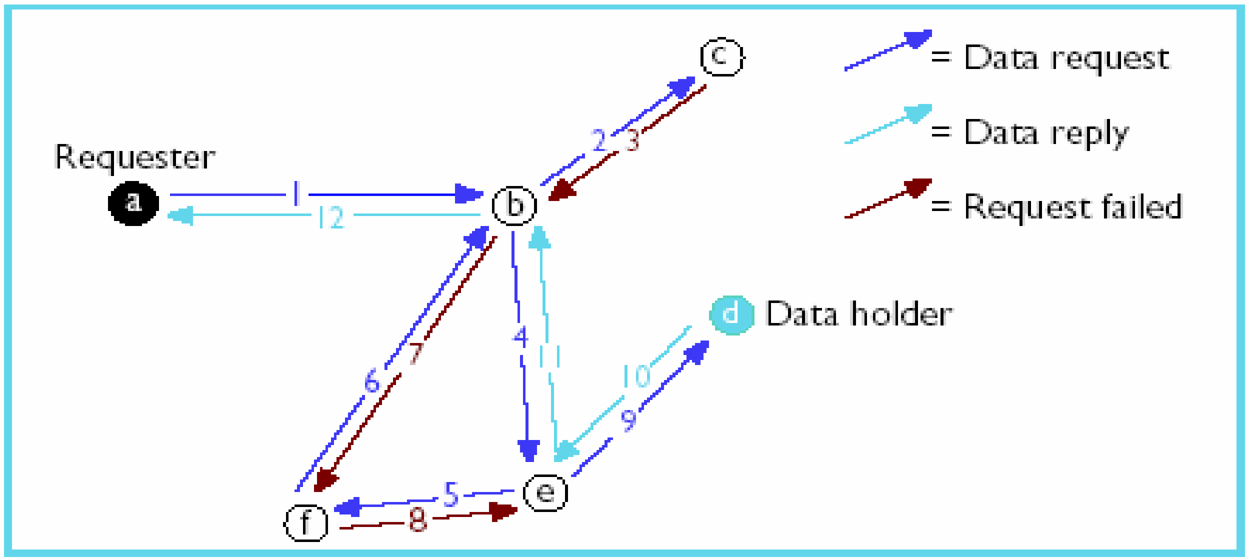


Figure 1. Typical request sequence. The request moves through the network from node to node, backing out of a dead-end (step 3) and a loop (step 7) before locating the desired file.

Structured P2P

- Second generation P2P overlay networks
- Self-organizing
- Load balanced
- Fault-tolerant
- Scalable guarantees on numbers of hops to answer a query
  - Major difference with unstructured P2P systems
- Based on a distributed hash table interface

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

- 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

DHT Generic Interface

- Node id: m-bit identifier (similar to an IP address)
- Key: sequence of bytes
- Value: sequence of bytes

- put(key, value)
  - Store (key,value) at the node responsible for the key
- value = get(key)
  - Retrieve value associated with key (from the appropriate node)

DHT Applications

- Many services can be built on top of a DHT interface
  - File sharing
  - Archival storage
  - Databases
  - Naming, service discovery
  - Chat service
  - Rendezvous-based communication
  - Publish/Subscribe

DHT Desirable Properties

- Keys mapped evenly to all nodes in the network
- Each node maintains information about only a few other nodes
- Messages can be routed to a node efficiently
- Node arrival/departures 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]
  - SkipNet [Microsoft Research US, Univ. of Washington]
  - Kademlia [New York University]
  - Viceroy [Israel, UC Berkeley]
  - P-Grid [EPFL Switzerland]
  - Freenet [Ian Clarke]
- These systems are often referred to as P2P routing substrates or P2P overlay networks

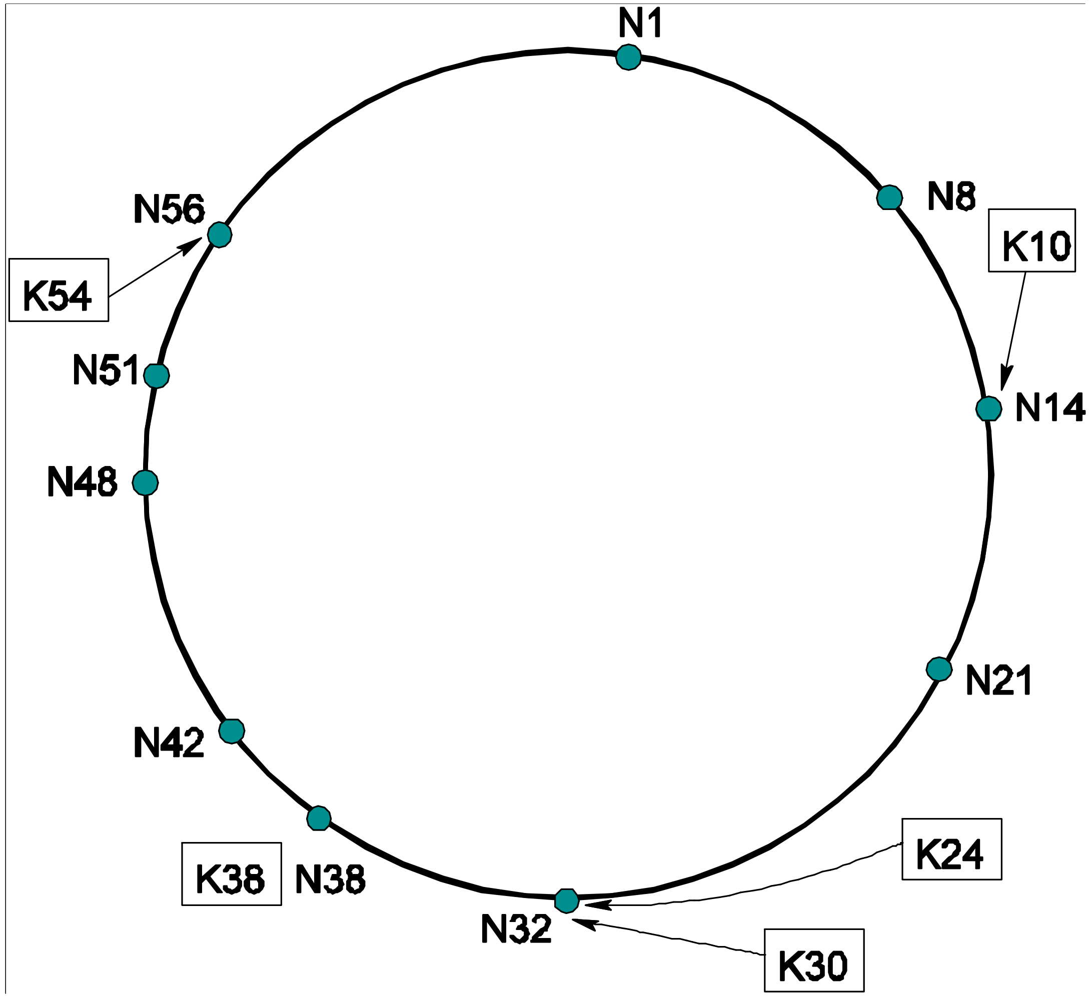
Chord API

- Node id: unique m-bit identifier
  (hash of IP address or other unique ID)
- Key: m-bit identifier (hash of a sequence of bytes)
- Value: sequence of bytes

- API
  - insert(key, value) → store key/value at r nodes
  - lookup(key)
  - update(key, newval)
  - join(n)
  - leave()

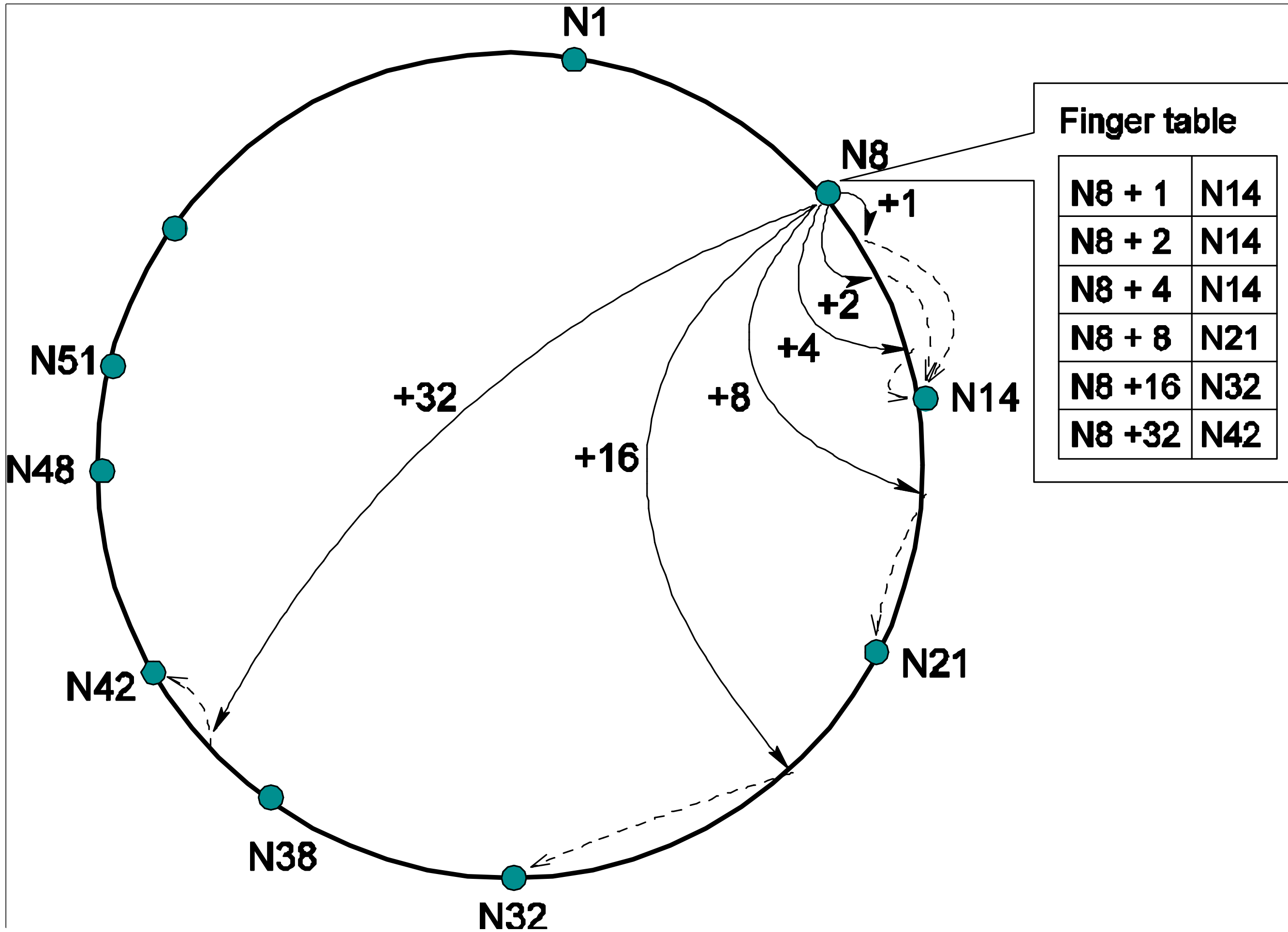
Chord Identifier Circle

- 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



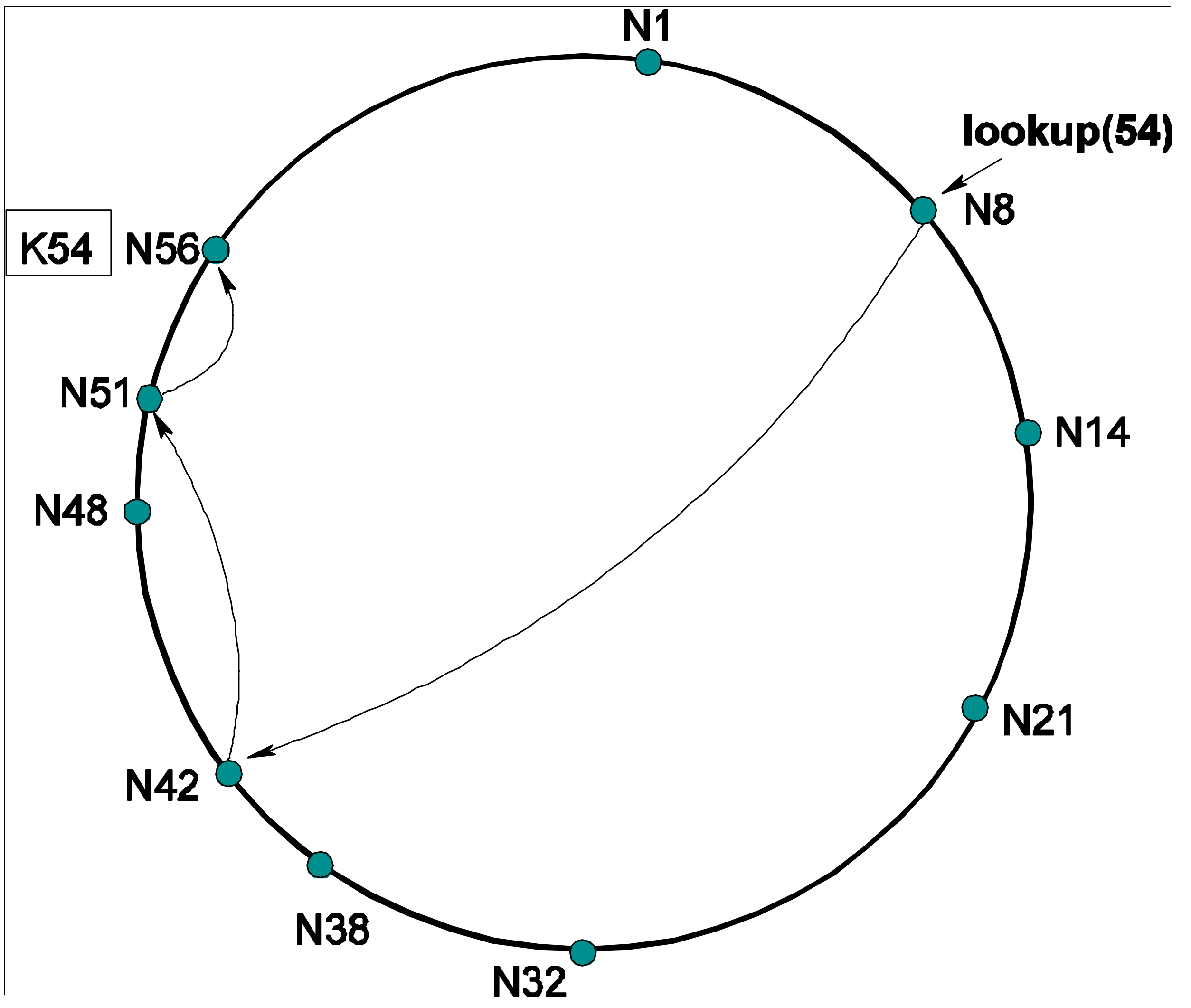
Chord Finger Table

- O(logN) table size
- i^{th} finger points to first node that succeeds n by at least 2^{i-1}



Chord Key Location

- Lookup in finger table the furthest node that precedes key
- Query homes in on target in O(logN) hops



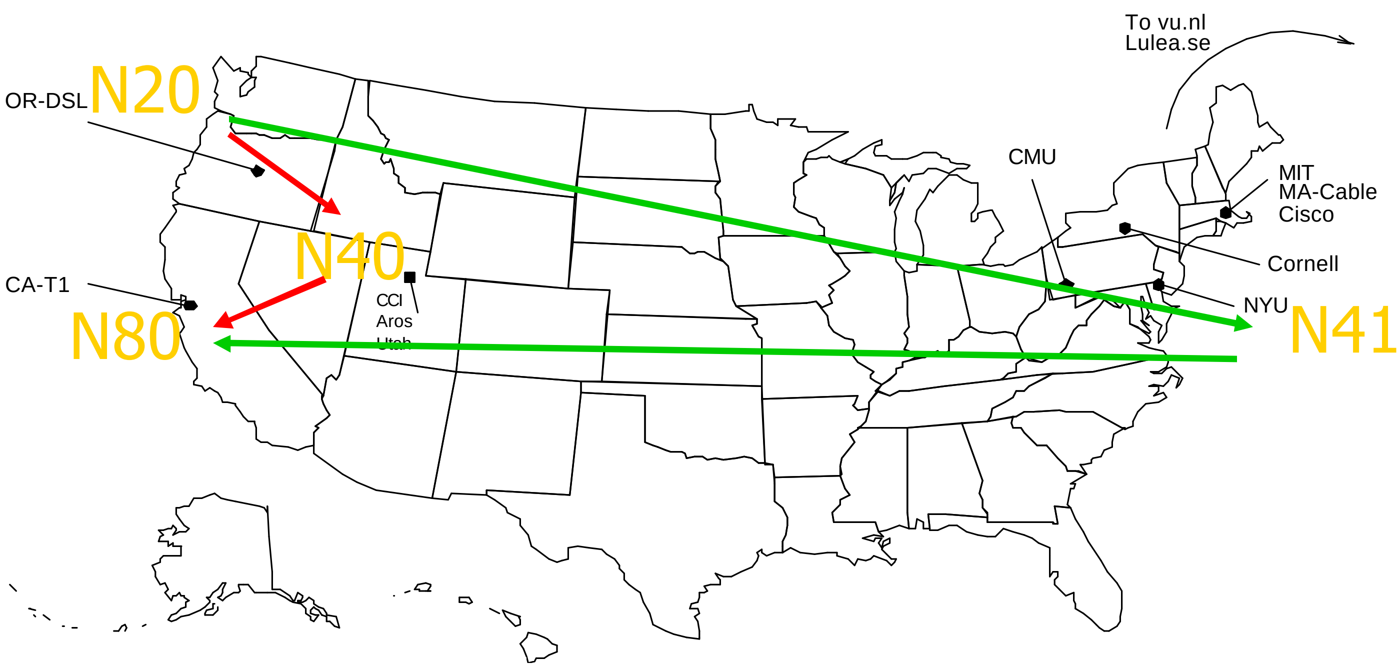
Chord Properties

- 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(logN) other nodes
  - lookups resolved with O(logN) hops

- No delivery guarantees
- 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

Pastry

- Similar interface to Chord
- Considers network locality to minimize hops messages travel
- New node needs to know a nearby node to achieve locality
- Each routing hop matches the destination identifier by one more digit
  - Many choices in each hop (locality possible)

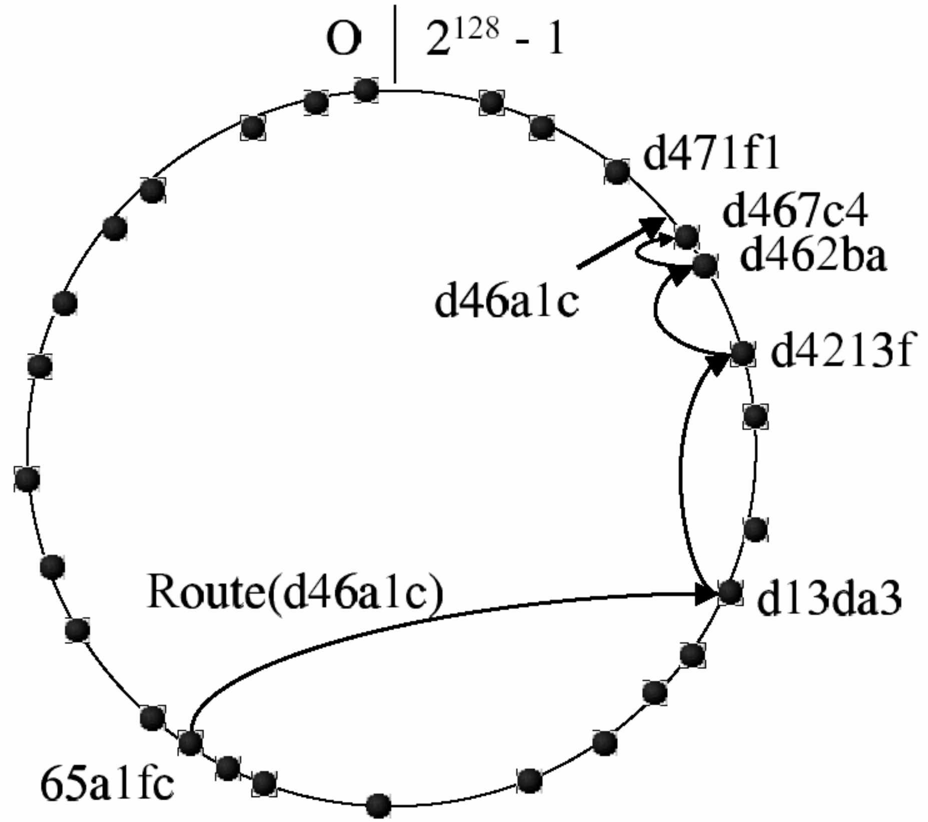


Figure 2: Routing a message from node $65a1fc$ with key $d46a1c$. The dots depict live nodes in Pastry's circular namespace.

CAN

- Based on a "d-dimensional Cartesian coordinate space on a d-torus"
- Each node owns a distinct zone in the space
- Each key hashes to a point in the space

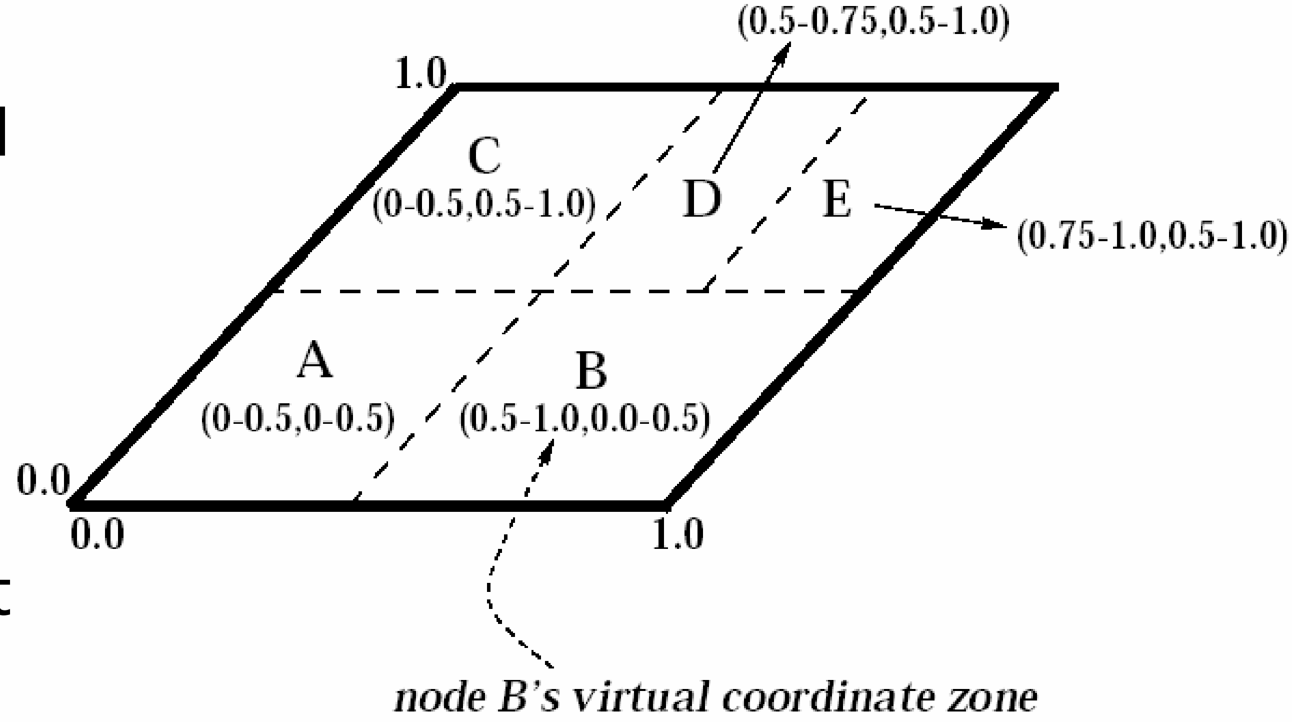
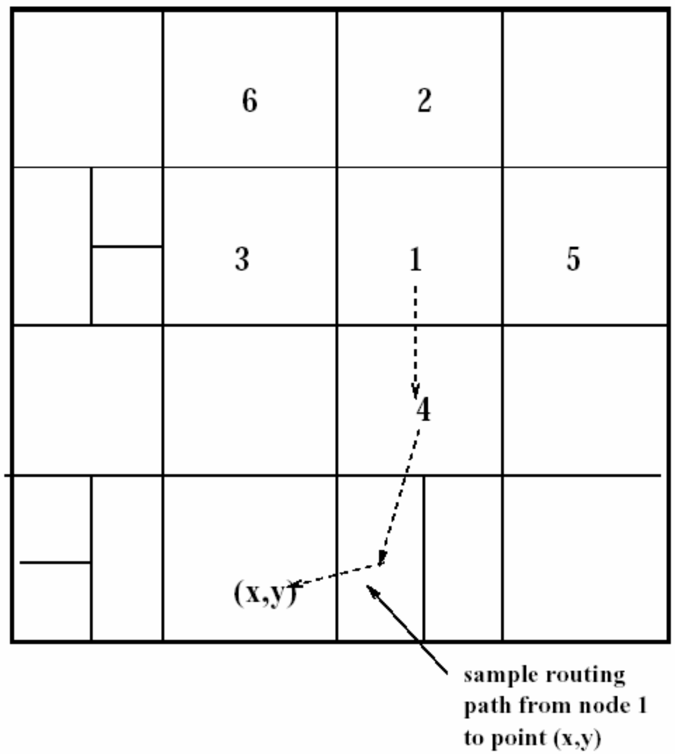


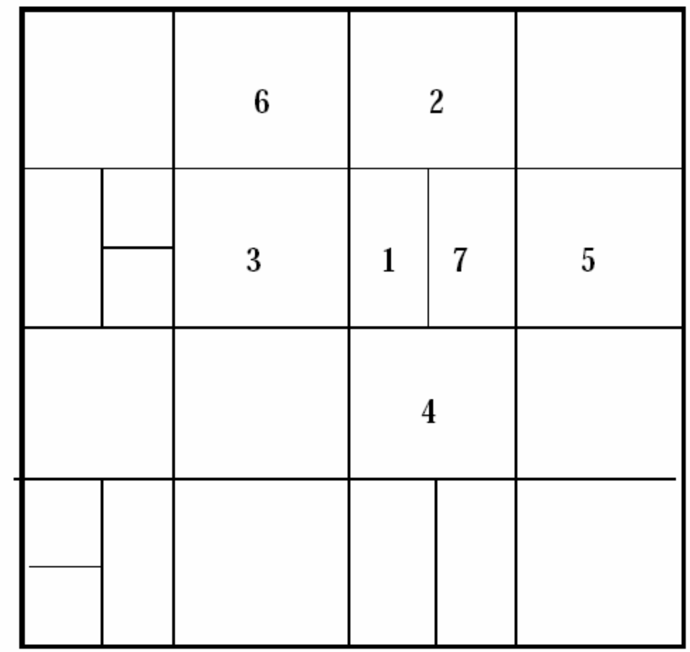
Figure 1: *Example 2-d space with 5 nodes*

CAN Routing and Node Arrival



1's coordinate neighbor set = {2,3,4,5}
7's coordinate neighbor set = { }

Figure 2: ***Example 2-d space before node 7 joins***



1's coordinate neighbor set = {2,3,4,7}
7's coordinate neighbor set = {1,2,4,5}

Figure 3: ***Example 2-d space after node 7 joins***

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)

Conclusions

- P2P connects devices at the edge of the Internet
- Popular in "industry"
  - Napster, Kazaa, etc. allow users to share data
  - Legal issues still to be resolved
- Exciting research in academia
  - DHTs (Chord, Pastry, etc.)
  - Improve properties/performance of overlays
- Applications other than file sharing are being developed