A perspective on P2P paradigm and services

"If a million people use a Web site simultaneously, doesn't that mean that we must have a heavy-duty remote server to keep them all happy? No; we could move the site onto a million desktops and use the Internet for coordination. Could amazon.com be an itinerant horde instead of a fixed Central Command Post? The answer is yes."

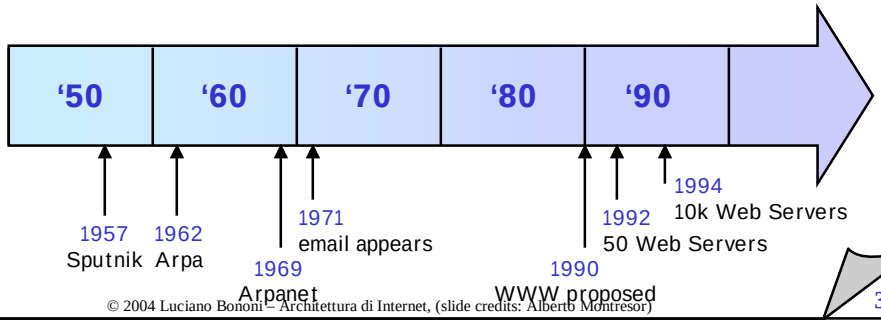
David Gelernter, *The Second Coming: A Manifesto*

Introduction

- **Recently, the *peer-to-peer* (P2P) paradigm for building distributed applications has gained attention from both industry and the media**
- **Peer-to-peer: "basic" definition**
  - A P2P system is composed of a distributed collection of *peer* nodes
  - Each node is both a server and a client:
    - may provide services to other peers
    - may consume services from other peers
- **Completely different from the client-server model, where:**
  - Few specialized servers provide services to a large number of clients

P2P History: 1969 - 1990

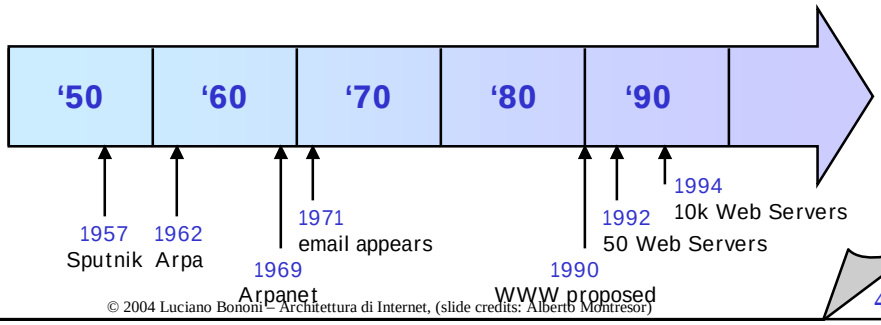
- **1969 – 1990: the origins**
  - In the beginnings, all nodes in Arpanet/Internet were peers
  - Every node was capable to
    - perform routing (locate machines)
    - accept ftp connections (file sharing)
    - accept telnet connections (distributed computation)



3

P2P History: 1995 - 1999

- **1990 – 1999: the Internet explosion**
  - The original "state of grace" was lost
  - Current Internet is organized hierarchically (client/server)
    - Relatively few servers provide services
    - Client machines are second-class Internet citizens (cut off from the DNS system, dynamic address)



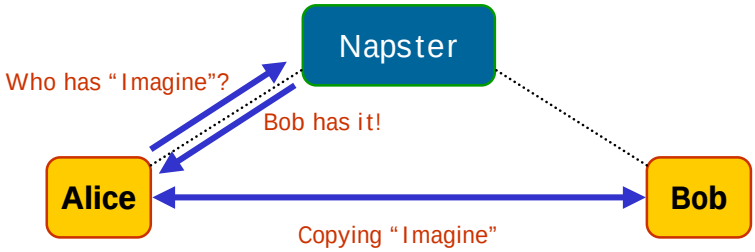
4

P2P History: 1999 - today

- **1999 – 2001: The advent of Napster**
  - Jan 1999: the first version of *Napster* is released by Shawn Fanning, student at Northeastern University
  - Jul 1999: Napster, Inc. founded
- **In short time, Napster gains an enormous success, enabling millions of end-users to establish a file-sharing network for the exchange of music files**
  - Jan 2000: Napster unique users > 1.000.000
  - Nov 2000: Napster unique users > 23.000.000
  - Feb 2001: Napster unique users > 50.000.000

Napster

- **The Napster architecture:**
  - Napster works by operating a central server which directs traffic between individual registered users
  - Each time a user submits a request for a song, the central server creates a list of users who are currently connected to Napster whose collections include the specified song



Napster is not alone

- **Following the success of Napster, other file-sharing systems started to appear, such as:**
  - Gnutella `gnutella.wego.com`
  - Freenet `freenet.sourceforge.net`
- **Moreover, other applications appeared, capable to establish communities comprising millions of cooperating nodes:**
  - Distributed Computing
    - Seti @ Home `setiathome.ssl.berkeley.edu`
    - Distributed.net `www.distributed.net`
  - Messaging and collaborative tools
    - Groove `www.groove.net`

P2P History: 2000 - today

- **2000 - today: the new peer-to-peer revolution**
  - Since 1999, the IT community started to search a label to define the new distributed model suggested by Napster and these other applications
  - By July 2000, this label was found: *peer-to-peer*
  - The label, however, didn't clarified things
    - Following the classical definition of peer-to-peer, Napster is not peer-to-peer (centralized server)
    - But Napster is what originated the discussion!
- **A new definition for peer-to-peer was needed**

Alternative Definitions of Peer-to-Peer

- **Peer-to-peer**
  - is the class of applications that put together resources available at peer machines located *at the edges of the Internet*
  - takes advantage of existing resources allowing users to leverage their *collective power to the 'benefit' of all*
  - is the sharing of computer resources and services by *direct exchange* between systems
  - refers to a class of systems and applications that employ distributed resources to perform a critical *function* in a *decentralized* manner

P2P and Piracy

- **Most of the material exchanged through P2P file-sharing systems is copyrighted**
- **Some of the P2P projects have *anonymity* among their goals:**
  - Freenet
  - Freehaven
- **This has resulted in the following equation:**

  P2P = subversion of intellectual property

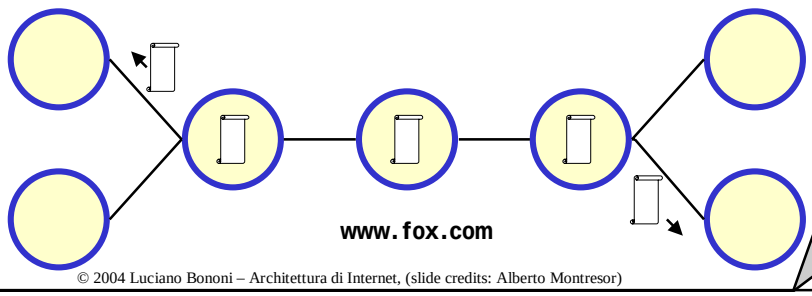
Why P2P?

- **Despite its poor reputation, P2P is extremely interesting from a technical point of view:**
  - Its completely decentralized model enables the development of applications with
    - *high-availability*
    - *fault-tolerance*
    - *scalability*

    characteristics previously unseen in Internet
  - It exploits what has been defined the “dark matter” of Internet
  - Moreover, P2P is not limited to file-sharing, but it can be applied to distributed computing and collaboration tools

P2P Examples

- **Example: avoiding the “Slashdot effect”**
  - “The more popular a piece of information is, the less available it becomes”
  - On the contrary, the number of replicas of a document in Freenet increases proportionally to its popularity
  - Can be applied to the distribution of movie trailers

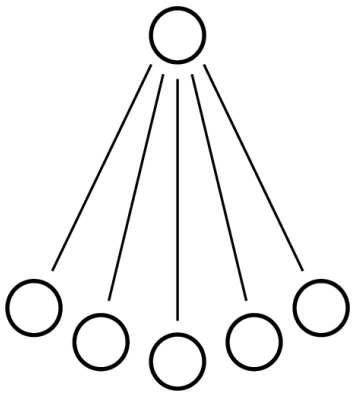


P2P Services

- **Areas of applicability of P2P**
  - sharing of *content*
    - distributed web servers, distributed media repository
  - sharing of *storage*
    - distributed file system, distributed search engine
  - sharing of *CPU time*
    - parallel computing
  - sharing of *human presence*
    - the “P” in P2P is “Person”

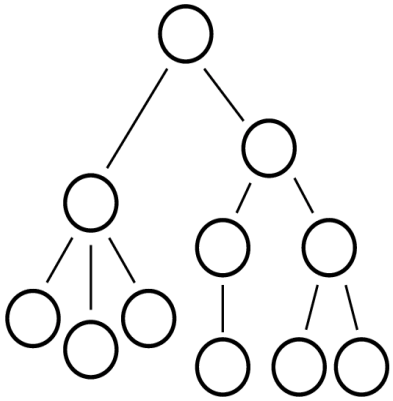
P2P topologies: Centralized

Manageable	✓	**System is all in one place**
Coherent	✓	**Information is centralized**
Extensible	X	**No**
Fault Tolerant	X	**Single point of failure**
Secure	✓	**Simply secure one host**
Lawsuit-proof	X	**Easy to shut down**
Scalable	?	**In theory, no**



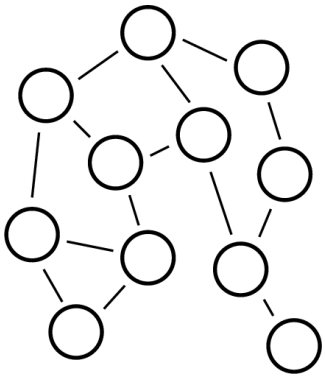
P2P topologies: Hierarchical

Manageable	½	Chain of authority
Coherent	½	Cache consistency
Extensible	½	Add more leaves, rebalance
Fault Tolerant	½	Root is vulnerable
Secure	X	Too easy to spoof links
Lawsuit-proof	X	Just shut down the root
Scalable	✓	Hugely scalable – DNS



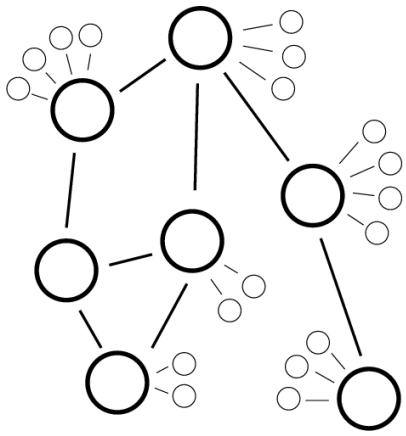
P2P topologies: Decentralized

Manageable	X	Very difficult, many owners
Coherent	X	Difficult, unreliable peers
Extensible	✓	Anyone can join in!
Fault Tolerant	✓	Redundancy
Secure	X	Difficult, open research
Lawsuit-proof	✓	No one to sue
Scalable	?	Theory – yes : Practice – no



P2P topologies: Centralized + Decentralized

Manageable	X	Same as decentralized
Coherent	½	Better than decentralized
Extensible	✓	Anyone can still join!
Fault Tolerant	✓	Plenty of redundancy
Secure	X	Same as decentralized
Lawsuit-proof	✓	Still no one to sue
Scalable	?	Looking very hopeful



Best architecture for P2P networks?

Pure Peer-to-Peer: Key Questions

- **Does it work?**
  - can we find the data?
  - query success rates
    - length of query paths
- **Does it scale?**
  - logarithmic / linear / polynomial
- **Is it robust?**
  - participants are unreliable
  - different failure modes possible

Peer-to-Peer Systems

- **Pure Peer-to-Peer Content Sharing Systems**
  - Gnutella
  - Freenet
- **Master-Slave Cycle Sharing Systems**
  - Seti@Home
  - distributed.net

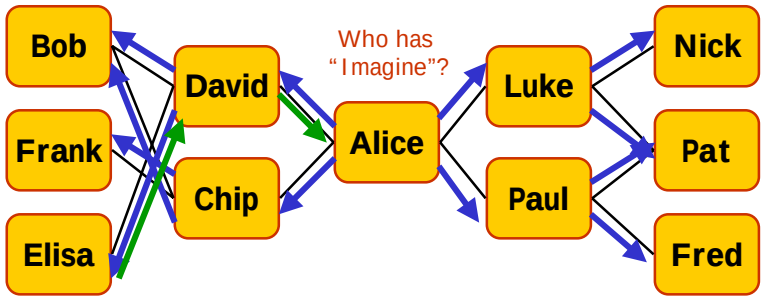
Gnutella

- **The Gnutella protocol consists of:**
  - a set of message types representing the ways in which servents communicate over the network
  - a set of rules governing the inter-servent exchange of messages
- **How to connect to a Gnutella network**
  - A Gnutella servent connects to the network by establishing a connection with another servent currently on the network
  - The acquisition of another servent's address is not part of the protocol definition
    - "Out-of-band" methods

Two types of messages

- **Broadcast**
  - sent to all nodes with which the sender has open TCP connections
  - This poses serious problems of scalability
  - Mechanisms are used to reduce the number of messages
- **Back-propagate**
  - sent on a specific connection on the reverse of the path taken by an initial broadcasted message

Two types of messages



Gnutella Messages

- **Each message is composed of:**
  - A message type field
    - PING, PONG
    - QUERY, QUERYHIT
    - PUSH
  - A Time-To-Live (TTL) Field
    - The number of times the message will be forwarded by servents before it is removed from the network
    - Decremented at each hop, lowered if needed
  - A 16-byte ID field uniquely identifying the message on the network
    - Randomly generated
    - Not related to the address of the requester (anonymity)

Gnutella Message Types

- **PING**
  - essentially, an “are you there” message directed to a host
  - a servent receiving a PING is expected to respond with a PONG message
  - no recommendation as to the frequency of PING messages
- **PONG**
  - the response to a PING
  - has the same ID of the corresponding PING message
  - contains:
    - address of connected Gnutella servent
    - total size and total number of files shared by this servent

Gnutella Message Types

- **QUERY**
  - The primary mechanism for searching the distributed network
  - Contains the query string
  - A servent is expected to respond with a QUERYHIT message if a match is found against its local data set
- **QUERYHIT**
  - the response to a query
  - has the same ID of the corresponding QUERY message
  - contains enough information to acquire the data matching the corresponding query
    - IP Address + port number
    - List of file names

Gnutella Message Forwarding Rules

- **A servent receiving a message must forward it to each of the other servents to which it is connected**
- **Exceptions:**
  - TTL is zero
  - PONG messages must be routed along the same path of the incoming PING
  - QUERYHIT messages must be routed along the same path of incoming QUERY messages
  - Messages with duplicated ID should be discarded
- **A cache of message IDs, along with sender, is needed**

Interpretations of Query Strings

- **Gnutella is a simple protocol**
  - Defines only how a query string is passed from one site to another
  - Uses Http to effectively download the data
- **How queries are interpreted?**
  - Different implementations may interpret a string in different ways
    - Ex: by searching the string in set of filenames
    - Ex: by running grep on a set of files
  - Flexibility:
    - each site may contribute to a distributed search in a complex way
    - different gnutella networks may solve distinct problems

Network Topology

- **Application-level virtual network**
- **Autonomous, self-organized, dynamic**
- **Multiple access points**
- **Advantages:**
  - Increase system reliability
  - Less dependant on a single server

Mismatch between Gnutella Network and Internet Infrastructure

- **Only 2-5% of Gnutella connections link nodes located within the same AS.**
- **Most Gnutella generated traffic crosses AS border, making the traffic more expensive**
- **May affect ISPs to change pricing scheme**

Scalability

- **Limited horizon**
  - The number of reachable nodes is limited by *TTL* and the number *N* of concurrent connections
- **Scalability**
  - The number of messages exchanged increases exponentially with the increase of TTL and N
  - With data packet (s) = 83 bytes
  - TTL = 8,
  - number of connections (n)= 8
  - Number of user reached:
  - Bandwidth incurred = 1,275,942,400 bytes
  - Therefore, 18 bytes of query generated 1.2GB of traffic!

Security

- **Denial of Service attacks**
  - Flooding the system with requests
    - Strange traffic observed in Gnutella
  - Solution: keep statistics about frequency of requests and close connections with offending nodes
- **Privacy attacks**
  - A site advertised file names that appeared to offer child pornography
  - It logged the IP address and domain name of every download request (included in HTTP)
  - Solutions: none at present

Gnutella: Conclusions

- **Gnutella pros**
  - Simple architecture,easily implementable, could be profitably used for small groups
- **Gnutella cons**
  - Not scalable

Freenet

- **Freenet:**
  - An adaptive peer-to-peer application that permits the publication, replication and retrieval of data while protecting the anonymity of users
- **Philosophy:**

  I worry about my child and the Internet all the time, even though she's too young to have logged on yet. Here's what I worry about. I worry that 10 or 15 years from now, she will come to me and say 'Daddy, where were you when they took freedom of the press away from the Internet?'"

  Mike Godwin, Electronic Frontier Foundation

Freenet Goals

- **Socio-political goals of Freenet**
  - Publisher anonymity
  - Reader anonymity
  - Server anonymity
  - Resistance to attempts by attackers to deny access to data
    - Denial-of-service attacks
    - Removal attacks
- **Technical goals of Freenet**
  - Decentralization of all network functions
  - Data replication/distribution without human intervention
  - Efficient dynamic storage and routing of information
  - High availability for popular data

Freenet Design

- **Freenet is a P2P network of nodes storing data files**
- **Data files:**
  - are named by location-independent *keys*
- **Freenet nodes:**
  - maintain a *datastore* and make it available to the network:
    - for reading
    - for writing
  - maintain a *dynamic routing table* containing
    - addresses of other hosts
    - keys they are thought to hold
  - query one another to store and retrieve data files

Freenet Design

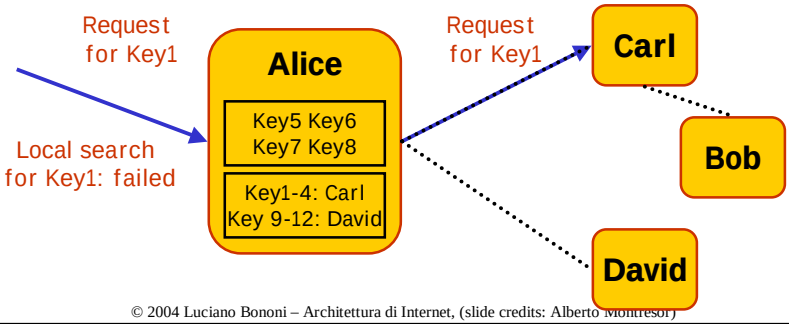
- **Freenet differs from Napster and Gnutella**
  - it is not based on a central server
  - it is not based on broadcasts
- **Freenet is adaptive**
  - responds adaptively to usage patterns
  - transparently moves, replicates, deletes files as needed
- **Freenet and persistency:**
  - it is not intended to guarantee permanent file storage
  - but most files may persist indefinitely, if enough nodes join

Freenet Messages

- **All messages contain:**
  - A randomly-generated 64 bit *Transaction ID*
    - Unique “with high probability”
  - A *TTL* field (Hop-to-Live)
  - A *Depth* field (number of hops performed so far)
- **Messages are forwarded from node to node**
  - TTL *decremented* at each hop
    - Message not discarded when TTL reaches 1
    - Randomly forwarded for other steps (for anonymity)
  - Depth *incremented* at each hop
    - Used in reply messages to set the TTL
    - Does not start at 0 (for anonymity)

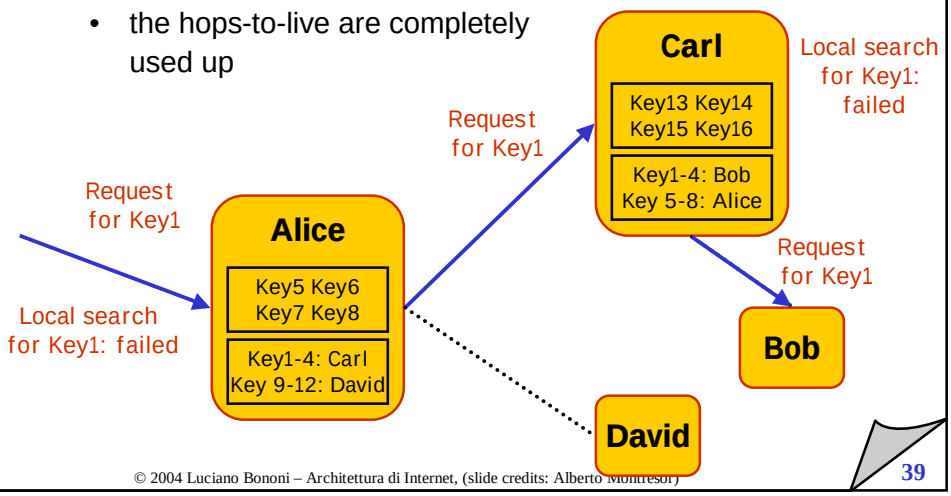
Freenet Algorithm: Request

- **When a node receive a request for a key:**
  - Attempts to retrieve the file locally, if possible
  - Otherwise, forwards the request to another node
    - Which node? Local decision in IP-style
    - Decision depends on the key



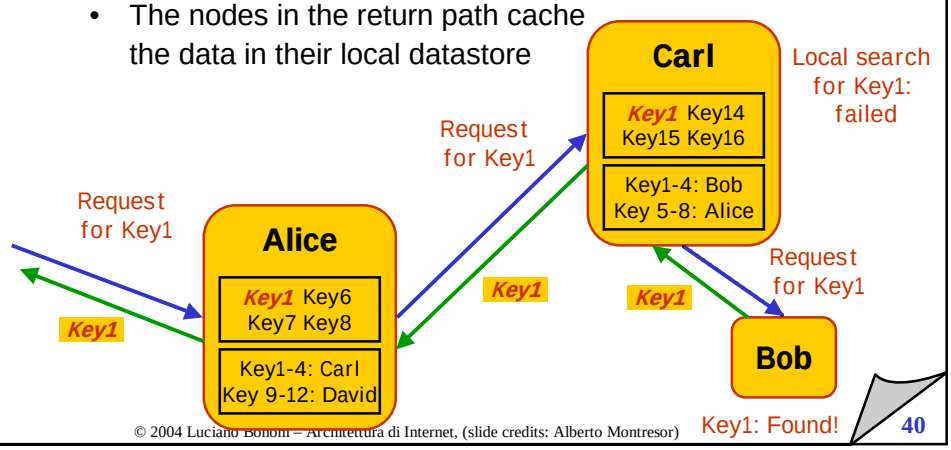
Freenet Algorithm: Request

- **Requests are passed from node to node until**
  - the requested key is found
  - the hops-to-live are completely used up



Freenet Algorithm: Request

- **When a request is successful (found in the local datastore)**
  - The data are returned to the requester along the same path of the incoming request
  - The nodes in the return path cache the data in their local datastore



Freenet Pros

- **Removal of unwanted documents:**
  - The LRU policy of the datastore:
    - removes outdated documents
    - removes rarely accessed documents
- **Reader/Publisher Anonymity**
  - A node in a request path cannot tell whether its predecessor in the path initiated the request or not
    - Messages not immediately discarded when TTL=1
    - Depth starting with a value greater than 0
  - Note: Possible use of traffic analysis
- **Server anonymity / Deniability:**
  - Difficult to relate a document to a server
  - Operators can deny to know the content of its datastore

Freenet Cons

- **Search Problem**
  - The keys are Freenet's current weak point
  - Hashing of keys:
    - Human-Rights.doc and HumanRights.doc have completely different hash values
  - Hashing renders Freenet unusable for random searches
    - Need an "out-of-band" communication of keys
- **A solution:**
  - Gateway to the Web exists under the name of Fproxy
  - Possibility of using hyperlinks

P2P: The Next Generation

- **First generation:**
  - Napster, Gnutella, Freenet…
  - intended for large scale sharing of data files
  - reliable content location was not guaranteed
  - self-organization and scalability: to be addressed
- **Second generation:**
  - Pastry, Tapestry, Chord, CAN…
  - guarantee a definite answer to a query in a bounded number of network hops.
  - form a self-organizing overlay network

P2P: The Next Generation

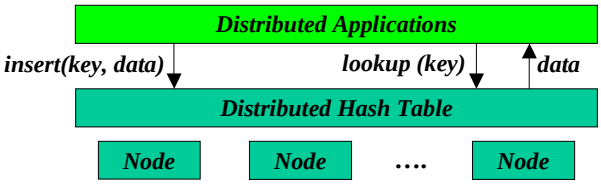
- **Content-based addressing**
  - hash content to key
  - route message to computer hosting that key
- **Dynamic caching and proxying**
  - local computers stand in for remote ones
  - faster access, reduced load on key holder
- **Replication and automatic failover**
  - store at K computers adjacent to key holder
- **Multicast cascade for group communication**
  - each computer needs a spanning tree of routes for reaching every other computer

Overlay Networks

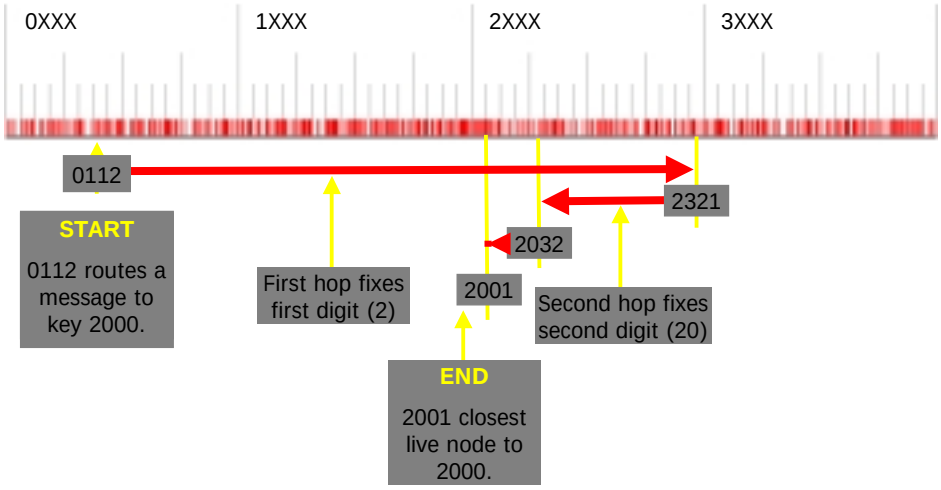
- **Peer-to-peer requires richer routing semantics than IP**
  - IP routes to destination computer, not content
  - URLs route to destination computer, not content
  - IP multicast isn't widely deployed
- **Solution: Overlay networks**
  - allow applications to participate in hop-by-hop routing decisions
- **Ideal overlay is efficient, self-organizing, scalable, and fault-tolerant**

Distributed Hash Tables

- **Each node handles a portion of the hash space and is responsible for a certain key range**
  - no global knowledge
  - absence of single point of failures
  - greater scalability
  - uniform distribution of resources
  - Examples: CAN, Chord, Pastry, Tapestry



Pastry routing



Pastry routing table

Routing table:
For each level, nearest peer for other domains

Namespace leaf set:
nearest lds to "left" and "right" in name space

Each entry gives IP address for host associated with Id