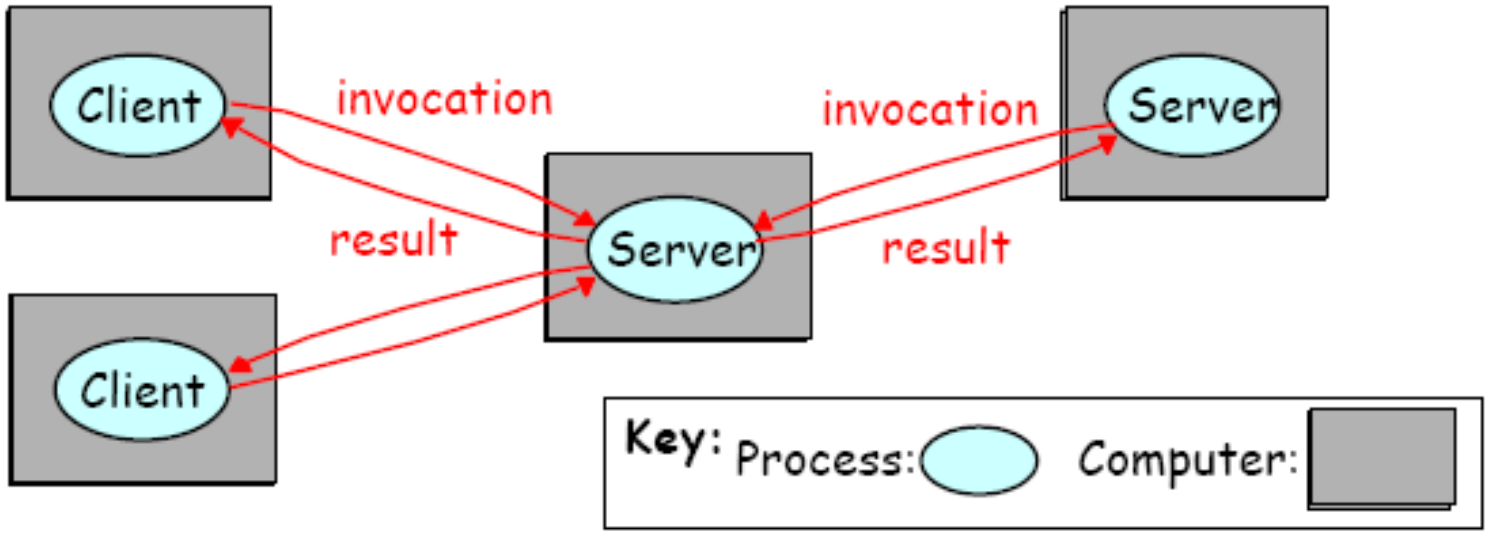
Peer-to-Peer Networking

Acknowledgements: Most materials presented in the slides are based on the tutorial slides made by Dr. Raouf Boutaba, Dr. Keith W. Ross, and Dr. Dan Rubenstein.

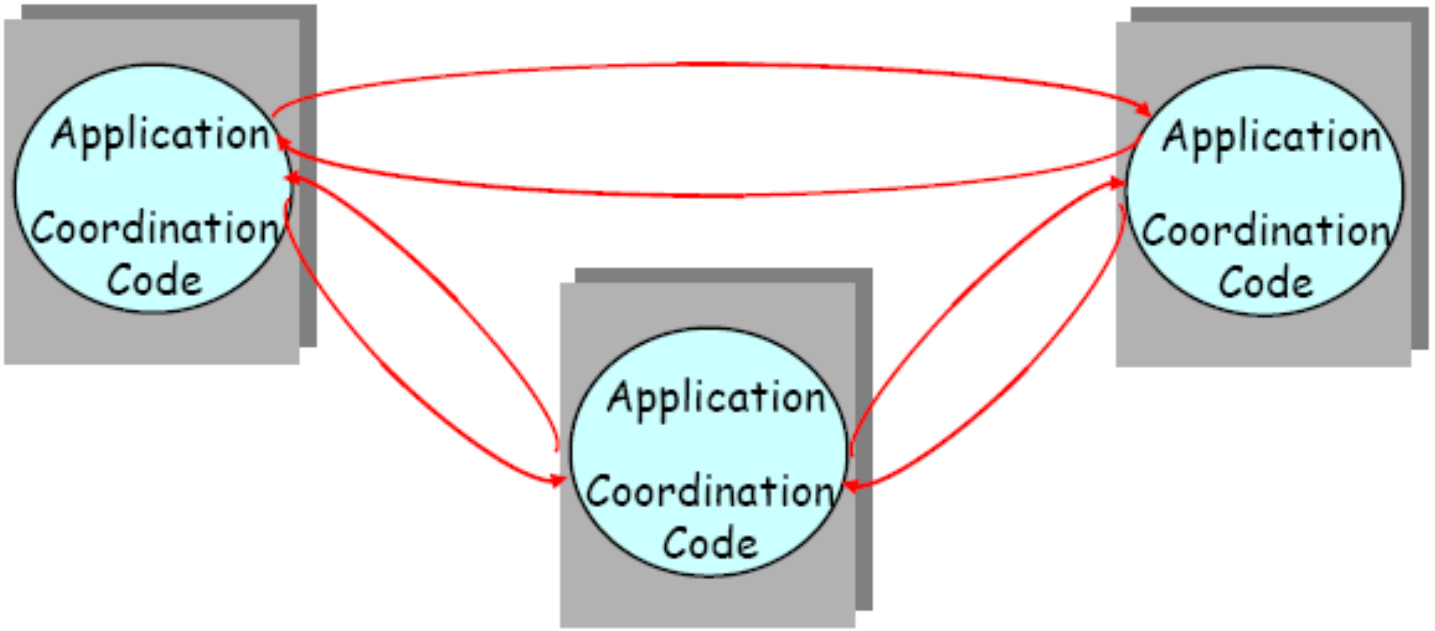
Client/Server Computing Model



- **Client:** Process wishing to access data, use resources or perform operations on a different computer
- **Server:** Process managing data and all other shared resources amongst servers and clients, allows clients access to resource and performs computation
- **Interaction:** invocation / result message pairs

The Peer-to-Peer Model

- Applications based on peer processes
  - Not Client-Server
  - processes that have largely identical functionality



The Peer-to-Peer Mania

- Started in the middle of 2000
- When the Internet fallen into predictable patterns
- Computer field shocks:
  - Napster, SETI@home, Freenet, Gnutella, Jabber, …
  - many early P2P projects have an overtly political mission
- Emergence of sporadically connected Internet nodes (laptops, handhelds, cell phones, appliances, …)

These developments return content, choice, and control to ordinary users. Tiny endpoints on the Internet, sometimes without even knowing each other, exchange information and form communities.

The Peer-to-Peer Mania (cont.)

- A new energy erupting in the computing field
- Yet, P2P is the oldest architecture in the world of communications
  - Telephones are peer-to-peer
  - Usenet implementation of UUCP
  - Routing in the Internet
  - Internet endpoints have historically been peers

P2P technologies return the Internet to its original vision, in which everyone creates as well as consumes

Outline

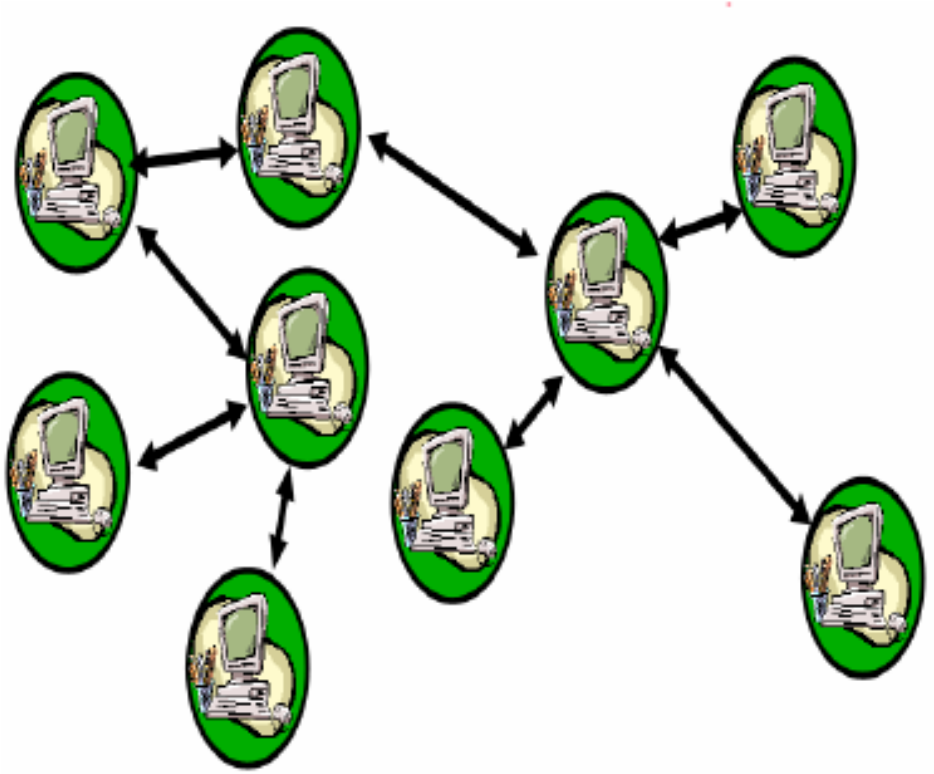
- General overview
  - Definition
  - Overlay networks
  - Goals
  - Classification of P2P systems
- P2P Search activities
  - Applications
  - Search
  - Security
  - Resource management
- Case Studies
  - File-sharing systems
  - Distributed Hash Tables (DHT)
  - Trust and reputation management

Outline

- General overview
  - Definition
  - Overlay networks
  - Goals
  - Classification of P2P systems
- P2P research activities
- Case Studies

Definitions

- Everything except the client/server model
- Network of nodes with equivalent capabilities/responsibilities (symmetrical)
- Nodes are both Servers and clients called "Servents"
- Direct exchange of information between hosts at the edge of the Internet



Definitions (cont.)

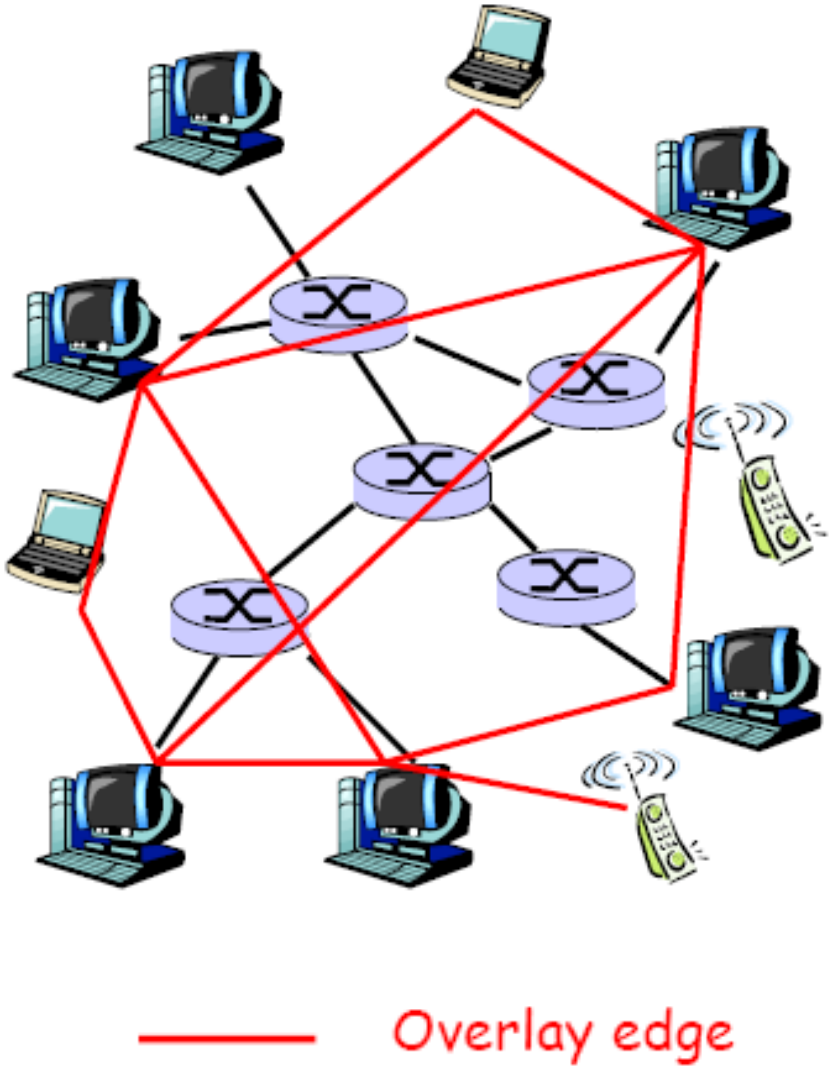
- A transient network that allows a group of computer users to connect with each other and collaborate by sharing resources (CPU, storage, content).
- The connected peers construct a virtual overlay network on top of the underlying network infrastructure
- Examples of overlays:
  - BGP routers and their peering relationships
  - Content distribution networks (CDNs)
  - Application-level multicast
  - And P2P apps !

Outline

- General overview
  - Definition
  - Overlay networks
  - Goals
  - Classification of P2P systems
- P2P research activities
- Case Studies

Overlay Networks

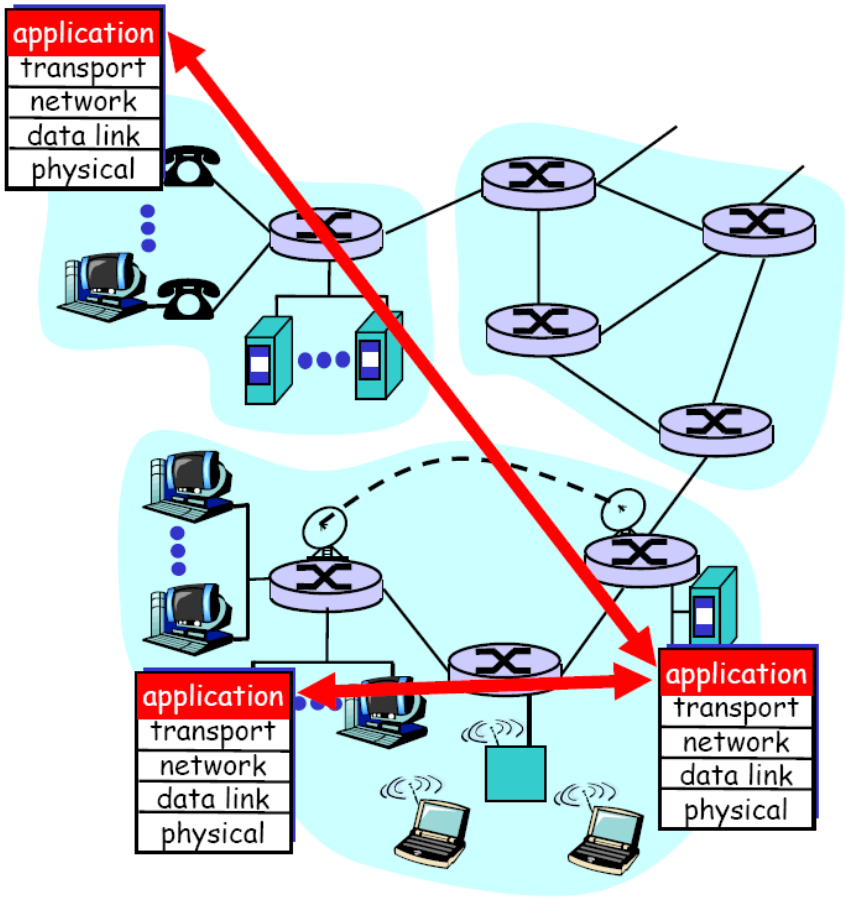
- An overlay network is a set of logical connections between end hosts
- Overlay networks can be unstructured or structured
- Proximity not necessarily taken into account
- Overlay maintenance is an issue



Overlays: All in the application layer

- Design flexibility
  - Topology
  - Protocol
  - Messaging over TCP, UDP, ICMP

Underlying physical net is transparent to developer



Outline

- General overview
  - Definition
  - Overlay networks
  - Goals
  - Classification of P2P systems
- P2P research activities
- Case Studies
- Future research directions

Goals

- Cost reduction through cost sharing
  - Client/Server: Server bears most of the cost
  - P2P: Cost spread over all the peers (+Napster, ++Gnutella,…)
  - aggregation of otherwise unused resources (e.g., seti@home)
- Improved scalability/reliability
  - resource discovery and search algorithms (eg. Chord, CAN, …)
- Interoperability
  - for the aggregation of diverse resources (storage, CPU, …)
- Increased autonomy
  - independence from servers, hence providers (e.g., A way around censorship, licensing restrictions, etc.)

Goals (cont.)

- Anonymity/privacy
  - Difficult to ensure with a central server
  - Required by users who do not want a server/provider to know their involvement in the system
  - Freenetis a prime example
- Dynamism and Ad hoc communications
  - Resources (e.g., compute nodes) enter and leave the system continuously
  - Mechanisms are required to avoid polling (e.g., "buddy lists" in Instant messaging)
  - P2P systems typically do not rely on an established infrastructure
  - they build their own, e.g. logical overlay in CAN

Outline

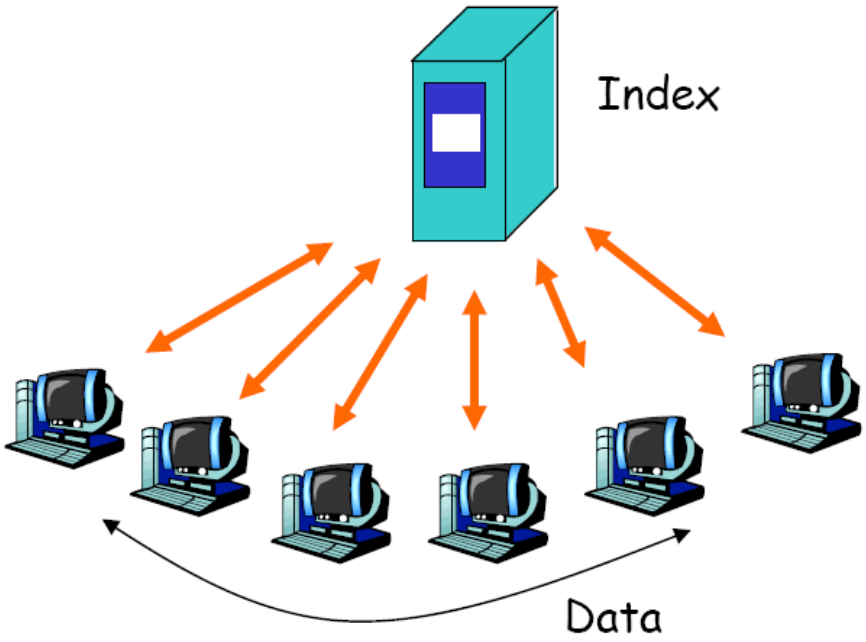
- General overview
  - Definition
  - Overlay networks
  - Goals
  - Classification of P2P systems
- P2P research activities
- Case Studies

P2P Classification

- Degree of P2P decentralization
  - Hybrid decentralized P2P
  - Purely decentralized P2P
  - Partially centralized P2P
- Degree of P2P structure
  - Structured P2P
  - Loosely structured P2P
  - Unstructured P2P

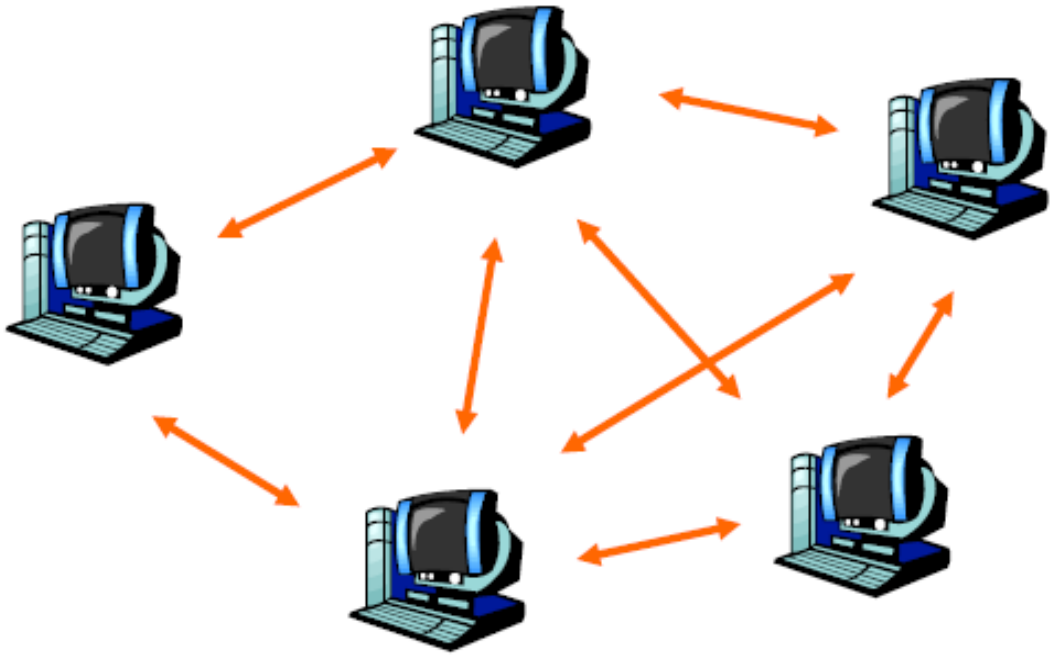
Hybrid decentralized P2P

- Central server facilitates the interaction b/w peers.
- Central server performs the lookups and identifies the nodes of the network.
- example: Napster
- (-) Single point of failure, scalability?, …



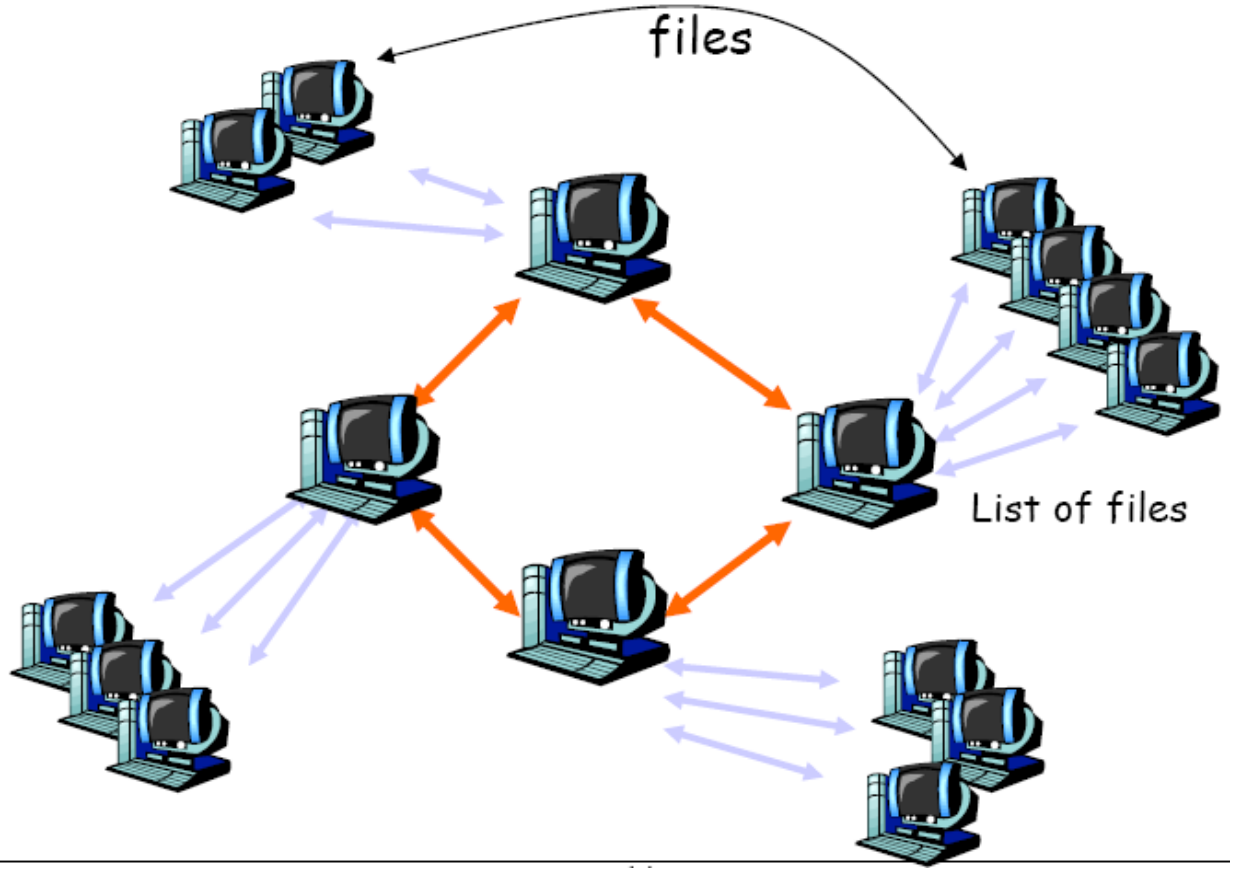
Purely decentralized P2P

- network nodes perform the same tasks (Servents)
- no central coordination activity
- examples: original Gnutella, Freenet
- (-) data consistency?, Manageability?, Security?, Comm. overhead



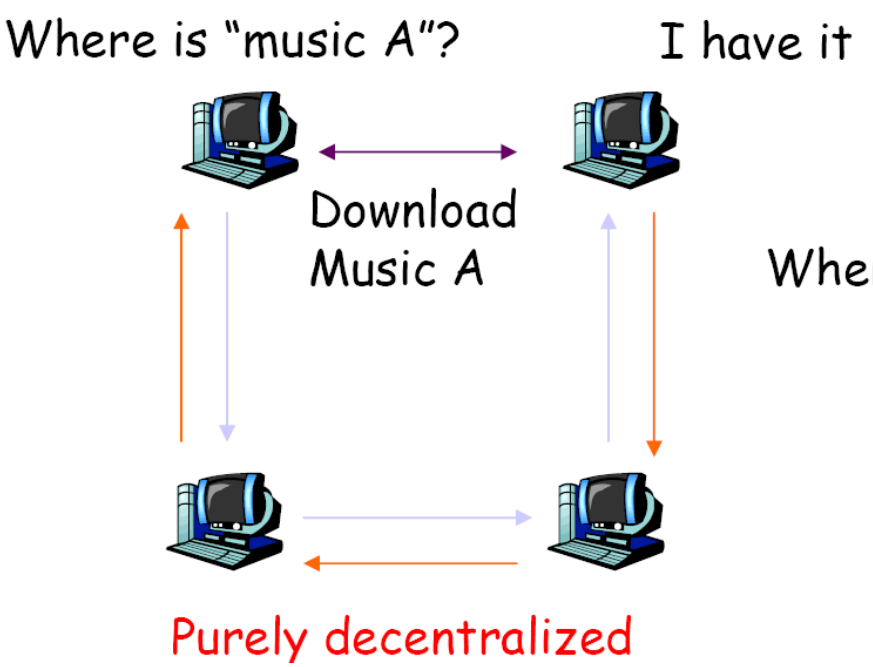
Partially centralized P2P

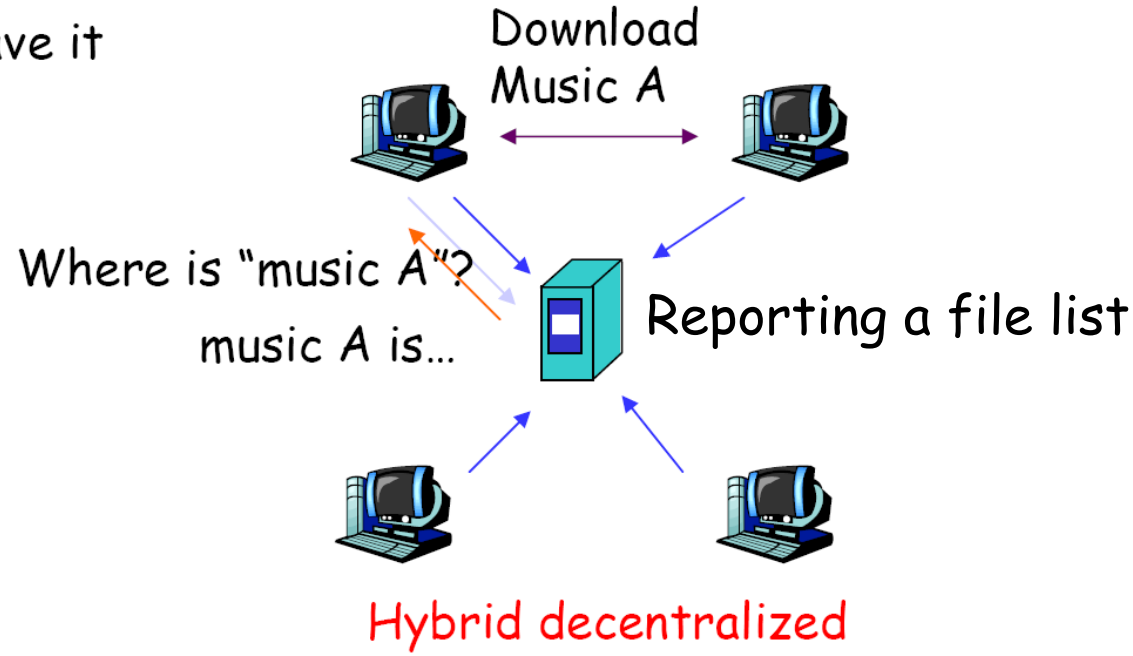
- some of the nodes assume a more important role
- Supernodes act as local central indexes
- examples: Kazaa, recent Gnutella



Unstructured P2P

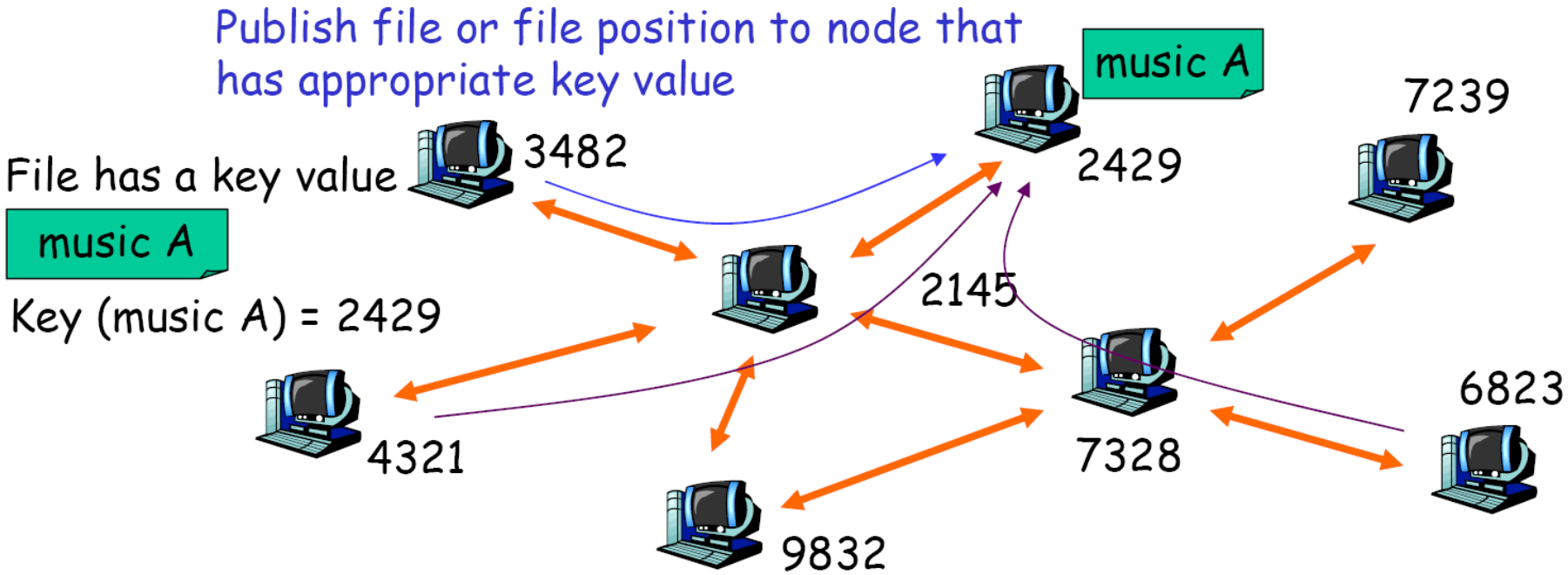
- data is distributed randomly over the peers and broadcasting mechanisms are used for searching.
- placement of data is unrelated to the overlay topology.
- examples: Napster, Gnutella, KaZaa





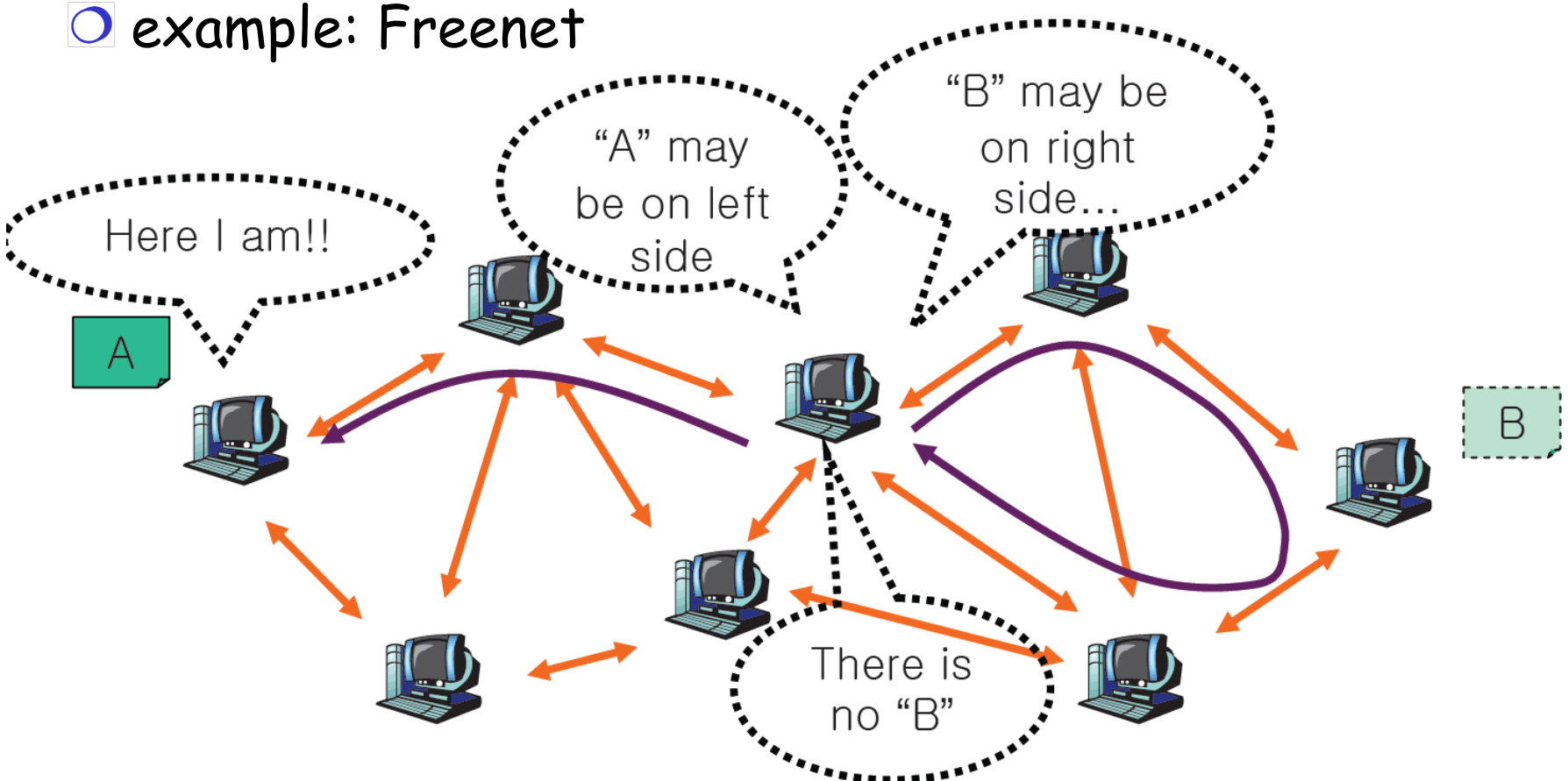
Structured P2P

- Network topology is tightly controlled and files are placed at precisely specified locations.
- Provide a mapping between the file identifier and location
- Examples: Chord, CAN, PAST, Tapestry, Pastry, etc.



Loosely Structured P2P

- Between structured and unstructured
- File locations are affected by routing hints, but they are not completely specified.
- example: Freenet



P2P classification summary

	Unstructured Networks	Loosely Structured Networks	Structured Networks
Hybrid Decentralized	Napster		
Pure Decentralized	Gnutella	Freenet	Chord, CAN, Tapestry
Partially Centralized	KaZaa, new-Gnutella		

Outline

- General overview
- P2P research activities
  - Applications
  - Search
  - Security
  - Resource Management
- Case Studies
- Future research directions

Outline

- General overview
- P2P research activities
  - Applications
  - Search
  - Security
  - Resource Management
- Case Studies

P2P Applications

- File Sharing
- Communication
- Collaboration
- Computation
- Databases
- Others

P2P File Sharing

- File exchange: Killer application!
- (+) Potentially unlimited file exchange areas
- (+) High available safe storage: duplication and redundancy
- (+) Anonymity : preserve anonymity of authors and publishers
- (+) Manageability
- (-) Network bandwidth consumption
- (-) Security
- (-) Search capabilities

P2P File Sharing (cont.)

- Examples of P2P file sharing applications:
  - Napster
    - disruptive; proof of concept
  - Gnutella
    - open source
  - KaZaA
    - at some point, more KaZaAtraffic than Web traffic!
  - eDonkey
    - popular in Europe
    - Some implementations use Kademlia(DHT) for search
  - BitTorrent:
    - 53% of all P2P traffic in June 2004 was BitTorrenttraffic
  - and many others…
- How do you explain this success?

P2P Communication

- Instant Messaging (IM)
  - User A runs IM client on her PC
  - Intermittently connects to Internet; gets new IP address for each connection
  - Registers herself with "system"
  - Learns from "system"that user B in her "buddy list"is active
  - User A initiates direct TCP connection with User B: P2P
  - User A and User B chat.
  - Can also be voice, video and text.
- Audio-Video Conferencing
  - Example: Voice-over-IP (Skype)

P2P Collaboration

- Application-level user collaboration
- Shared Applications
  - Shared file editing (eg. Distributed Powerpoint)
  - Example: Groove
- Online games
  - Multi-players, distributed
  - Example: Descent (www.planetdescent.com)
- Technical challenges
  - Locating of peers
  - Fault tolerance
  - Real-time constraints

P2P Computation

- Achieves processing scalability by aggregating the resources of large number of individual PC.
- Application areas
  - Financial applications
  - Biotechnology
  - Astronomy,…
- Related projects
  - seti@home
  - Avaki
  - Entropia
  - Gridella

P2P Databases

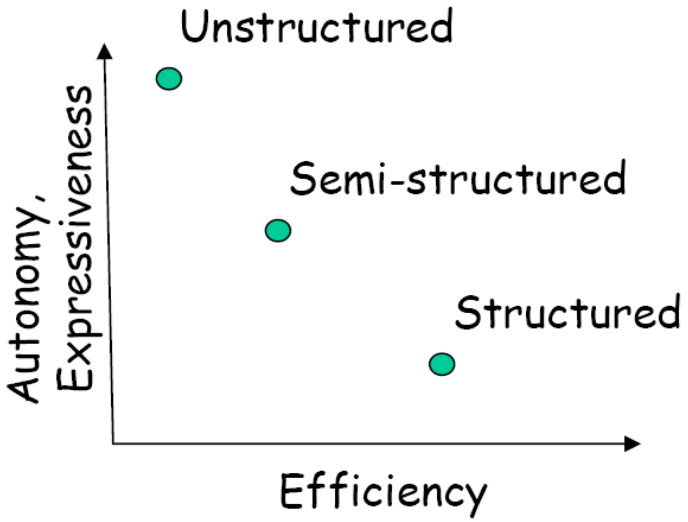
- Fragments large database over physically distributed nodes
- Overcomes limitations of distributed DBMS
  - Static topology
  - Heavy administration work
- Dissemination of data sources over the Internet
  - Each peer is a node with a database
  - Set of peers changes often (site availability, usage patterns)
- Examples:
  - AmbientDB (http://homepages.cwi.nl/~boncz/ambientdb.html)
  - XPeer: self-organizing XML DB
- No global schema

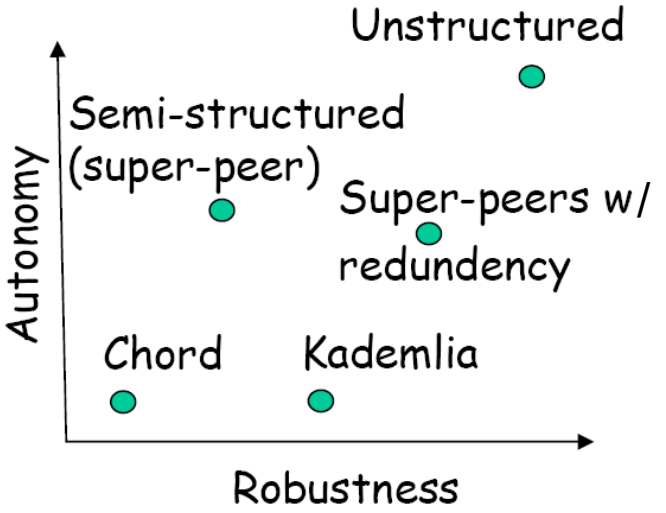
Outline

- General overview
- P2P research activities
  - Applications
  - Search
  - Security
  - Resource Management
- Case Studies

Search

- Challenges:
  - Efficiency
    - Measure: resource (mostly bandwidth) consumed
  - Autonomy:.
    - Level of control at individual node on data/index placement, connectivity & message routing
  - Robustness
    - Stability in the presence of transient population of peers
  - Scalability
    - Accommodate millions of nodes without degrading efficiency
  - Expressiveness
    - of query expression: e.g. key lookup, keyword lookup, range queries etc.
  - Completeness:
    - Guarantee on successful search and retrieval of all matches





Search taxonomy

- Blind search
- Informed search
  - Local Index
    - Proactive
    - Reactive
  - Distributed Index
    - DHT-based systems

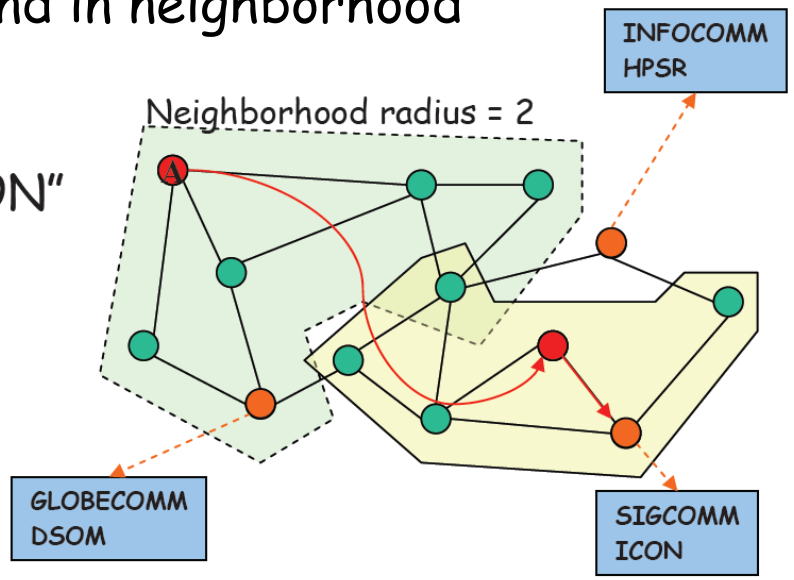
Search taxonomy

- Various **Blind Search** **Methods:**
  - **BFS/Flooding** (Gnutella)
    - forward incoming query to every neighbor.
    - Flooding is restricted by TTL (time-to-live)
  - **Modified BFS**
    - Flood only a portion of the neighbors,
    - Reduces search traffic at the cost of lower hit rate.
  - **Random walk**
    - forward query message to randomly chosen neighbors
    - Uses a number of walkers in parallel
  - **Iterative Deepening**
    - uses consecutive BFS at increasing depth.
    - Suitable when number of hits required is pre-specified and the possibility of matches at neighbors within a small radius.

Search Taxonomy

- Local index: example **proactive** methods
  - Neighborhood Signature Search
    - Each node maintains a set of bloom-filters (hash summary) of the keywords stored at peers within a small radius (r)
    - Query is forwarded to a peer out of neighborhood radius in case no match is found in neighborhood

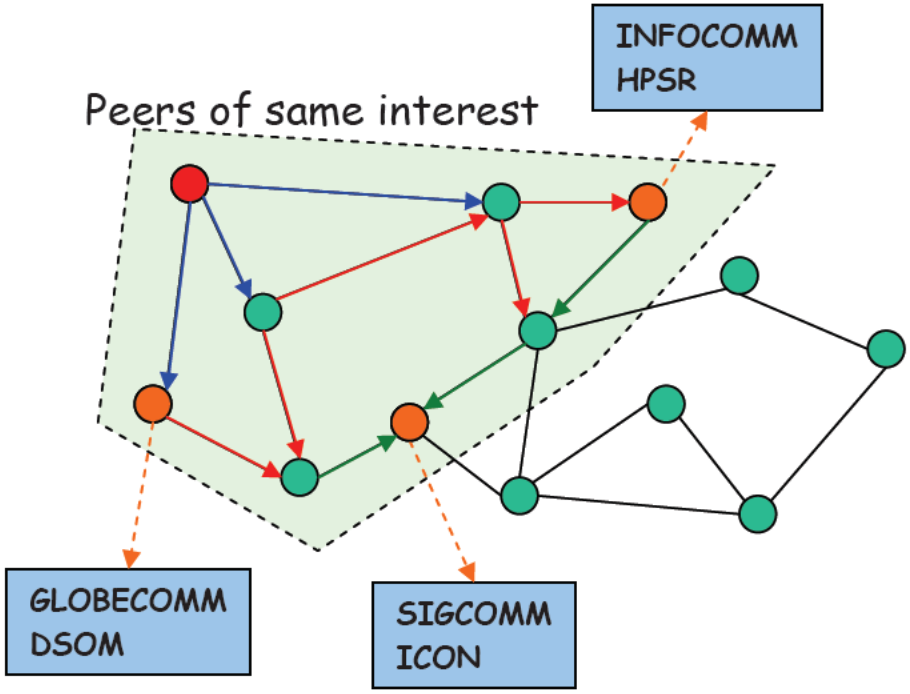
Peer A is looking for "ICON"



Neighbor signature Search

Search Taxonomy

- Local index: example **proactive** methods
  - Associative Search
    - Nodes are arranged into interest groups.
    - Query is flooded in one or more target interest group(s)



Associative Search

Search Taxonomy

- Local Index: example **reactive** methods
  - Distributed Resource Location Protocol (DRLP)
    - Uses random walk when no information is available
    - In case of a hit, the document location is stored at each node along the reverse path to the requester
  - Intelligent BFS:
    - Informed version of modified-BFS
    - Query routing is guided by past routing decision
    - Nodes store query-neighborID tuples
    - In case of a hit, routing information at all nodes on the reverse path is updated.

What is a DHT?

- Hash Table
  - data structure that maps "keys" to "values"
- Interface
  - put(key, value)
  - get(key)
- Distributed Hash Table (DHT)
  - similar, but spread across the Internet
  - challenge: locate content

What is a DHT? (cont.)

- Single-node hash table:

  Key = hash (data)

  put(key, value)

  get(key)->value

- Distributed Hash Table (DHT):

  Key = hash (data)

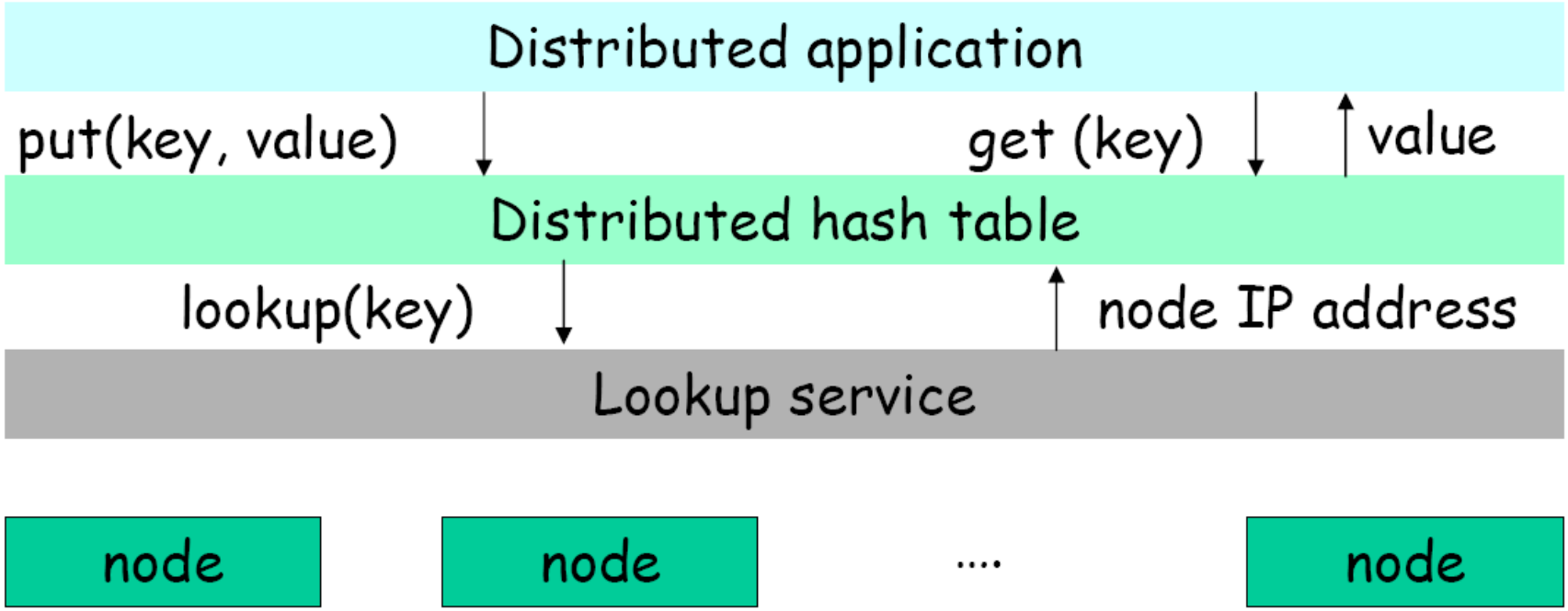
  Lookup (key) -> node-IP@

  Route (node-IP@, PUT, key, value)

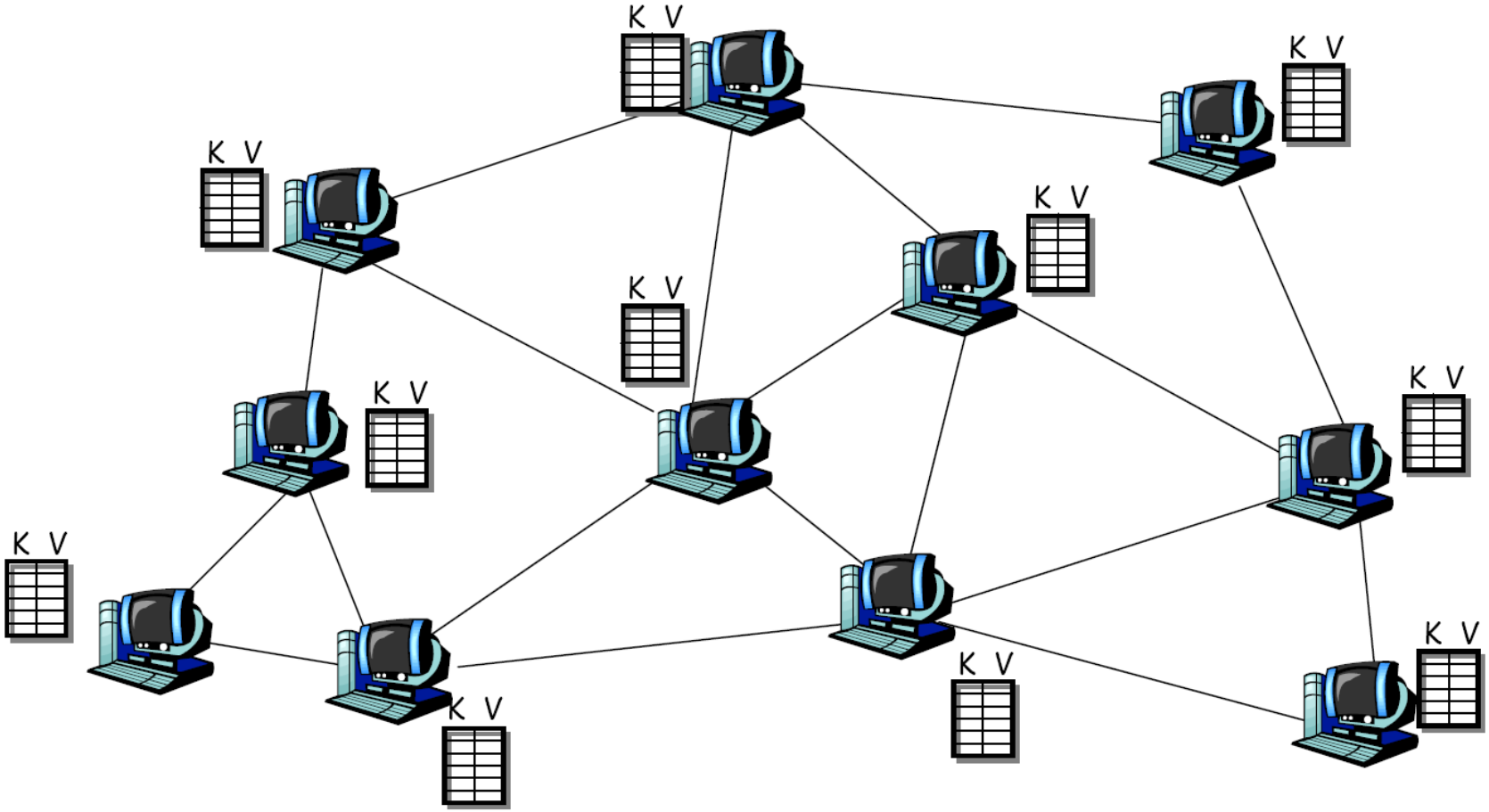
  Route (node-IP@, GET, key) -> value

- Idea:
  - Assign particular nodes to hold particular content (or reference to content)
  - Every node supports a routing function (given a key, route messages to node holding key)

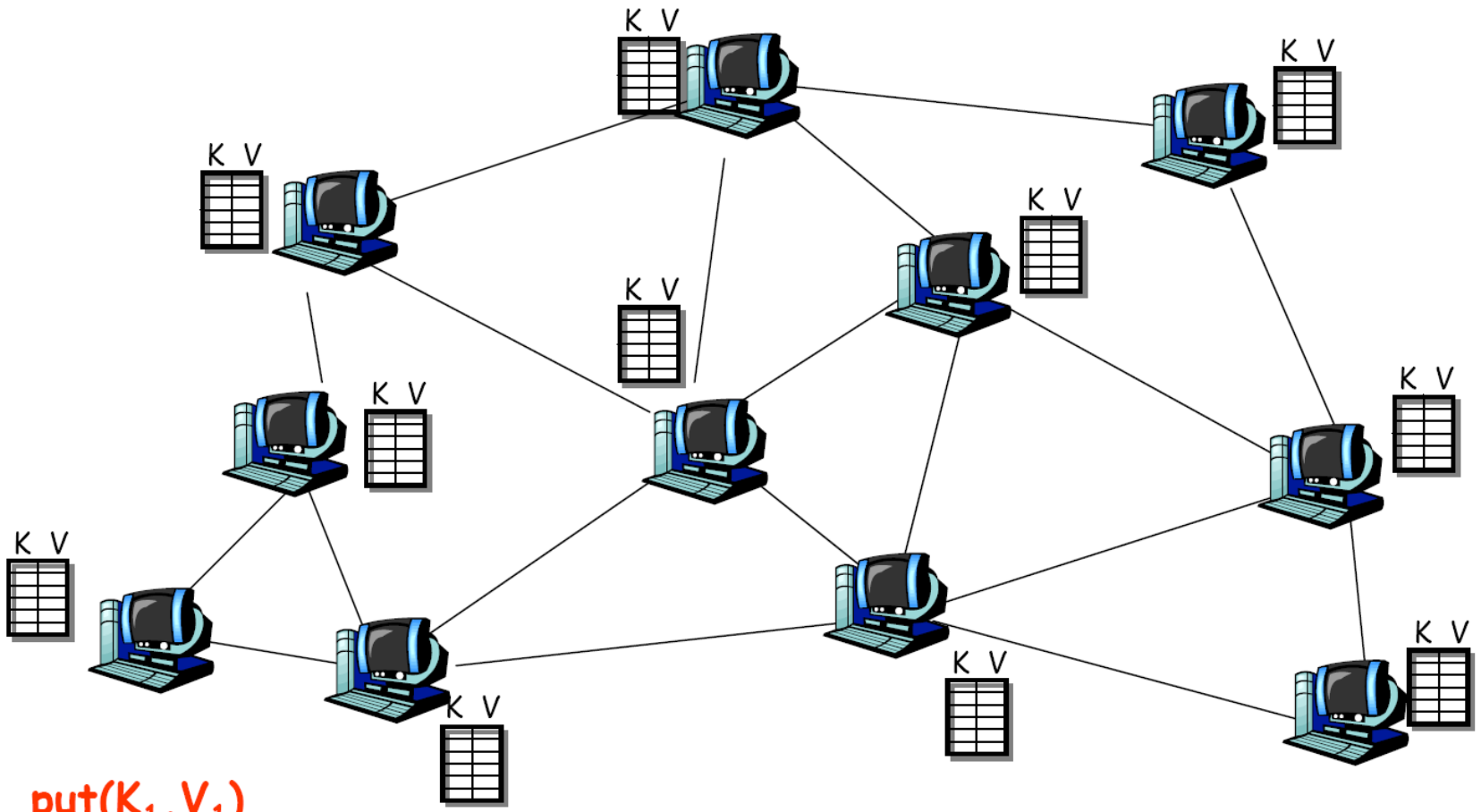
What is a DHT? (cont.)



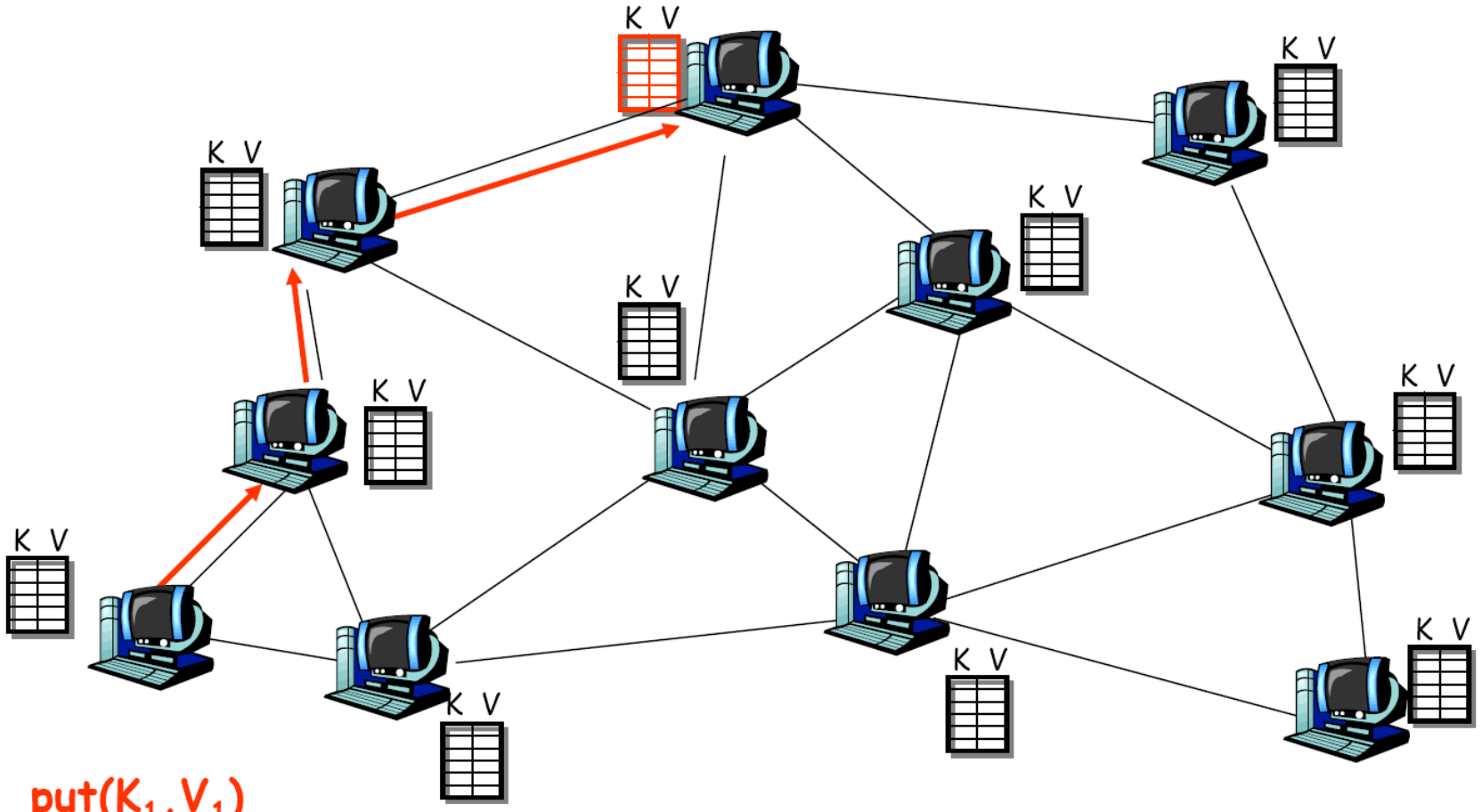
DHT in action



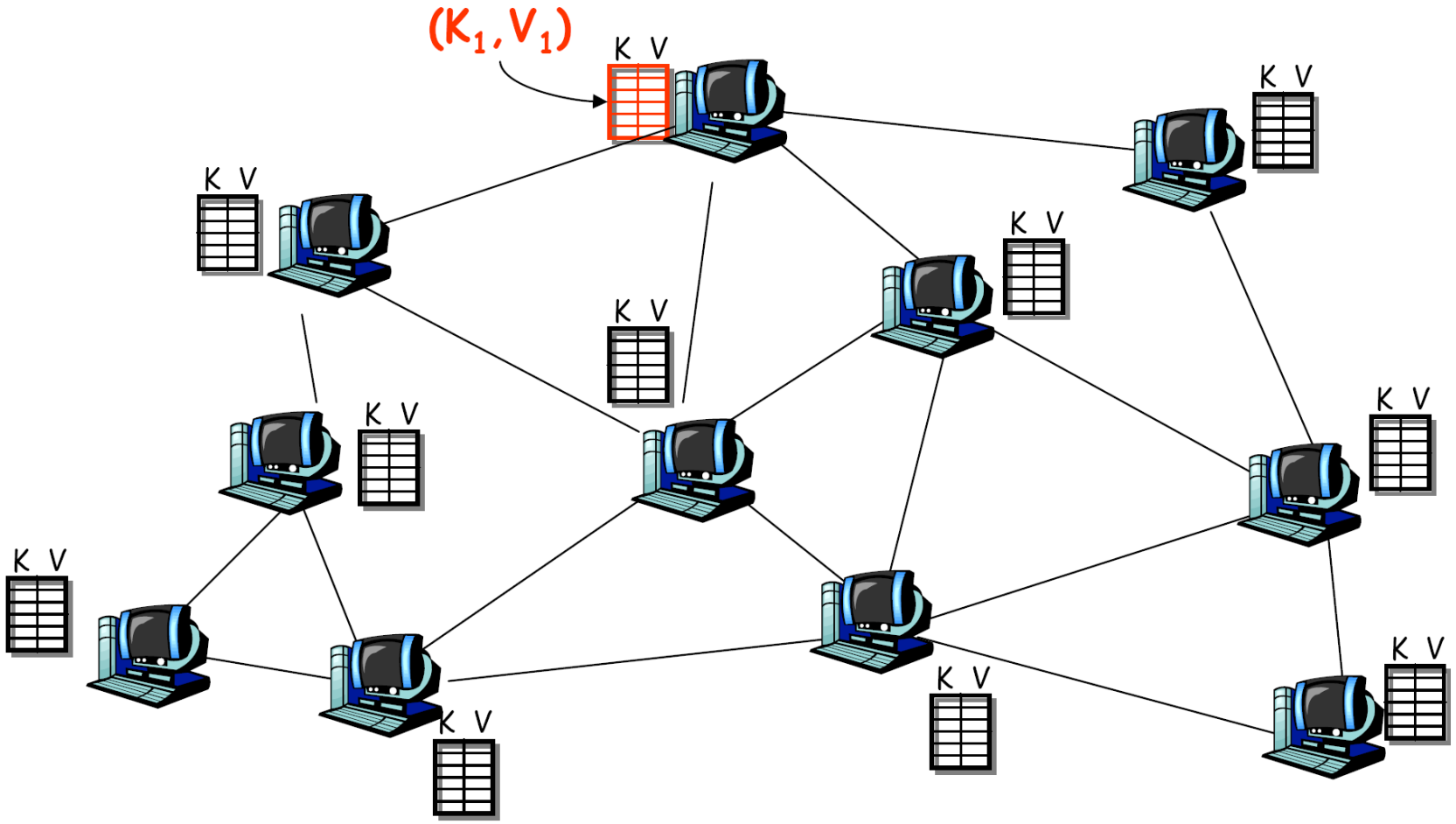
DHT in action: put()



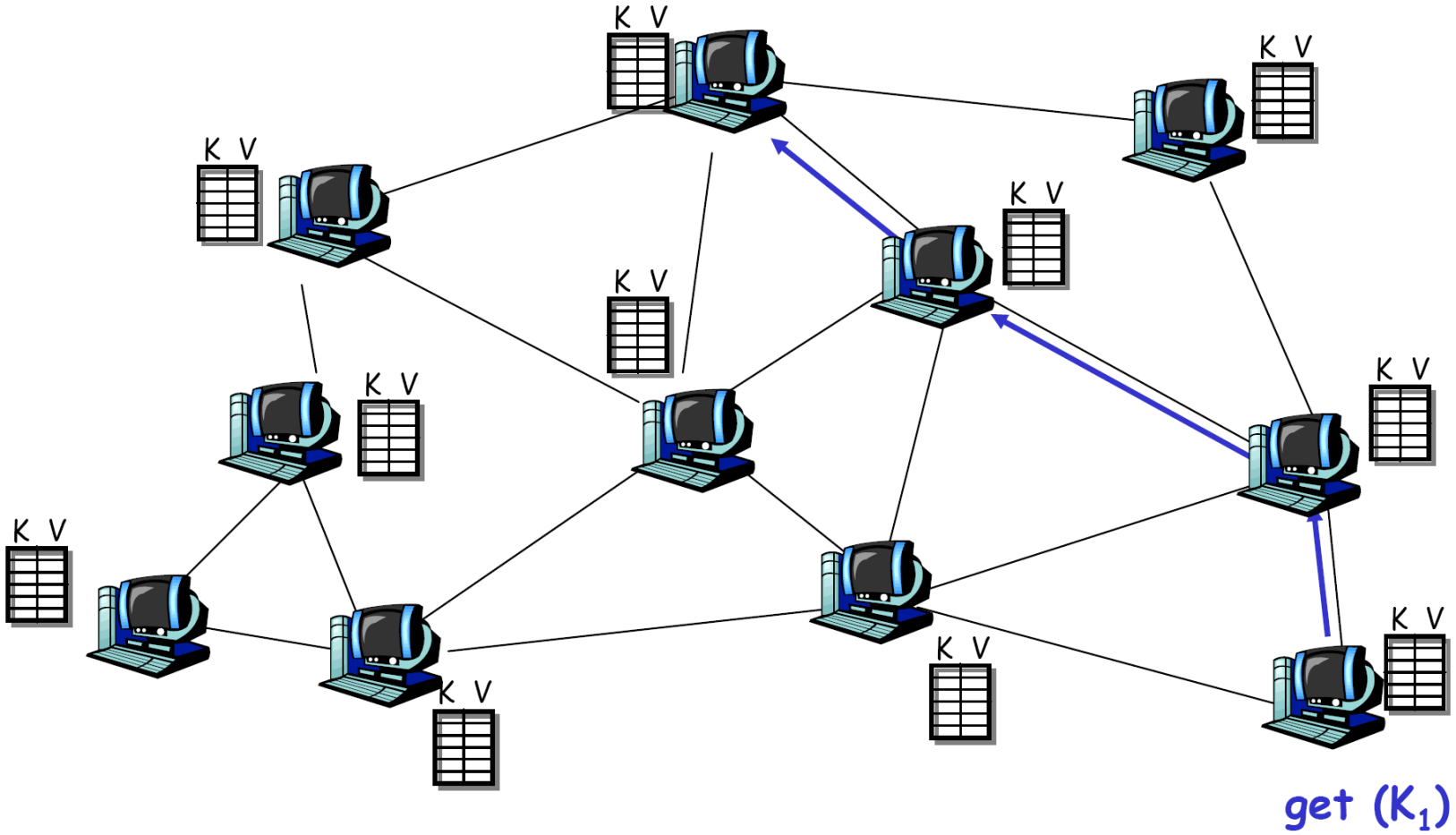
DHT in action: put()



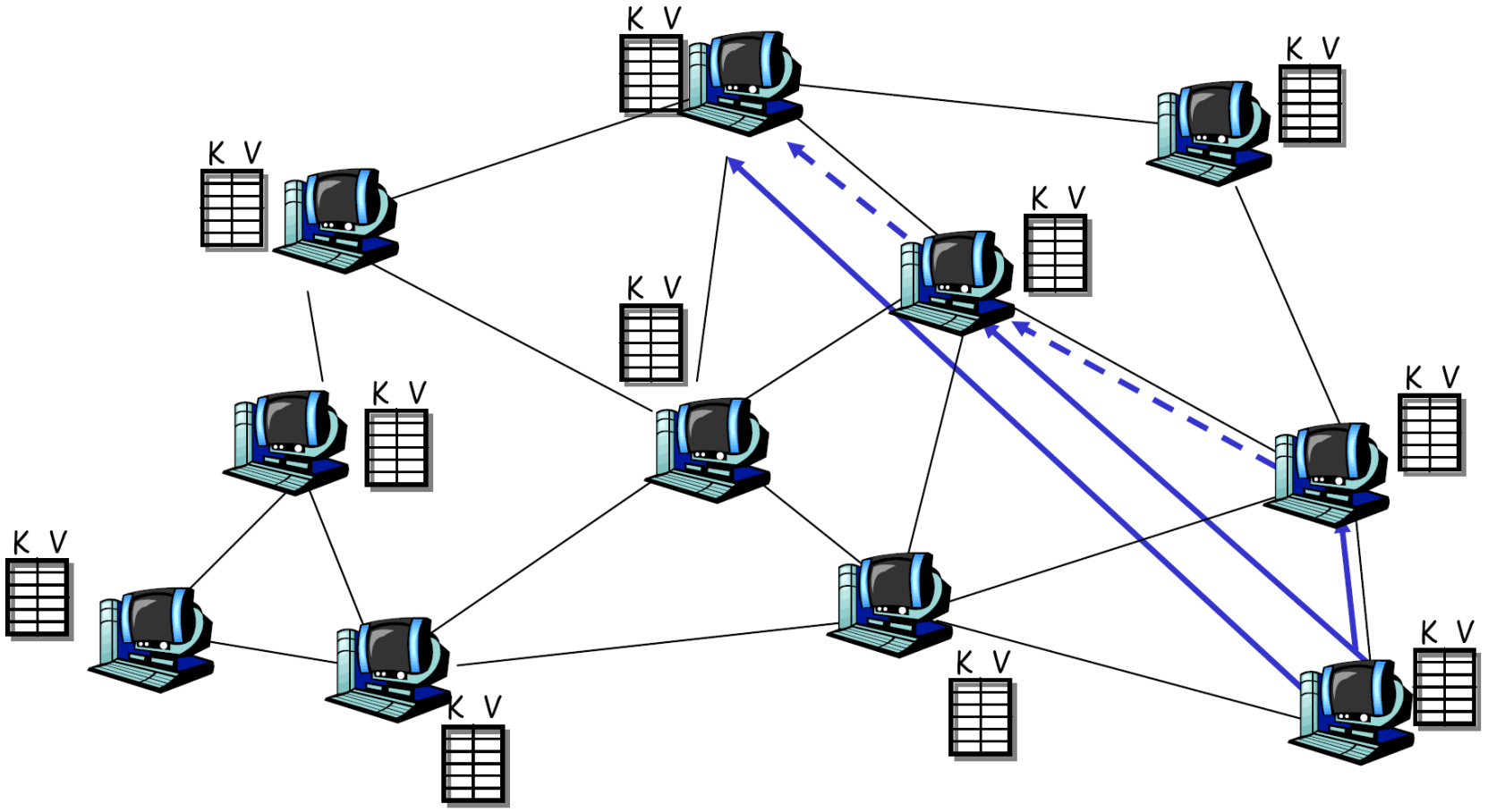
DHT in action: put()



DHT in action: get()



Iterative vs. Recursive Routing



Peers vs Infrastructure

- Peer Based DHT:
  - Application users provide nodes for DHT
  - Examples: file sharing, etc

- Infrastructure Based DHT
  - Set of managed nodes provide DHT service
  - Perhaps serve many applications

DHT Design Goals

- An "overlay" network with:
  - Decentralization and self-organization, i.e. no central authority, local routing decisions
  - Flexibility in mapping keys to physical nodes and routing
  - Robustness to joining/leaving
  - Scalability, i.e. low communication overhead
  - Efficiency, i.e. low latency
- A consistent "storage" mechanism with
  - No guarantees on persistence
  - Maintenance via soft state

Comparison to Other Facilities

Facility	Abstraction	Easy Use/Prg	Scalability	Load-Balance
DHT	high	high	high	yes
Centralized Lookup	medium	medium	low	no
P2P flooding queries	medium	high	low	no
Distributed FS	low	medium	medium	no

Facility	Fault-Tolerance	Self-Org	Admin
DHT	high	yes	low
Centralized Lookup	low	no	medium
P2P flooding queries	depends	yes	low
Distributed FS	medium	no	high

Outline

- General overview
- P2P research activities
  - Applications
  - Search
  - Security
  - Resource Management
- Case Studies

Security

- Protection against:
  - Routing attacks
    - Redirect queries in wrong direction or to non-existing nodes
    - Misleading updates
    - Partition
  - Storage/Retrieval attacks
    - Node responsible for holding data item does not store or deliver it as required
  - Inconsistent behavior
    - Node sometimes behaves and sometimes does not
  - Identity spoofing
    - Node claims to have an identity that belongs to another node
    - Node delivers bogus content
  - DoS attacks
    - Let legitimate (authenticated) users through
  - Rapid joins/leaves

- In P2P systems, security issues are raised by the presence of malicious peers
  - Trust assessment and management

Security

- Need for providing levels of:
  - Data availability
  - Privacy
  - Confidentiality
  - Integrity
  - Authenticity
- Challenges:
  - Secure storage
  - Secure routing
  - Access control, authentication and identity management
  - Anonymity
  - Reputation management

Security (cont.)

- Secure storage: cryptographic algorithms and protocols for securing the published and stored content
  - Self-certifying data (integrity)
  - Information Dispersal (availability)
  - SHA (confidentiality and availability)

Security (cont.)

- Secure routing: relies on 3 primitives
  - Secure assignment of IDs to nodes
  - Secure maintenance of routing tables
  - Secure forwarding of messages

Security (cont.)

- Access control, authentication and identity management: often ignored but essential to counter attacks like Sybil attacks, denial of service, etc.
  - Central authentication/identification authority
  - Access keys and author-based access control list
  - Intellectual property and digital rights management

Anonymity

- The degree to which a system allows for anonymous transactions
- Do not want someone to identify author, publisher, reader, server, or document on the systems

Project	Types and techniques of Anonymity			
	Publisher	Reader	Server	Document
Gnutella	Multicasting, covert paths	N/A	N/A	N/A
Freenet	covert paths, identity spoofing	covert paths	Non-voluntary placement	encryption
APFS	covert paths	covert paths	N/A	N/A
FreeHaven	Covert paths (remailer)	covert paths	broadcast	Encryprion/split files into shares
Publius	Covert paths (remailer)	N/A	Non-voluntary placement	Encryption/split key
PAST	N/A	N/A	Non-voluntary placement	encryption

Outline

- General overview
- P2P research activities
  - Applications
  - Search
  - Security
  - Resource Management
- Case Studies

Resource Management

- Focus here is on p2p content distribution systems

- Main resources to be managed:
  - Content
  - Storage capacity
  - Bandwidth

Resource Management (cont.)

- Content management: deletion, update and versioning
  - Often not supported for security, robustness to attacks, lack of synchronization between peers
  - Update and deletion provided to publishers (Publius)
  - Complex content history archival (OceanStore)

Resource Management (cont.)

- Storage capacity management: content expiration, storage vs. contribution
  - Contracts with duration in FreeHaven; expired content is evicted
  - Used capacity storage in exchange for compensation (MojoNation)
  - Peer contribution in exchange to the used disk space (PAST)

Resource Management (cont.)

- Bandwidth management: routing performance, search facility
  - Structured systems for an optimized network overhead
  - Indirect files for an optimized network overhead in key-word based search
    - Indirect files point to regular files

Outline

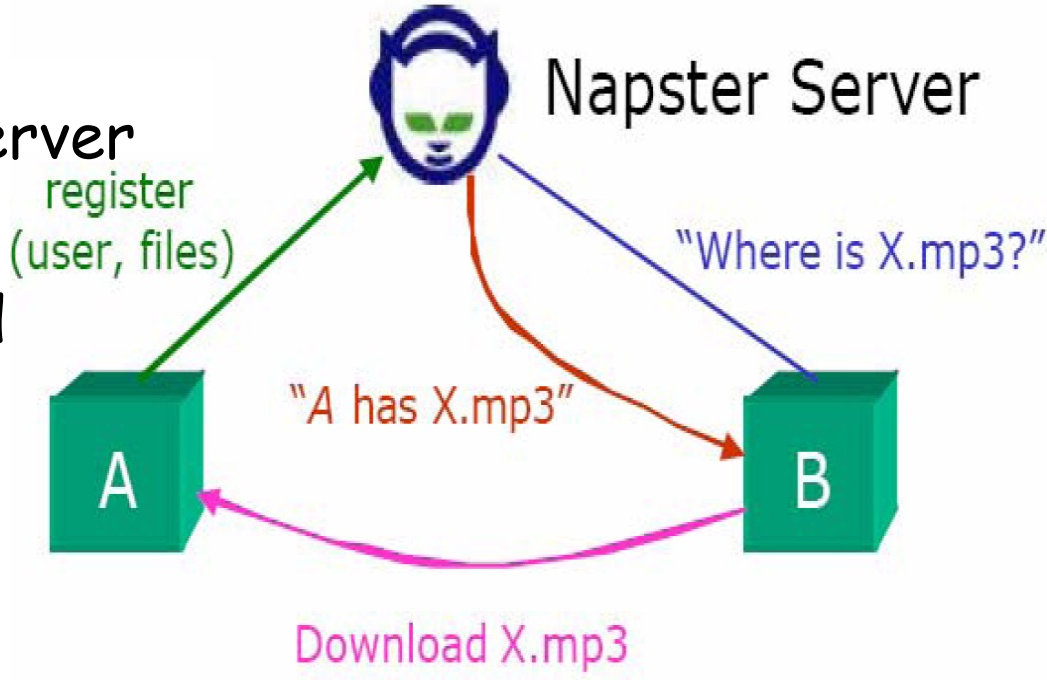
- General overview
- P2P research activities
- Case Studies
  - File-sharing systems
  - Distributed Hash Tables (DHT)
  - Distributed Computing
  - Trust and reputation management

Outline

- General overview
- P2P research activities
- Case Studies
  - File-sharing systems
    - Napster
    - Gnutella
    - Freenet
    - KazaA
  - Distributed Hash Tables (DHT)
  - Distributed Computing
  - Trust and reputation management

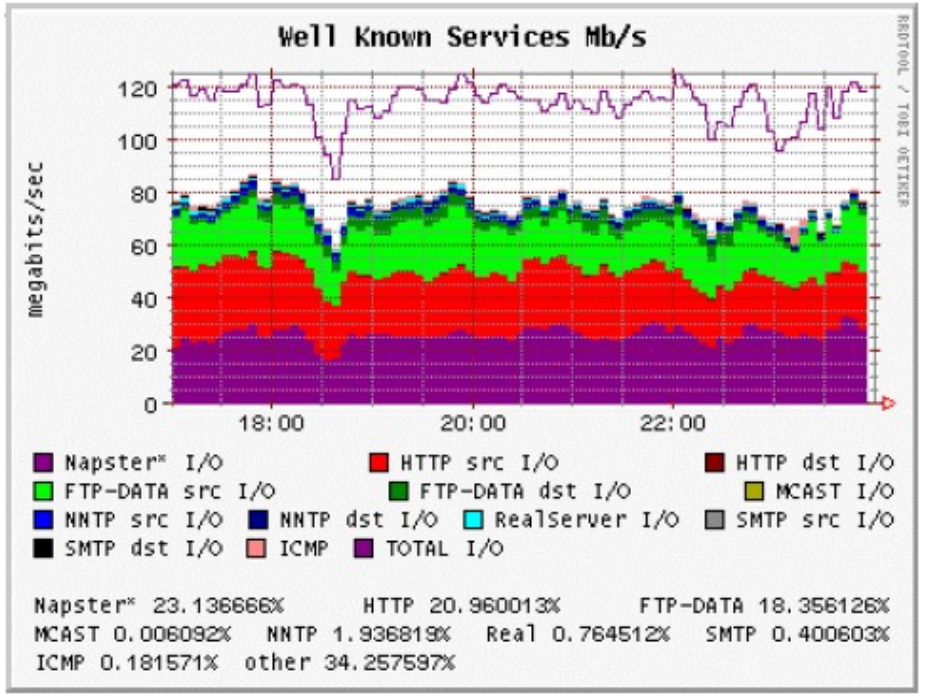
Napster

- Hybrid decentralized, unstructured
- Combination of client/server and P2P approaches
- A network of registered users running a client software, and a central directory server
- The server maintains 3 tables:
  - (File_Index, File_Metadata)
  - (User_ID, User_Info)
  - (User_ID, File_Index)



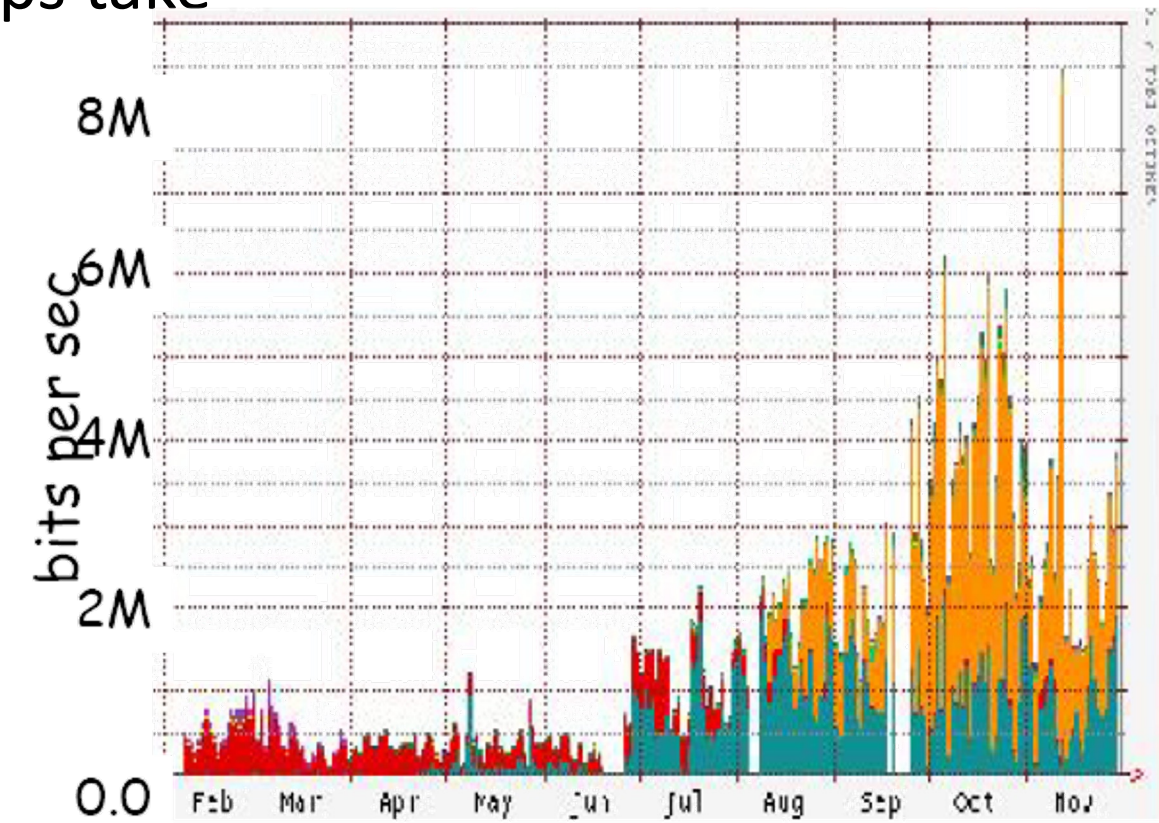
Napster (cont.)

- History
  - 5/99: Shawn Fanning (freshman, Northeasten U.) founds Napster Online music service
  - 12/99: first lawsuit
  - 3/00: 25% UWisc traffic Napster
  - 2/01: US Circuit Court of Appeals: Napster knew users violating copyright laws
  - 7/01: # simultaneous online users: Napster 160K, Gnutella: 40K, Morpheus (KaZaA): 300K



Napster (cont.)

- judge orders Napster to pull plug in July ’01
- other file sharing apps take over!



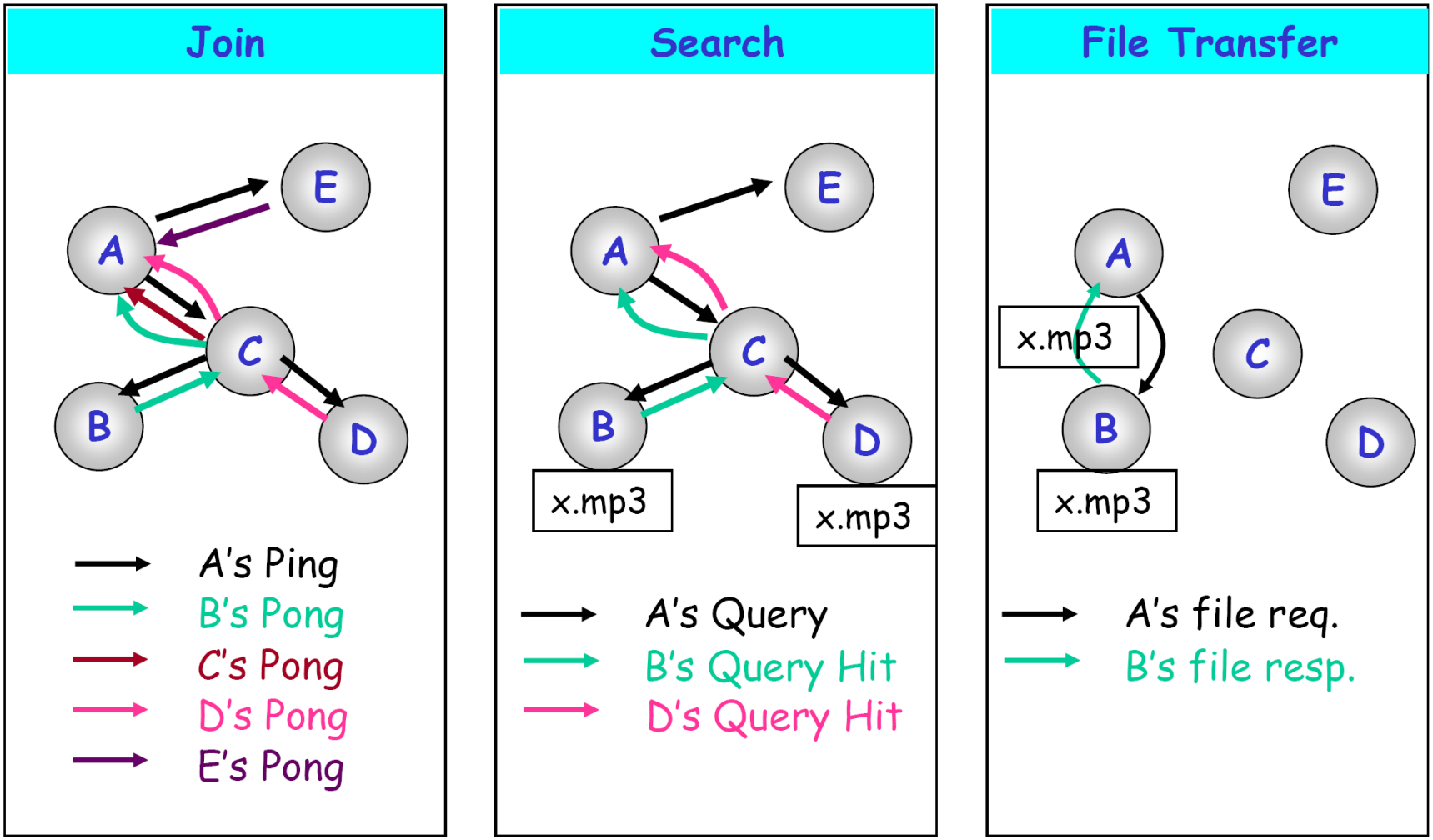
Gnutella

- Pure decentralized, unstructured
- Characteristic:
  - Few nodes with high connectivity.
  - Most nodes with sparse connectivity.
- Goal: distributed and anonymous file sharing
- Each application instance (node) :
  - stores/serves files
  - routes queries to its neighbors
  - responds to request queries

Gnutella (cont.)

- History:
  - 3/14/00: released by AOL, almost immediately withdrawn
  - Became open source
  - Many iterations to fix poor initial design (poor design turned many people off)
- Issues
  - How much traffic does one query generate?
  - How many hosts can it support at once?
  - What is the latency associated with querying?
  - Is there a bottleneck?

Gnutella (cont.)



Gnutella (cont.)

- Advantages:
  - Robustness to random node failure
  - Completeness (constrained by the TTL)
- Disadvantages:
  - Communication overhead
  - Network partition (controlled flooding)
  - Security
- Free riding problem
- Tradeoff:
  - Low TTL => low communication overhead
  - High TTL => high search horizon

Freenet

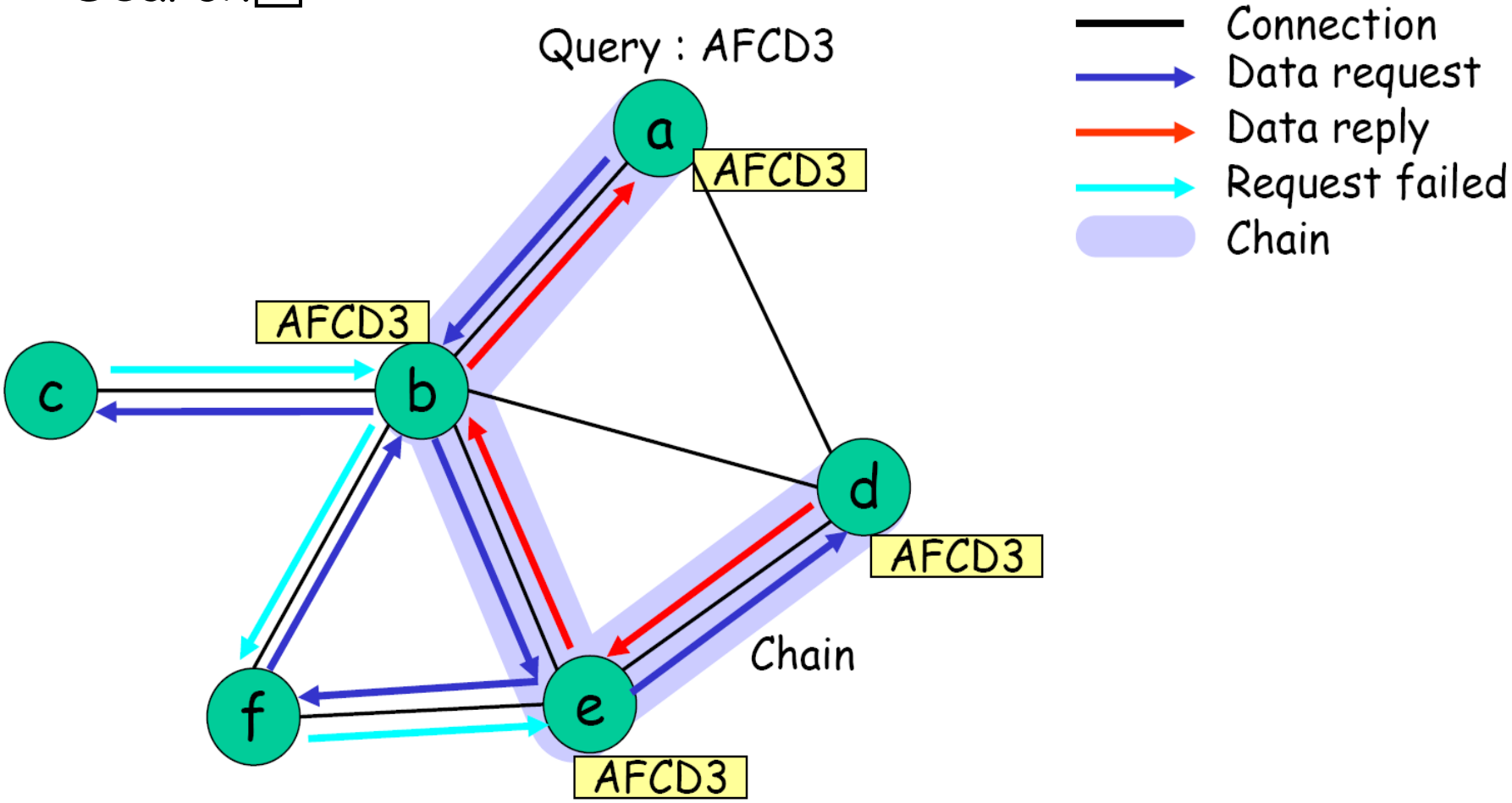
- Purely decentralized, loosely structured
- Goal: provide anonymous method for storing and retrieving data (files)
- 2 operations:
  - Insertion
  - Search
- Indexes (keys) are used for:
  - file clustering
  - Assisting in routing
  - Search optimization
- Each node maintains local data-store (files with similar keys) and routing table

Freenet (cont.)

- Bootstrapping
  - No explicit solution
- File insertion
  - User assigns a hash key to file
  - sends an insert message to the user's own node
  - Node checks its data store for collision
  - Node looks up the closest key and forwards the message to another node
  - This is done recursively until TTL expires or collision detected.
  - If no collision, user sends data down the path established by the insert message.
  - Each node along the path stores it and creates a routing table entry

Freenet (cont.)

- Search

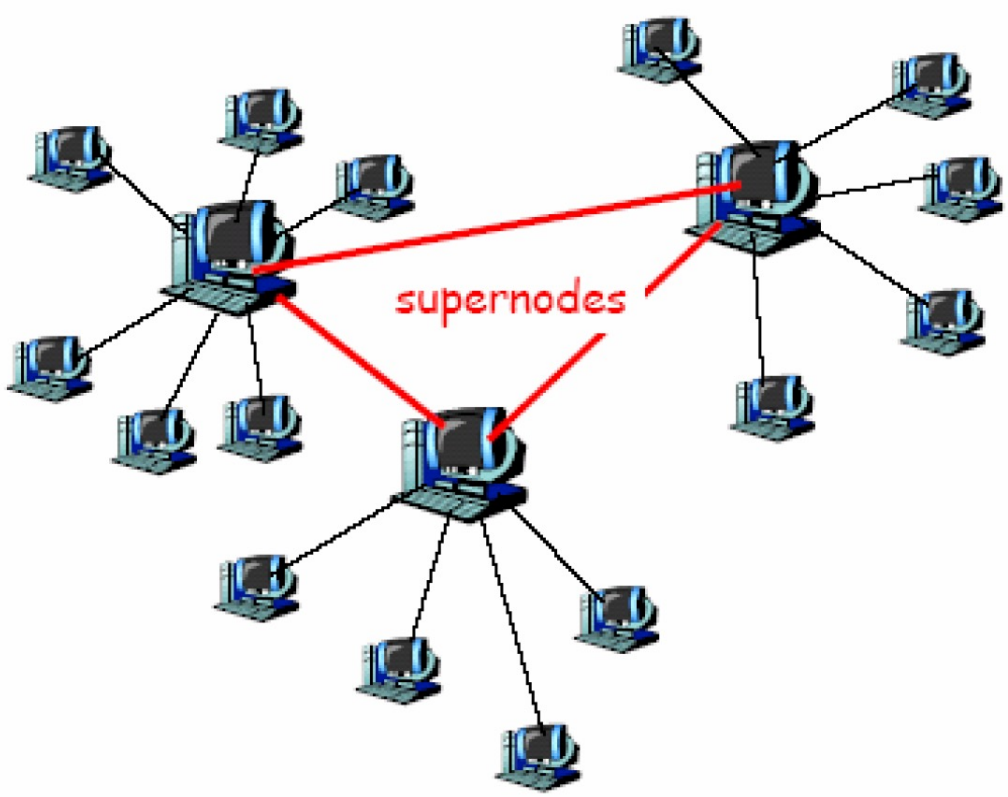


Freenet (cont.)

- Advantages:
  - Anonymity
  - Robustness to random node failure
  - Low communication overhead
  - More scalable
  - Self-organized
- Disadvantages:
  - Poor routing decision
  - Spam
  - Security

KaZaA

- Partially centralized, unstructured
- Every peer is either a supernode(SN) or an ordinary node (ON) assigned to a supernode
- Each supernode knows where the other supernodes are (Mesh structure)



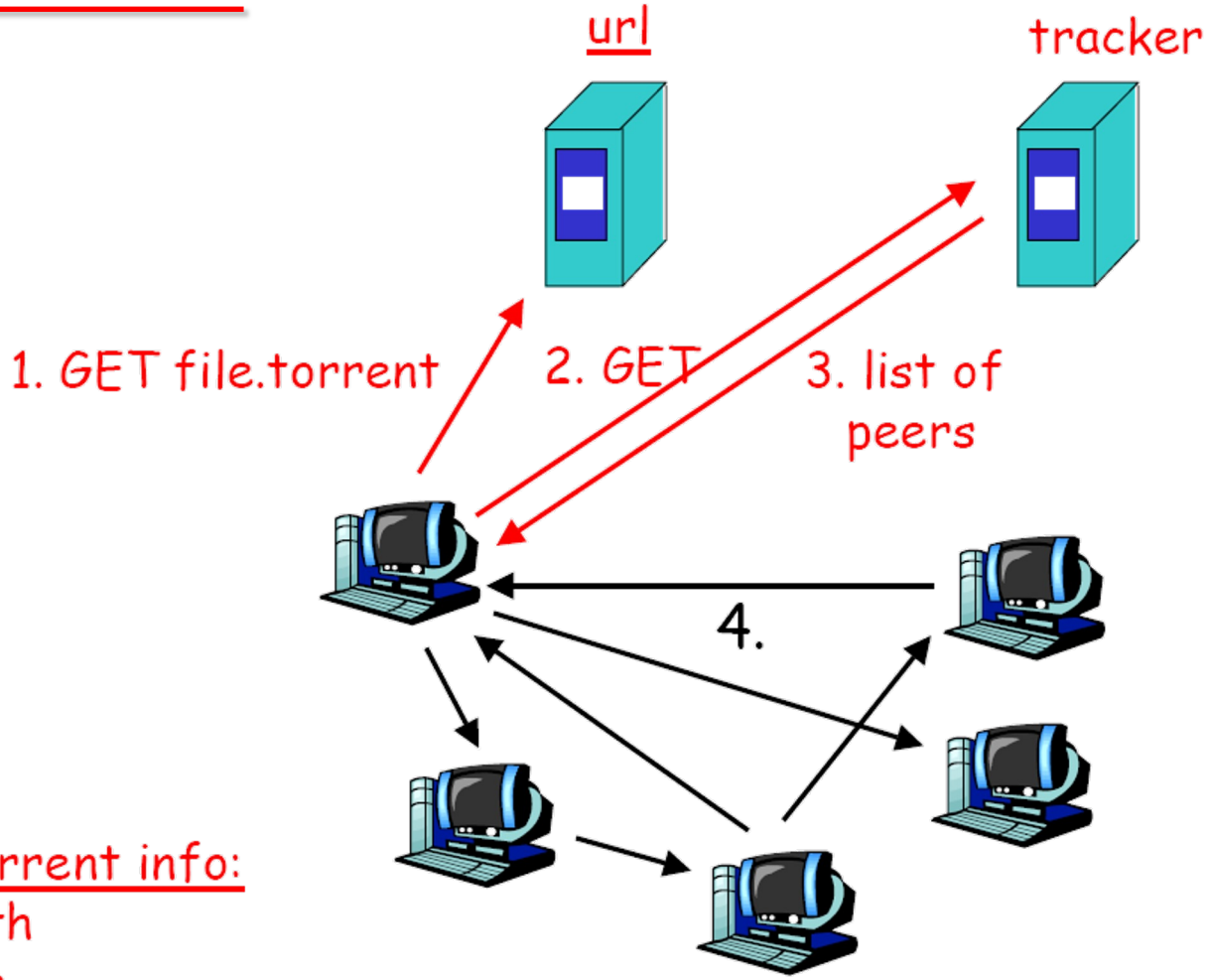
KaZaA (cont.)

- Bootstrap:
  - Connection to a known SN
  - Upload METADATA for shared files
    - file name
    - file size
    - file descriptor
    - ContentHash
- Search:
  - ON -> SN (-> SN)*
  - Keyword-based
  - ContentHash in result
- Download:
  - HTTP
  - ContentHash-based
  - Failure -> automatic search on ContentHash□

KaZaA (cont.)

- Advantages:
  - Scalability
  - Efficiency
  - Exploits heterogeneity of peer
  - Fault-tolerance
- Disadvantages
  - Pollution
  - DoS attacks on SN

BitTorrent



file.torrent info:

- length
- name
- hash
- url of tracker

BitTorrent (cont.)

- File is broken into pieces
  - Typically piece is 256 KBytes
  - Upload pieces while downloading pieces
- Piece selection
  - Select rarest piece
  - Except at beginning, select random pieces
- Tit-for-tat
  - Bit-torrent uploads to at most four peers
  - Among the uploaders, upload to the four that are downloading to you at the highest rates
  - A little randomness too, for probing

Outline

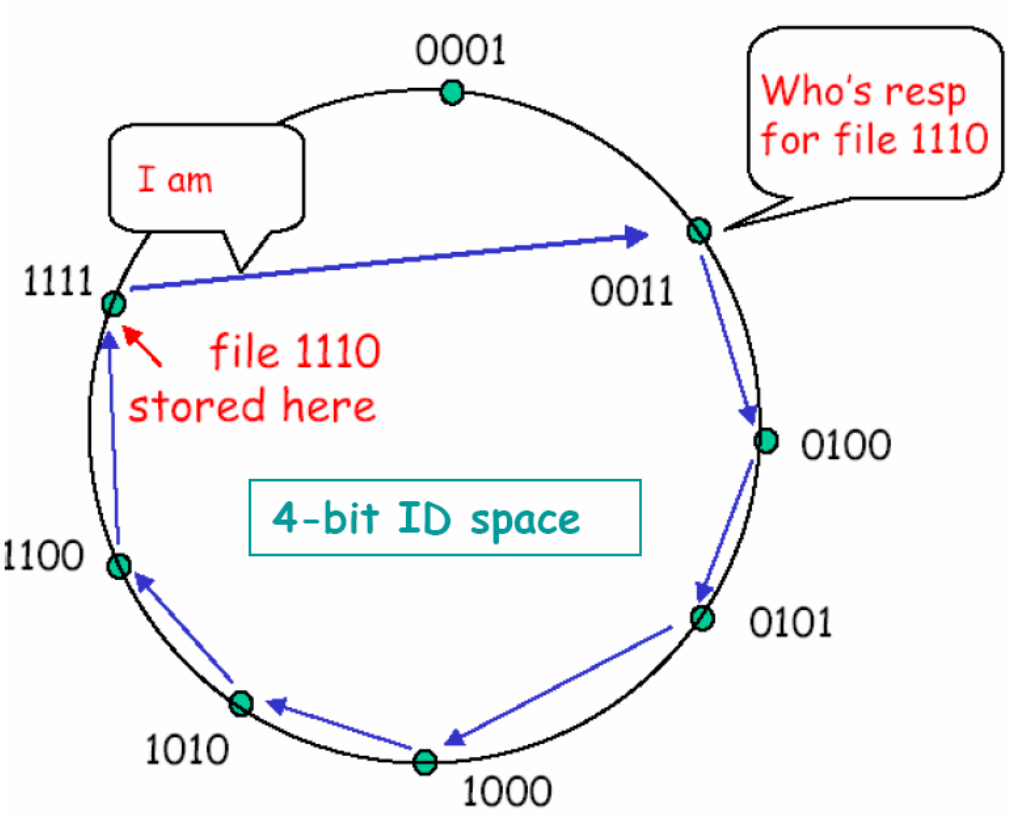
- General overview
- P2P research activities
- Case Studies
  - File-sharing systems
  - Distributed Hash Tables (DHT)
    - Consistent hashing
    - Kademlia
    - Analysis of DHT systems
  - Distributed Computing
  - Trust and reputation management

DHT Approaches

- Consistent Hashing
- CAN (Content Addressable Network)
- Kademlia

Consistent Hashing

- Consistent hashing [Karger97]
- Overlay network is a circle
- Each node has randomly chosen id
  - Example: hash on the IP address
  - Keys in same id space
- Node's successor in circle is node with next largest id
  - Each node knows IP address of its successor
- Key is stored in closest successor



Consistent Hashing (cont.)

- Principles:

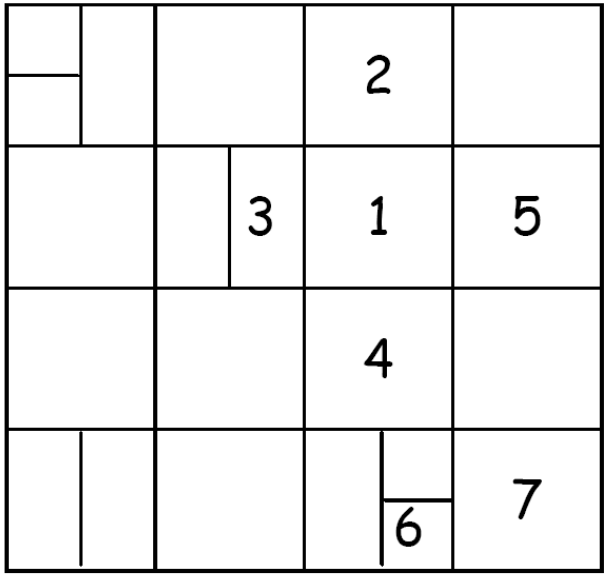
Node departures	Node joins
Each node must track $s \geq 2$ successors If your successor leaves, take next one Ask your new successor for list of its successors; update your s successors	You are a new node, id k Ask any node n to find the node n' successor for id k Get successor list from n' Tell your predecessors to update their successor lists Thus, each node must track its predecessor

- # neighbors = s+1: O(1)
  - Routing table: (neighbor_id, neighbor_ip@)
- Average # of messages to find key:

Can we do better?

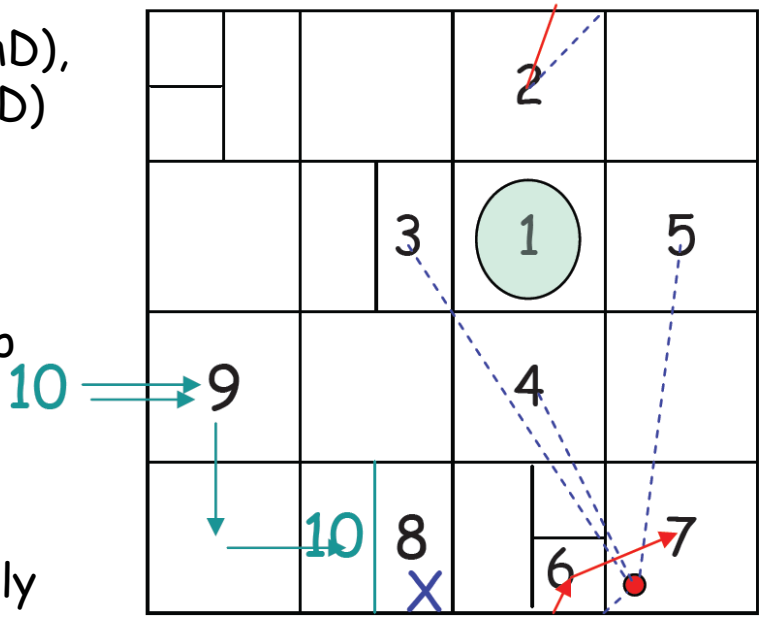
CAN

- Routing geometry: Hypercube
- A hash value = a point in a D-dimensional Cartesian space
- Each node responsible of a D-dimensional cube
- Neighbors are nodes that "touch" in more than a point
  - Example: D=2
    - 2,3,4,5 are 1's neighbors
    - 6 is a neighbor of 2
  - # neighbors: 0(D)



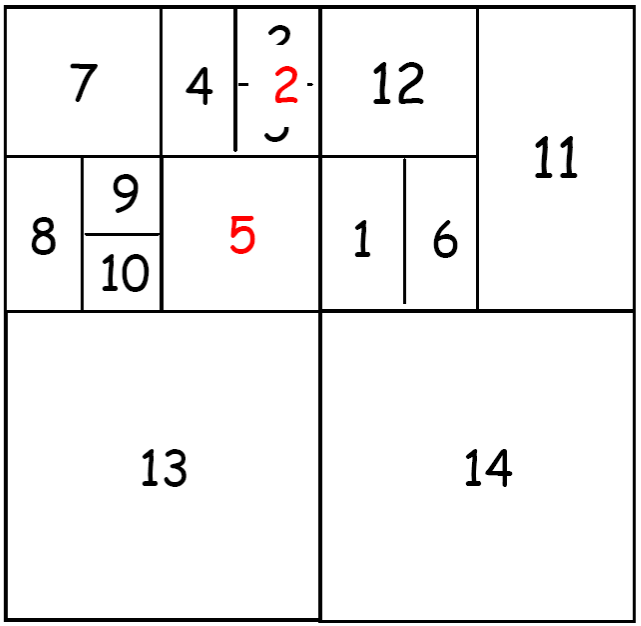
CAN (cont.)

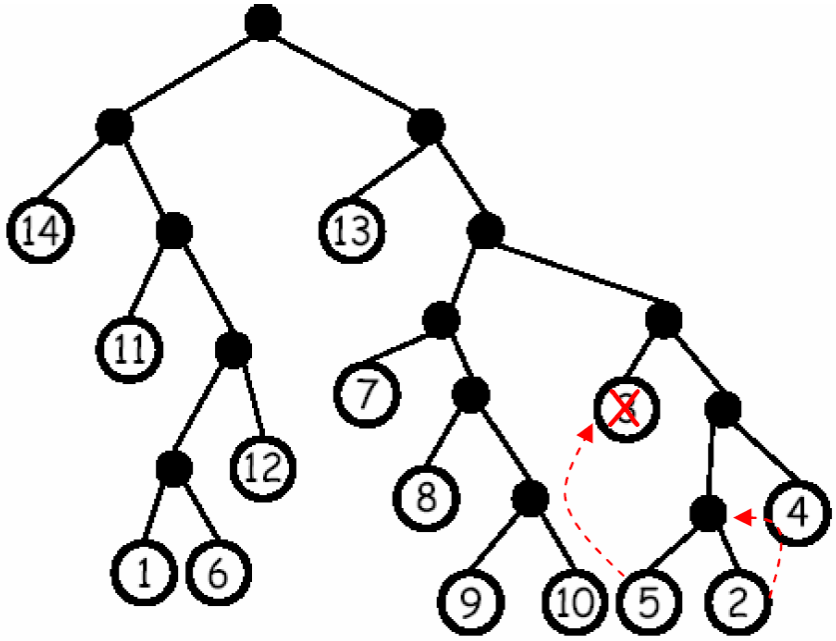
- Routing:
  - Recursively, from (n1, …, nD) to (m1, …, mD), choose the closest neighbor to (m1, …, mD)
  - expected # overlay hops: $0(DN^{1/D})/4$
- Node Join:
  - find some node in the CAN (via bootstrap process) (9)
  - choose a point in the space uniformly at random (X)
  - using CAN, inform the node that currently covers the space (8)
  - that node (8)splits its space in half
    - 1st split along 1st dimension, if last split along dimension i< D, next split along i+1st dimension
  - keeps half the space and gives other half to joining node



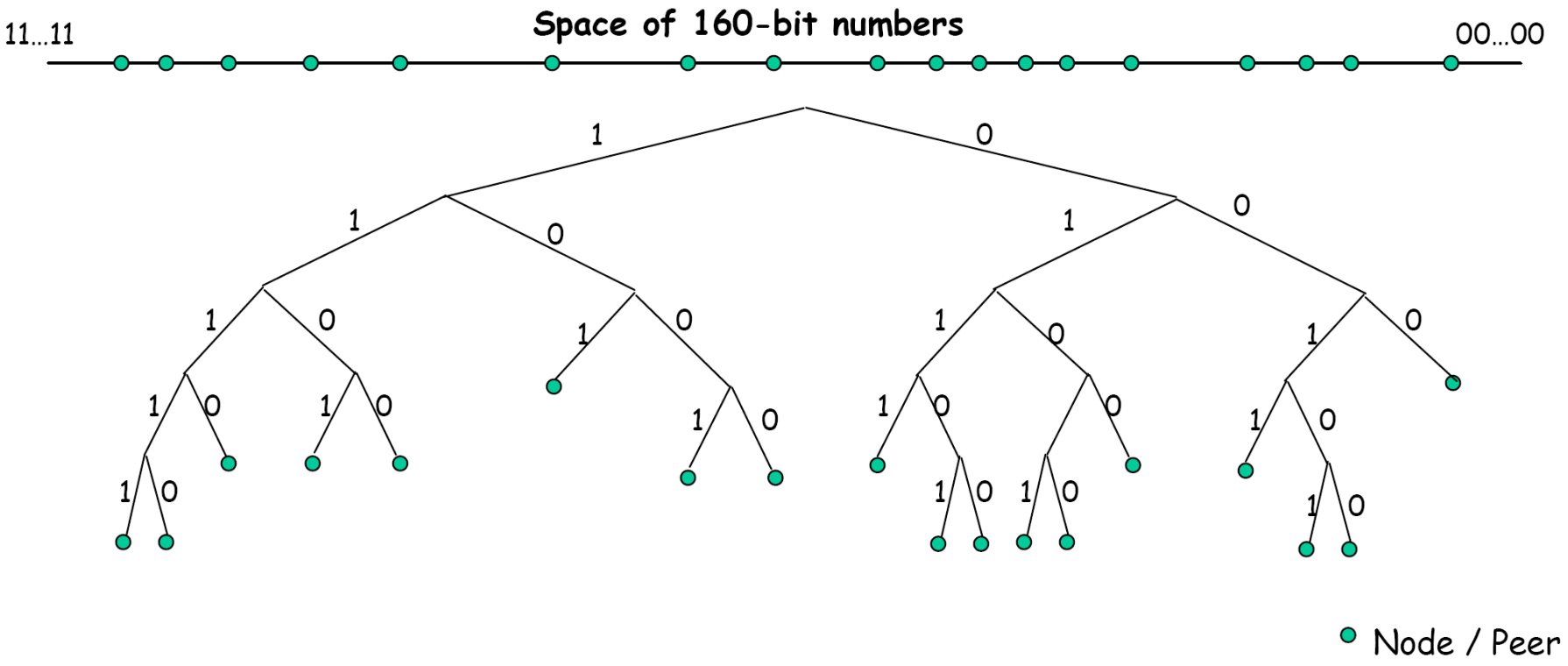
CAN (cont.)

- Node Leave:
  - leaf (3) removed
  - Find a leaf node that is either
    - a sibling
    - descendant of a sibling where its sibling is also a leaf node (5)
  - (5) takes over (3)'s region (moves to (3)'s position on the tree)
  - (5)'s sibling (2)takes over (5)'s previous region
  - The cube structure remains intact



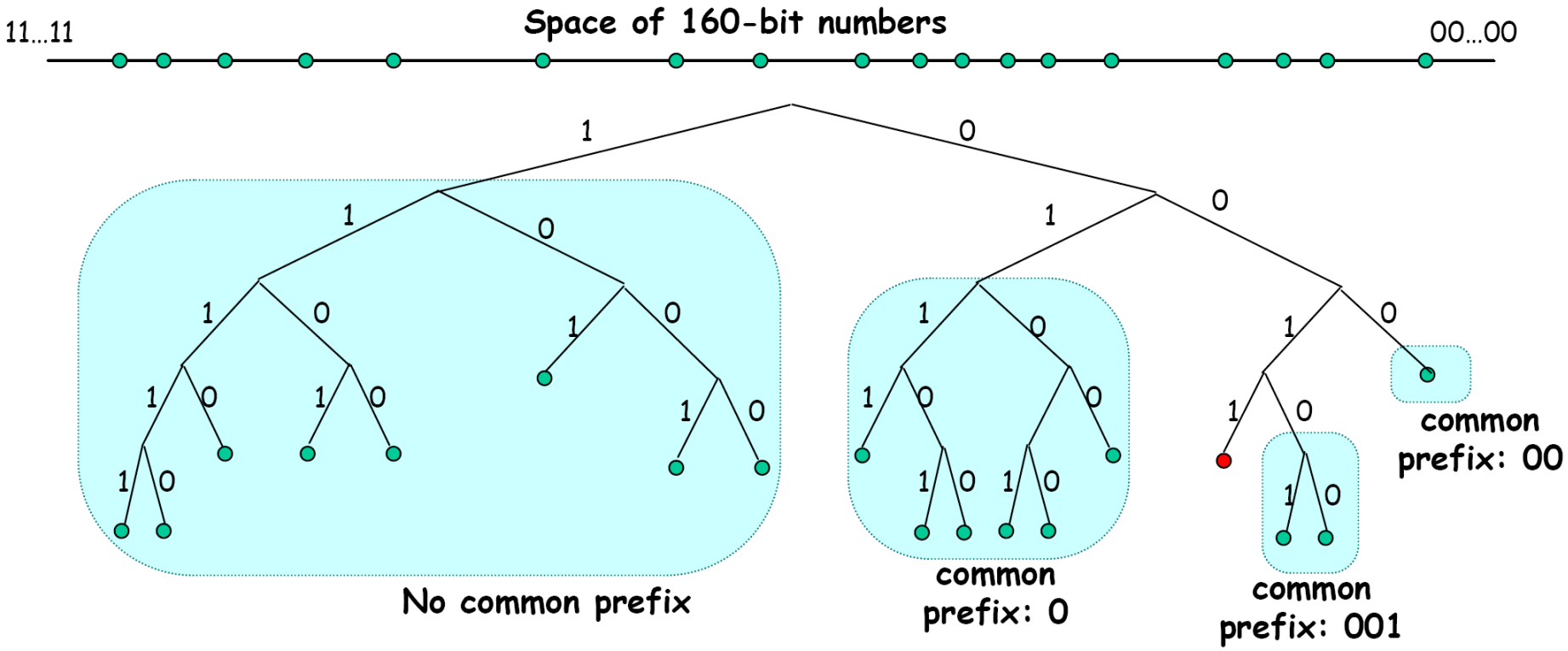


Kademlia



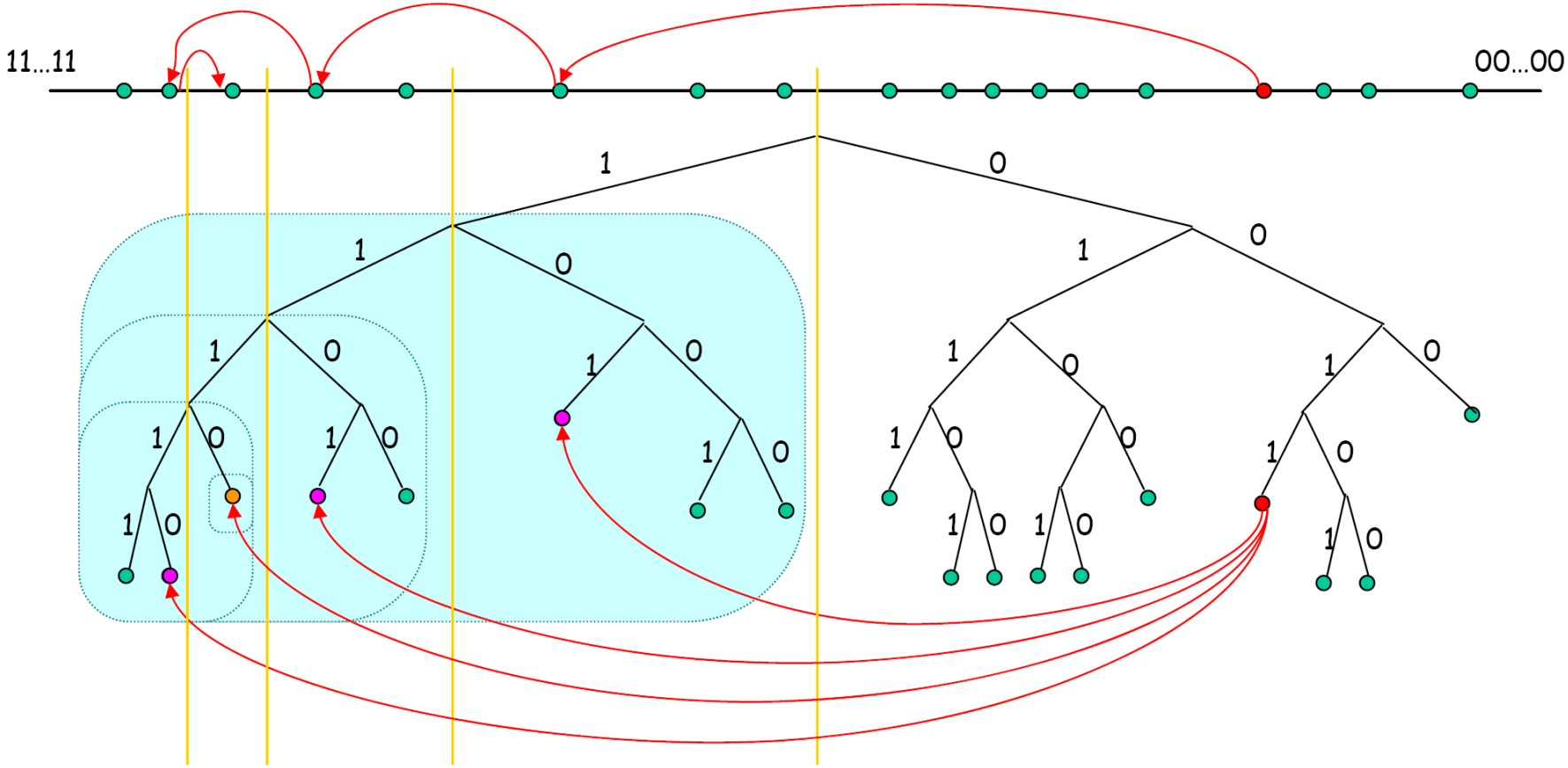
- Nodes are treated as leaves in a binary tree
- Node's position in the tree is determined by the shortest unique prefix of its ID
- A Node is responsible for all "closest" IDs, i.e. IDs having same prefix as itself
- Distance between ID x and y is measured as d(x,y) = x $\oplus$y
  - e.g. d(010101_b, 110001_b) = $100100_b \Leftrightarrow$ d(21_{10}, 49_{10}) = 36_{10}
  - Nodes/IDs tree (i.e. with longest common prefix) are closer

Kademlia: Index distribution



- For any node (say the red node with prefix **0011**) the binary tree is divided into a series of maximal subtrees that do not contain the node.
- A node must know at least one node in each of these subtrees.

Kademlia: Query Routing



- Consider a query for ID 1110… initiated by node 00111…

DHT Challenges

- Consistency
  - Soft-state publication
  - Tradeoff between consistency and communication overhead
- Performance
  - Load-balancing
  - Query Response time (optimize # of hops)

  => Replication?
- Reduce latency in routing
  - Neighbors in a ring and far away in the internet

  => Location-awareness
- Geometry
  - Ring topology adds flexibility and helps in resilience
- Bootstrapping
  - Relies on known node

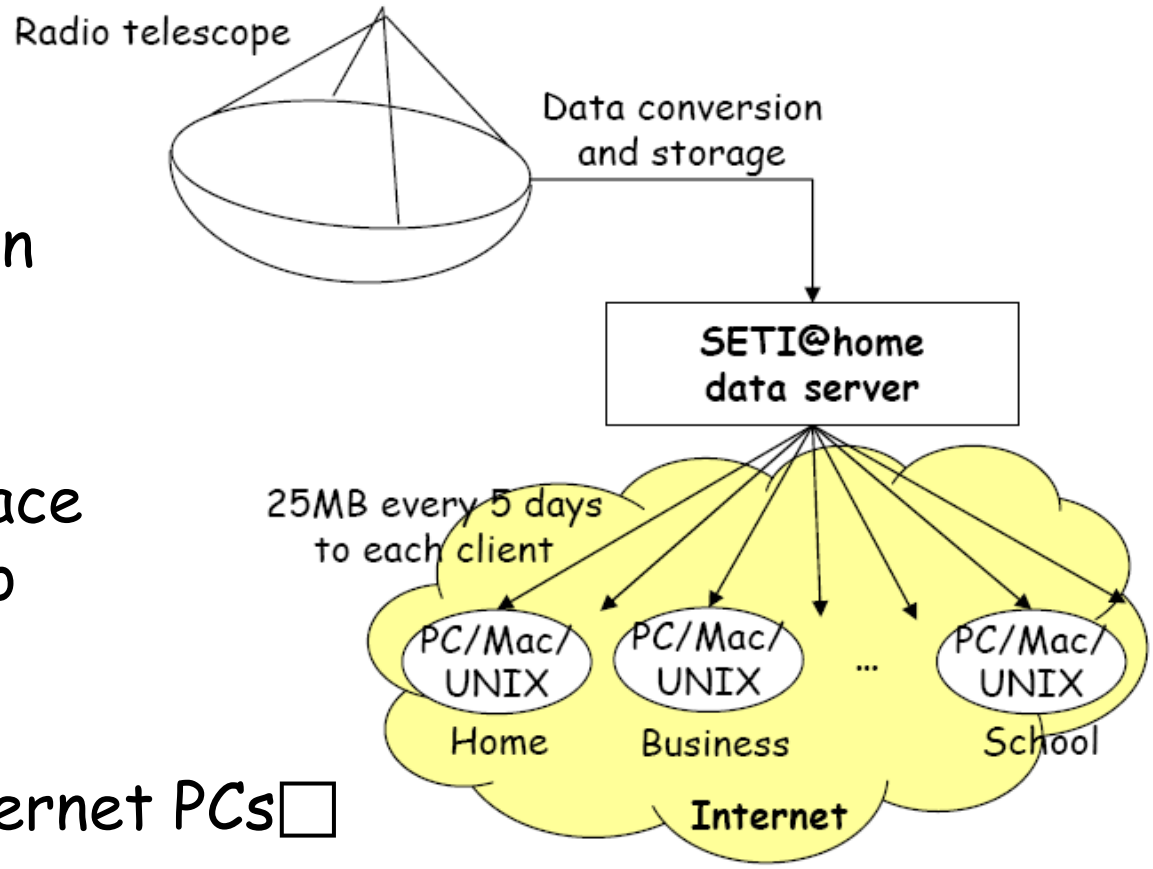
DHT Applications

- Global file systems [OceanStore, CFS, PAST, Pastiche, UsenetDHT]
- naming services [Chord-DNS, Twine, SFR]
- DB query processing [PIER, Wisc]
- Internet-scale data structures [PHT, Cone, SkipGraphs]
- Communication services [i3, MCAN, Bayeux]
- File sharing [OverNet, eMule, eDonkey, etc.]
- Event notification [Scribe, Herald]

Outline

- General overview
- P2P research activities
- Case Studies
  - File-sharing systems
  - Distributed Hash Tables (DHT)
  - Distributed Computing
  - Trust and reputation management
  - Measurements of existing P2P systems
- Future research directions

seti@home

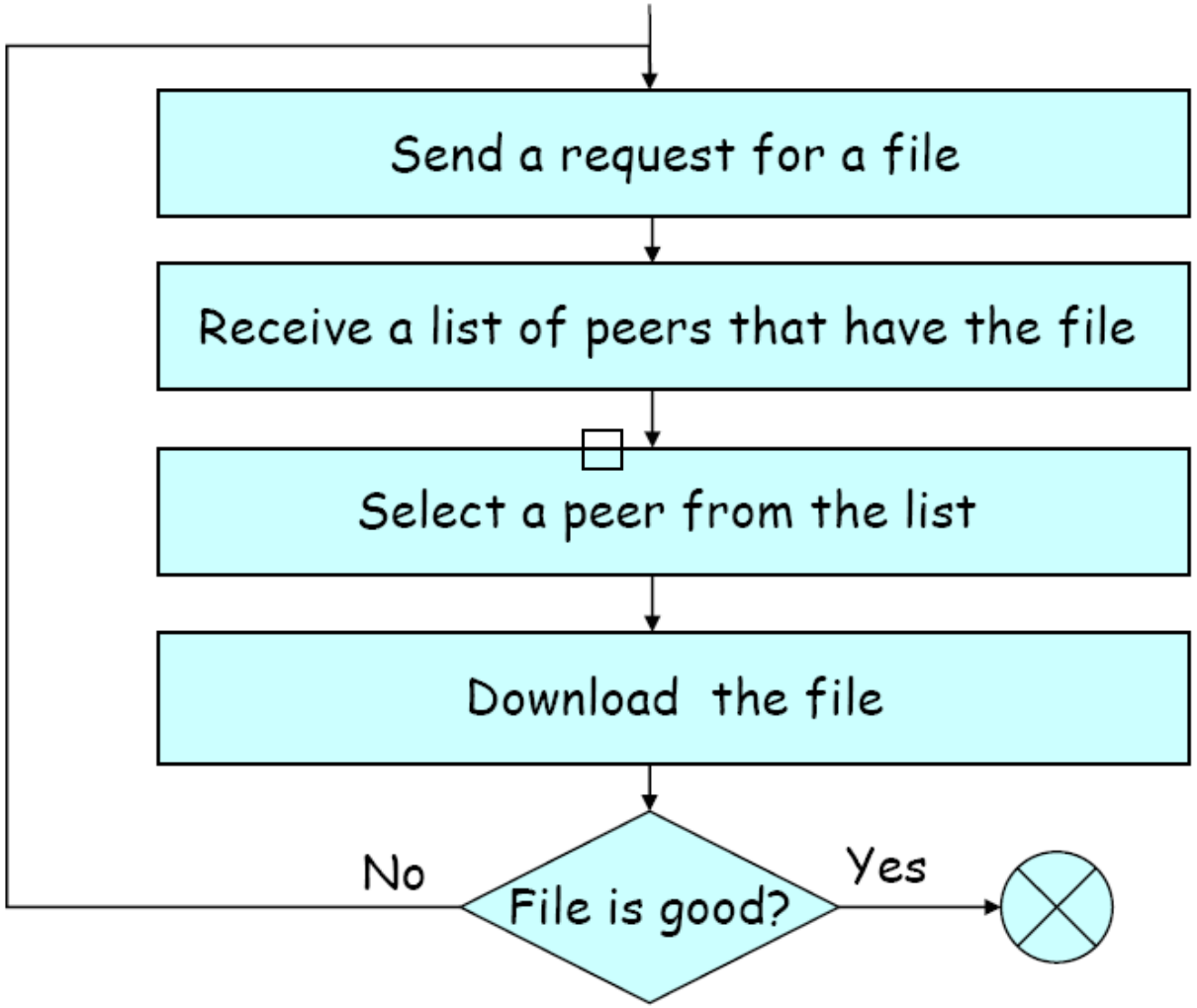


- Goal : to discover alien civilizations
- Analyzes the radio emission from the space collected by the radio telescopes using processing power of millions of unused internet PCs
- Two major components : a database server and clients
- Supported platforms : Windows, Linux, Solaris, and HP-UX

Outline

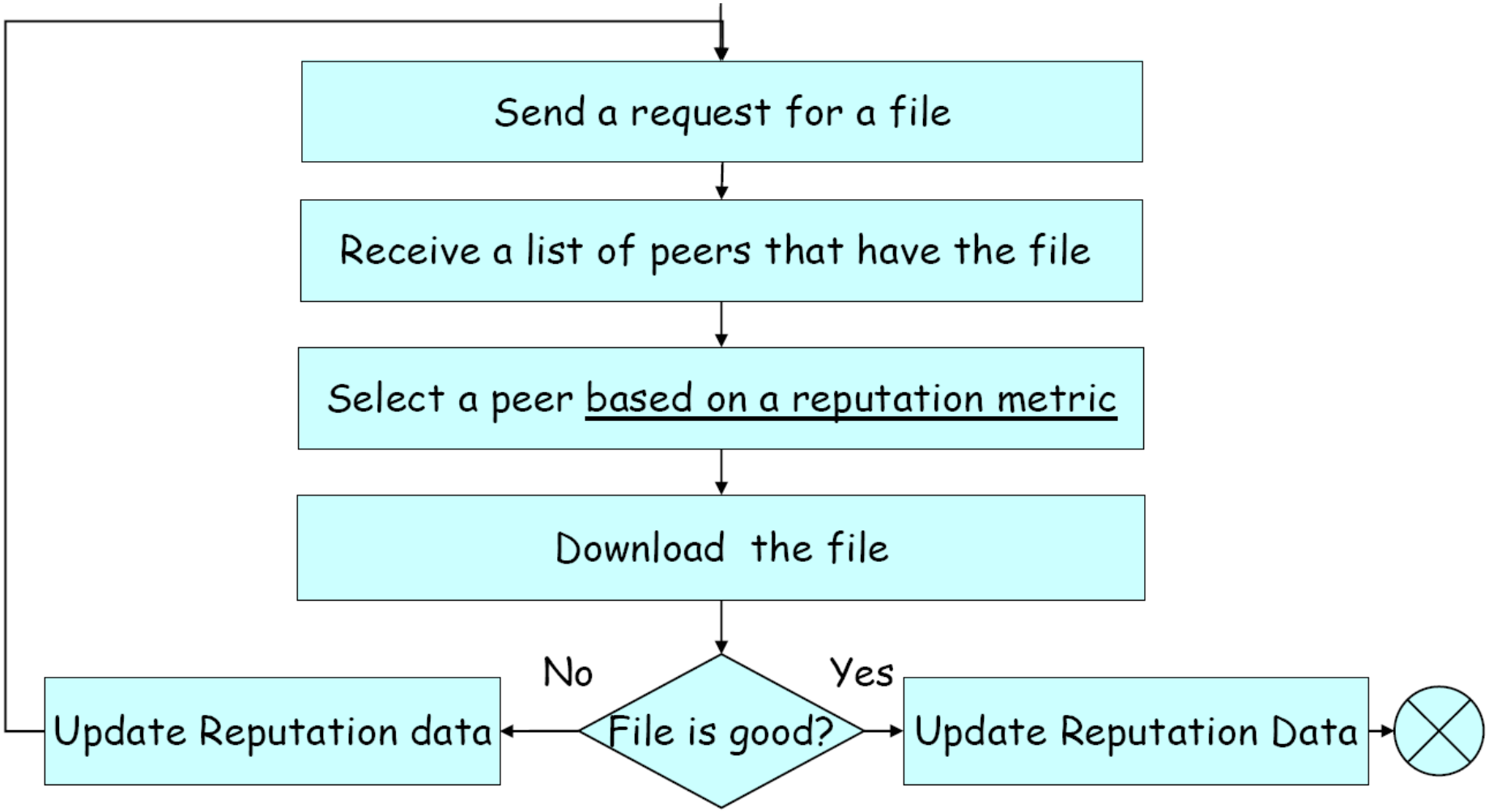
- General overview
- P2P research activities
- Case Studies
  - File-sharing systems
  - Distributed Hash Tables (DHT)
  - Distributed Computing
  - Trust and reputation management
  - Measurements of existing P2P systems
- Future research directions

File Sharing in a P2P system

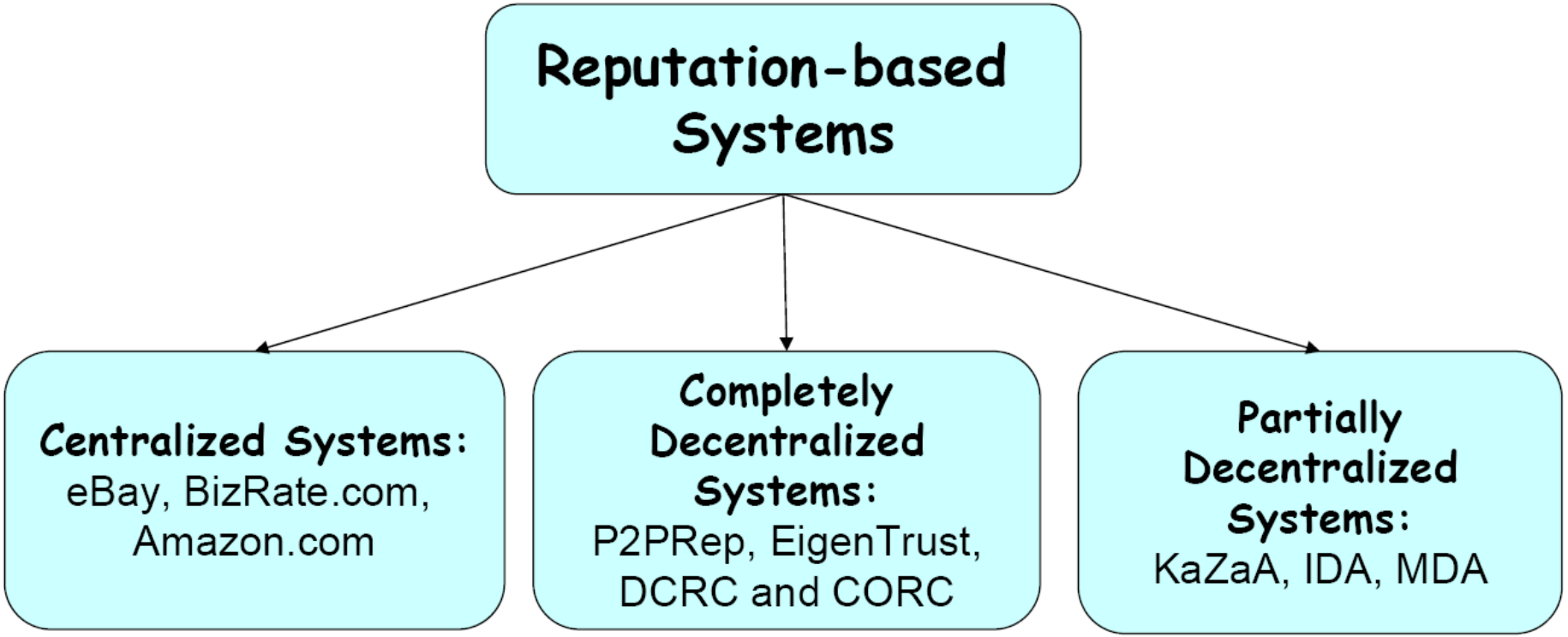


→ Need for a Reputation Management scheme

File Sharing in a Reputation-Based P2P system



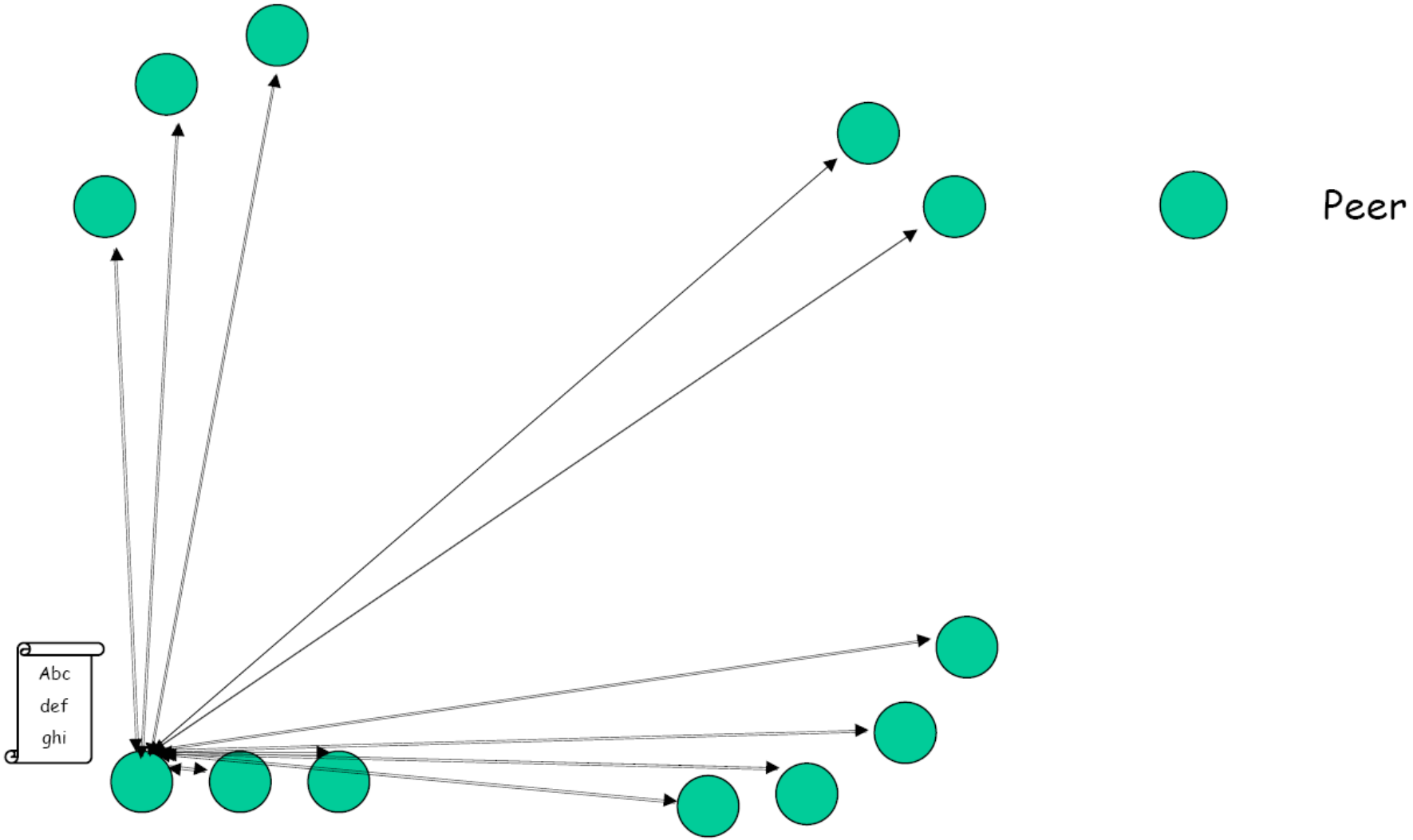
Reputation-based Systems



RM in partially-decentralized p2p systems

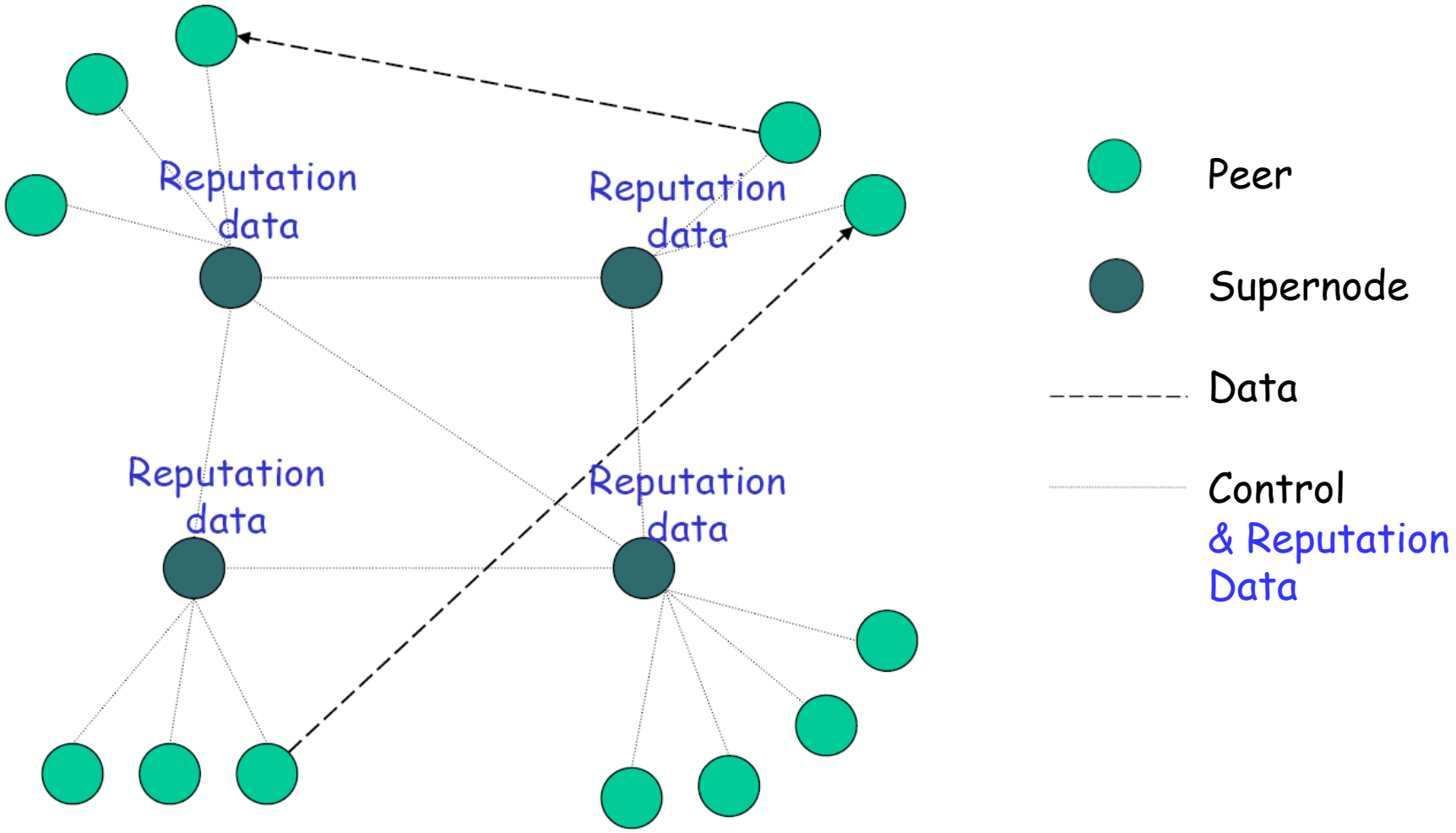
- KaZaA Reputation System
  - Integrity Rating of Files :
    - Excellent, average, poor, delete
  - Participation Level for rating peers:
    - (Uploads in MB/Downloads in MB)* 100
  - Drawbacks of KaZaA reputation system:
    - No distinction between good and malicious peers
    - Designed to reward who are practicing good behavior
    - Does not punish malicious peers
- The Inauthentic Detector Algorithm (IDA)
- The Malicious Detector Algorithm (MDA)

RM Design: Completely Decentralized

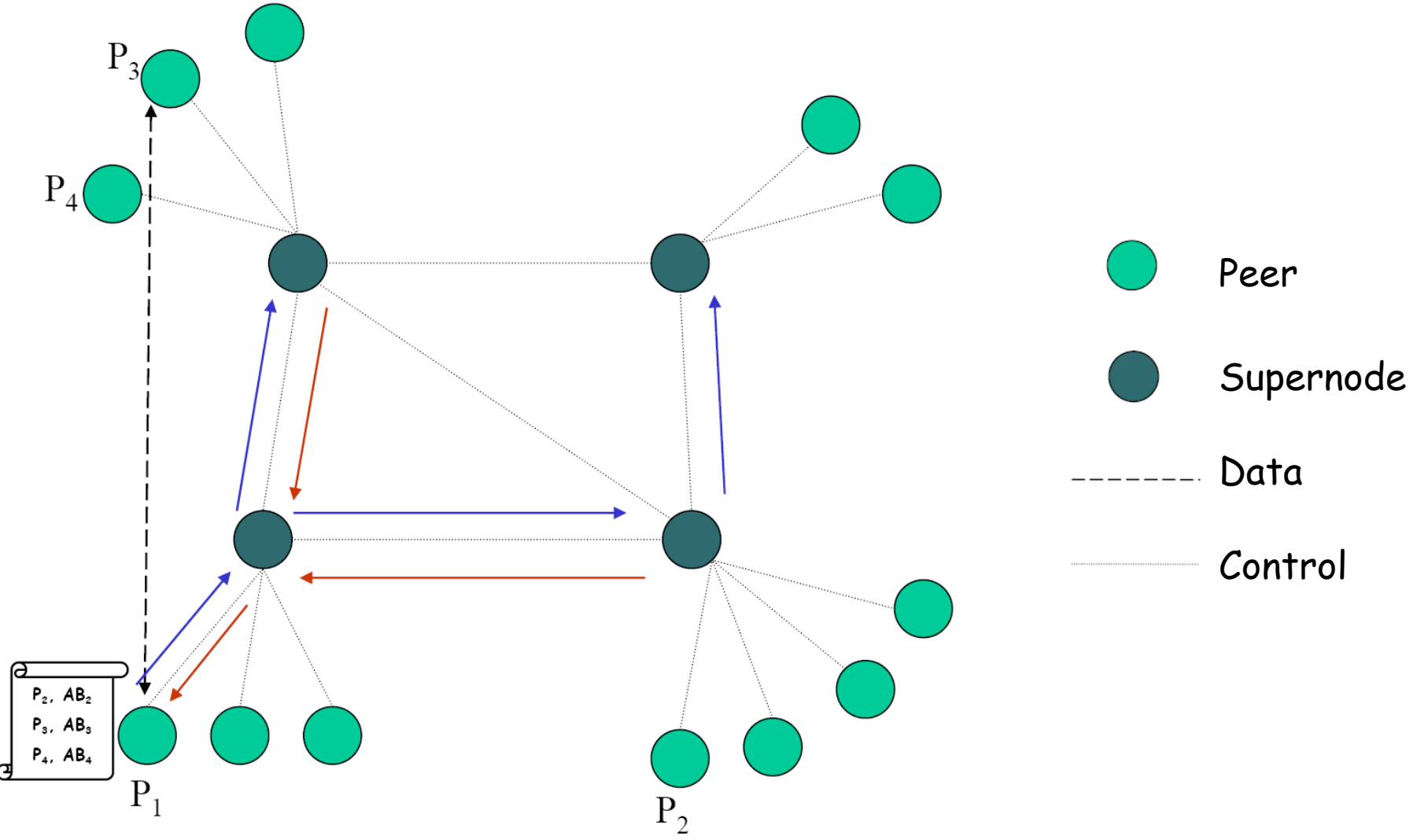


RM Design: Partially Decentralized

Proposed Approach: Use Supernodes for RM



RM Overhead



Effect of liar peers

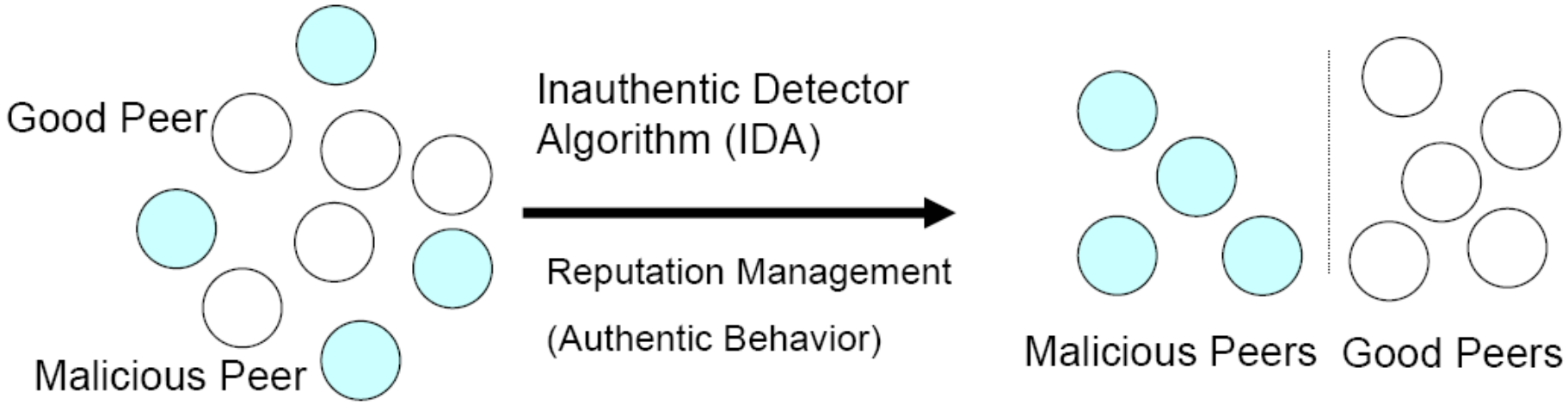
- No liar peers
  - Good peers get high reputation values
  - Malicious peers get low reputation values

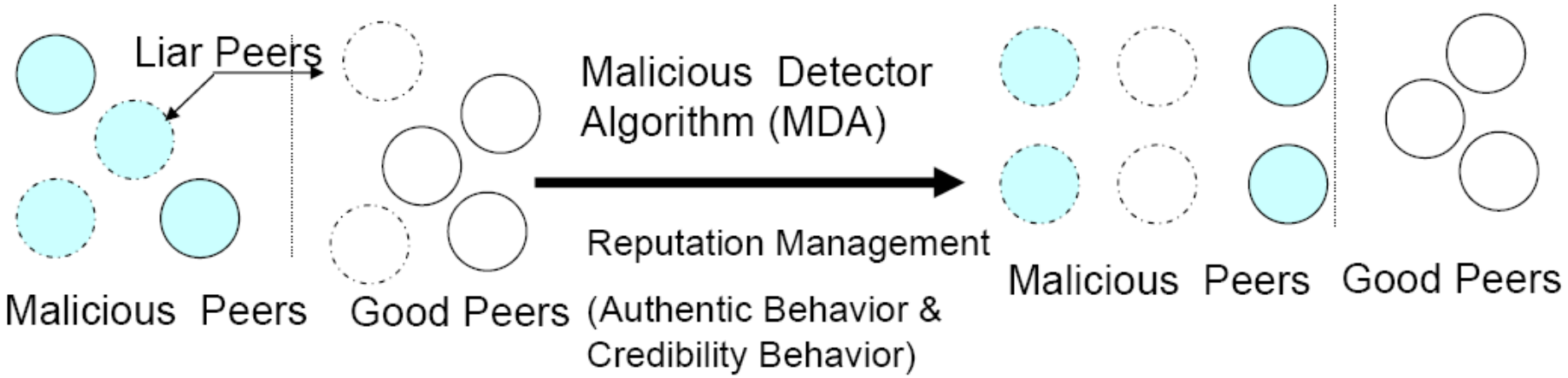
□

- With liar peers
  - Good peers can have low reputation values
  - Malicious peers can have high reputation values

→ Need for a mechanism to detect liar peers

IDA and MDA


Good Peer
Malicious Peer
Inauthentic Detector Algorithm (IDA)
Reputation Management
(Authentic Behavior)
Malicious Peers
Good Peers


Liar Peers
Malicious Peers
Good Peers
Malicious Detector Algorithm (MDA)
Reputation Management
(Authentic Behavior & Credibility Behavior)
Malicious Peers
Good Peers

RM Challenges

- Reputation systems have the following challenges:
  - Eliciting feedbacks:
    - Peers may not send feedbacks
    - It is difficult to ensure an honest feedback
  - Distributing the feedback history is challenging:
    - Peers may change their pseudonyms easily to erase prior feedback history.
    - The lack of portability from system to system.
    - The use of a more universal framework will be beneficial.
  - Aggregating feedbacks is a difficult task

Outline

- General overview
- P2P research activities
- Case Studies
- Future research directions

Future Research Directions

- P2P research is an exciting area with many open problems and opportunities, including the design of:
  - New distributed object placement and query routing
  - New hash table data structures and algorithms
  - Efficient security and privacy
  - Semantic grouping of information in P2P networks
  - Incentive mechanisms and reputation systems
  - Convergence of Grid and P2P systems
  - Providing transactional and atomic guarantees on P2P

Future Research Directions

- P2P and Management:
  - Content management is key
  - Autonomic behavior (e.g., self-organizing communities of interest)
  - Trust and reputation management is a challenge
  - Further exploiting P2P architectures for management apps