A Survey of Peer-to-Peer Content Distribution Technologies

Stephanos Androutsellis-Theotokis and Diomidis Spinellis

ACM Computing Surveys, December 2004

Presented by

Sangyoun Paik

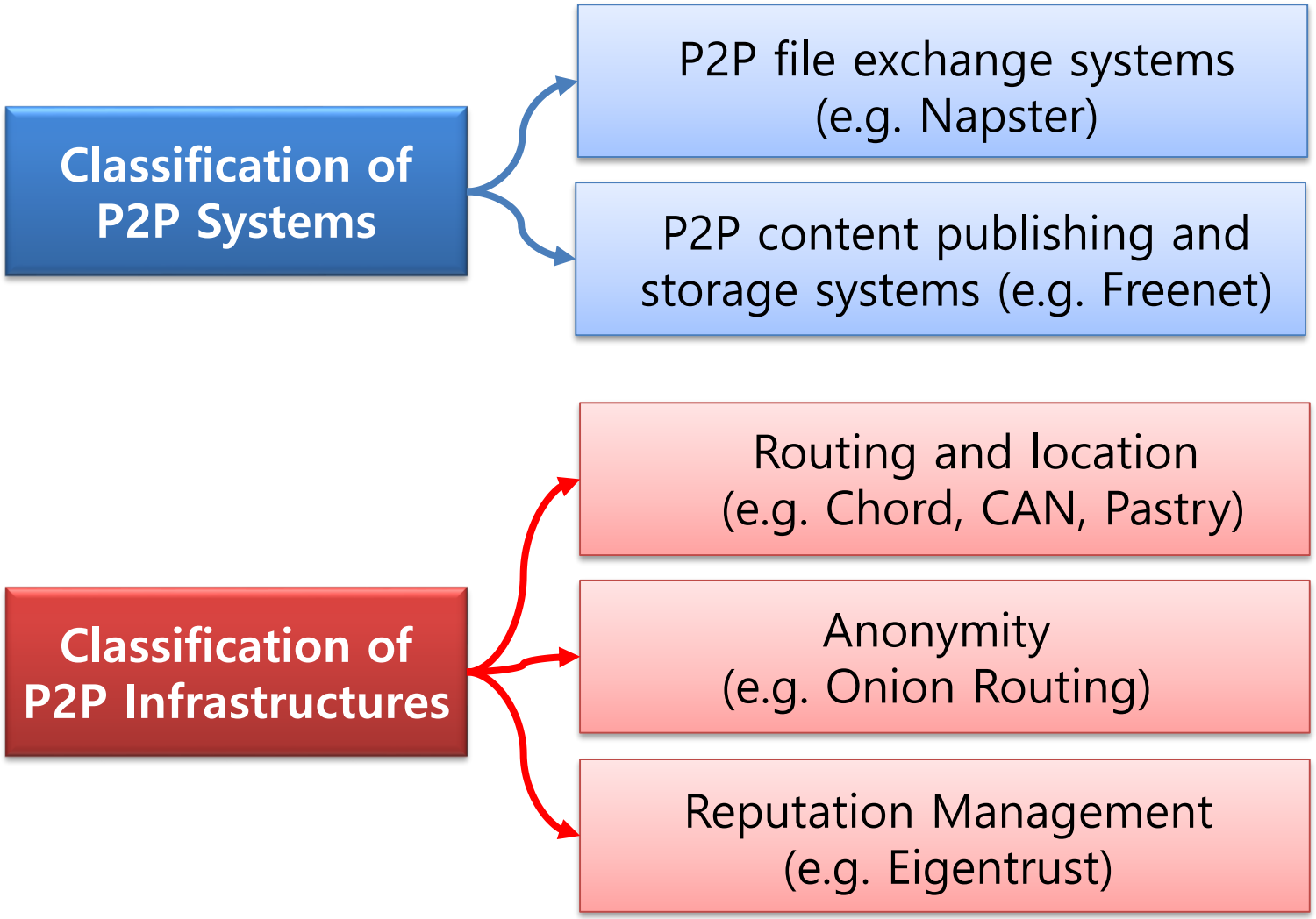
Defining Peer-to-Peer Computing

- Peer-to-peer systems are ***distributed systems consisting of interconnected nodes able to self-organize into network topologies*** with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, ***capable of adapting to failures*** and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, ***without requiring*** the intermediation or ***support of a global centralized server*** of authority.

Classification of P2P Applications

- Communication and Collaboration
- Distributed Computation
- Internet Service Support
- Database Systems
- **Content Distribution**
  - **Systems/infrastructures for the sharing of data between users**
  - **Napster, Gnutella, Freenet, Chord etc**
  - **the survey focuses on**

P2P Content Distribution



Analysis Framework

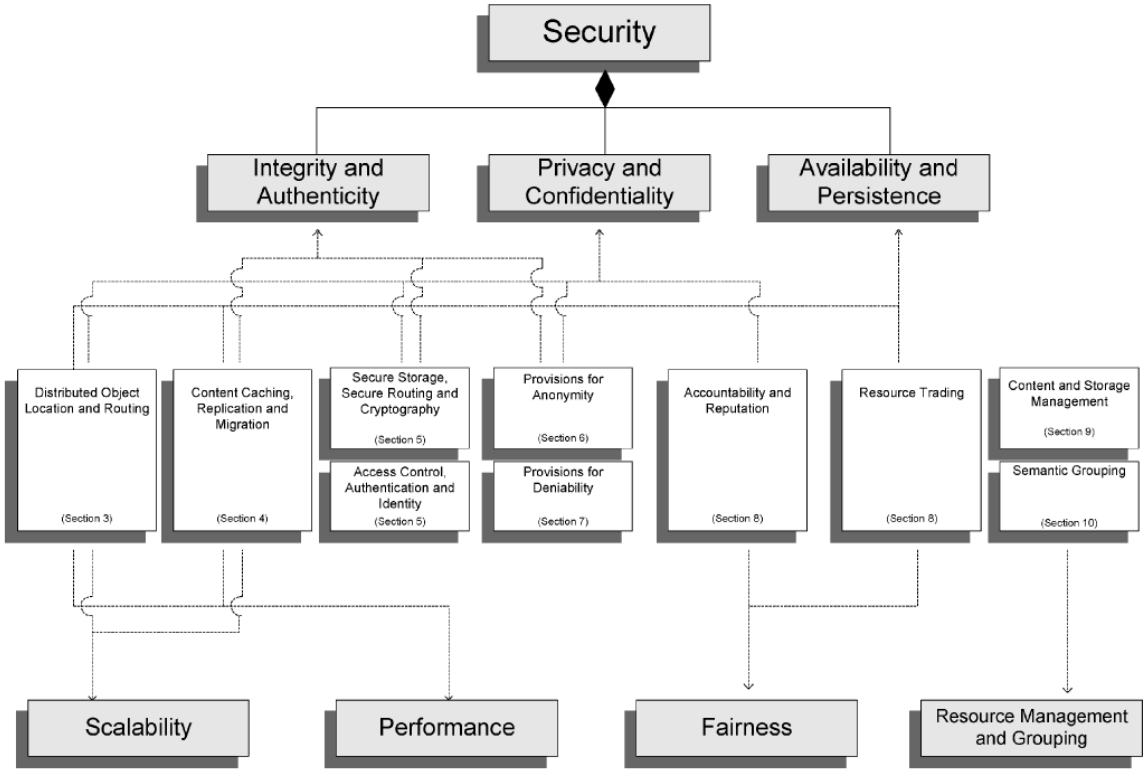


Fig. 1. Illustration of the way in which various design features affect the main characteristics of peer-to-peer content distribution systems.

Analysis Framework

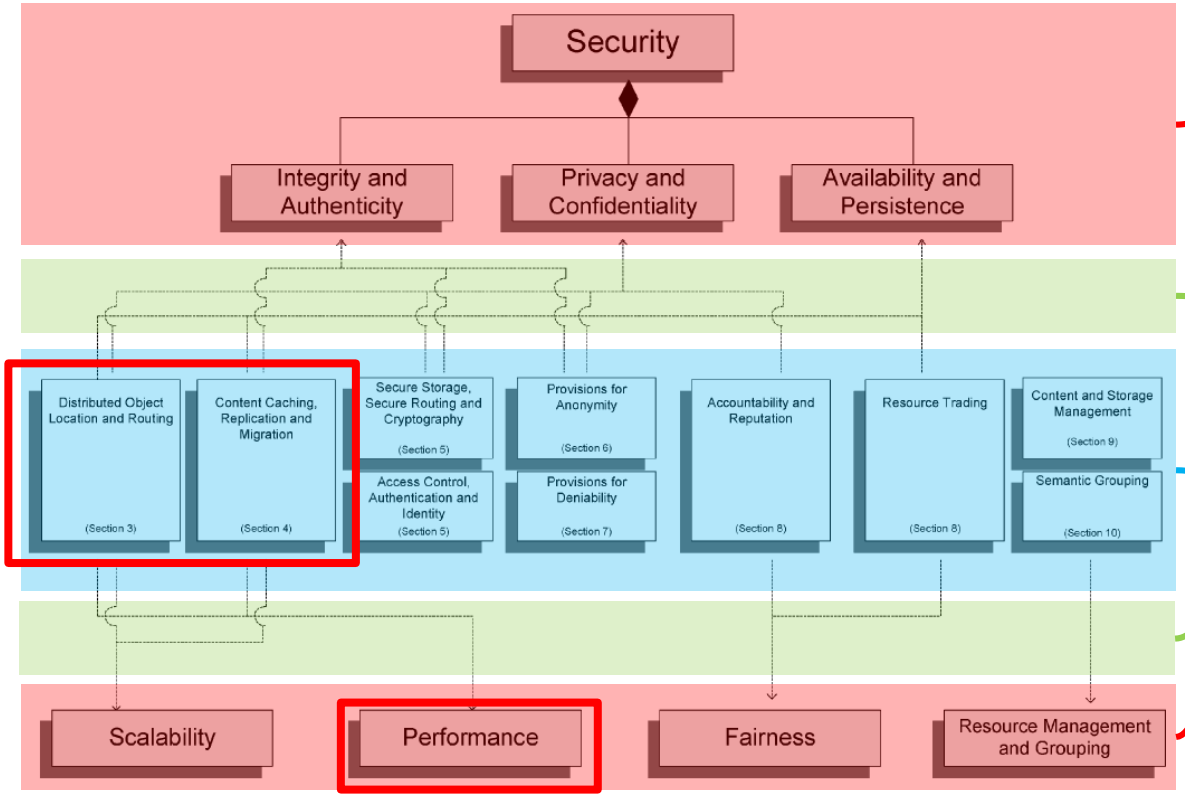


Fig. 1. Illustration of the way in which various design features affect the main characteristics of peer-to-peer content distribution systems.

Characteristics of P2P content distributed systems

Design Decisions

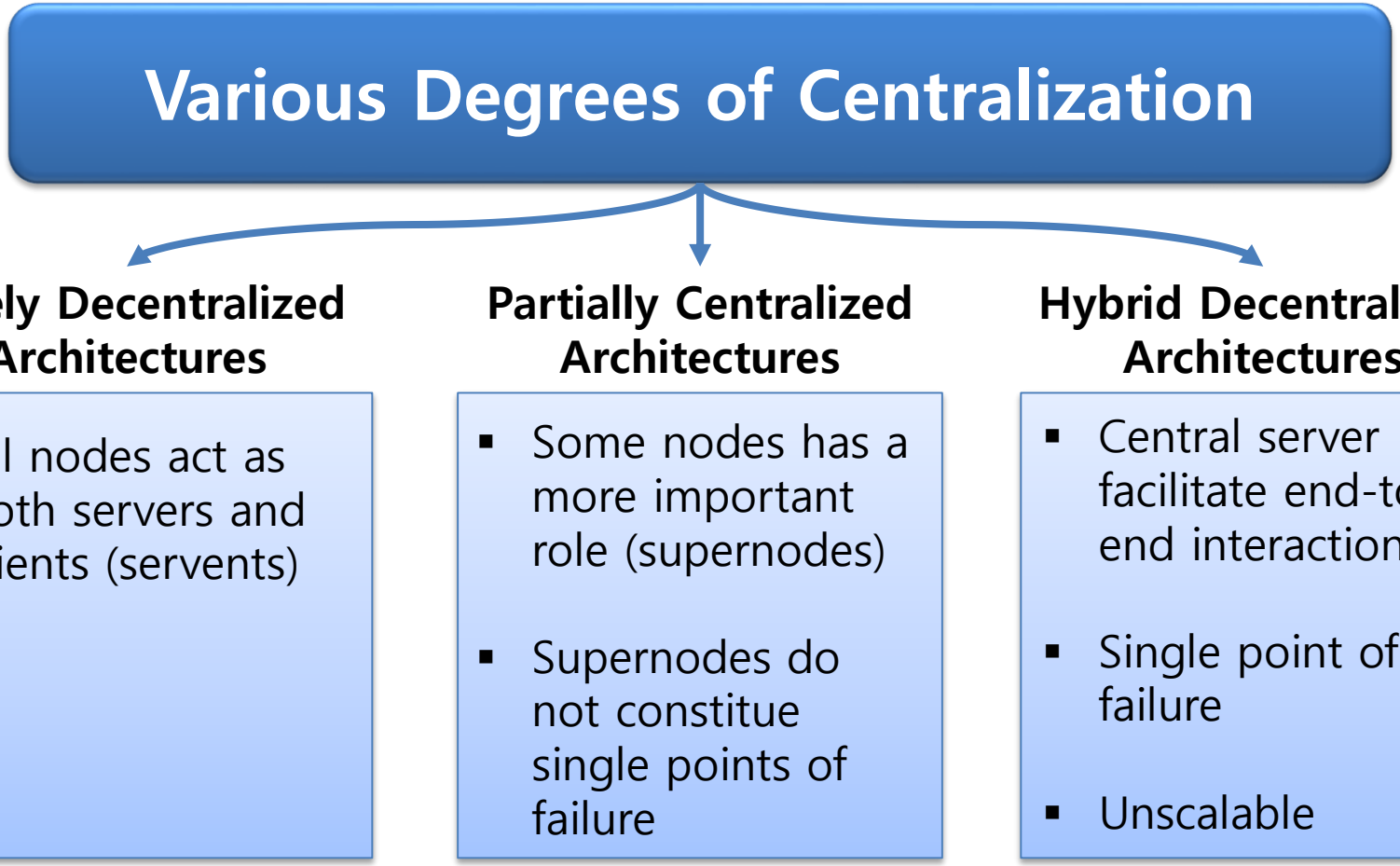
Independent Design features

P2P Distributed Object Location and Routing

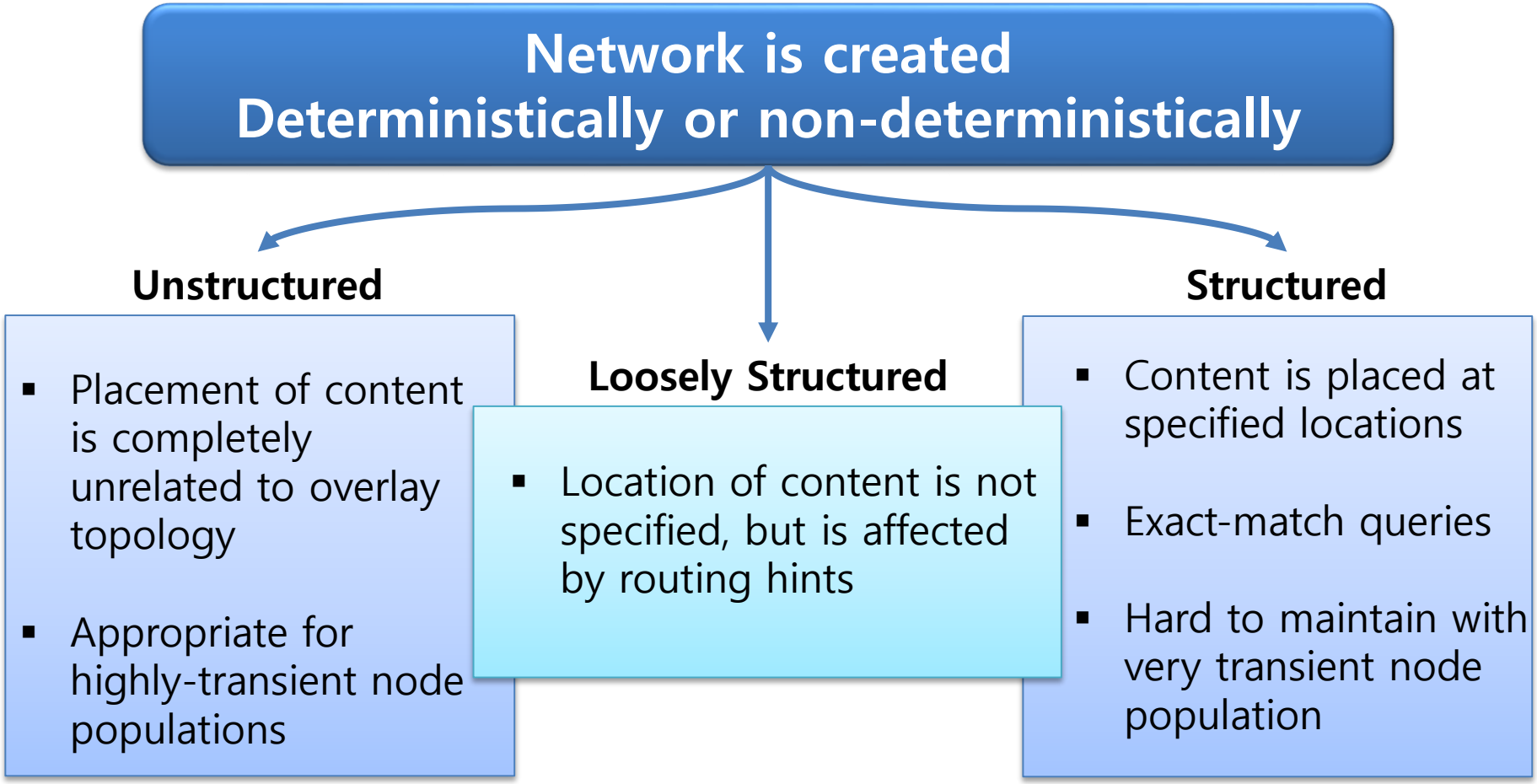
- Overlay network – Network of peer computers and connections between them, formed on top of underlying physical computers

- Distinguish Overlay network in terms of centralization and structure

Overlay Network Centralization



Network Structure



Classification of P2P Content Distribution

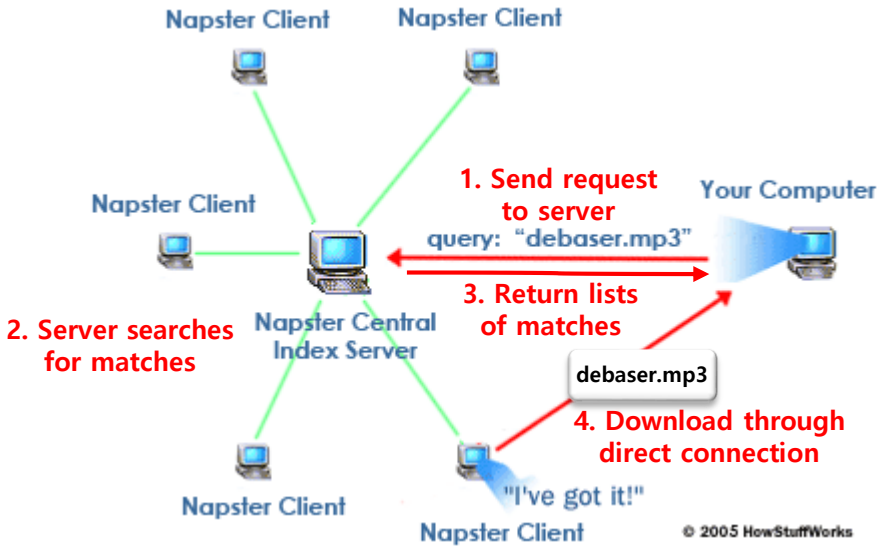
	Centralization		
	Hybrid	**Partial**	**None**
Unstructured	Napster	Kazza	Gnutella
Loosely Structured			Freenet
Structured			Chord, CAN, Tapestry

Unstructured Architectures

	Centralization		
	Hybrid	**Partial**	**None**
Unstructured	Napster	Kazza	Gnutella
Loosely Structured			Freenet
Structured			Chord, CAN, Tapestry

Unstructured Architectures

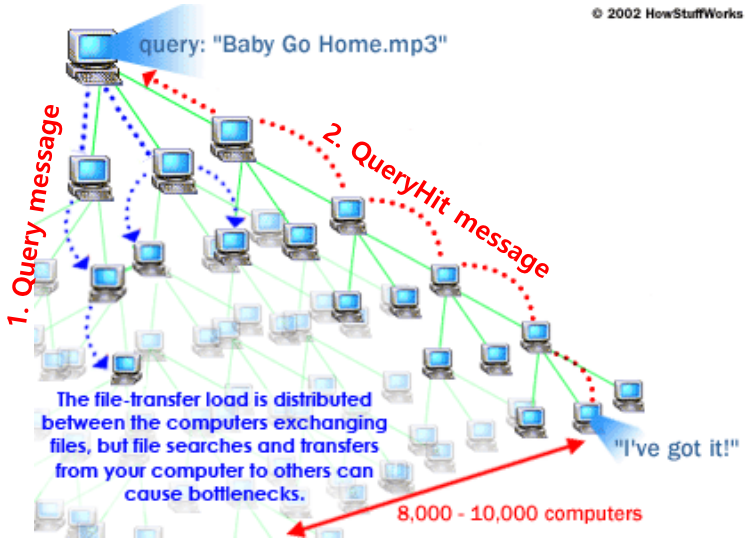
- Hybrid Decentralized
  - e.g. Napster
  - Clients stores content
  - Clients connect to central server
    - ✓ Table of user connection
    - ✓ Table listing files each user holds
  - Advantage
    - ✓ Simple implementation
    - ✓ Locate files quickly
  - Disadvantage
    - ✓ Vulnerable to censorship
    - ✓ Unscalable



http://www.umkc.edu/is/security/p2p_explanation.asp

Unstructured Architectures

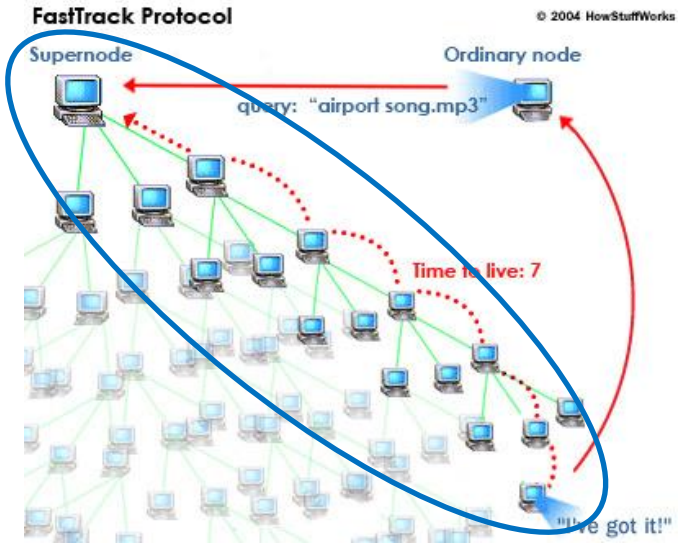
- Purely Decentralized
  - e.g. Gnutella
  - Users connect to each other
  - No central Coordination
  - Application level protocol
    - ✓ Ping, Pong, Query, QueryHits
    - ✓ Time-to-live (TTL) in message header
    - ✓ Scalability issue
      - ➢ With TTL, messages can not reach
      - ➢ Without TTL, network swamped



http://www.umkc.edu/is/security/p2p_explanation.asp

Unstructured Architectures

- Partially Centralized
  - e.g. Kazza
  - Dynamically assign Supernodes
  - Supernodes index the files shared by peers connected to them
  - Advantage
    - ✓ Discovery time is reduced
    - ✓ Nodes are lightly loaded



Supernode servicing a subpart of network by Indexing/Caching

http://computer.howstuffworks.com/kazaa3.htm

Classification of P2P Content Distribution

	Centralization		
	Hybrid	**Partial**	**None**
Unstructured	Napster	Kazza	Gnutella
Loosely Structured			Freenet
Structured			Chord, CAN, Tapestry

Structured Architectures - Freenet

- Loosely Structured
  - knows which node is most likely to store certain content
- Uses **Chain Mode Propagation**
  - local decision
  - avoid blindly broadcasting request message
- File are identified by unique binary keys
- Node maintains **local data store** and **dynamic routing table**
- Messages
  - Node ID, Timeout(hops-to-live), Source ID, Destination ID
- Message Types
  - Data insert
  - Data request
  - Data reply
  - Data failed

Structured Architectures - Freenet

- Data Insert
  - Node calculates a binary key for the file
  - Node sends a data insert message to itself
  - If the key is not found
    - ✓ Node looks up the **closest key** in its routing table, and forwards
    - ✓ Files are placed at nodes possessing files with **similar keys**
  - If the key is found
    - ✓ Node returns the preexisting file

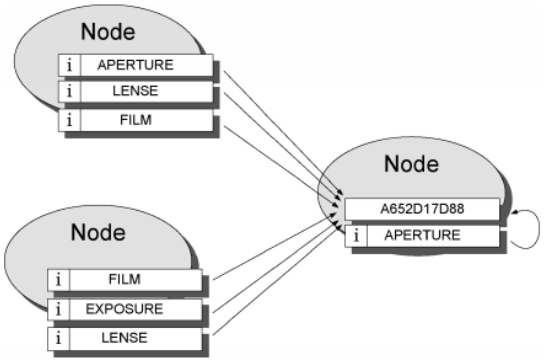
- Data Request
  - If the node does not store the file
    - ✓ Forwards the request to its neighbor that is most likely to have the file
    - ✓ Forming a chain
  - If the node does store the file
    - ✓ Reply is passed back with the actual data

Structured Architectures - Freenet

- Data Fail
  - A node receives a “data failed” messages
  - Selects the next best node and forwards
  - If all nodes failed, sends back “data failed” message to node that sent the request

Structured Architectures - Freenet

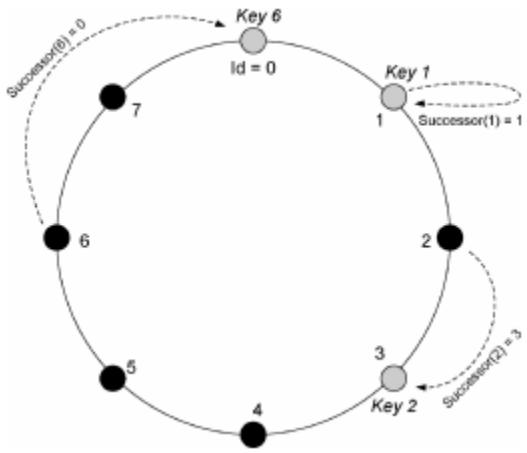
- Indirect Files
  - A special class of lightweight files
  - Named according to search keywords
  - Contain pointers to the real file
  - Multiple files w/ the same key



- Properties
  - Nodes specialize in searching for similar keys
  - Nodes store similar keys
  - Similarity of keys does not reflect similarity of files
  - Routing does not reflect the underlying network topology

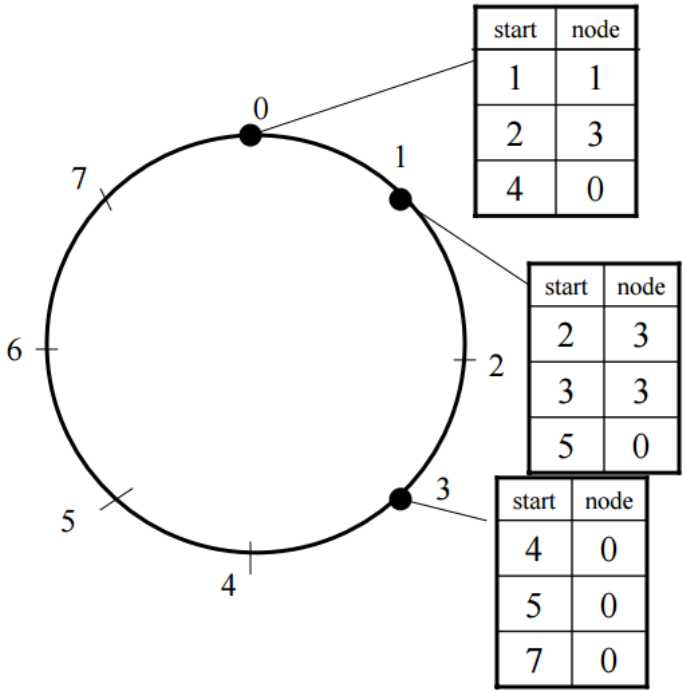
Structured Architectures - Chord

- Mapping of file ID onto node ID
- Node stores (key, data item)
- Node is identified by keys
- Key is assigned both to files and nodes
- All node IDs are ordered in an **identifier circle**
- Only routing information required is **each node's successor node**



Structured Architectures - Chord

- Additional Routing Information
  - finger table
- Time required for lookups
  - O(N) -> O(logN)
- Node 0 searches for id 5
  - Searches 5's predecessor
  - Node is not 5's predecessor
  - Scan finger table, search for the closest preceding finger
  - Node 3 is the first node that is between 0 and 5
  - Node 3 is 5's predecessor
  - Node 3's successor is returned



Structured Architectures - CAN

- Hash Table
  - Maps file names to location
  - (key,value) pairs in the table
  - Support insertion, lookup, and deletion
- Node **stores a zone** of the hash table
- Cartesian coordinate space store (key K, value V) pairs
- Key K is mapped onto a point P
- Routing works by following the straight line path through Cartesian space.
- A node Joins,
  - Allocated its portion by splitting the allocated zone of an existing node in half
- A node leaves,
  - Hash table entry is handed over to one of its neighbors

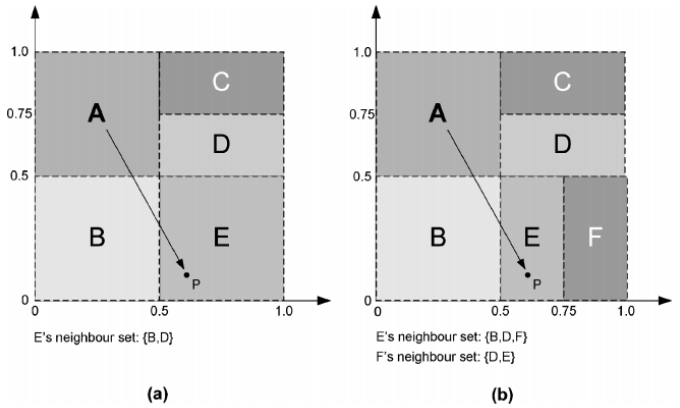


Fig. 6. CAN: (a) Example 2-d $[0, 1] \times [0, 1]$ coordinate space partitioned between 5 CAN nodes; (b) Example 2-d space after node F joins.

Structured Architectures - Tapestry

- Network is self-organizing
- Routing and Location Information is distributed among the network nodes
- **Plaxton mesh**
  - Nodes take role of servers, routers and clients
- Each node maintains a **neighbor map**
  - Level + Entry

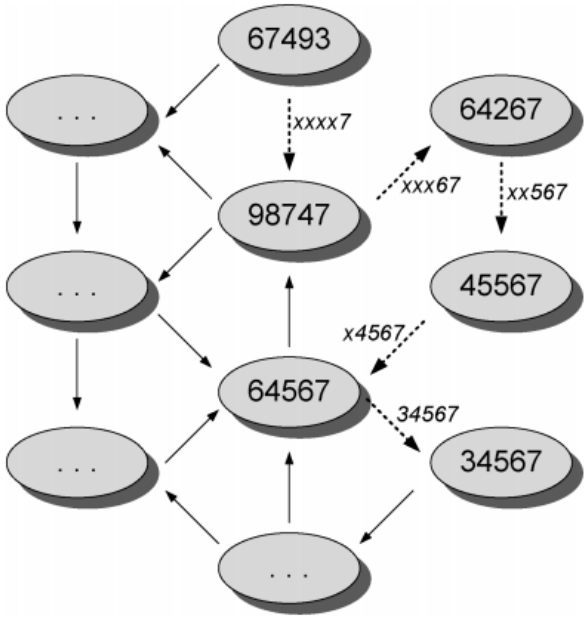
Table IV. The Neighbor Map Held by Tapestry Node With ID 67493

	Level 5	Level 4	Level 3	Level 2	Level 1
Entry 0	07493	x0493	xx093	xxx03	xxxx0
Entry 1	17493	x1493	xx193	xxx13	xxxx1
Entry 2	27493	x2493	xx293	xxx23	xxxx2
Entry 3	37493	x3493	xx393	xxx33	xxxx3
Entry 4	47493	x4493	xx493	xxx43	xxxx4
Entry 5	57493	x5493	xx593	xxx53	xxxx5
Entry 6	**67493**	x6493	xx693	xxx63	xxxx6
Entry 7	77493	x7493	xx793	xxx73	xxxx7
Entry 8	87493	x8493	xx893	xxx83	xxxx8
Entry 9	97493	x9493	xx993	xxx93	xxxx9

Each entry in this table corresponds to a pointer to another node.

Structured Architectures - Tapestry

- Messages route digit-by-digit
- Example path taken by a message
  - node ID 67493 -> node ID 34567
  - Digits are resolved right to left
  - xxxx7->xxx67->xx567->x4567->34567



Structured Architectures - Tapestry

- Plaxton mesh uses a **Root node** for each object
  - Provides a guaranteed node object can be located
  - Stores data in the form of (object id o, storer id ns)

- Advantage
  - Fault-handling
  - Scalability (bottleneck at root node)
- Disadvantage
  - Need for global knowledge for root node
  - Vulnerability of root node

A survey of peer to peer content distribution technologies

Heymo Kou

Contents

- Content caching, replication & migration
- Security
- Provisions for anonymity
- Provisions for deniability
- Incentive mechanisms and accountability
- Resource management capabilities
- Semantic grouping of information

Content caching, replication & migration

- Passive Replication - Napster
- Cache-Based Replication
- Active Replication -
- Introspective replica management - OceanStore, MojoNation, Scan & Gnutella
  - Watch traffic and manage replica
  - Endures DDOS

Content caching, replication & migration

- Dynamic replica management
- Data consistency & synchronization is always an issue
  - especially when deletion and update is allowed
  - Some weaken their consistency restriction
- For Chord, replication is challenging
  - Since objects' location is mapped with object identifiers

Security

- Securing storage is very important

- Self-Certifying Data - Compare Key w/ hash function result
- Information Dispersal - endcode files into m blocks
  - Redundancy factor = m / n
- Shamir's Secret Sharing Scheme(SHA) - widely used
  - encrypts a file with a key K
  - split K into l shares
  - any k splits can reproduce K
  - but k - 1 splits cannot reproduce nor give any hint about K
  - For the file to become inaccessible
- At least (l - k - 1) servers containing the key must shut down
  - (high fault-tolerant)

Security

- Anonymous Cryptographic Relays - PAST
  - Role : Publisher, Forwarder, Storer & Clients
  - Publisher - select forwarders to send encrypted pieces
  - Forwarder - select random nodes for storing the shards & send them
    - Once all storers received the pieces publisher destroys the shares
    - Announce file name & list of forwarders
  - Client - Reach forwarders for the shares
    - Forwarders - request Storers to send the shares to client
  - Visible to potential attacker
  - BUT less likely to be attacked since forwarder don't store any content

- Smartcards - Issue digitally signed tickets
  - Impractical to issue large number of nodes

Security

- Distributed Steganographic File System
  - Encrypted blocks are indistinguishable
  - Randomly write blocks
  - Replace part of them with files in a pseudo-randomly chosen locations
  - Considerable amount of replication is required to avoid collisions
- Erasure Coding
  - Data is broken in to blocks and spread over many servers
  - Only a fraction is needed to reconstruct the original block
  - Broken blocks named globally unique using a secure hash
    -
- Sybil Attack [Douceur, 2002]
  - P2P is exposed to attack unless a central certification or identification authority

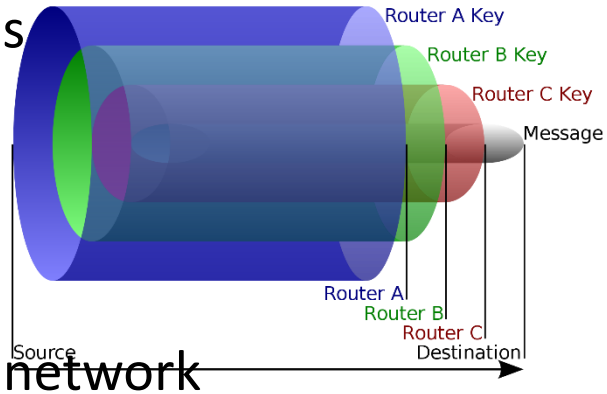
Provisions for anonymity

- Hiding information
  - Author(publisher), identity of nodes,
  - Identity of content itself, query for retrieval of the content

- Disassociation of Content Source ar
  Freenet
  - Infeasible to discover the true origin c
    file
  - When a search for a file succeeds
  - Sends the file to previous search node and so on
  - Hops-to-live counter is randomly chosen

Provisions for anonymity

- Anonymous Connection Layers - Onion routing
  - splits documents into encrypted shares
  - Use an anonymizing layer of nodes(forwarders)
  - Choose which nodes to store the shares
  - Construct "onions" around them
  - anonymously forward them



- Tarzan system [Freedman et al. 2002]
  - completely decentralized anonymizing network
  - distributes anonymous IP tunnels between peers
  - Clients can communicate without leaking identity
    - Sequences of mix relays chosen from a large pool of volunteer nodes
    - Packets are routed through randomly chosen Tarzan peers
  - Performance is sacrificed

Provisions for deniability

- Deniability of Stored Content
  - Stored encrypted shared of files does not offer any information without key
- Deniability of Content in Transit
  - Offered through anonymous connection layers

Incentive mechanisms and accountability

- Decentralized P2P system relies on voluntary participation
- Providing incentives stimulates cooperative behavior
  - "Free-rider" - only consume resources without contributing
- 2 solutions
  - Trust-based incentive mechanisms / Reputation mechanisms
  - Trade-based incentive mechanisms
    - Offer service explicitly for contribution

Incentive mechanisms and accountability

- Reputation mechanisms
  - Centralized reputation systems (i.e. eBay, Gmarket, Auction)
    - People trust reputation information presented by them
  - No organization manages the reputation information in P2P
    - Peers' reputation is spread out through the network
      - produce global reputation

- Various reputation management algorithm
  - Eigen Trust algorithm [Kamvar et al. 2003]
    - Produces global reputation rating based on users' history of uploads
    - Rating can be used by others to decide where to download files from
    - Reduced the number of fake file / malware
  - Reputation computation agents [Gupta et al. 2003]
    - Partially centralized
    - Reputation data is stored at reputation agent

Incentive mechanisms and accountability

- Feedback-based reputation mechanism [Xiong and Liu 2002]
  - Trust value for a peer is based on satisfaction received by other peers
- Sun's JXTA [Jxta 2003]
  - Provides penalty to misbehaving nodes
- Micropayments Mechanisms
  - Currency-based micropayment system
  - MojoNation [MojoNation 2003]
    - Mojo - currency gained by offering PC resources
  - Impractical for most applications [Buragohain et al. 2003]

Incentive mechanisms and accountability

- Resource Trading Schemes
  - Transitive chains of peer transaction[Anagnostakis and Greenwald 2004]
    - node publishes digitally signed list of files
    - no node uses more resources than it is providing
    - balanced system is achieved
  - Flexible algorithm [Cooper and Garcia-Molina 2002]
    - Peers decide
      - how much data to trade
      - what resources to contribute
      - how many trades to make
    - Ensures fairness while maximizing content preservation
  - FreeHaven [Dingledin et al. 2000]
    - Contract - nodes store peer's material for a certain period of time
    - When fullfiled the contract, node's reputation increases
    - Contracts are formed when new data is inserted in the network
    - Each node maintains a database of other’s reputation

Resource management capabilities

- Resources in P2P - content, storage & bandwidth
- Basic operation in P2P - insertion, search & retrieval

- Content Deletion and Update
  - MojoNation, PAST & FreeHaven applied immutable files
    - NO update allowed
    - Requires new version with different name for update
  - Publius offers both deletion and update of content
    - Only publisher will be able to delete / modify with password file

- Content Expiration - FreeHaven

- Content Versioning - OceanStore
  - Offers a version-based archival storage
  - OceanStore maintains active maps of document IDs
    - redirects to the most recent version on request

Resource management capabilities

- Directory Structure - Mnemosyne
  - Manage as a whole
  - Entire distributed directory structure system

- Content Searching
  - Unstructured systems - Gnutella, Kazaa & FreeHaven
    - Offer keyword-search mechanisms
    - do not scale
  - Structured systems
    - Very efficient in searching through file identifiers
    - Keyword search is still problem
    - One solution by Freenet
      - Use of indirect files - pointer of file
      - Name the indirect file with search keywords
      - Still managing in large volume situation remains unsolved

Semantic grouping of information

- “Peer communities" [Khambatti et al. 2003]
  - Membership depends on the relationship between peers that share common interests
  - Efficient searching & better result
  - Communities are formed between peers that share similar attributes

- Semantic overlay clustering [Loeser et al. 2003]
  - Creates logical layers above the physical network topology
  - Clusters with similar semantic profiles matched