Τεχνολογία Πολυμέσων

Section 1

- Cloud and Multimedia Distribution

Media Distribution Catalog

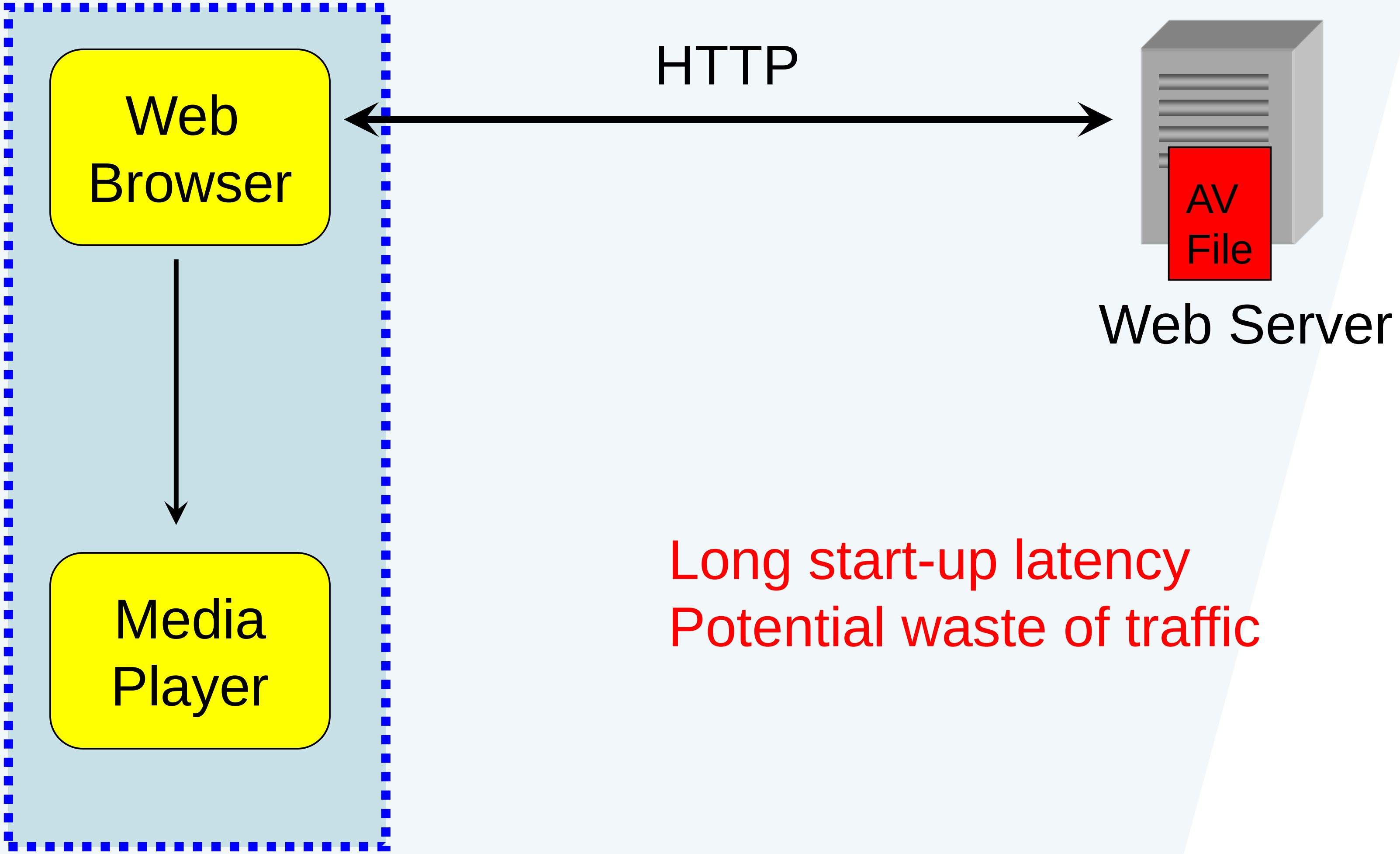
- Media distribution - Deliver media contents to users

- Delivery via disc:
  - Merits: Large storage, high audiovisual quality
  - Demerits: long delivery time, inflexible
- Delivery via Internet:
  - Non realtime delivery:
    - Called download service:

      >download ***all*** data, save to disc, and play
    - Using data file transfer protocols like ftp and http via ftp or web server
  - Realtime delivery:
    - Called streaming service:

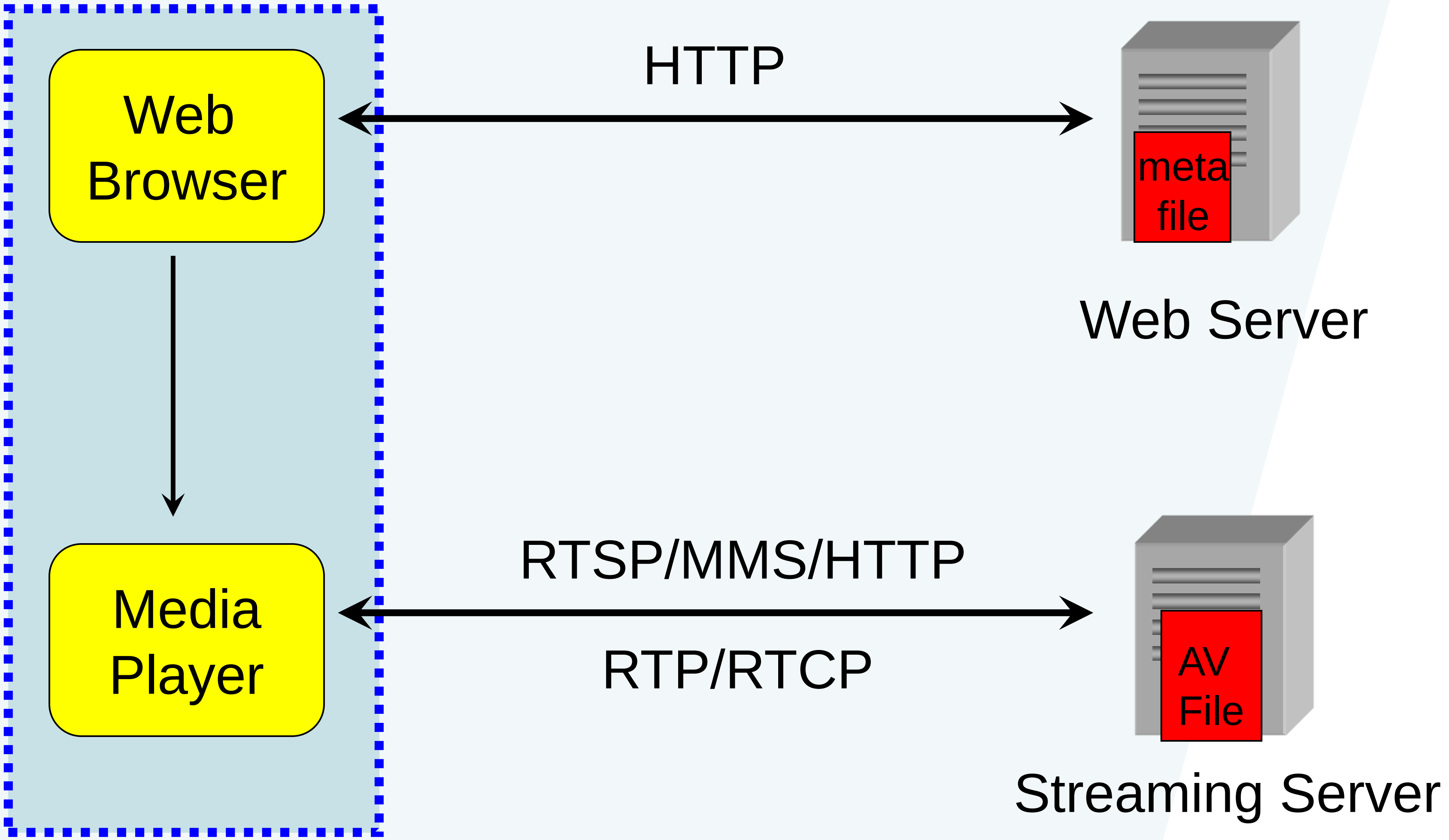
      >download & play simultaneously, partial data in buffer, no data in disc
    - May use http and web server to provide limited streaming service
    - Often use RTSP/RTP and media server for rich streaming service

Non Realtime Delivery: Download service

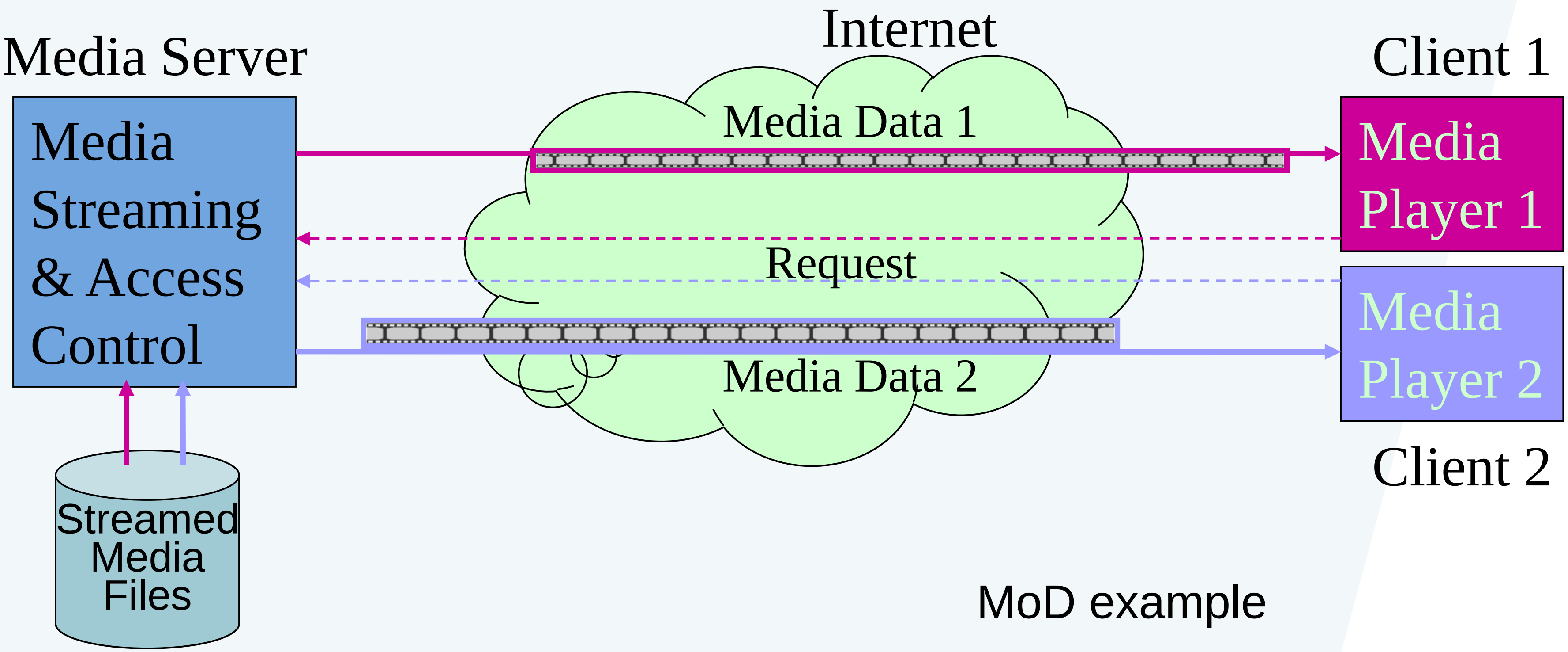


Long start-up latency
Potential waste of traffic

Realtime Delivery: Stream Service



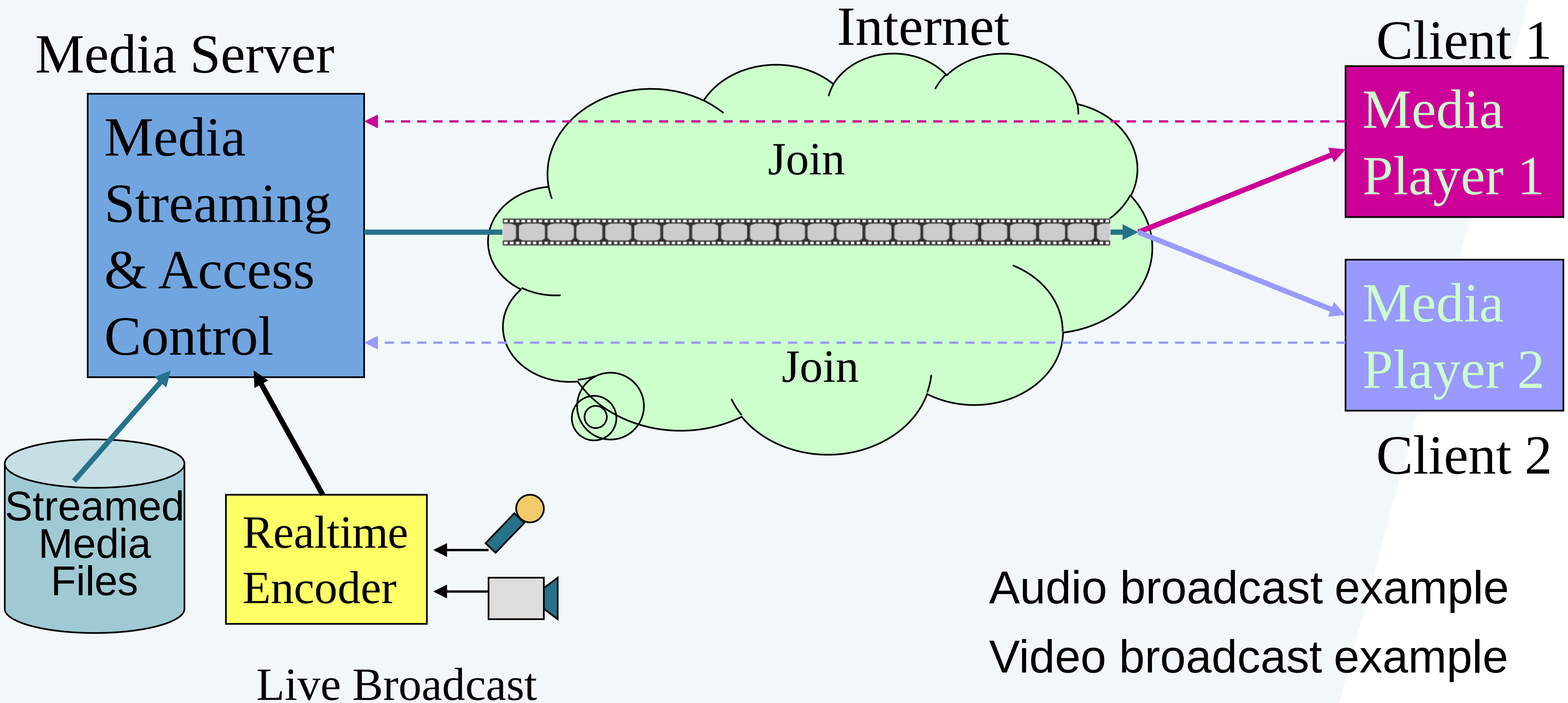
Streamed Media On Demand Delivery



MoD example

- Media on demand (MoD)
  - Streamed media are saved in media server as streamed file format
  - Clients, i.e., media player, access media contents independently
  - Media content is played from the file beginning for each client's request
  - User can control playing, such fast forward, pause, …
  - Like rent a video tape or DVD and replay it in your cassette/DVD palyer

Streamed Media Broadcast

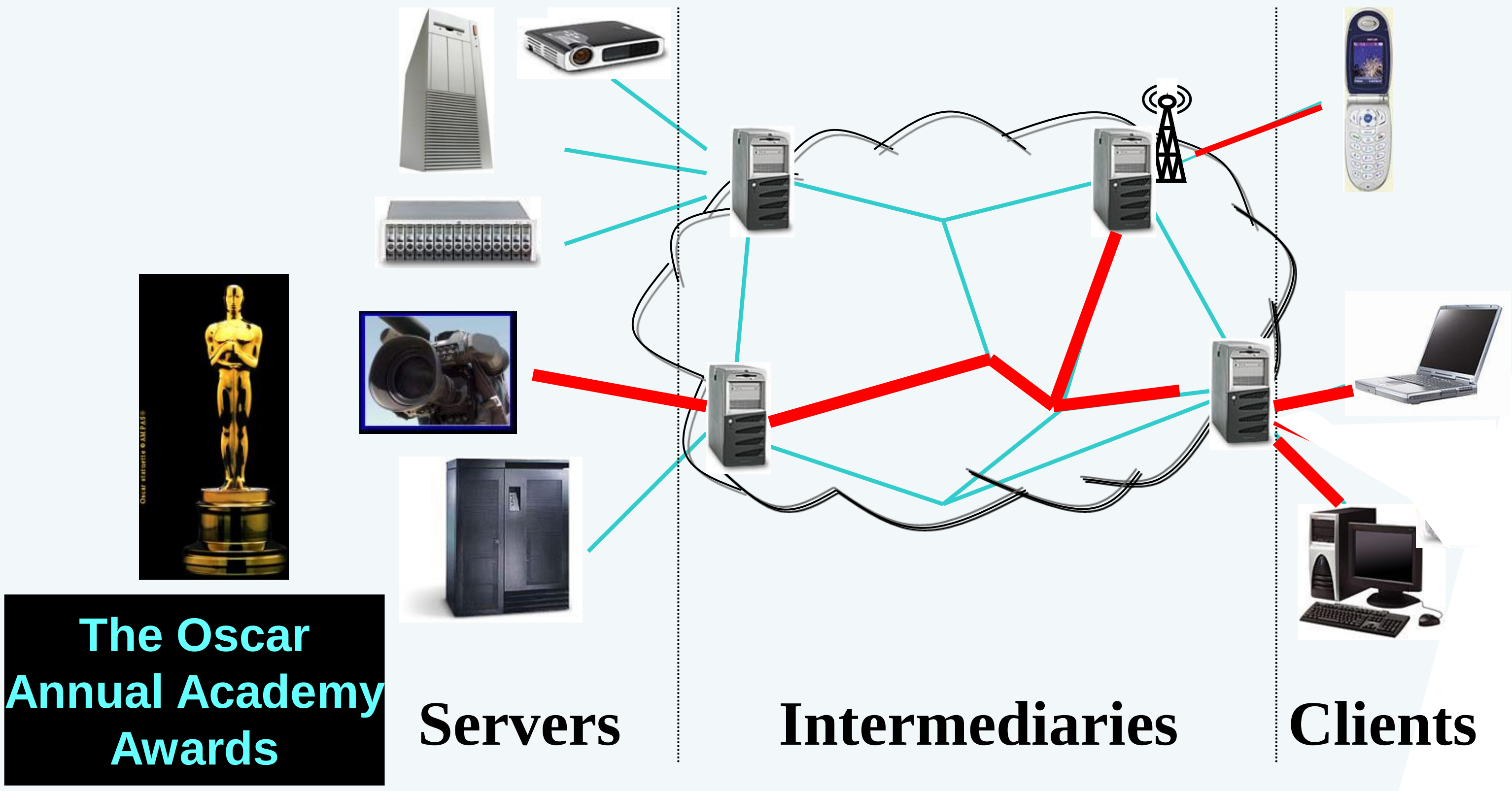


Audio broadcast example

Video broadcast example

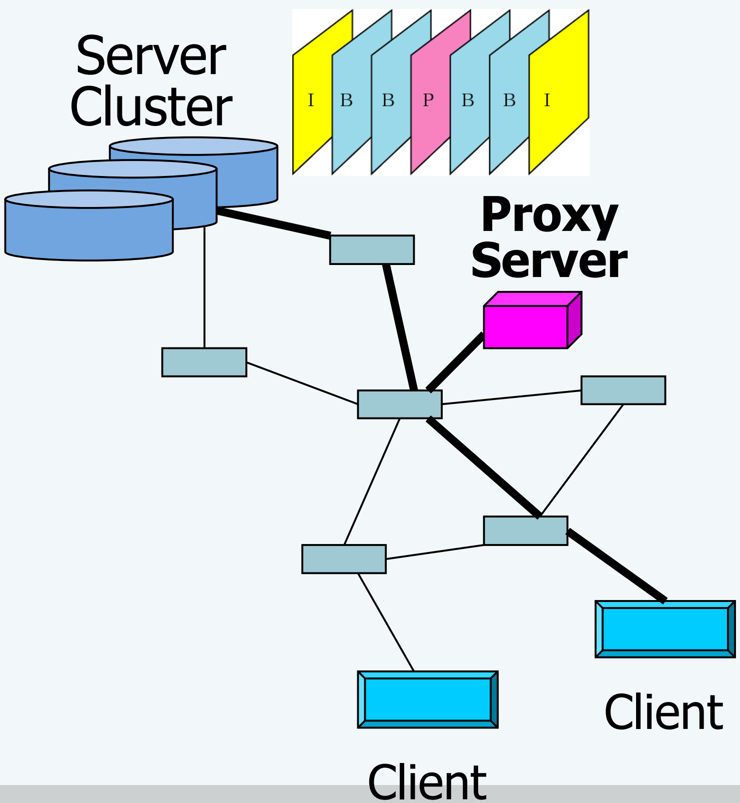
- Media Internet Broadcast (MIB) or Webcast
  - Media may be stored in server or captured lively and encoded in realtime
  - Clients can join a broadcast and same media content goes to all clients
  - Users watch/listen the broadcast from the current state not from beginning
  - Users can’t control its playing such fast forward, stop, etc.
  - Like conventional radio and TV broadcast

Multicast Example: Single Stream and Copy



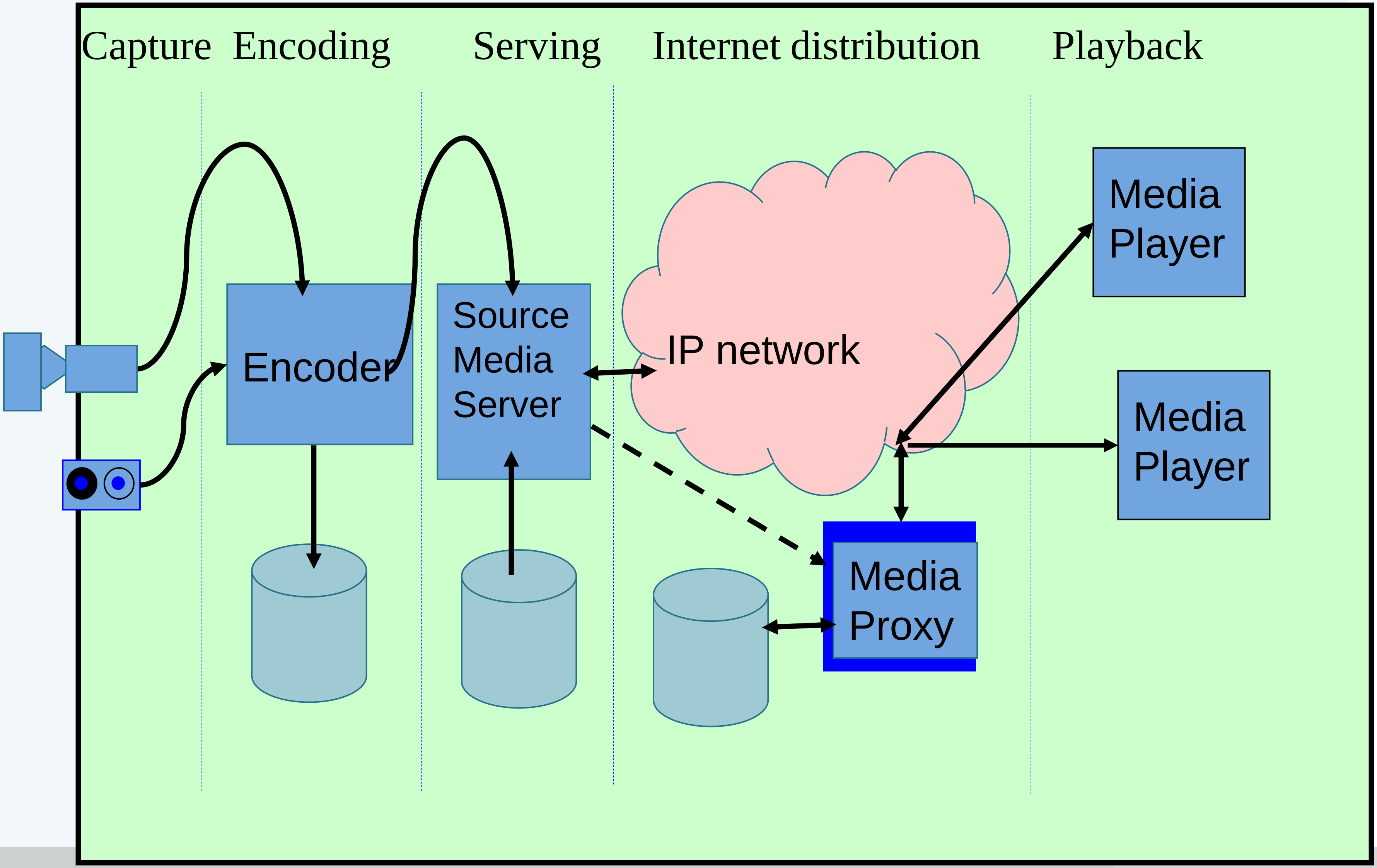
Key Points in Streaming Media Service (Cont)

- <u>Cache technology</u>
  - Increase IO via putting media data in memory
  - The larger memory, the better
- <u>Distributed server cluster and proxy media server</u>
  - Use a group of servers to improve processing performance
  - Use proxy server to reduce number of users' direct accesses to server

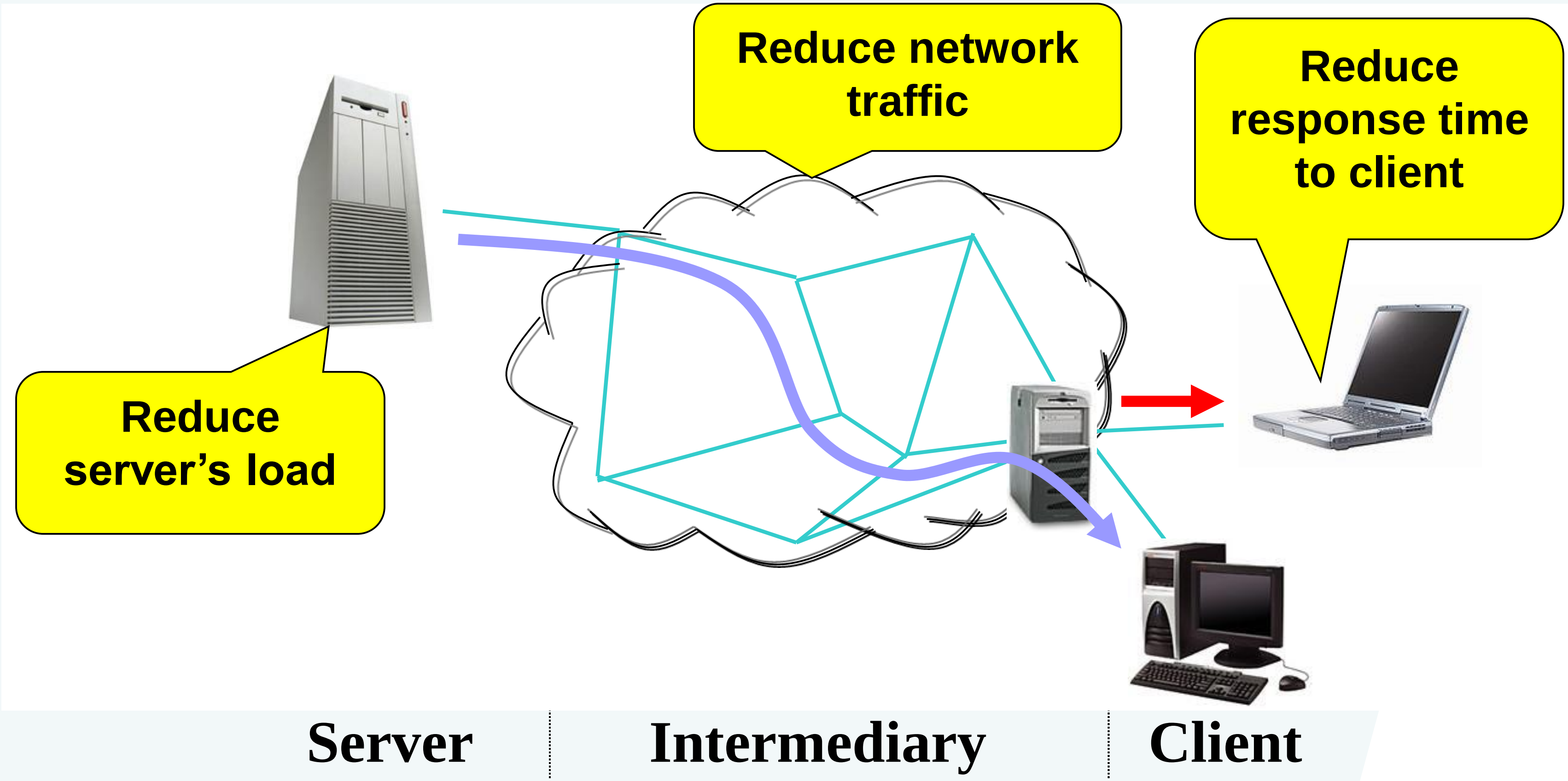


- Drop frames
  - Drop B,P frames if not enough bandwidth
- Quality Adaptation
  - Transcoding
    - Change quantization value
    - Change coding rate
- Video staging, caching, patching
  - Staging: store partial frames in proxy
  - Prefix caching: store first few minutes of movie
  - Patching: multiple users use same video

Proxy Media Server



Proxy Server: Reduce Traffic, Time, Load

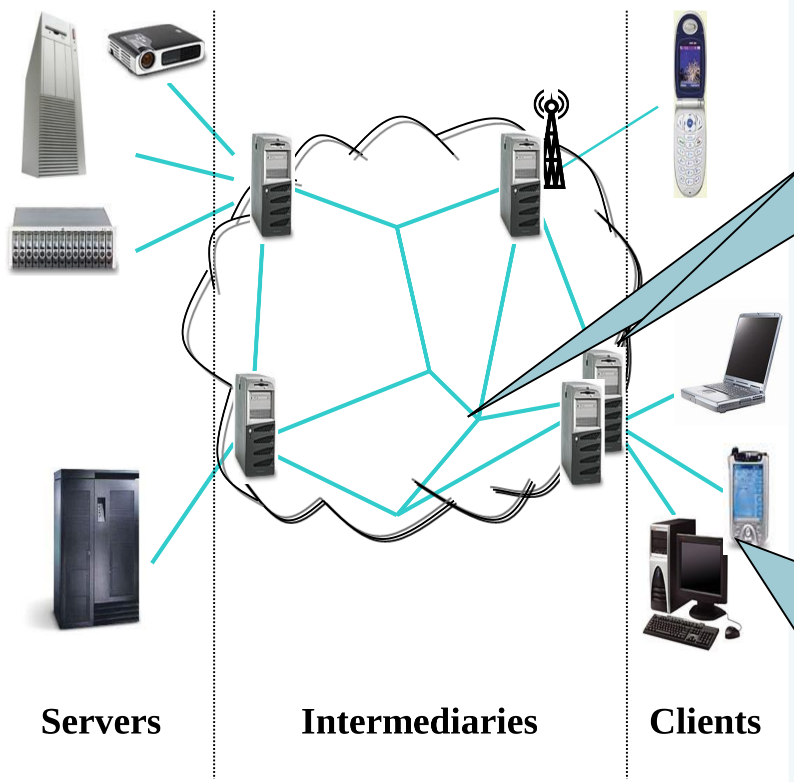


Distributed Proxy Servers

Very large sizes

Media Objects

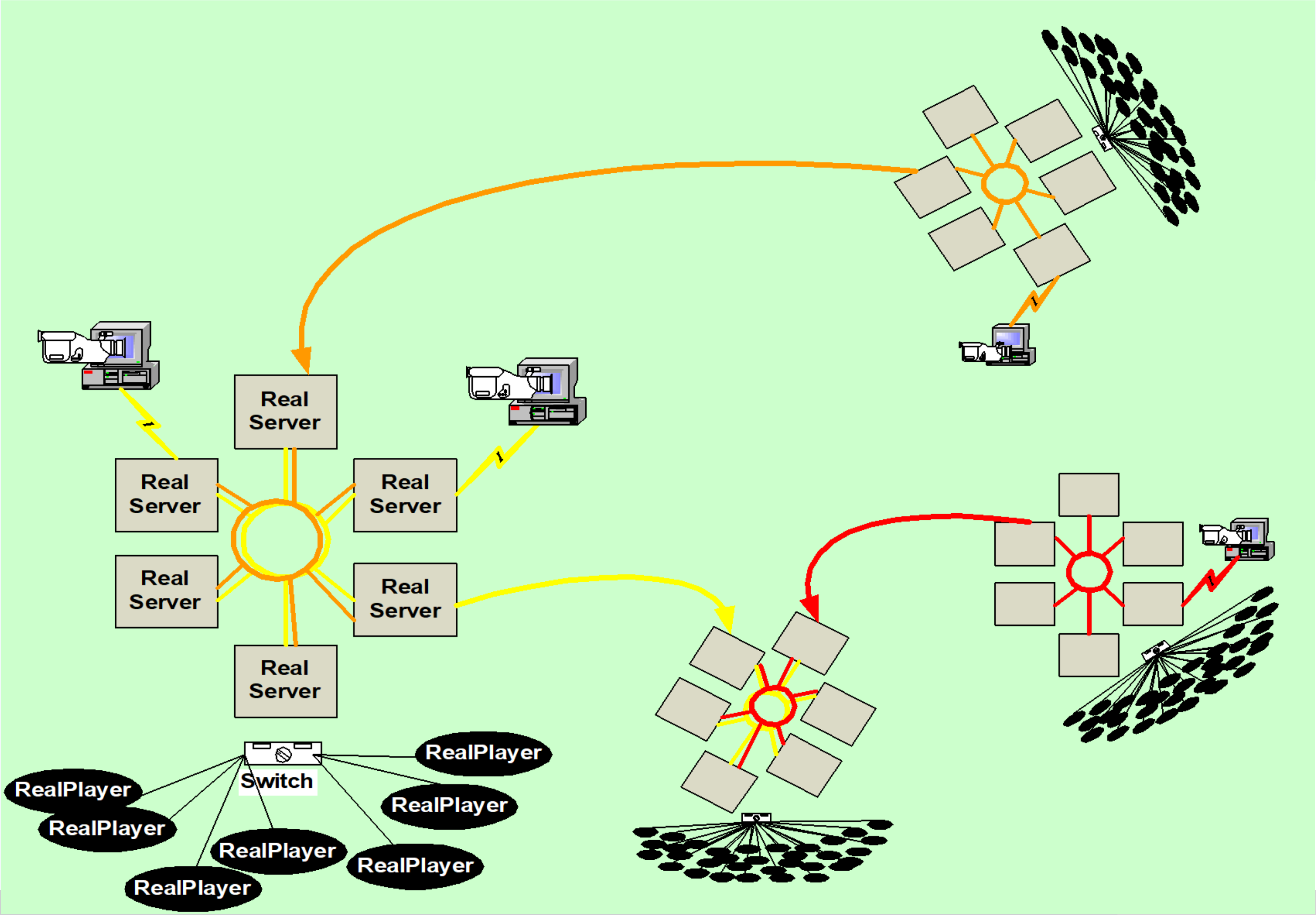
Very rigorous real-time delivery constrains: small startup latency, continuous delivery



A large number of proxies with: disk, memory, and CPU cycles

Diverse client access devices: computers, PDAs, cell-phones

Distributed Server Clustering



Section 2 -DHTs

- Distributed discovery services

What is a DHT?

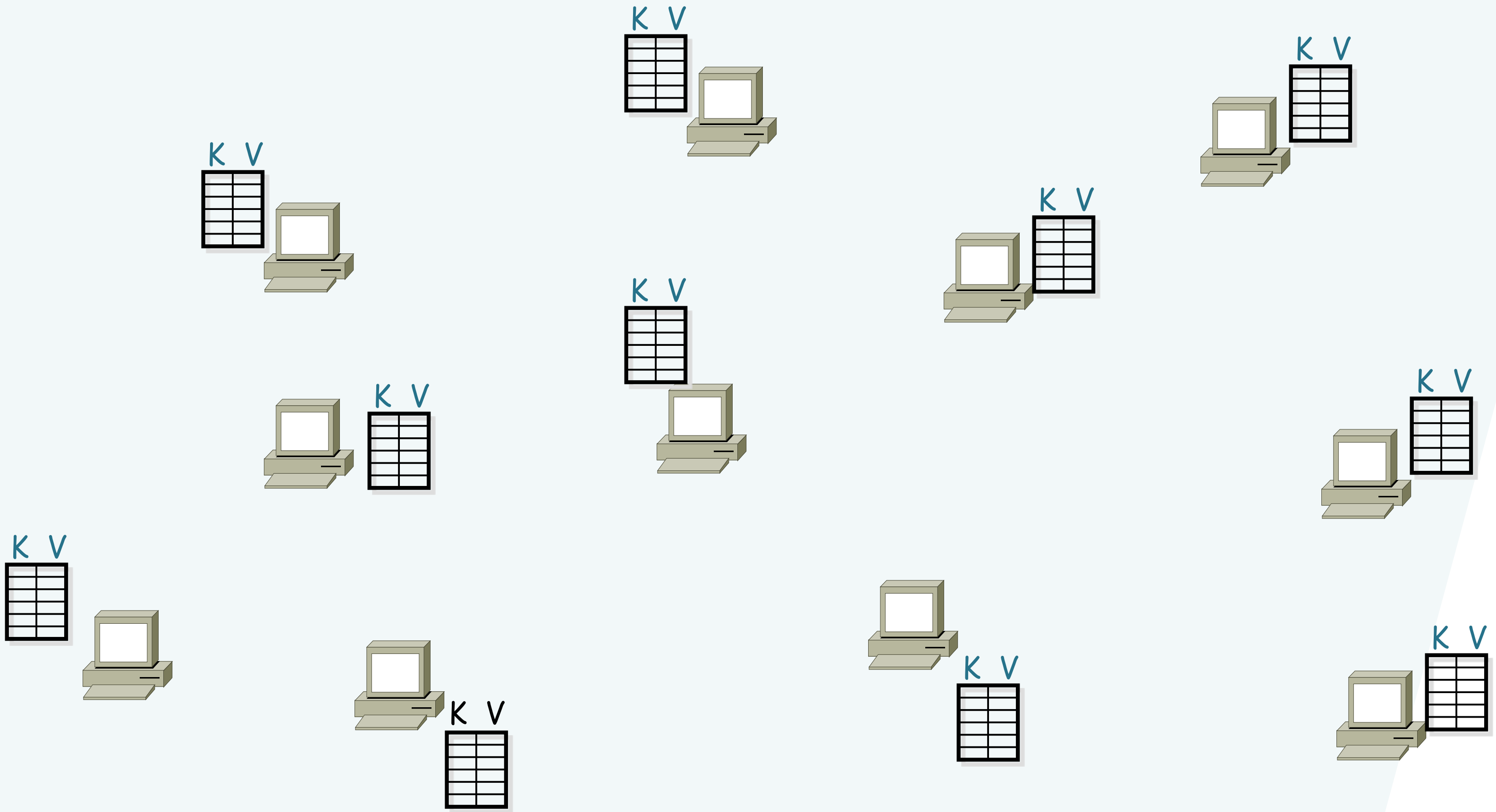
- Hash Table
  - data structure that maps “keys” to “values”
  - essential building block in software systems
- Distributed Hash Table (DHT)
  - similar, but spread across many hosts
- Interface
  - insert(key, value)
  - lookup(key)

How do DHTs work?

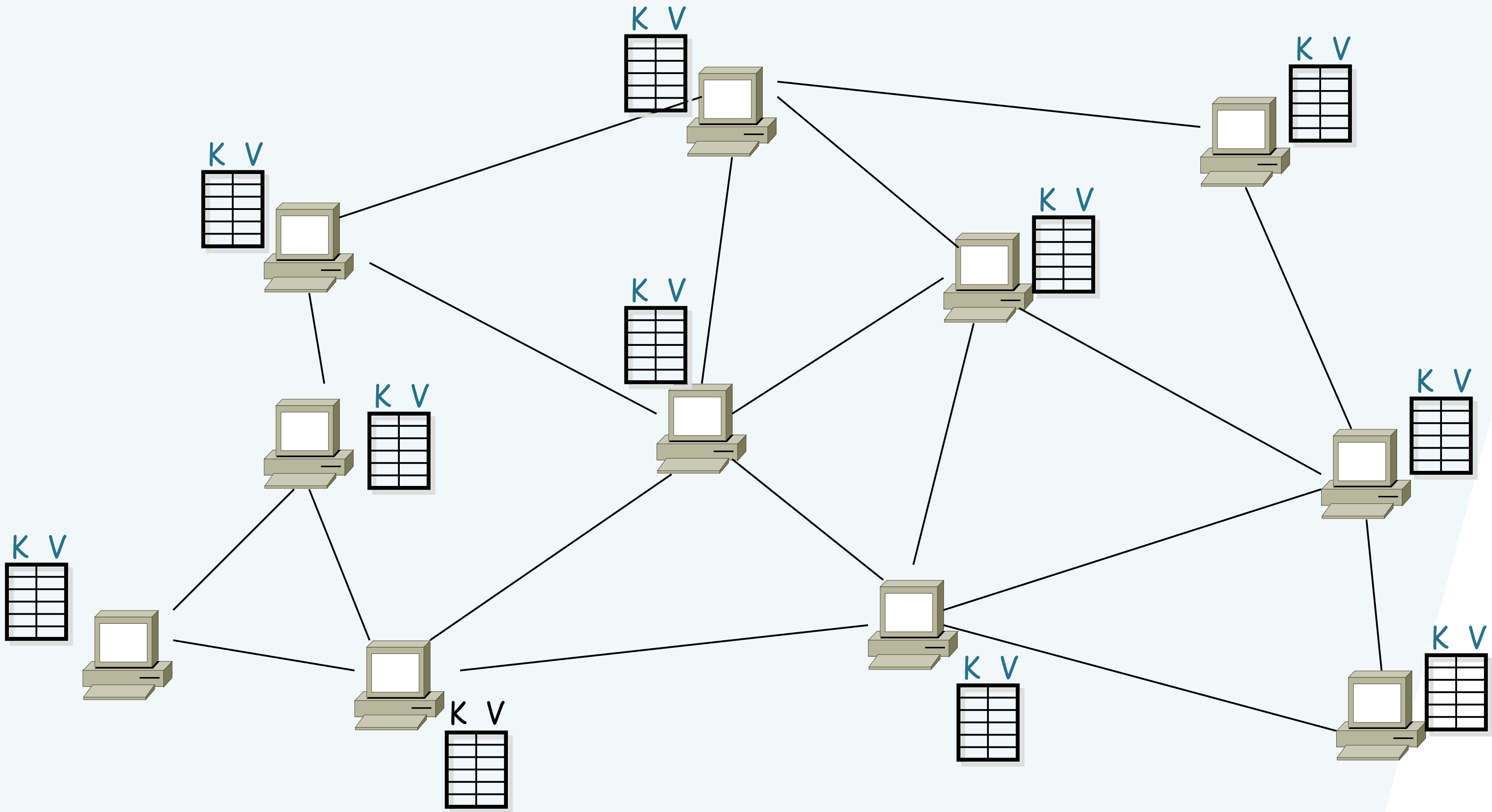
Every DHT node supports a single operation:

- Given *key* as input; route messages to node holding *key*
  - DHTs are *content-addressable*

DHT: basic idea

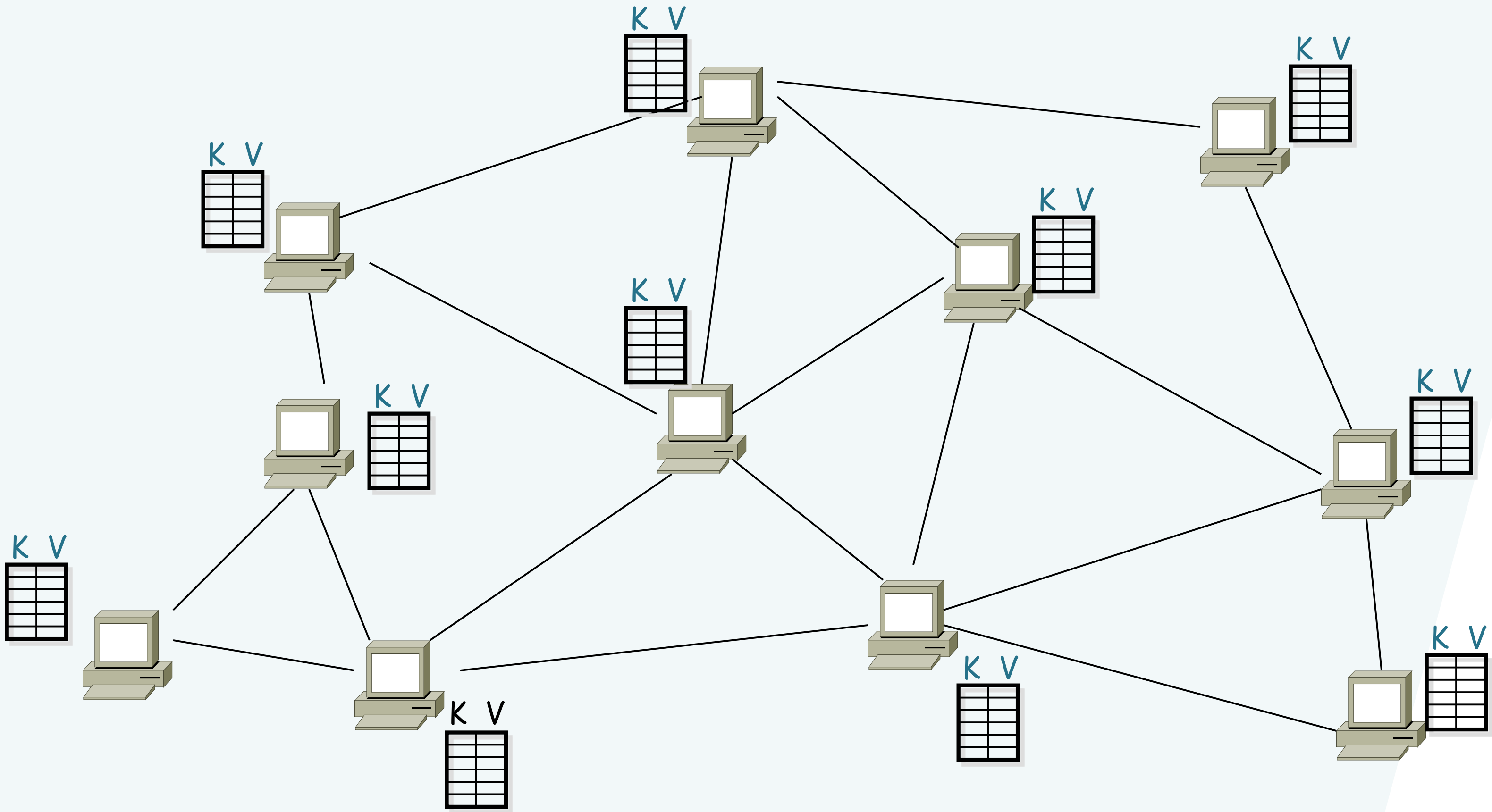


DHT: basic idea



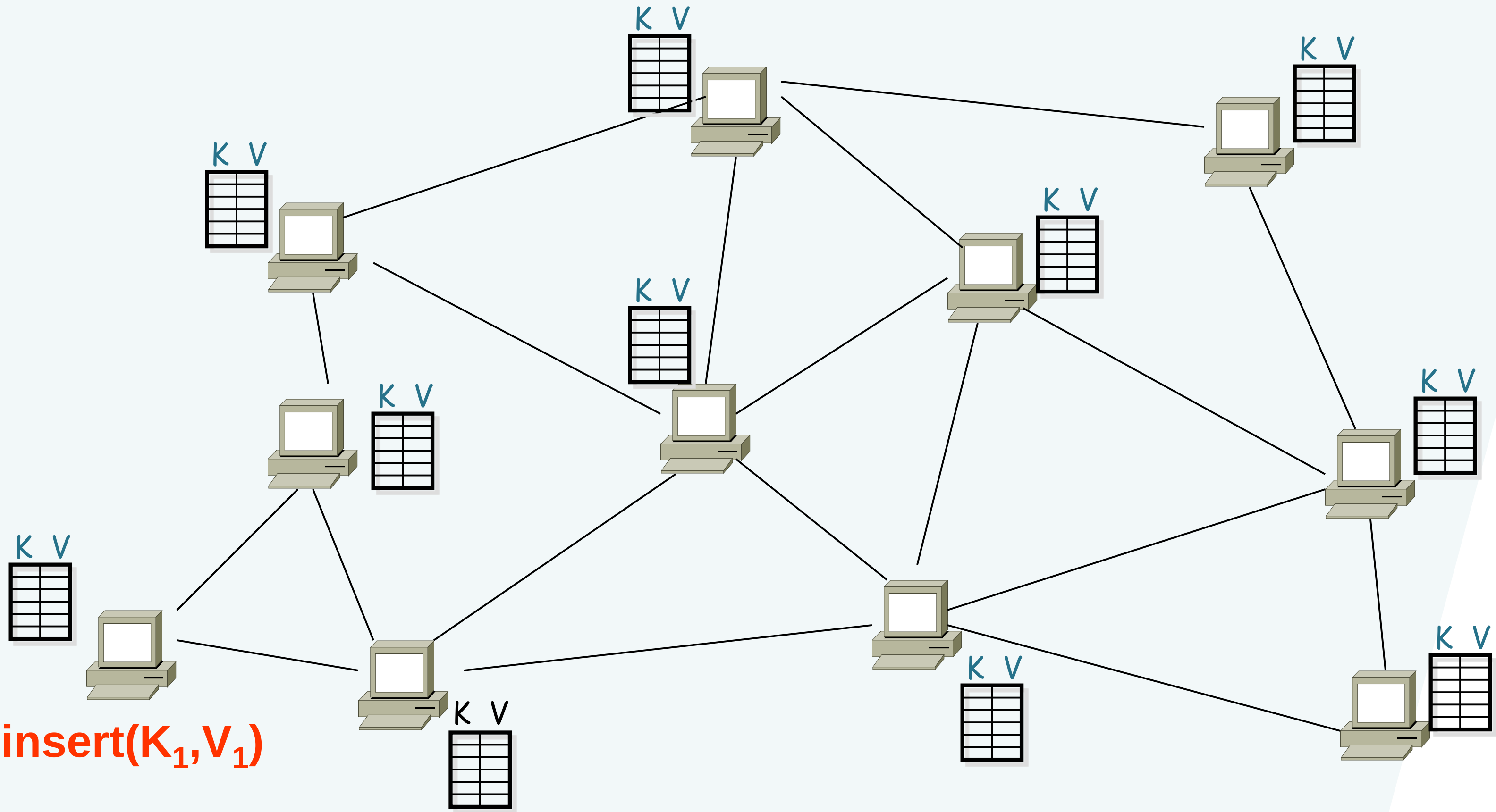
Neighboring nodes are “connected” at the application-level

DHT: basic idea



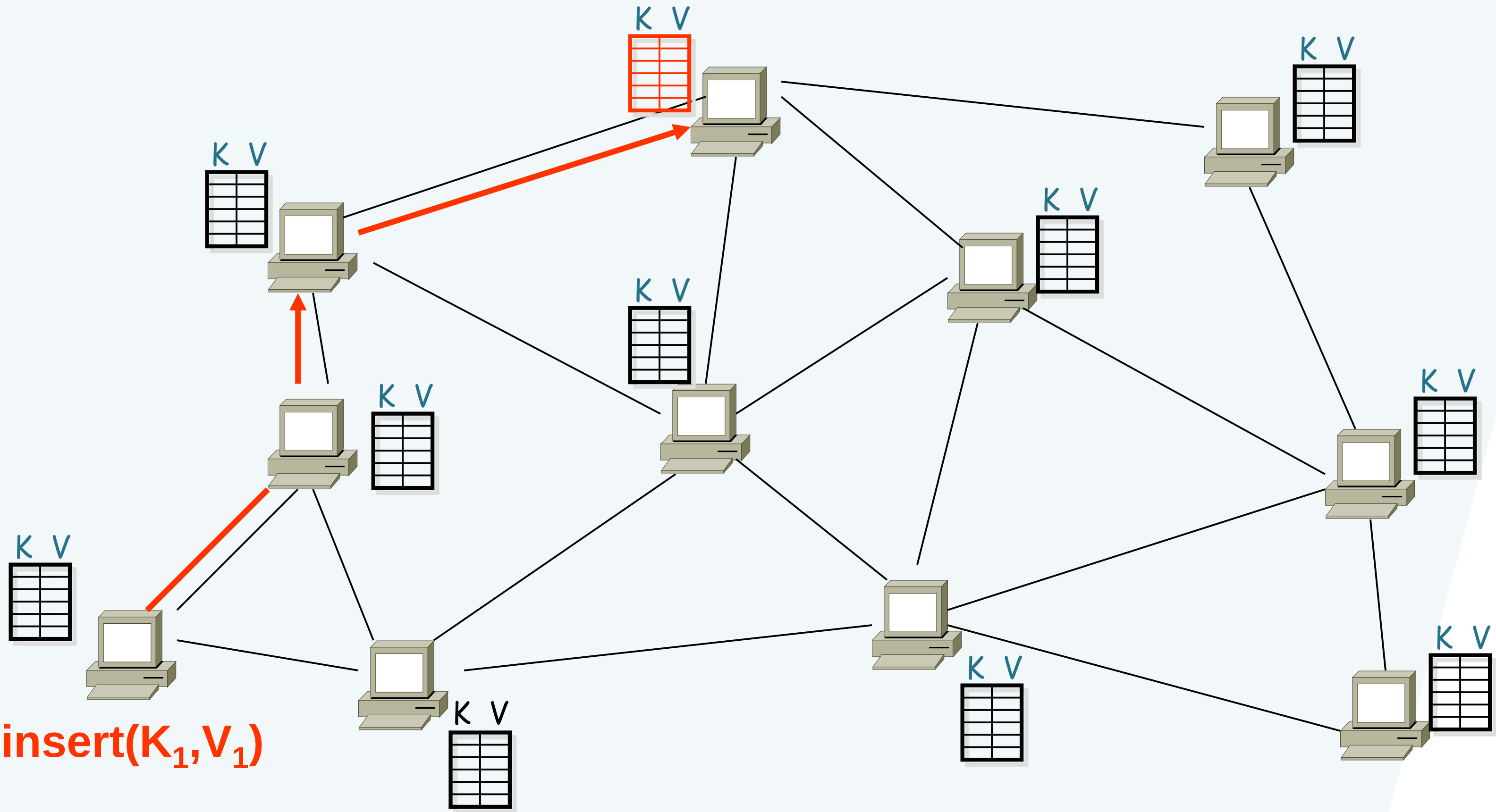
Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea



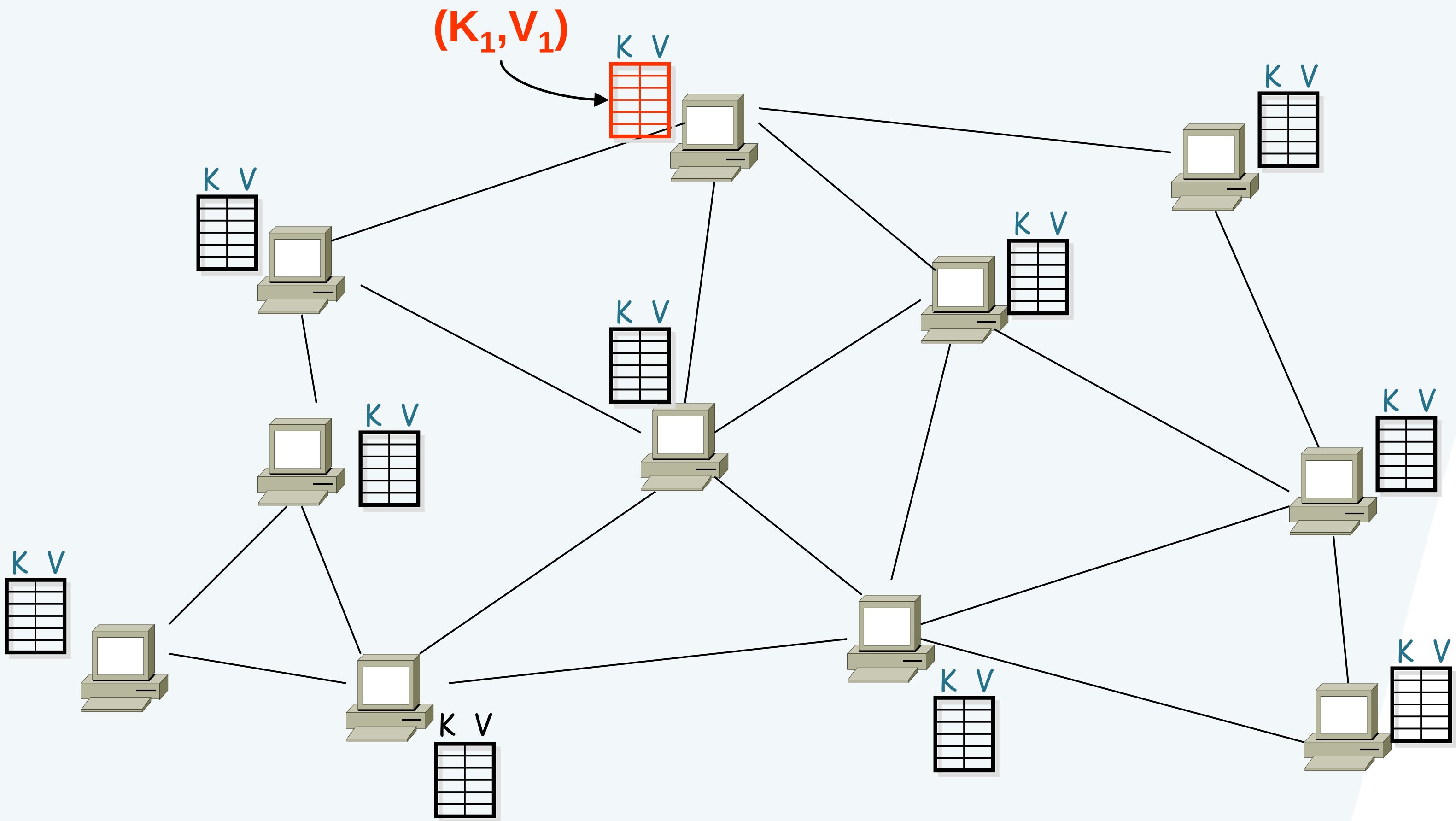
Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea



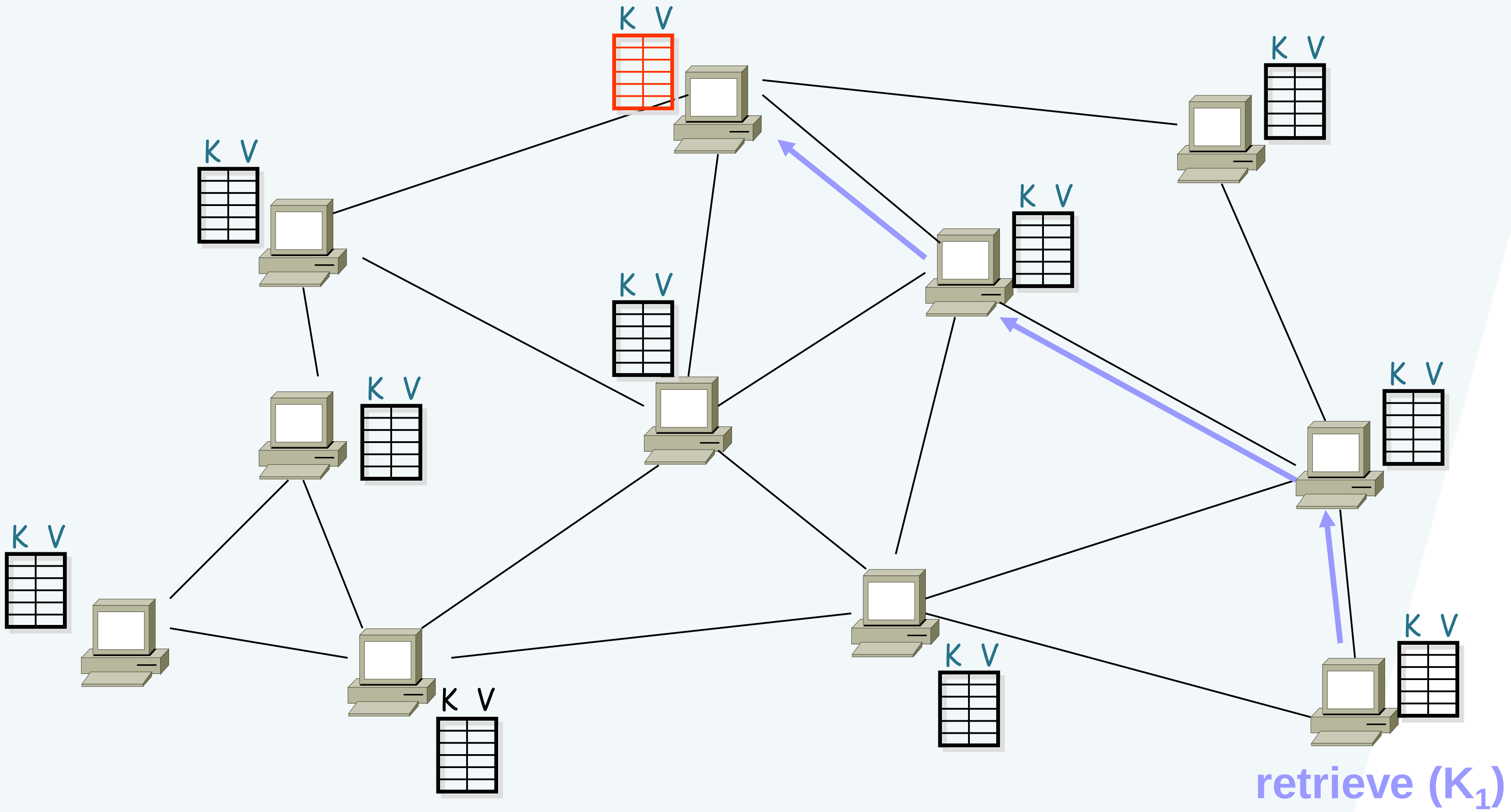
Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea



Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea



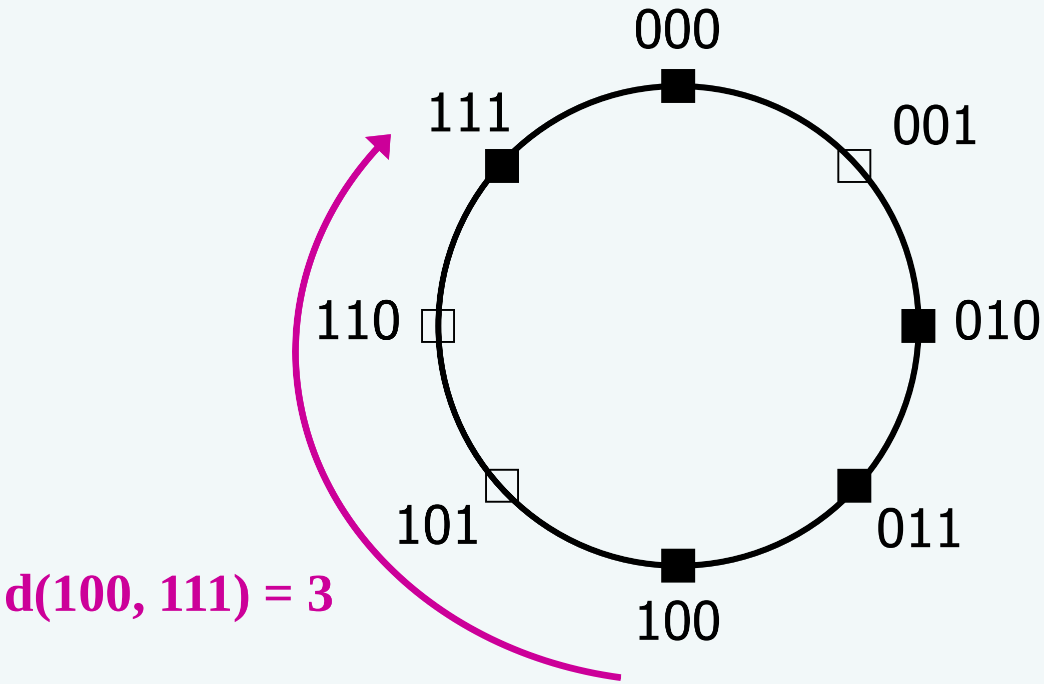
Operation: take *key* as input; route messages to node holding *key*

How to design a DHT?

- State Assignment:
  - what “(*key, value*) tables” does a node store?
- Network Topology:
  - how does a node select its neighbors?
- Routing Algorithm:
  - which neighbor to pick while routing to a destination?

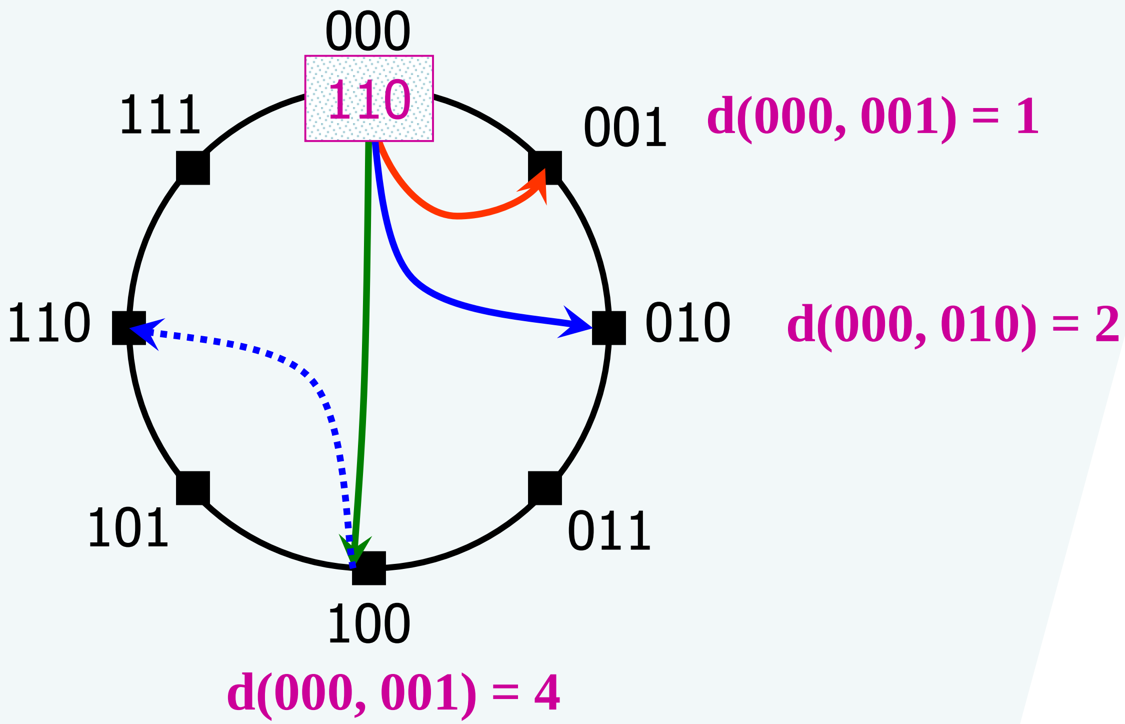
- Various DHT algorithms make different choices
  - CAN, Chord, Pastry, Tapestry, Plaxton, Viceroy, Kademlia, Skipnet, Symphony, Koorde, Apocrypha, Land, ORDI …

State Assignment in Chord DHT



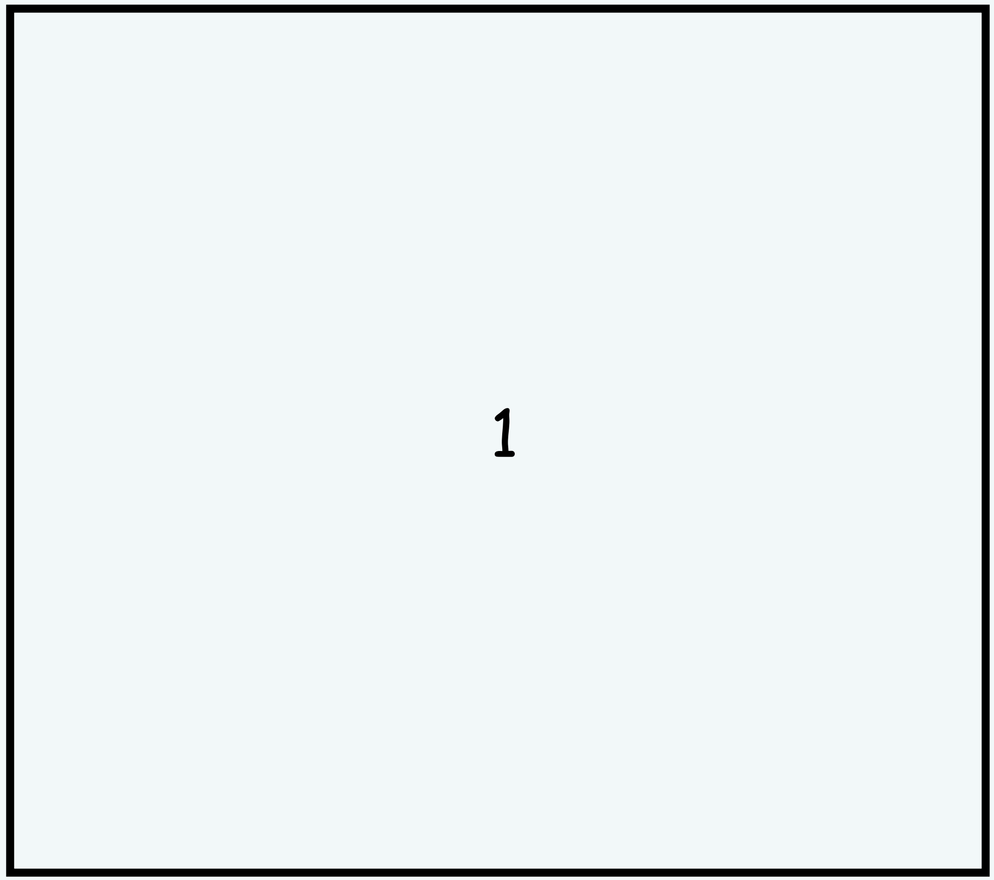
- Nodes are randomly chosen points on a clock-wise Ring of *values*
- Each node stores the *id space* (*values*) between itself and its predecessor

Chord Topology and Route Selection



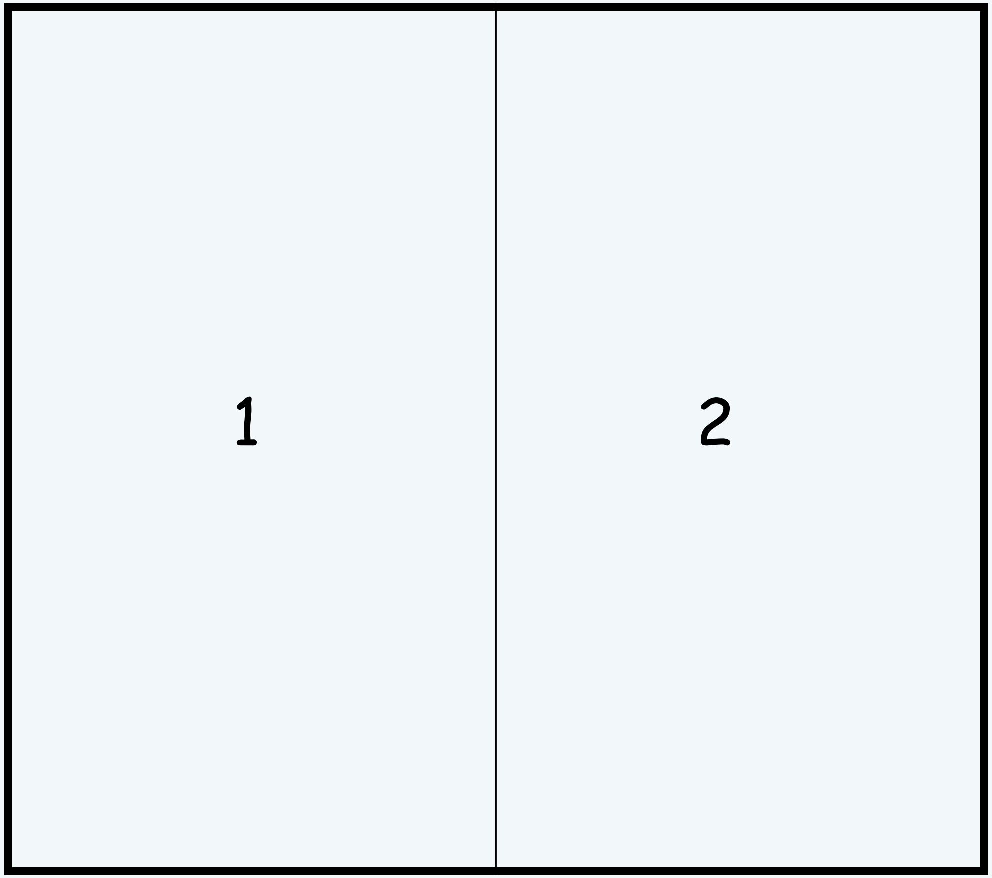
- Neighbor selection: **i**th neighbor at **2**^{i} distance
- Route selection: pick neighbor closest to destination

State Assignment in CAN



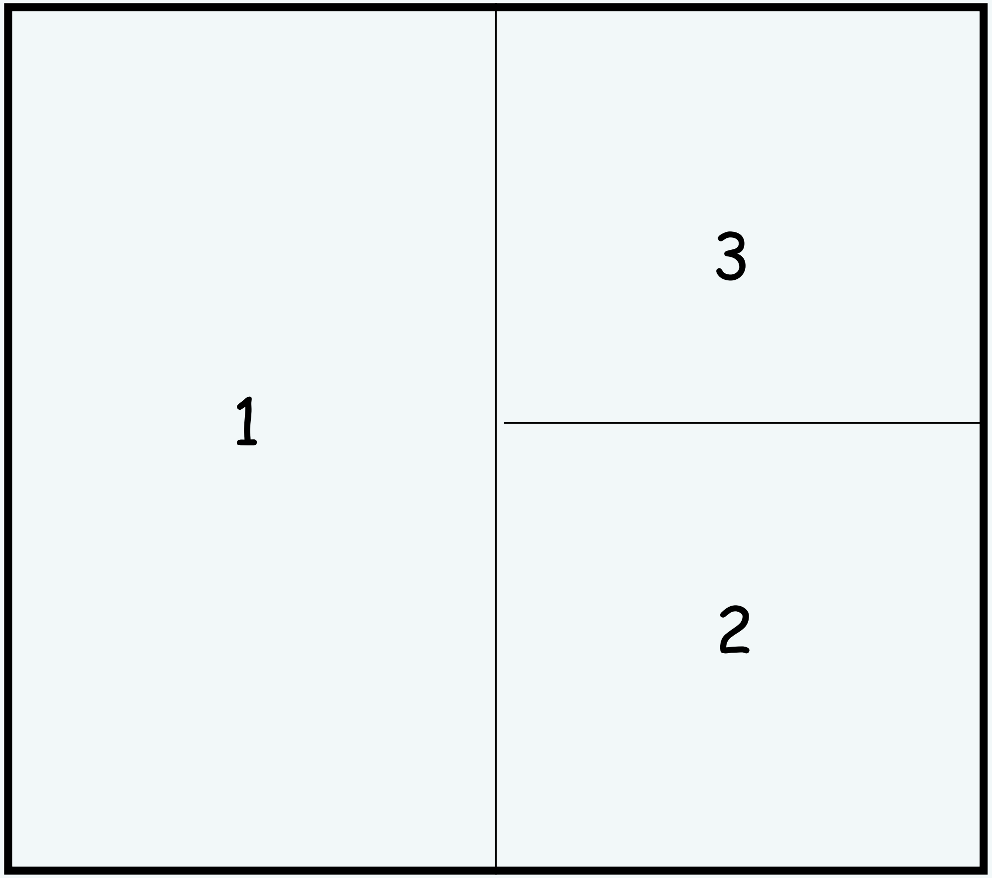
Key space is a virtual d-dimensional Cartesian space

State Assignment in CAN



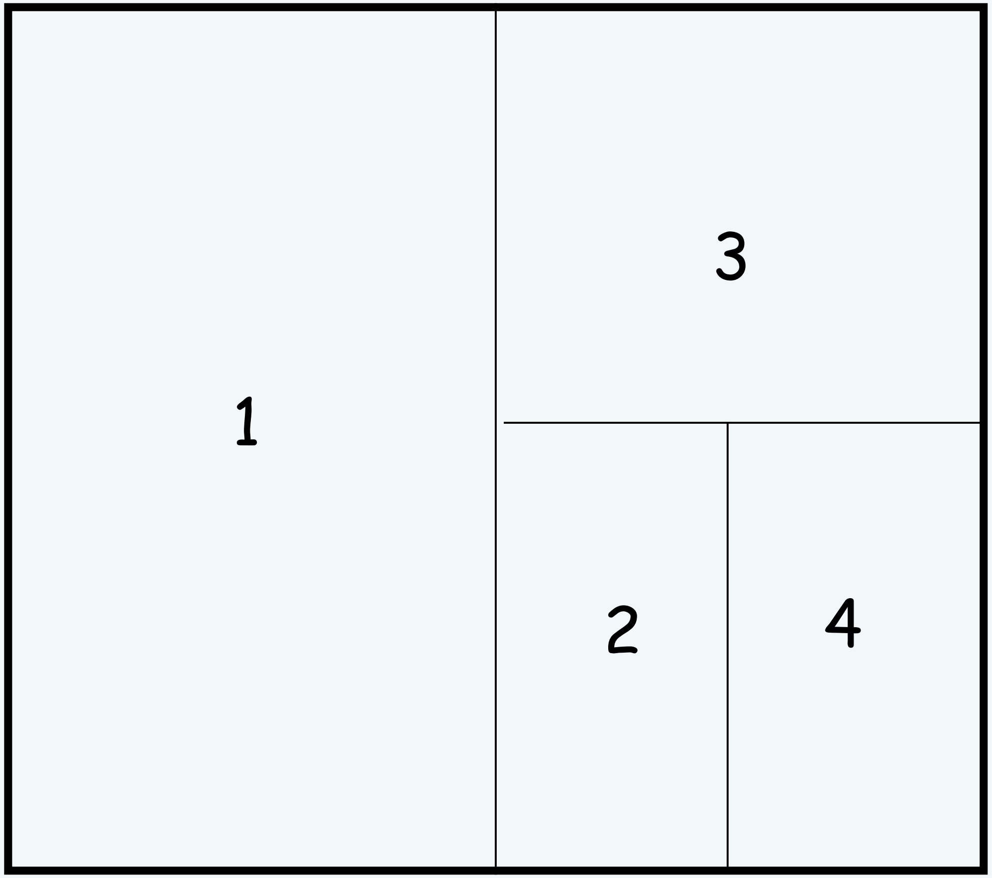
Key space is a virtual d-dimensional Cartesian space

State Assignment in CAN



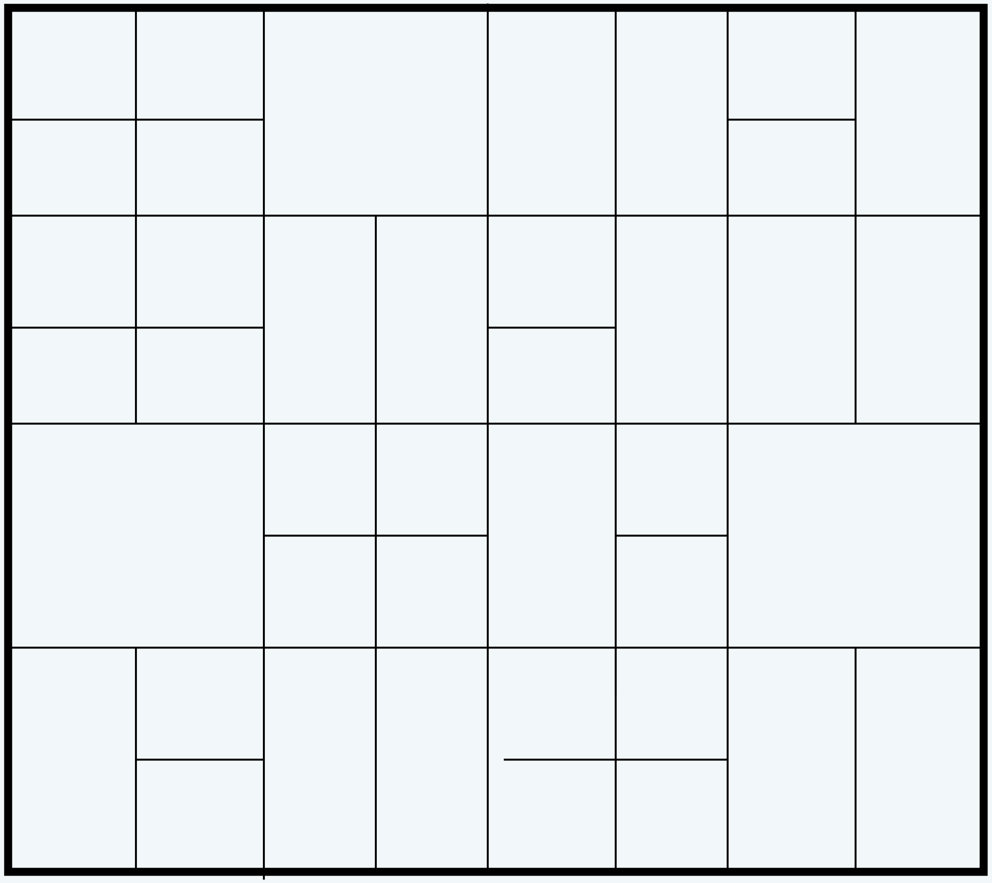
Key space is a virtual d-dimensional Cartesian space

State Assignment in CAN



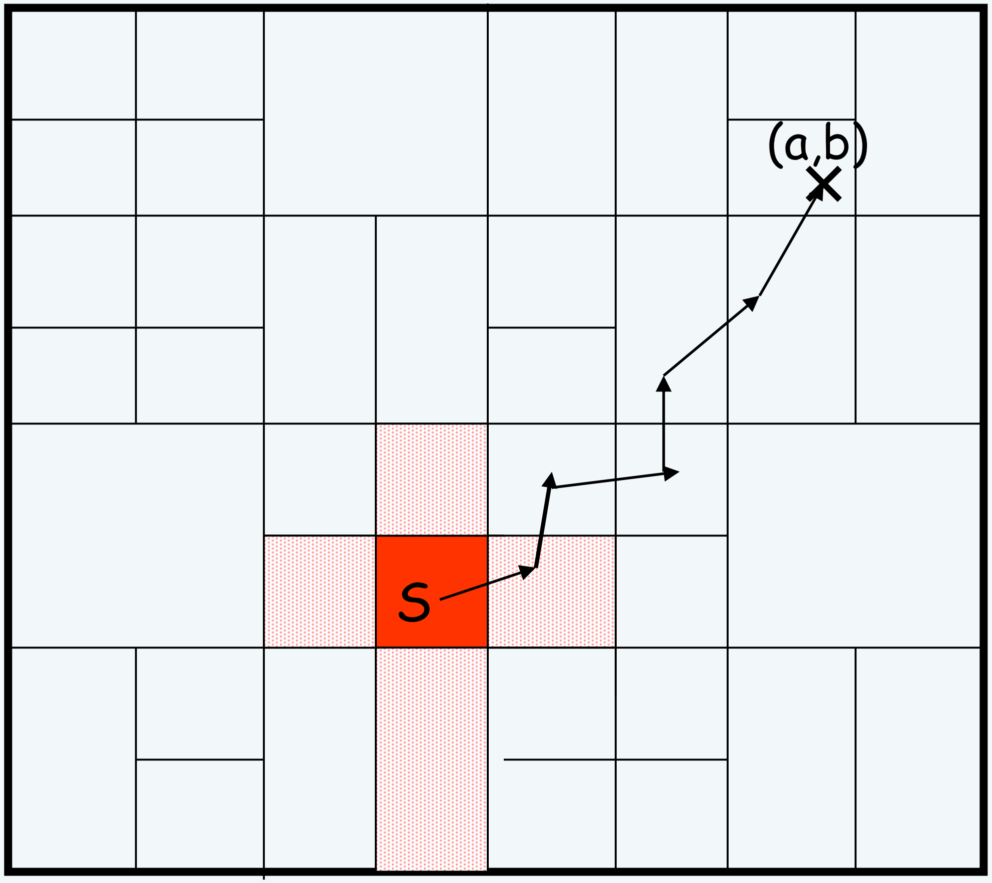
Key space is a virtual d-dimensional Cartesian space

State Assignment in CAN



Key space is a virtual d-dimensional Cartesian space

CAN Topology and Route Selection



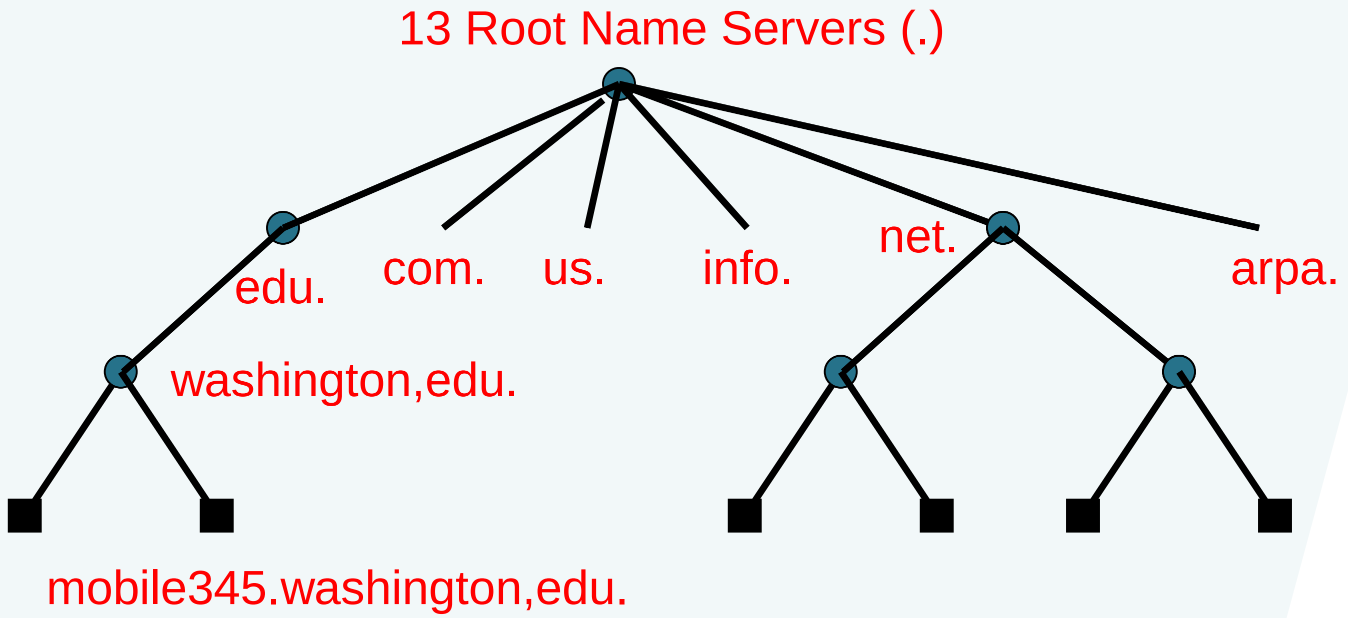
Route by forwarding to the neighbor "closest" to the destination

Interesting properties of DHTs

- Scalable
  - each node has O(*logN*) neighbors
  - hence highly robust to churn in nodes and data
- Efficient
  - lookup takes *O(logN)* time
- Completely decentralized and self-organizing
  - hence highly available
- Load balanced
  - all nodes are equal

Are DHTs panacea for building Scalable Distributed Systems?

Domain Name System Today



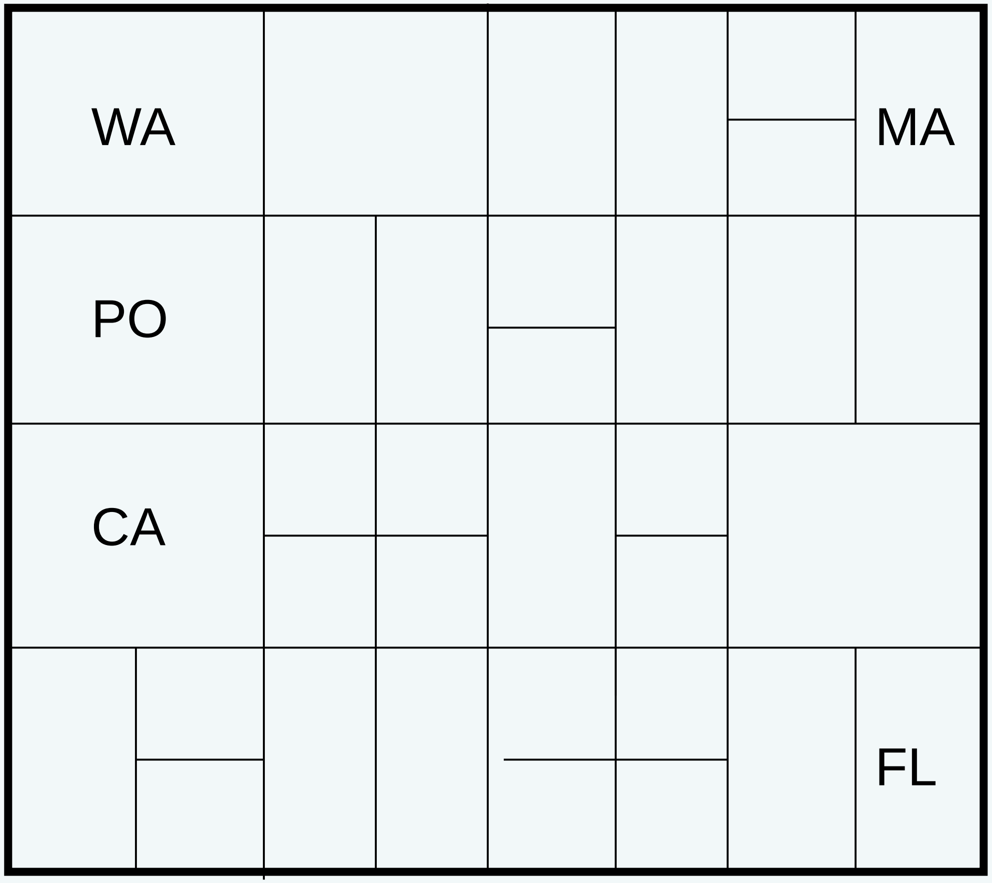
"Hierarchy is a fundamental way to accommodating growth and isolating faults "
-- Butler Lampson on Grapevine

Hierarchical DNS vs. DHT based DNS

Points of comparison

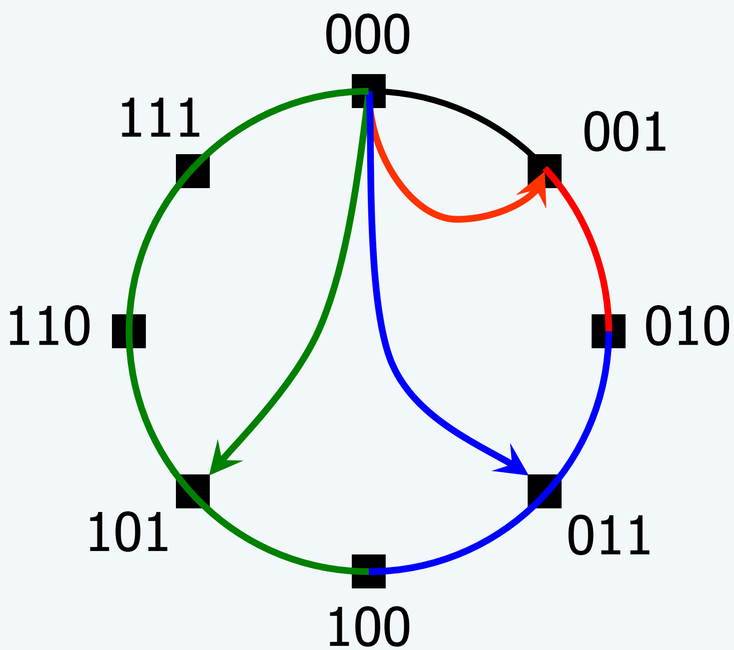
- Scalability: Number of neighbors per node
- Efficiency: Time taken per query
- Load Balancing: Per node state and lookup load
- Self-organization and Decentralization
- Fault isolation and Security

Topological Sensitivity in CAN DHT



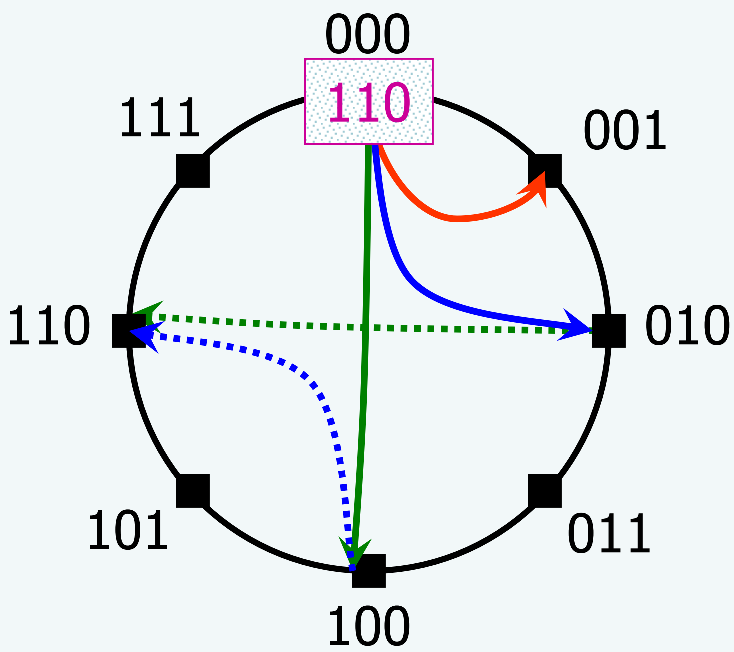
Insert nodes according to Virtual Coordinates derived from location

Topological Sensitivity in Chord DHT



- Chord algorithm picks **i^th^** neighbor at **2^i** distance
- A different algorithm picks **i^th^** neighbor from **$[2^i , 2^{i+1})$**

Topological Sensitivity in Chord DHT



- Chord algorithm picks neighbor closest to destination
- CFS algorithm picks the best of alternate paths

Section 3

- P2P streaming architectures

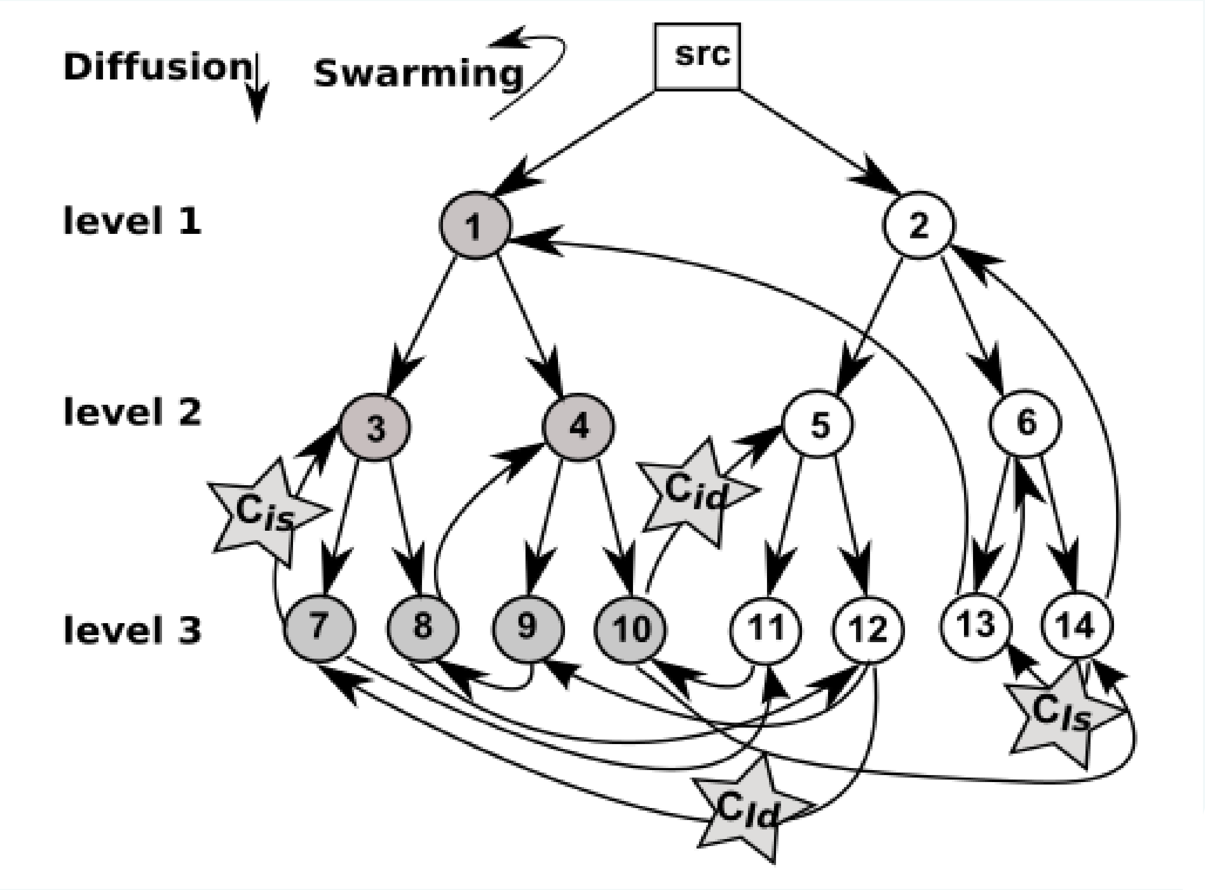
P2P streaming

- Using P2P overlay for streaming live media over network
- Participating end-systems (or peers) actively contribute their resources by forwarding their available content to their connected peers.
- Push based content delivery over multiple tree shaped overlays.
- The tree-based P2P streaming approach expands on the idea of end-system multicast by organizing participating peers into multiple diverse trees.
- Mesh-based approach uses swarming content delivery over a randomly connected mesh.

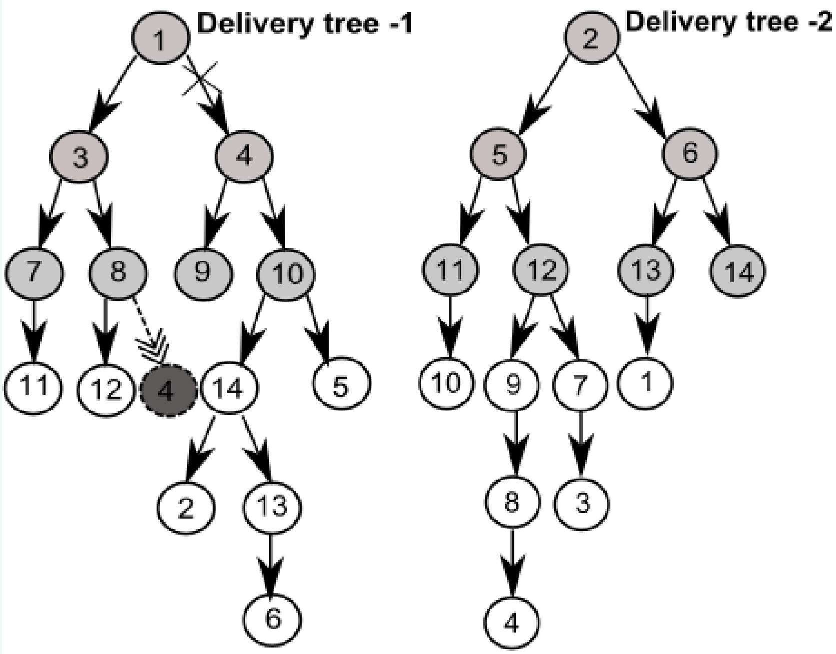
Terms

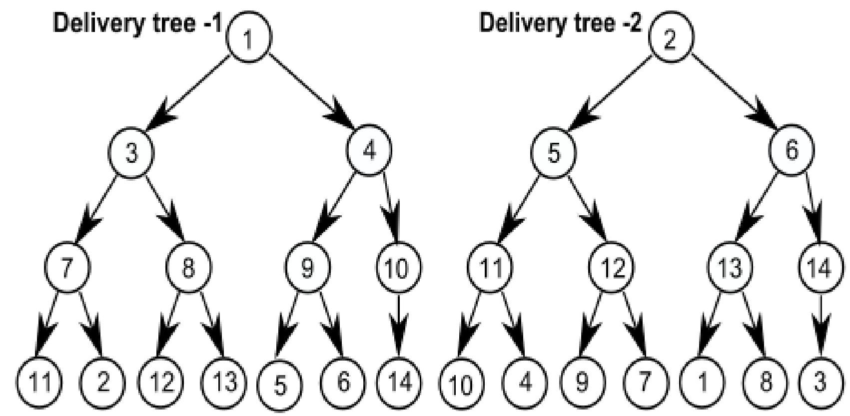
- Churn:
  - a peer can leave or join the p2p system at arbitrary time
- Deadlock:
  - In the presence of churn, a tree could become saturated and thus unable to accept any new leaf node.
- Content Bottleneck:
  - When a parent does not have sufficient number of useful packets for a child peer, the bandwidth of its congestion controlled connection to that child peer can not be fully utilized.
- Bandwidth Utilization:
  - ratio of the number of data packets to the total number of delivered packets.
- Average Quality:
  - the average number of descriptions (of Multiple Description Coded (MDC) content) it receives during a session.
- Multiple Description Coding (MDC):
  - Encoding streams into multiple sub-streams called description. Each description can be independently decoded. Furthermore, receiving multiple unique descriptions results in a higher quality.

Organized view of Random Mesh



Delivery Trees





Mesh – based approach

Tree – based appr

Tree Overlay Construction

- Peer decides number of trees to join based on its access link bandwidth
- Each peer is placed as an internal node in only one tree and as a leaf node in other trees.
- Join:
  - peer contacts the bootstrapping node to identify a parent in the desired number of trees
- Leave:
  - subtree nodes rejoin the tree
- Balance tree:
  - peer is added as an internal node to the tree that has the minimum number of internal nodes.
- Short tree:
  - a new internal node is placed as a child for the node with the lowest depth

Mesh Overlay Construction

- Participating peers form a randomly connected overlay
- Each peer tries to maintain a certain number of parents (i.e., incoming degree)
- Each peer serves a specific number of child peers (i.e., outgoing degree).
- Upon arrival, a peer contacts a bootstrapping node to receive a set of peers that can potentially serve as parents.

…Mesh Overlay Construction

- The bootstrapping node maintains the outgoing degree of all participating peers. Then, it selects a random subset of peers that can accommodate new child peers in response to an incoming request for parents.

- Individual peers periodically report their newly available packets to their child peers and request specific packets from individual parent peers

- A parent peer periodically receives an ordered list of requested packets from each child peer, and delivers the packets in the requested order. The requested packets from individual parents are determined by a packet scheduling algorithm at each child peer.

Similarities

- Both approaches leverage MDC to accommodate the bandwidth heterogeneity among participating peers.
- Superimposed view of multiple diverse trees is same as directed random mesh overlays.
- Content delivery in both enables individual peers to receive different pieces of content.
- All peers receive data from multiple parents and send it down to different child peers.
- Both require peers to maintain a loosely synchronized playout time that is sufficiently (τ seconds) behind source's playout time.

Differences

Tree based approach	Mesh based approach
Delivery tree for all packets of a particular description is corr overlay tree for that description	Delivery tree for individual packets is dynamically shaped as packet travels through the overlay. When a connection has lower bandwidth than description b/w, its descendant peers can still receive packets from alternate path from other parents.
Push based content delivery over multiple tree shaped overlays extending idea of end-system multicast [1]	content delivery over a randomly connected mesh extending file swarming mechanisms like in bitTorrent.
Inferior performance due to static mapping of content to a particular tree. The placement of each peer as an internal node in one tree and as a leaf in all other trees.	Superior performance as there mutiple type of connections among peers and parents. More dynamic to increase in description bandwidth.
Sweet spot for peer bandwidth where it can effectively utilize available resources and provide the desired quality.	Swarming content delivery couples push content reporting with pull content requesting.