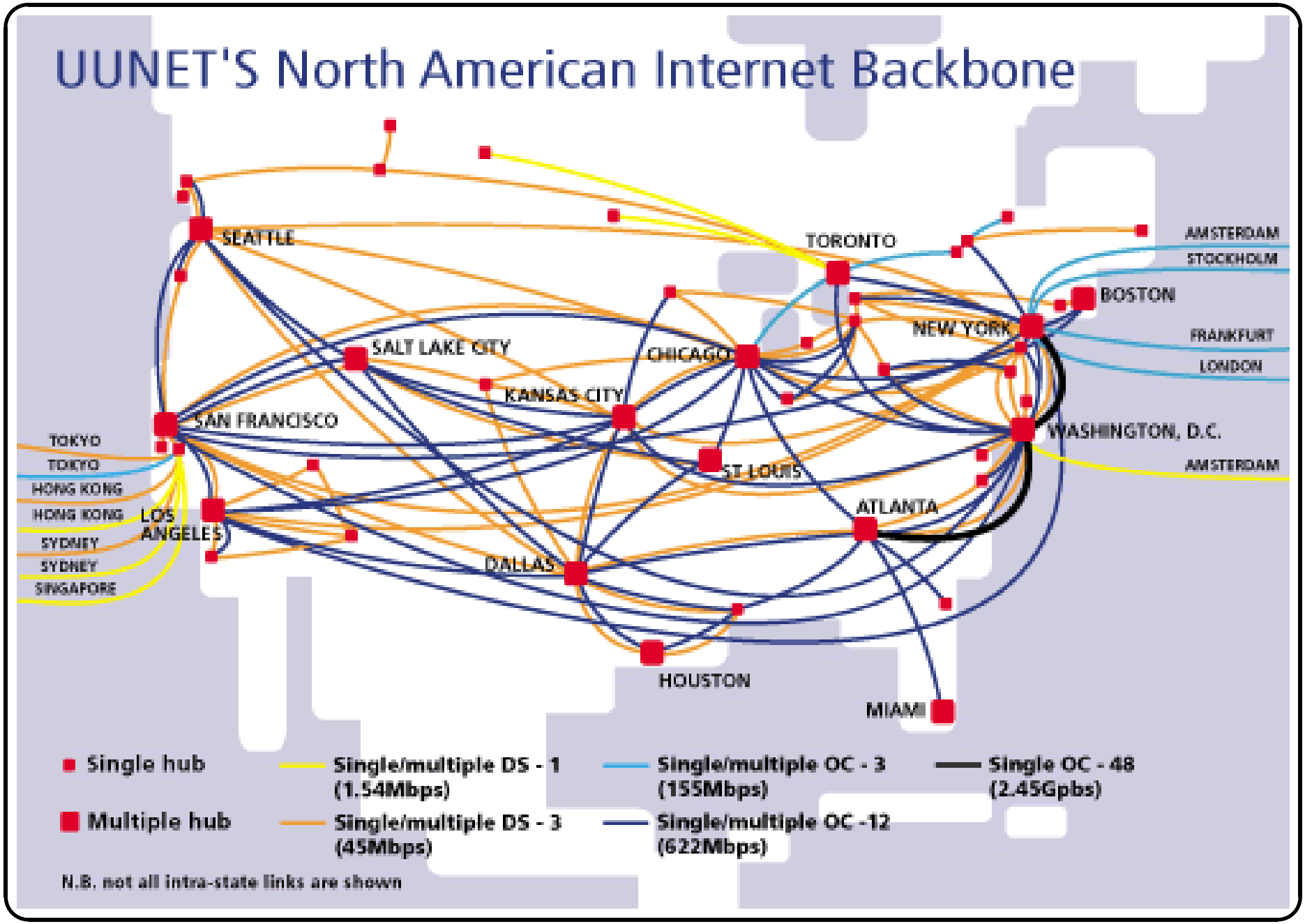
Data Networks

UdS and IMPRS-CS

Lecture 23: Overlay networks

Idealized View of the Internet

- A collection of IP routers and point-to-point physical links connecting routers
- Point-to-point links between two routers are physically direct connections
  - Copper wire, coax cable or fiber laid from one router to another


UUNET'S North American Internet Backbone
SEATTLE
TORONTO
AMSTERDAM
STOCKHOLM
BOSTON
NEW YORK
FRANKFURT
LONDON
SALT LAKE CITY
CHICAGO
KANSAS CITY
SAN FRANCISCO
WASHINGTON, D.C.
AMSTERDAM
TOKYO
TOKYO
HONG KONG
HONG KONG
SYDNEY
SYDNEY
SINGAPORE
LOS ANGELES
ST LOUIS
ATLANTA
DALLAS
HOUSTON
MIAMI
Single hub
Multiple hub
Single/multiple DS - 1 (1.54Mbps)
Single/multiple DS - 3 (45Mbps)
Single/multiple OC - 3 (155Mbps)
Single/multiple OC -12 (622Mbps)
Single OC - 48 (2.45Gpbs)
N.B. not all intra-state links are shown

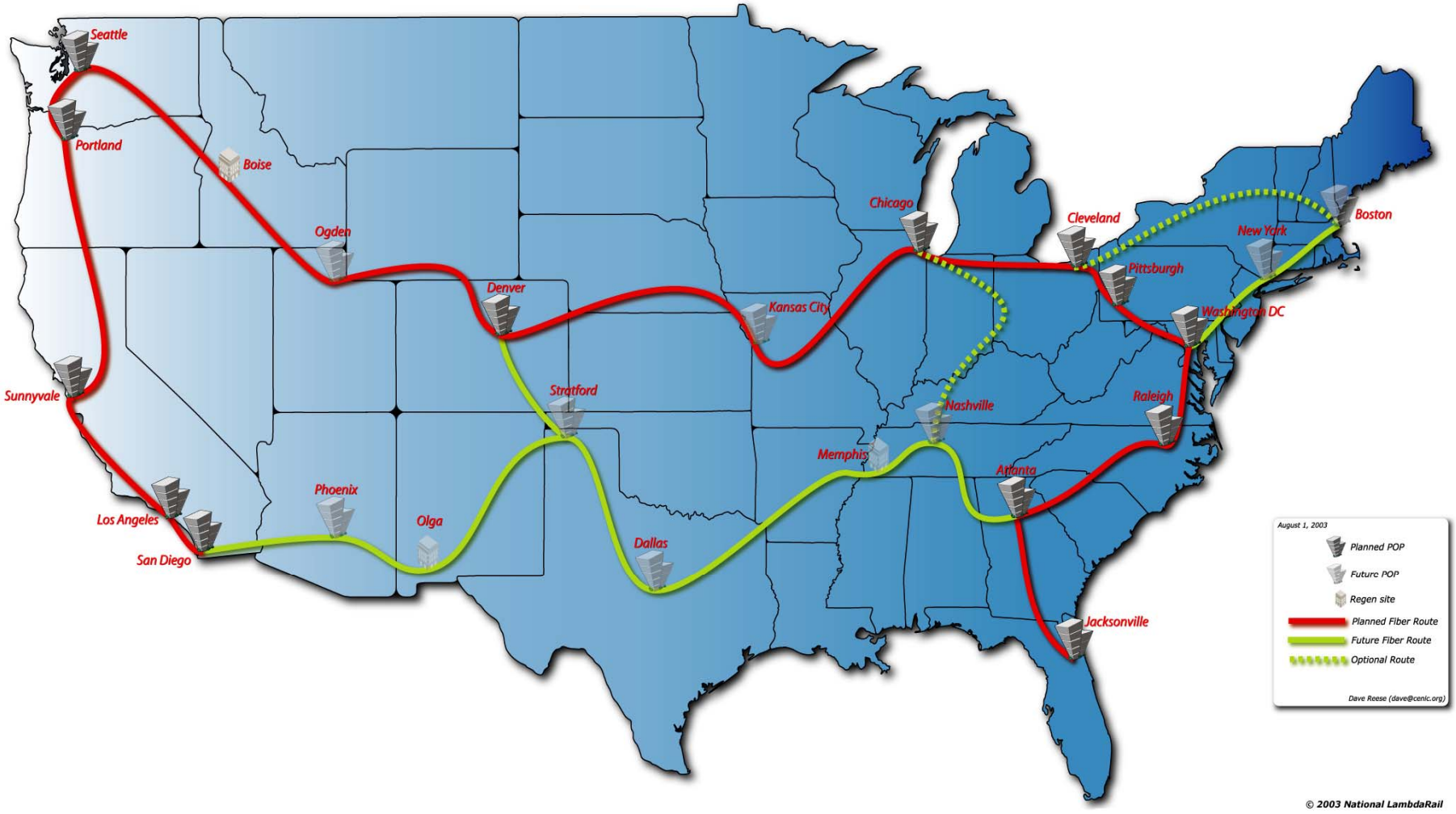
Reality

- Laying Fiber/wire is subject to practical constraints
  - You can't just dig up the ground everywhere
  - Right-of-way issue
  - Most fibers are laid along railroads
- Physical fiber topology often very far from the topology you want

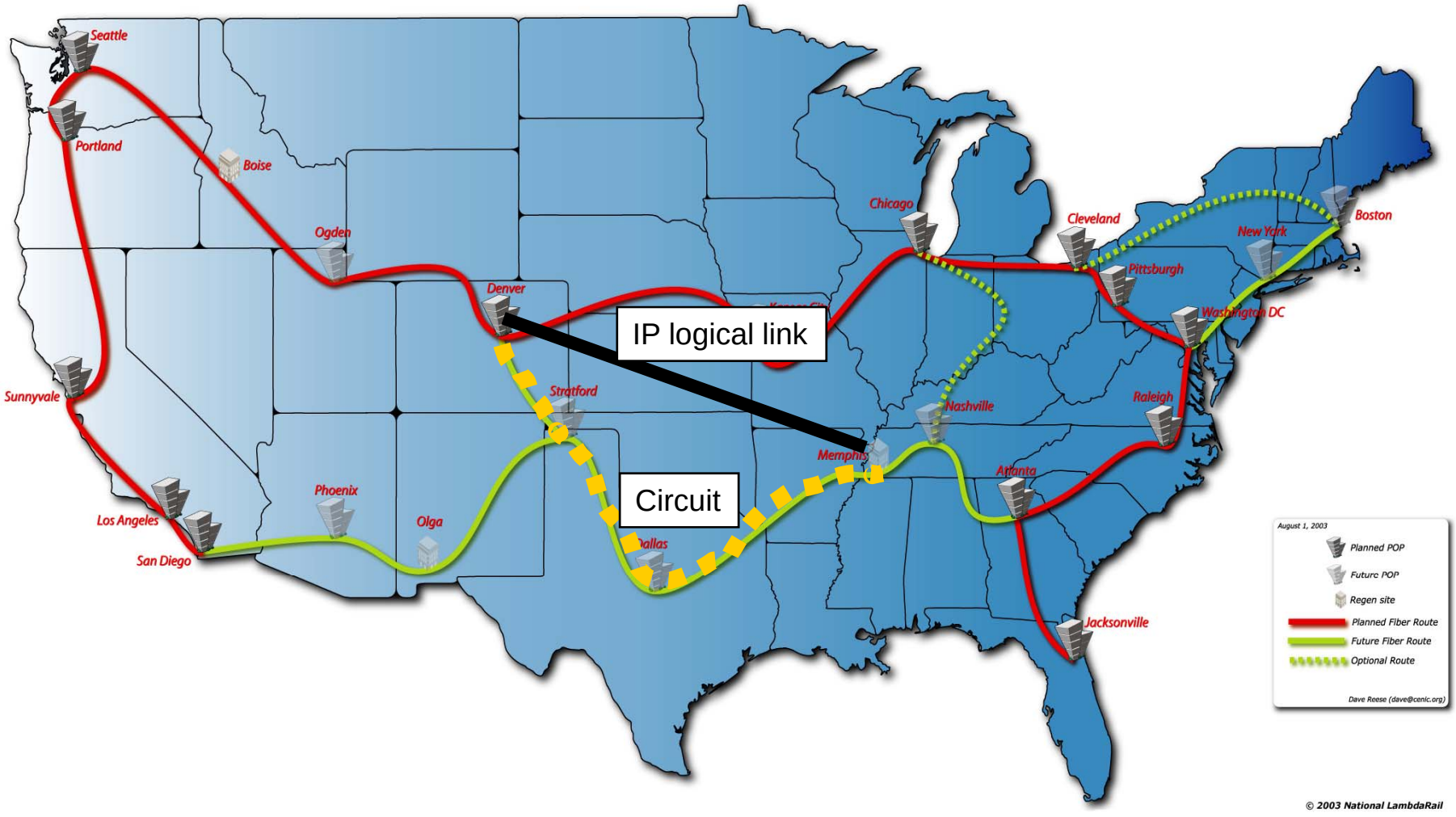
- IP Internet is over-laid on top of this physical fiber topology
- IP Internet topology is only logical!

- Concept: IP Internet is an overlay network

E.g. National Lambda Rail Project – Fiber Topology

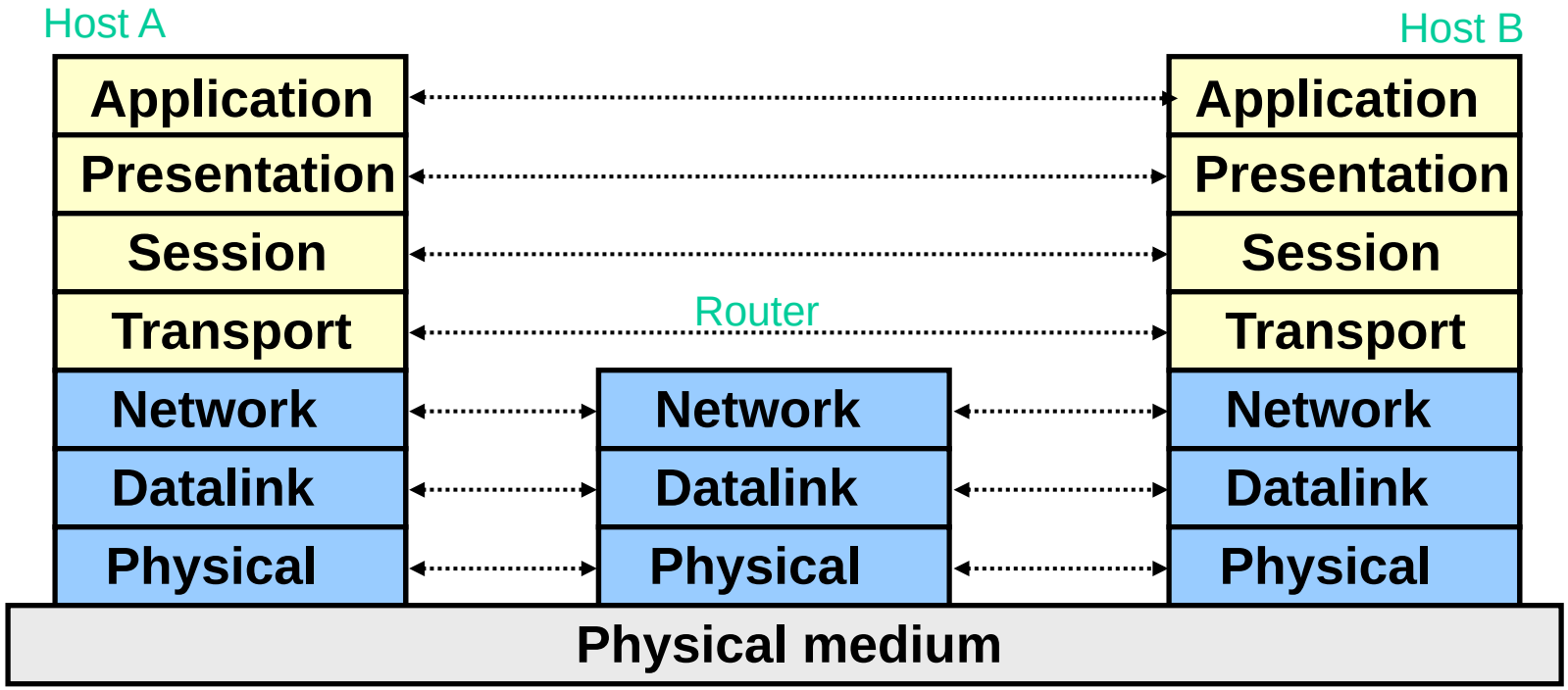


E.g. An IP logical link overlaid on a circuit



Made Possible by Layering

- Layering hides the detail of lower layer from higher layer
- IP operates on datalink layer (say ATM or SONET) logical topology
- ATM/SONET creates point-to-point circuits on the fibers



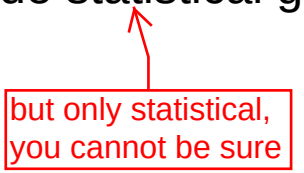
Overlay networks

- Overlay is a general concept
  - You can keep overlaying one network on another, it's all logical
- IP Internet overlays on top of physical topology
  - Why stop here?

- Something else can overlay on top of IP Internet
  - Use IP *tunnels* to create yet another logical topology
  - E.g. Virtual Private Networks (VPNs)

Reasons to Overlay Upon the IP Internet

- IP provides basic best effort datagram service
- Some functions you may want in a network but not be supported
- Like what?
  - Multicast
  - Reliable performance-based routing
  - Content-based addressing, content distribution, caching
  - Security
- Can we build an overlay network upon IP that provides QoS guarantees?
  - Overlay links must have guaranteed performance characteristics, otherwise, the overlay network cannot guarantee anything!
  - May be able to provide statistical guarantees



Unicast Routing Overlay

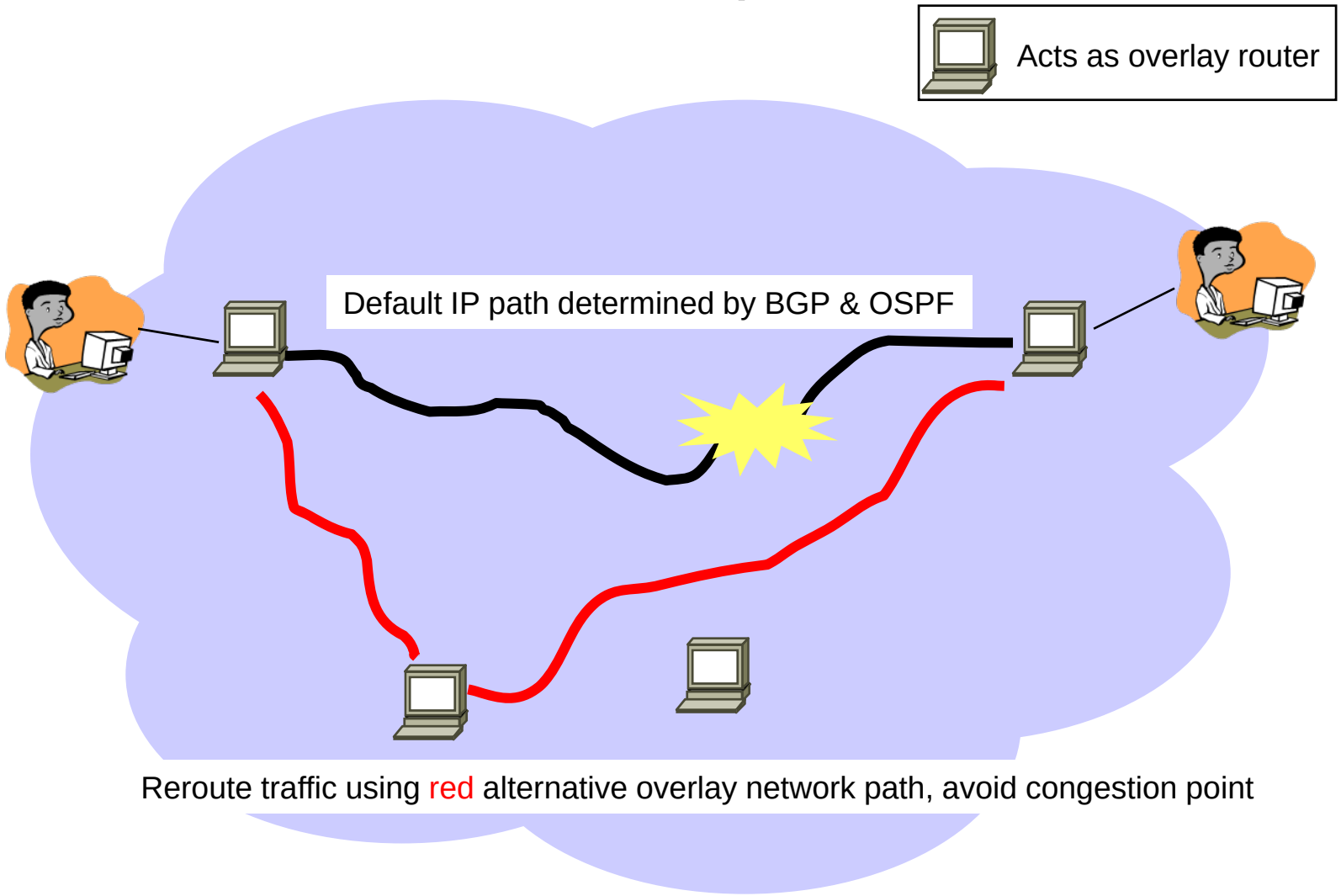
- Internet routing is built upon Intra-domain and Inter-domain router protocols
  - OSPF/RIP; BGP
- OSPF/RIP routing based on shortest link weight routing
  - Link weights are typically very static
  - Does not necessarily give you best performance path (delay, throughput, loss rate)
- BGP routing based mostly on policy
  - Policy may have nothing to do with performance
  - BGP very slow to react to failure
  - no reaction to high loss rate

Resilient Overlay Network (RON)

- Install N computers all over the place on the Internet
- Each computer acts as an overlay network router
  - Between each overlay router is a *IP tunnel* (logical link)
  - Logical overlay topology is all-to-all (N^2)
- Computers actively measure each logical link in real time for
  - Packet loss rate, latency, throughput, etc.
- Route overlay network traffic based on measured characteristics

- Able to consider multiple paths in addition to the default IP Internet path given by BGP/OSPF

Example


Acts as overlay router
Default IP path determined by BGP & OSPF
Reroute traffic using red alternative overlay network path, avoid congestion point

Potential Problems…

- Scalability of all these network measurements!
  - Overhead
  - Interference of measurements?
  - What if everyone has his/her own overlay network doing this?
- Stability of the network? Oscillation? Keep rerouting back and forth?
- How much can you really gain?
  - In delay/bandwidth, may not be that much
  - But is much faster to react to complete link failures than BGP

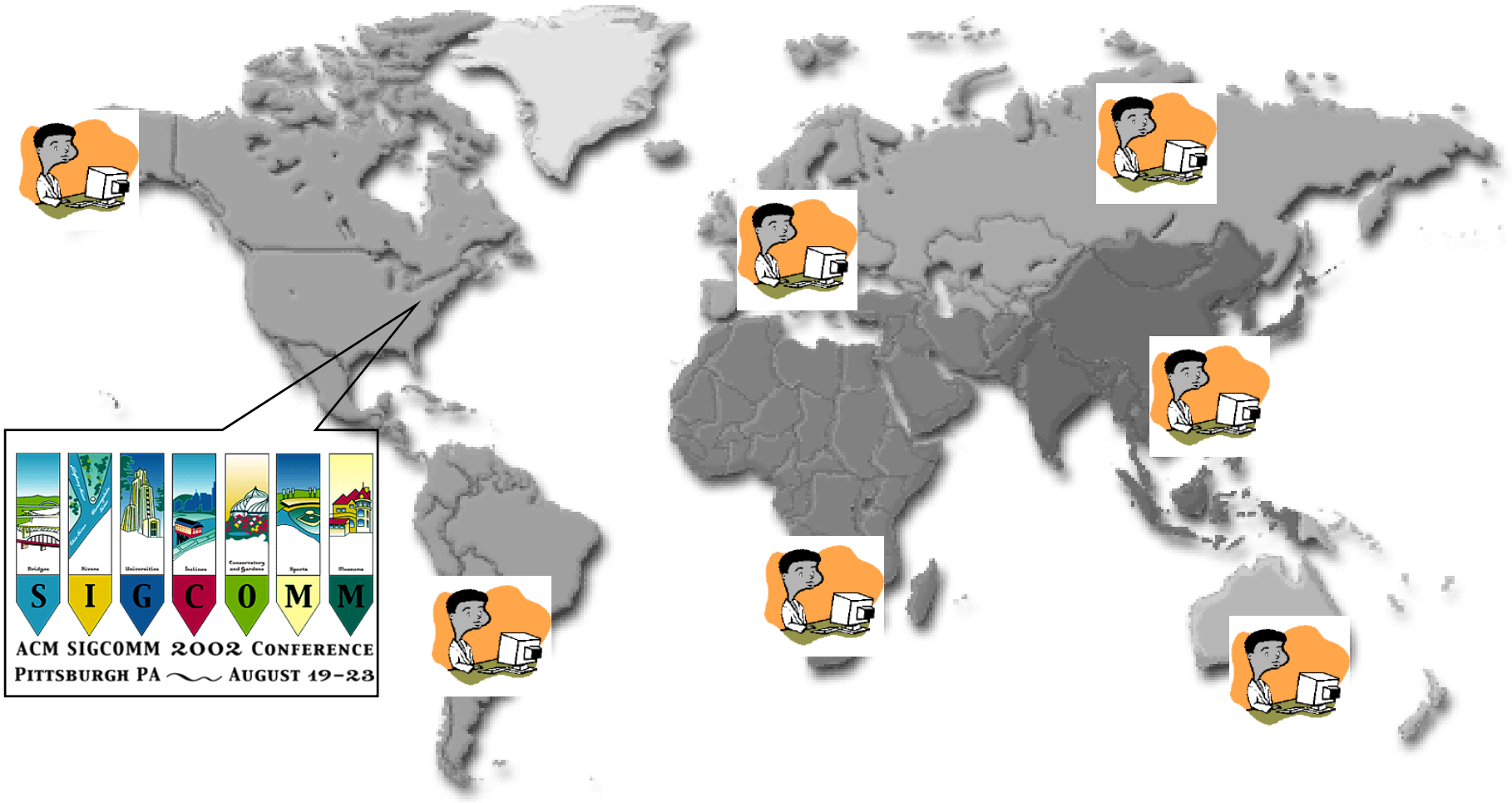
Multicast Overlay

- IP multicast provides one-to-many packet delivery
- IP multicast routers maintain group membership, duplicate packets appropriately and send to all members

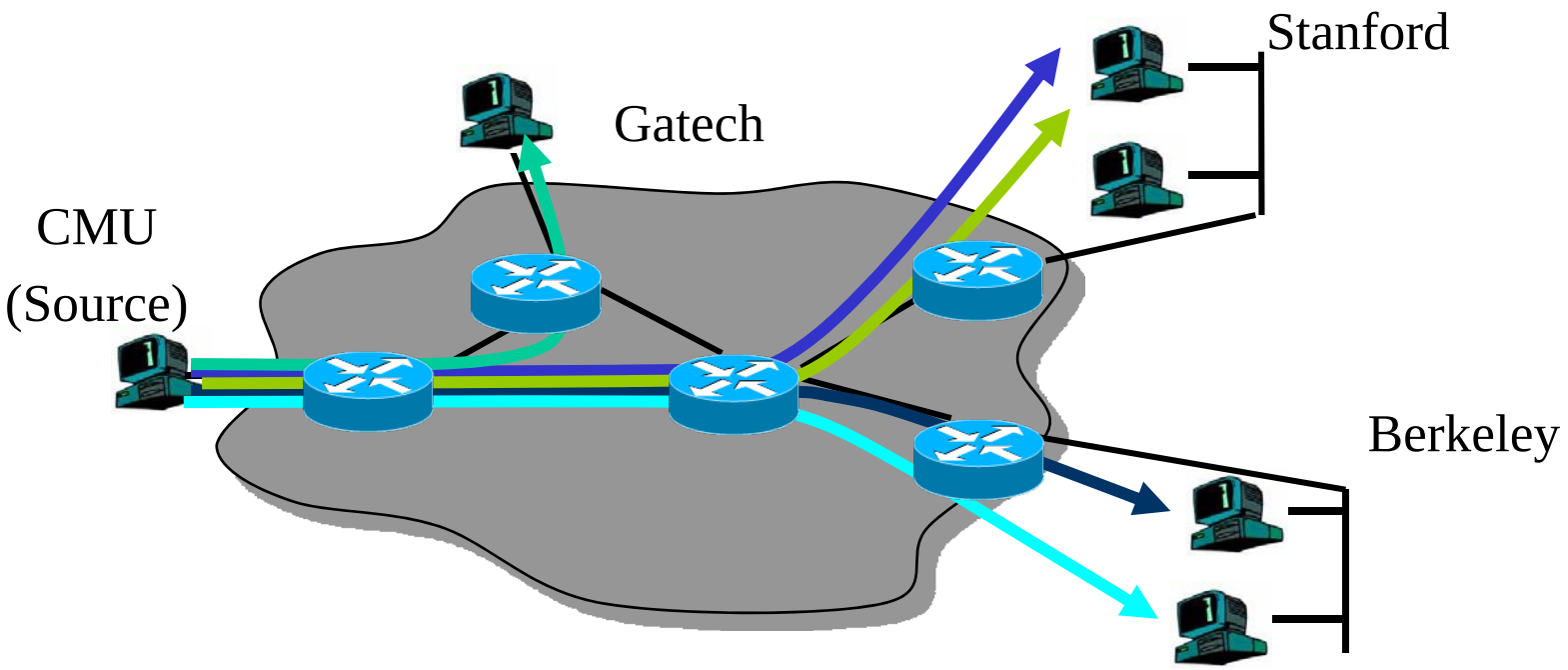
- BUT: In today's Internet, IP multicast is not widely available
- Business model, billing, security have proved major hurdles to enabling IP multicast on a global scale

- Solution: Overlay multicast

Motivating Example: Conference Attendance

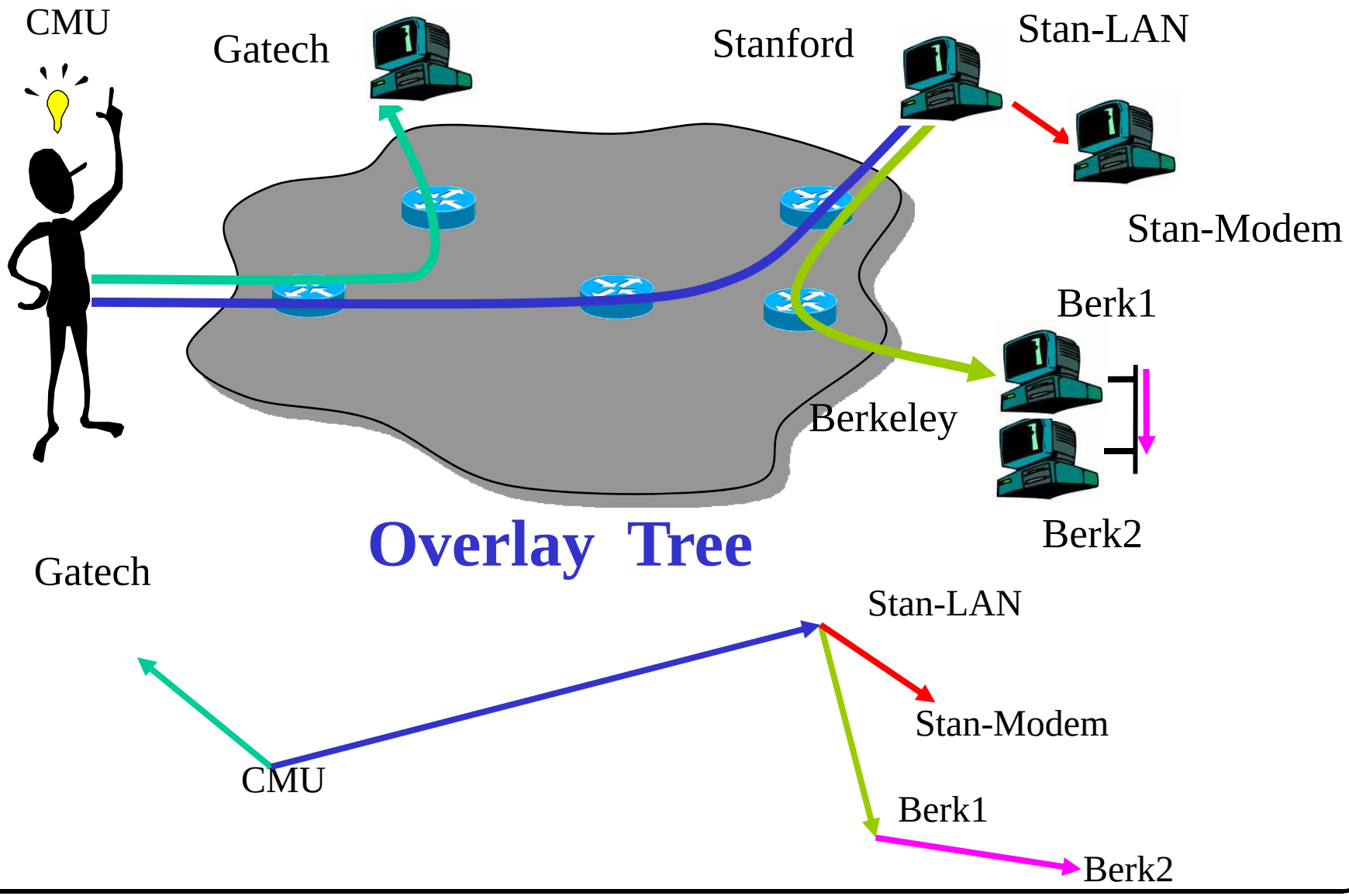


Solution based on Unicast



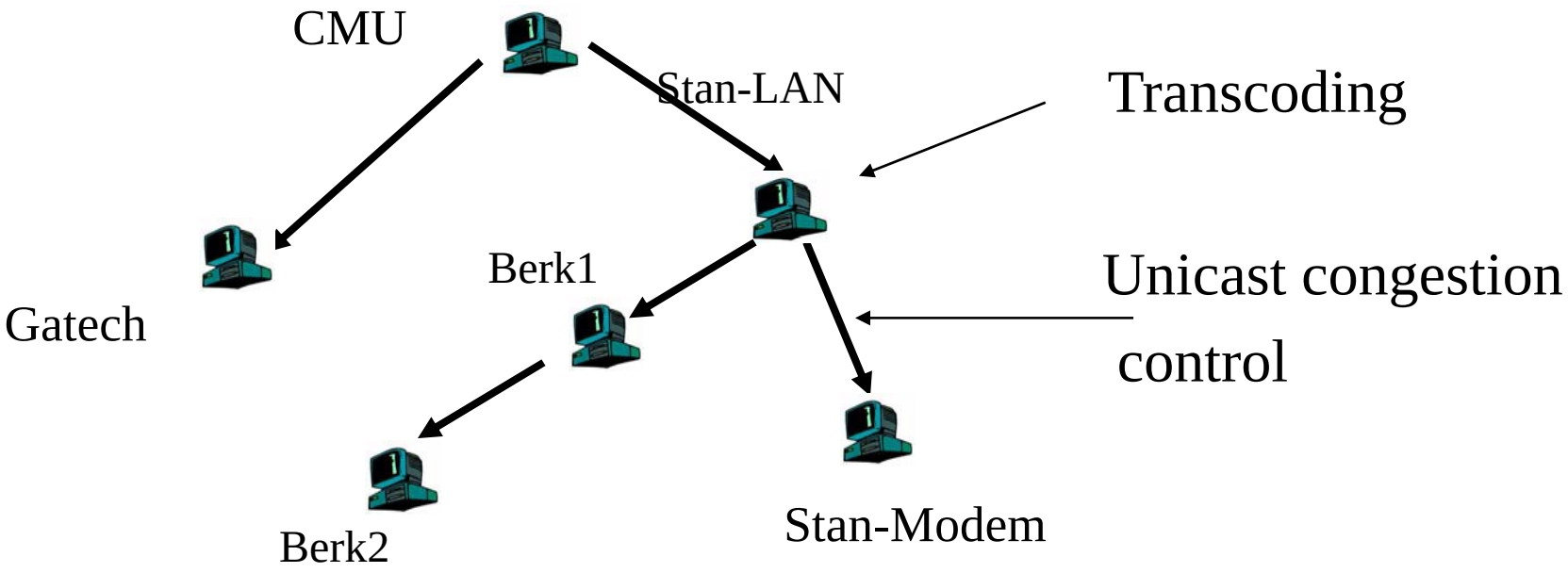
- Client-server architecture (the Web)
- Does not scale well with group size
  - Source host is the bottleneck

End System Multicast



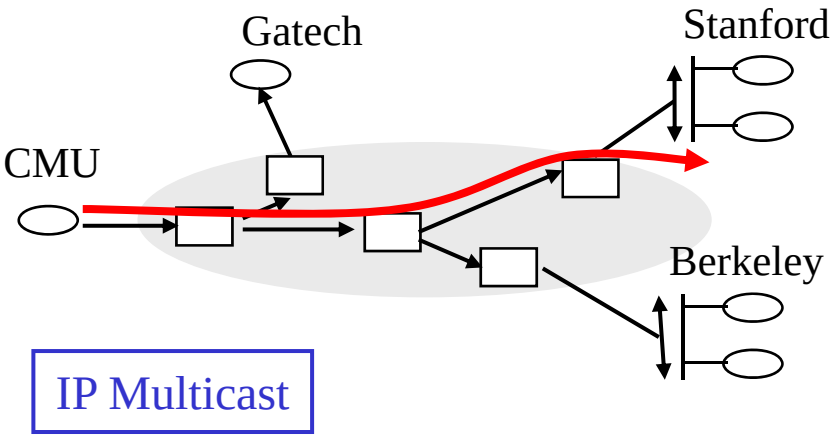
End System Multicast: Benefits

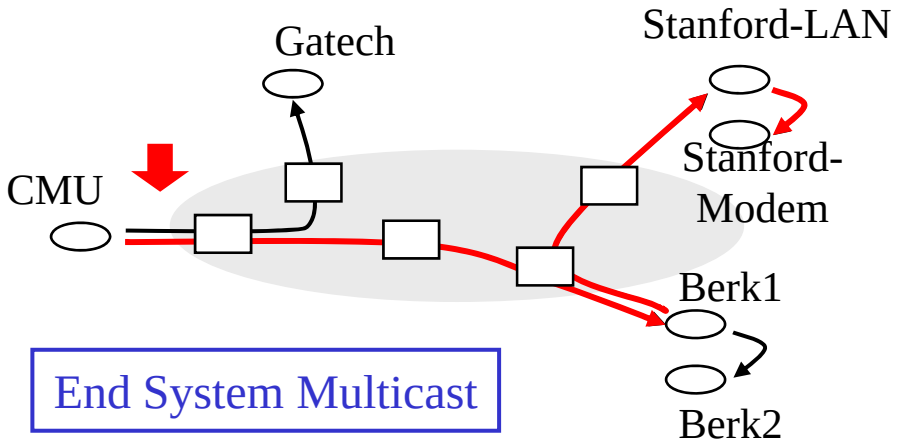
- Scalability
  - Routers do not maintain per-group state
- Easy to deploy
  - Works over the existing unicast IP infrastructure
- Can simplify support for higher level functionality



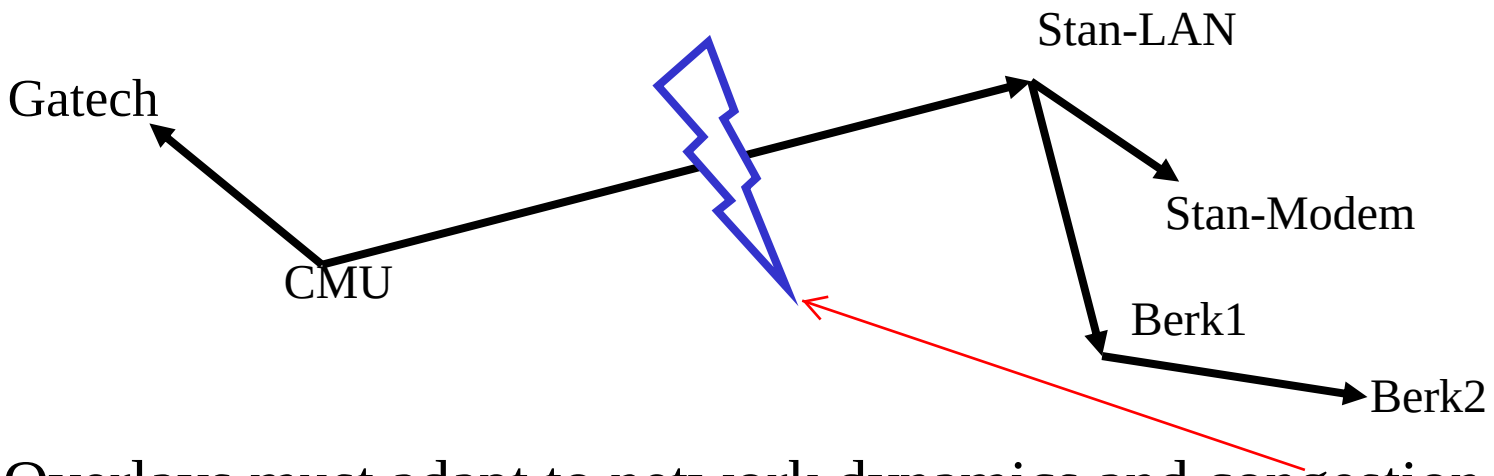
Concerns with End System Multicast

- Challenge to construct efficient overlay trees
- Performance concerns compared to IP Multicast
  - Increase in delay
  - Bandwidth waste (packet duplication)

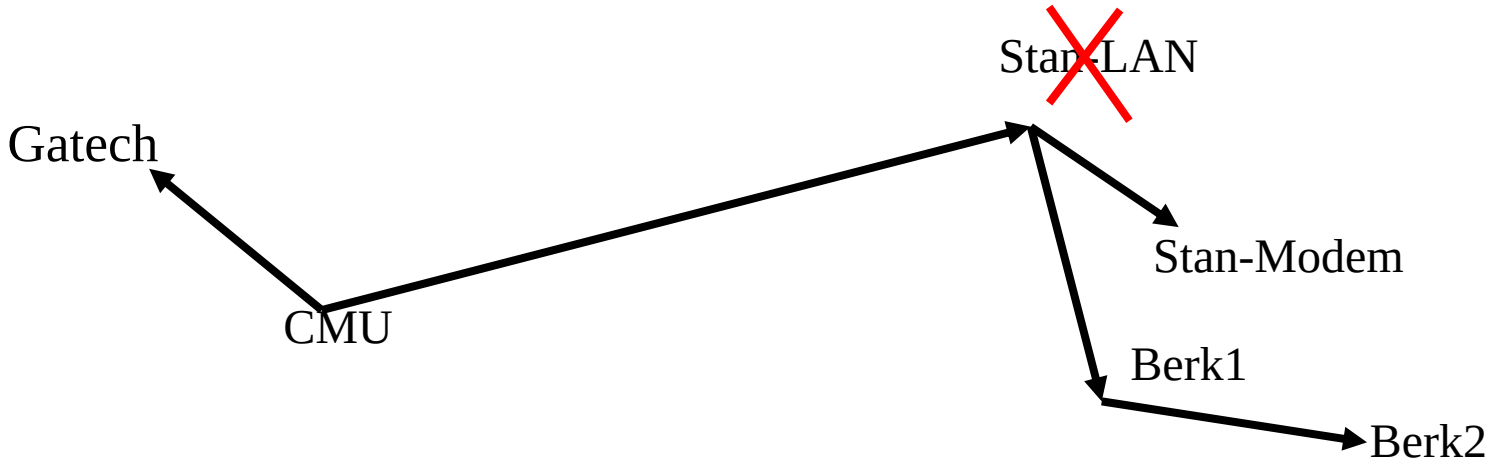




More Challenges

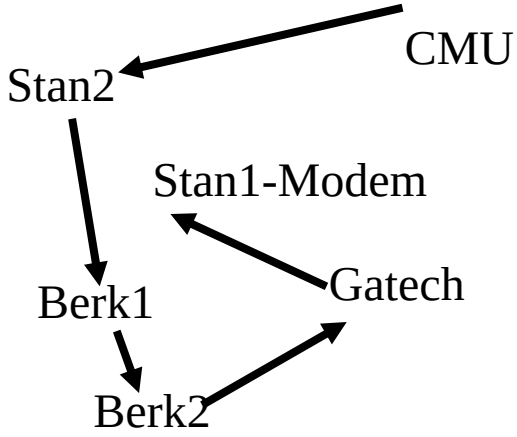


Overlays must adapt to network dynamics and congestion



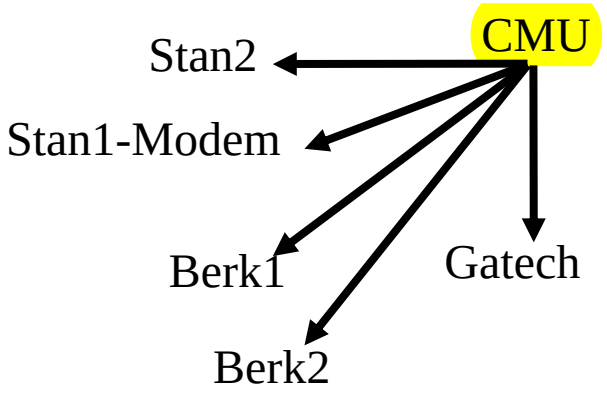
Group membership is dynamic: members can join and leave

Inefficient Overlay Trees



High latency

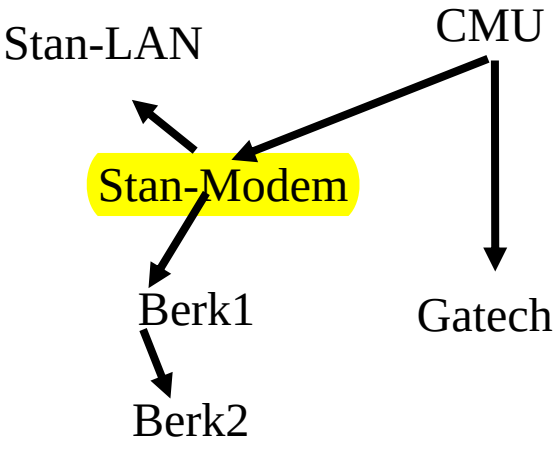
from CMU to Stan1-Modem



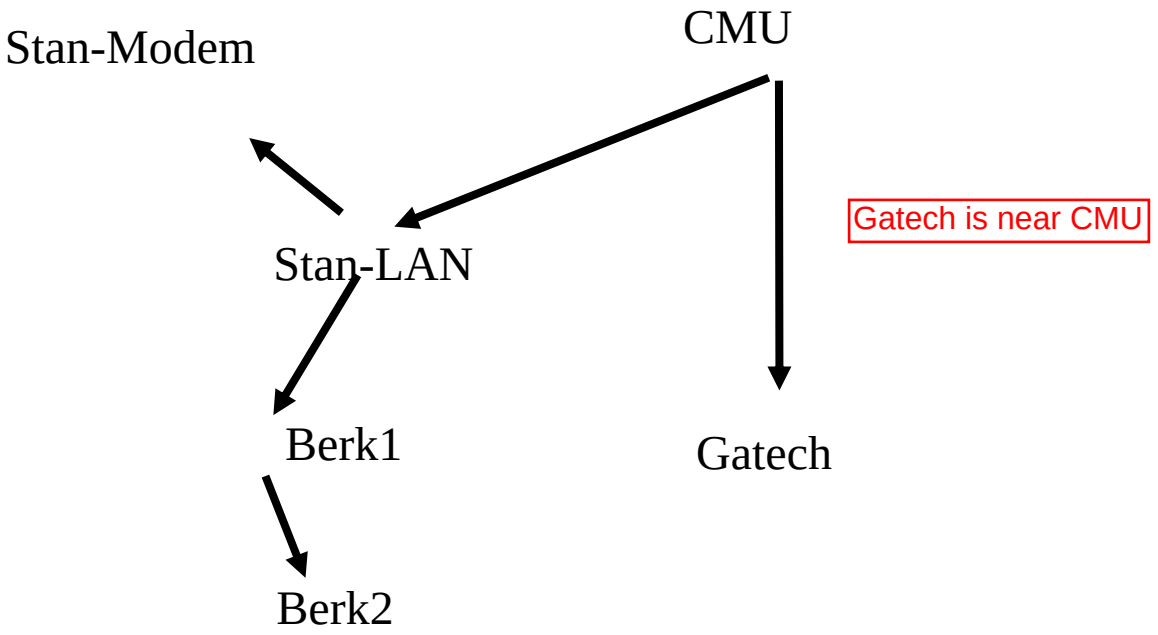
-Poor network usage

-Potential congestion near CMU

Poor bandwidth to members



An Efficient Overlay Tree

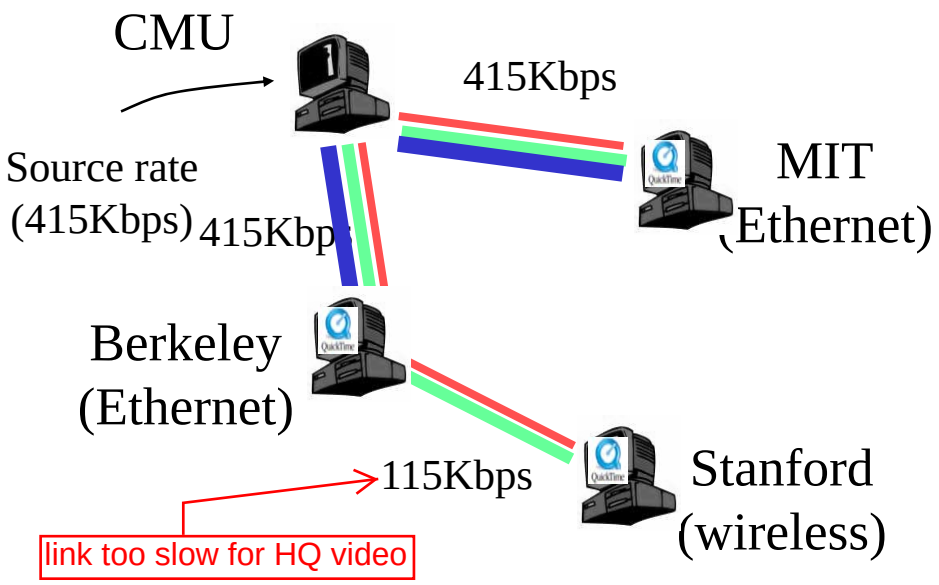


End System Multicast System (ESM)

- Focus on video broadcast applications
- Implementation
  - Integrate with Apple QuickTime
  - Support for receiver heterogeneity
  - Support peers behind NATs or firewalls
  - Run on Windows and Linux platforms
- Showcase
  - SIGCOMM (max 60 simultaneous users)
  - Several CMU Distinguished Lectures
  - Slashdot (max 180 simultaneous users)

Adapt to Receiver Heterogeneity

- Congestion control: hop-by-hop TCP
  - Determine acceptable data rate
- Prioritization of data streams
  - Parent maintains a priority queue
  - Drop video packets before audio packets



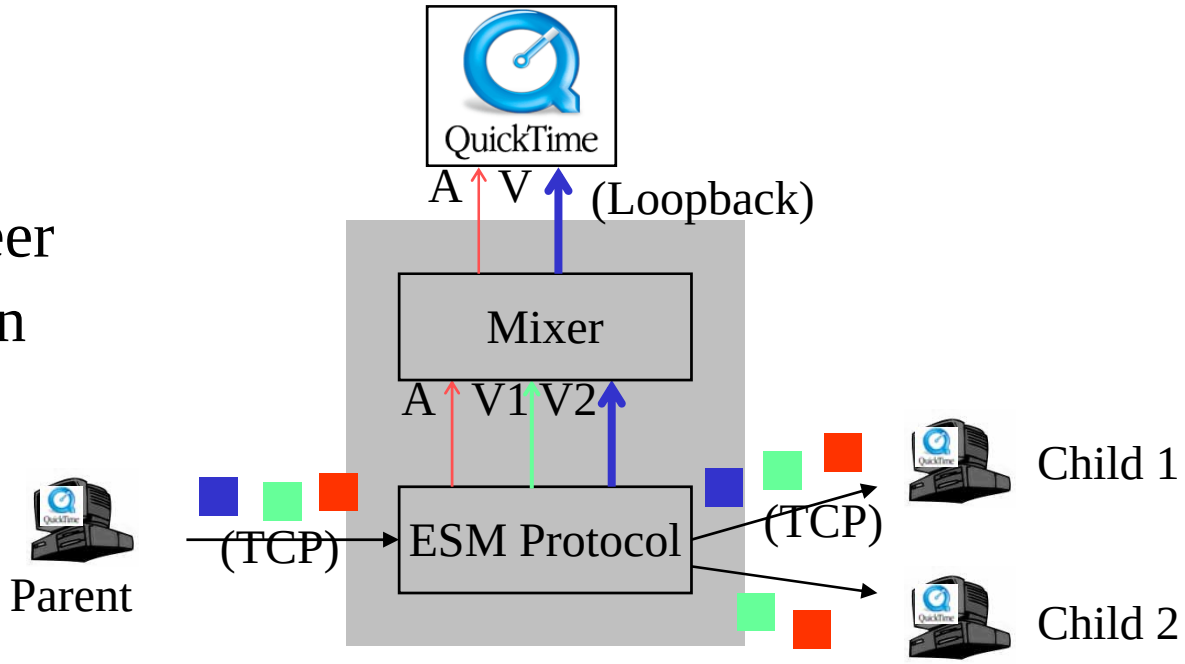
Type	Priority	Bandwidth
audio	high	15Kbps
LQ video		100Kbps
HQ video	low	300Kbps

Interface with Application (QT)

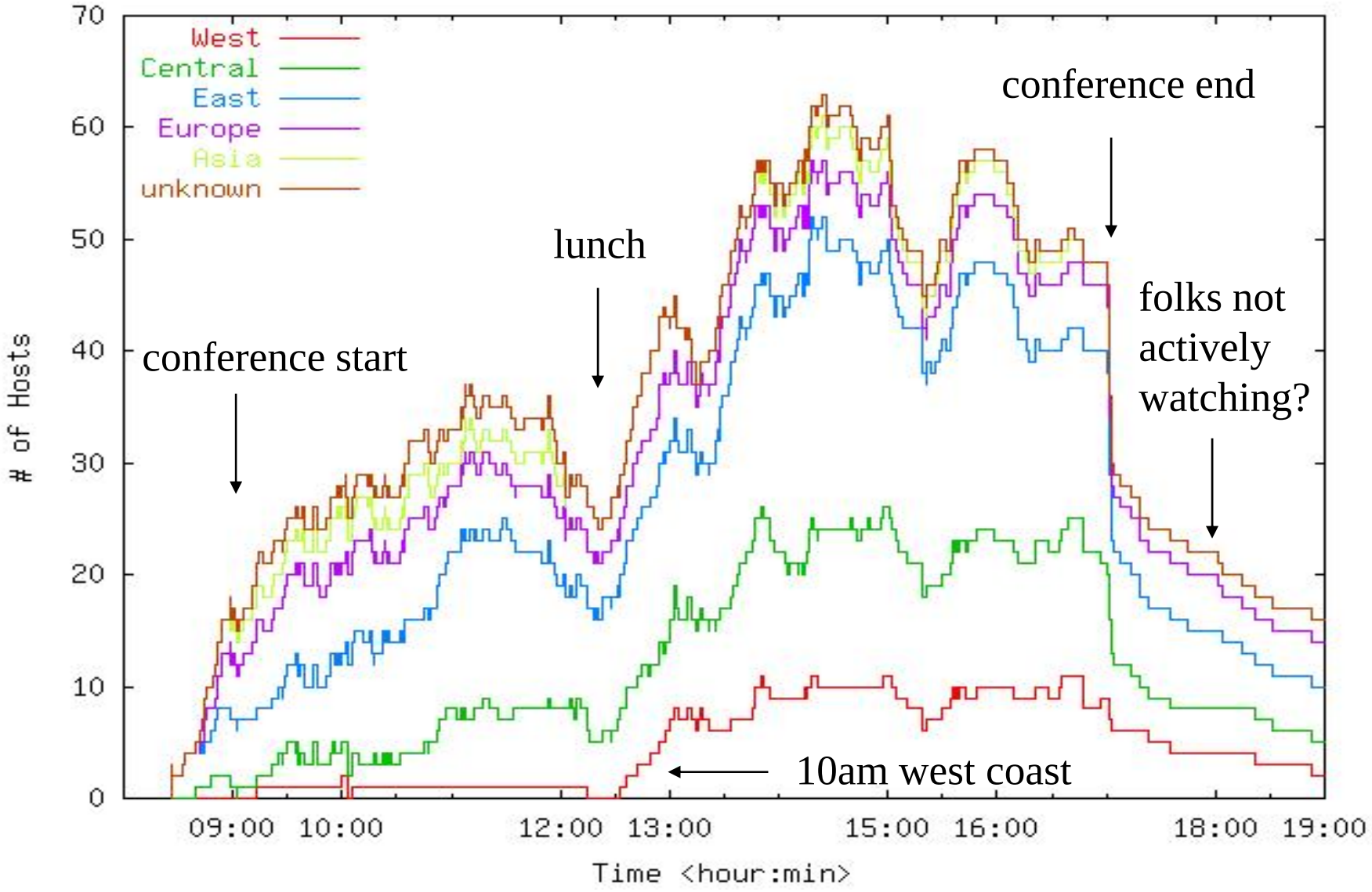
Mixer

- Selects best viewable video stream dynamically
- Switches between video streams for seamless playback

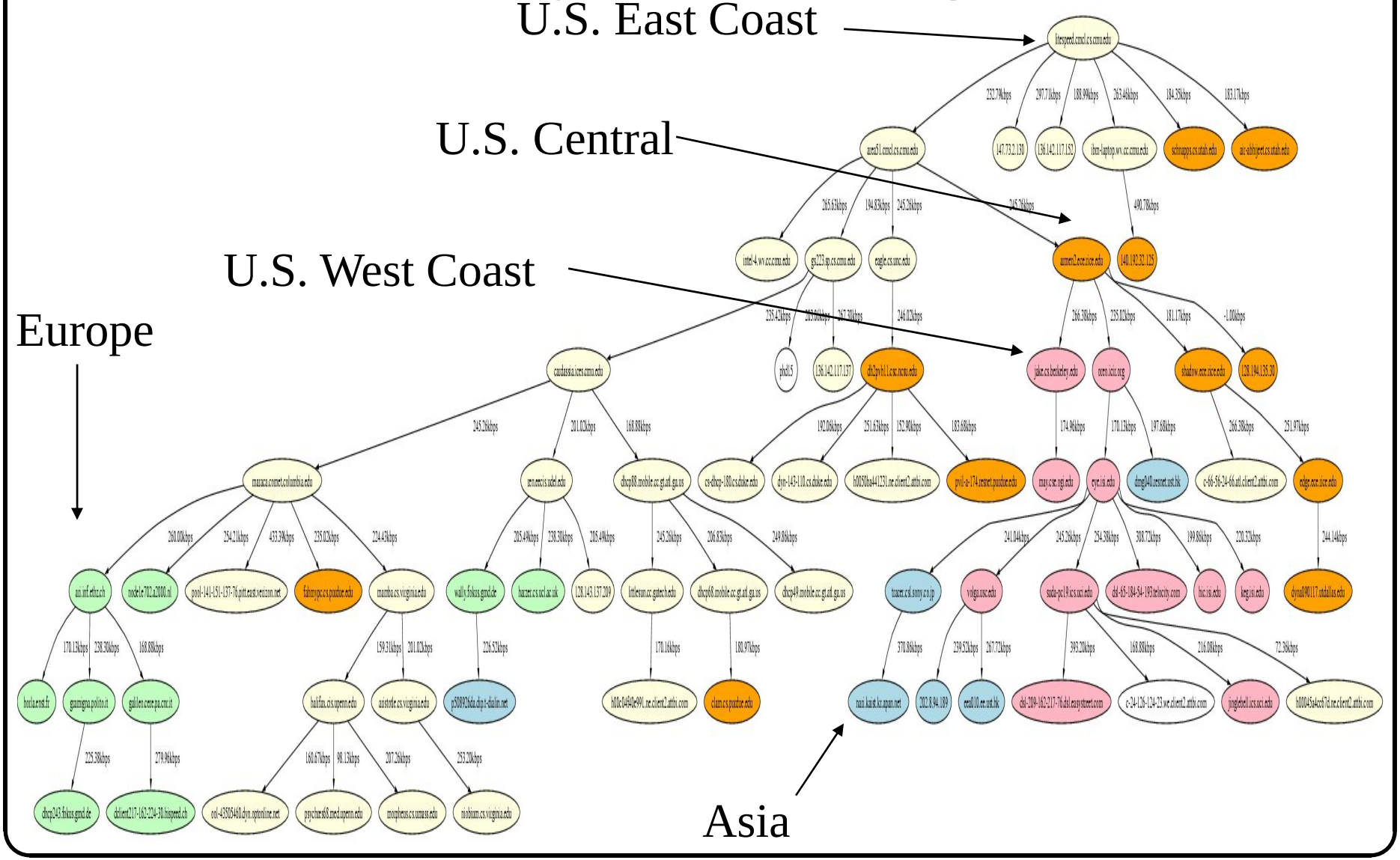
Example: a peer with 2 children



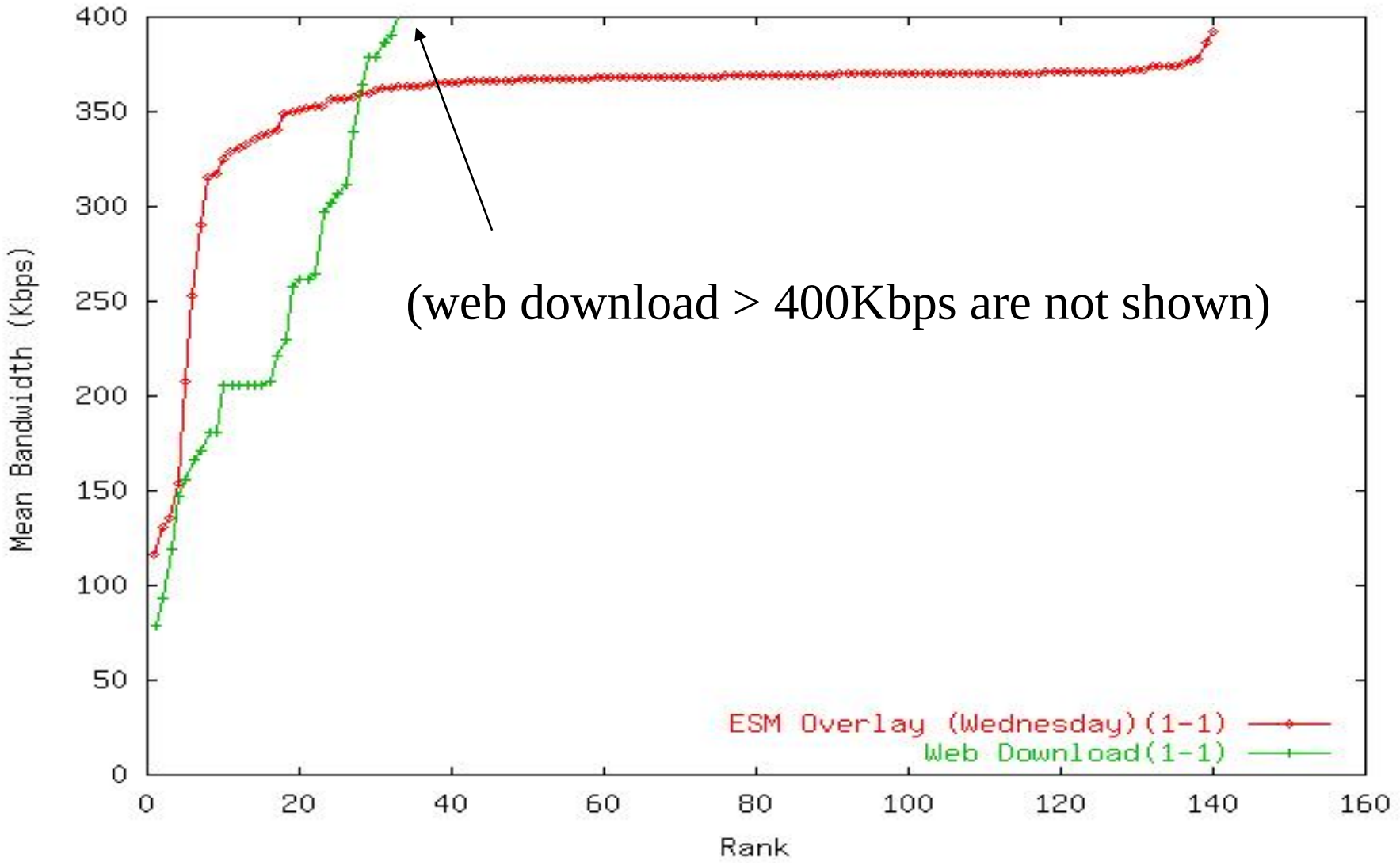
Group Dynamics



Overlay Tree at 2:09pm

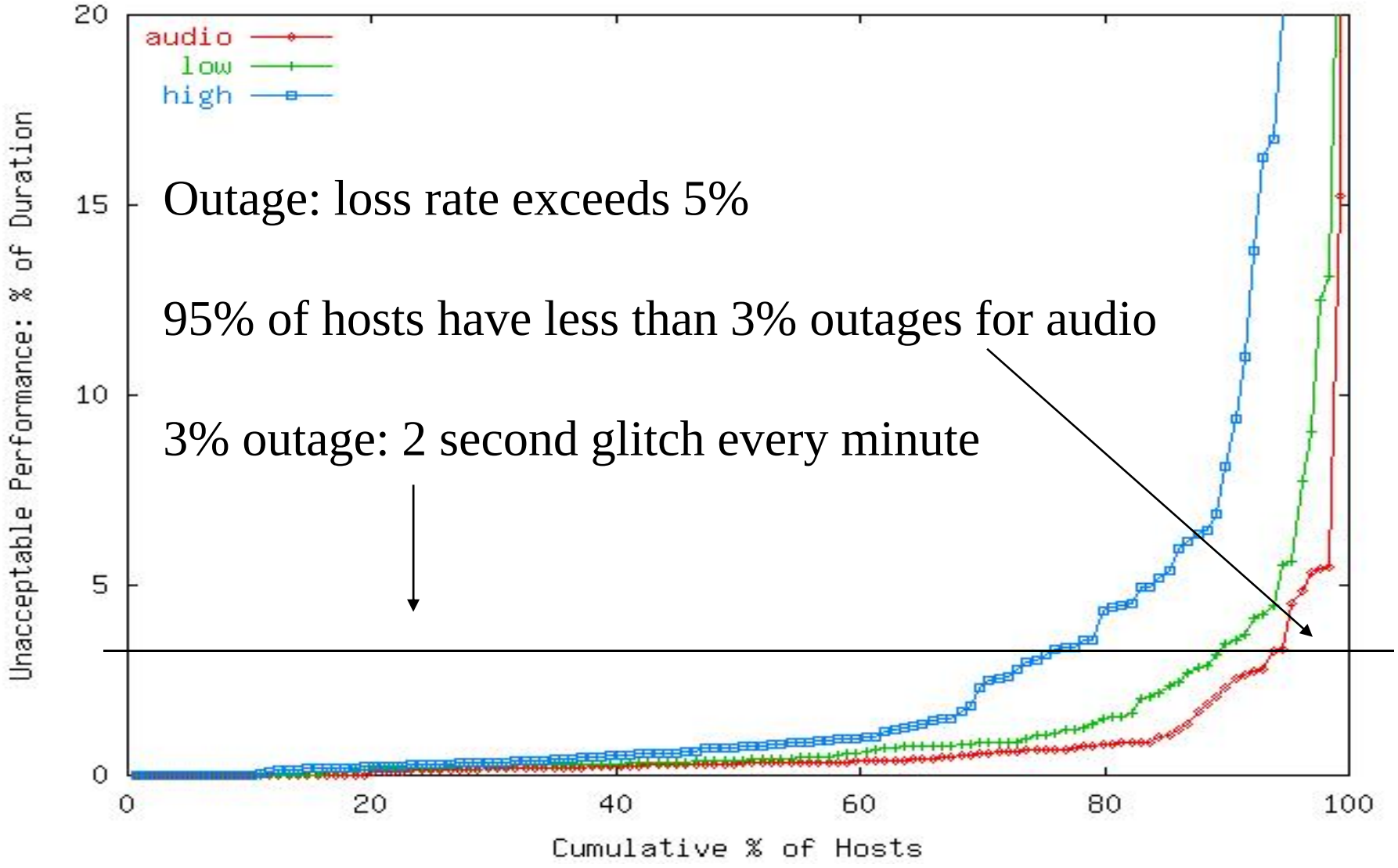


Receiver Bandwidth



(web download > 400Kbps are not shown)

Transient Performance: Outages



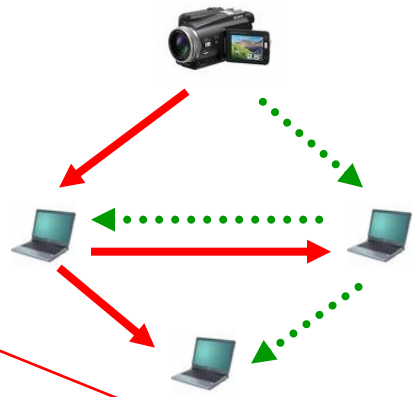
Advanced content distribution

Limitations of single-tree overlay multicast

(e.g. ESN):

- Node departure/failure disrupts a potentially large number of receivers while tree is being repaired
- Only a small number of interior tree nodes contribute forwarding bandwidth -> unfair
- Slow to adapt to variations in available forwarding bandwidth

Advanced content distribution: Multi-tree



Multiple trees:

- Each participant joins *k* trees
- The trees are interior-node-disjoint
- Content is striped evenly into the trees

Result:

- Forwarding load can be evenly distributed
- A single failure may affect at most *1/k* of the content

because of

Advanced content distribution: Swarming

Idea (e.g., BitTorrent):

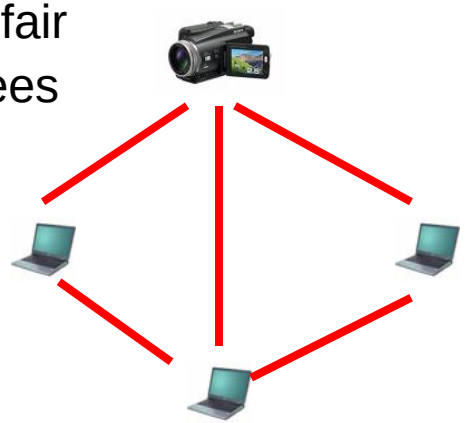
- Construct a random mesh of participants
- Initially, source has all the blocks of a file
- In each round, neighbors exchange bitmap of available blocks
- Then, swap missing blocks
- Continue until everyone has all blocks of the file

Result:

- Easily adapts to failures and bandwidth fluctuations, fair
- But: longer delays, less efficient network use than trees

BitTorrent is widely used for bulk content distribution

- Movies, music, software updates



Advanced content distribution: Swarming

Can also be used for streaming media broadcast:

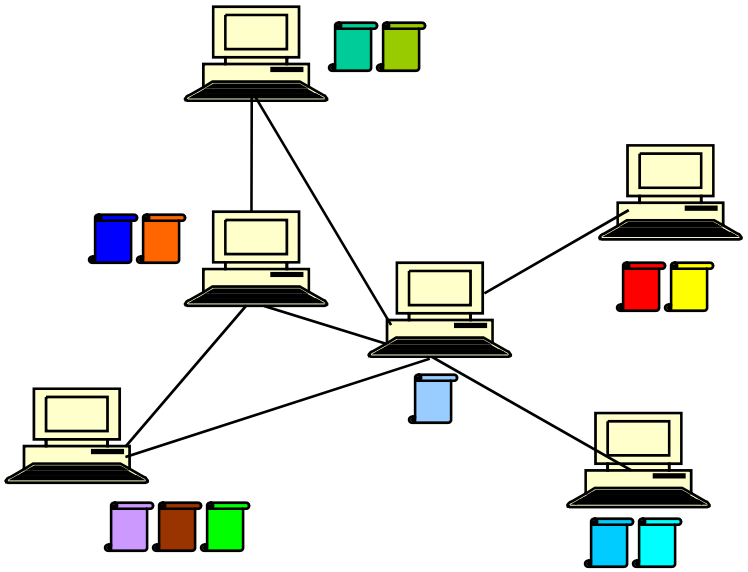
- Nodes maintain a sliding window of recent blocks (e.g. 60 seconds worth of blocks)
- Swarming on the current window
- Typical delivery delay is on the order of window size

Systems:

CoolStreaming, **SOPcast**, **PPLive**, **PPStream**

Challenge: Decentralized object location

- Resources (objects) partitioned among participating nodes
- Mapping from objects to nodes is dynamic



Unicast routing doesn't help

- don't know who to talk to
- don't know where to store objects
- want to address *resources* (*objects)*, not *nodes*!

Object location: Approaches

Unstructured overlays

- Objects can be placed arbitrarily
- Queries flooded through network
- Powerful search (content-based)
- BUT: Cannot achieve both scalability and recall

Structured overlays

- Objects uniquely mapped to a (set of) live node
- Reliable and efficient object lookup function
- BUT: Limited to identity lookup

Structured p2p overlays

Given a dynamic set of participating nodes, is it possible to

- evenly partition state among the set of participating nodes?
- reliably and efficiently look up any state object?
- dynamically maintain spanning trees among any subset of participating nodes?
- do this efficiently, despite dynamic node membership, node and network failures?

Structured p2p overlays

Key-based routing primitive (KBR):

route(M, X): route message *M* to the live node with *node id* closest to key *X*

- Node ids and keys are selected from a large, sparse id space

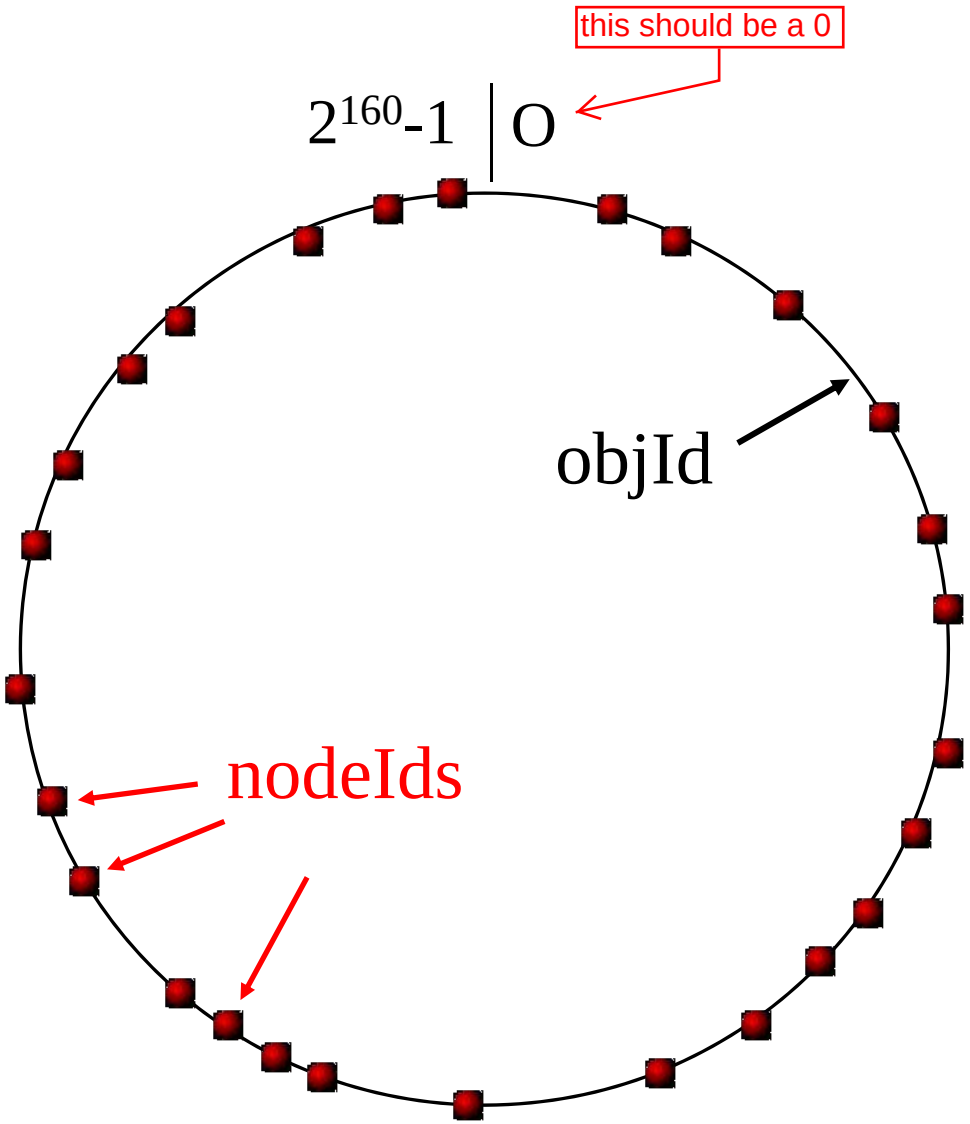
Example: **Pastry**

(others include Chord, CAN, Tapestry, Bamboo, Kademlia, SkipNet, Kelips, Accordeon, etc.)

Pastry

- Self-organizing overlay network
- Lookup/insert object in < $log_{16} N$ routing steps (expected)
- *O(log N)* per-node state
- Network proximity routing

Pastry: Key distribution



Consistent hashing [*Karger et al. '97*]

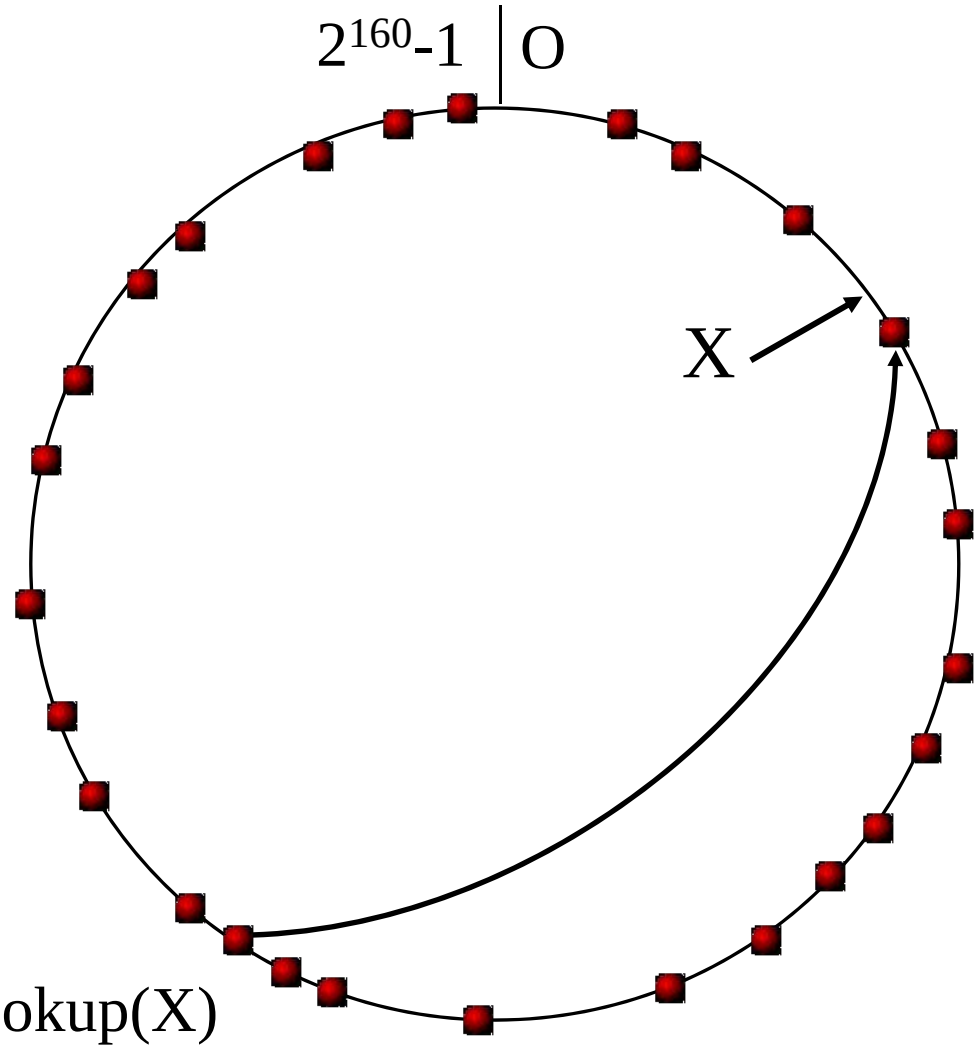
160 bit circular id space

nodeIds (uniform random)

objIds (uniform random)

Each key is mapped to the live node with "closest" nodeId

Pastry: Lookup



Msg with key *X* is routed to live node with nodeId closest to X

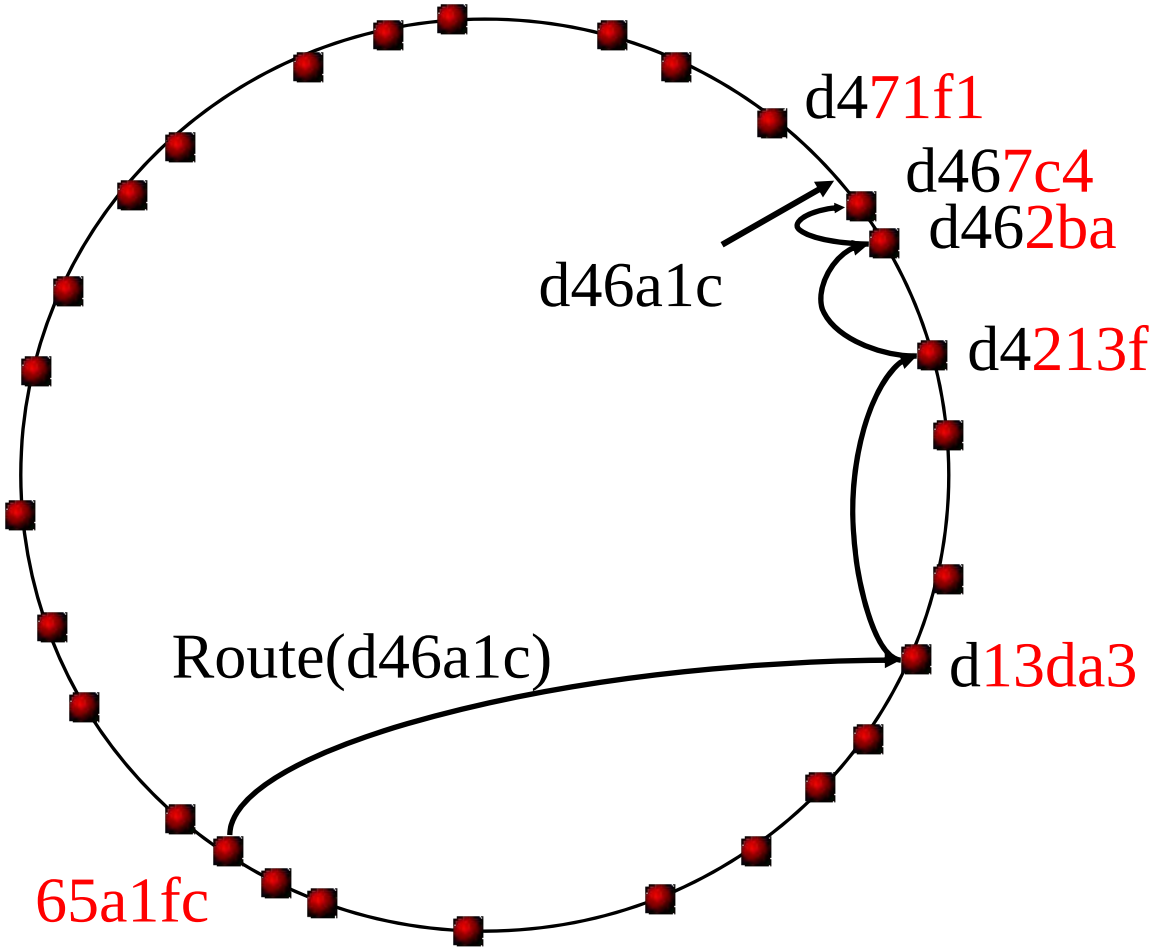
Problem:
complete routing table not feasible

Pastry: Routing

Tradeoff

- O(*log N*) routing table size
- O(*log N*) message forwarding steps

Pastry: Routing



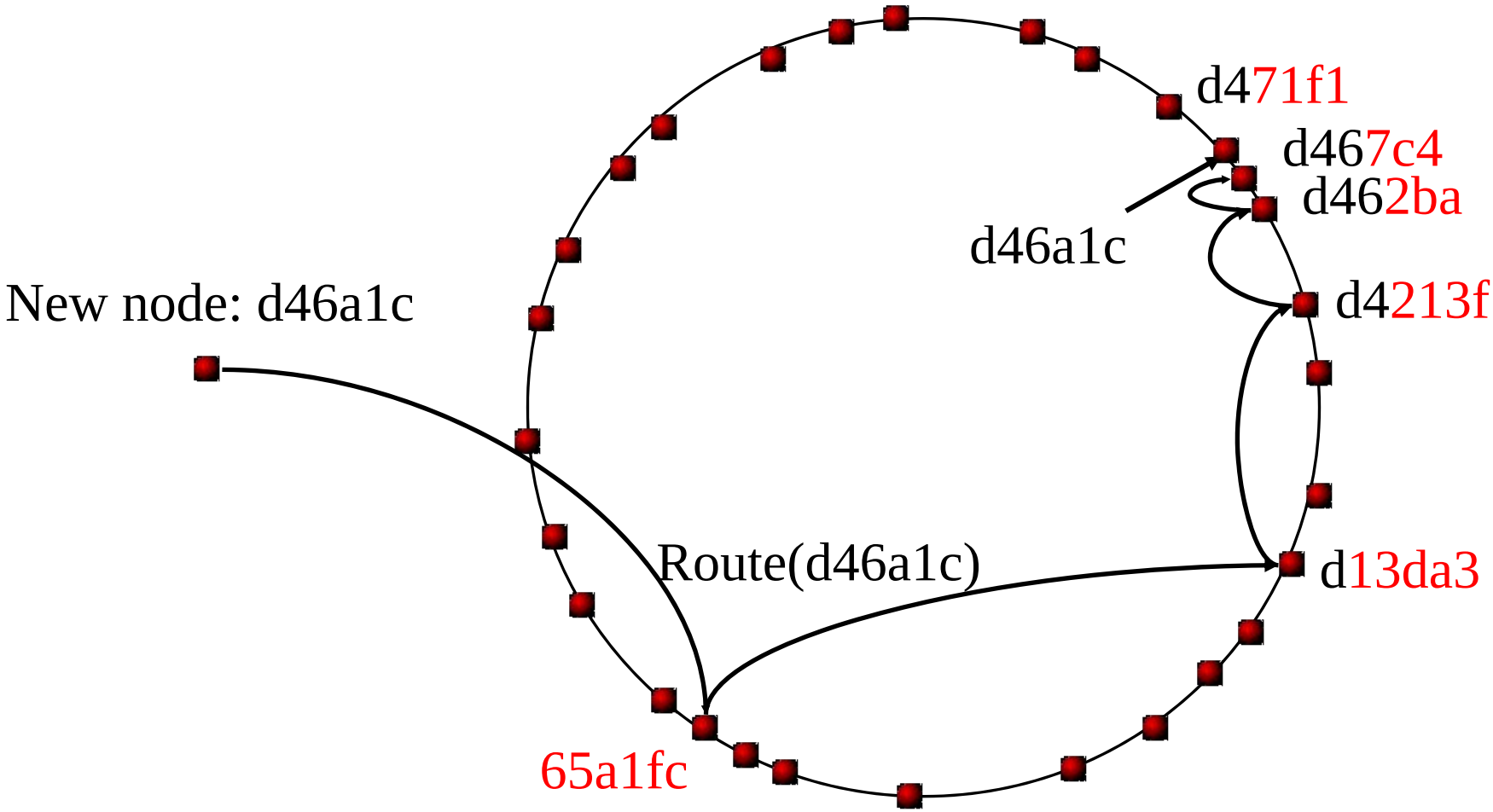
Properties

- $\log_{16}$ N steps
- O(*log N*) state

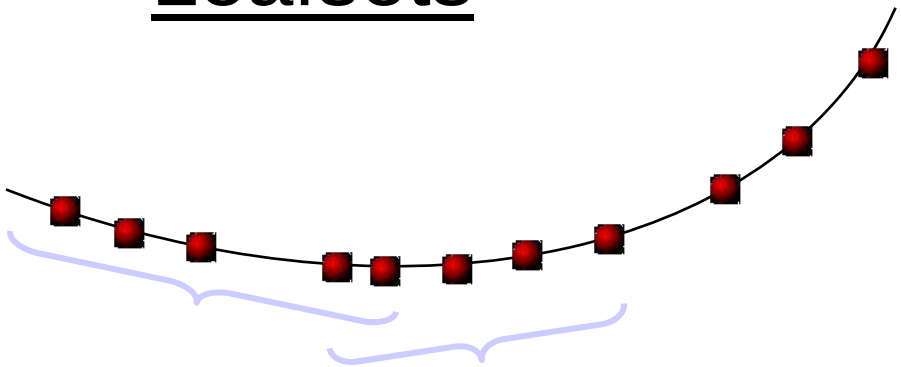
Pastry: Routing table (# 65a1fcx)

Row 0	0x	1x	2x	3x	4x	5x		7x	8x	9x	ax	bx	cx	dx	ex	fx
Row 1	60x	61x	62x	63x	64x		66x	67x	68x	69x	6ax	6bx	6cx	6dx	6ex	6fx
Row 2	650x	651x	652x	653x	654x	655x	656x	657x	658x	659x		65bx	65cx	65dx	65ex	65fx
Row 3	65a0x		65a2x	65a3x	65a4x	65a5x	65a6x	65a7x	65a8x	65a9x	65aax	65abx	65acx	65adx	65aex	65afx
$\log_{16}$ N rows																

Pastry: Node join



Leafsets



Stabilization protocol ensures eventual consistency

- aids routing consistency
- enables secure routing
- localizes fault detection within set
- enables application-specific local coordination (e.g., object replica management)

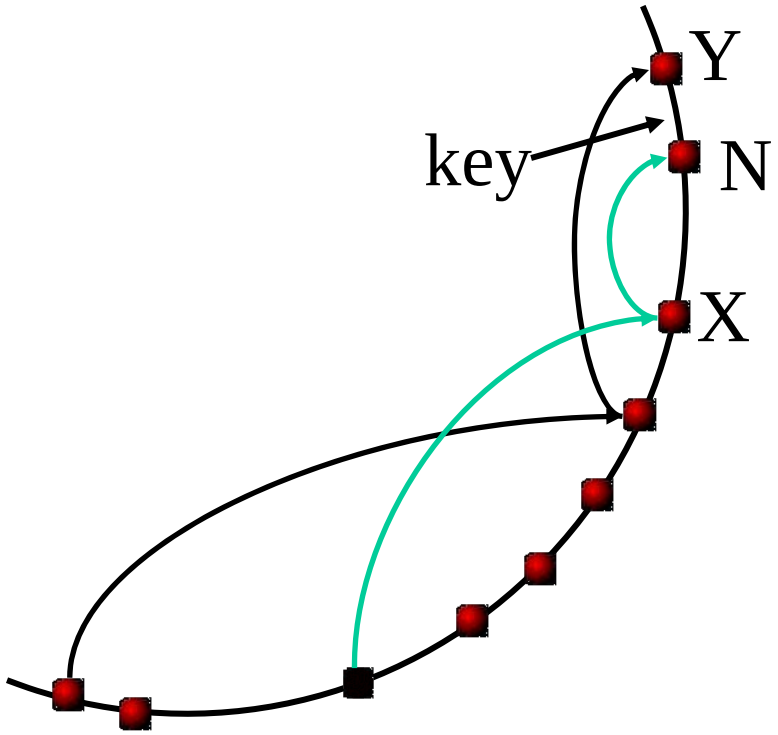
Challenge: Inconsistent routing

Routing consistency*:*

"At any point in time, at most one overlay node accepts messages with a given key"

Necessary for consistency of mutable data

New node N has informed X, but not yet Y of its arrival

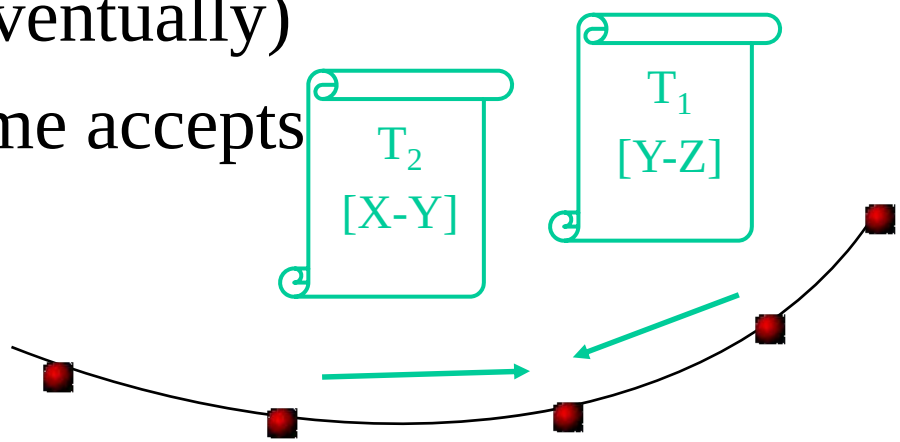


Ensuring routing consistency

- To accept a message with key *k*, a node requires a lease from its neighbors, for an interval $X<k<Z$
- Lease can be issued if grantor has a lease and previous lease has expired

Ensures:

- properly formed ring (eventually)
- at most one node at a time accepts messages with key k
- => routing consistency



Challenge: Self-organization

Initializing and maintaining node state (overlay construction and maintenance)

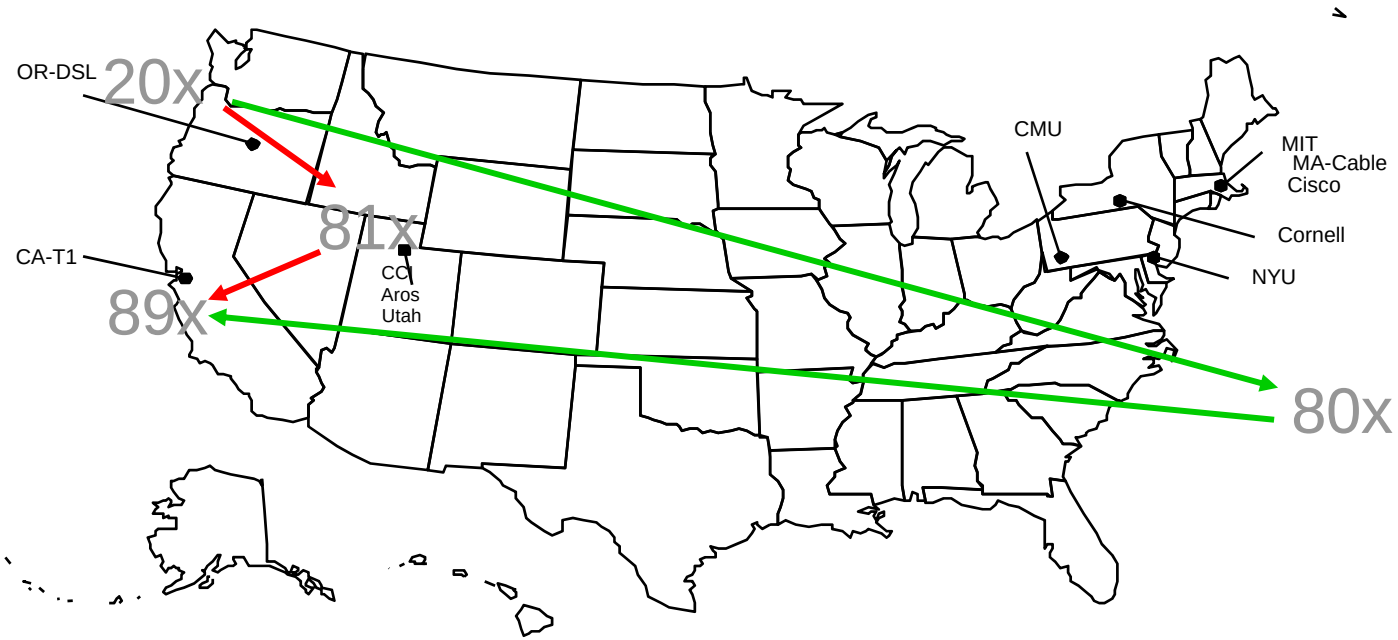
- Node addition
- Node departure (failure)

Pastry: Node departure (failure)

Leaf set members exchange keep-alive messages (failure detection, leaf set stabilization)

- **Leaf set repair (eager):** request set from farthest live node in set
- **Routing table repair (lazy):** get table from peers in the same row, then higher rows

Challenge: Overlay route efficiency



- Nodes *close* in id space, but *far away* in Internet
- **Goal:** choose routing table entries that yield few hops *and* low latency

Solution: Proximity neighbor selection (PNS)

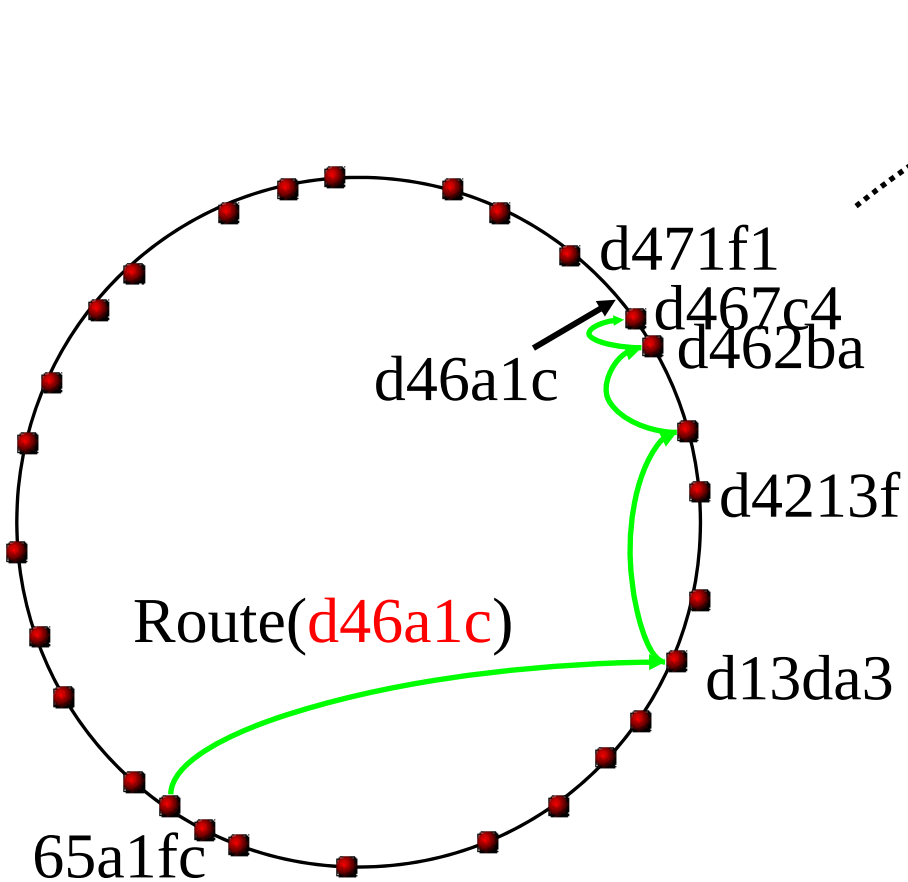
Assumptions:

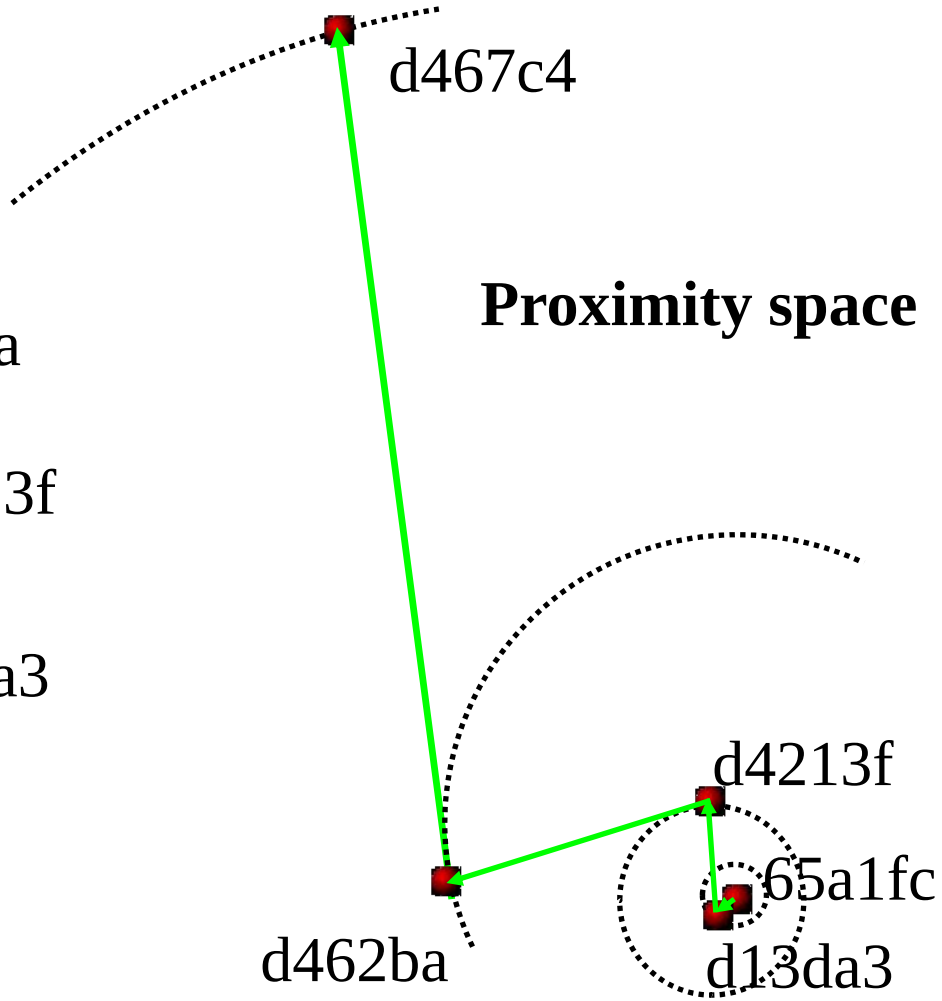
- scalar proximity metric (e.g., RTT)
- a node can probe distance to any other node

Proximity invariant:

Each routing table entry refers to a node close to the local node (in the physical network), among all nodes with the appropriate nodeId prefix.

PNS: Routes in proximity space





PNS Properties

1) **Low-delay routes:** *Average delay penalty, relative to IP, is a small constant (1.3 - 2.2)*

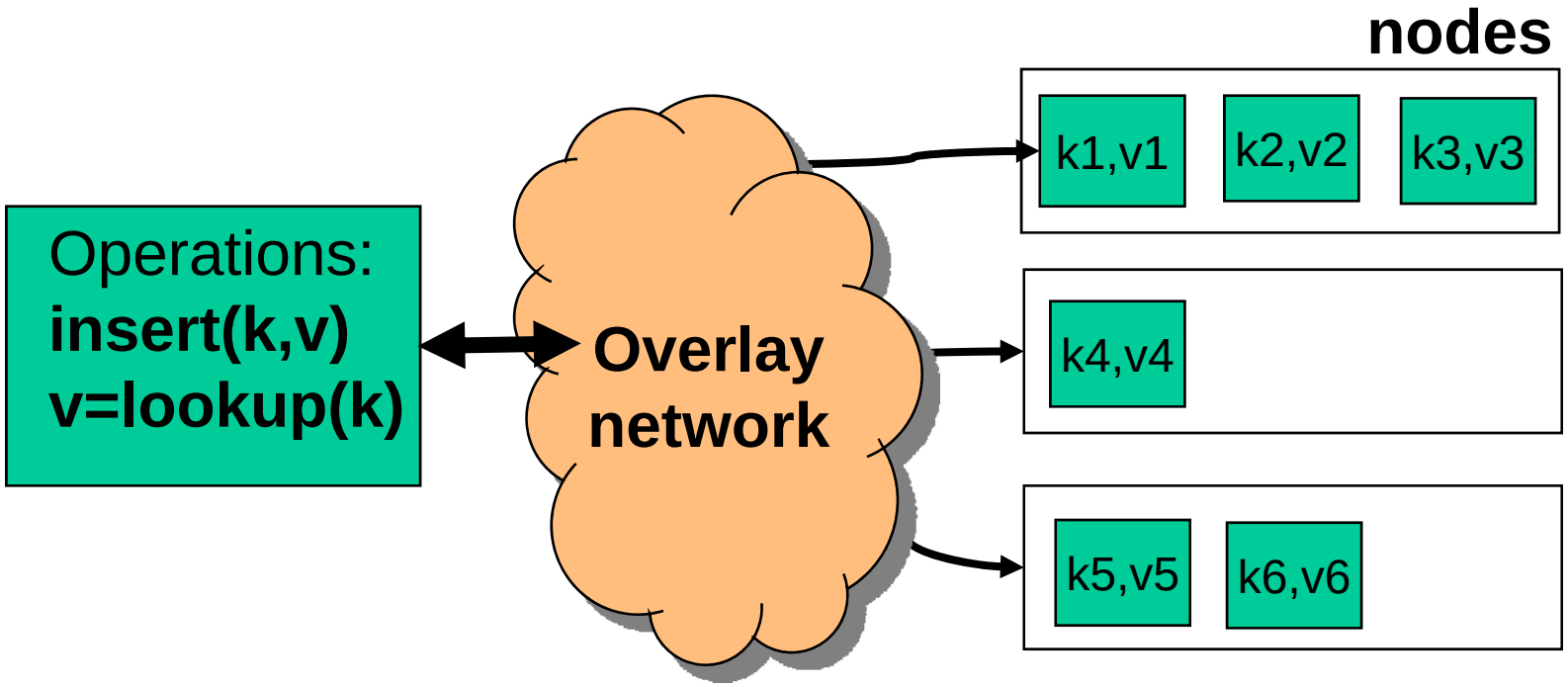
2) **Route convergence:** *Routes of messages sent by nearby nodes with same keys converge at a node near the source nodes*

Sharing state: Distributed hash tables (DHT)

- Layered on top of a structured overlay
- Scalability, Robustness
- Persistence storage
- High availability

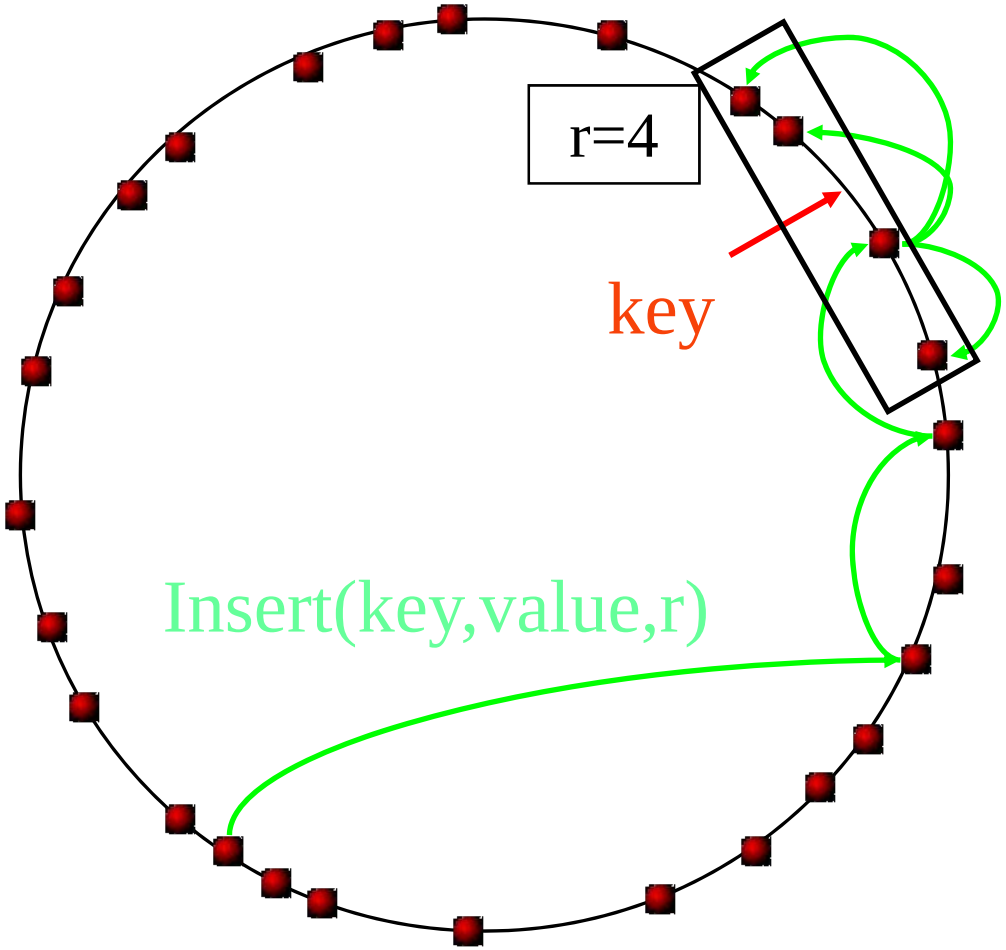
- Example: PAST [SOSP'01]

Distributed hash table (DHT)



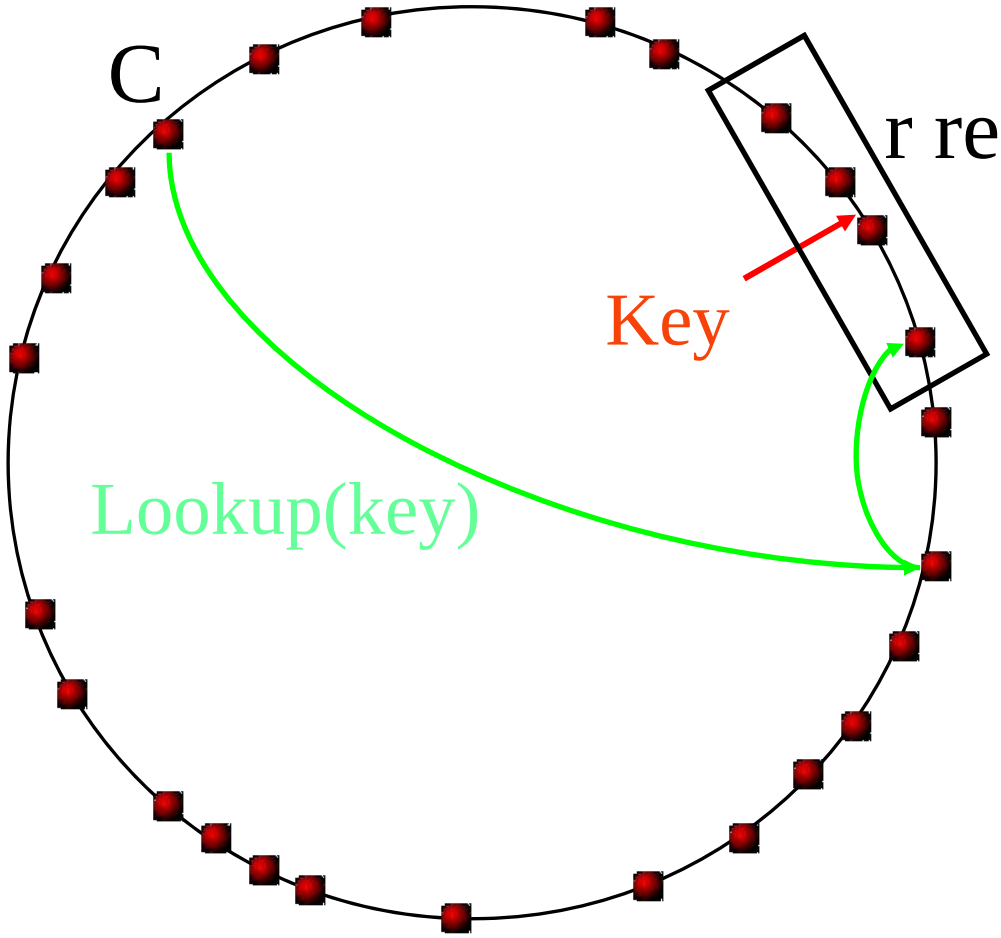
- Structured overlay maps keys to nodes
- Decentralized and self-organizing
- Scalable, robust

DHT: Insertion and replication



Storage Invariant: Tuple replicas are stored on *r* nodes with *nodeIds* closest to *key*

DHT: Lookup



Object located in $\log_{16}$ N steps (expected)

usually locates replica nearest client C

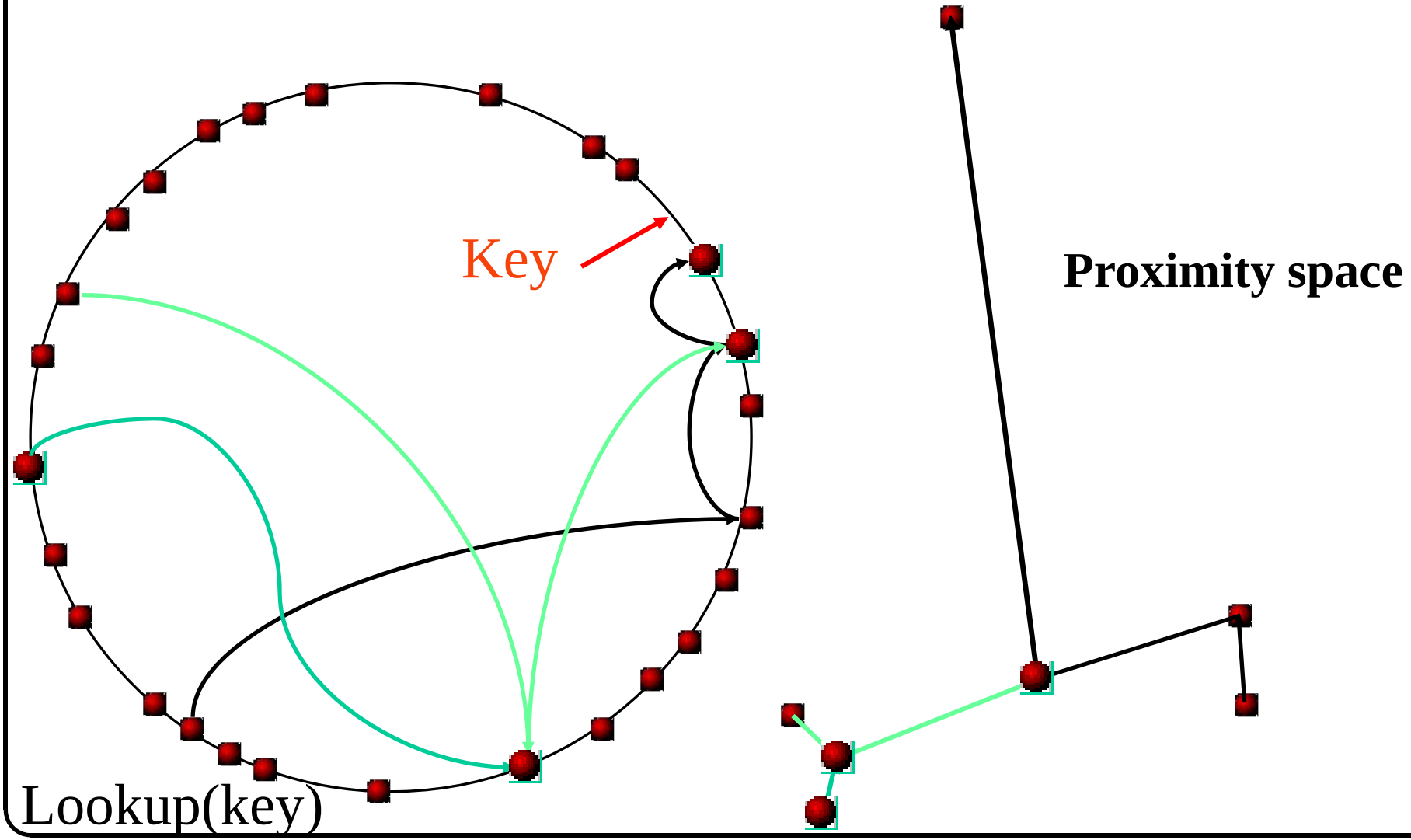
DHT: Dynamic caching

- Nodes cache tuples in the unused portion of their allocated disk space
- Files cached on nodes along the route of lookup and insert messages

Goals:

- maximize query xput for popular tuples
- balance query load
- improve client latency

DHT: Dynamic caching

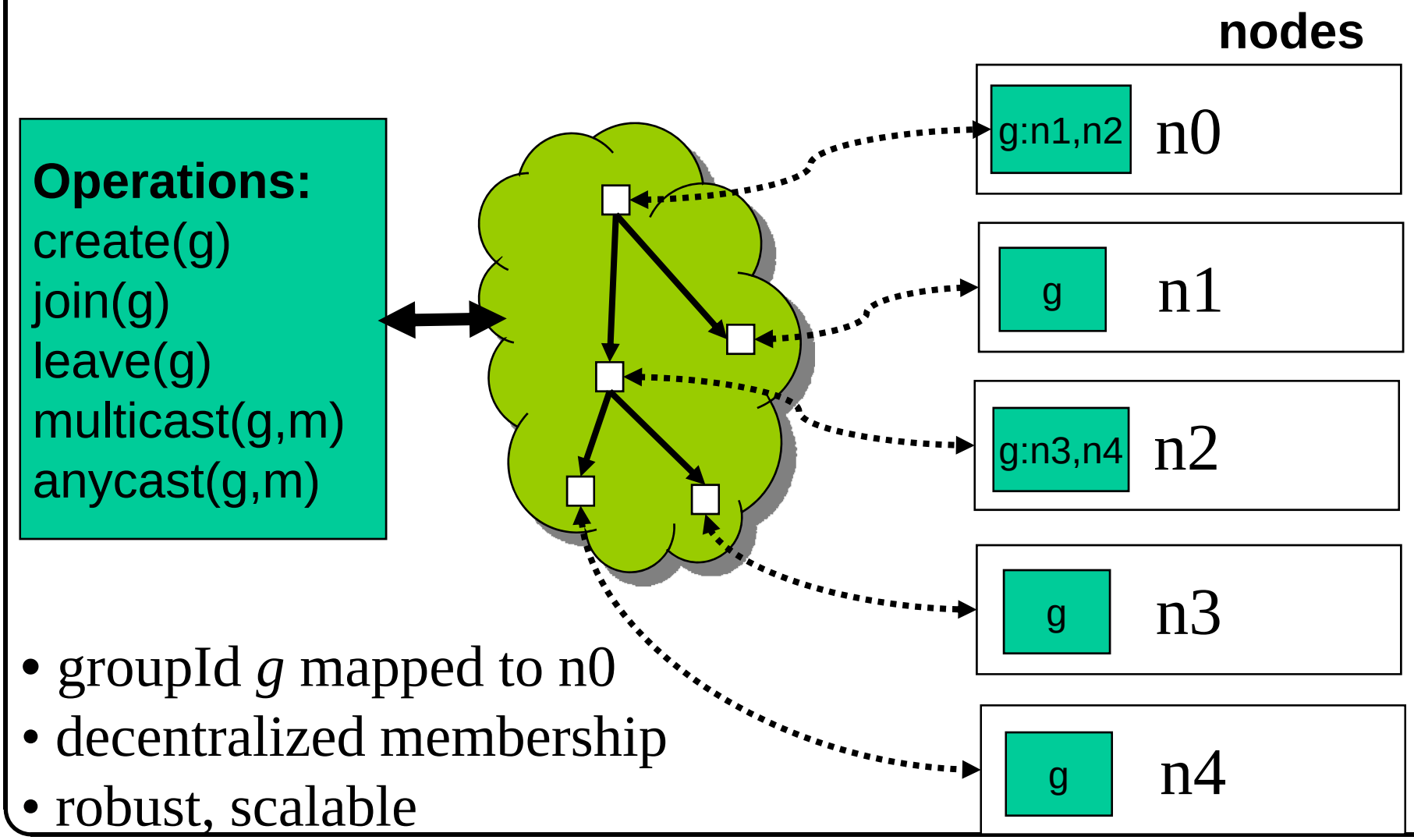


Coordination: Cooperative group communication

- Scribe: Tree-based group management
- Multicast, anycast primitives
- Scalable: large numbers of groups, members, wide range of members/group, dynamic membership

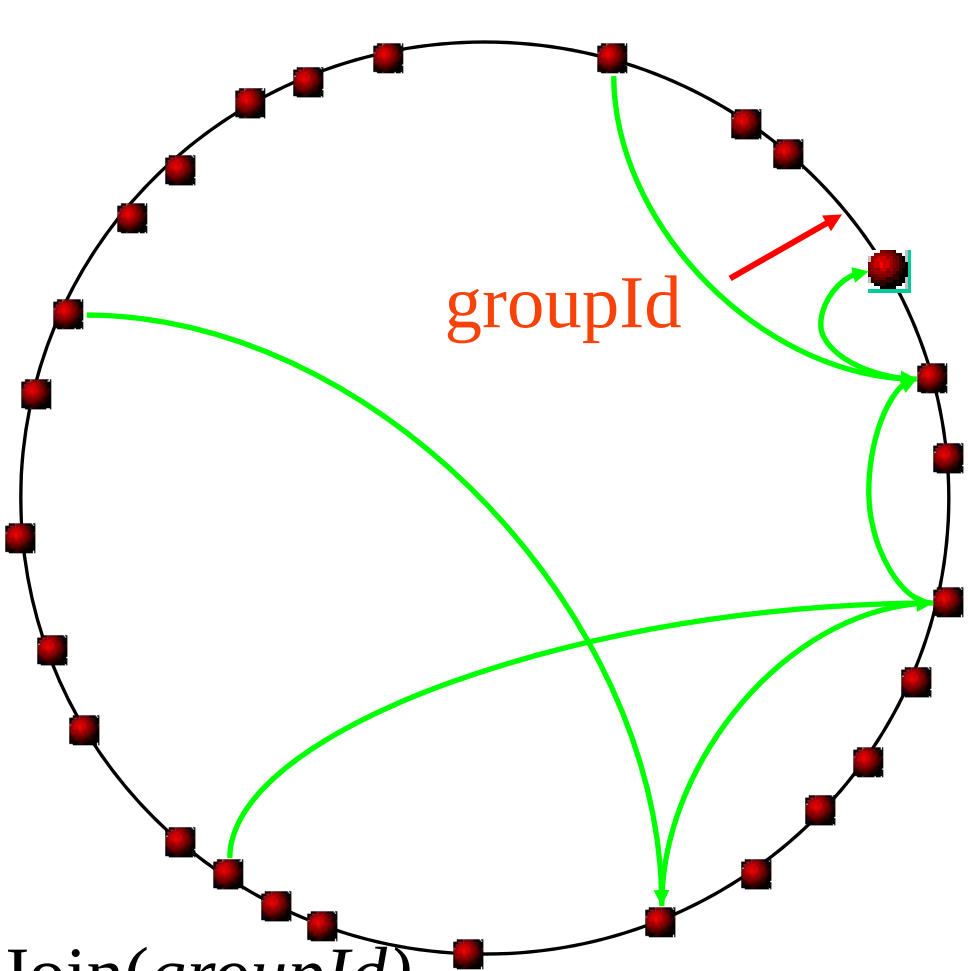
  [IEEE JSAC ’02]

Cooperative group communication

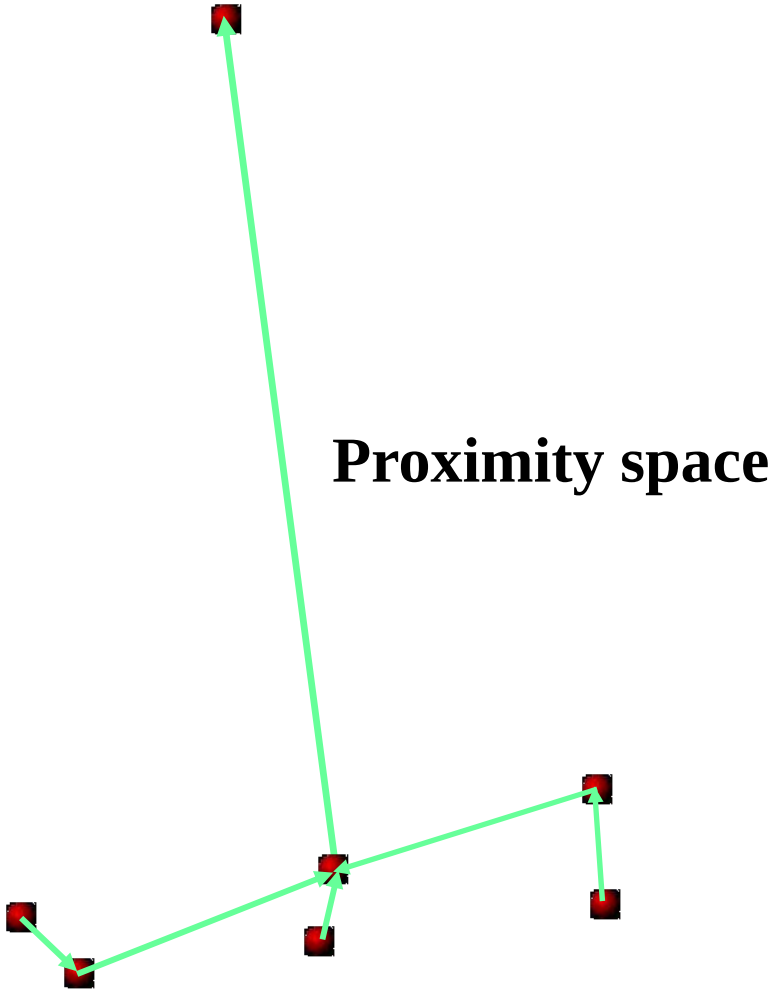


- groupId *g* mapped to n0
- decentralized membership
- robust, scalable

Scribe



Join(*groupId*)



Scribe: Anycast