MULTI+

Building Topology-Aware Overlay Multicast Trees

Luis Garcés-Erice

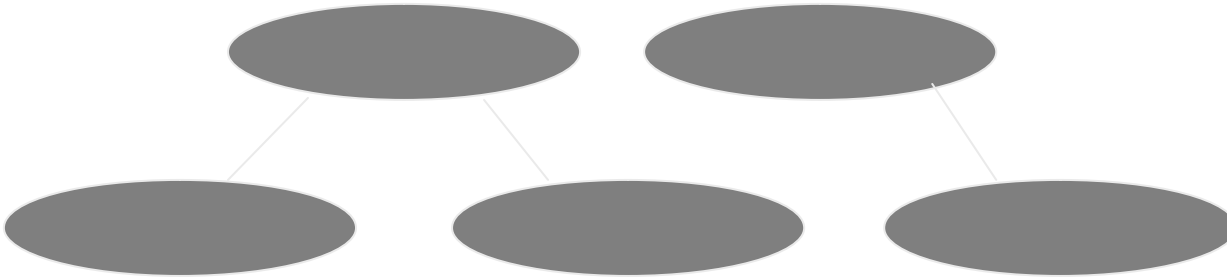
Ernst W. Biersack

Pascal A. Felber

Institut Eurécom – Sophia Antipolis, France

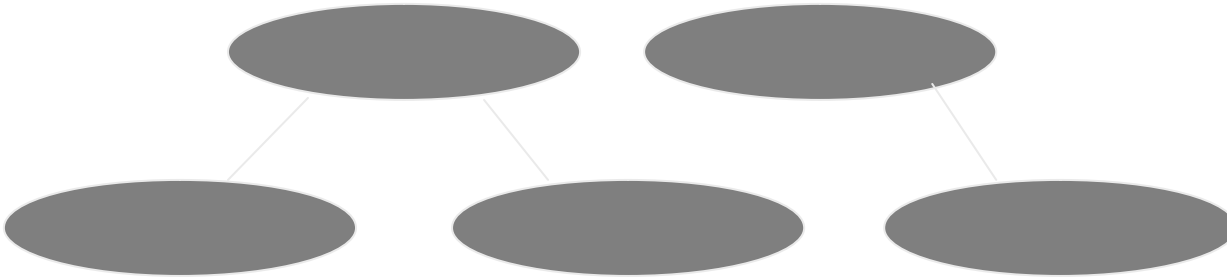
Outline

- Our Goals
- Introduction to TOPLUS
- How MULTI+ works
- Analyzing MULTI+ Multicast trees
- Conclusion and Perspectives



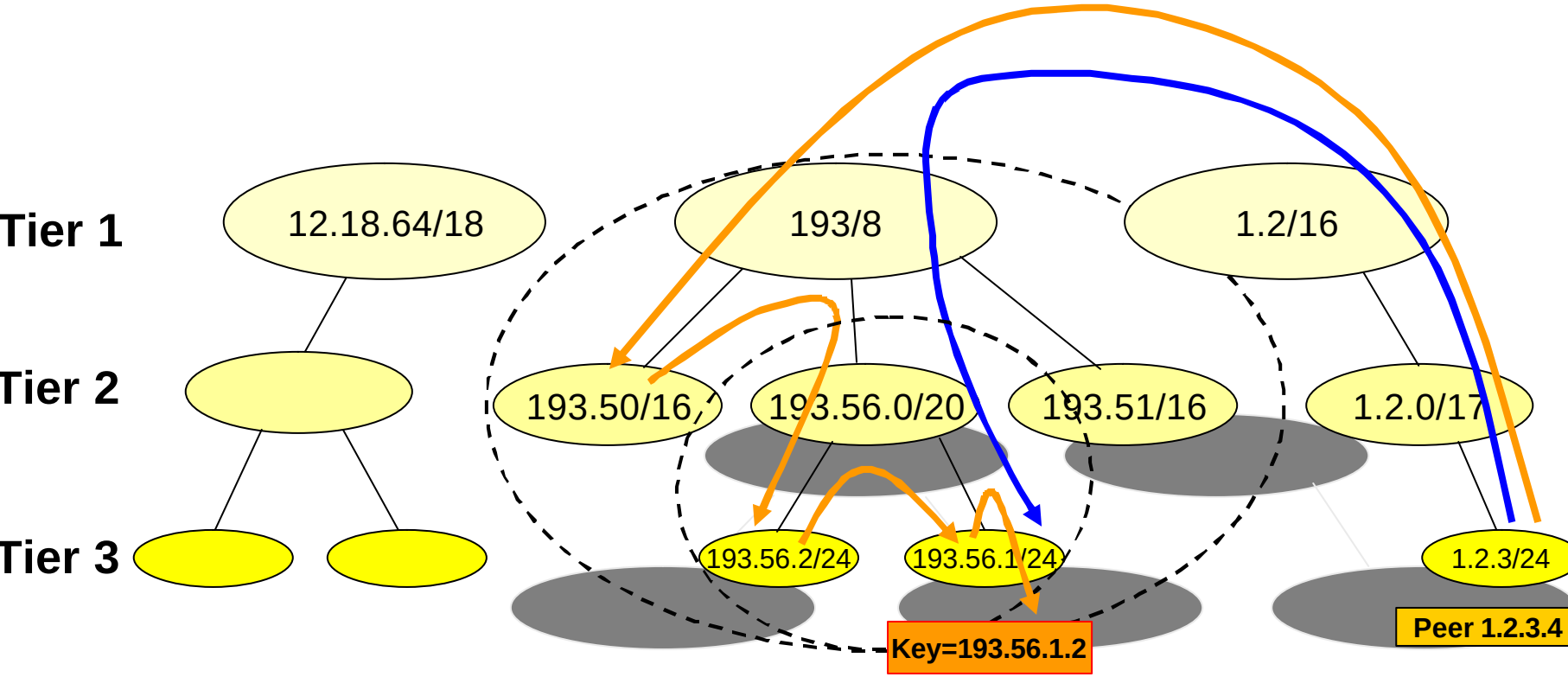
Goals

- Propose an Application-level Multicast that:
  - Takes topology into account for better bandwidth usage
  - Provides low root-to-leaf Multicast tree latency
- Avoiding:
  - Constant probing
  - Periodic rearrangement of peers to optimize Multicast trees
- Thus:
  - Providing good enough Multicast trees by construction



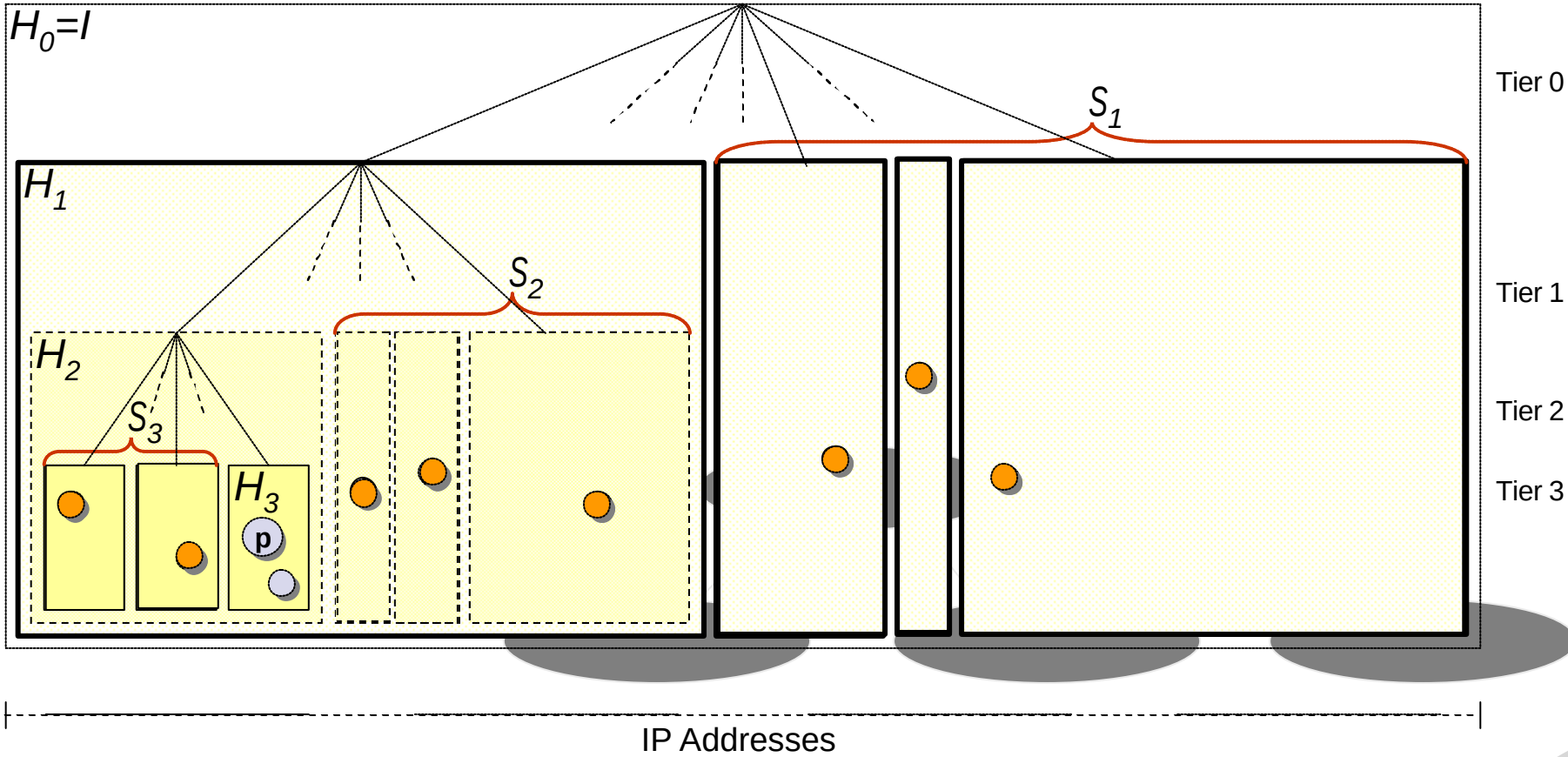
TOPLUS

- **Peer ID = IP@ of peer**
- IP Prefix ranges define (nested) groups
- Every peer needs to know a “delegate” in (some of) the other groups



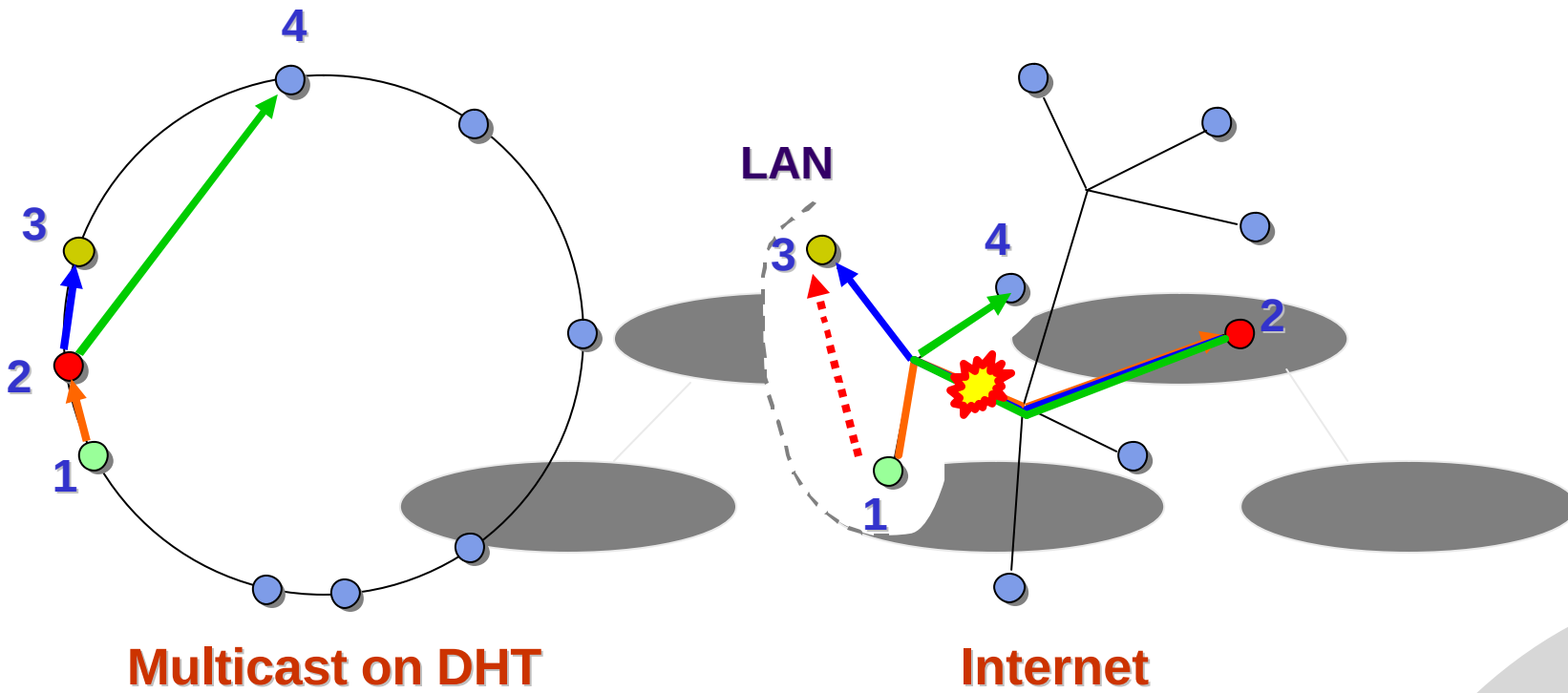
TOPLUS: Peer State

- More detail about closer groups
- Coarser knowledge about groups far away



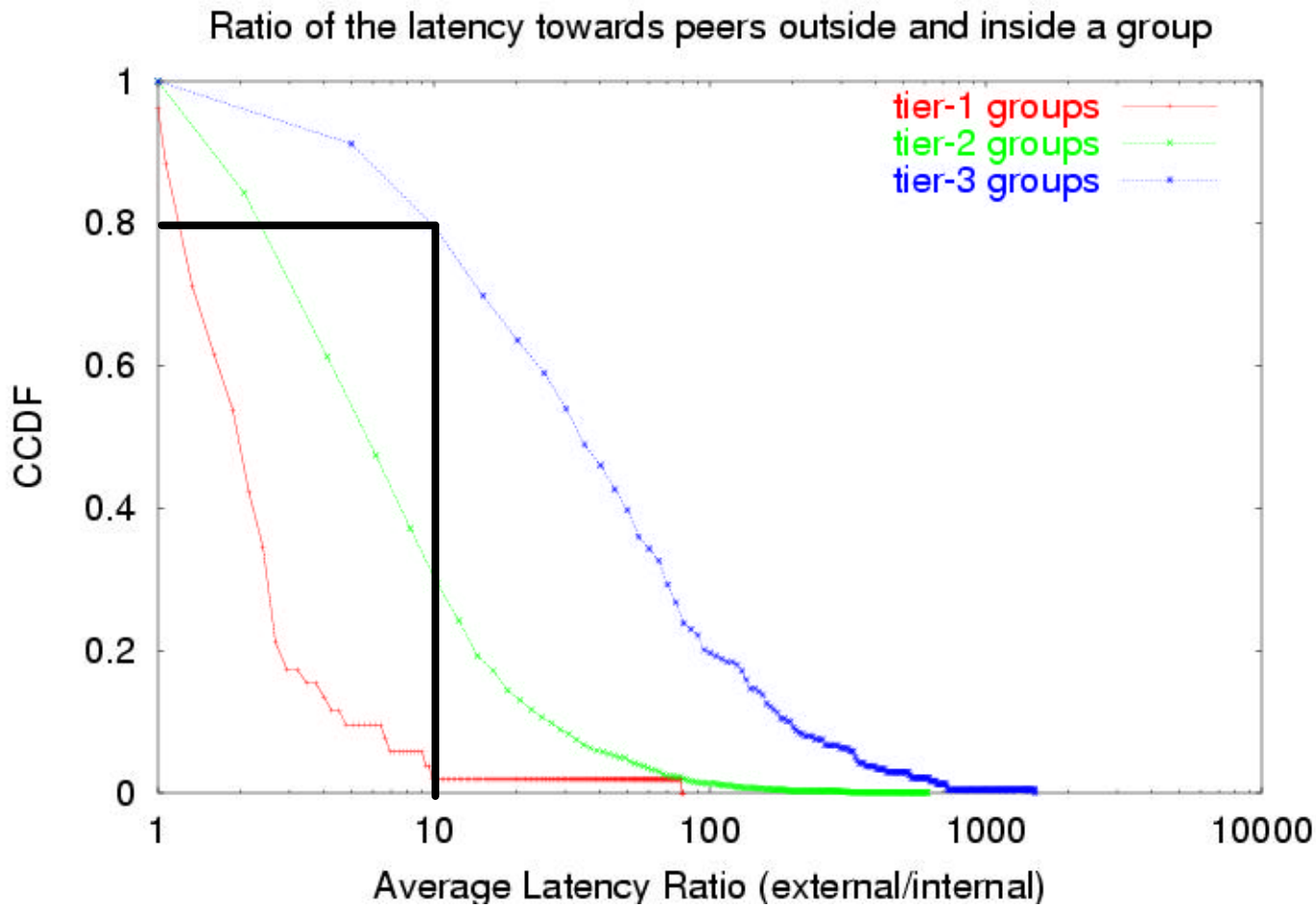
Topology-Awareness

- Application-level Multicast on P2P networks
  - Peers are not aware of the Internet topology
    - End-to-end latency increases
    - Inefficient bandwidth usage



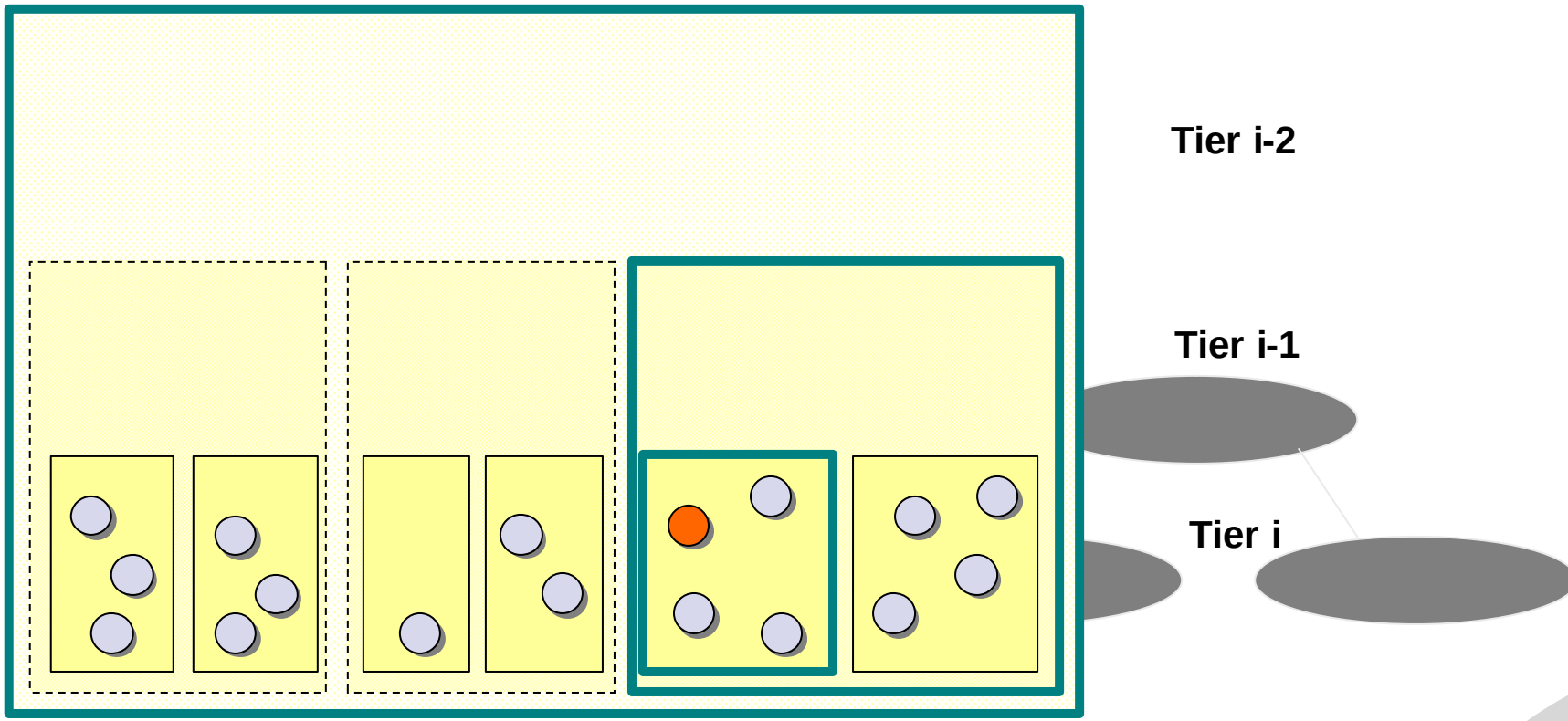
Locality in TOPLUS

- The deeper in the hierarchy, the more local the peers



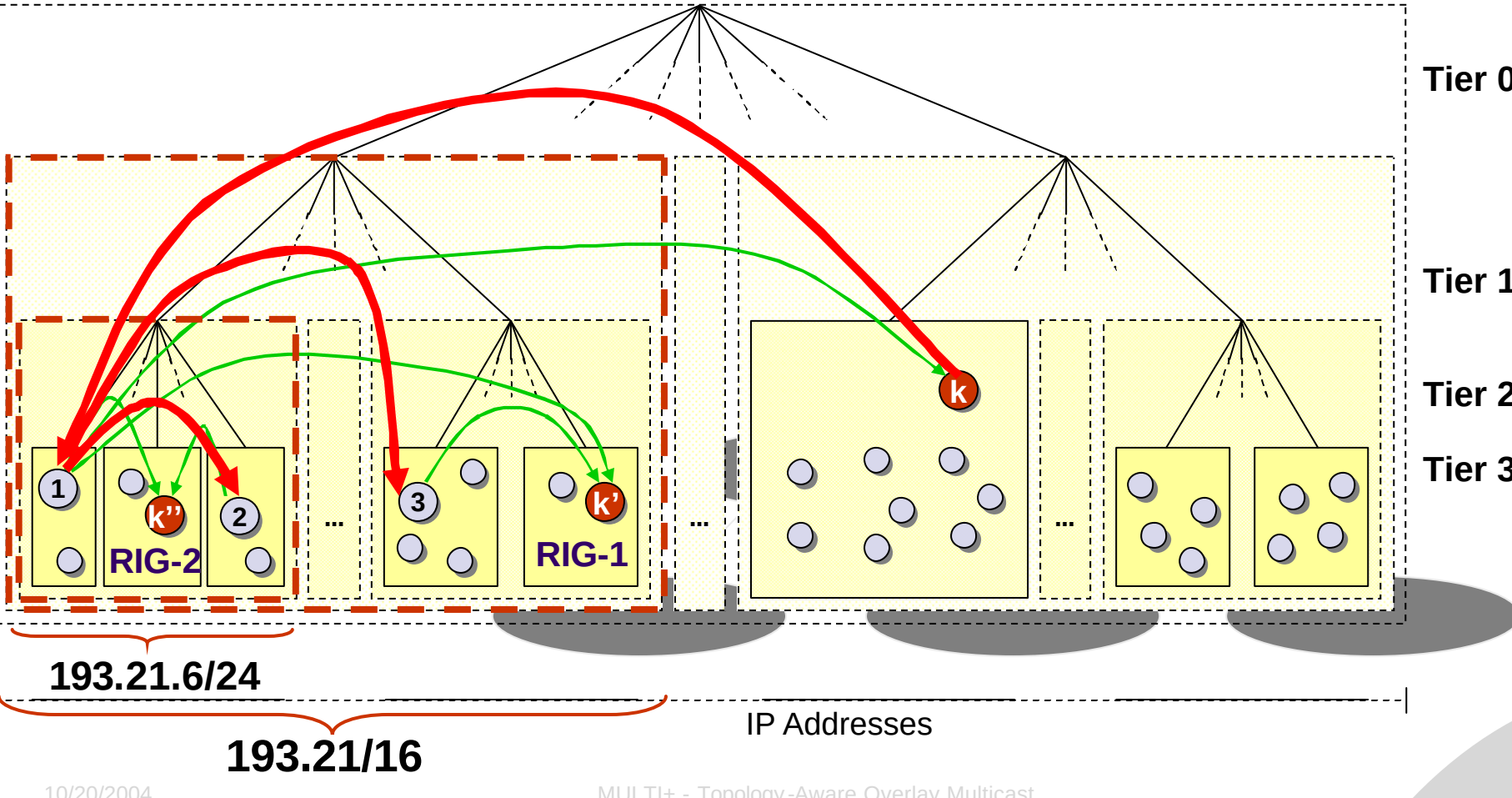
How does MULTI+ work ? (I)

- In TOPLUS, a peer is contained in telescoping sets
- MULTI+ attempts to find a close parent in the innermost scope set where one is available



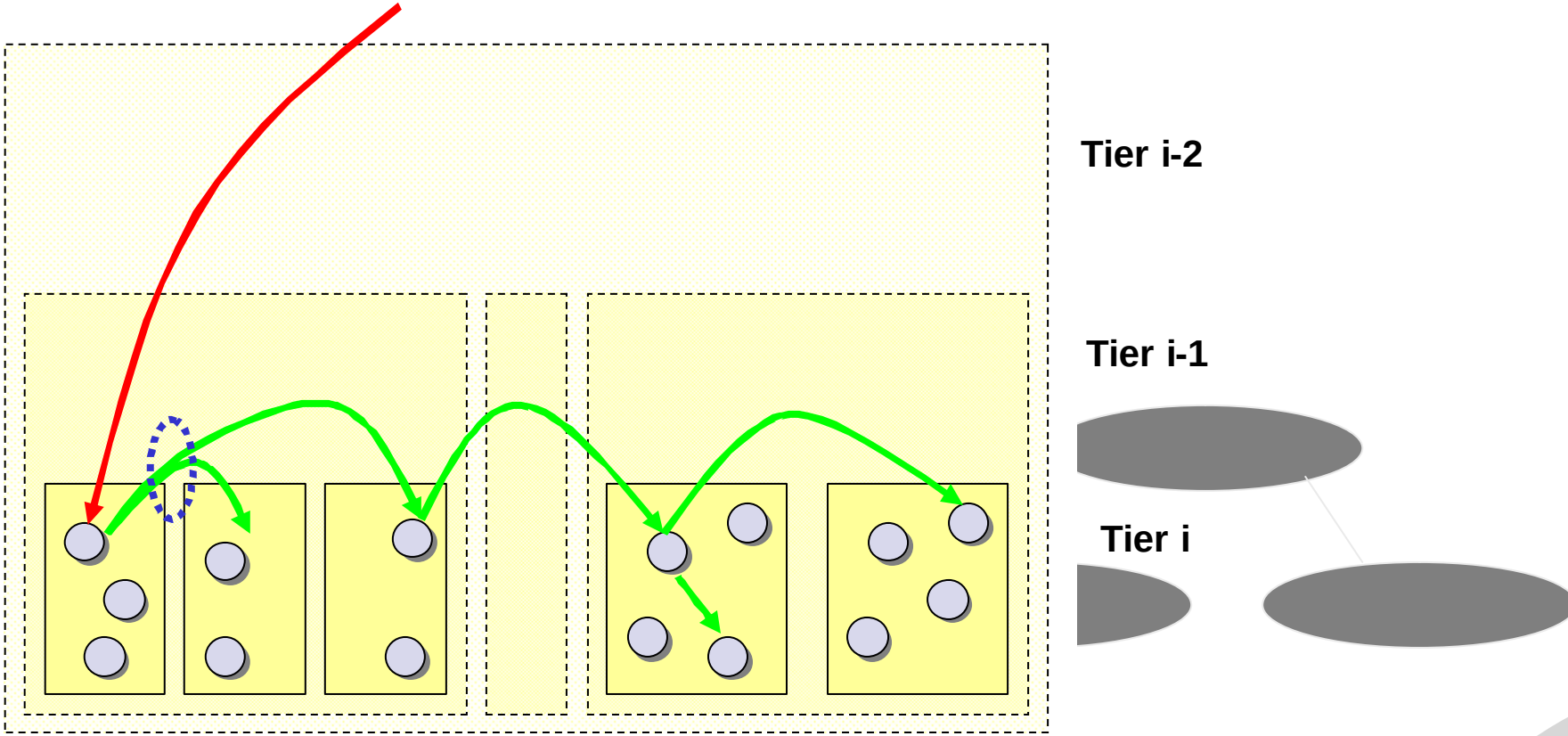
Finding a close parent

- Multicast address (group) **m** ➡ key **k**



Efficient bandwidth utilization

- One inbound flow per network
- Bounded number of outbound flows



Parent Selection Algorithm

- ***RIG***'s actually provide a list of peers
- How do we choose one as parent ?
  - **FIFO**: We choose the first in the list with an available connection
  - **Proximity-aware**: When the first in the list has no available connections, choose the closest peer
- First in the list has connected before the others, so it is closer to the source (hops)
  - Minimize hops to the source
  - Else, connect to a close parent
    - Just by trusting TOPLUS: FIFO
    - TOPLUS selection refinement: Proximity-aware

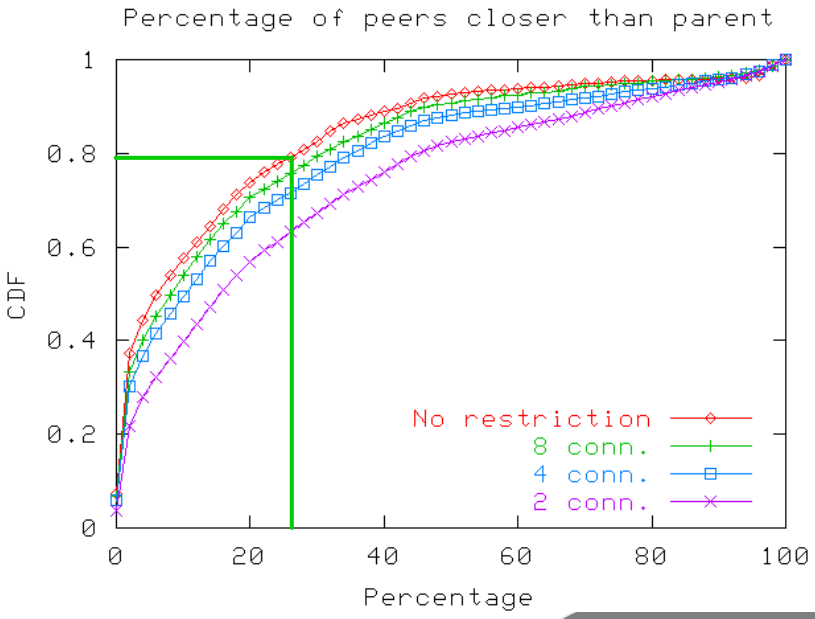
Experimental Evaluation

- P2P Systems: Very large networks
  - Experiments must account for thousands of peers
  - Can't determine connections *a priori*
  - **O(n^2)** cost full inter-host distance matrix
- How do we evaluate deployment in the Internet?
  - **Ping**: Log on each host, measure latency to all others
  - **King**: Use DNS to estimate latencies
  - **Coordinate space**: Each host is a point, euclidean distance between points estimates latency
    - **TC Coordinates**: From **T**ang and **C**rovella
- Experiments using **5,000** peers

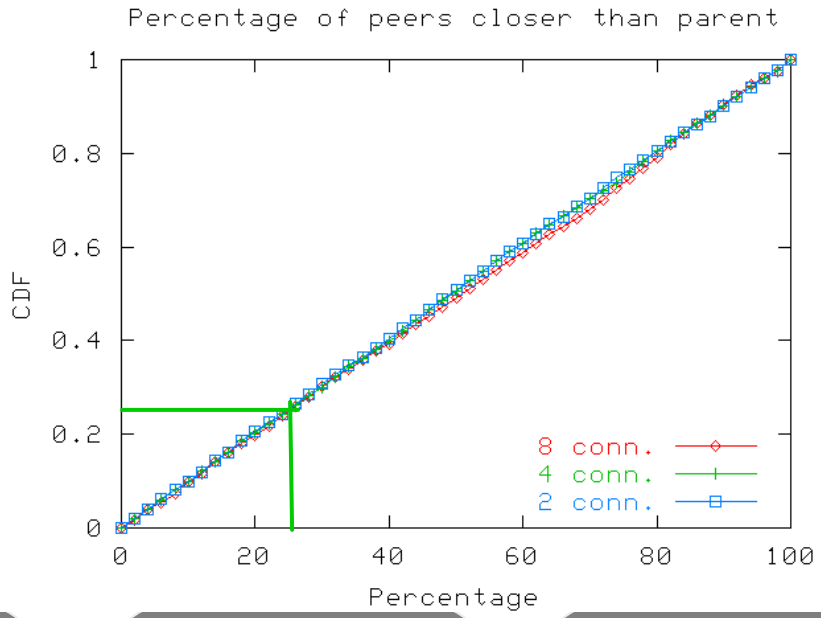
Closeness to parent

- In most cases, MULTI+ peers connect to a parent that is closer than most other peers

FIFO



Random

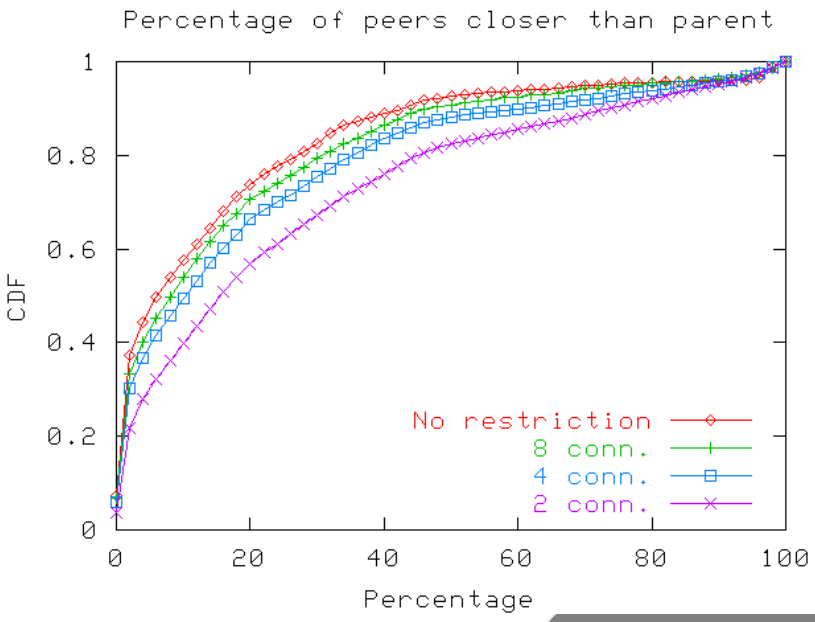


- Only a small percentage of peers is closer to a peer than its own parent (for most peers)

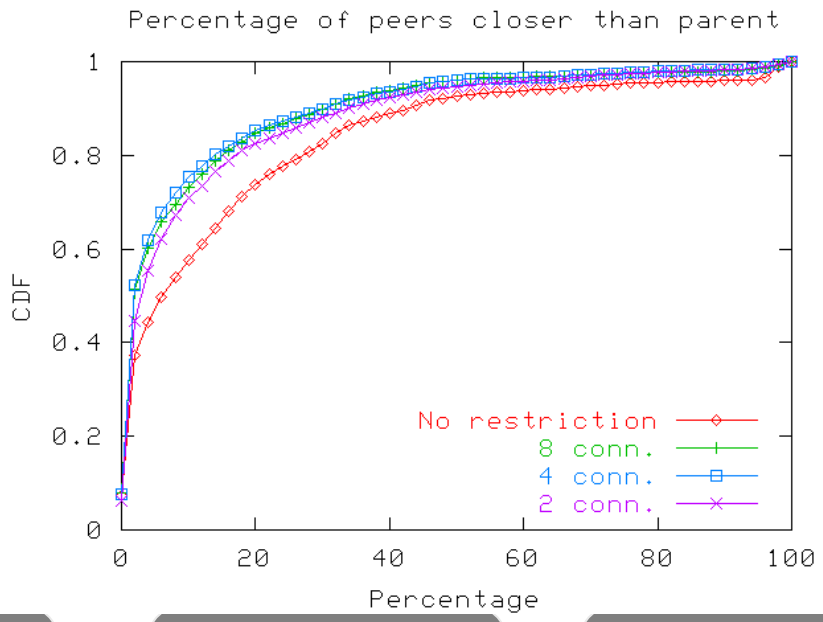
Closeness to parent

- Proximity-aware selection improves latency to parent

FIFO



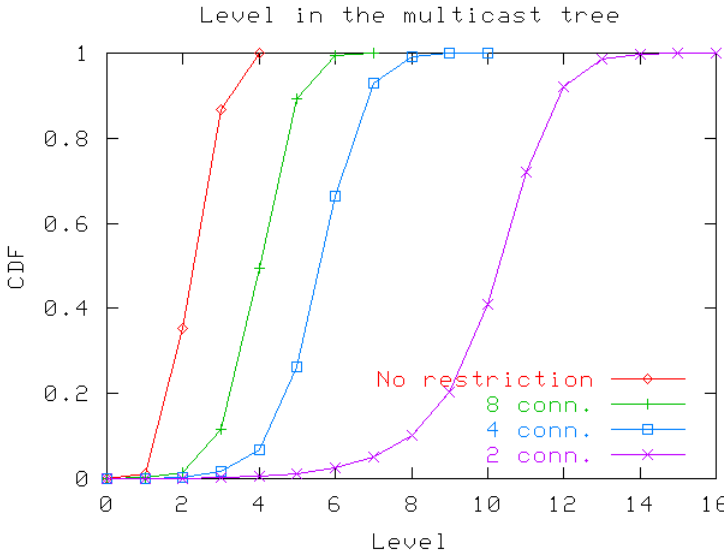
Proximity-aware



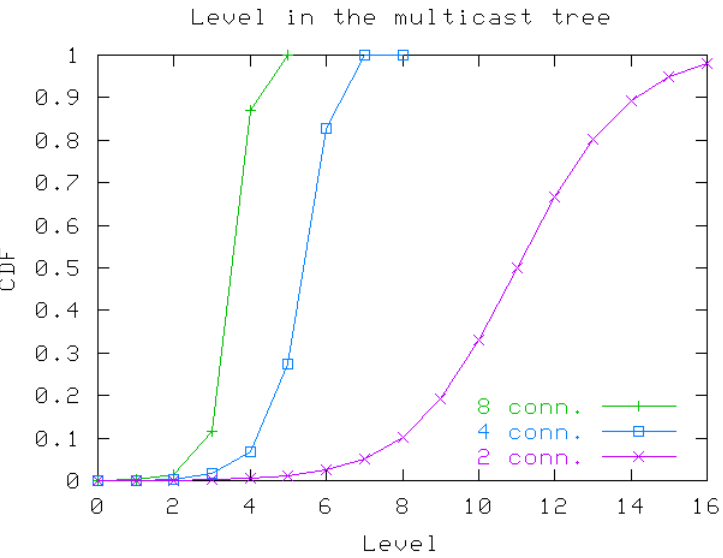
Levels in the Multicast Tree

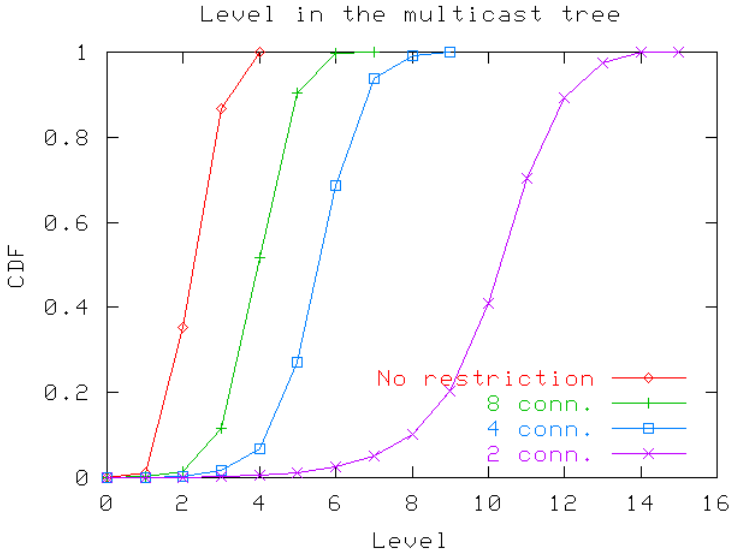
- Small effect of topology-aware constraints

FIFO



Random



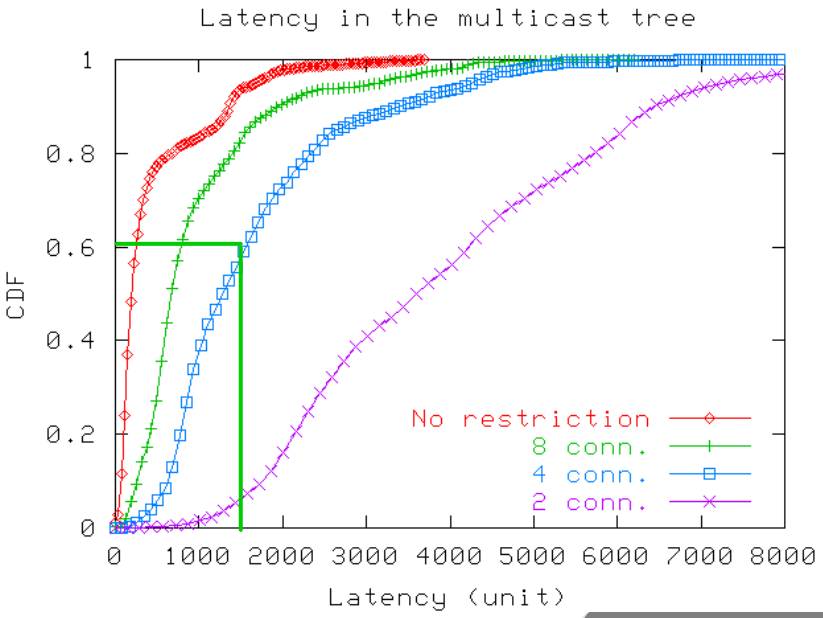


Proximity aware

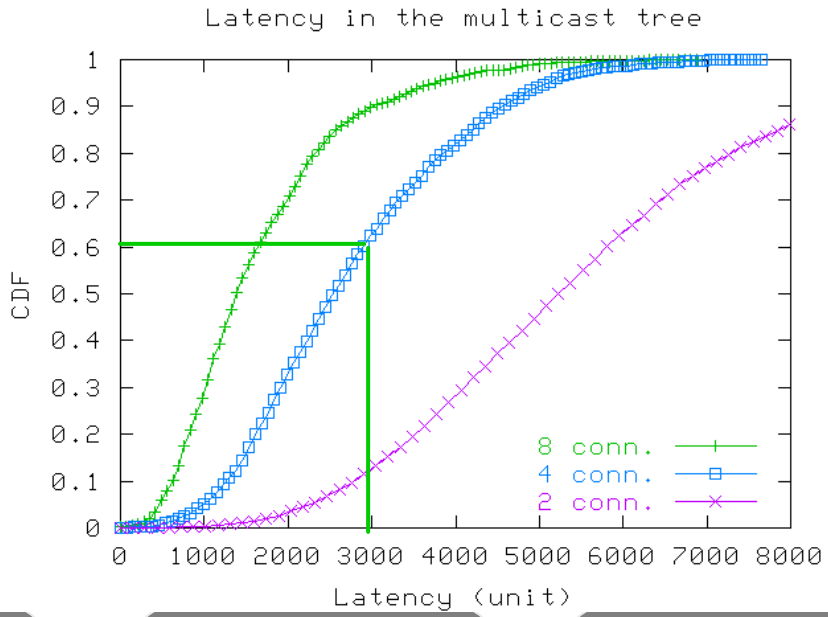
Root-to-Leaf Latency

- Without probing, MULTI+ improves end-to-end latency

FIFO



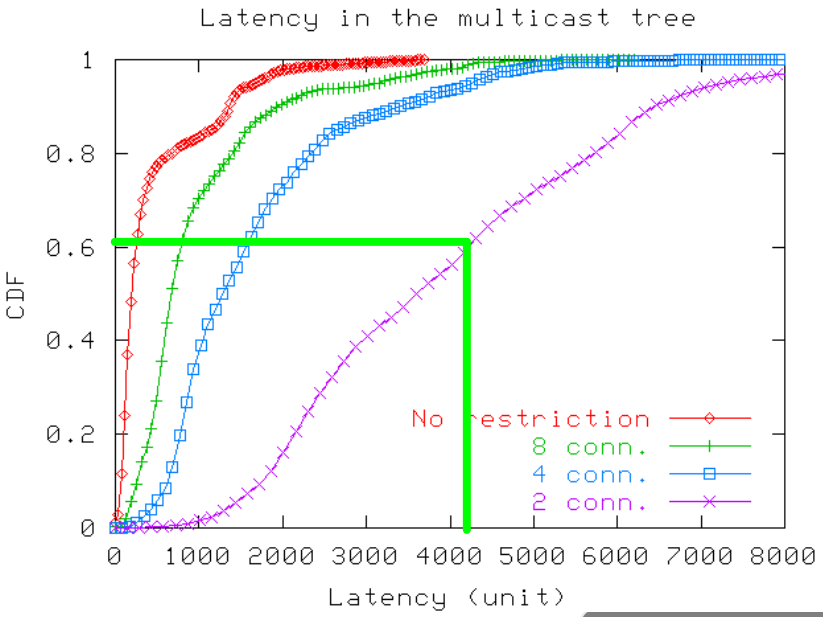
Random



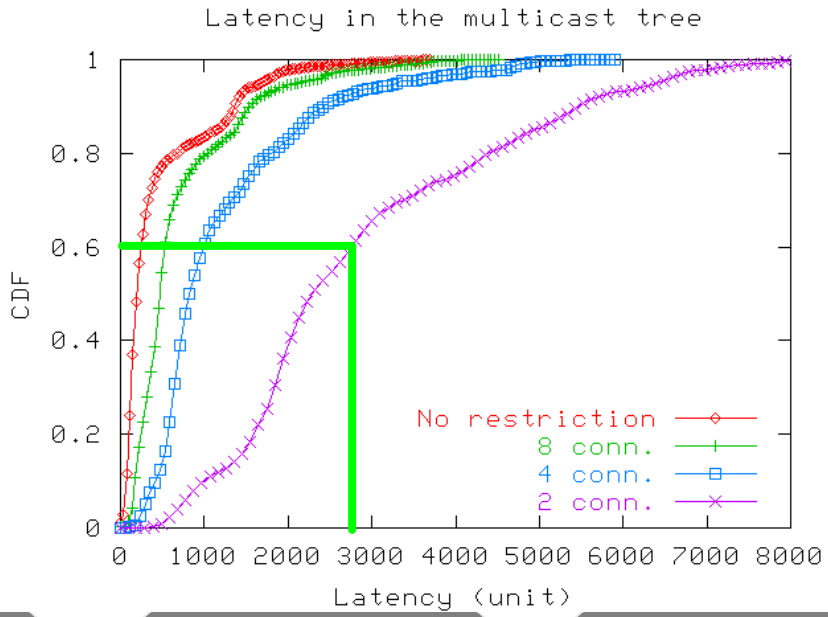
Root-to-Leaf Latency

- Proximity-aware slightly improves end-to-end latency

FIFO



Proximity-aware



Average Root-to-Leaf Latency

- In TC Coordinate Space units

Connections / Parent	2	3	6	8
Random	**5,339**	3,291	2,072	1,646
FIFO	3,905	2,527	1,135	965
Proximity-aware	**2,871**	1,630	797	**698**

Conclusion & Perspectives

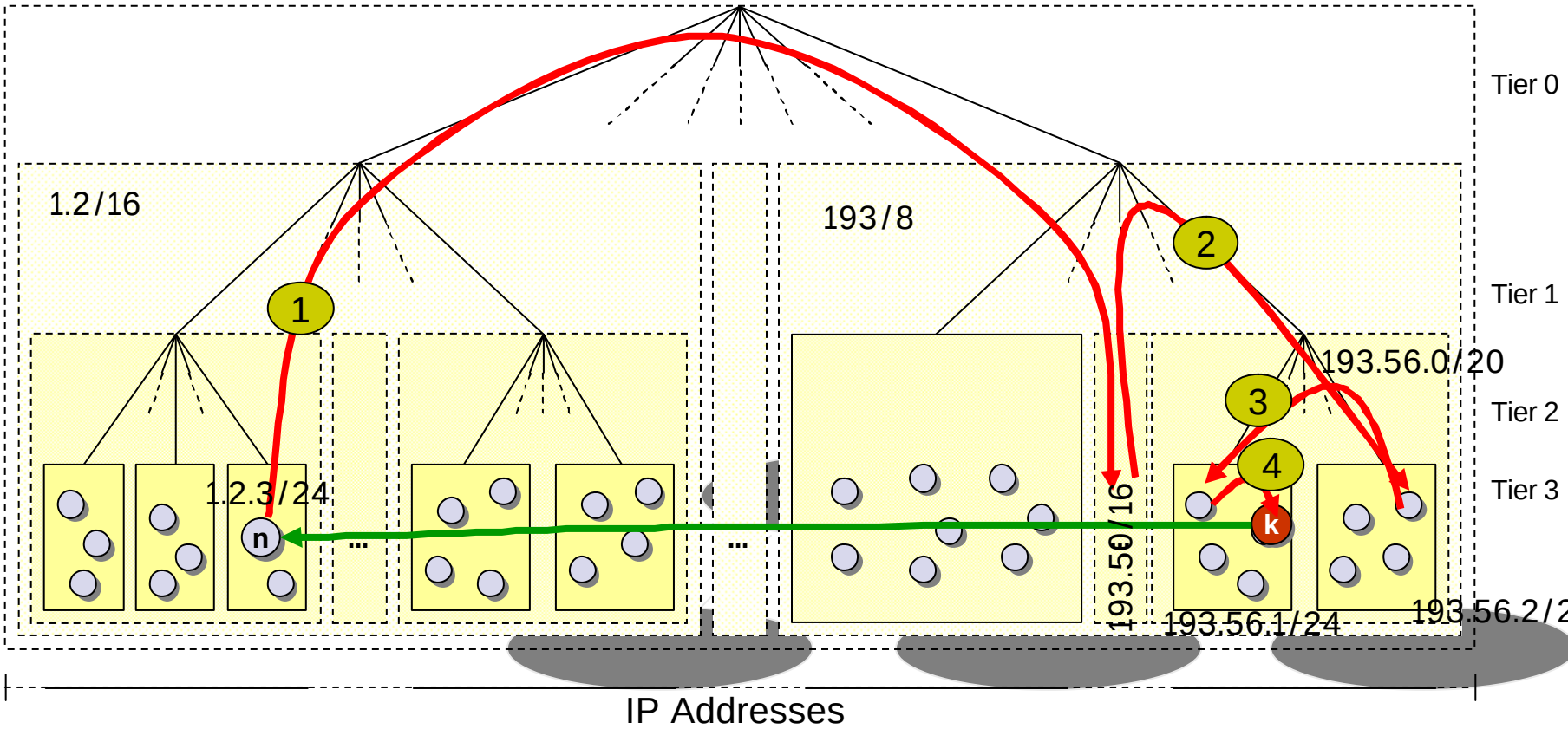
- MULTI+: Topology-aware Overlay Multicast trees
- Features:
  - Low end-to-end delay without introducing extra probing traffic
  - Improves bandwidth utilization
  - Small (4) number of connections per peer sufficient to assure low latency
- Current work:
  - Effect of peer failure on tree properties
  - More straightforward TOPLUS network layout

Introduction to DHTs

- P2P Lookup Services
  - **Assign** resources to peers
  - **Locate** resources upon request
- Structured P2P systems:
  - A **key** is a resource identifier.
  - A Hash Table maps a key to a bucket through a hash function **h()**
    - **h(**key**)** $\rightarrow$ 0,1,2 … N-1 , all N possible outputs of **h()**
  - In Distributed Hash Tables (DHTs) each bucket is a peer
    - Key assigned to peer with the **"closest" id**
- Chord, CAN, Pastry
  - Scalability: Each peer knows a subset of all peers
  - **h(**key**)** must be **routed** to corresponding "bucket"

TOPLUS: Key Look-Up (I)

- Node **n=1.2.3.4** looks-up key **k=193.56.1.2**



Number of hops £ H+1, H = height of tree

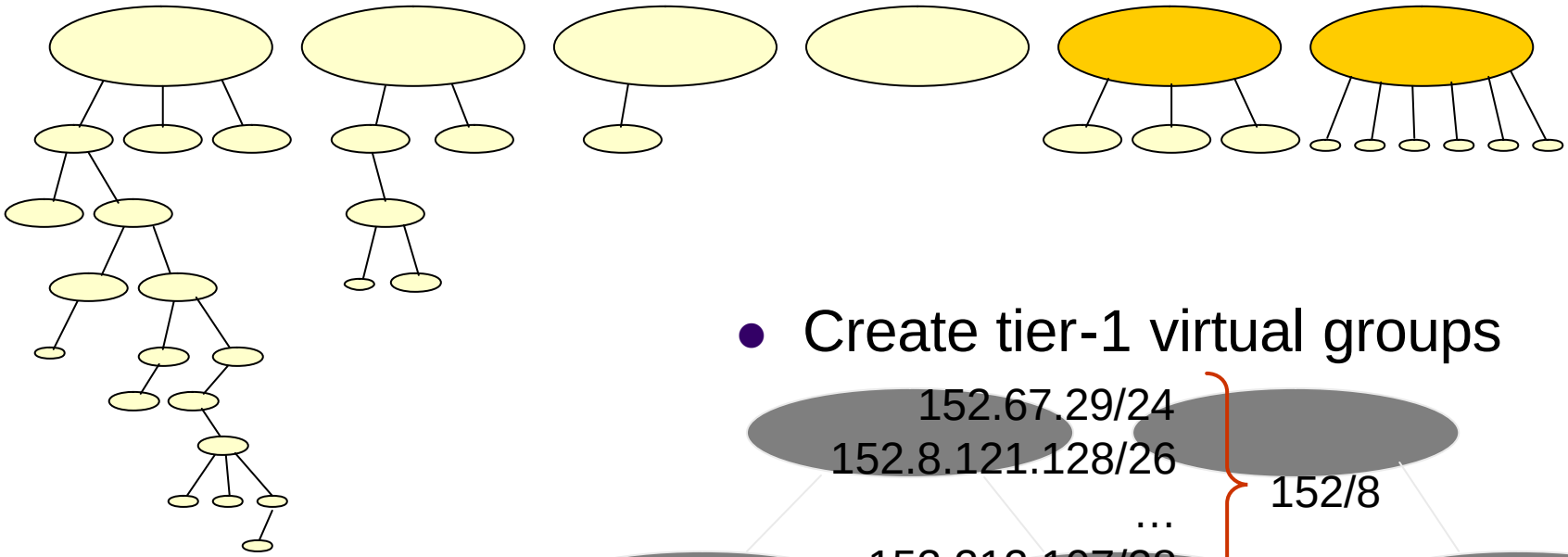
TOPLUS: Key Look-Up (II)

- Each key *k* is a bit string of length **m** > 32*:*
  - First 32-bits used for routing to node responsible for that key
- XOR metric:
  - Let node $j = j_{31}j_{30}\ldots j_0$ and $k = k_{31}k_{30}\ldots k_0$: $d(j,k) = \sum_{i=0}^{31} |j_i - k_i| \cdot 2^i$
  - Note that closest ID is unique:
    - $d(j,k) = d(j',k)$ Û $j = j'$
  - Refinement of longest-prefix match
  - Example (8 bits)

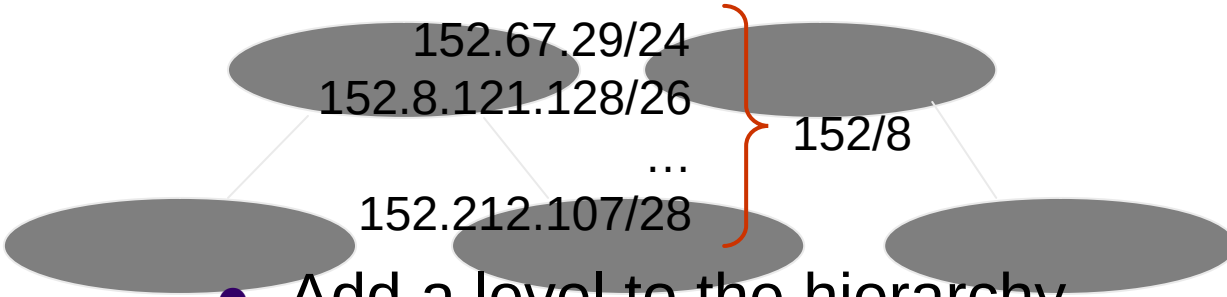
k	= 10010110	
j	= 10110110	d(j,k) = 2^5 = 32
j'	= 10001001	d(j',k) = $2^4 + 2^3 + 2^2 + 2^1 + 2^0$ = 31

Partial Order Tree Construction

- **Modified** trees:
  - Reduce routing table size
  - Respect topology as possible



- Create tier-1 virtual groups

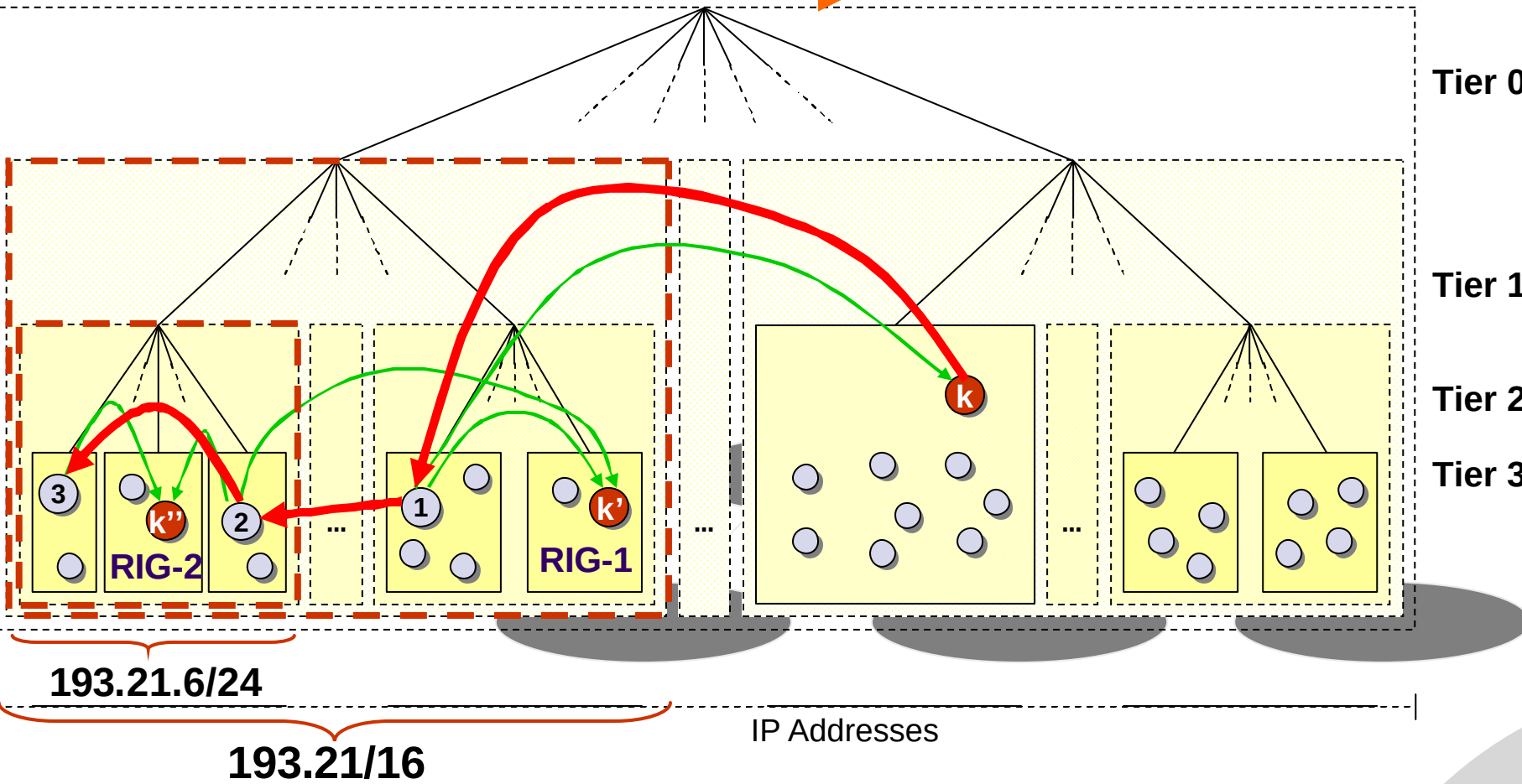


- Add a level to the hierarchy

Finding a close parent

k = 212.17.89.100 $\xrightarrow{193.21/16}$ k' = 193.21.89.100

k = 212.17.89.100 $\xrightarrow{193.21.6/24}$ k'' = 193.21.6.100



How does MULTI+ work ?

- Given:
  - Multicast address (group) ***m***,
  - its corresponding key ***k***
- Inside a given group ***x.y.z.w/b*** at tier-***i***
  - ***k'*** = Substitute the ***b*** first bits of ***k*** by those of ***x.y.z.w***
  - ***RIG-i*** is the **R**esponsible **I**nner **G**roup for ***k'***
  - Ask a peer in ***RIG-i*** for a parent
  - Proceed to group at tier-(**i-1**) if no parent available
- ***RIG***'s memorize peers asking for a parent in Multicast group ***m***
  - Subsequent peers receive previous peers as parents

TC Coordinate Space

- From **T**ang and Pr. **C**rovella
- A 12-dimensional space representing the Internet
  - More than **190,000** hosts
  - Each host is a point in the TC coordinate space
  - Estimation of latency:
    - Euclidean distance between two hosts x,y
    - d(x,y)= $\sqrt{\sum_{i=1}^{12}(x_i - y_i)^2}$
- Can substitute active measurement for proximity-aware parent selection

Flows per Network

- Maximum number of flows found in one network

Connections / Parent	2	3	6	8
Random	**885**	919	931	897
FIFO	260	198	145	107
Proximity-aware	**165**	109	39	**55**