



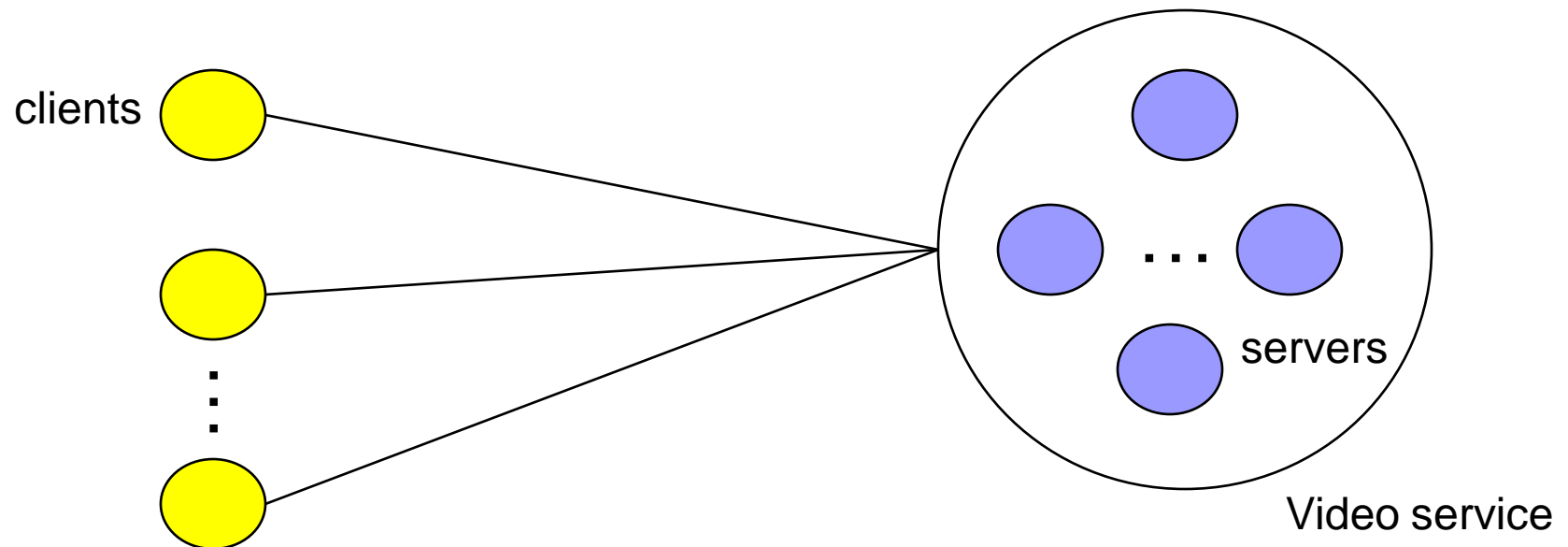
CS 414 – Multimedia Systems Design

Lecture 24 – P2P Streaming

Klara Nahrstedt
Ramsés Morales

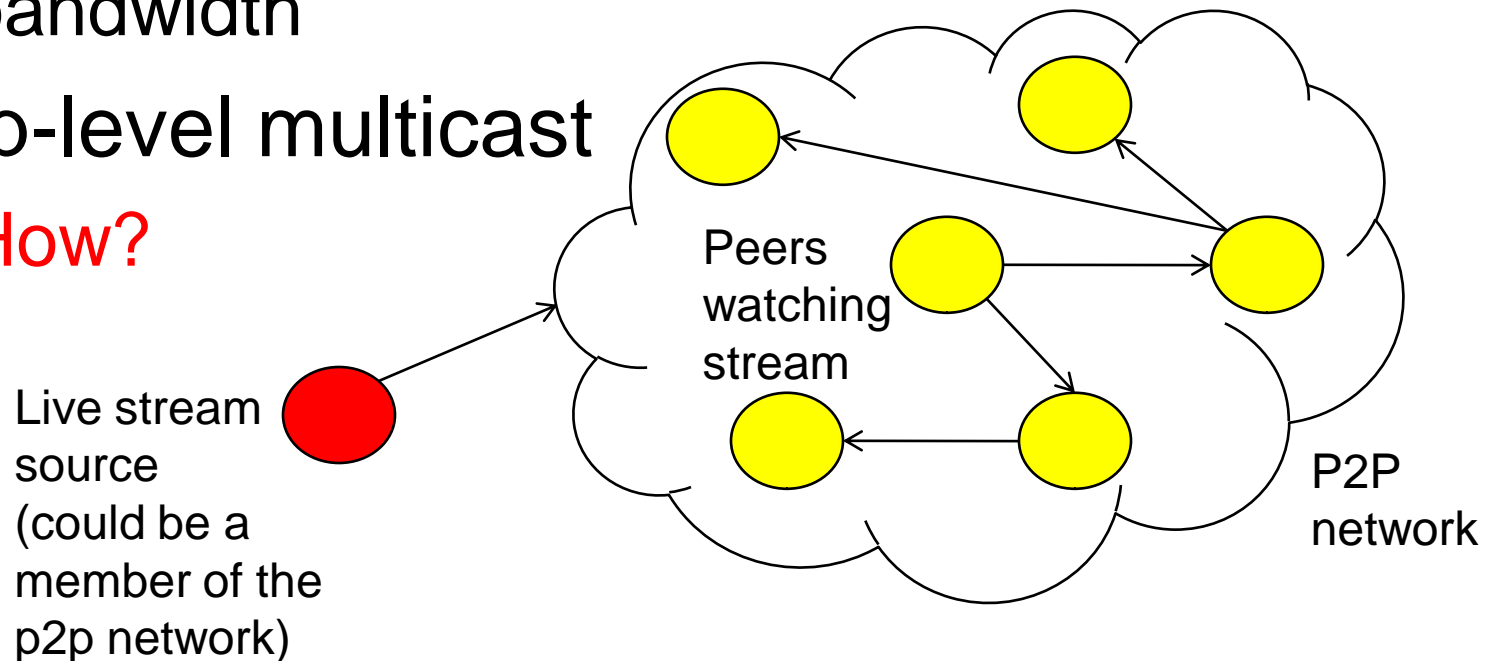
Streaming from servers

- Bandwidth at video service and number of servers have to grow with demand
 - Flash crowds have to be taken into account



P2P Streaming

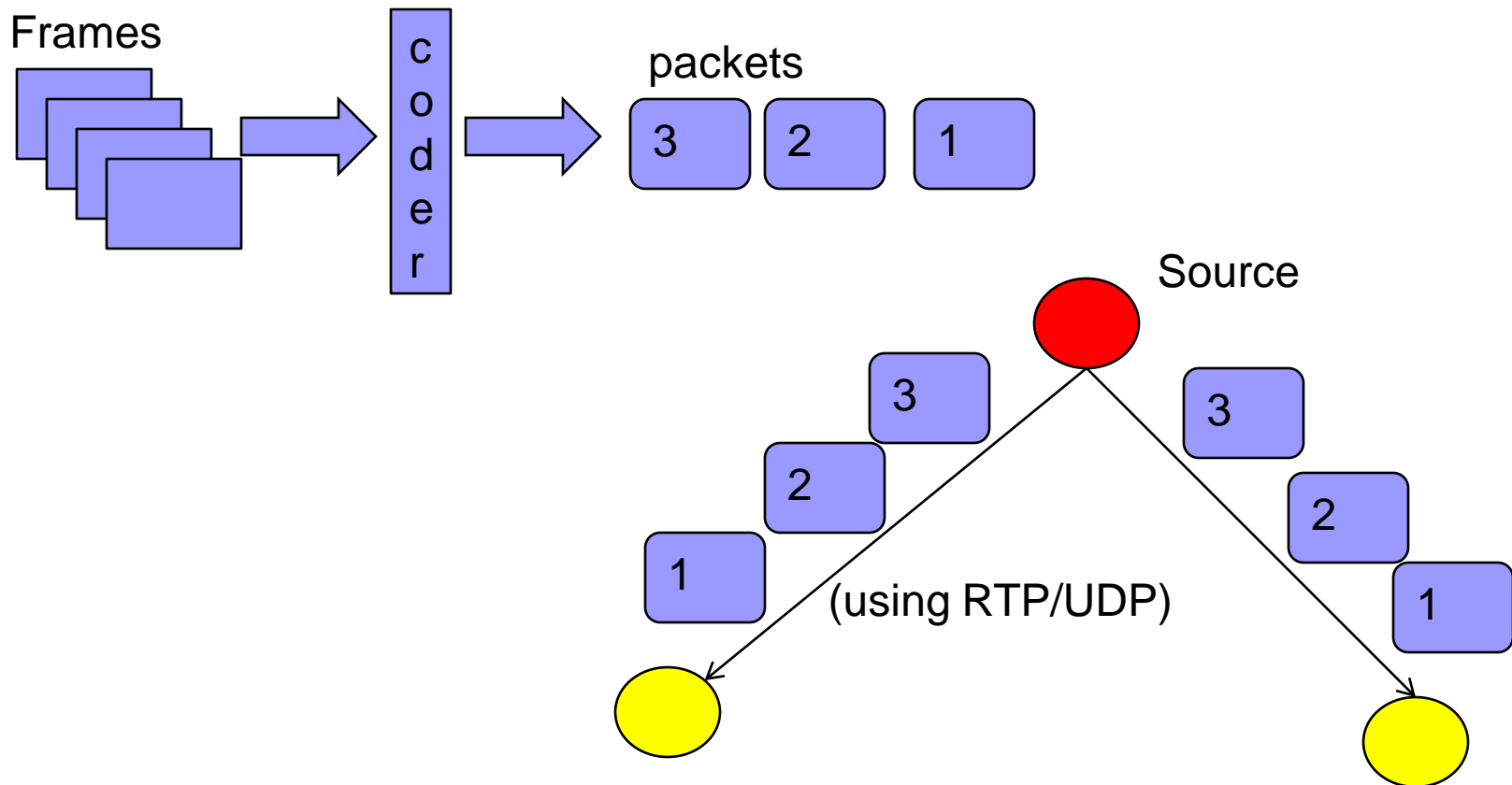
- Use the participating node's bandwidth
 - More nodes watching stream = more shared bandwidth
- App-level multicast
 - How?



P2P Streaming

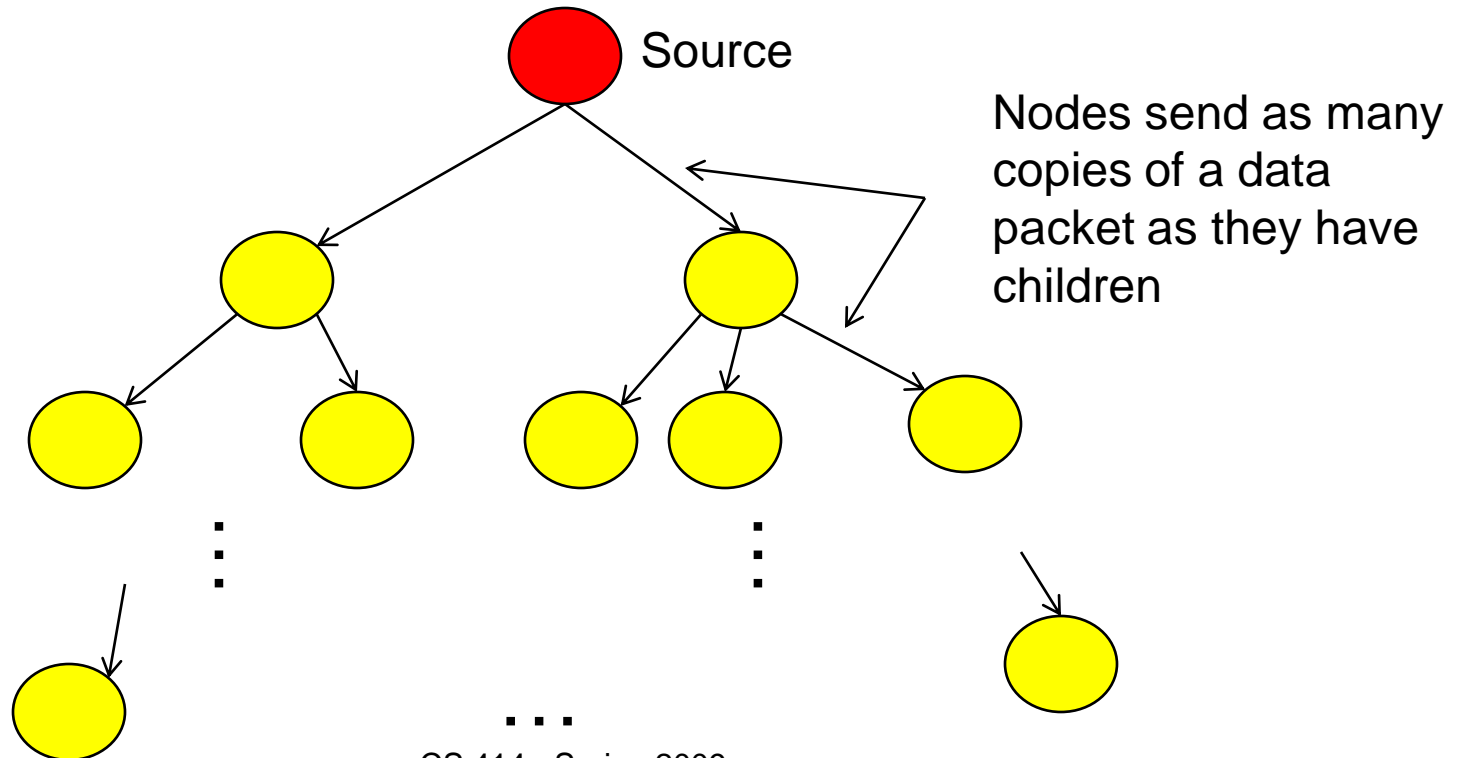
- Common arrangements to multicast the stream
 - Single Tree
 - Multiple Tree
 - Mesh-based
 - All nodes are usually interested in the stream
- They all have to deal with **node dynamism** (join/leave/fail/capacity-changes)

Streaming in a single tree



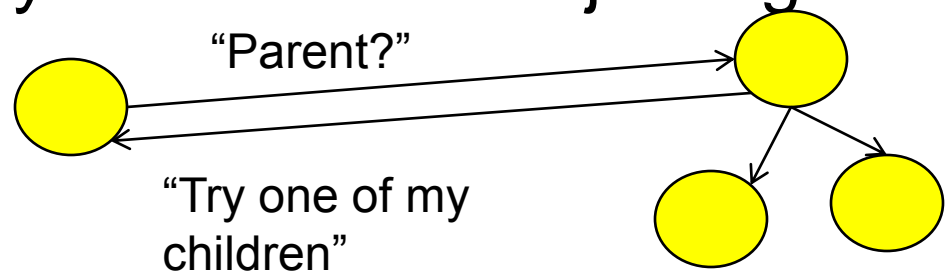
Single Tree

- Peers interested in the stream organize themselves into a tree



Joining the tree

- Find a node with spare capacity, then make it parent
- If contacted node lacks capacity, pick child
 - Random child
 - Round robin
 - Child closest in physical network to joining node

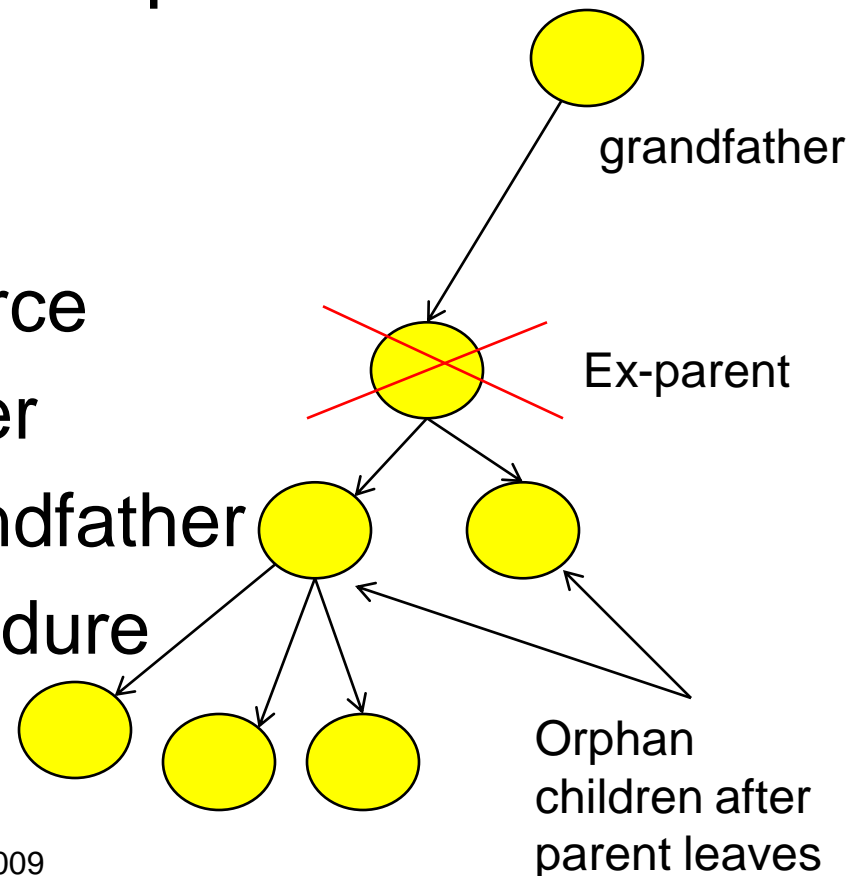


Leaving the tree or failing

- Orphan nodes need a new parent

- Policies for new parent

- ☐ Children pick source
- ☐ Subtree nodes pick source
- ☐ Children pick grandfather
- ☐ Subtree nodes pick grandfather
- ☐ ...then repeat join procedure

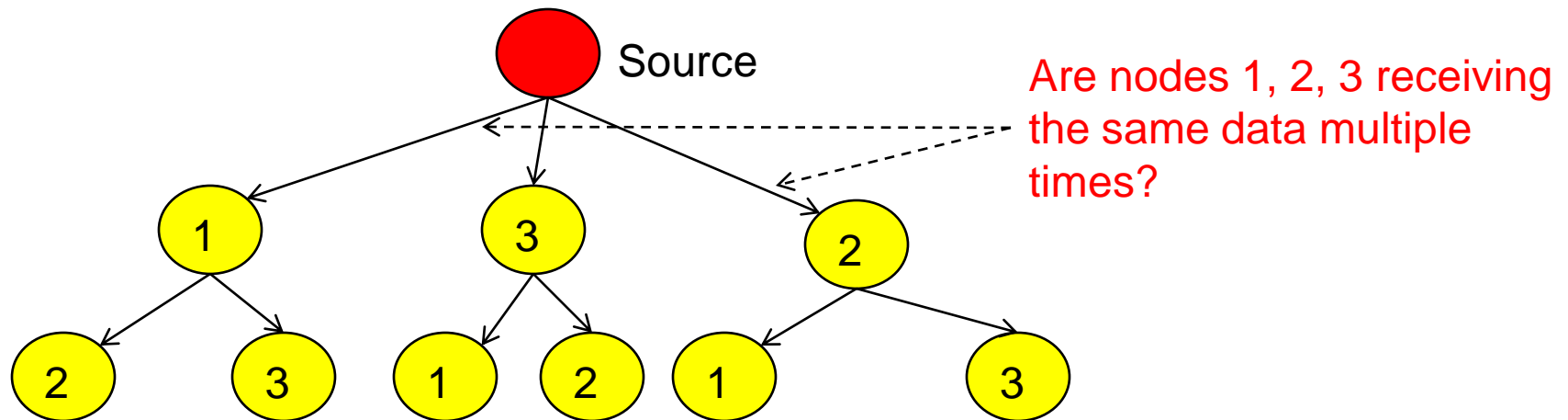


Single tree issues

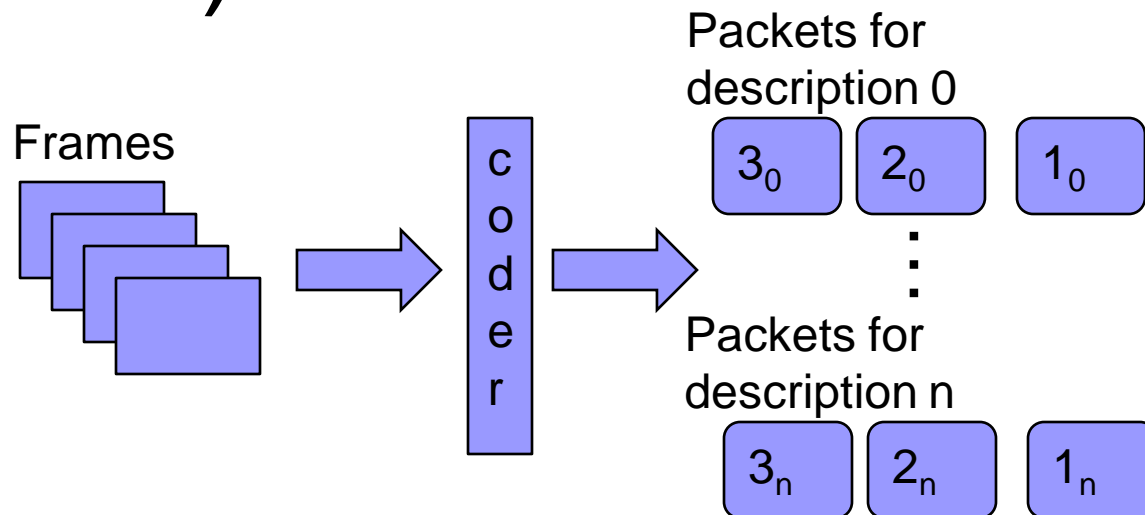
- **Leaves** do not use their outgoing bandwidth
- Packets are lost while recovering after a parent leaves/fails
- Finding unsaturated peer could take a while
- Tree connections could be rearranged for better transfer

Multiple Trees

- **Challenge:** a peer must be internal node in only one tree, leaf in the rest



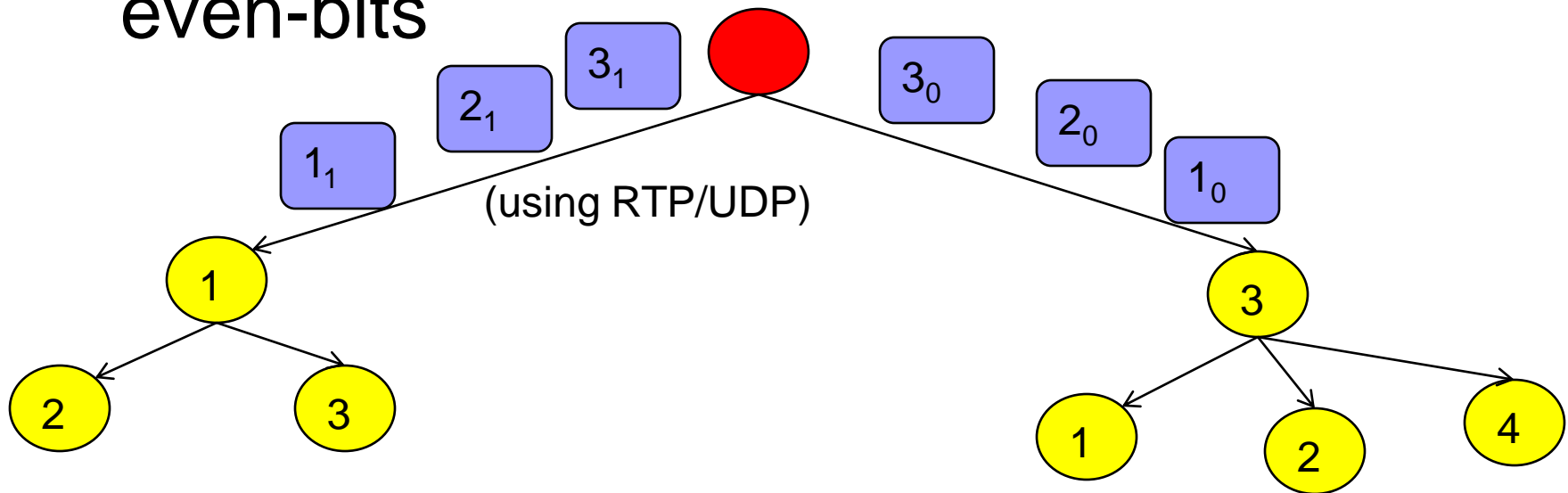
Multiple Description Coding (MDC)



- Each description can be **independently decoded** (only one needed to reproduce audio/video)
 - More descriptions received result in higher quality

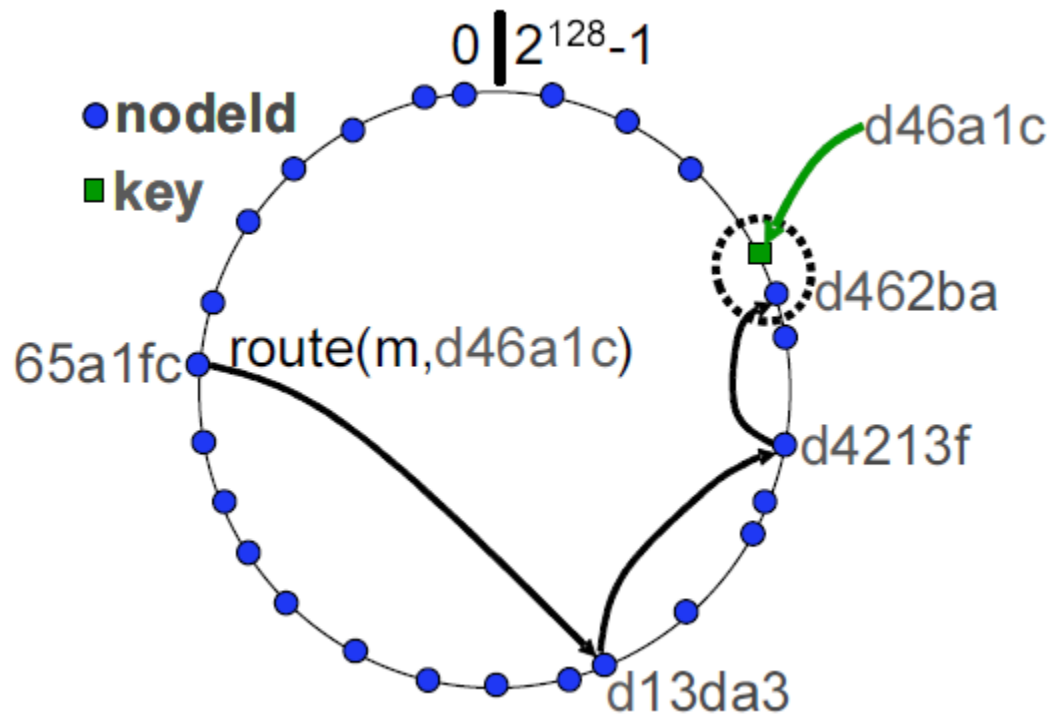
Streaming in multiple-trees using MDC

- Assume **odd-bit/even-bit** encoding -- description 0 derived from frame's odd-bits, description 1 derived from frame's even-bits



Multiple Tree Streaming in Pastry DHT -- SplitStream

- Pastry routes messages using **id prefix matching**

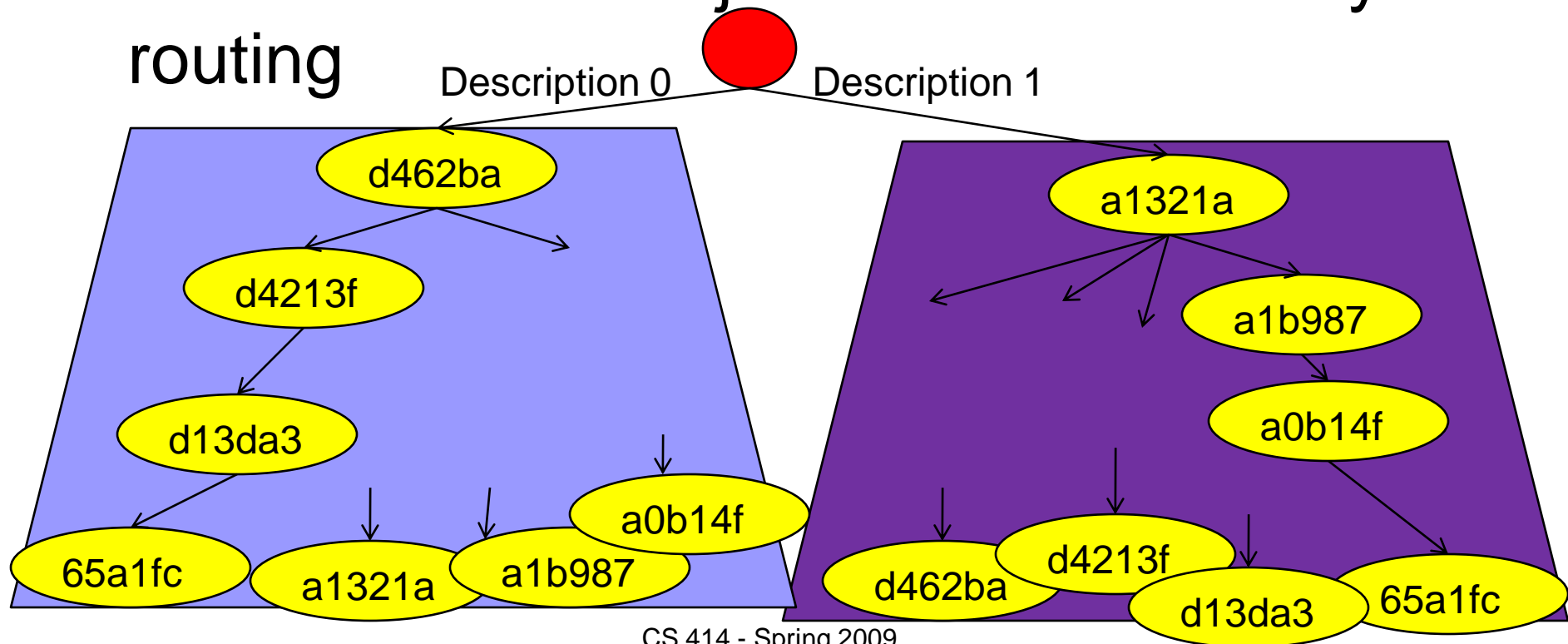


Multiple Tree Streaming in Pastry DHT -- **SplitStream**

- Assign an id to each coded description
 - If most significant digit is different, then trees will be **interior-node-disjoint**, example,
 - Id tree 1: d46a1c
 - Id tree 2: a1321d
 - 65a1fc: route(m, d46a1c) -> d13da3 -> d4213f -> d462ba
 - 65a1fc: route(m, a1321d) -> a0b14f -> a1b987 -> a1321a
 - Stream flows through reverse path

Multiple Tree Streaming in Pastry DHT -- **SplitStream**

- Peer is internal node in only one tree, due to interior-node-disjoint trees and Pastry's routing



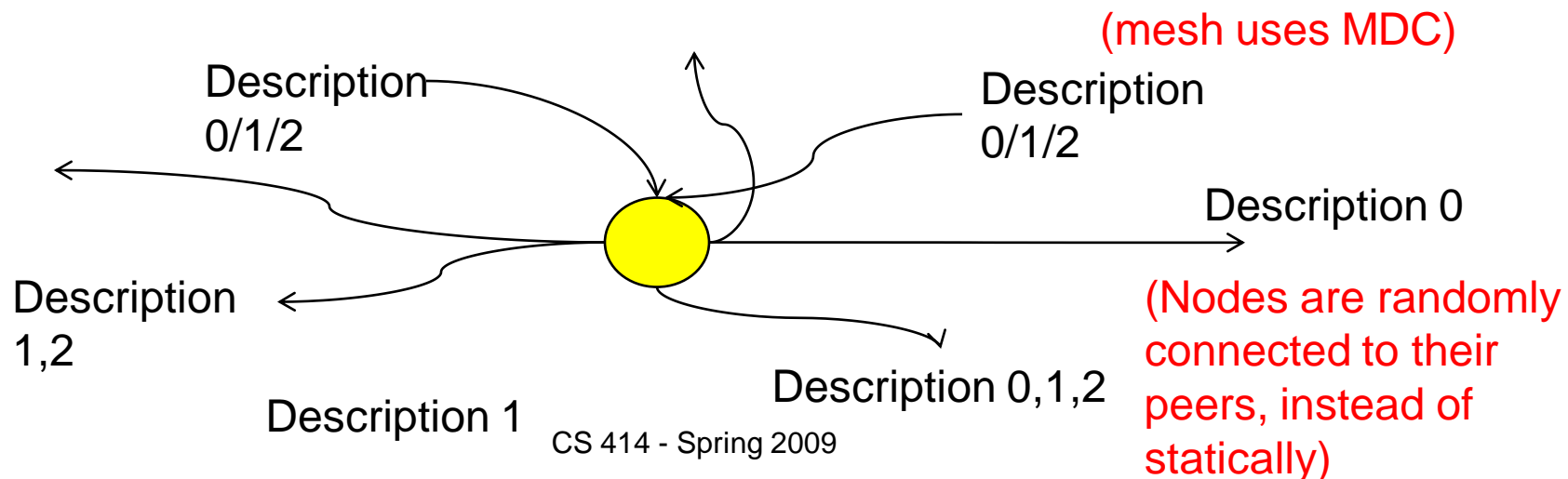
Multiple-Tree Issues

- Complex procedure to locate a potential-parent peer with spare out-degree
 - Degraded quality until a parent found in every tree
- **Static mapping in trees**, instead of choosing parents based on their (and my) bandwidth
 - An internal node can be a bottleneck

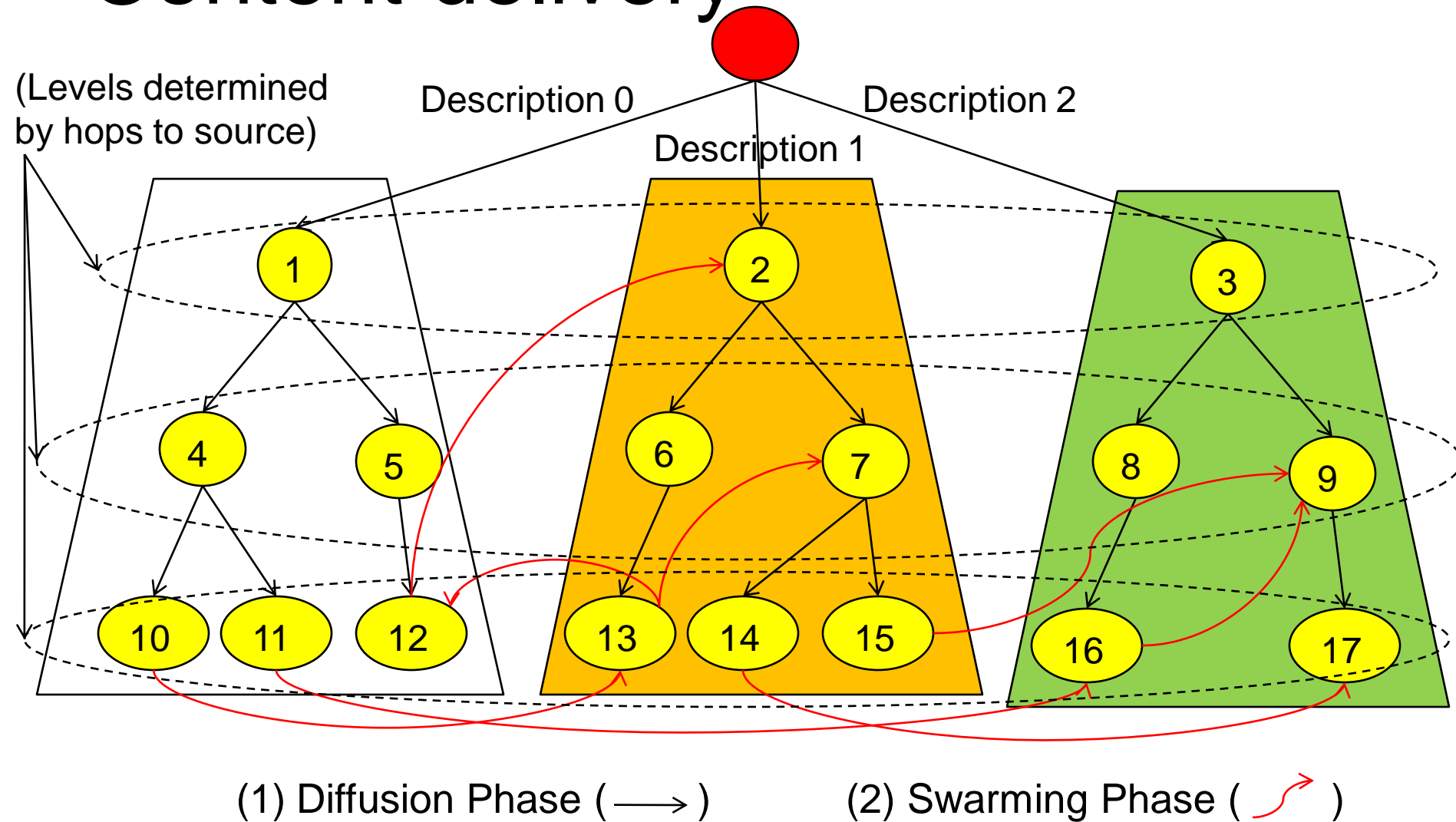
Mesh-based streaming

■ Basic idea

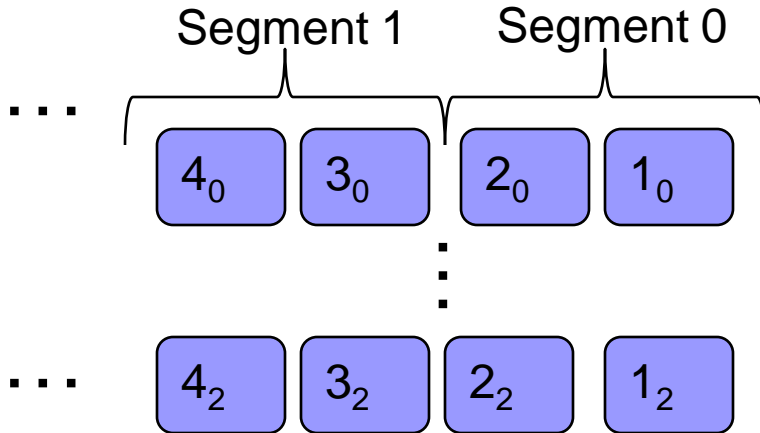
- Report to peers the packets that you have
- Ask peers for the packets that you are missing
- Adjust connections depending on in/out bandwidth



Content delivery

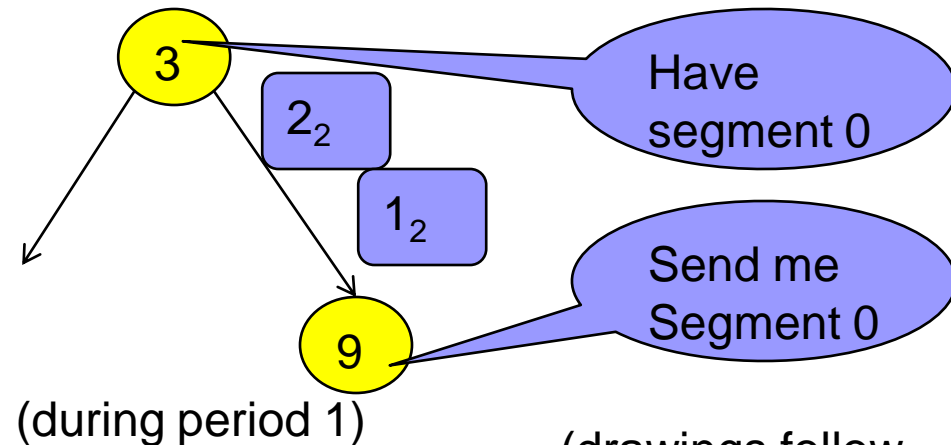
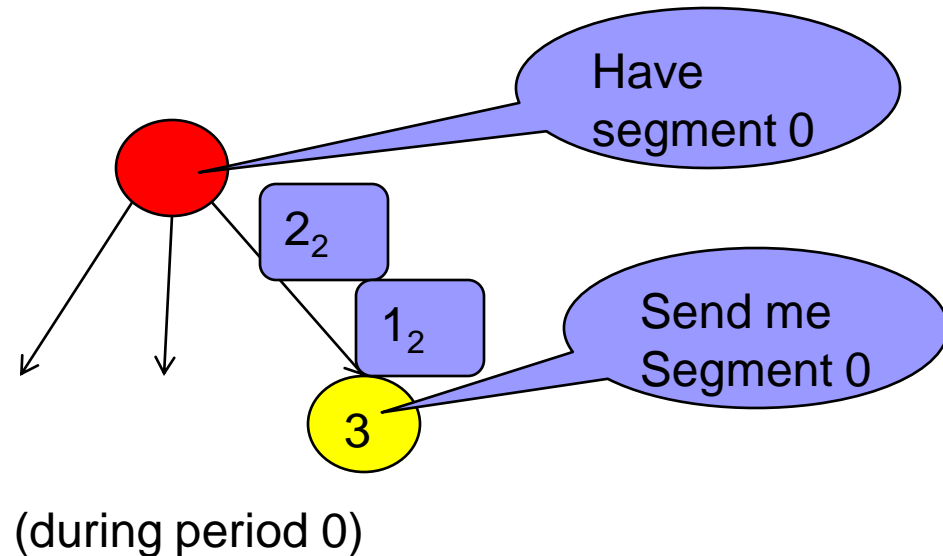


Diffusion Phase

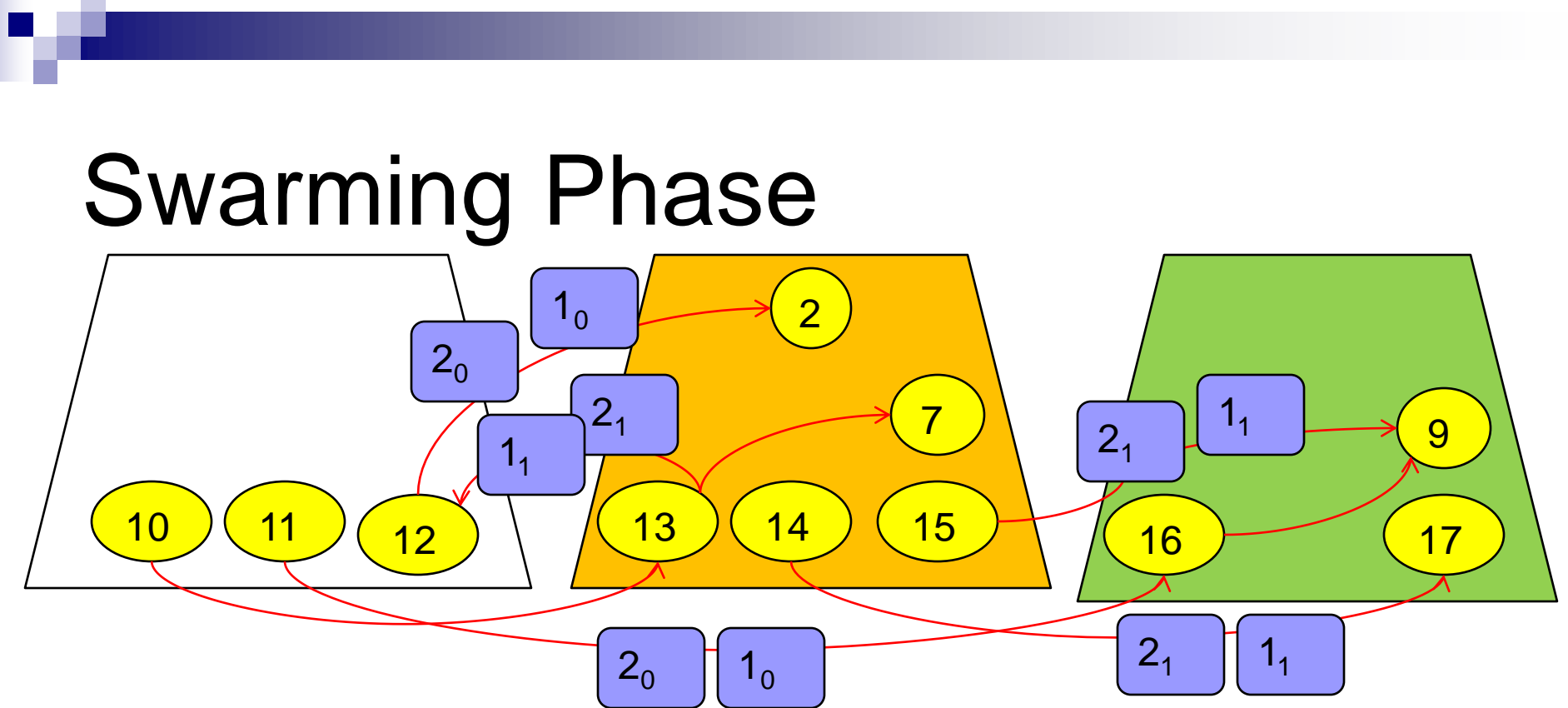


■ As a **new segment** (set of packets) of length L becomes available at source every L seconds

- Level 1 nodes **pull data units** from source, then level 2 pulls from level 1, etc.
- Recall that reporting and pulling are performed periodically



(drawings follow previous example)



- # Swarming Phase
-
- The diagram illustrates the Swarming Phase with three segments: a white segment on the left, an orange segment in the middle, and a green segment on the right. Each segment contains yellow circular nodes. Above the segments are blue rectangular boxes representing data units, labeled with subscripts (e.g., 1_0 , 2_0 , 1_1 , 2_1). Red arrows show the flow of data units from higher levels (e.g., 1_0 to node 2, 2_0 to node 12) and between segments (e.g., from node 15 to node 16, from node 13 to node 10). The nodes are numbered 2, 7, 9, 10, 11, 12, 13, 14, 15, 16, and 17.
- At the end of the diffusion all nodes have at least one data unit of the segment
 - **Pull missing data units** from (swarm-parent) peers located at same or lower level
 - Can node 9 pull new data units from node 16?
 - Node 9 cannot pull data in a single **swarm interval**

Buffering

- Due to diffusion nodes need a buffer
 - Size: $c * L$
 - (diffusion tree dept + minimum number of swarming intervals) $\leq c$

Receiver driven packet scheduling

- Peers maintain parents'

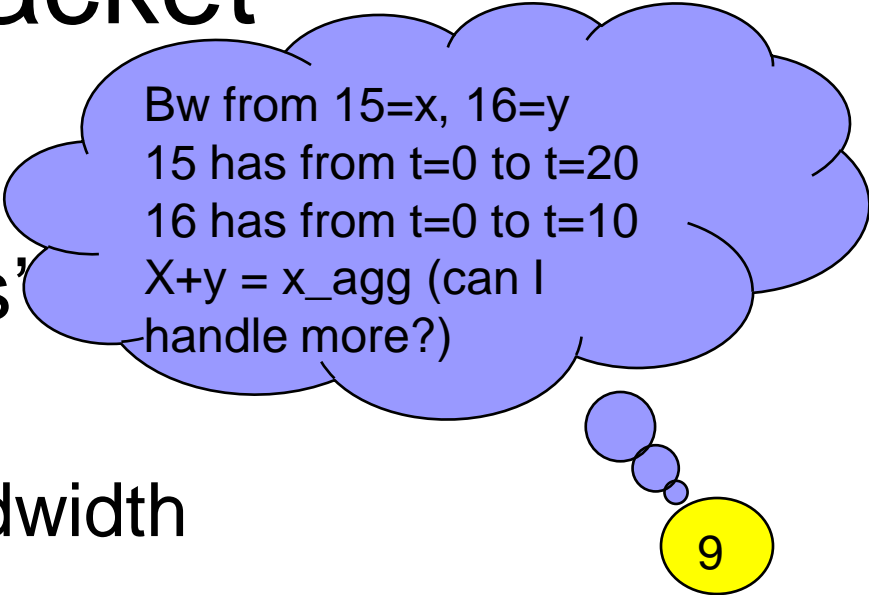
- Available packets

- Weighted average bandwidth

- Peers monitor aggregate bandwidth from all parents

- Requested descriptions adapt to this measure

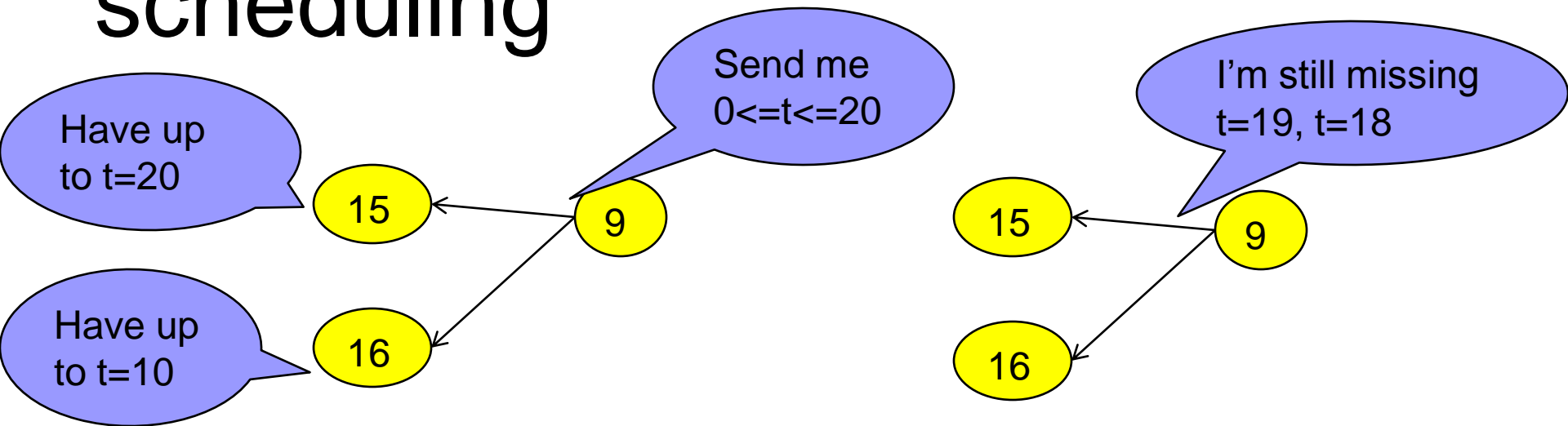
- Designed to maximize utilization of available bandwidth from parents



Bw from 15=x, 16=y
15 has from t=0 to t=20
16 has from t=0 to t=10
 $X+y = x_agg$ (can I handle more?)

9

Receiver driven packet scheduling



- Scheduler identifies packets with highest timestamp available at parents (during last L seconds)
 - Then request all these packets
- Identify missing packets for each timestamp
 - Then request a random subset of these missing packets from all parents (even if they don't have it) **to fully utilize their bandwidth**

Many more details in references and source code

- M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment," SOSP 2003.
- H. Deshpande, M. Bawa, H. Garcia-Molina. "Streaming Live Media over Peers," Technical Report, Stanford InfoLab, 2002.
- N. Magharei, R. Rejaie. "PRIME: Peer-to-Peer Receiver driven MESH-Based Streaming," INFOCOM 2007.
- N. Magharei, R. Rejaie, Y. Guo. "Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches," INFOCOM 2007.
- <http://freepastry.org>