P2P Media Streaming

Amir H. Payberah (amir@sics.se)

Introduction

Media Streaming

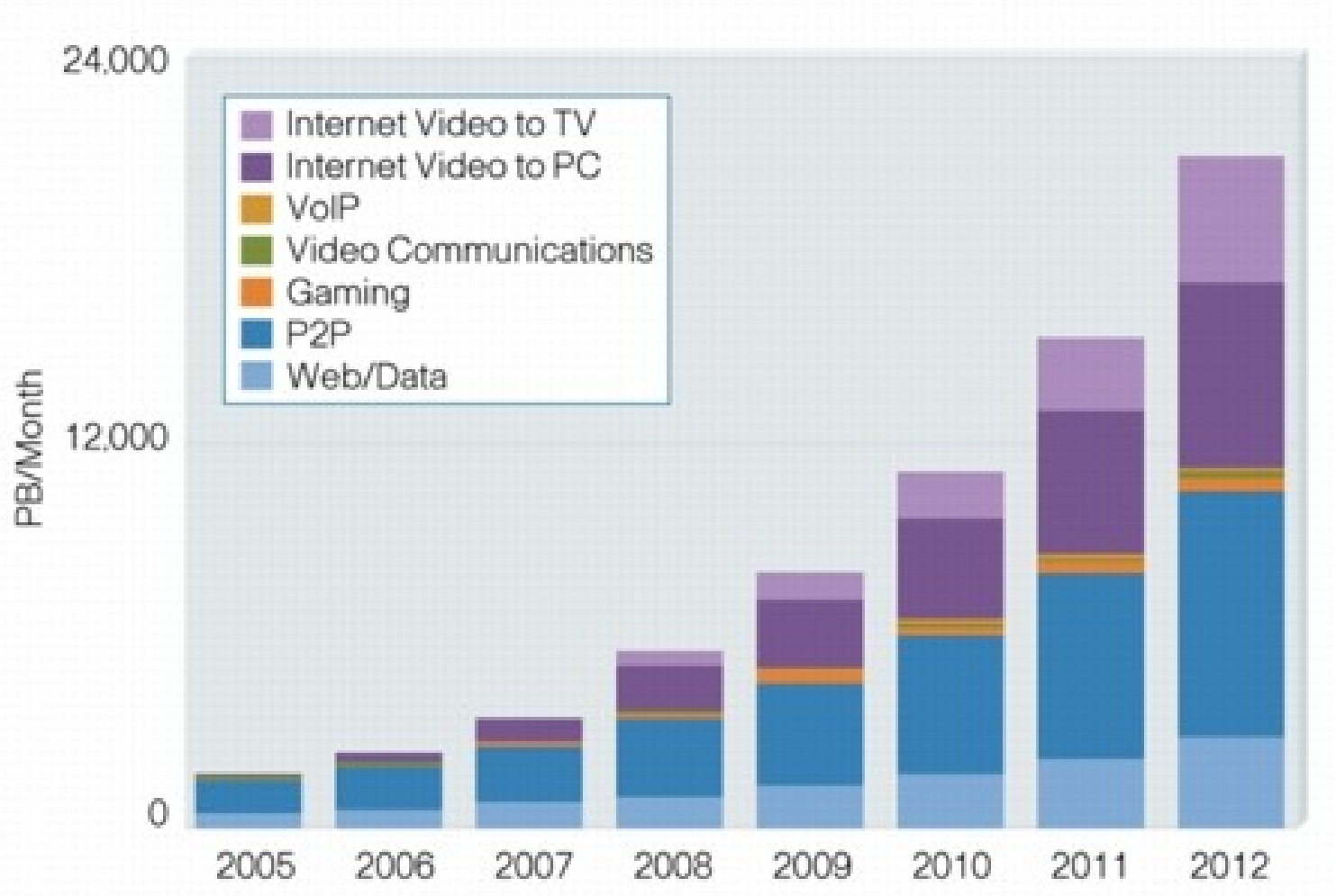
- Media streaming is a multimedia that is sent over a network and played as it is being received by end users.
- Users do not need to wait to download all the media.
- They can play it while the media is delivered by the provider.

Media Streaming

- Live Media Streaming
  - The streams are only available at one particular time.

- Video on Demand (VoD)
  - The streams are stored on a server and are available to be transmitted at a user's request.
  - It provides a large subs e.g., pause, fast forward, fast rewind and ...

Media Streaming Trend

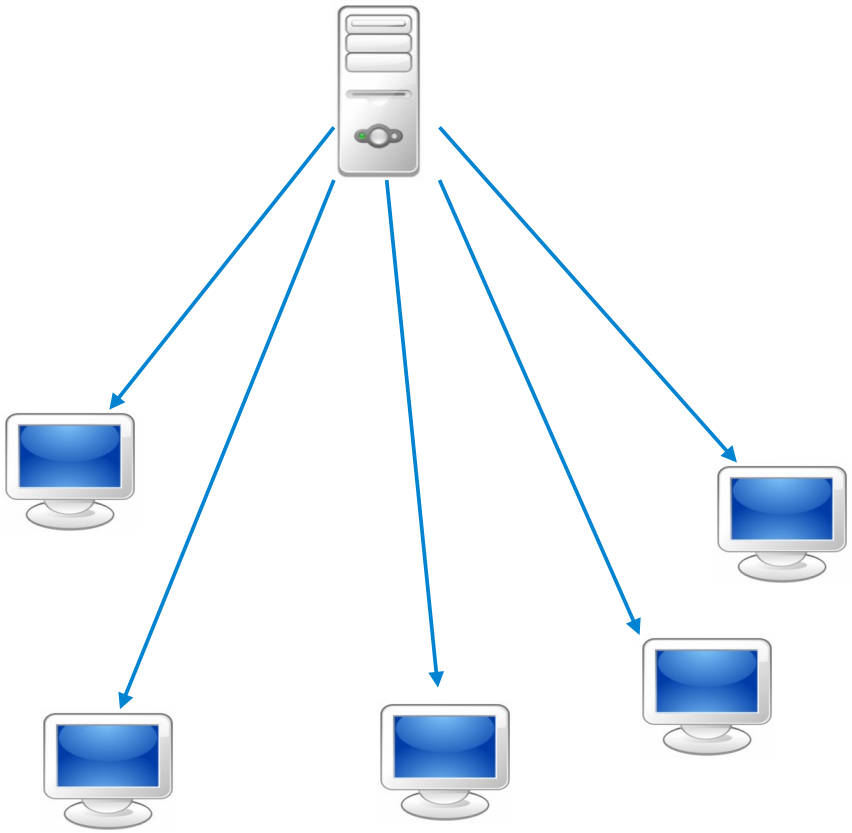


Cisco's global consumer Internet traffic forecast

Solutions for Media Streaming

- Client-Server solution

Client – Server



Client – Server

- What is the problem of Client-Server model? [d]

Client – Server

- What is the problem of Client-Server model?

- Scalability
- Single point of failure

Client – Server

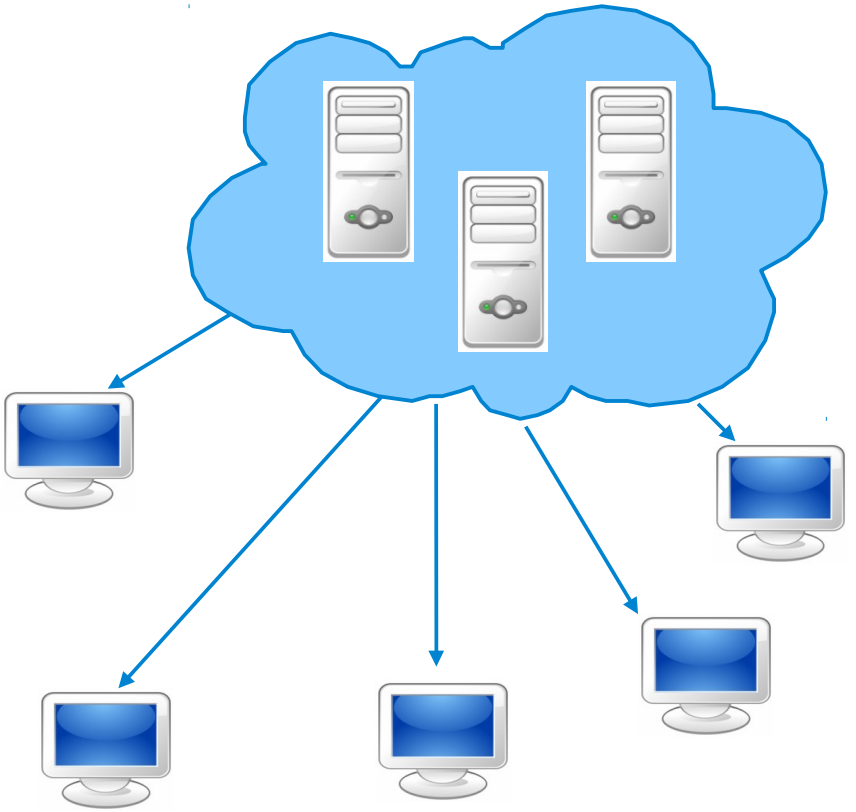
- What is the problem of Client-Server model?

- Scalability
- Single point of failure

- Providing a scalable service, which is resistant to failure is very expensive.

Client – Server

Distributed servers
Content Delivery Network (CDN)

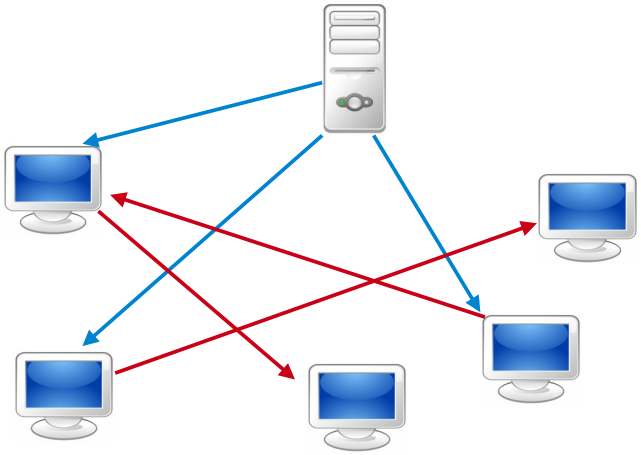


Solutions for Media Streaming

- Client-Server solution
- Peer-to-Peer solution

Peer-to-Peer

- The peers can help each other.
- The peers who have parts of the data can forward it to other requesting peers.
- The capacity increases with the number of peers.



P2P Media Streaming

QoS in P2P Media Streaming Systems

- A negligible startup delay
- High playback continuity: Smooth playback
- Short playback latency (only for Live Streaming)

P2P Media Streaming Challenges

- How to build and maintain the streaming overlay.
- Nodes join, leave and fail continuously (churn).
- Free-riding problem.
- Network capacity changes.
- Connectivity Problem (NAT).
- Security

P2P Media Streaming Challenges

- How to build and maintain the streaming overlay.
- Nodes join, leave and fail continuously (churn).
- Free-riding problem.
- Network capacity changes.
- Connectivity Problem (NAT).
- Security

Main Questions in Designing a P2P Streaming System

- What overlay topology is built for data dissemination?
- What algorithm is used for data dissemination?
- How to construct and maintain this overlay?

Main Questions in Designing a P2P Streaming System

- What overlay topology is built for data dissemination?
- What algorithm is used for data dissemination?
- How to construct and maintain this overlay?

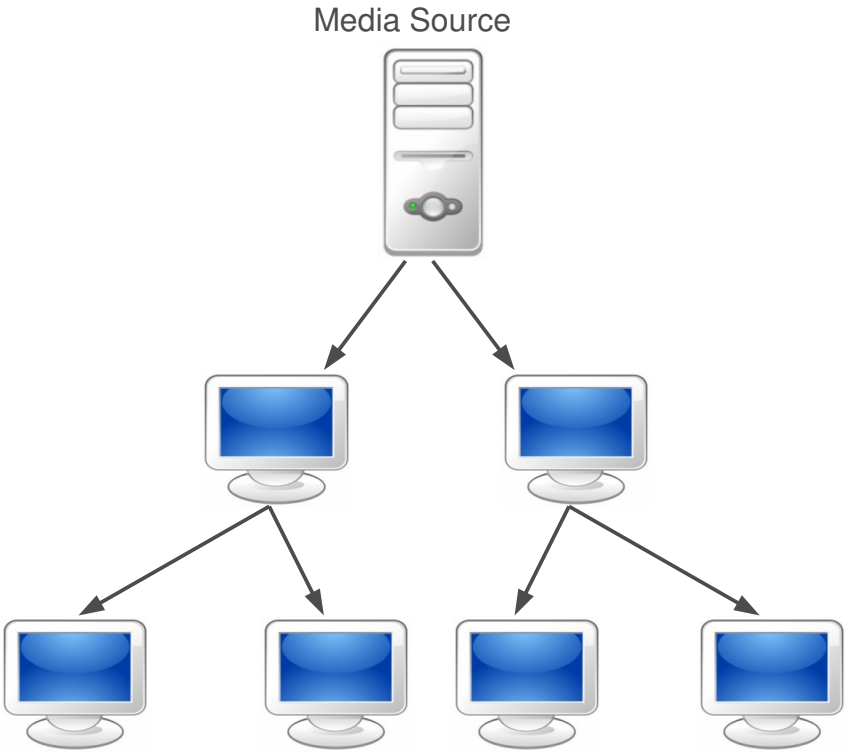
Data Dissemination Overlay

- What overlay topology is built to distribute data messages.

- It could be:
  - Single tree
  - Multiple tree
  - Mesh

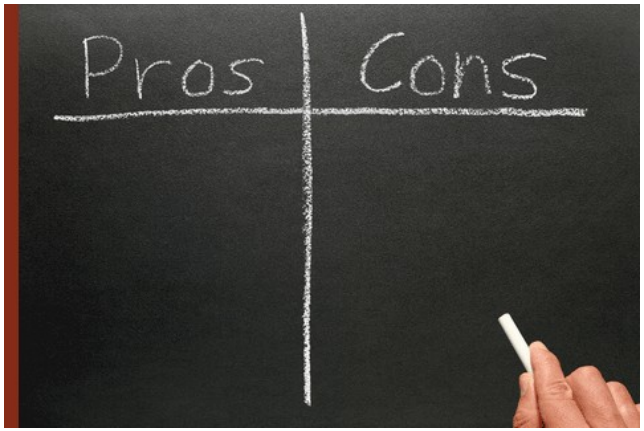
Single Tree Structure

- Build a single multicast tree, in which the root is the media source and the interior nodes and leaves are peers.



Single Tree Advantage/Disadvantage?

- Advantage/Disadvantage [d]



Single Tree Advantage/Disadvantage?

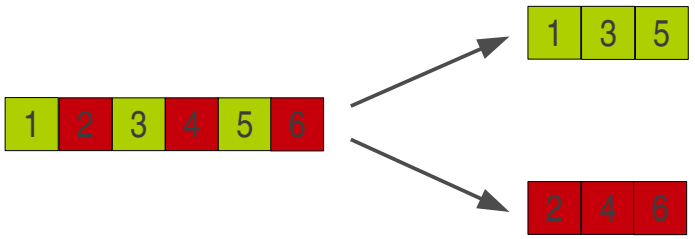
- Advantage/Disadvantage

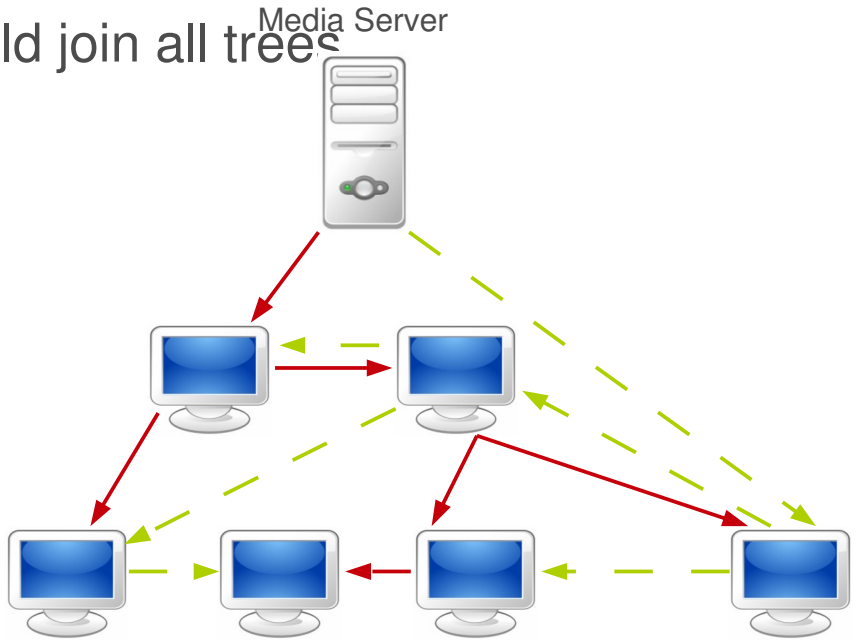
- Advantage
  - The short latency of data delivery
  - Easy to implement

- Disadvantage
  - The fragility of the tree structure upon the failure of nodes close to the root
  - All the traffic is only forwarded by the interior nodes

Multiple-Tree Structure

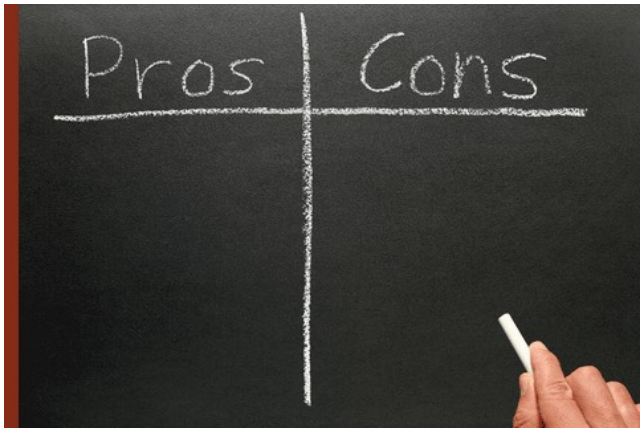
- The media source splits the stream into a set of sub-streams.
- A single tree is created for each sub-stream.
- A peer to receive the whole media should join all trees





Multiple-Tree Advantage/Disadvantage?

- Advantage/Disadvantage [d]



Multiple-Tree Advantage/Disadvantage?

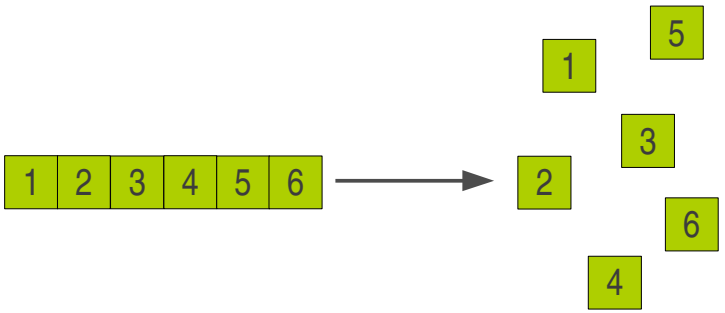
- Advantage/Disadvantage

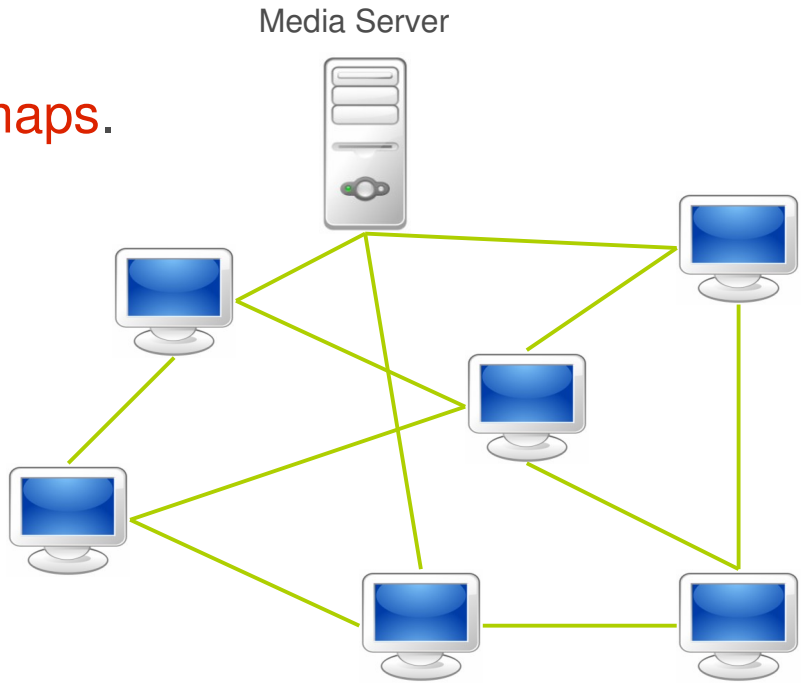
- Advantage
  - Resilient to node failure
  - Good load balancing

- Disadvantage
  - Difficult to implement
  - If a node fails, the sub-tree rooted at that node does not receive data, while they rejoin the system again

Mesh-based Structure

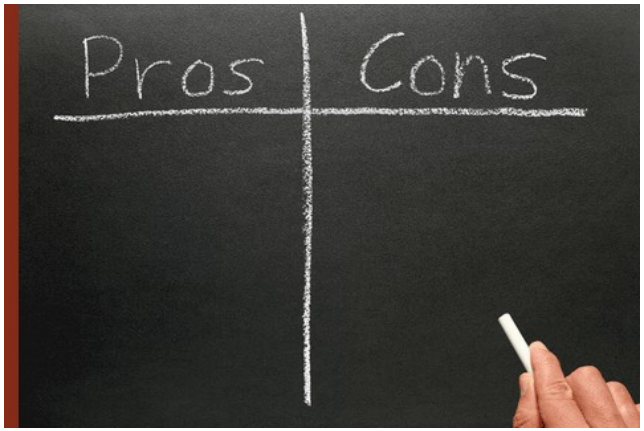
- The media source into small blocks.
- Nodes are connected in a mesh-network.
- Nodes periodically exchange their buffer maps.





Mesh Advantage/Disadvantage?

- Advantage/Disadvantage [d]



Mesh Advantage/Disadvantage?

- Advantage/Disadvantage

- Advantage
  - Resilient to node failure
  - Good load balancing
  - Easy to implement

- Disadvantage
  - Unpredictable latencies due to the frequent exchange of notifications and requests

Main Questions in Designing a P2P Streaming System

- What overlay topology is built for data dissemination?
- What algorithm is used for data dissemination?
- How to construct and maintain this overlay?

Data Dissemination Algorithms

- How to distribute data messages.

- It could be:
  - Push-based
  - Pull-base
  - Push-Pull-based

Push-based Data Dissemination

- A node actively pushes a received block to its neighbours.
- Mostly used in tree-based overlays.
- What about mesh-based overlays? [d]

Push-based Data Dissemination

- A node actively pushes a received block to its neighbours.
- Mostly used in tree-based overlays.
- What about mesh-based overlays?
  - Redundant messages: a node might blindly push a block to a node already has that block.

Pull-based Data Dissemination

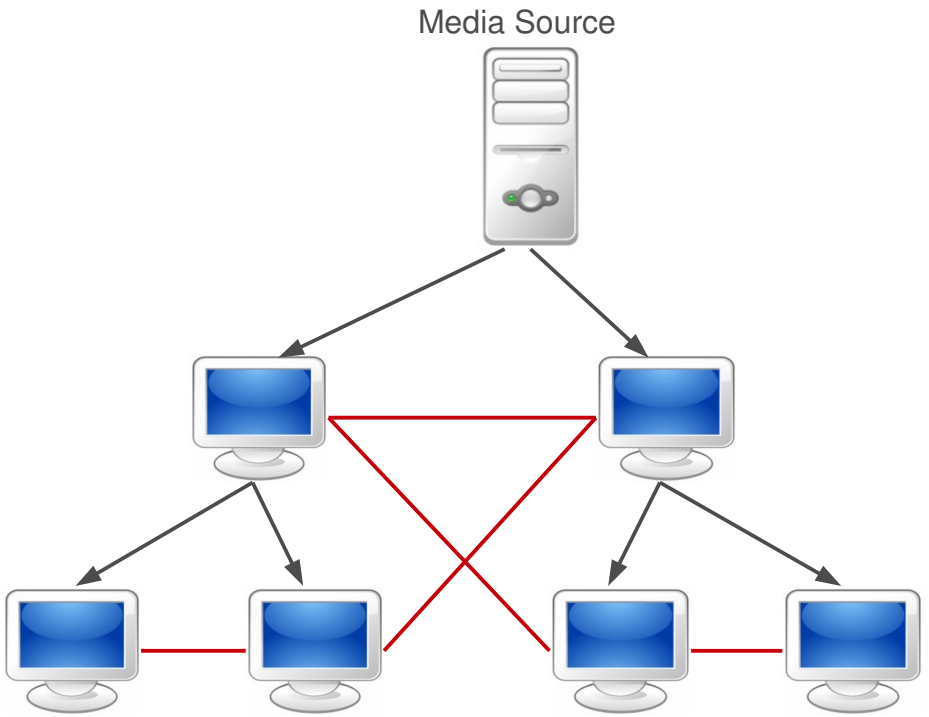
- Nodes periodically exchange data availability (buffer maps).
- After receiving a buffer map, a node can decide and schedule to pull which block from which node.
- Mostly used in mesh-based overlays.

Pull-based Data Dissemination

- Nodes periodically exchange data availability (buffer maps).
- After receiving a buffer map, a node can decide and schedule to pull which block from which node.
- Mostly used in mesh-based overlays.

Push-Pull-based Data Dissemination

- Usually blocks are pushed through a tree and missed blocks are pulled from the mesh neighbours.



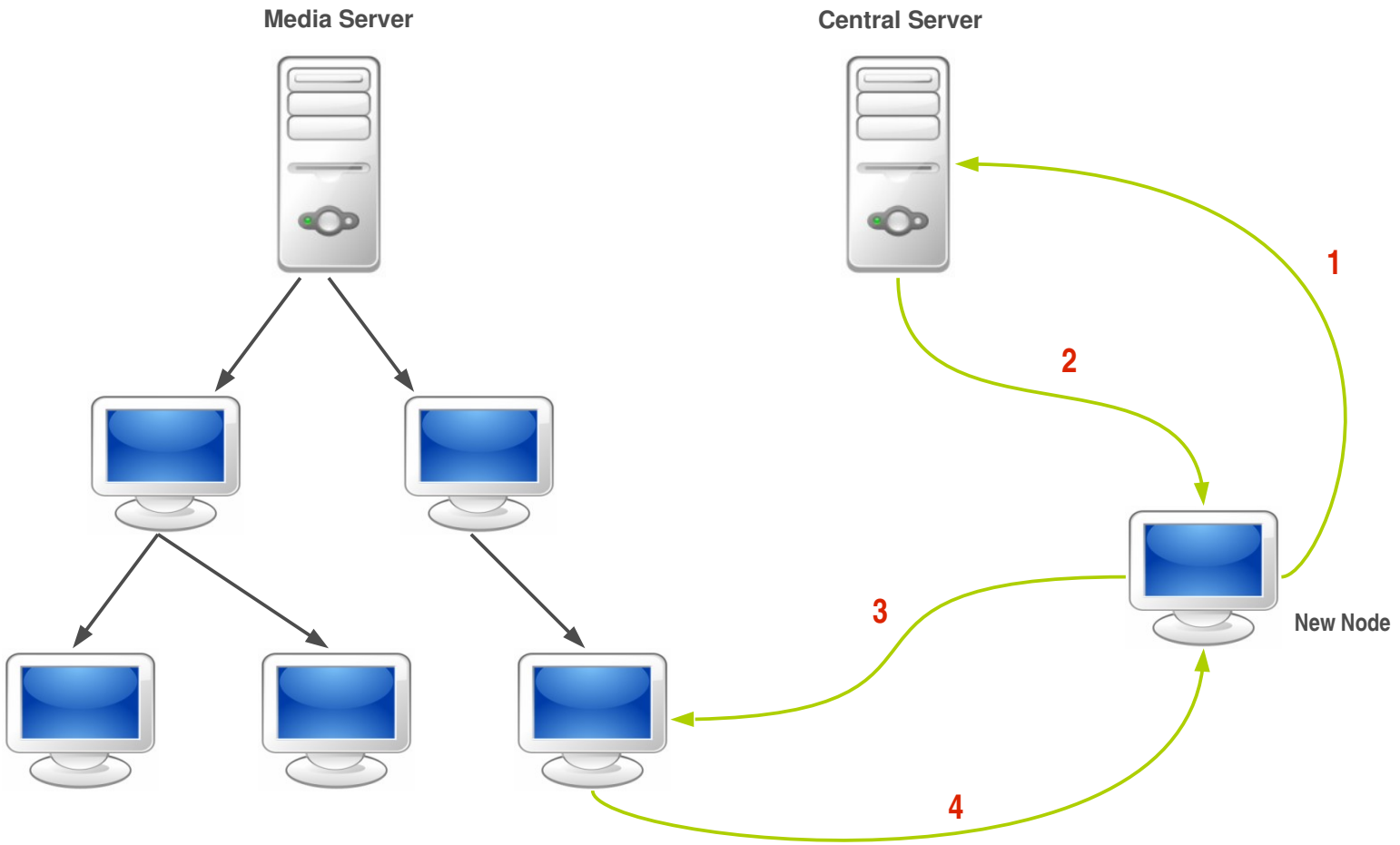
Main Questions in Designing a P2P Streaming System

- What overlay topology is built for data dissemination?
- What algorithm is used for data dissemination?
- How to construct and maintain this overlay?

The Overlay Construction and Maintenance

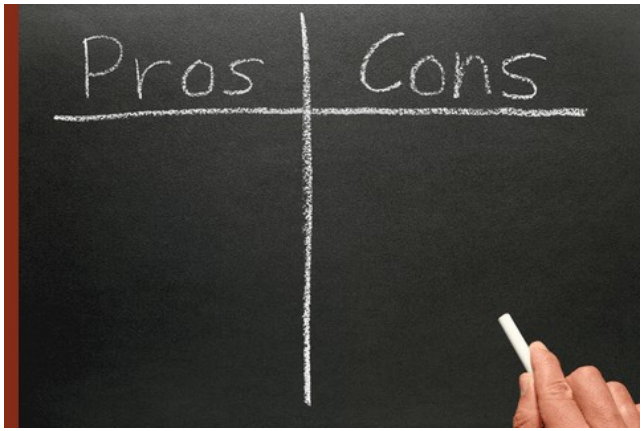
- How to build and maintain the data distribution overlay.
- Using the control messages for this purpose.
- It could be:
  - Centralized
  - Hierarchical
  - DHT-based
  - Control flooding
  - Gossip-based

Centralized Method



Centralized Advantage/Disadvantage?

- Advantage/Disadvantage [d]



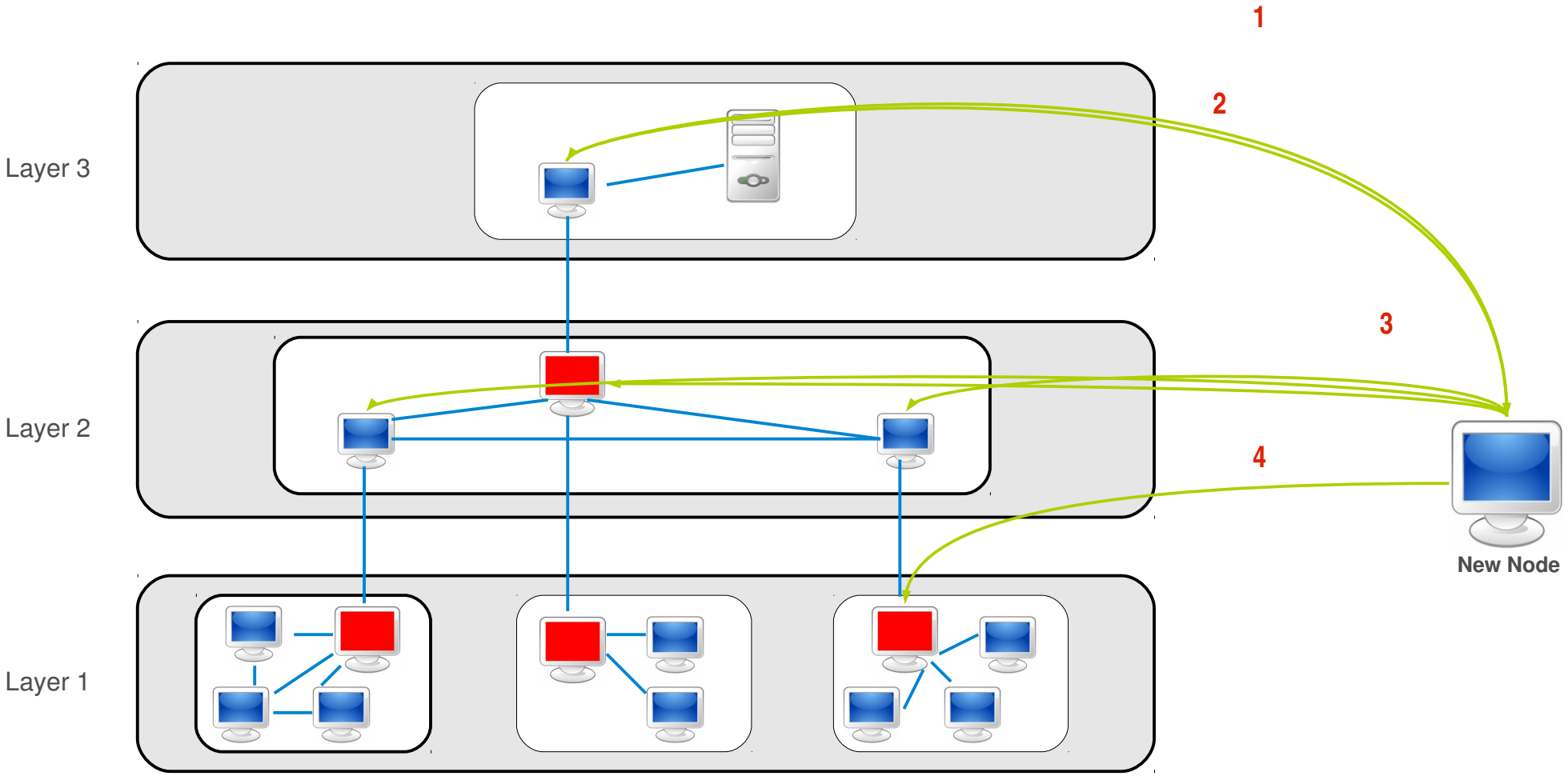
Centralized Advantage/Disadvantage?

- Advantage/Disadvantage

- Advantage
  - Fast
  - Easy to apply optimization methods
  - Easy to implement

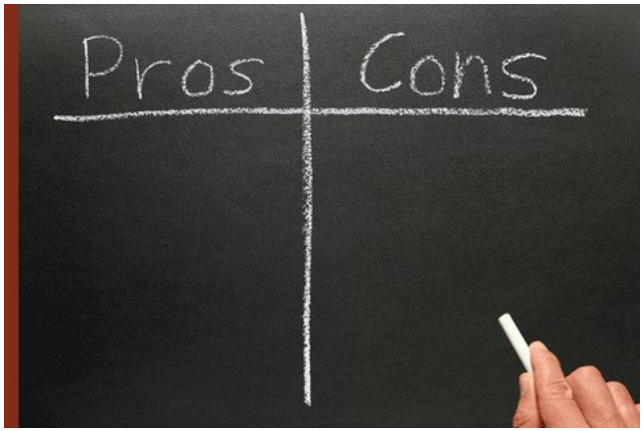
- Disadvantage
  - Not scalable
  - Single point of failure

Hierarchical Method



Hierarchical Advantage/Disadvantage?

- Advantage/Disadvantage [d]



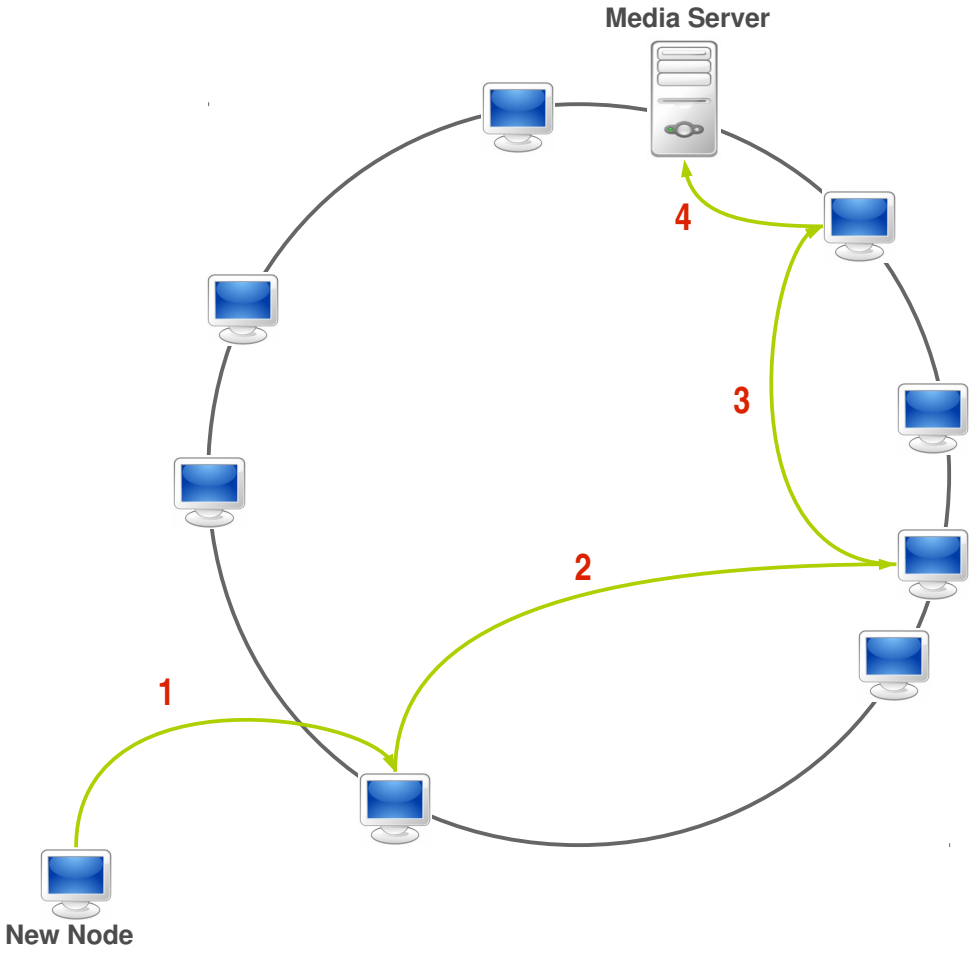
Hierarchical Advantage/Disadvantage?

- Advantage/Disadvantage

- Advantage
  - Scalable
  - No single point of failure

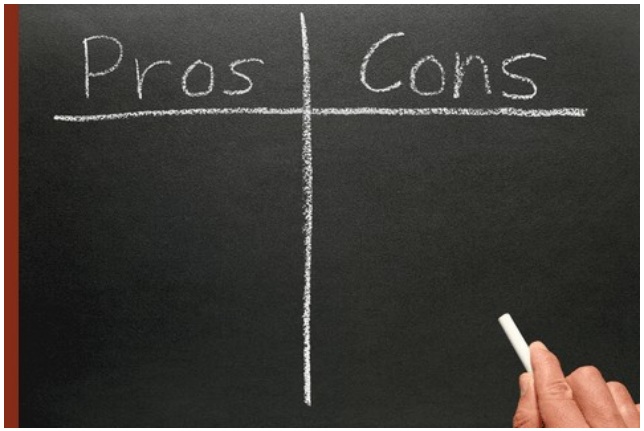
- Disadvantage
  - Slow convergence
  - Difficult to implement

DHT-based Method



DHT-based Advantage/Disadvantage?

- Advantage/Disadvantage [d]



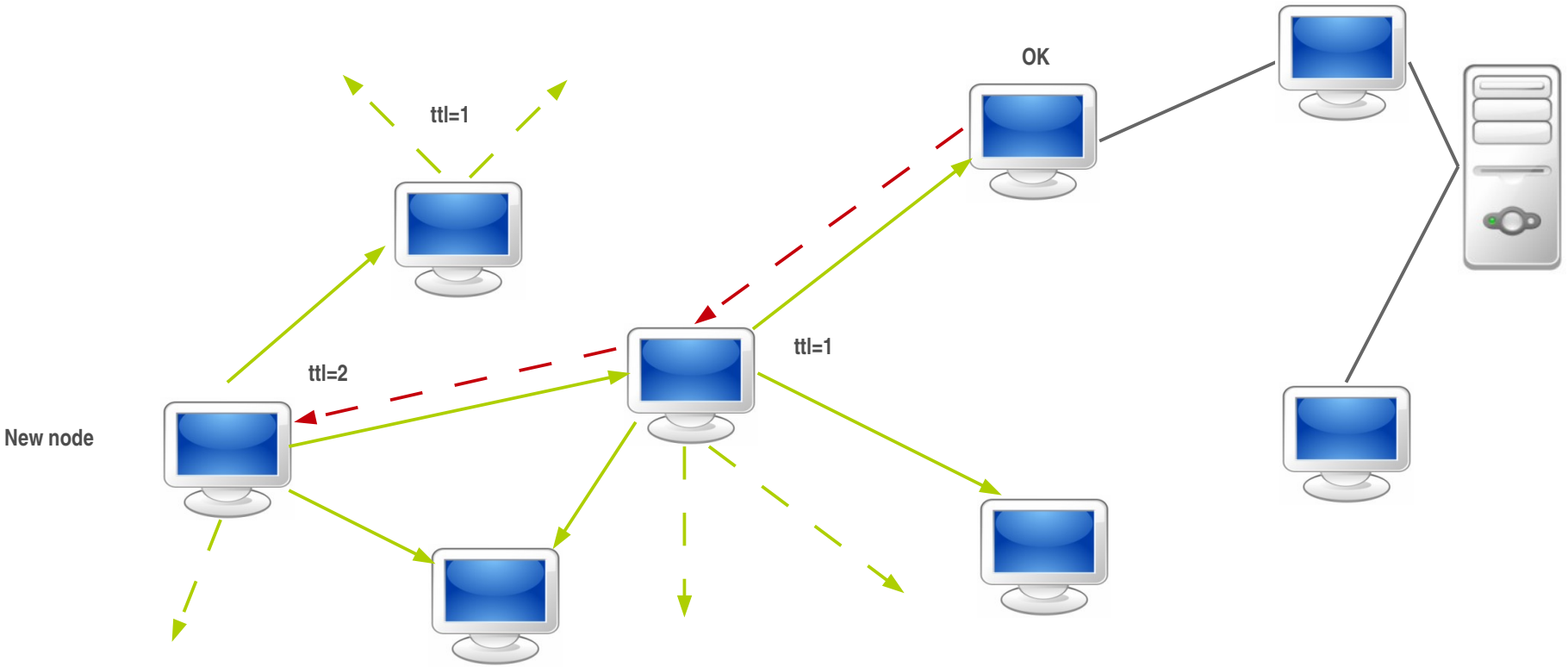
DHT-based Advantage/Disadvantage?

- Advantage/Disadvantage

- Advantage
  - Scalable
  - No single point of failure

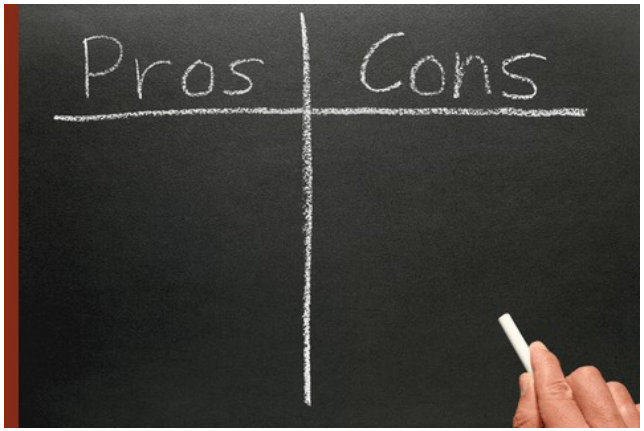
- Disadvantage
  - Maintaining DHT

Controlled Flooding Method



Flooding Advantage/Disadvantage?

- Advantage/Disadvantage [d]



Flooding Advantage/Disadvantage?

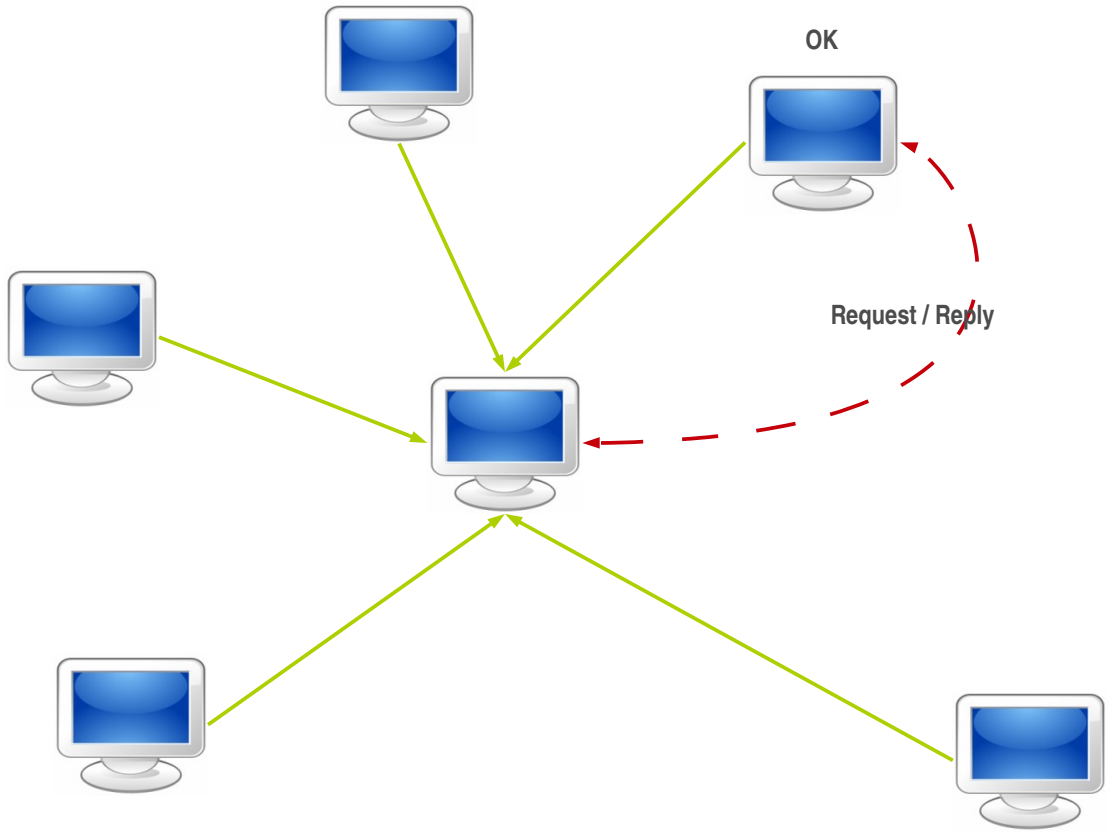
- Advantage/Disadvantage

- Advantage
  - Scalable
  - No single point of failure

- Disadvantage
  - No guarantee to find supplier node
  - Slow convergence

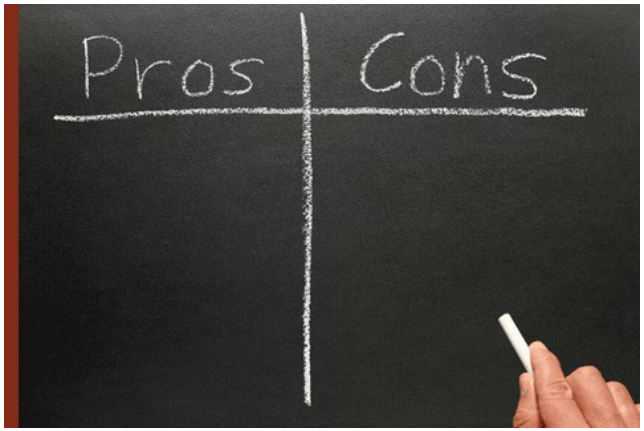
Gossip-based Method

- Peers periodically send their data availability to their neighbours.



Gossip-based Advantage/Disadvantage?

- Advantage/Disadvantage [d]



Gossip-based Advantage/Disadvantage?

- Advantage/Disadvantage

- Advantage
  - Scalable
  - No single point of failure
  - Easy to implement

- Disadvantage
  - No guarantee to find supplier node in time

Classification of P2P Streaming Solutions

Related Work

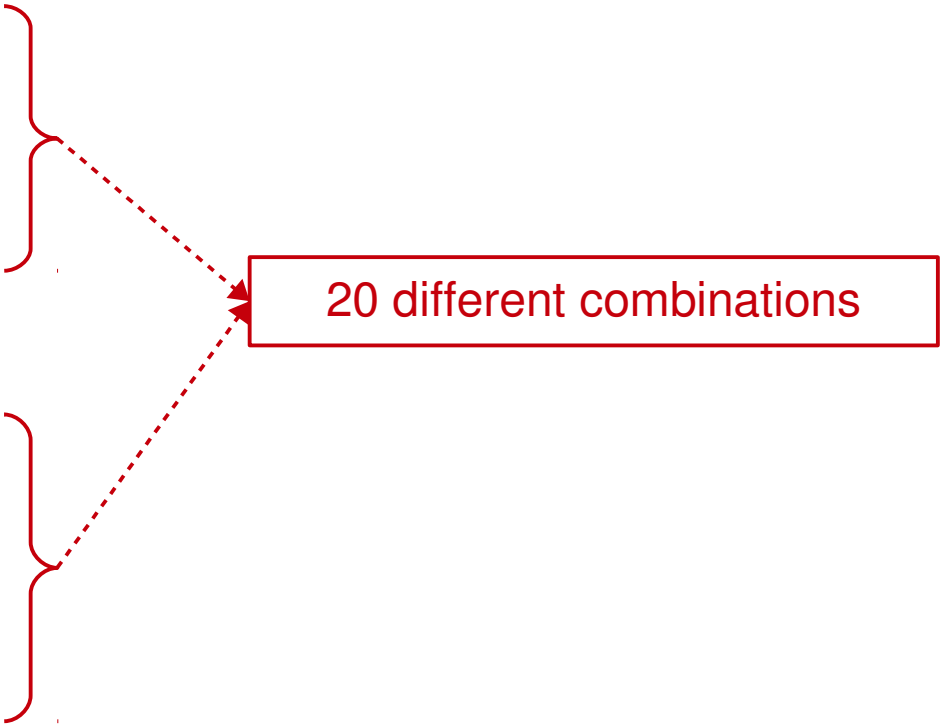
- SplitStream
- DONet/Coolsteraming
- CoopNet
- Orchard
- Bullet
- Prime
- Pulsar
- NICE
- Zigzag
- DirectStream
- MeshCast

- mtreeBone
- PULSE
- GnuStream
- SAAR
- ChainSaw
- ChunkySpread
- BulkTree
- ForestCast
- AnySee
- DagStream
- Climber

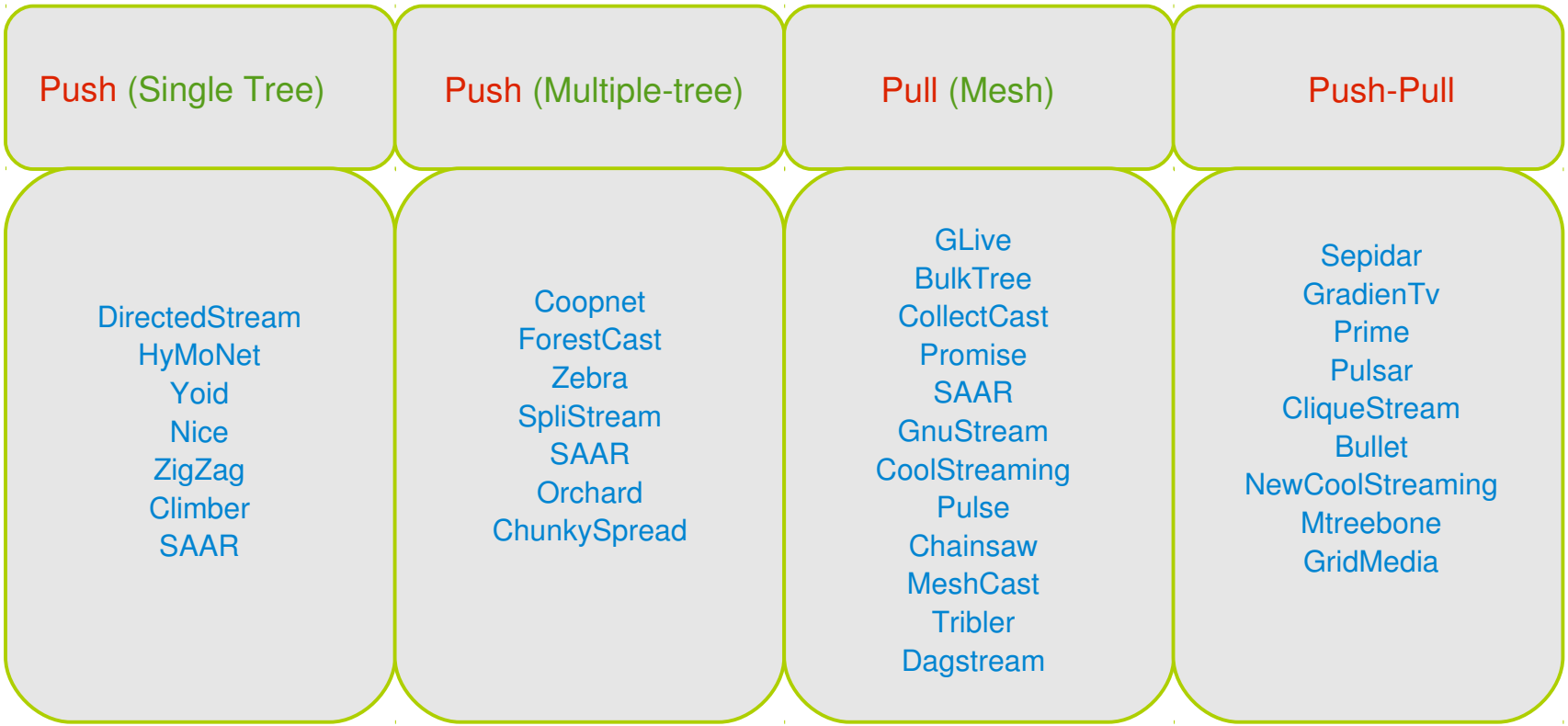
- CollectCast
- HyMoNet
- GridMedia
- Promise
- Yoid
- Zebra
- Tribler
- CliqueStream
- GradienTv
- Sepidar
- GLive

Data Dissemination Overlay

- Data dissemination:
  - Push – Single-tree
  - Push – Multiple-tree
  - Pull – Mesh
  - Push-Pull
- Overlay maintenance:
  - Centralized
  - Hierarchical
  - DHT-based
  - Control flooding
  - Gossip-based



Data Dissemination Overlay



Overlay Construction and Maintenance Methods

Centralized	DirectedStream, HyMoNet, Yoid, CoopNet ForestCast, Zebra, Prime
Hierarchical	NICE, ZigZag, Climber, BulkTree, Prime
DHT-based	SAAR, SplitStream, CollectCast, Promise, CliqueStream, Pulsar
Flooding	GnuStream
Gossip-based	GLive, Sepidar, GradienTv, Orchard, ChunkySpread, CoolStreaming, Pulse, Chainsaw MeshCast, Tribler, DagStream, Bullet, mTreebone, GridMedia

All Together

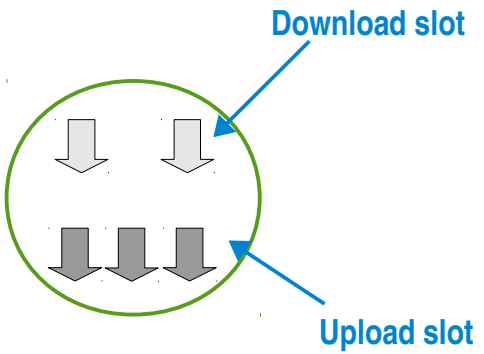
	Push (Single tree)	Push (Multiple-tree)	Pull (Mesh)	Push-Pull
Centralized	DirectedStream HyMoNet Yoid	Coopnet ForestCast Zebra		Prime
Hierarchical	NICE ZigZag Climber		BulkTree	Prime
DHT-based	SAAR	SAAR SplitStream	SAAR CollectCast Promise	Pulsar CliqueStream
Flooding			GnuStream	
Gossip-based		Orchard ChunkySpread	Glive - CoolStreaming – Pulse - Chainsaw – MeshCast - Tribler - DagStream	Sepidar - GradienTv Bullet - mTreebone GridMedia

Sepidar

Problem Description

Problem Description (1/5)

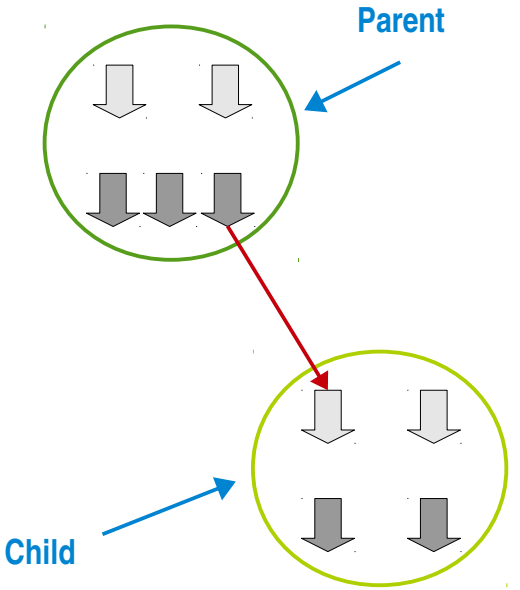
- Building and optimizing a P2P overlay for live media streaming
  - minimize playback latency
  - improve timely delivery of the stream
- The media stream is split into a number of sub-streams or stripes.
- A node can create a bounded number of download connections, and accept a bounded number of upload connections.



Problem Description (2/5)

- In order to provide the full media to all the nodes
  - every download-slot needs to be assigned to an upload-slot.
  - download-slots at a node must download different stripes.

- This problem can be defined as an assignment problem.



Problem Description (3/5)

- A connection between a download-slot i and an upload-slot j for a stripe k is associated with a cost cijk, which is the number of hops from the owner of the upload-slots j, to the media source for the stripe k.

- A complete assignment, A, is an assignment that each download-slot is assigned to an upload-slot.

Problem Description (4/5)

- Formulating as an optimization problem:

- Objective function
  - We want to find a complete assignment over all the complete assignments that minimizes the total cost:

$$\sum_{(i,j,k)\in A} c_{ijk}$$

- Subject to
  - Every download-slot is assigned to exactly one upload-slot.
  - Each upload-slot is assigned to at most one download-slot.
  - The download-slots owned by the same node download distinct stripes.

Problem Description (5/5)

- Centralized solution:
  - Needs global knowledge.
  - Possible for small system sizes.
- Distributed market-based approach:
  - Inspired by auction algorithms.
  - Each node knows only a small number of nodes in the system (partial view).

Sepidar Solution

Design Space

- What overlay topology is built for data dissemination?
  - Tree
  - Multiple-tree
  - Mesh
- What algorithm is used for data dissemination?
  - Push
  - Pull
  - Push-Pull
- How to construct and maintain this overlay?
  - Centralized
  - DHT
  - Gossip-based
  - ...

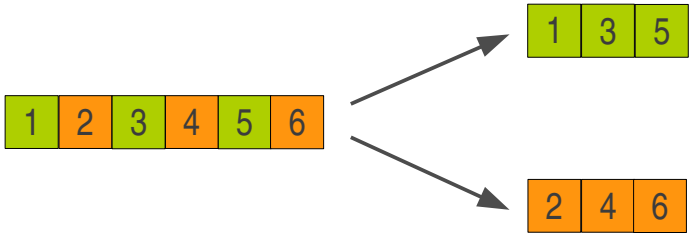
Design Space

- What overlay topology is built for data dissemination?
  - Tree
  - Multiple-tree
  - Mesh
- What algorithm is used for data dissemination?
  - Push
  - Pull
  - Push-Pull
- How to construct and maintain this overlay?
  - Centralized
  - DHT
  - Gossip-based
  - ...

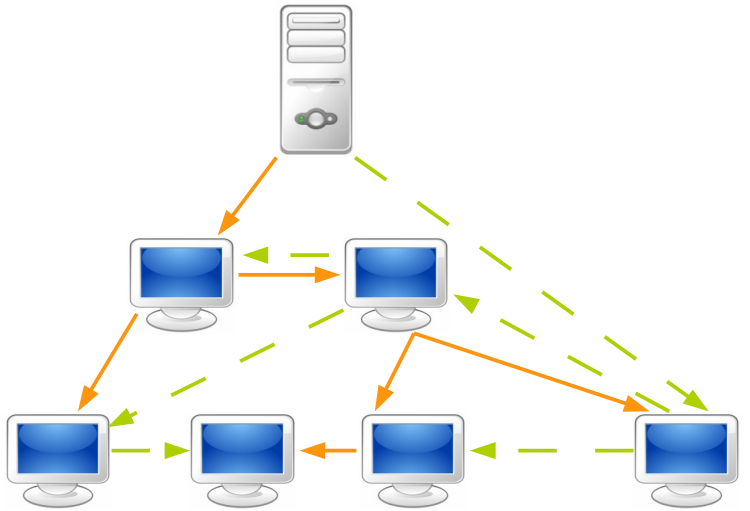
Multiple-Tree Overlay

- Split the main stream into a set of sub-streams, and divides each sub-stream into a number of blocks.

- In case of having 2 stripes:
  - Sub-stream 0: 0, 2, 4, 6, …
  - Sub-stream 1: 1, 3, 5, 7, …

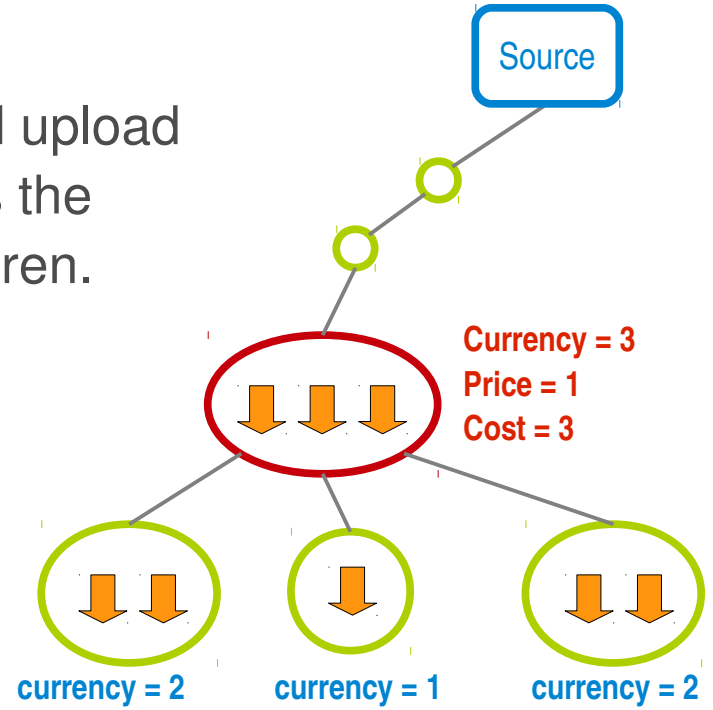


- Construct one tree for each stripe.



Market Model – Node Properties

- Currency: The the number of upload slots at a node.
- Price: The price of a node that has an unused upload slot is zero, otherwise the node's price equals the lowest currency of its already connected children.
- Cost: The length of its path to the root.

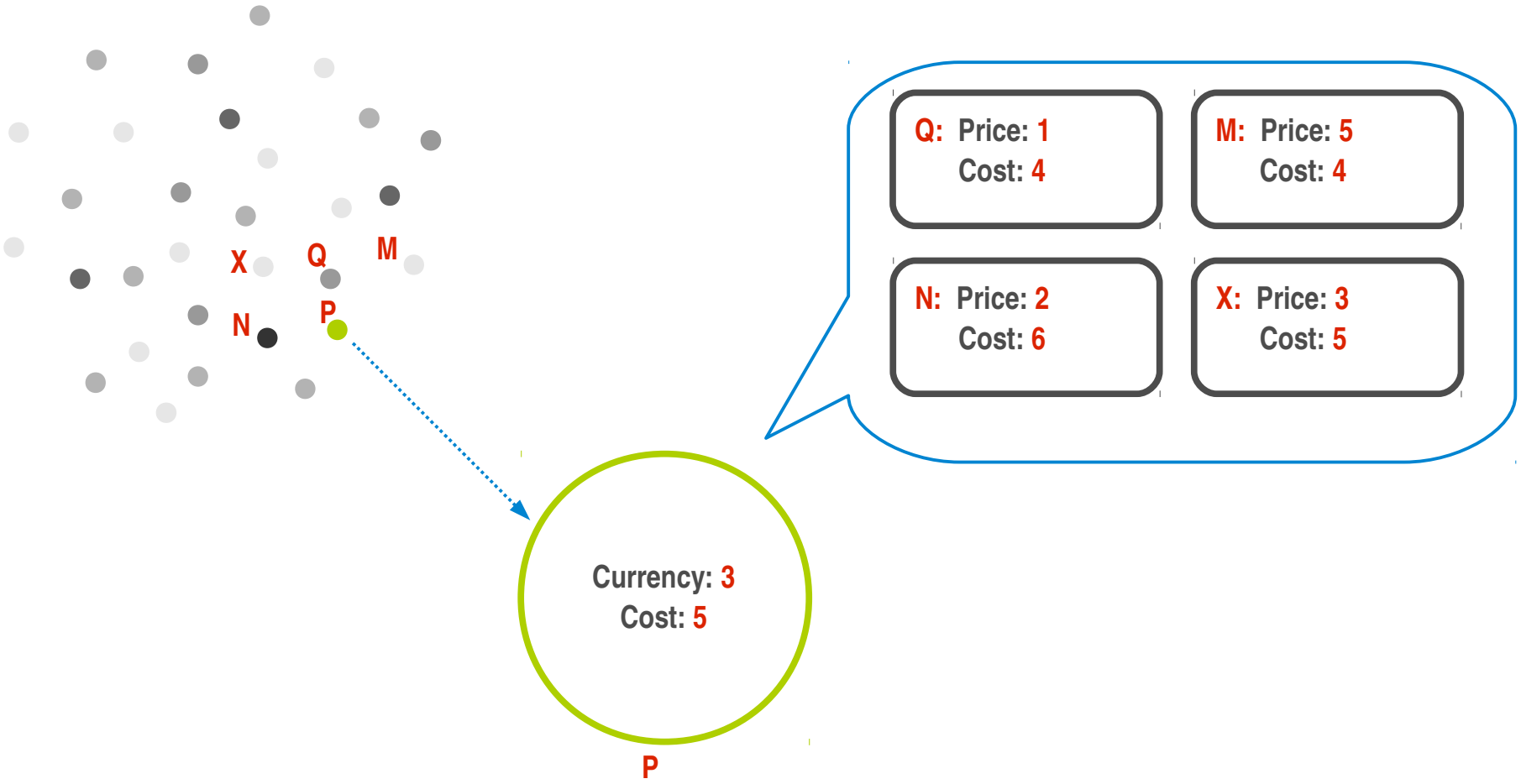


Market Model – Streaming Overlay Construction

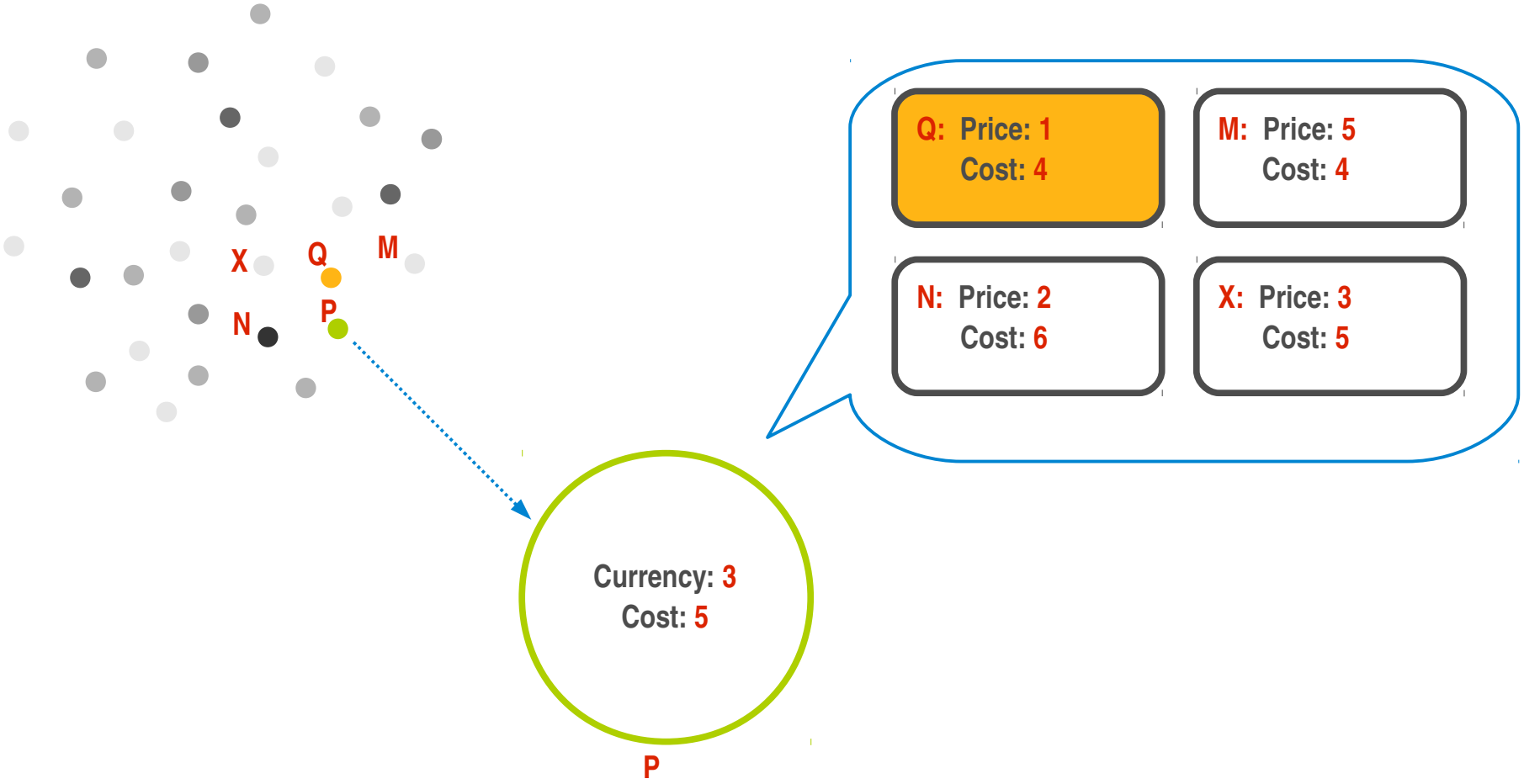
- Our market model is based on minimizing costs through nodes iteratively bidding for upload slots.

- The depth of a node in each tree is inversely proportional to its currency.

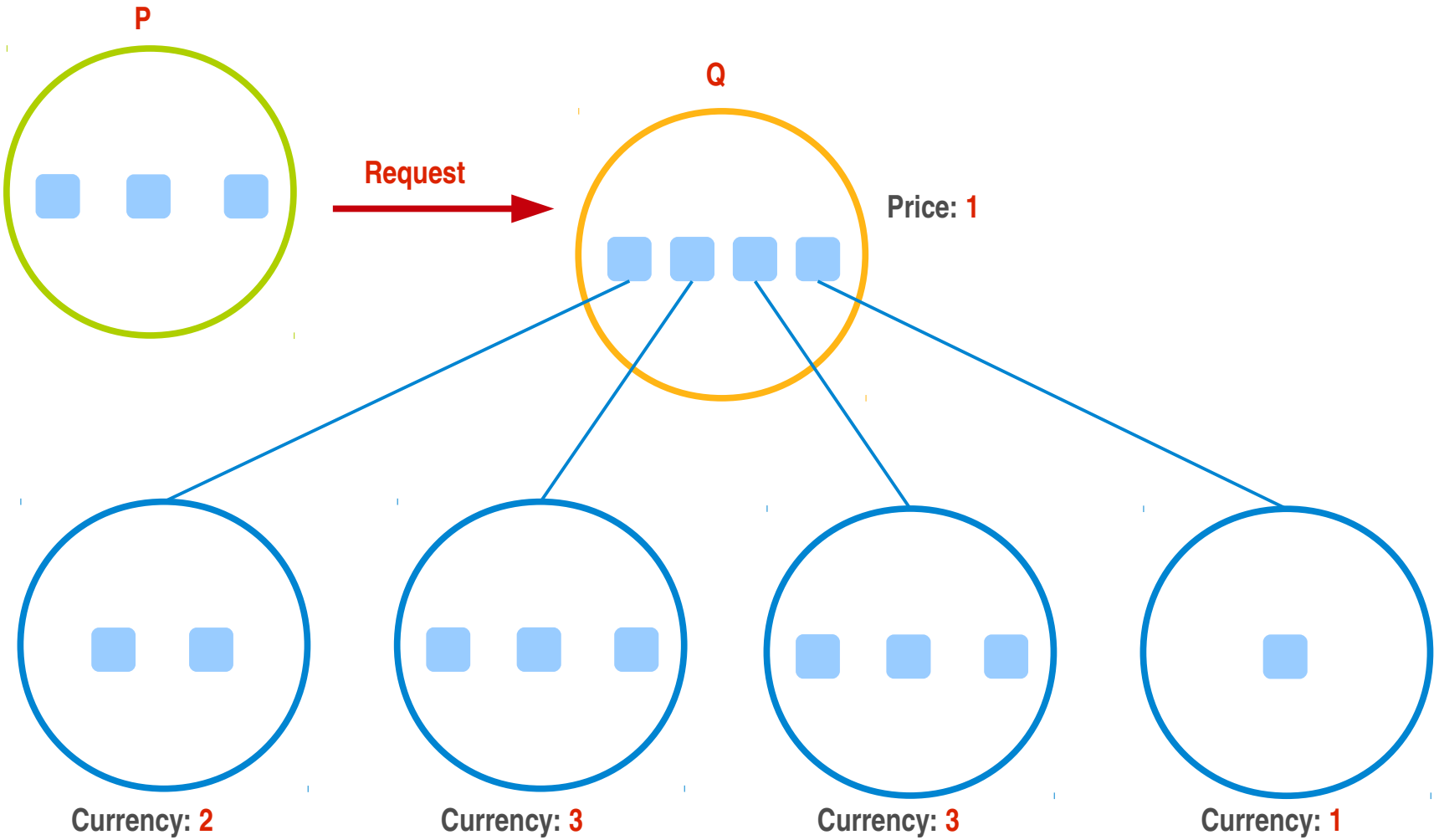
Market Model – Childe Side (1/2)



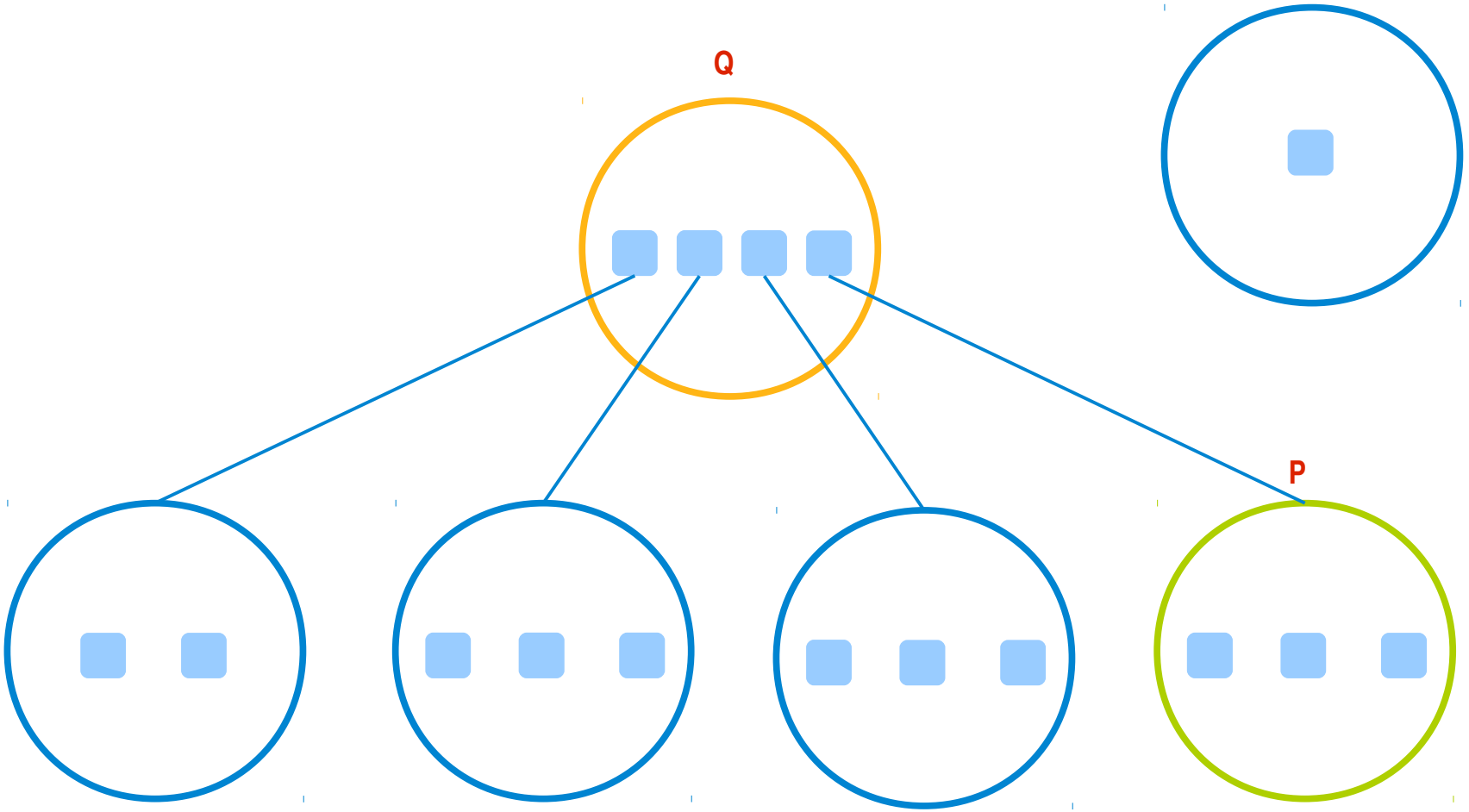
Market Model – Childe Side (2/2)



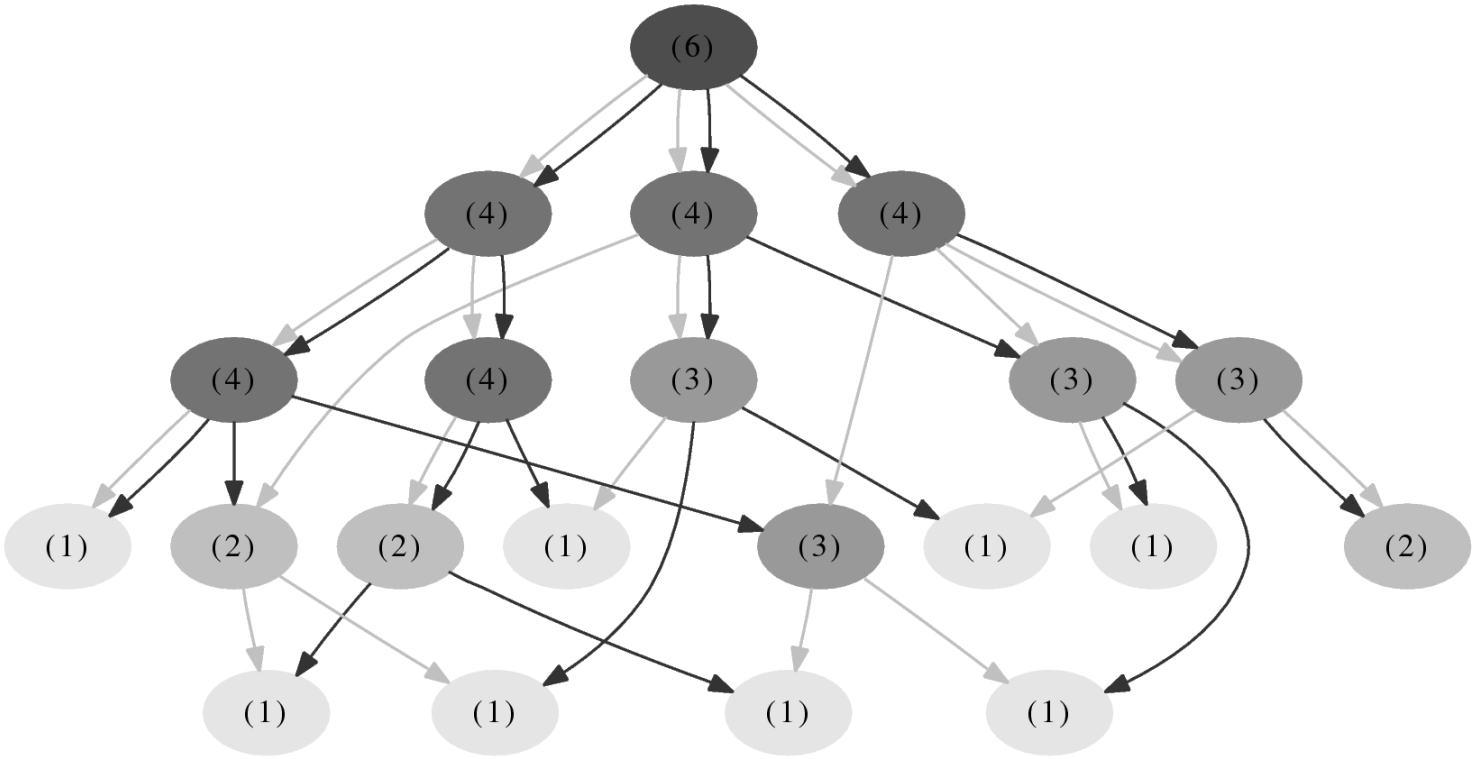
Market Model – Parent Side (1/2)



Market Model – Parent Side (2/2)



Constructed Streaming Overlay

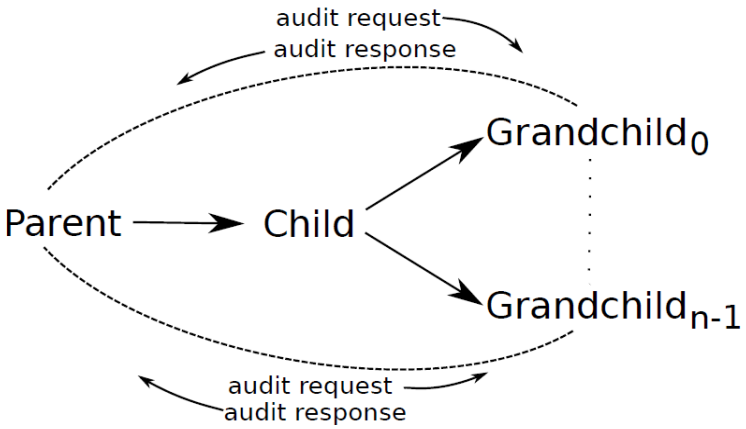


- Constructed 2-tree overlay.
- Darker nodes have more upload capacity than lighter ones.

Freeriders

Freerider Detector

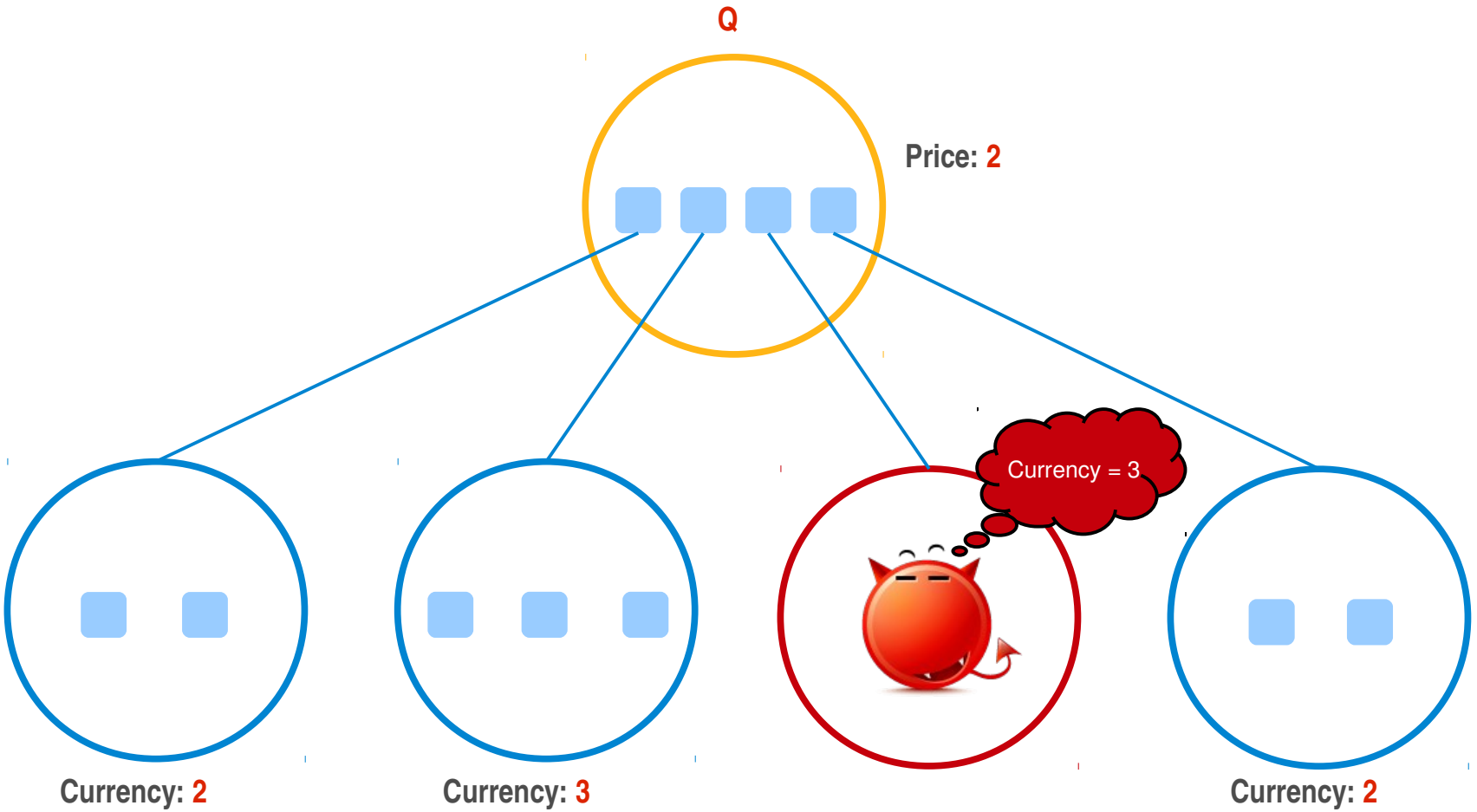
- Freeriders are nodes that supply less upload bandwidth than claimed.
- Nodes identify freeriders through transitive auditing using their children's children.



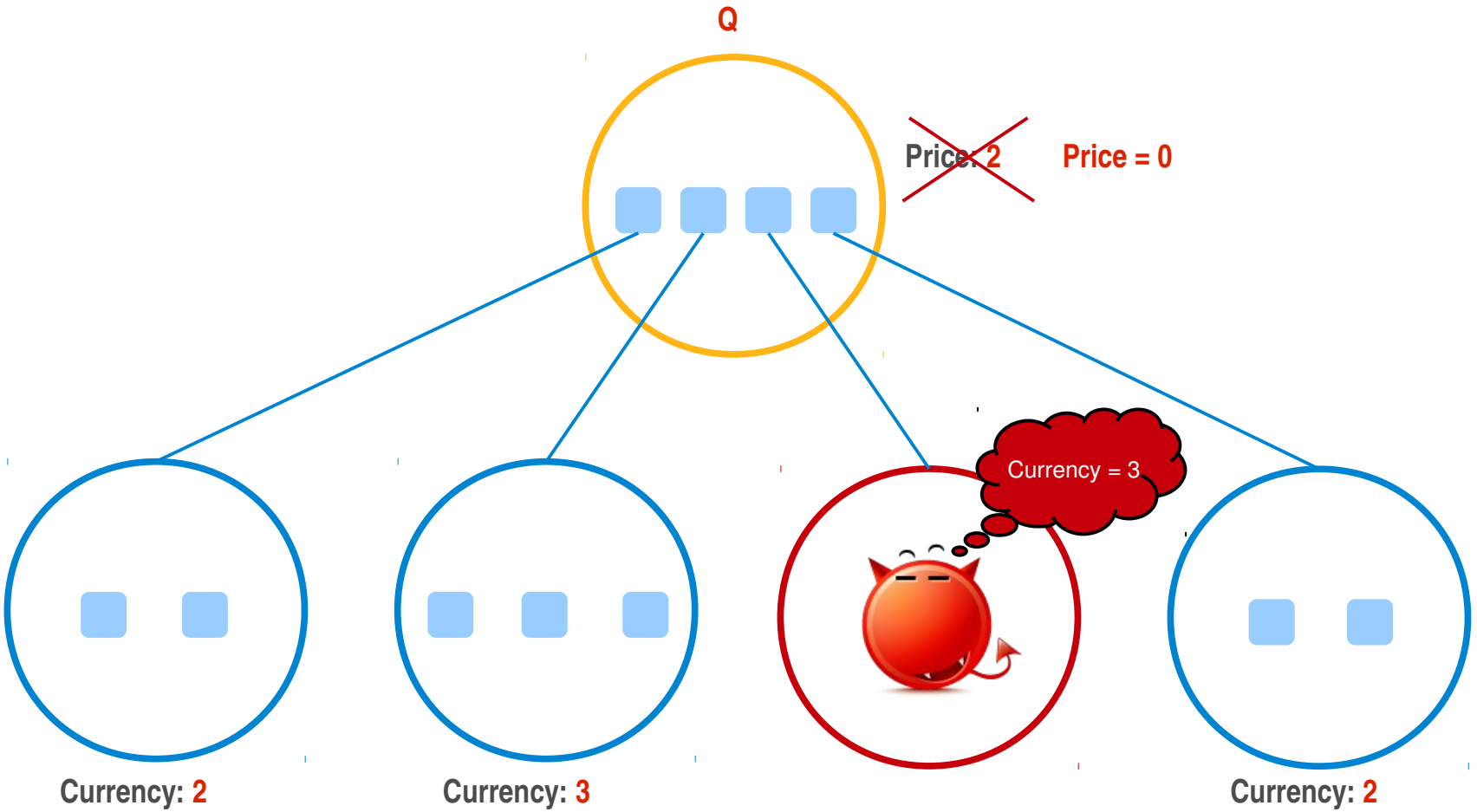
Freerider Detector

- F is the sum of
  - the number of audit responses not received before a timeout.
  - the number of negative audit responses.
  - the free upload slots.
- If F is more than M% of claimed upload slots, Q is suspected as a freerider.
- If Q becomes suspected in N consecutive iterations, it is detected as a freerider.
- The higher the value of N, the more accurate but slower the detection is.

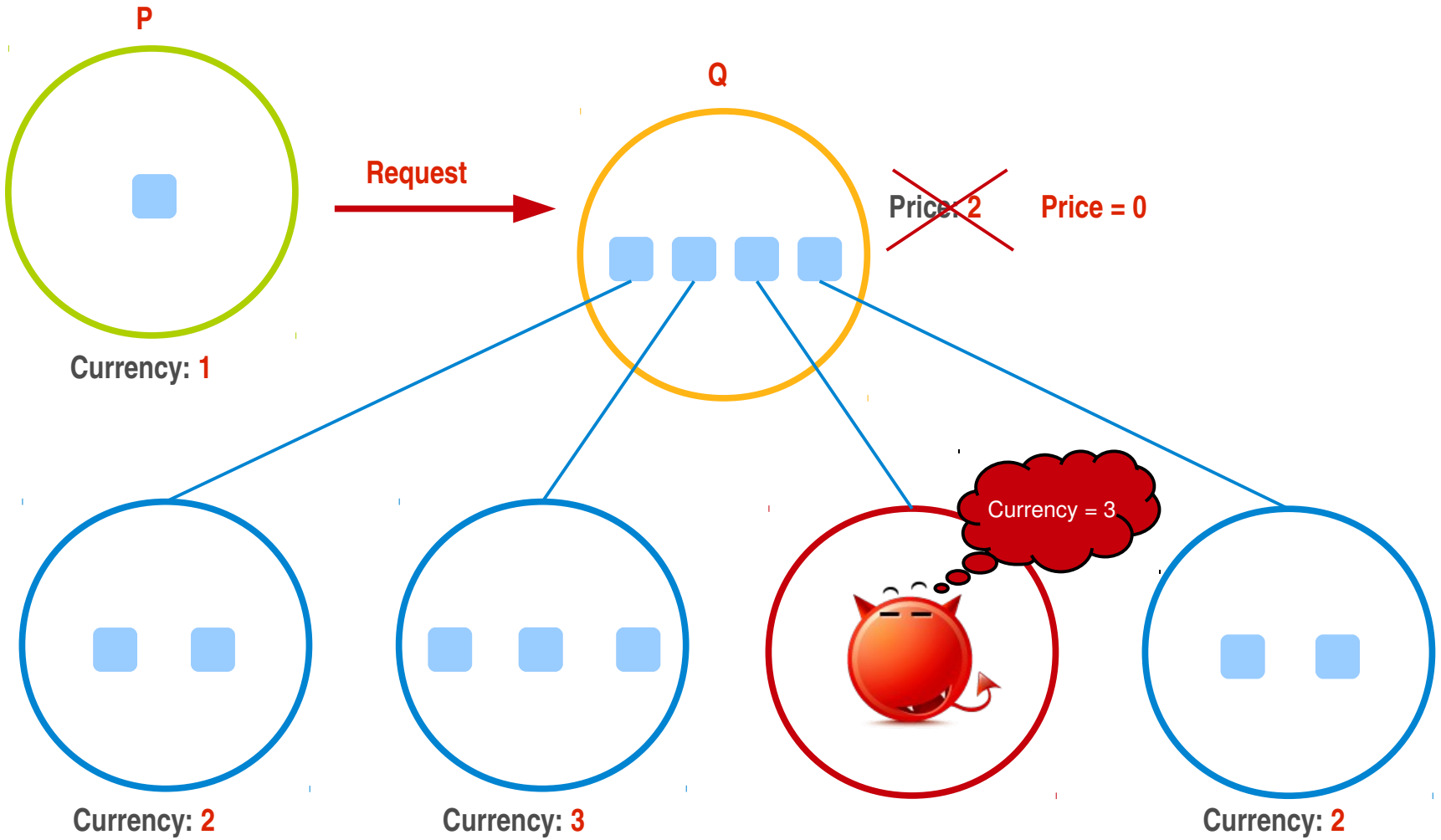
Freerider Detector – Punishment (1/4)



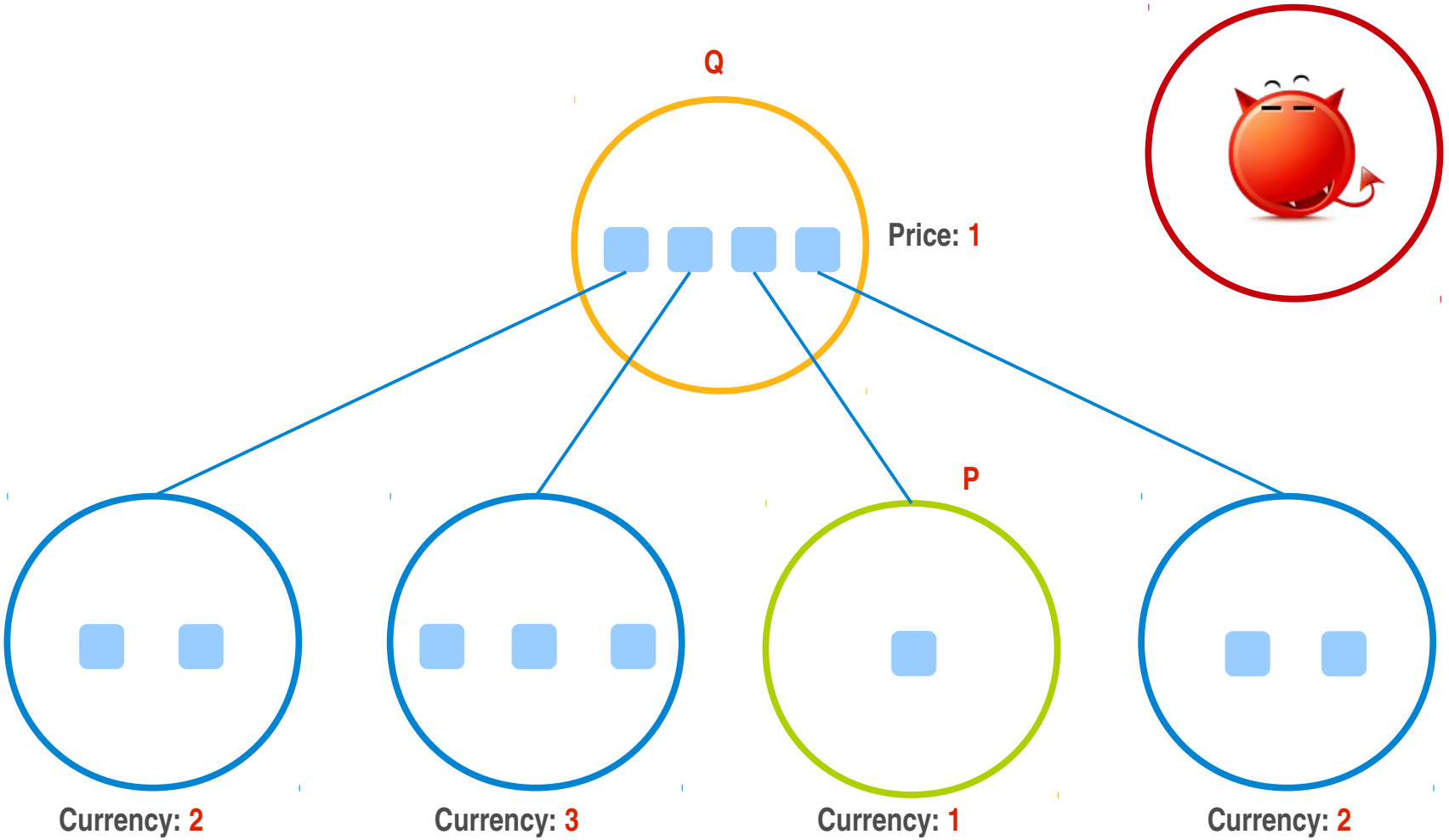
Freerider Detector – Punishment (2/4)



Freerider Detector – Punishment (3/4)



Freerider Detector – Punishment (4/4)

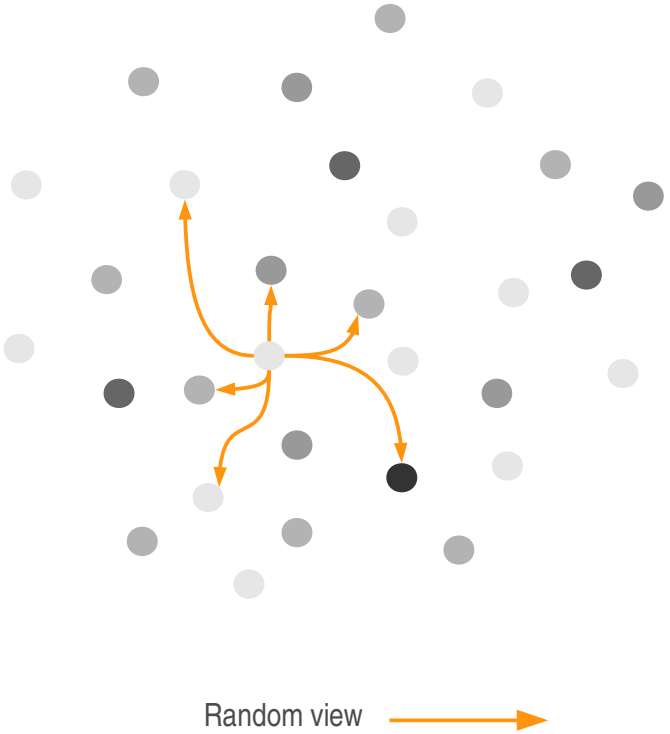


Optimization

Node Discovery

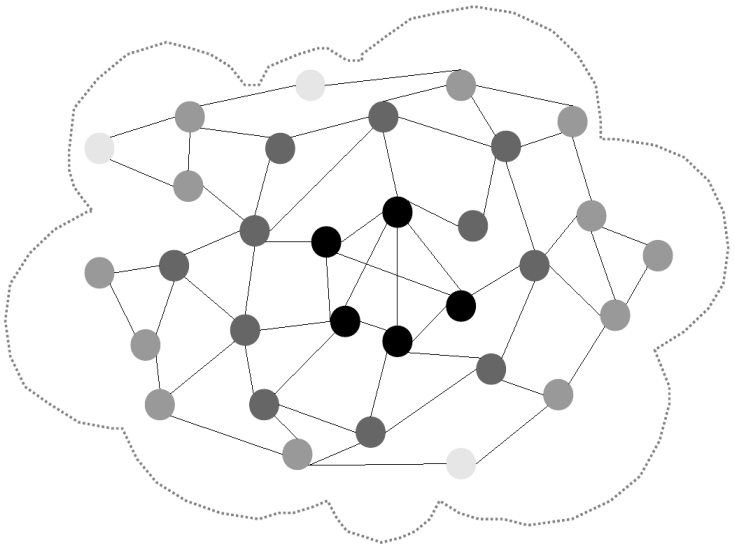
- Naïve solution: nodes in partial views are selected randomly from all the nodes.

- Optimization: nodes use the Gradient overlay to construct and maintain their partial view of the system.



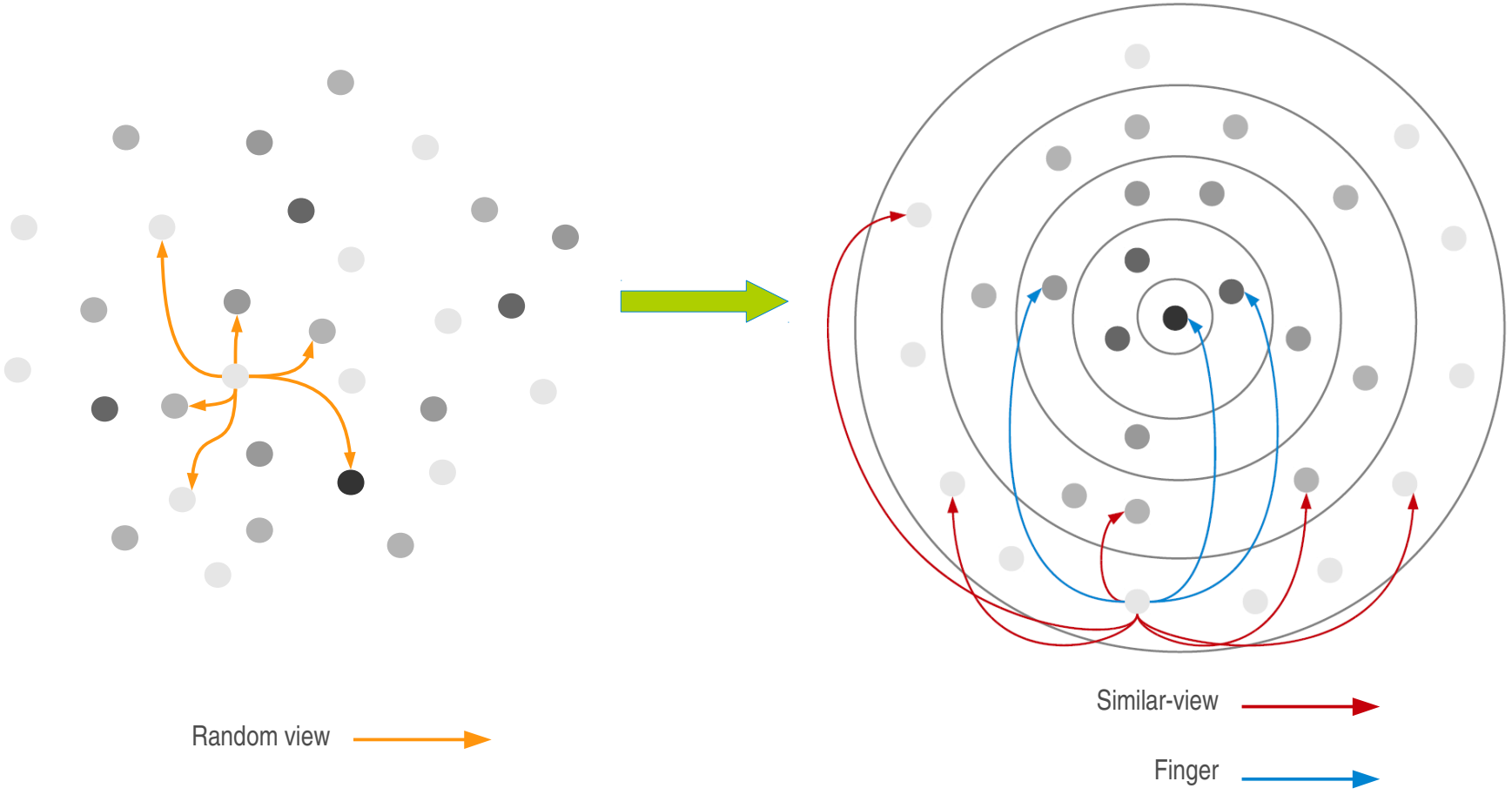
The Gradient Overlay

- The Gradient overlay is a class of P2P overlays that arranges nodes using a local utility function at each node, such that nodes are ordered in descending utility values away from a core of the highest utility nodes.



Peer Partners

- Rather than have nodes explore the whole system for better parents, the Gradient enables nodes to limit exploration to the set of nodes with asimilar number of upload slots.



CLive

Problem Description

Problem?

- Bottlenecks in P2P video streaming systems: upload bandwidth
- A potential solution: P2P network is assisted by a cloud computing.

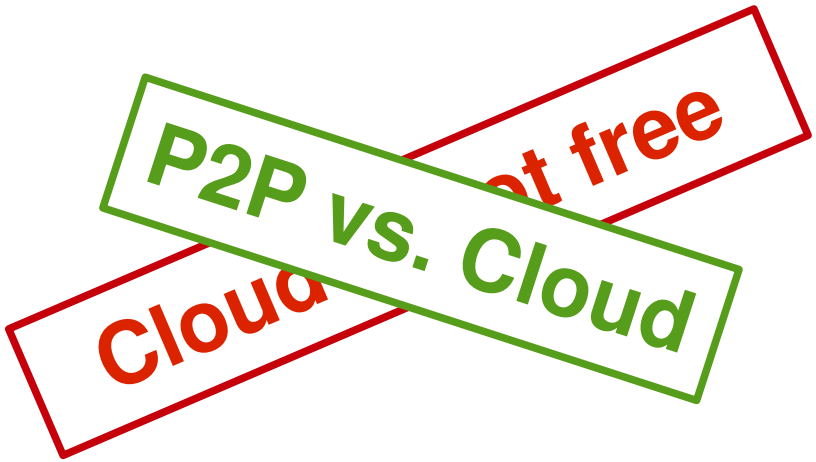
Problem?

- Bottlenecks in P2P video streaming systems: upload bandwidth

- A potential solution: P2P network is assisted by a cloud computing.

Problem?

- Bottlenecks in P2P video streaming systems: upload bandwidth
- A potential solution: P2P network is assisted by a cloud computing.



P2P vs. Cloud

- P2P pros and cons
  - P2P resources are cheap
  - Churn may compromise availability
- Cloud pros and cons
  - Superior availability
  - Cloud resources are not free

If You Cannot Beat Them, …? ;)

- The cloud as a support group for P2P.
- Reduce the number of (costly) cloud interactions as much as possible.

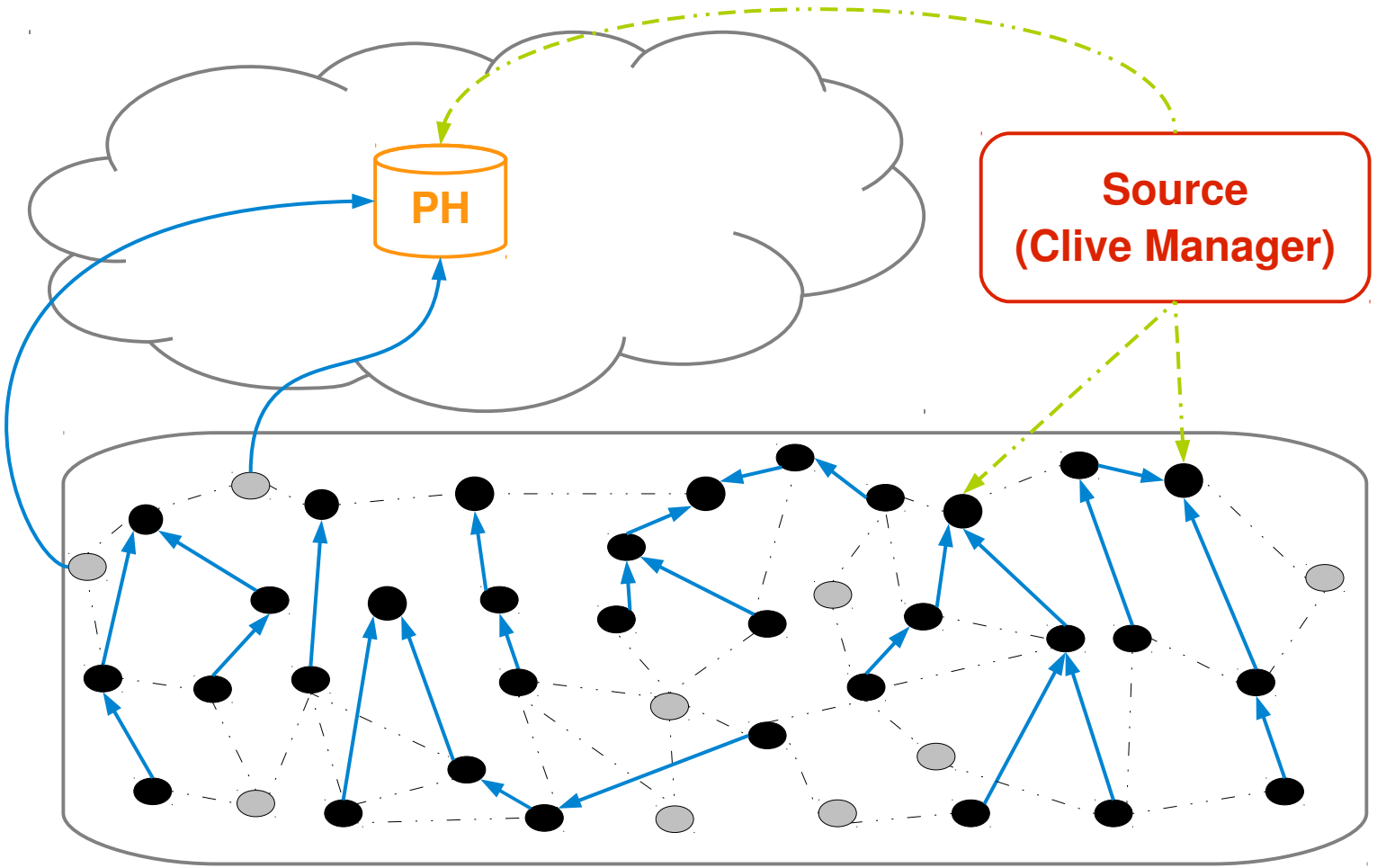
If You Cannot Beat Them, …? ;)

- The cloud as a support group for P2P.
- Reduce the number of (costly) cloud interactions as much as possible.

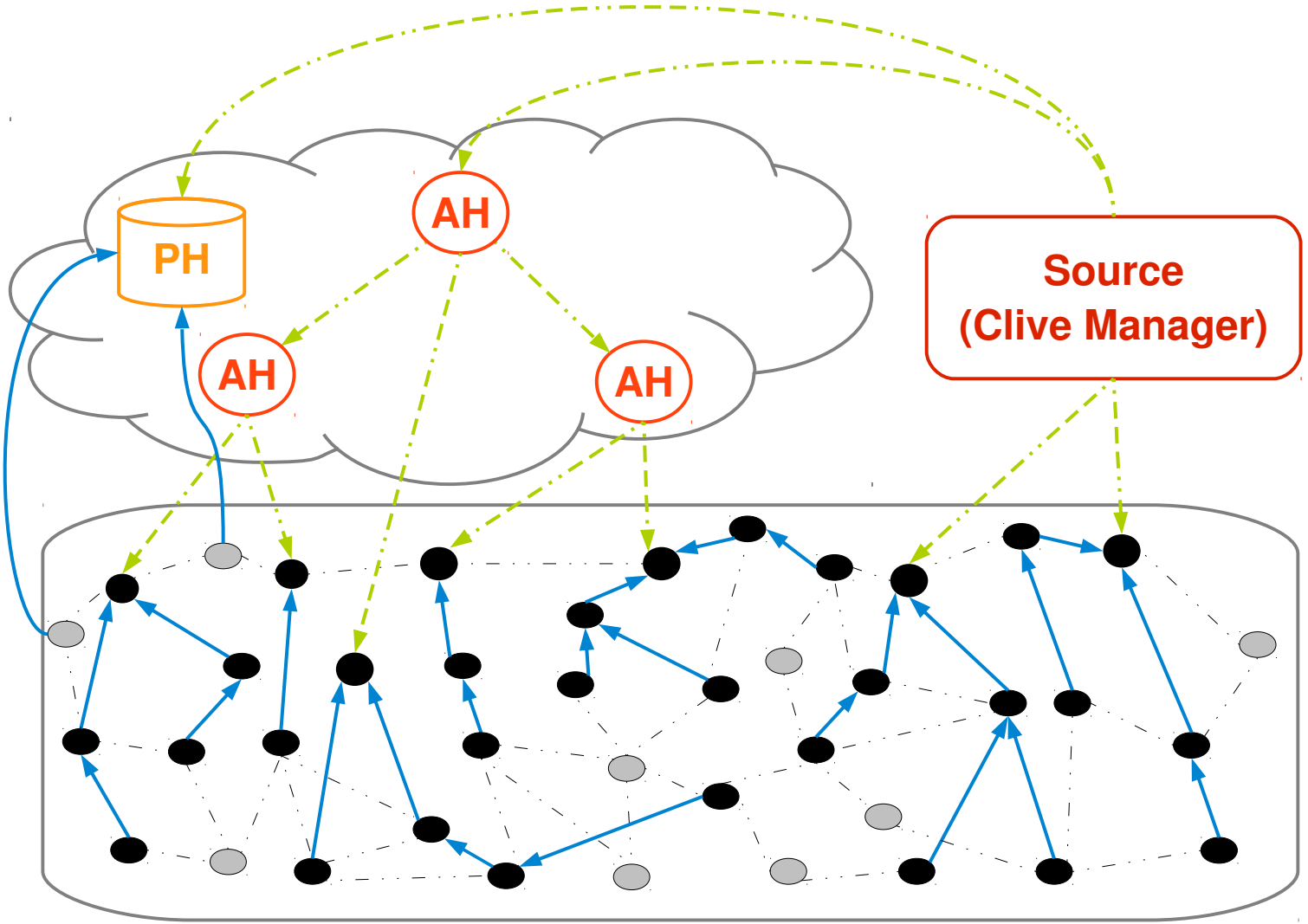
The problem to be solved becomes minimizing the economical cost, provided that a set of constraints on QoS is satisfied.

CLive Solution

Baseline Model

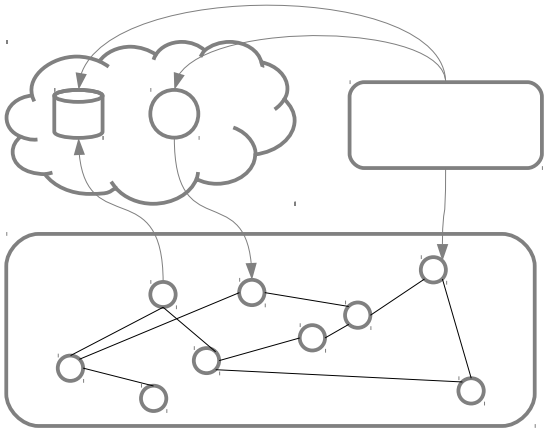


Enhanced Model



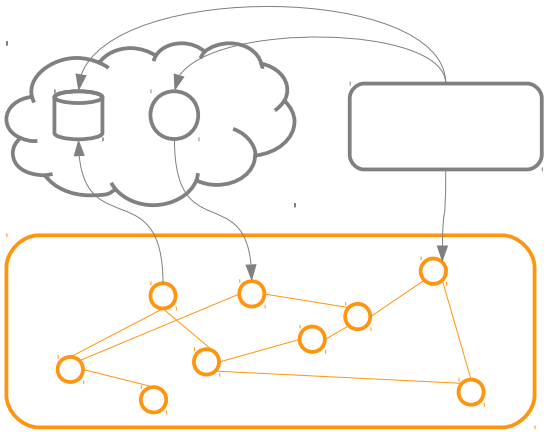
CLive Components

- P2P streaming overlay (swarm)
- Media source
- Passive Helpers
- Active Helpers
- Management component



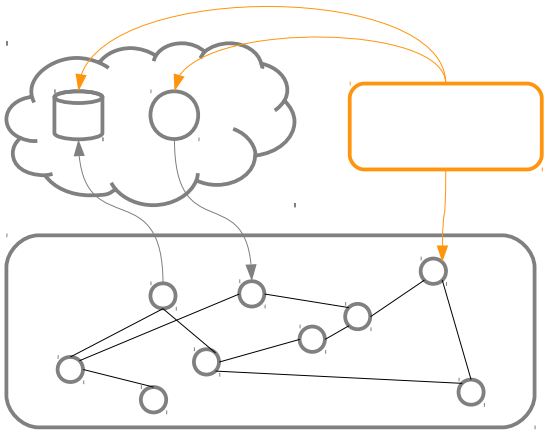
CLive Components: P2P Streaming Component

- We assume that nodes in the system use a mesh-pull model for data distribution.
- Nodes periodically send their buffer maps to their neighbours.
- The other nodes pull the required chunks from those nodes who own the chunks.



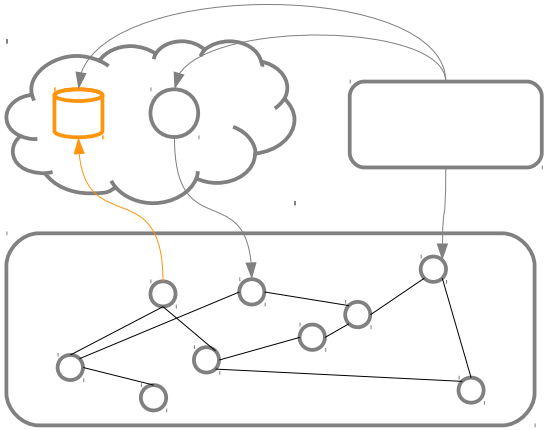
CLive Components: Media Source

- A media source is a node that generates data chunks and pushes them to the swarm.



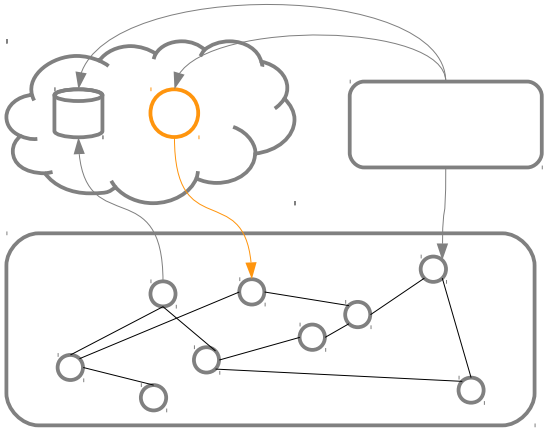
CLive Components: Passive Helper

- PH is a passive element that plays the role of a data storage, e.g., Amazon S3.
- The source pushes chunks to PH, as they are generated.
- The swarm nodes pull the missed chunks from it.
- We assume that a PH can serve as many requests as it receives.



CLive Components: Active Helper

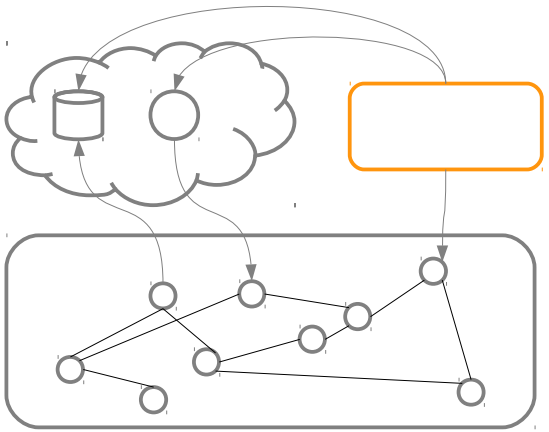
- AHs are active elements, e.g., Amazon EC2, that cooperate with other swarm nodes to accelerate the data dissemination.
- The source pushes chunks to AH, as they are generated.
- AHs forwards chunks to the swarm and other AH, as they are received.



CLive Components: Management

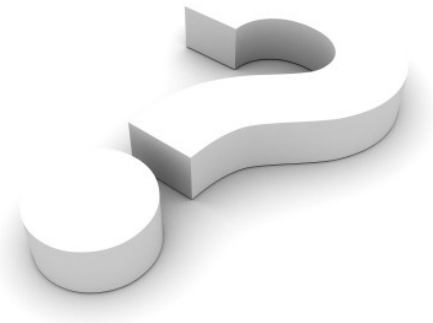
- Participates in a gossip algorithm to estimate the available resources in the system.

- Adds/removes AHs to/from the system, based on the estimation.



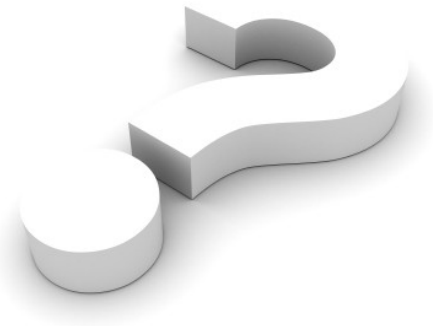
Two Main Questions in CLive?

- How to estimate the extra load in the overlay?
- How to relay the load to cloud with a minimum cost?



Two Main Questions in CLive?

- How to estimate the extra load in the overlay?
- How to relay the load to cloud with a minimum cost?



Main Idea

- Infected nodes: the number of nodes that can be served with the existing resources in the system.

- load = swarm size – infected nodes

The Swarm Size

- The swarm size estimation is easy: gossip-based aggregation

The Infected Nodes

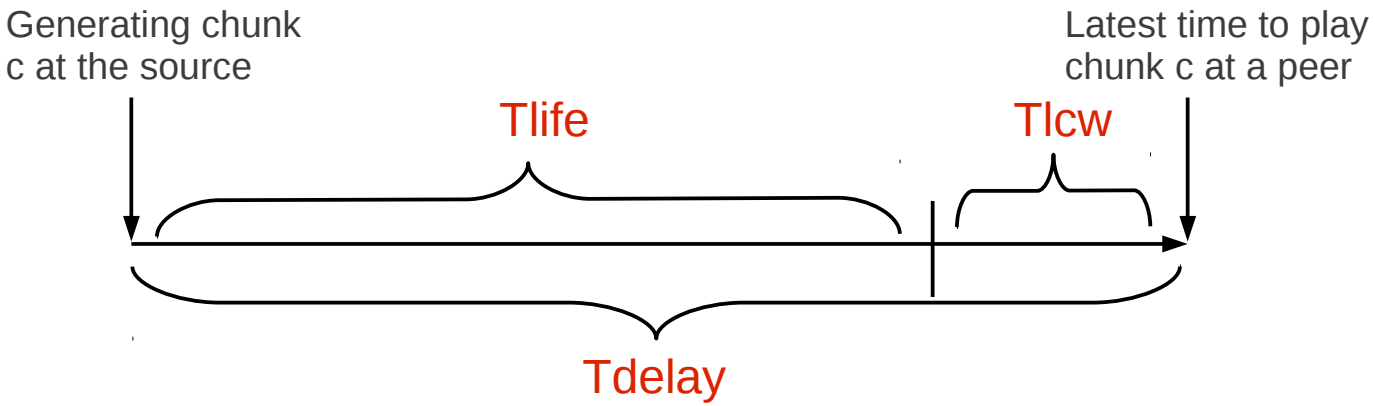
- It is shown that each streamed chunk through mesh overlays follows a tree-based diffusion pattern [1].

- To compute the number of infected nodes, we model a diffusion tree for a chunk and estimate the number of nodes in that tree.

[1] B. Biskupski, M. Schiely, P. Felber, and R. Meier, “Tree-based analysis of mesh overlays for peer-to-peer streaming,” in Proc. of the 8th int. conf. on distr. app. and interoperable systems. Springer, 2008.

Estimate the Tree Depth

- T_{delay}: maximum acceptable latency.
- $T_{life} = T_{delay} - T_{lcw}$



- If a node can not receive a chunk in T_{life}, it pulls that chunk from PH.
- The tree depth
  - $D_{max} = T_{life} / T_d$
  - T_d: average latency among the peers

Estimate the Fan-out Distribution

- The upload bandwidth distribution
  - Adam2
  - Gossip-based aggregation

Estimate the Number of Nodes in a Tree

- Estimate the tree depth
- Estimate the fan-out distribution

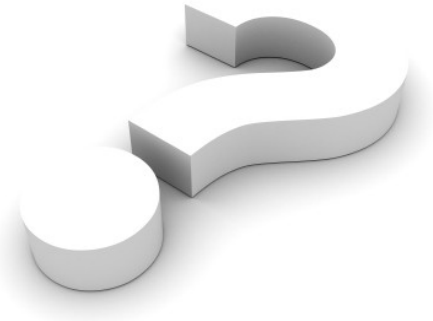
The Number of Infected Nodes

- N_{inf} = (the number of trees).n_{tree}
- The number of trees: source fanout + AHs fanout

Two Main Questions in CLive?

- How to estimate the extra load in the overlay?

- How to relay the load to cloud with a minimum cost?



AH Cost

- AH cost in one round: $C_{ah} = C_{vm} + m.C_{chunk}$
  - C_{vm}: virtual machine cost
  - C_{chunk}: chuck transfer cost
  - m: number of chunks that one AH upload per round

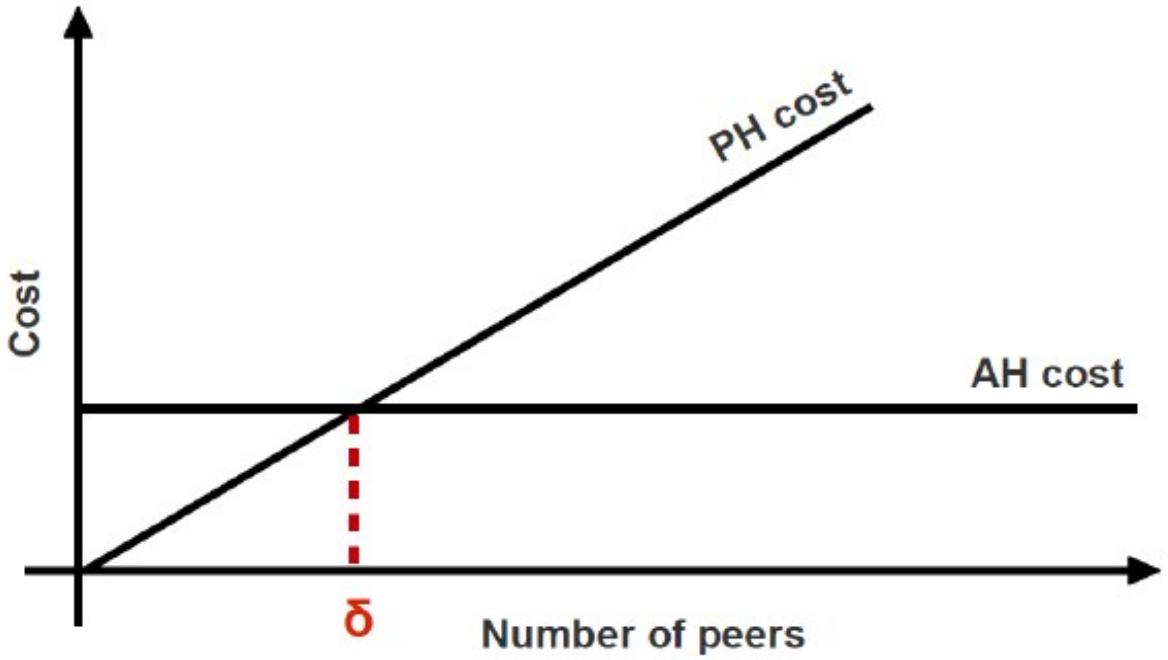
PH Cost

- PH cost in one round: $C_{ph} = C_{storage} + r.(C_{chunk} + C_{req})$
  - $C_{storage}$: the storage cost
  - C_{chunk}: chuck transfer cost
  - C_{req}: chuck request
  - r: the number of retrieved chunks from PH in one round

∂

- We define ∂ as the number of peers that is economically reasonable to serve with PH utilization instead to run an additional AH for them.

- $\partial = C_{ah} / C_{ph}$

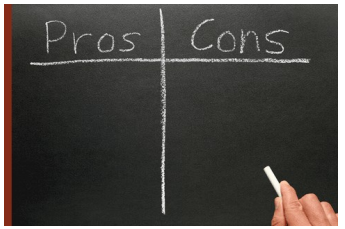


Manage the Cloud Resources

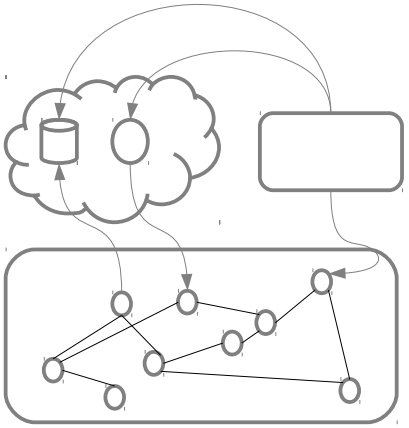
- If load > ∂: add AH
- If load < ∂ - H: remove AH
  - H: number of peers served by one AH.
- Otherwise don't change AHs.

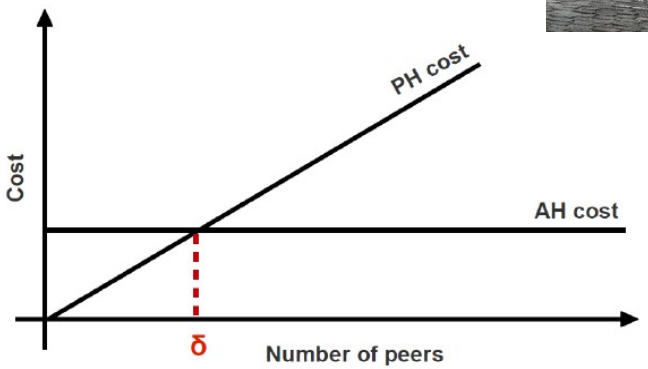
Summary


WARNING
CHALLENGES
AHEAD


Pros
Cons


WHAT
IS OUR
ONE
DEMAND?
#OCCUPYWALLSTREET
SEPTEMBER 17TH.
BRING TENT.




PH cost
AH cost
Cost
δ
Number of peers

Any Questions?