HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

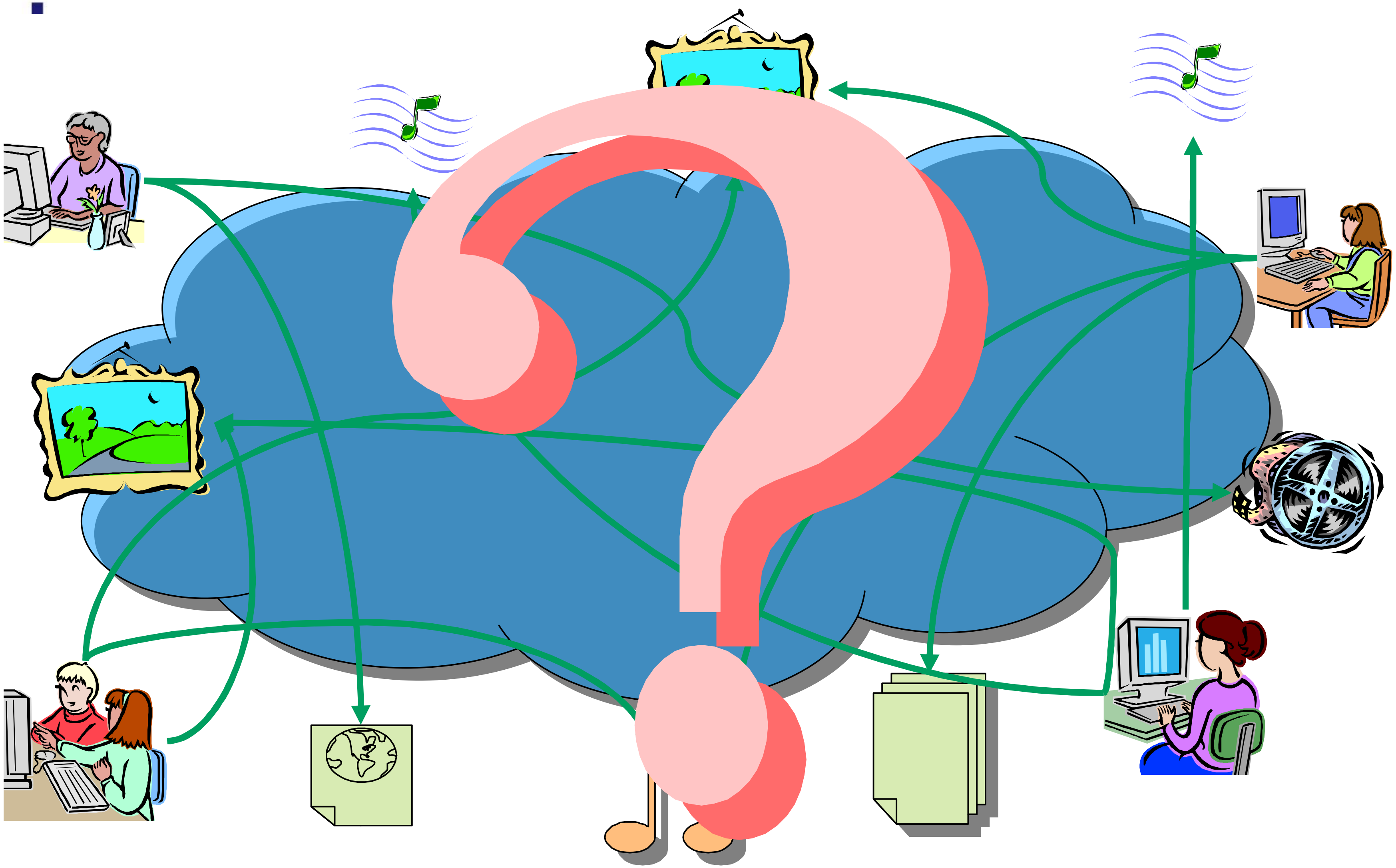
Peer-to-Peer Networks

Chapter 6: P2P Content Distribution

Chapter Outline

- Content distribution overview
- Why P2P content distribution?
- Network coding
- Peer-to-peer multicast

What is the Content Distribution Problem?

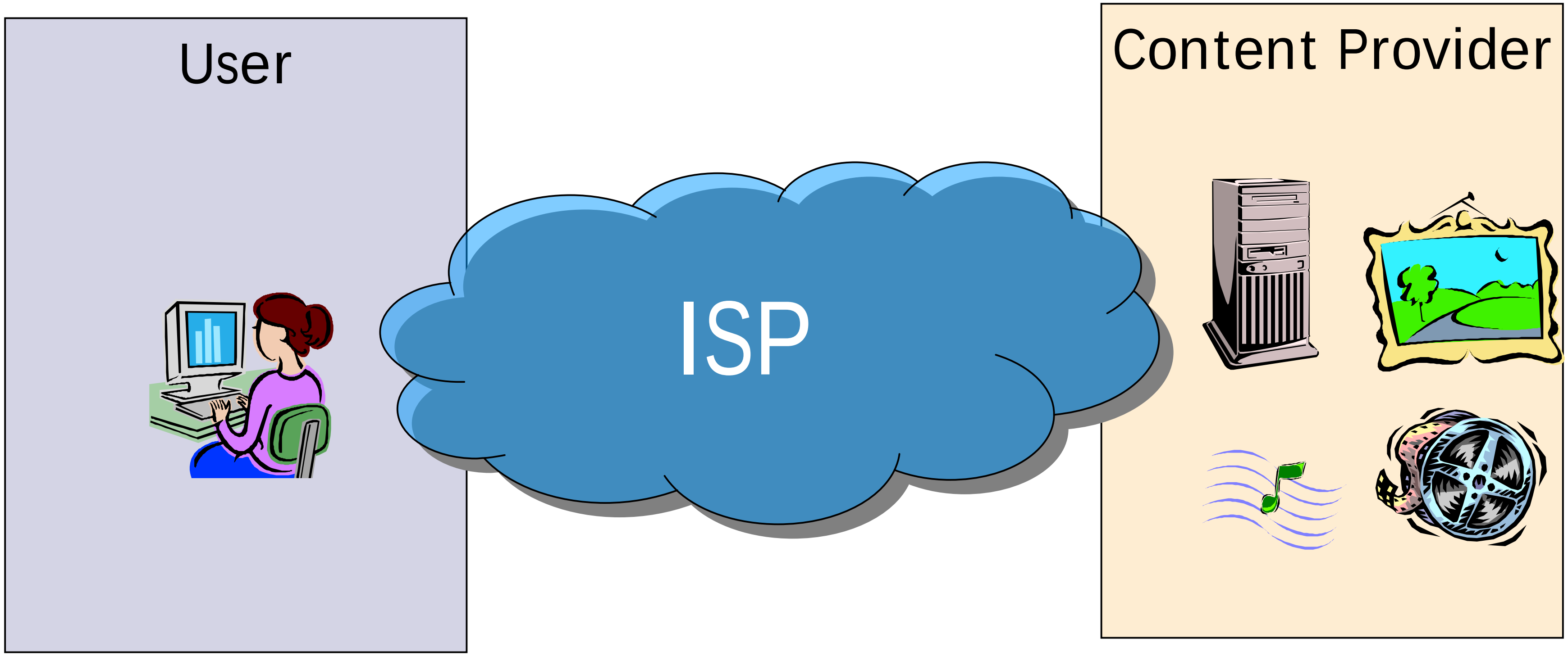


Problem Definition

- How to distribute (popular) content efficiently on Internet to a large number of geographically distributed people?
- …while keeping your costs manageable! :-)
- Who is “you”?
- So, we need to answer the following questions…

1. Who are the players?
2. What is the content?
3. How are costs determined?
4. How is performance measured?

The Players



- ISP represents the network path between the user and the content provider
- Typically, such a path contains several ISPs

Roles of the Players

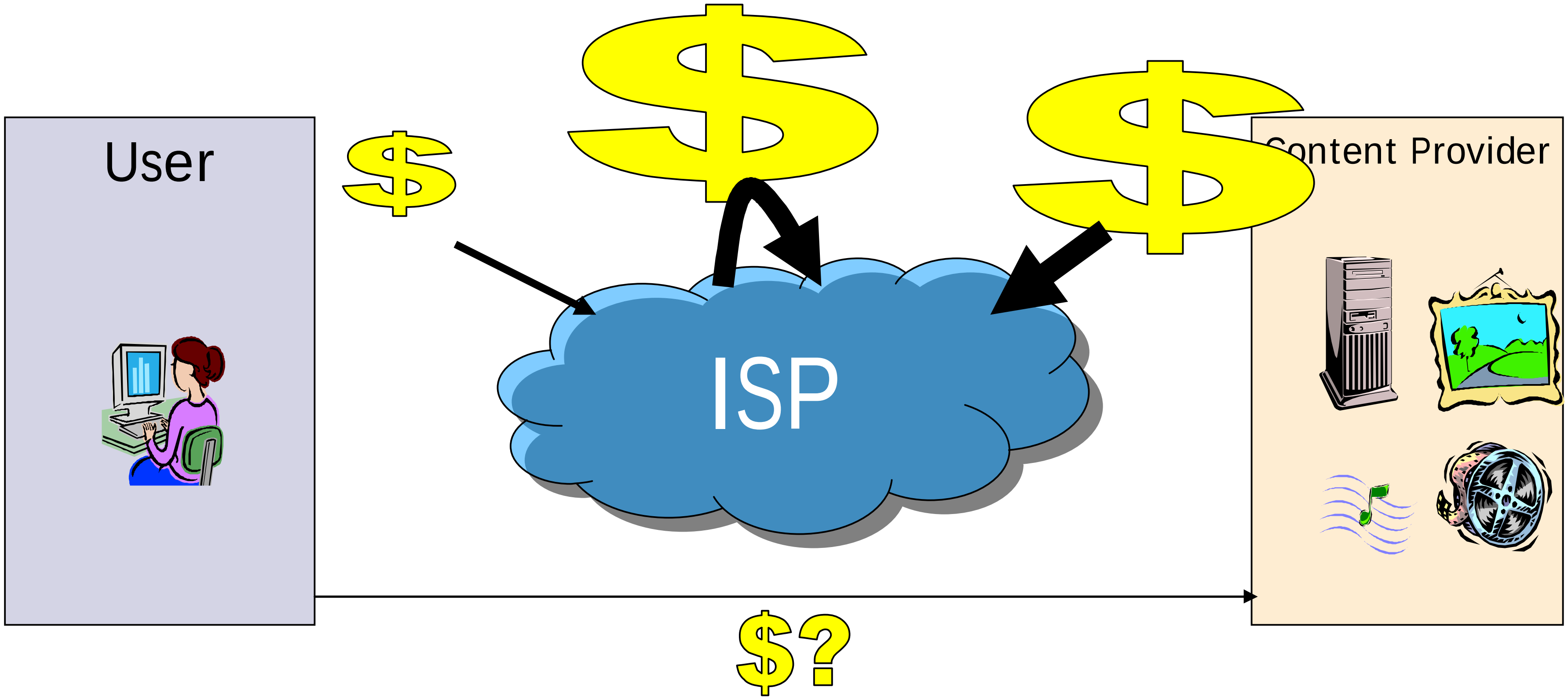
- User
  - Wants to access content provided by content provider
  - Buys access to network from ISP
  - Possibly pays content provider for content
- Content provider
  - Produces content
  - Runs a server (or outsources this)
  - Buys access from an ISP
- ISP
  - Provides the network connection for the others

What Is Content?

- Content is any digital content the users want
- Examples:
  - Web pages
  - Music files
  - Videos
  - Streaming media
  - and so on…

Follow the Money

- Where does the money come from?



Business Relationships

- User
  - Pays money to ISP for connectivity
  - These days often flat-rate, can be per-byte
  - Might rarely pay content provider
- Content provider
  - Pays ISP for connectivity
  - Pays for running a server
  - Needs to get money from somewhere
  - *May or may not need to make a profit from content*
- ISP
  - Gets money from others
  - Uses money to run network
  - *Wants to make a profit*

Performance Targets

- User
  - Wants content fast
  - Does not want to pay (too much)
- Content provider
  - Wants to get as many users as possible
  - While maintaining costs as low as possible
- ISP
  - Wants to make as much money as possible
- Which goal is the most important?

- In real world, user needs often not "relevant"
  - Users do not contribute enough money
  - Users might not have a choice, e.g., only 1 ISP possible
- ISP is the most important in many cases
- Content providers have some say in some cases
  - Compare proxy caching and CDNs

Content Distribution Technology

- Content distribution is an “old” problem
  - Read: Originates from the Web
- Technology currently in use:
  - Client-side web proxies/caches
  - Server-side load balancing
  - Mirror servers
  - Content Distribution Networks (CDN)
  - Peer-to-peer content distribution
  - Multicast (at least in theory…)

File Sharing?!?

- Is file sharing content distribution?
- YES:
  - There is content being distributed
- NO:
  - Goal is to *make content available*, not the actual distribution
  - Main focus is on searching and locating content
  - Actual distribution just a simple download with nothing fancy
- We exclude file sharing from content distribution

Peer-to-Peer Content Distribution

- Definition:

  Peer-to-peer content distribution is content distribution which is implemented in a peer-to-peer fashion

- Best/only current example: BitTorrent
  - One file, replicate to everyone as fast as possible

- Why P2P content distribution?

P2P Content Distribution: Pros and Cons

Advantages

- Cheap for content provider
  - Need to seed file, that's it!
  - Server farms, CDNs, etc. are very, very expensive
- Capacity increases with increasing number of users

Disadvantages

- Users must want to help
  - Incentives?
- Need to trust users
- No (easy) way to optimize
  - Topological locality
- P2P has "bad reputation"

Current State

- Real world:
- BitTorrent and nothing much else

- Research world:
- BitTorrent under active research
  - Piece and peer selection algorithms
  - Measurement work
- Avalanche from Microsoft
  - Same as BitTorrent, but uses network coding
- P2P Streaming
  - Lots of proposals

Network Coding: Motivation

- Main question in BitTorrent:
  Which piece to download next?
- BitTorrent's answer: Rarest first!
- Rare blocks might disappear from system?
- How about advanced coding techniques?
  - Erasure codes at the source
  - Network coding by the peers

Erasure Codes at the Source

- Original source creates redundant blocks
  - Reed-Solomon, Tornado, Digital Fountain codes, …

- Original file has *N* blocks, source creates *K* blocks
- Generally applies:

  Any *N* out of *N*+*K* are sufficient to reconstruct file
  - Some coding schemes require a bit more than *N*

- Problem: Redundant data generated by one entity
- Problem: *K* is fixed
  - Some codes allow for “infinite *K*”

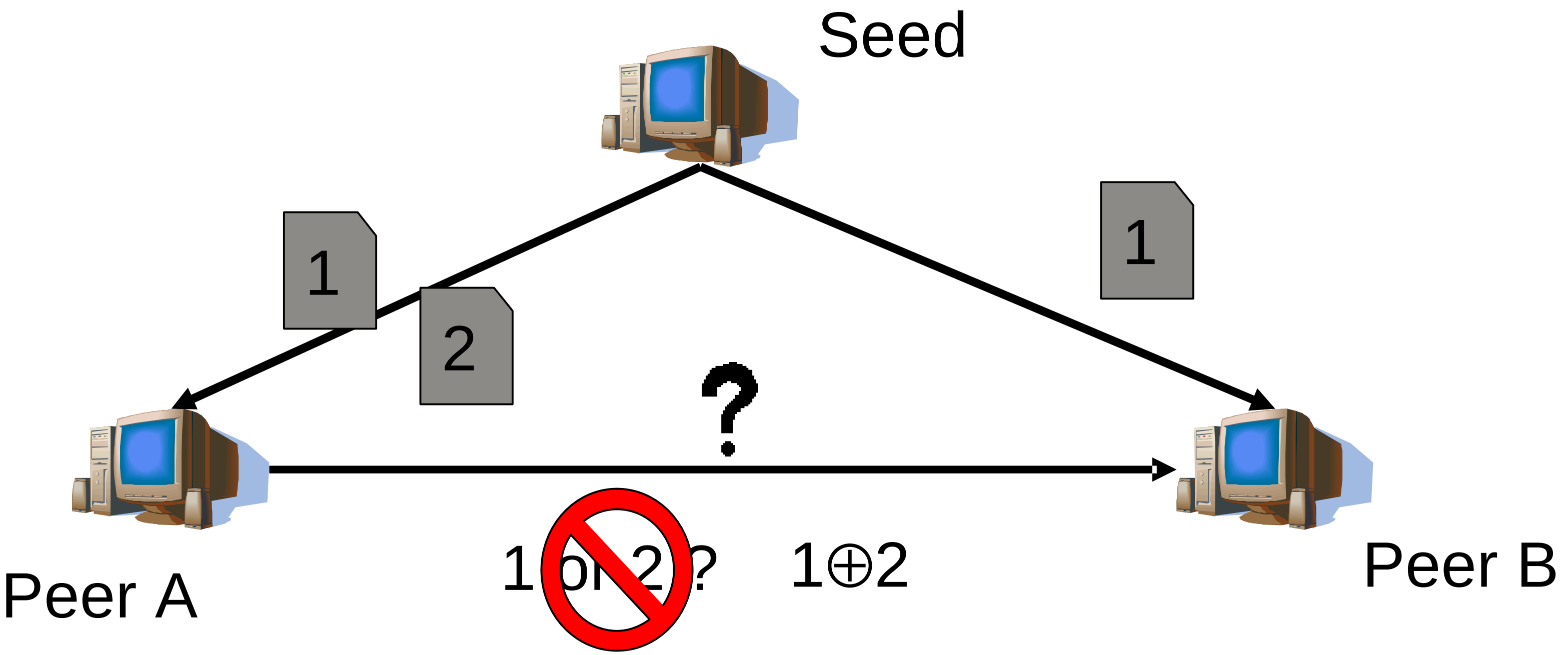
Network Coding

- How about letting peers do the coding?
- Solution: Network coding

Idea:

- Peers create linear combinations of blocks they have
- Send block with coefficients to others
- Sufficient number of linearly independent blocks allows for reconstruction of file

Network Coding



- Peer A sends a linear combination of blocks 1 and 2
- Peer B already has block 1, can compute block 2

Network Coding: Math

- File has *N* blocks
  - For example $F = [x_1, x_2]$ (two blocks)
- Pick two numbers $a_{i,1}, a_{i,2}$

- Code $E_i (a_{i,1}, a_{i,2}) = a_{i,1} * x_1 + a_{i,2} * x_2$
  - Can create an infinite number of E_i's

- When we have two linearly independent $E_i (a_{i,1}, a_{i,2})$'s, we can re-create original blocks x_1 and x_2
  - Same as solving a set of linear equations

- Note: All math in some finite field, e.g., GF(216)

Practical Considerations

- In principle, *information* is spread out better
- Need for linear independence
  - May need to download more data than file has
  - No analysis of relevance of this done yet
- Performance has been found to be good
  - Download times shorter and more *predictable*
- System very robust against departures of seeds
- Does not need altruistic users to stay logged on

Peer-to-Peer Multicast and Streaming

- Streaming content delivery gaining importance
  - Mainly for video-on-demand systems
- Many systems for P2P streaming or multicast

	One recipient	Multiple recipients
Bulk data	Traditional unicast	Multicast
Media data	Unicast streaming	Multicast streaming

Why Multicast?

- Multicast is sending data to many receivers at the same time
- Compare: Broadcast is sending data to everybody
- Two-kinds of multicasting schemes:

1. *1-to-m multicast*
   - One sender, many receivers, e.g., video streaming
2. *n-to-m multicast*
   - Many senders, many receivers, e.g., video conferencing

- For content distribution 1-to-m multicast is more relevant (and more researched)
- Multicast has two main components:
  - Group management
  - Routing

Multicast Group Management and Routing

- Group is another name for the receivers
- Group management is about how and who manages the membership in the group
  - Centralized, de-centralized, …

- Multicast routing is how to route the packets
- Multicast property: Ideally, each packet will go over any given network link at most once
- Achieved with a multicast tree, rooted at the sender
- Requires support from the routers in the network
- IPv4 has multicast address block (block D, 224.x.x.x)
  - Support from routers not guaranteed
- IPv6 should have native support for multicast

Why ALM?

- IP-level multicast is efficient, but requires support from all the routers in the network
- Most routers do not have multicast support turned on
- Result: No Internet-wide multicast possible
  - mbone overlay network offers some support
- Application level multicast performs multicasting on the application level
- Benefit: No support from routers needed
- Disadvantage: Not as efficient as native multicast
- ALM builds an overlay network
- Overlay typically consists of the receivers and senders
  - Some overlays assume fixed nodes in network

ALM Details

- ALM can be classified according to:

1. How they build distribution tree?
   - First mesh, then tree, or directly as spanning tree?
2. How to build overlay?
   - Use a structured network, e.g., DHT as basis?
   - Build directly on top of end-user machines?

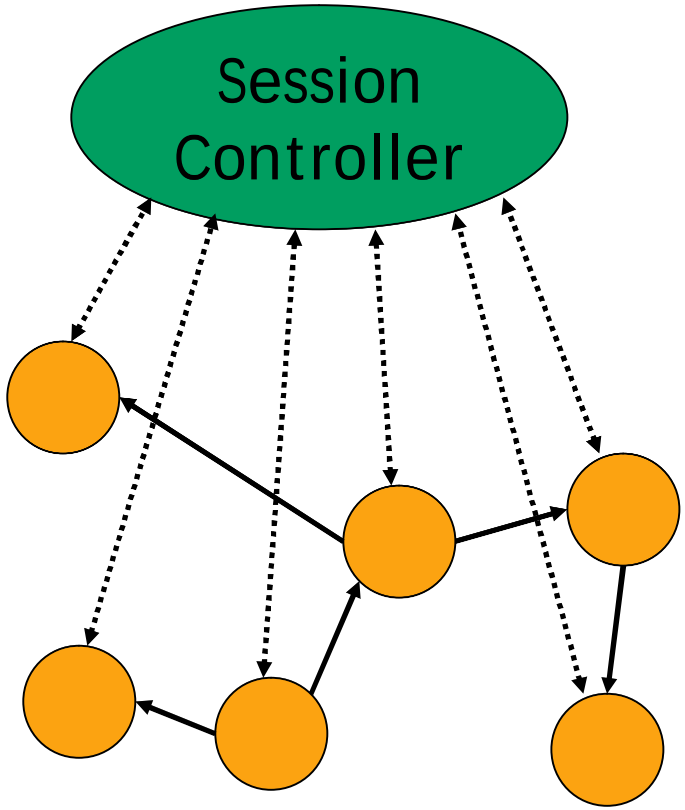
- ALM metrics:
  - Relative Delay Penalty: How much does the one-way delay increase because of overlay (ratio of overlay/direct path)
  - Throughput
  - Stress: How many times a packet traverses a given link
  - Control overhead

ALM Architectures

- Architectures based on unstructured overlays
  - Centralized architecture
  - De-centralized architecture
- Architectures based on structured overlays
  - Flooding-based multicast
  - Tree-based multicast
- Structured overlays are DHTs in this case

Application Level Multicast Infrastructure

- Also known as ALMI
- ALMI is a centralized, unstructured overlay system
- One session controller controls the distribution
  - Not part of actual data transfer, just control
- Session controller calculates optimal multicast tree



ALMI Details

- When a node wants to join the tree, it contacts the session controller
- Node is assumed to know address of session controller
  - For example, well-known URL, etc.
- Session controller assigns ID to node and gives information about where the node is in the tree
- Joining node contacts its parent in the tree
  - Parent adds new node as child
- When node leaves, it informs the session controller

ALMI Pros and Cons

Advantages

- High control over overlay
- Easy to implement
- Easy to detect malicious nodes
  - Because of centralized nature

Disadvantages

- Poor scalability
- Session controller is a single point of failure
  - Backup controllers monitor main controller
  - When main controller fails, one of the backups takes over

End System Multicast (ESM)

- Fully distributed ALM scheme based on end nodes
- Group management handled by Narada protocol
- Idea of Narada:

1. Overlay nodes organize themselves in a mesh
2. Source-specific multicast trees are built on top of mesh

- Hence, quality of multicast depends on quality of mesh
- Goal of Narada is to match overlay requirements with underlay quality
  - Meaning: Overlay link should have same properties as the corresponding underlay link between the same nodes
- Limited out-degree of nodes to limit load

Narada Group Management

- Every node participates in group management
- Does not scale well, but Narada's target is small to medium sized multicast networks
- Joining node contacts some existing members
  - Again, addresses assumed to be somehow known
- Every member sends periodic heartbeats messages
- Heartbeats announce e.g., arrival of new node
- Eventually, all nodes know about the new node
- Departures detected by missing heartbeats
- Can also repair network partitions:
  - If heartbeats missing, then send probe to that node
  - If node answers, then reconnect overlay
  - Otherwise assume node is gone

Narada Performance

Relative Delay Penalty

- For longer underlay delays, RDP between 2 and 4
- For short underlay delays, RDP up to 14
- Explanation: For short underlay links, even a slightly suboptimal overlay will lead to high RDP

Stress

- Narada limits maximum stress to 9
- Sending everything over unicast has maximum stress 128

Flooding-Based Replication

- Use a structured overlay to construct multicast trees
- This case: Use Content Addressable Network, CAN
- Multicast in CAN works as follows

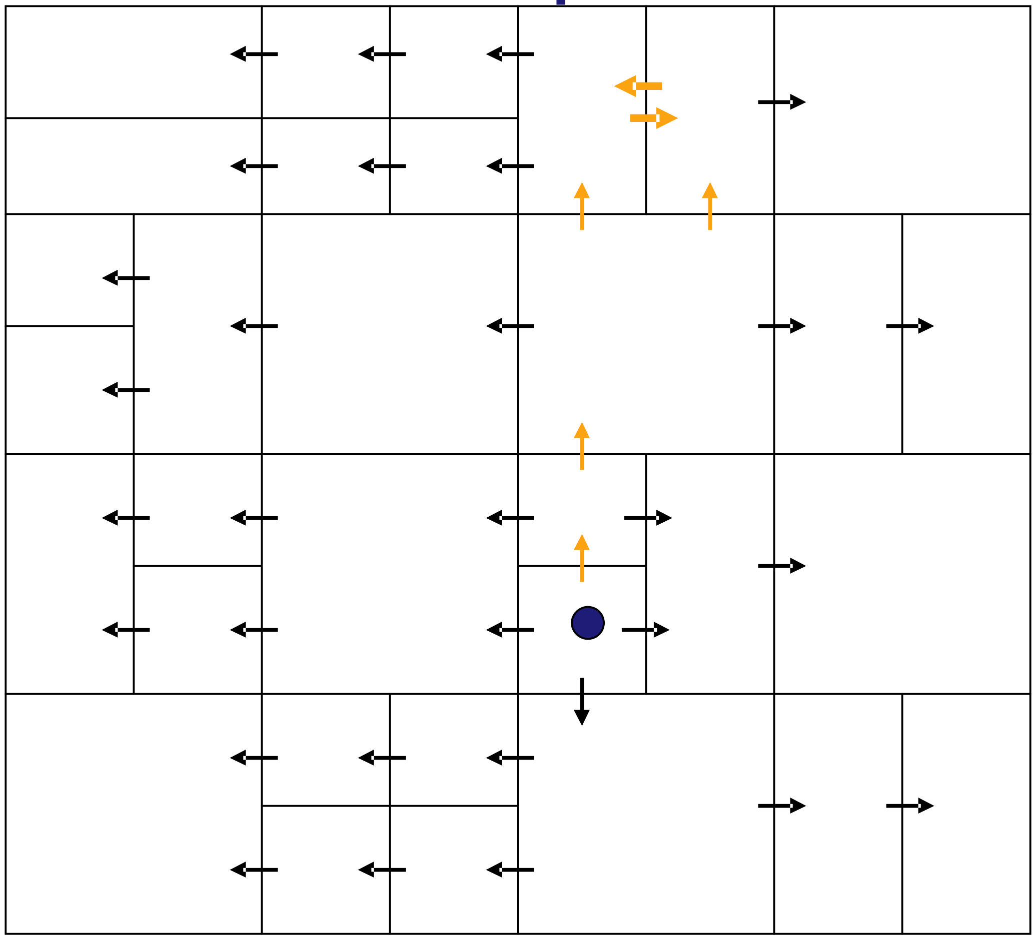
1. Multicast group has a well-known identifier which is mapped to some point in CAN
2. When node wants to join group, it contacts the responsible node for the group ID
3. Responsible node redirects new node to a mini-CAN
   - Mini-CAN has only nodes who are members of multicast group

CAN Multicast Rules

1. *Source forwards message to all its neighbors*
2. *When node receives message, it forwards it to all its neighbors, except the ones in the direction from which the message came*
3. *If message has traversed more than half of the distance across any dimension, message is not forwarded*
4. *Nodes cache sequence numbers of messages and discard any duplicates they see*

- Rules 1 and 2 ensure messages reach everywhere
- Rule 3 prevents routing loops
- Rule 4 is just performance enhancement

CAN Multicast Example



- Orange arrows lead to duplicates
- Note: Not all arrows shown in figure

Tree-Based Example

- Tree-based multicast with structured overlays
- Example: Scribe
  - Based on Pastry DHT
- Idea: Multicast group ID maps to a node in DHT
- That node is root of the multicast tree
  - Also called rendezvous point (RP)
- When a node joins, it sends message to RP
- Message routed in DHT
- Every node on the path checks if it knows about that multicast group
  - If yes, then add new node as child
  - If not, add new node, forward message towards RP

Scribe: Sending Data

- When a node wants to send data to a multicast group, it sends it to the rendezvous point
- Join messages have created the multicast tree rooted at the rendezvous point
- RP forwards the message to its children, who forward it to their children, etc.

Comparison of CAN and Scribe

Relative Delay Penalty

- CAN has higher RDP, typically between 6 and 8
- Scribe has RDP about 2

Stress

- Average stress a bit lower in CAN
- Maximum stress in CAN much lower

- Scribe better for delay-sensitive applications
  - For example, video conference
- CAN better for non delay-critical applications
  - For example, TV broadcast

Chapter Summary

- Content distribution overview
- Why P2P content distribution?
- Network coding
- Peer-to-peer multicast