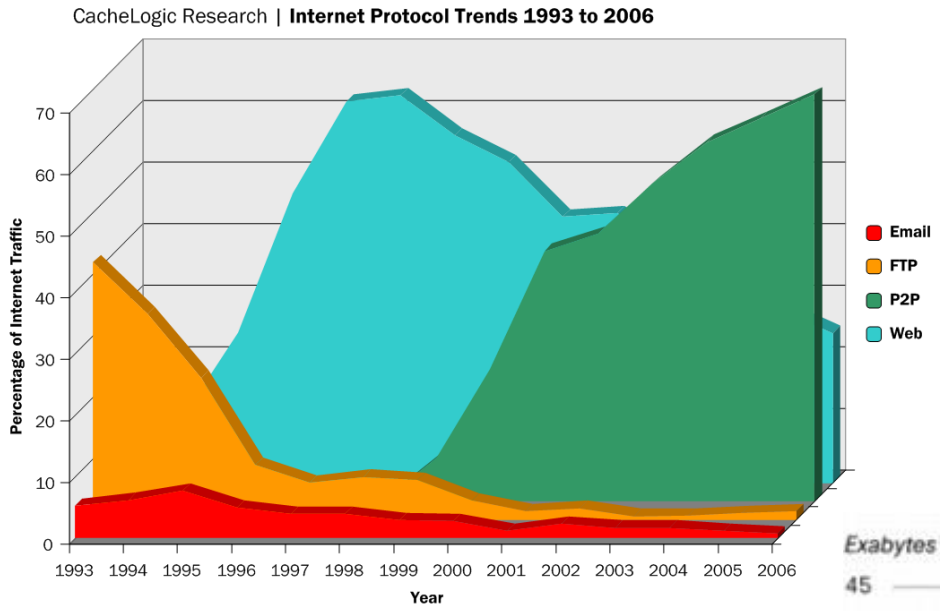
Content Distribution via P2P Networks

Thank for part of slides from Prof. Dah Ming Chiu from Chinese University of Hong Kong

P2P Traffic in Internet



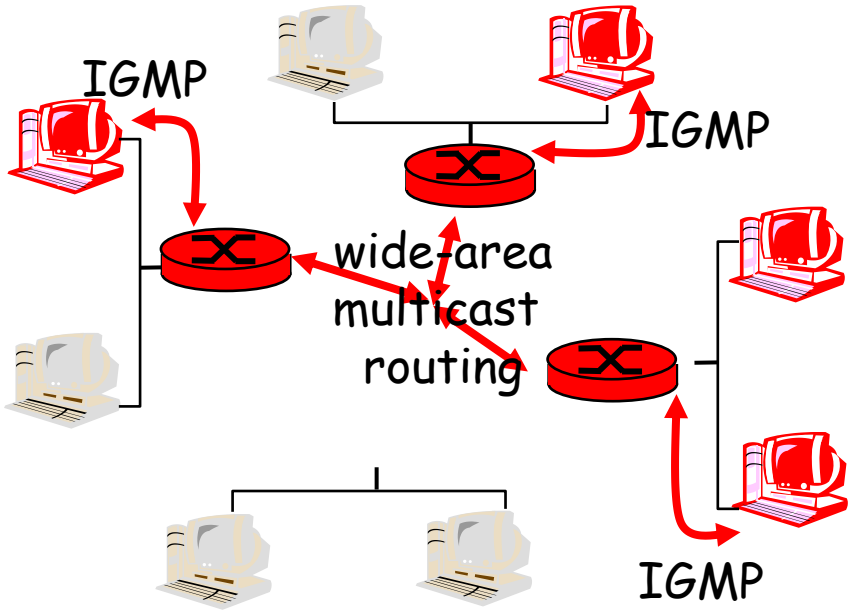
Content Distribution to Many

- There are three basic approaches:
  - Network multicast
  - Content Distribution Networks
    - Akamai
    - Coral – a "p2p based" of CDN
  - P2P content distribution
    - End-system multicast (application layer multicast)
    - Ways to maintain multiple trees
      - Tree-based (push)
      - Data-driven (pull)

Network Multicast

- Rely on routers to duplicate content and deliver it to all receivers – IP multicast
- How does it work?
  - Get a multicast address for given content, and let receivers know about it somehow
  - Receivers indicate interest to their local routers
    - Internet Group Management Protocol (IGMP)
  - Routers from a tree connecting to the source – this is a tricky part to implement
    - Several IP multicast routing protocols
  - Sender can send data using UDP socket interface
    - Data will traverse a link no more than once

IP Multicast Model



To the sender, it is like broadcasting – it does not know who are listening

Key Concerns with IP Multicast

- Scalability with number of groups
  - Routers maintain per-group state
  - Analogous to per-flow state for QoS guarantees
  - Aggregation of multicast addresses is complicated
- Supporting higher level functionality is difficult
  - IP Multicast: best-effort multi-point delivery service
  - End systems responsible for handling higher level functionality
  - Reliability and congestion control for IP Multicast complicated
- Deployment is difficult and slow
  - ISP's reluctant to turn on IP Multicast

Is There a Need for P2P

A good question for some discussion:

- Wide-area multicast still too demanding for IP multicast technology – use application layer multicast, i.e. P2P
- The curse of "synchrony":
  - IP multicast requires all receivers to receive at the same time
  - P2P is more flexible for many other applications:
    - file sharing
    - VoD
    - Distributed storage
- Local management overhead and costs
- …

Content Distribution without IP Multicast

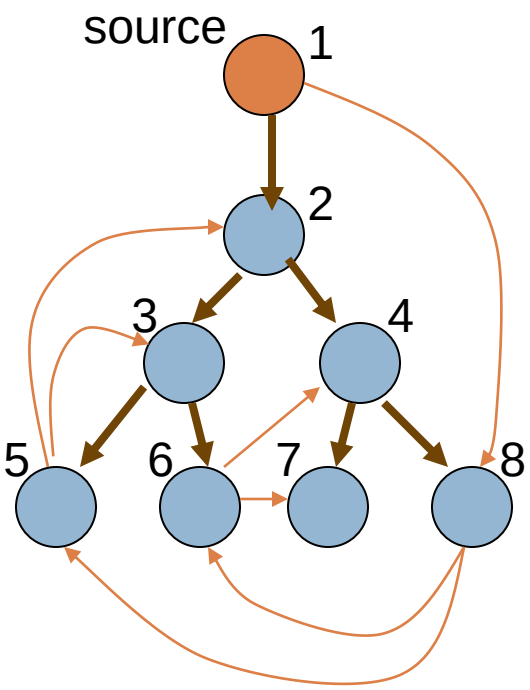
- Imagine a teach teaching a course of n lectures
  - Suppose there is NO classroom
  - The teacher can only teach one student at a time in his office
  - There is a large number (m students)
  - What to do?
- CDN:
  - Hire as many TAs as possible; teach the TAs, and make TAs teach students
- P2P:
  - Make a deal with students: after I teach you, you must be willing to share it with other students

CDN

- CDN = Content Distribution Networks
- A content provider deploy servers to replicate content at different parts of Internet
- A client request is “redirected” to a nearby server
- The most well-known CDN is Akamai
  - A well established business model
  - Akamai claims its platform delivers 20% of the Internet traffic
    - http://www.akamai.com

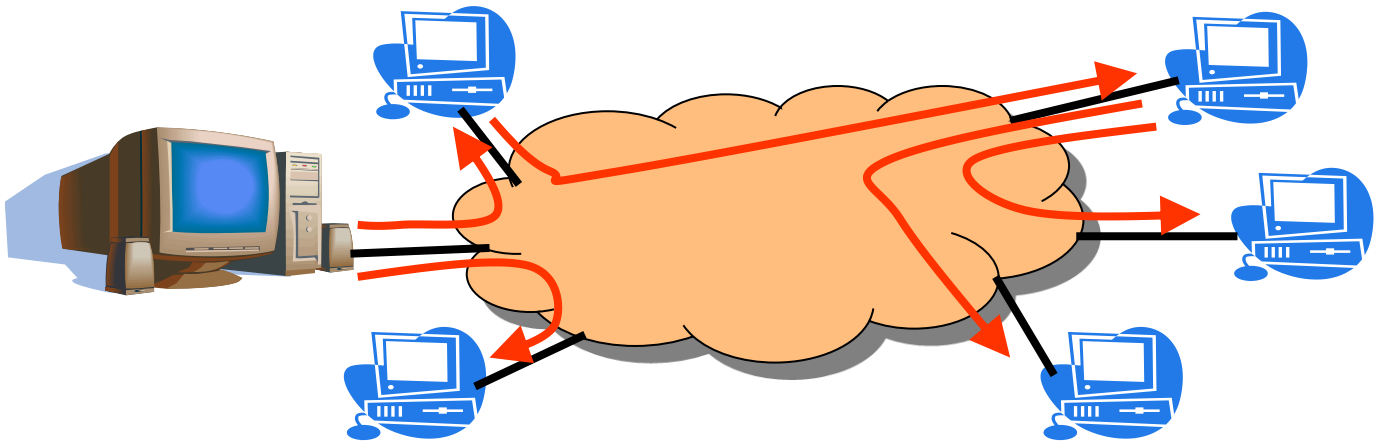
P2P

- In network multicast, or CDN, content follow a tree to reach all receivers
  - Does this work for P2P?
- The secret of P2P networks is
  - Divide content into multiple "chunks" or "stripes"
  - Use different trees to distribute different chunks or stripes
  - Every node serves as interior node in some tree
- Next
  - End-system multicast
  - Multiple tree construction

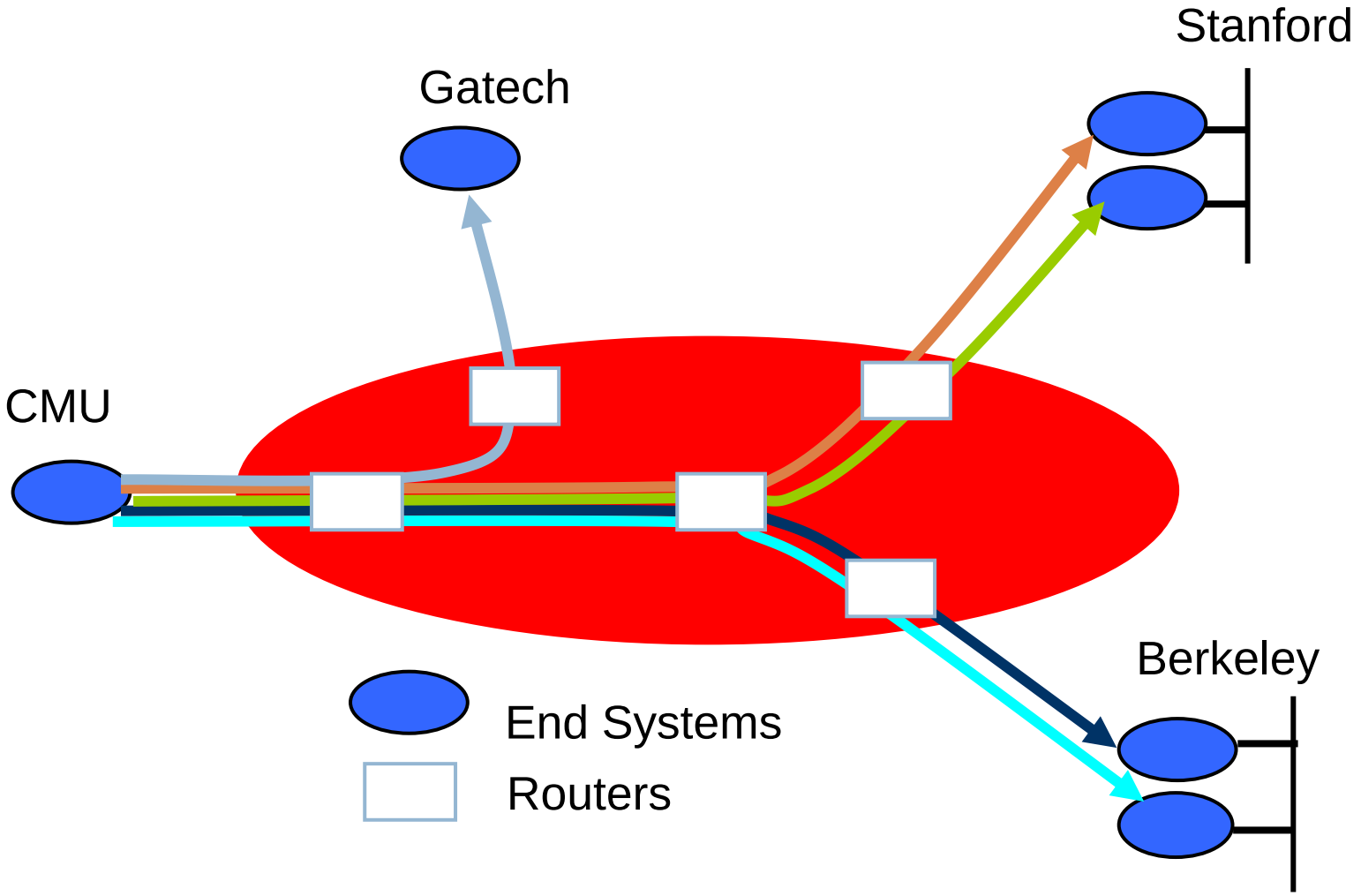


End-System Multicast

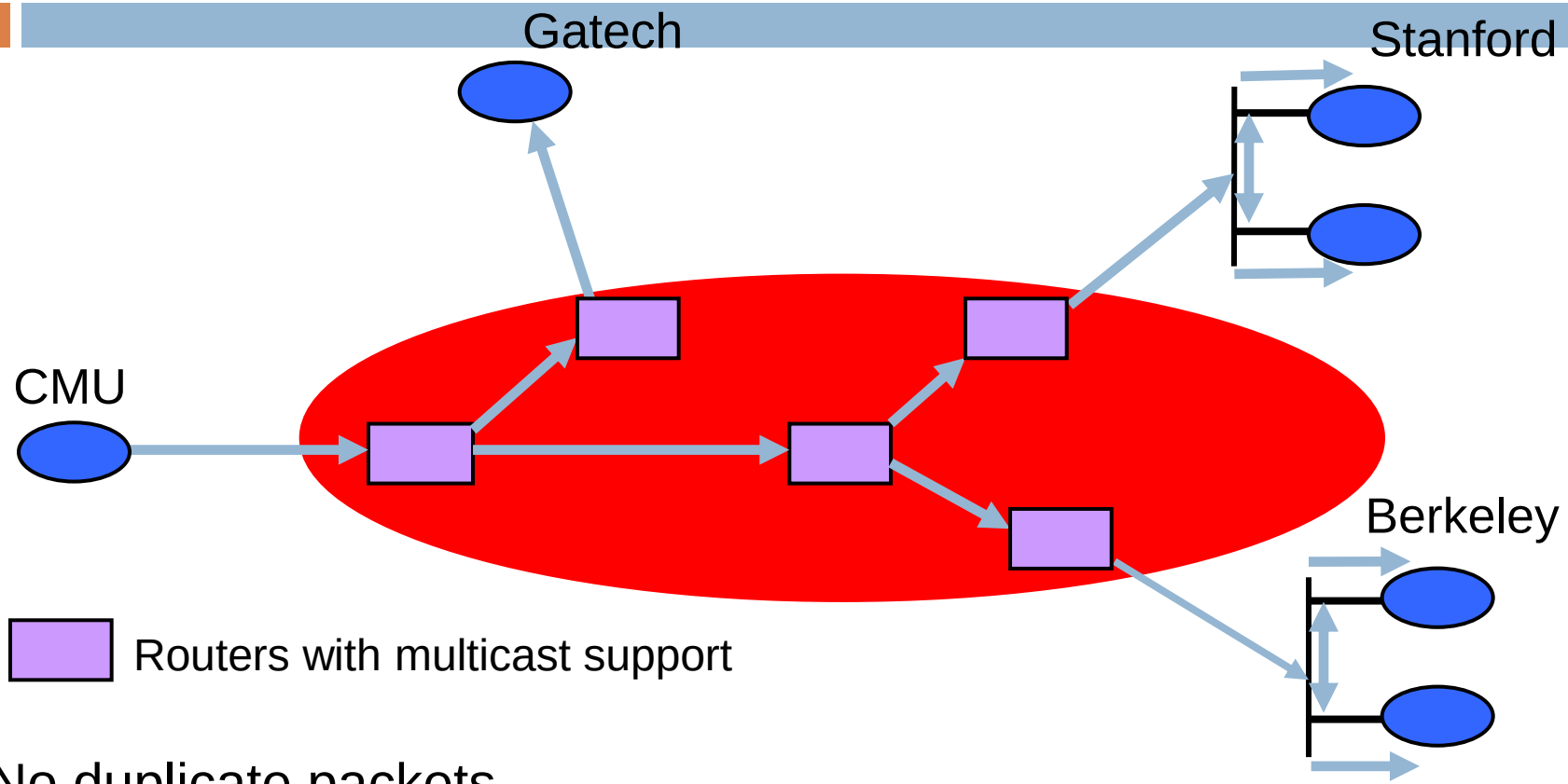
- IP multicast still is not widely deployed
  - Technical and business challenges
  - Should multicast be a *network*-layer service?
- Multicast tree of end hosts
  - Allow end hosts to form their own multicast tree
  - Hosts receiving the data help forward to others



Unicast Emulation of Multicast



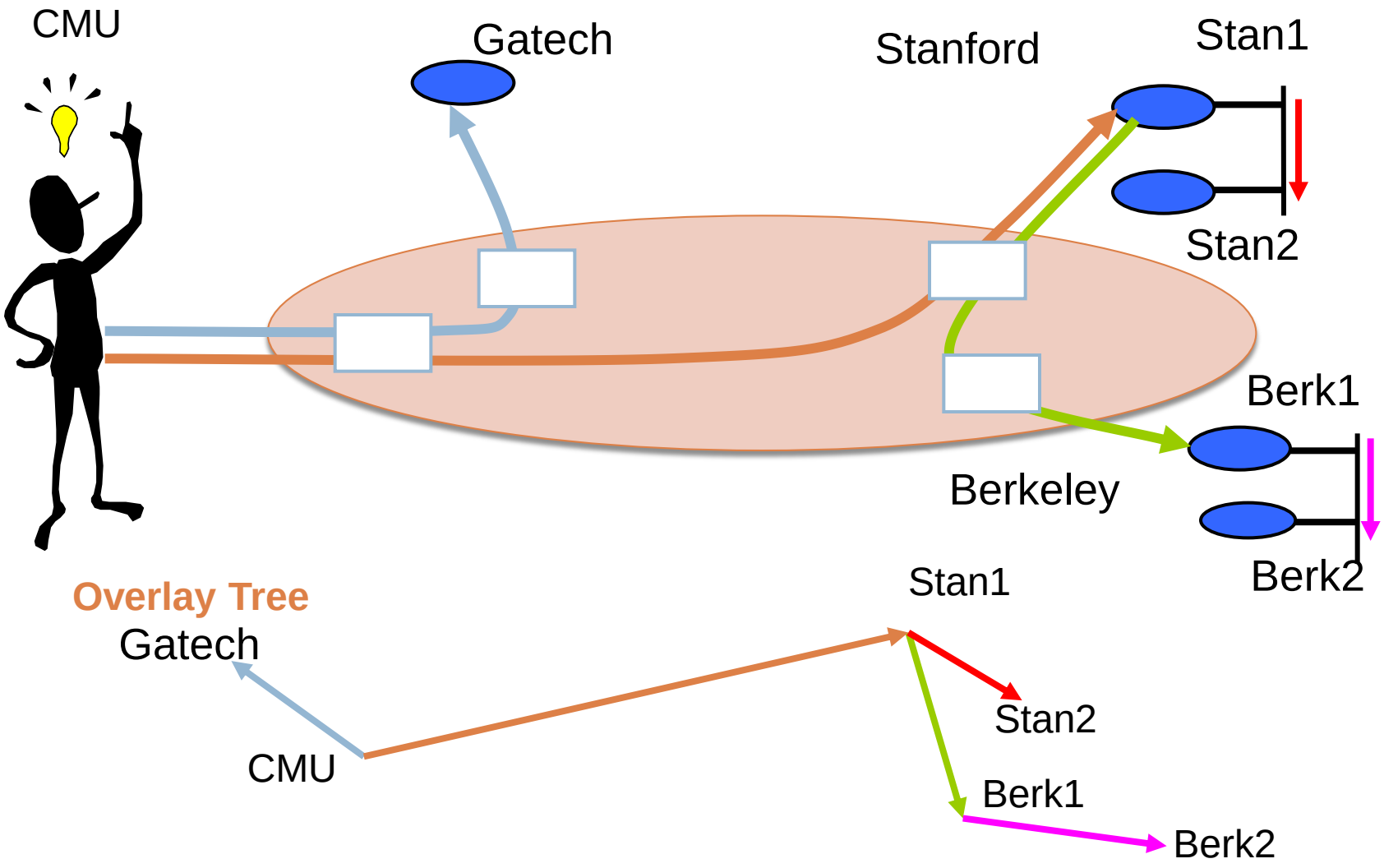
IP Multicast



- No duplicate packets
- Highly efficient bandwidth usage

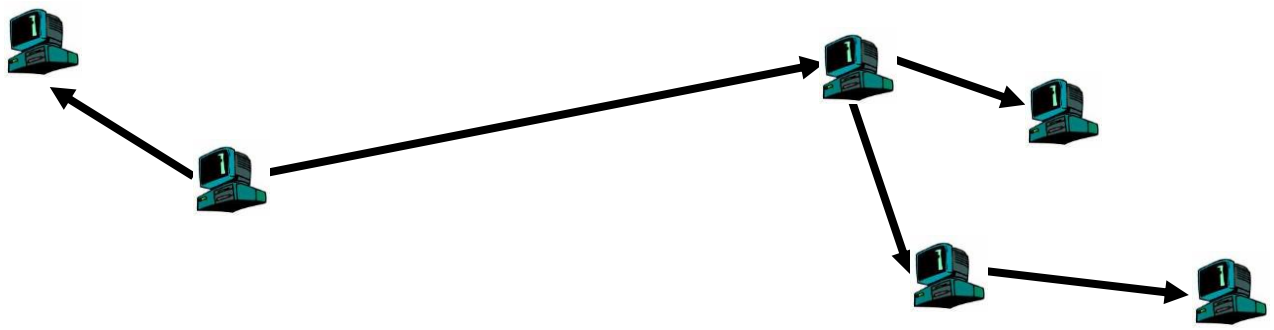
Key Architectural Decision: Add support for multicast in IP layer

End System Multicast



Potential Benefits

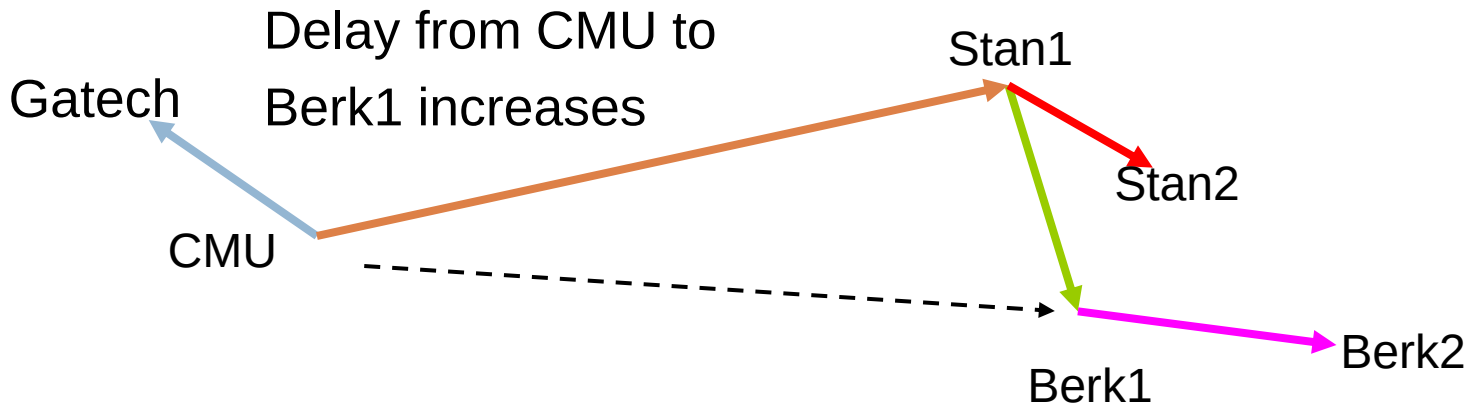
- Scalability
  - Routers do not maintain per-group state
  - End systems do, but they participate in very few groups
- Easier to deploy
- Potentially simplifies support for higher level functionality
  - Leverage computation and storage of end systems
  - For example, for buffering packets, transcoding, ACK aggregation
  - Leverage solutions for unicast congestion control and reliability

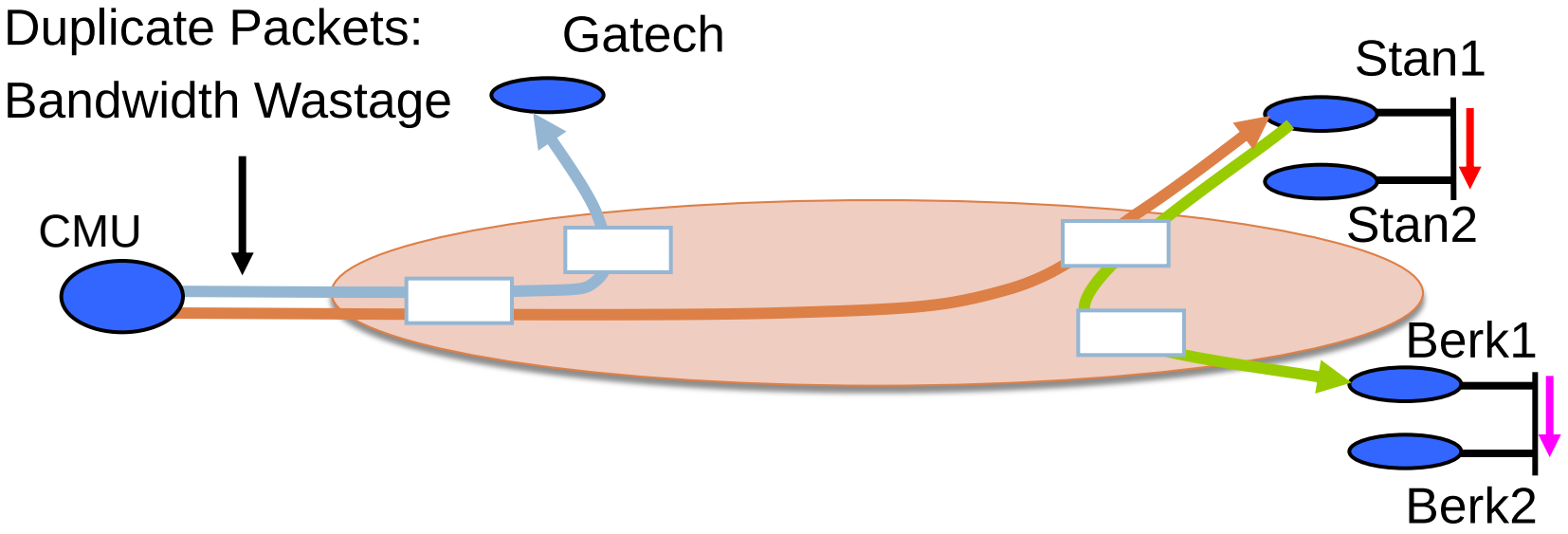


Design Questions

- Is End System Multicast Feasible?
- Target applications with small and sparse groups
- How to Build Efficient Application-Layer Multicast "Tree" or Overlay Network?
  - Narada: A distributed protocol for constructing efficient overlay trees among end systems
  - Simulation and Internet evaluation results to demonstrate that Narada can achieve good performance

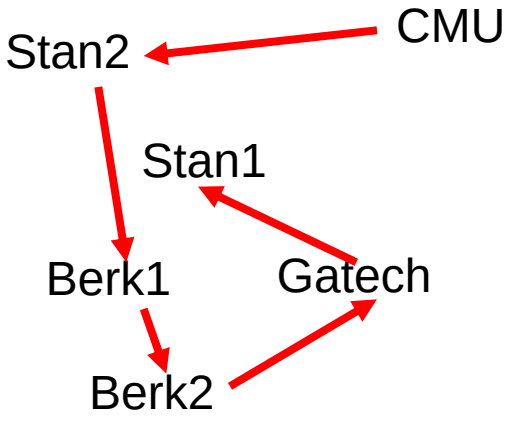
Performance Concerns



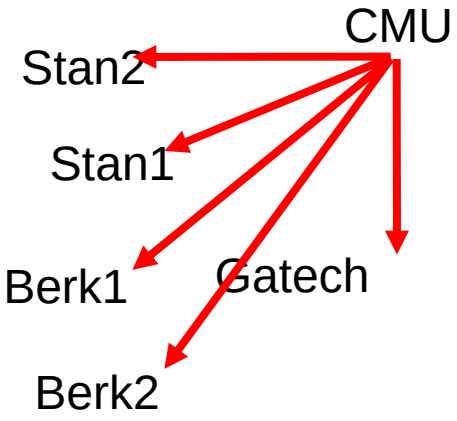


What is an Efficient Overlay Tree?

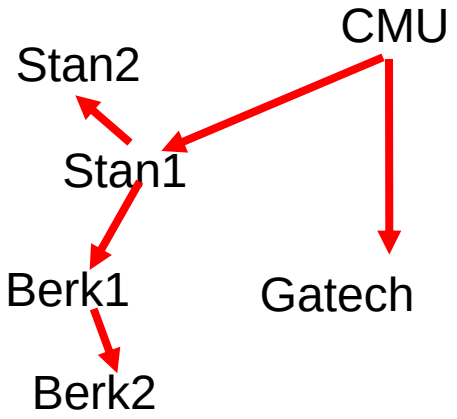
- The delay between the source and receivers is small
- Ideally,
  - The number of redundant packets on any physical link is low
- Heuristic used:
  - Every member in the tree has a small degree
  - Degree chosen to reflect bandwidth of connection to Internet



High latency



High degree (unicast)



"Efficient" overlay

Why is Self-Organization Hard?

- Dynamic changes in group membership
  - Members may join and leave dynamically
  - Members may die
- Limited knowledge of network conditions
  - Members do not know delay to each other when they join
  - Members probe each other to learn network related information
  - Overlay must self-improve as more information available
- Dynamic changes in network conditions
  - Delay between members may vary over time due to congestion

Narada Design

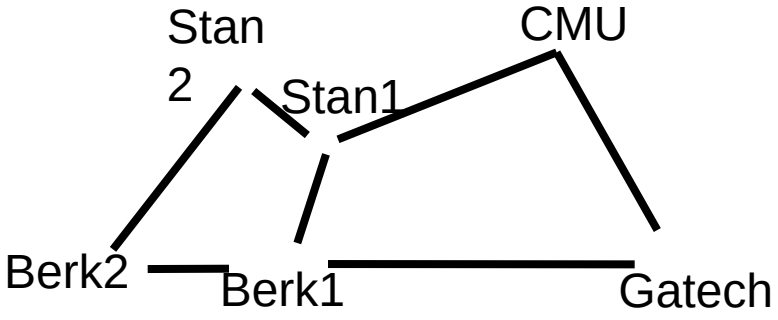
Step 1

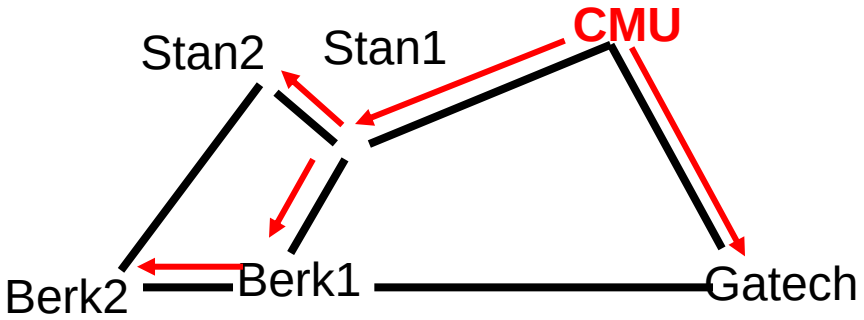
"Mesh": Richer overlay that may have cycles and includes all group members

- Members have low degrees
- Shortest path delay between any pair of members along mesh is small

Step 2

- Source rooted shortest delay spanning trees of mesh
- Constructed using well known routing algorithms
  - Members have low degrees
  - Small delay from source to receivers

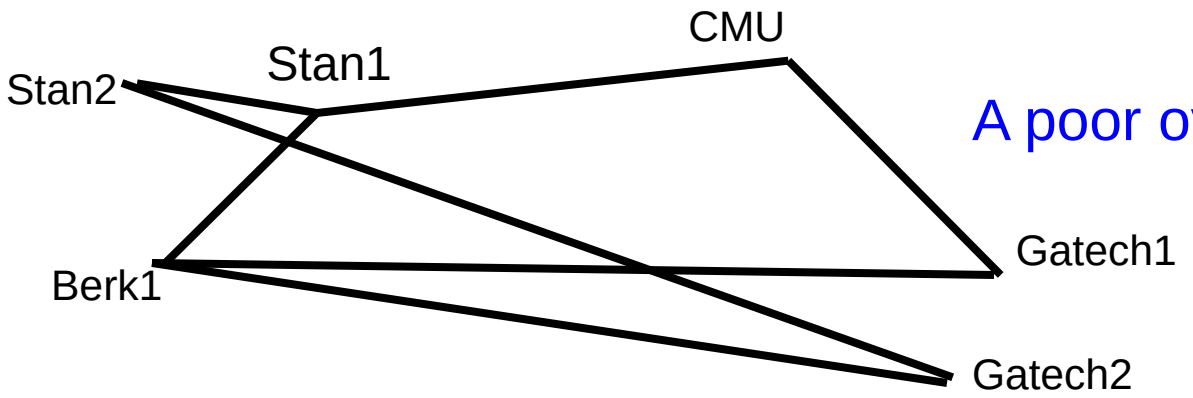




Narada Components

- Mesh Management:
  - Ensures mesh remains connected in face of membership changes
- Mesh Optimization:
  - Distributed heuristics for ensuring shortest path delay between members along the mesh is small
- Spanning tree construction:
  - Routing algorithms for constructing data-delivery trees
  - Distance vector routing, and reverse path forwarding

Optimizing Mesh Quality



A poor overlay topology

- Members periodically probe other members at random
- New Link added if

  Utility Gain of adding link > Add Threshold
- Members periodically monitor existing links
- Existing Link dropped if

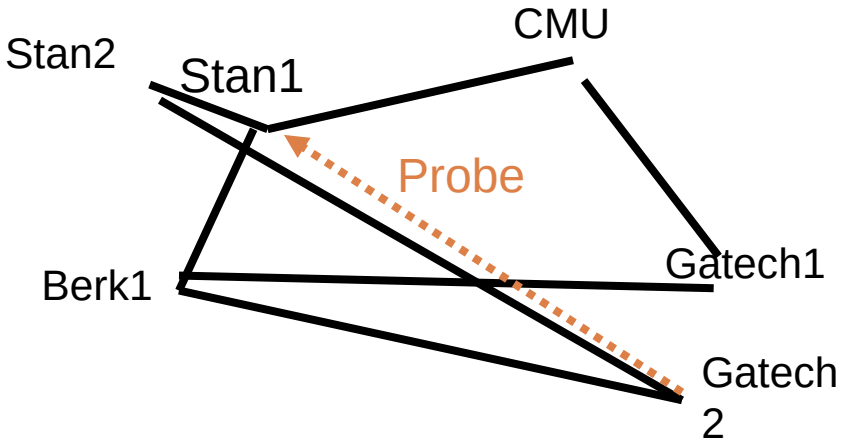
  Cost of dropping link < Drop Threshold

The Terms Defined

- Utility gain of adding a link based on
  - The number of members to which routing delay improves
  - How significant the improvement in delay to each member is
- Cost of dropping a link based on
  - The number of members to which routing delay increases, for either neighbor
- Add/Drop Thresholds are functions of:
  - Member's estimation of group size
  - Current and maximum degree of member in the mesh

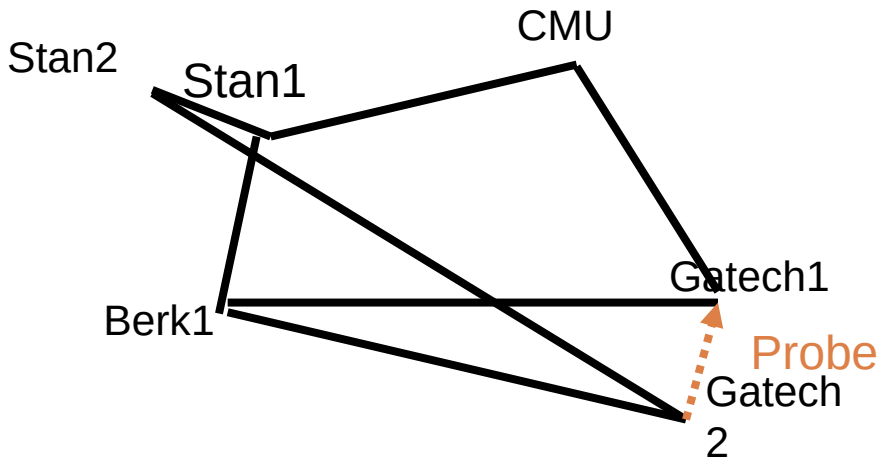
Desirable Properties of Heuristics

- Stability: A dropped link will not be immediately re-added
- Partition Avoidance: A partition of the mesh is unlikely to be caused as a result of any single link being dropped



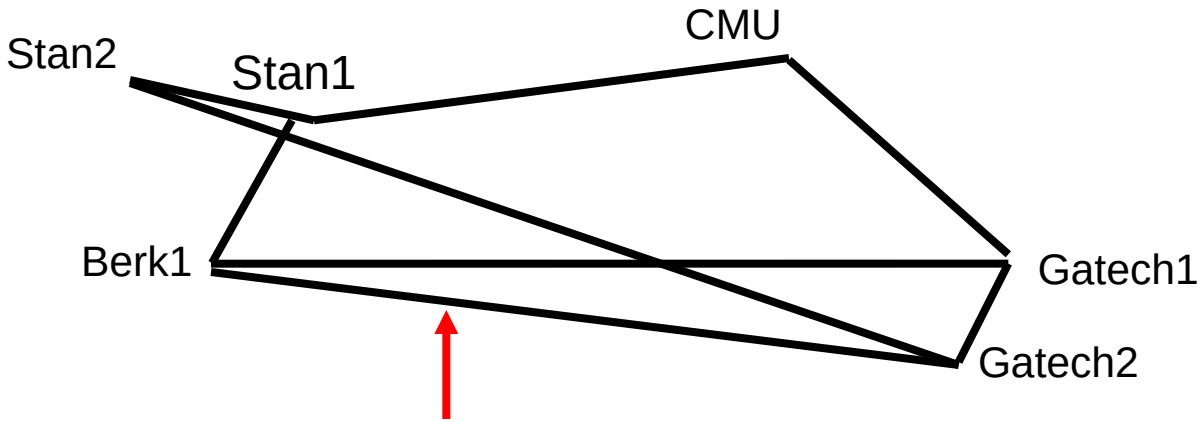
Delay improves to Stan1, CMU but marginally.

Do not add link!



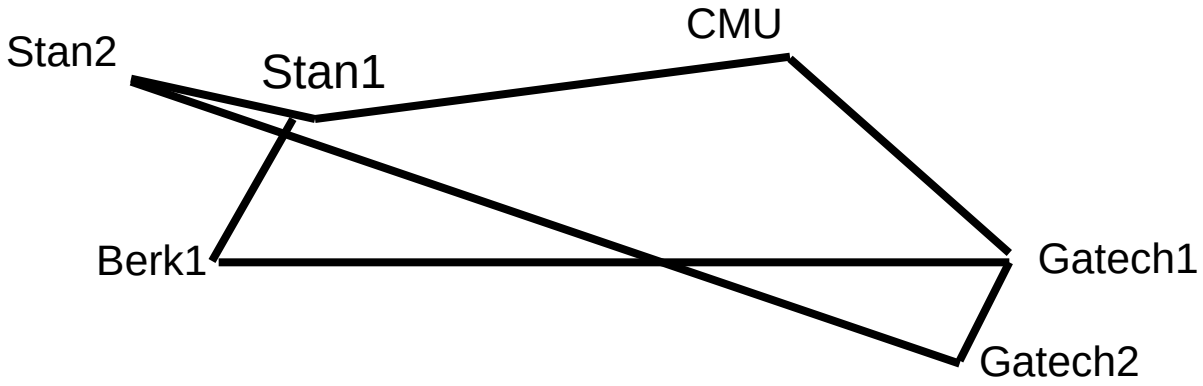
Delay improves to CMU, Gatech1 and significantly.

Add link!



Used by Berk1 to reach only Gatech2 and vice versa.

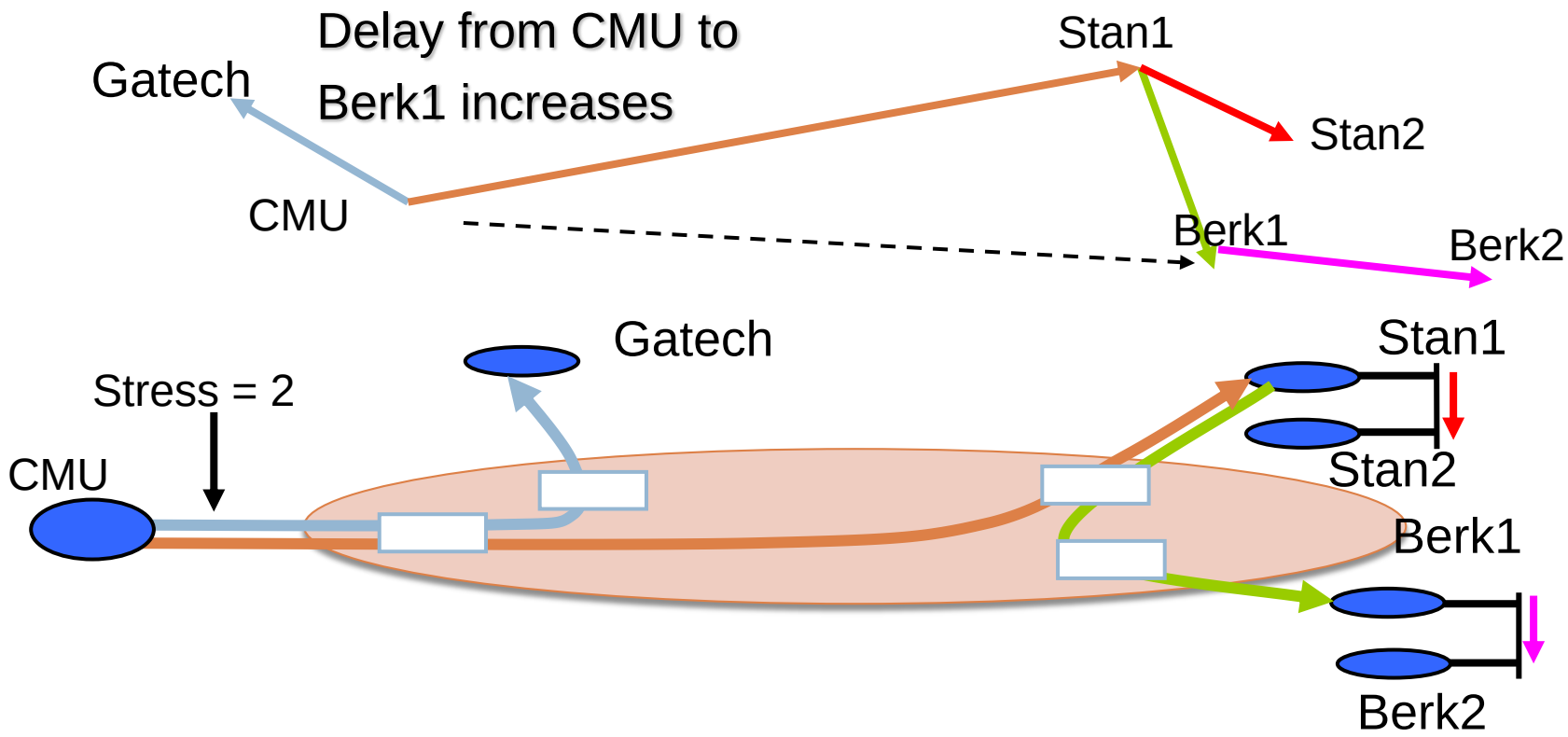
Drop!!



An improved mesh !!

Performance Metrics

- Delay between members using Narada
- Stress, defined as the number of identical copies of a packet that traverse a physical link



Factors Affecting Performance

- Topology Model
  - Waxman Variant
  - Mapnet: Connectivity modeled after several ISP backbones
  - ASMap: Based on inter-domain Internet connectivity
- Topology Size
  - Between 64 and 1024 routers
- Group Size
  - Between 16 and 256
- Fanout range
  - Number of neighbors each member tries to maintain in the mesh

ESM Conclusions

- Proposed in 1989, IP Multicast is not yet widely deployed
  - Per-group state, control state complexity and scaling concerns
  - Difficult to support higher layer functionality
  - Difficult to deploy, and get ISP's to turn on IP Multicast
- Is IP the right layer for supporting multicast functionality?
- For small-sized groups, an end-system overlay approach
  - is feasible
  - has a low performance penalty compared to IP Multicast
  - has the potential to simplify support for higher layer functionality
  - allows for application-specific customizations

The Construction of Multiple Trees

- Two approaches:
  - Tree-based, or push: build multiple trees **explicitly**
    - "Splitstream: High-bandwidth Multicast in Cooperative Environments", SOSP, 2003 (2025+ citations)
  - Data-driven, or pull: build multiple trees **dynamically** and **implicitly**
    - "Incentives Build Robustness in BitTorrent", 2003 (3734+ citations)
  - Combining the two approaches also possible
    - "Understanding the power of pull-based streaming protocol: Can we do better?", *IEEE JSAC*, 2007.

The Construction of Multiple Trees

- Tree-based, or push: build multiple trees explicitly
- “Splitstream: High-bandwidth Multicast in Cooperative Environments”, SOSP, 2003 (2025+ citations)

The SplitStream Approach

- Why SplitStream? The Problem:
  - Conventional tree-based multicast is not well-suited for cooperative P2P systems
  - The burden is carried by interior nodes (Small number of interior nodes bear forwarding burden): not expected in P2P systems
  - Most peers do not have enough capacity
  - Poor fault-tolerance – if one node fails, some nodes receive none of original content
  - Poor scalability
- The SplitStream Solution
  - Split the content!

SplitStream Solutions

- *Split* the original content into *k* stripes and multicast each stripe in a separate tree
- Nodes join trees of stripes they want to receive and specify upper-bound on number of children they will accept
- Solution has 2 main goals:
  1. Forest of trees is interior-node-disjoint
  2. Forest must satisfy node bandwidth constraints

Solutions (continued)

- Forwarding load is now distributed
- System more fault-tolerant (applications using SplitStream can use data encodings to reconstruct content from less than *k* stripes)
- Enhanced scalability

SplitStream Properties

- General condition for forest construction:
  - It should be possible to connect nodes such that each node receives no more stripes than its desired in degree and its out degree is no more than its capacity
  - Necessary Condition

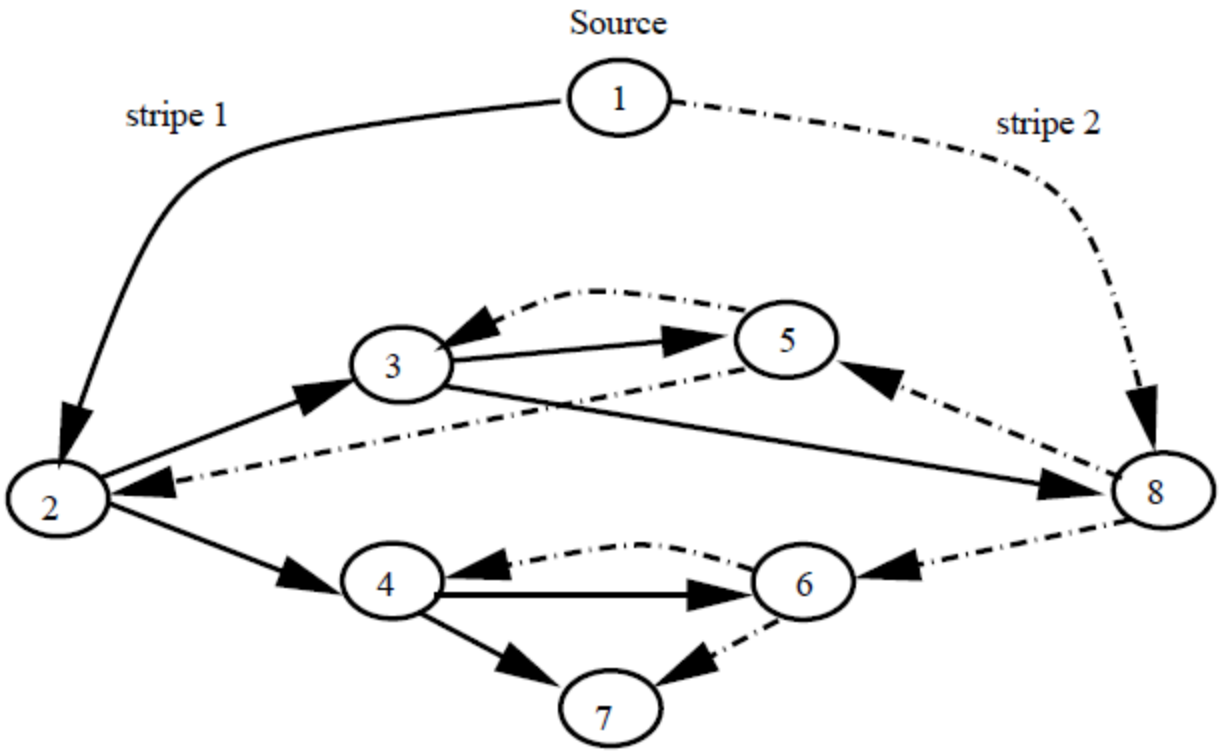
$$\sum_{\forall i \in N} I_i \leq \sum_{\forall i \in N} C_i$$

  - Sufficient Condition: if node can forward more than it wants to receive, it must receive (or originate, if source) all k stripes:

$$\forall i : C_i > I_i \Rightarrow I_i + T_i = k.$$

SplitStream Properties

- A simple example:



Detailed Design

- Building interior-node-disjoint trees
  - Each node in a set of trees is interior node in at most one tree and leaf node in the other trees
- Limiting Node Degree
  - Inbound bandwidth satisfied by joining trees of desired stripes
  - Main objective (for satisfying outbound bandwidth): handle the case where a node that has reached its maximum outdegree receives a join request from a prospective child:
    - The node looks for children in stripes that do not share a prefix
    - Selects a child to drop
    - The orphaned child will look for a new parent

The Construction of Multiple Trees

- Data-driven, or pull: build multiple trees **dynamically** and **implicitly**
  - "Incentives Build Robustness in BitTorrent", 2003 (3734+ citations)

BitTorrent

- BitTorrent build and maintain multiple trees implicitly and dynamically, using a “data-driven” approach
- BitTorrent has built-in incentives mechanisms so that peers have incentive to contribute more uploading bandwidth
  - By contributing more uploading bandwidth, a peer can finish downloading earlier

Components of BT

- A Web server
- The .torrent file
- A Tracker
- Peers

Web Server

- Content (file) discovery is handled outside of BT, using a web server
- To provide the “metainfo” file by HTTP
- For example: http://bt.btchina.net
- The information about each movie, or other content is stored in a metafile, such as “supergirl.torrent”

The .torrent File

- Static file storing necessary meta information :
  - Name
  - Size
  - Checksum
    - The content is divided into many “chunks” (e.g. 1/4 megabyte each)
    - Each chunk is hashed to a checksum value
    - When a peer later get the chunks (from other peers), it can check the authenticity by comparing the checksum
  - IP address and port of the **Tracker**

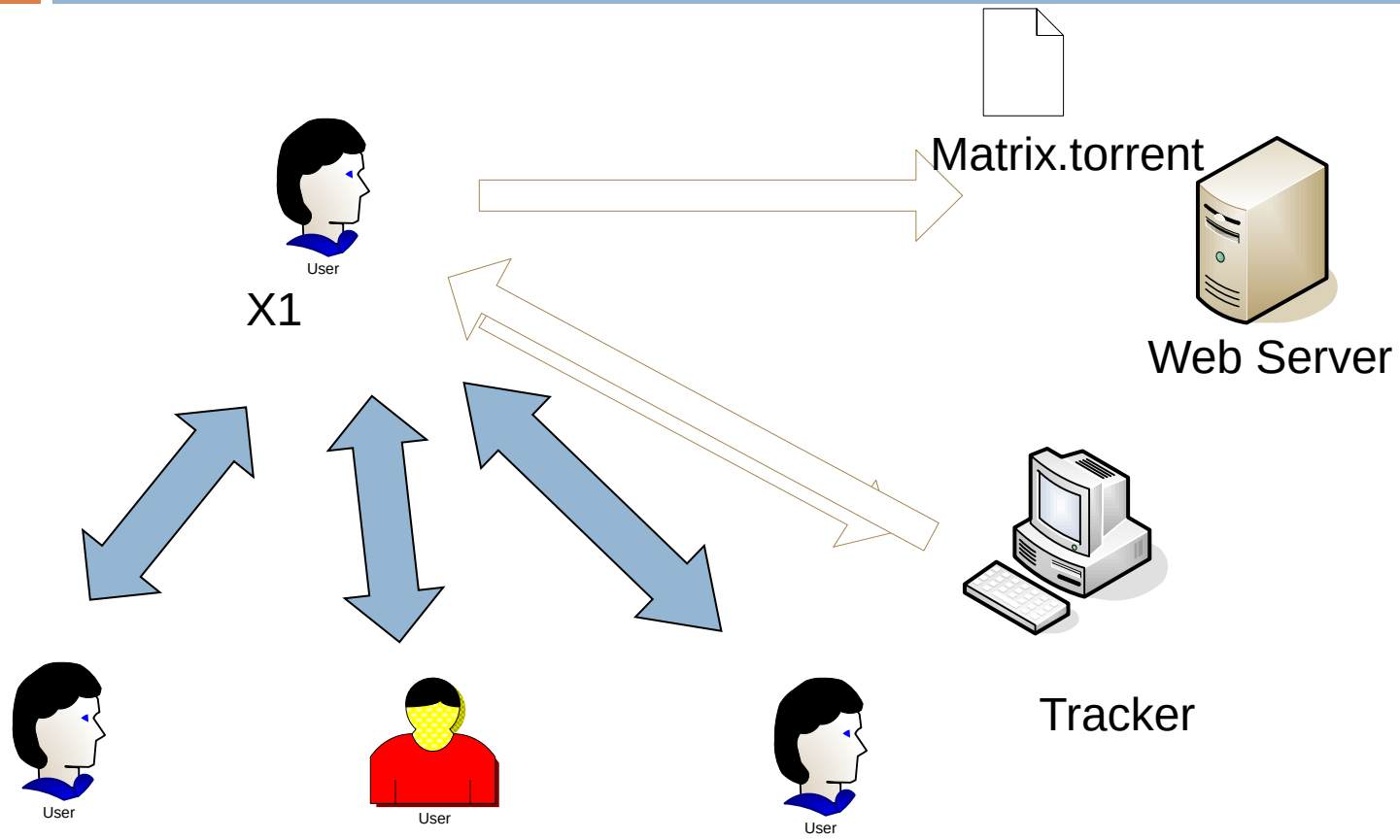
Tracker

- Non-content-sharing node
- Tracks peers
- For example:
  - http://bt.cnxp.com:8080/announce

Peers

- Users who want to use BitTorrent must install client software or plug-in for web browsers.
- Two types of peers:
  - *Downloader* (*leecher*) : A peer who has only ***a part (or none)*** of the file.
  - *Seeder*: A peer who has the ***complete*** file, and chooses to stay in the system to allow other peers to download

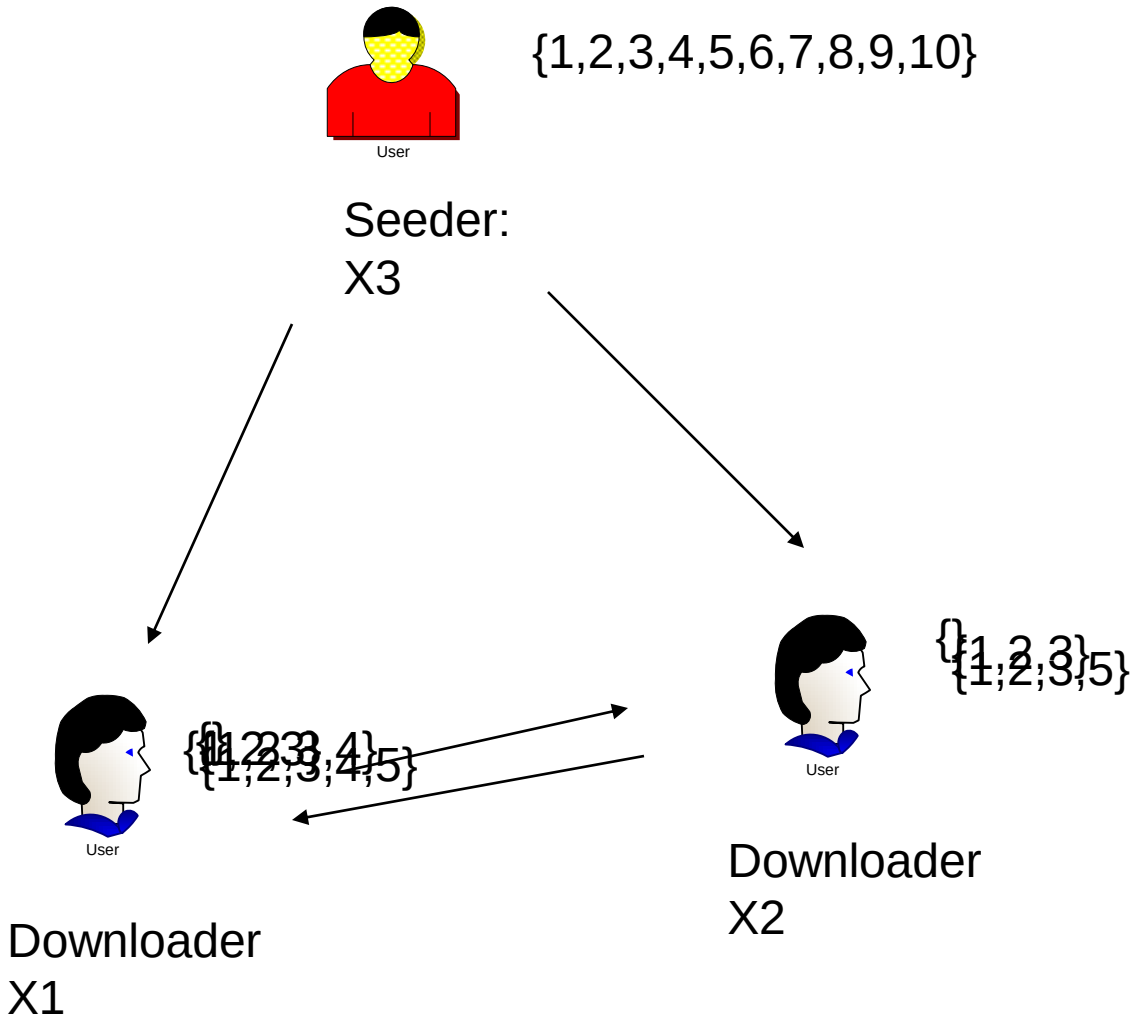
The Process


Matrix.torrent
User
X1
Web Server
Tracker
User
User
User
Downloader:
X2
Seeder:
X3
Downloader:
X4

Chunks

- A file is split into chunks of fixed size, typically 256Kb
- Each peer maintains a bitmap that indicate which chunks it has
- Each peer reports to all of its neighboring peers (obtained from tracker) what chunks it has
  - This is the information used to build the implicit trees

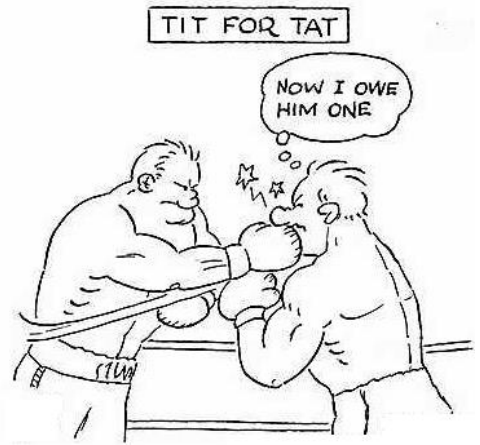
A Simple Example



Peer Selection

The basic idea of Tit-for-tat in BT:

- Maintain 4-5 "friends" to exchange chunks with
- If a friend is not exchanging enough chunks, get rid of him/her
  - known as "choking" in BT
- Periodically, randomly select one new friend
  - known as "optimistic unchoking" in BT
- If you have no friends, randomly select several new friends
  - known as "anti-snubbing" in BT



Choking Algorithm

- Choking is a temporary refusal to upload
- Choking evaluation is performed every 10 seconds.
- Each peer keeps a fixed number of "friends" (default = 4)
- The decision on which peers to maintain as "friend" is based solely on download rate, which is evaluated on a rolling, 20-second average

Optimistic Unchoking

- Each peer maintains a single '*optimistic unchoke*' friend
  - You upload to this special friend regardless of the current download rate from it. This friend is rotated every 30s

- Reason:
  - To discover new friends who have better uploading bandwidth than the current friends

Anti-Snubbing

- When a peer received no data in 60s, we assume it is choked by all other peers
- Break off from these peers and try more optimistic unchoking.
- This peer needs friends!

Chunk Selection Strategies

- Strict Priority

  Once you selected a chunk, try to get all segments of that chunk

- **Rarest First**

  During most of the download, always try to get the **rarest** chunk

  - Very important for scalability – will become clear in analysis

- Starting mode

  At the beginning, just try to get a (few) random chunks

- Endgame Mode

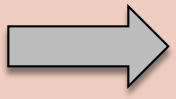
  Near the end, try hard to finish, ask all peers for chunks

  - The "last piece problem" – will become clear in analysis

Chunk Selection Strategies

- Str

A BT peer using this strategy selects the requesting chunk which has the smallest number of its neighbors



This mechanism results typically in an evenly spread number of sharing peers for all chunks of the file

- **Rarest First**

  During most of the download, always try to get the **rarest** chunk

  - Very important for scalability – will become clear in analysis

- Starting mode

  At the beginning, just try to get a (few) random chunks

- Endgame Mode

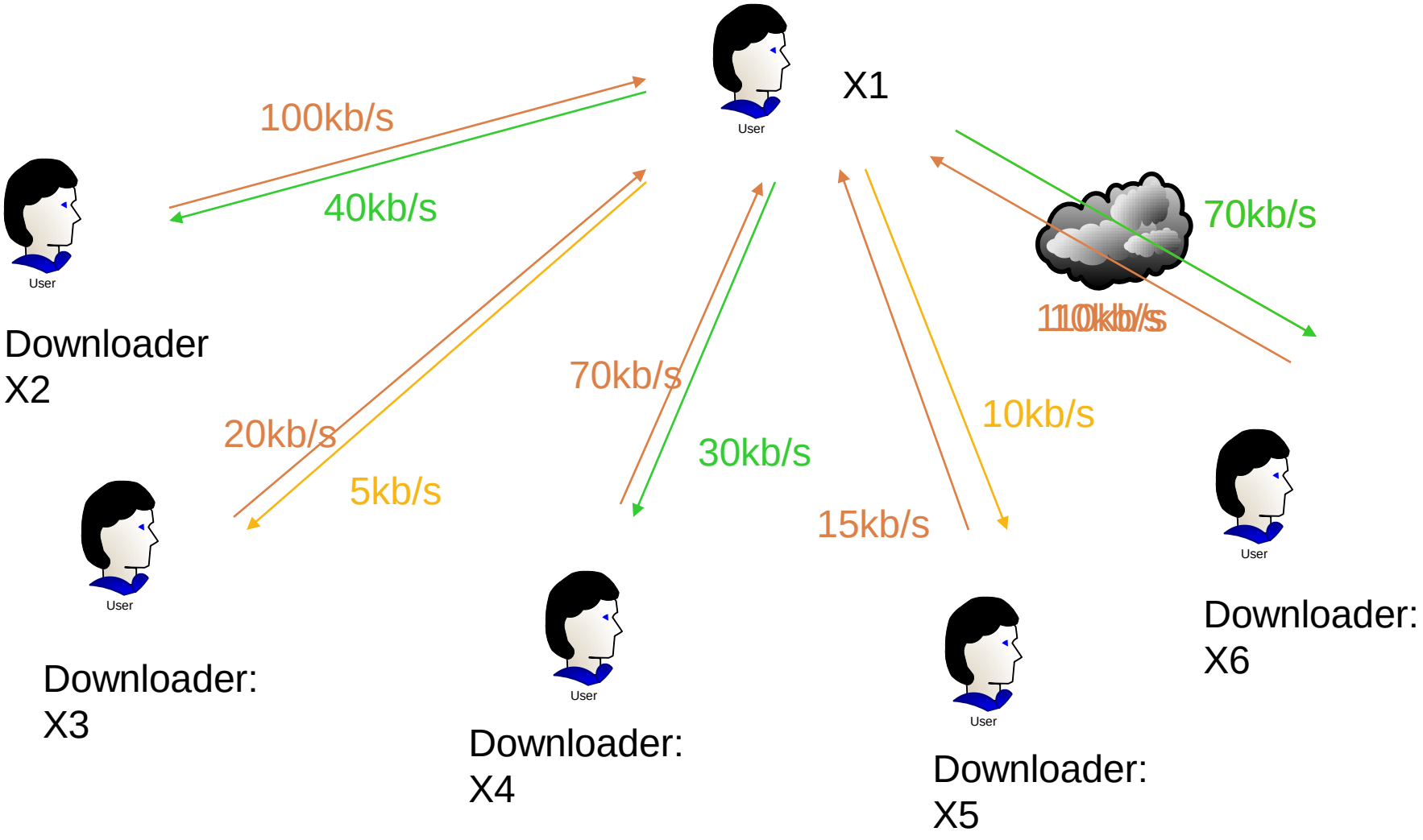
  Near the end, try hard to finish, ask all peers for chunks

  - The "last piece problem" – will become clear in analysis

If block propagation does not happen efficiently → some rare blocks only exist in few nodes

Node dynamic (server and the above mentioned few nodes leave) → then no node can finish the download

Example


User
X1
100kb/s
40kb/s
70kb/s
User
Downloader
X2
20kb/s
5kb/s
70kb/s
30kb/s
10kb/s
15kb/s
User
Downloader:
X3
User
Downloader:
X4
User
Downloader:
X5
User
Downloader:
X6

Dynamic Tree Forming Process

- In a BT session, the peers are dynamically adapting the trees
  - Network conditions change
  - Peers arrive and leave
  - Each peer can continuously search for friends to help him/her finish downloading earlier
- In BT, even if all peers arrive at the same time, not all of them will finish at the same time
  - Does tit-for-tat work for p2p streaming?

Interesting Issues

- BT already works in practice, but it is interesting to study it via modeling and analysis
  - What is its scalability? Given a set of peers of different upload capacity, what is the system throughput?
  - What is the result of using different sets of trees, in terms of average throughput, and "fairness" (the more one contributes the more one receives)?
  - The peer selection and chunk selection strategies take some effort. Can we do as well using random strategies?
  - Why does it take longer to finish the last few chunks?

Advanced Topics in P2P Networking

P2P Streaming Protocols and Analysis

Coolstreaming

XY Zhang, JC Liu, B Li, P Yum, "Coolstreaming/DONet: a data-driven overlay network for efficient live media streaming", Infocom 2005 (2120+ citations)

DONet – Data-driven Overlay Netwrok

Coolstreaming – Cooperative Overlay Streaming

Background:

- XY Zhang was a CUHK/HKUST student
- Built a prototype p2p streaming system, using Python (2000 lines)
- Experimented on PlanetLab
- "Deployed" in May 30, 2004 during European soccer finals;
- Total 30000 downloads, 4000 simultaneous users
- First **large scale** p2p streaming experiment - showed **feasibility**

Application Layer Multicast

- Tree: NICE, CoopNet, SpreadIt, ZIGZAG
- Mesh: Narada and its extension
- Multi-tree: SplitStream
- Issues
  - Structure construction
  - Node dynamics
  - Structure maintenance
    - Passive/proactive repairing algorithms

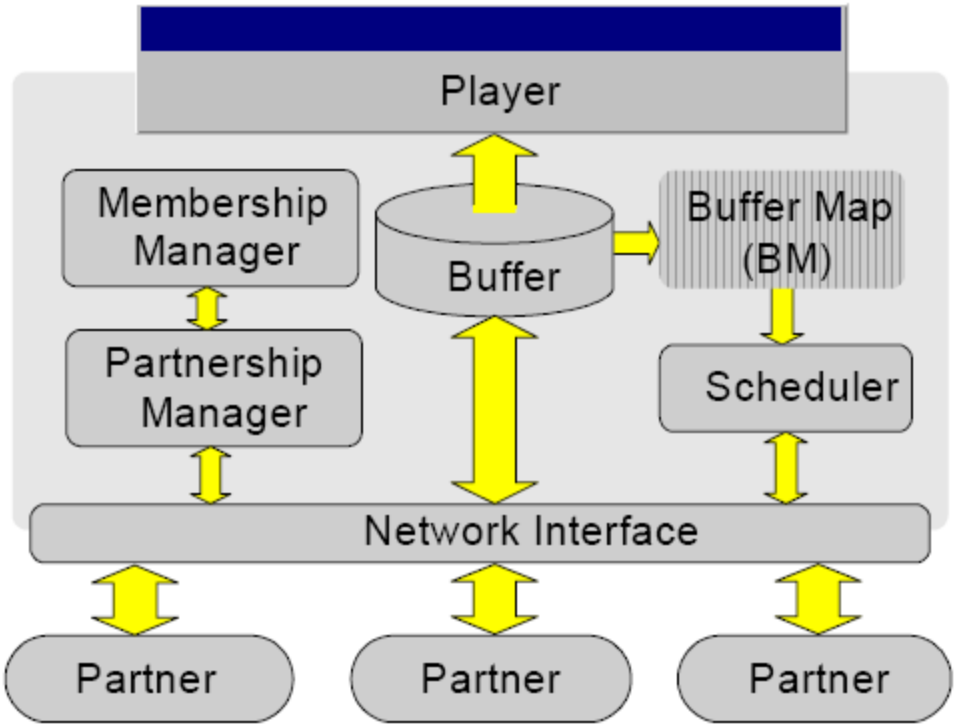
Gossip-based Dissemination

- Gossip
  - Iteration
    - Sends a new message to a random set of nodes
    - Each node does similarly in the next round
- Pros: Simple, robust
- Cons: Redundancy, delay
- Related
  - Peer-to-peer on-demand streaming

Data-driven Overlay (DONet)

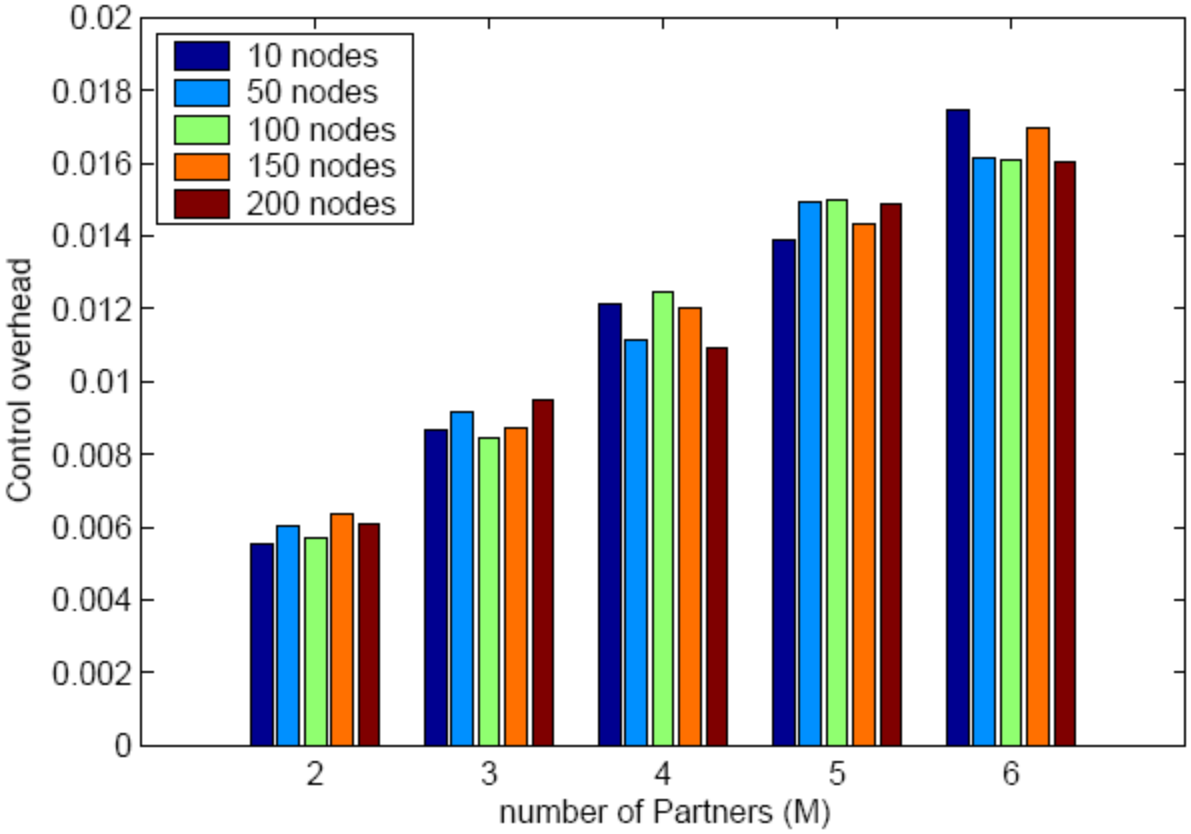
- Target
  - Live media broadcasting while no IP multicast support
- Core operations
  - Every node periodically exchanges data availability information with a set of partners
  - Then retrieves unavailable data from one or more partners, or supplies available data to partners
- Feature of DONet
  - *Easy to implement*: no need to construct and maintain a complex global structure
  - *Efficient*: data forwarding is dynamically determined according to data availability, not restricted by specific directions
  - *Robust and resilient:* adaptive and quick switching among multi-suppliers

System Diagram of Coolstreaming Software



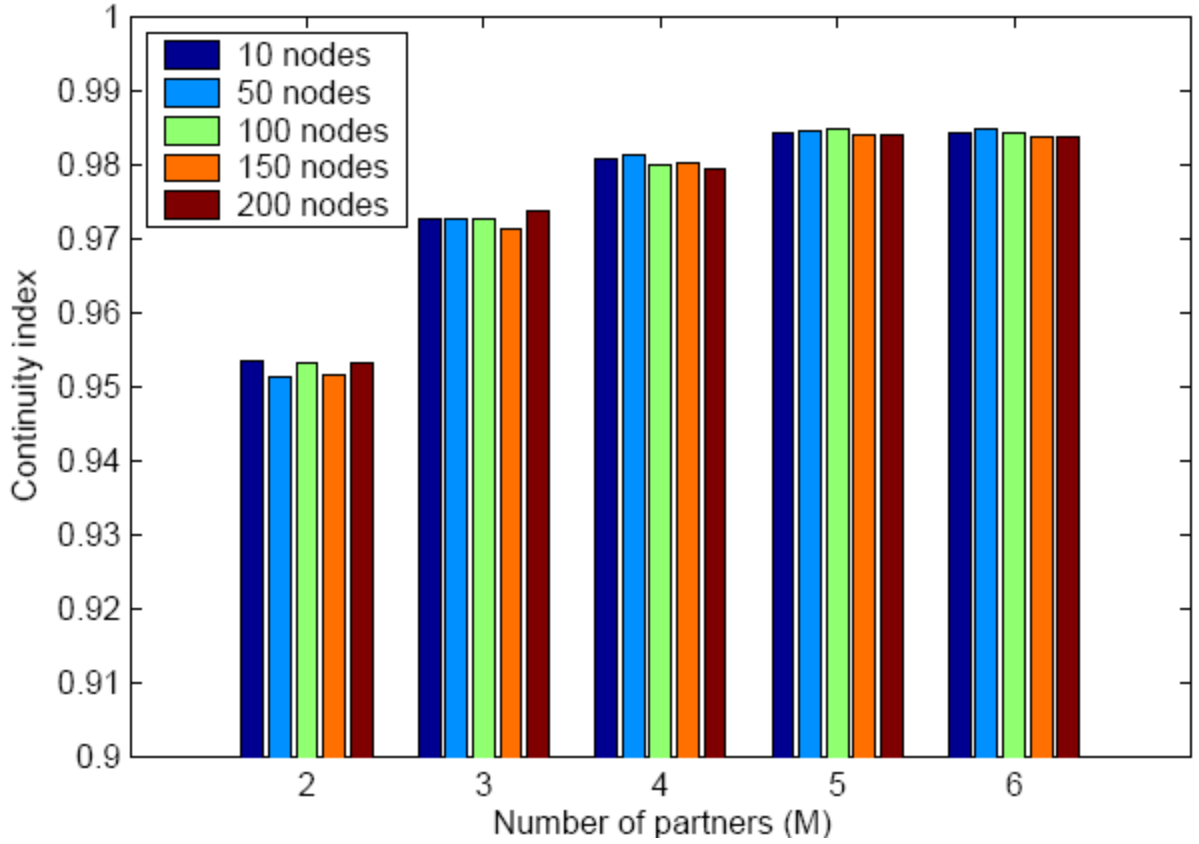
A generic system diagram for a DONet node.

Low Overhead



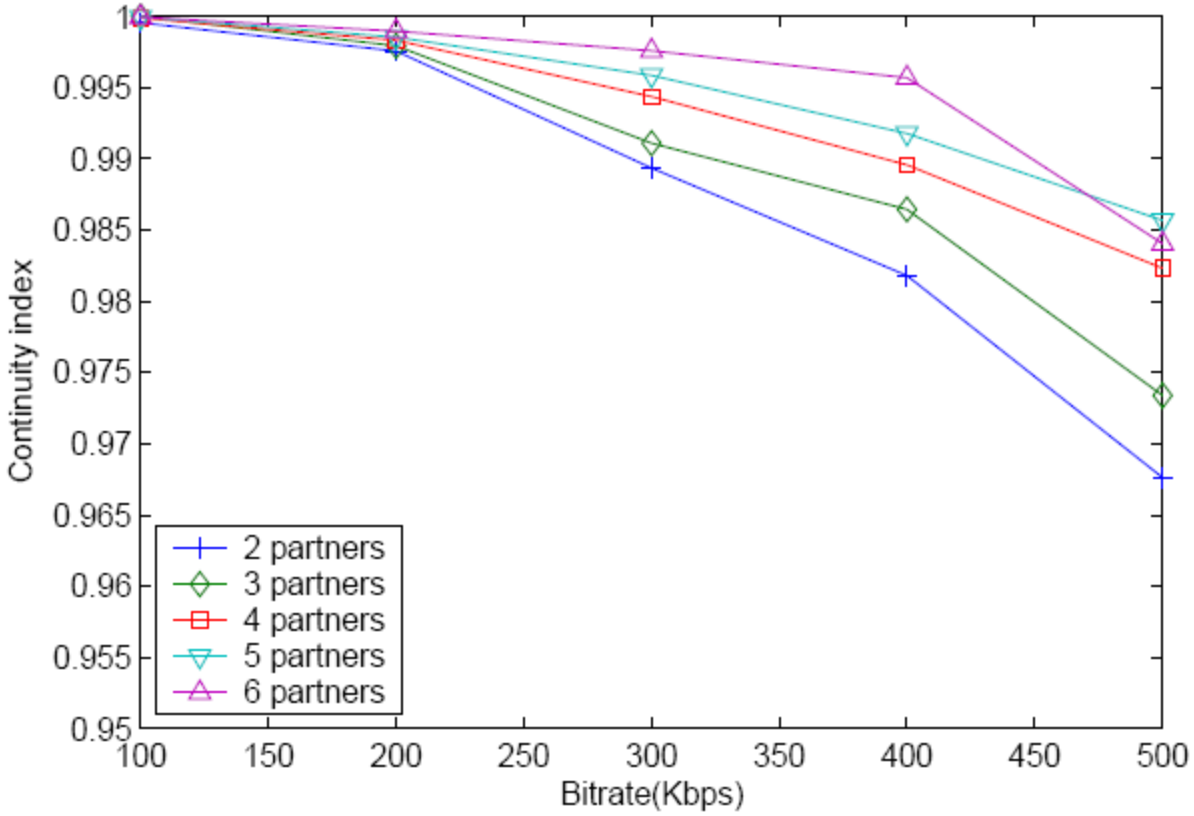
Control overhead as a function of the number of partners for different overlay sizes. (Control overhead= Control traffic volume/ Video traffic volume at each node).

Continuity - Playback Performance



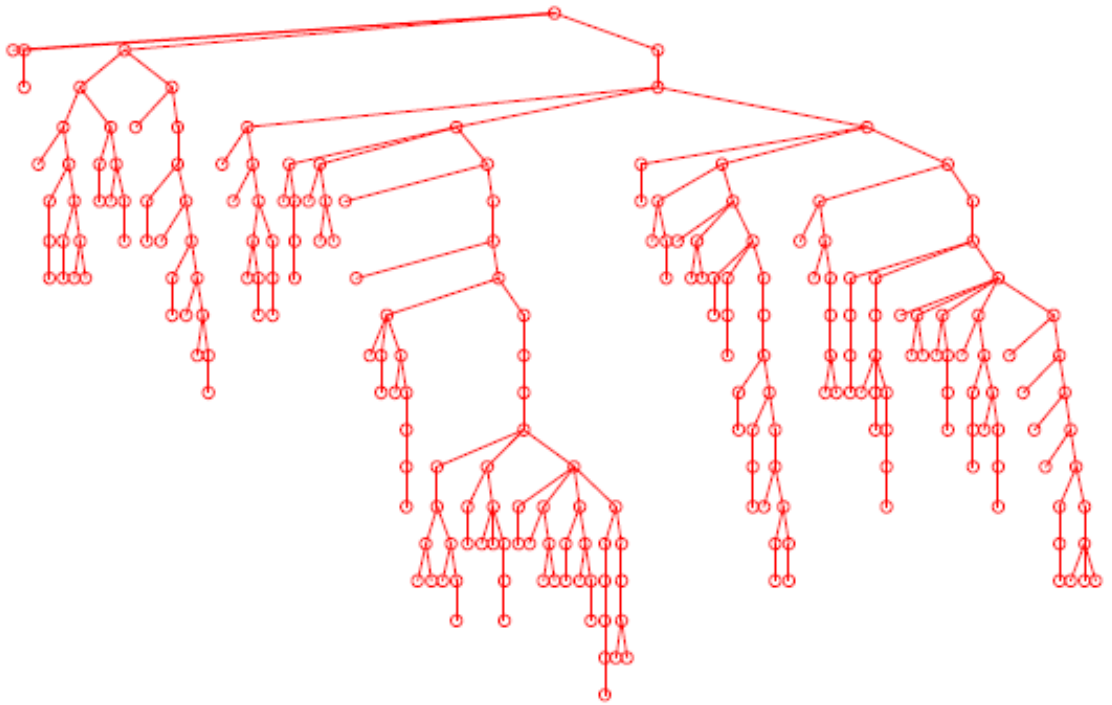
Continuity index as a function of the number of partners.

Performance for Different Streaming Rates



Continuity index as a function of the streaming rate.
Overlay size = 200 nodes.

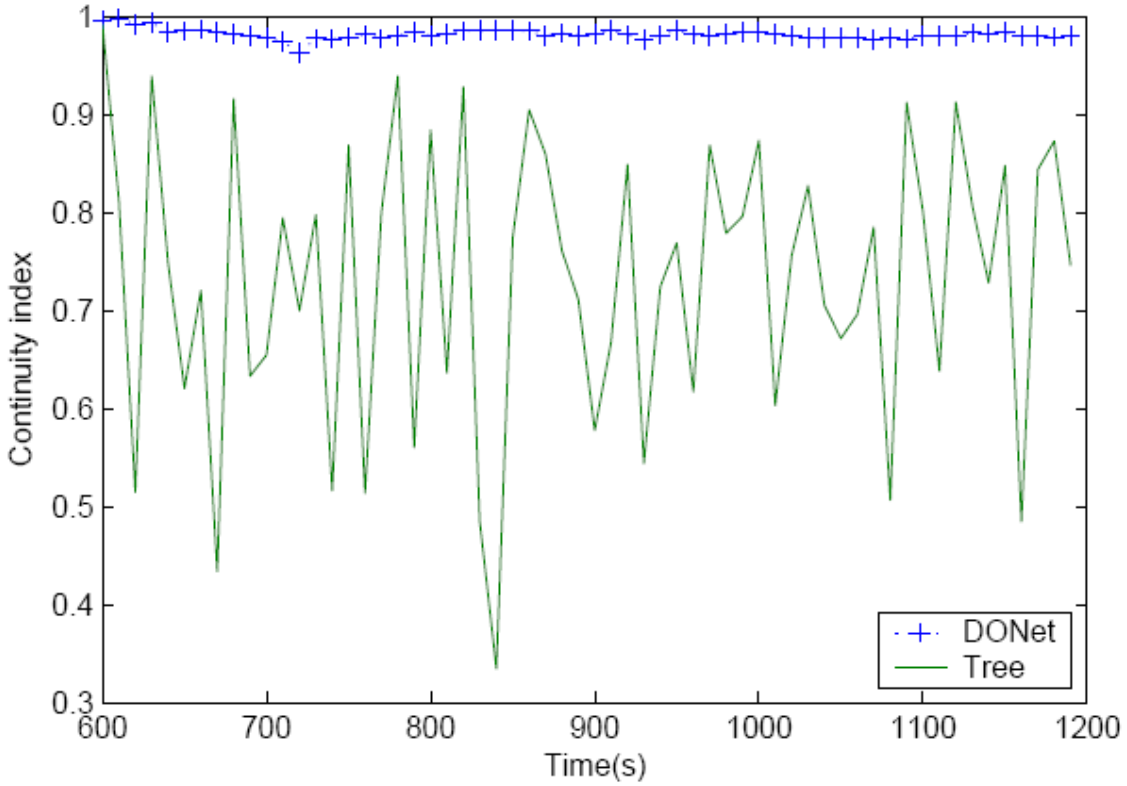
Compare to Tree-based Experiment



height = 19

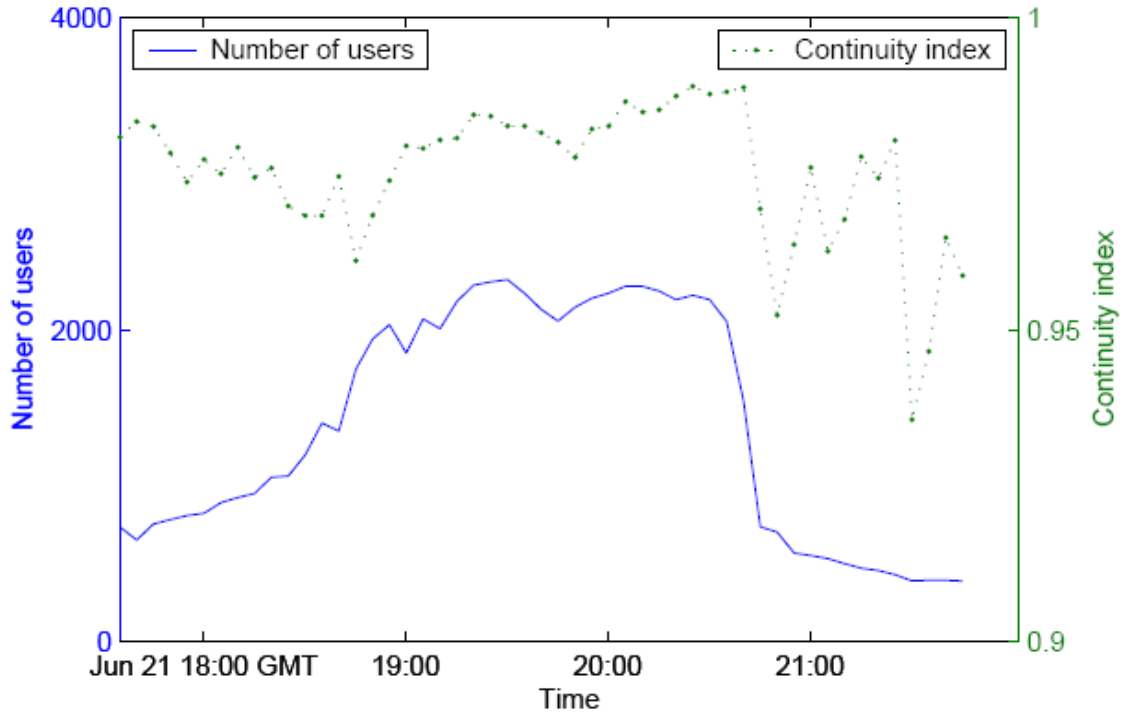
A snapshot of a tree-based overlay with 231 nodes.

Drastic Improvement over Tree-based



Samples of continuity indices for DONet and a tree-based overlay in a experiment (from 10 min to 20 min).

Peer Dynamics



Number of users and Continuity index over time.

2-3 years Later

- P2P streaming of live video (e.g. TV) is becoming established services (at least in China)
  - PPlive, PPstream, UUSee…
  - PPlive claims to have 75m installed base, and 20m active users
    - See http://www.sigcomm.org/sigcomm2007/p2p-tv/
  - More programs are legitimate content (purchased)
- Major issues
  - Managing service quality, QoS – integration with CDN?
  - Delivering advertisement
  - Managing relationships with ISPs

BiToS – BitTorent Streaming

A Vlavianos, M Iliofotou, M Faloutsos, "Bitos: Enhancing bit-torrent for supporting streaming applications" IEEE Infocom 2006

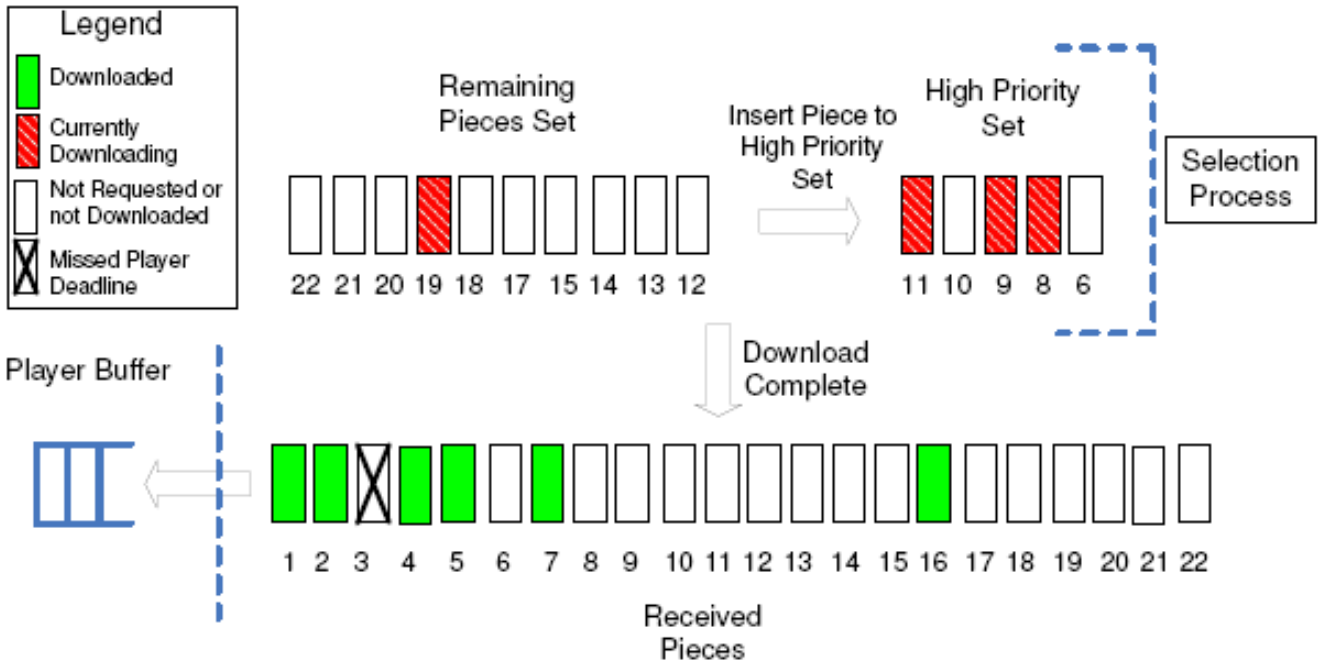
They ask – can BT (with small change to its chunk selection strategy) be used for streaming?

The modified BT by adapting chunk selection

- Select chunks in a sliding window
- Still use "rarest first" inside the window
- Also create a priority set of chunks to get

Their work is based on simulation

BiToS Design

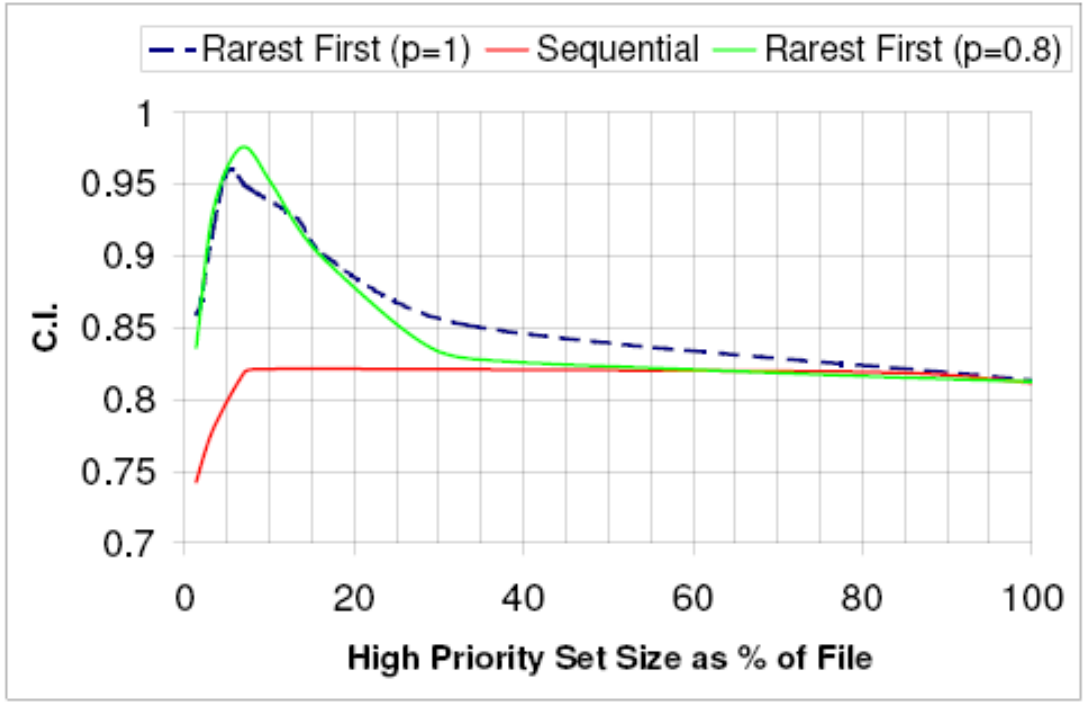


Approach for Supporting Streaming in BT

the peer chooses

- with probability p to download a piece of the video stream contained in the High Priority Set
- with probability 1 − p a piece contained in the Remaining Pieces Set

BiToS Result



High Priority Set Size as a percentage of File