# CS 268: Lecture 19 (Application Level Multicast)

Ion Stoica

March 22, 2001

(* Thanks to Yang-hua et al for making their slides available)

## Key Concerns with IP Multicast

- Scalability with number of groups
  - Routers need to maintain per-group state
    - Aggregation of multicast addresses is complicated
- Supporting higher level functionality is difficult
  - IP Multicast: best-effort multi-point delivery service
  - Reliability and congestion control for IP Multicast complicated
    - Need to deal with heterogeneous receiver → negotiation hard
- Deployment is difficult and slow
  - ISP's reluctant to turn on IP Multicast

## Approach

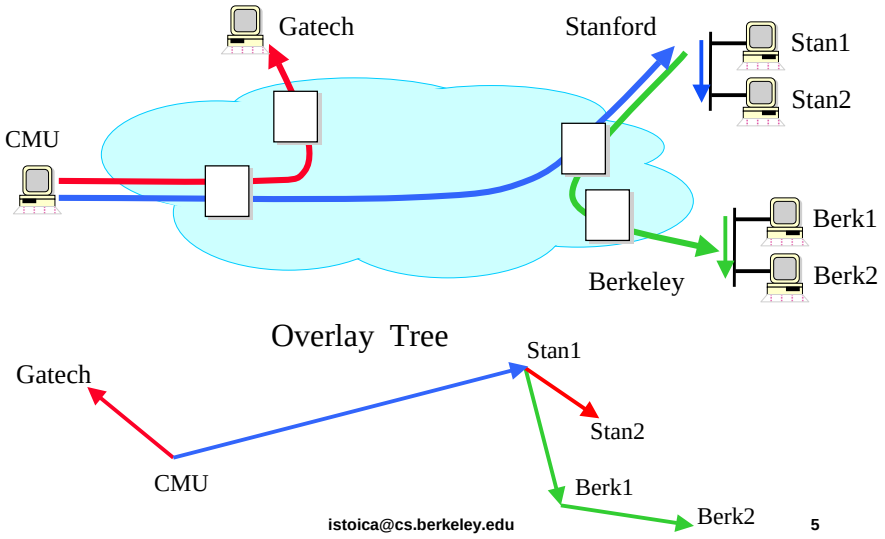
- Provide IP multicast functionality above the IP layer → application level multicast
- Challenge: do this efficiently



## Two Examples

- Narada [Yang-hua et al, 2000]
  - Multi-source multicast
  - Involves only end hosts
  - Small group sizes <= hundreds of nodes
  - Typical application: chat
- Overcast [Jannotti et al, 2000]
  - Single source tree
  - Assume an infrastructure; end hosts are not part of multicast tree
  - Large groups ~ millions of nodes
  - Typical application: content distribution

## Narada: End System Multicast






## Potential Benefits

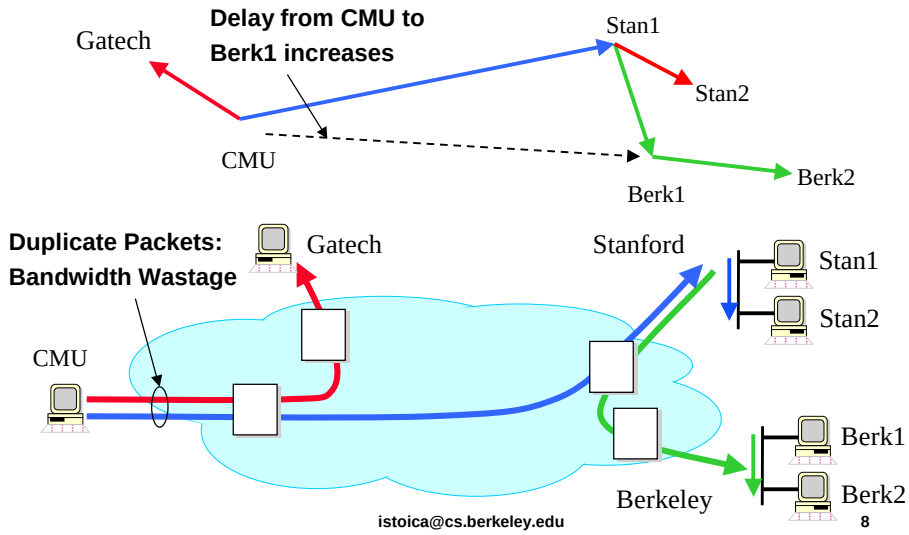
- Scalability
  - Routers do not maintain per-group state
  - End systems do, but they participate in very few groups
- Easier to deploy
- Potentially simplifies support for higher level functionality
  - Leverage computation and storage of end systems
  - For example, for buffering packets, transcoding,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

## End System Multicast: Narada

- A distributed protocol for constructing efficient overlay trees among end systems
- Caveat: assume applications with small and sparse groups
  - Around tens to hundreds of members

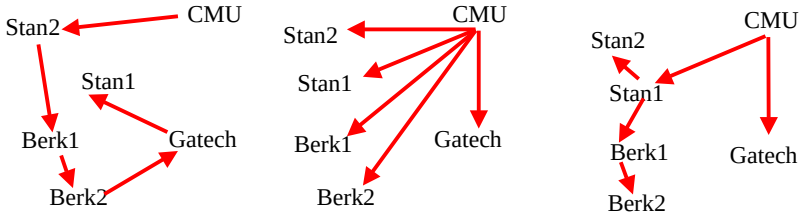


## Performance Concerns



## Overlay Tree

- The delay between the source and receivers is small
- Ideally,
  - The number of redundant packets on any physical link is low
- Heuristic:
  - 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

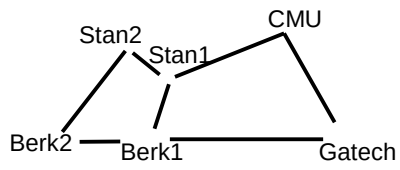
## Solution

- Two step design
  - Build a mesh that includes all participating end-hosts
  - Build source routed distribution trees



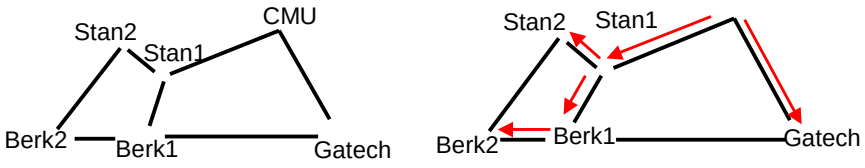
## Mesh

- Advantages:
  - Offers a richer topology → robustness; don't need to worry to much about failures
  - Don't need to worry about cycles
- Desired properties
  - Members have low degrees
  - Shortest path delay between any pair of members along mesh is small



## Overlay Trees

- Source routed minimum spanning tree on mesh
- Desired properties
  - Members have low degree
  - Small delays from source to receivers

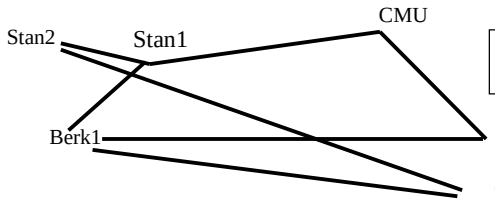





## Narada Components/Techniques

- 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:
Long path from Gatech2 to CMU

- 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

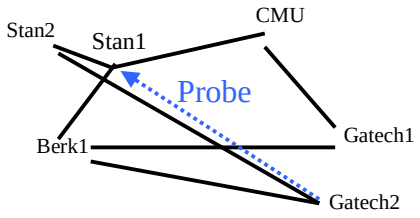


## The terms defined

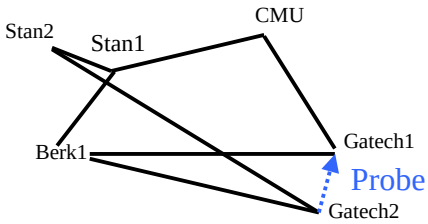
- Utility gain of adding a link based on
  - The number of members to which routing delay improves
  - How significant the improvement in delay to each member is
- Cost of dropping a link based on
  - The number of members to which routing delay increases, for either neighbor
- Add/Drop Thresholds are 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



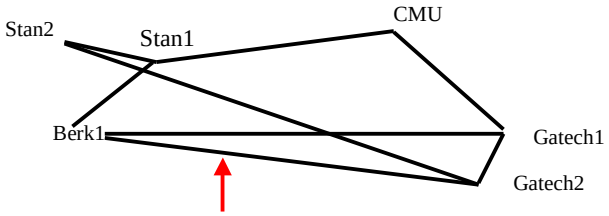

Delay improves to Stan1, CMU but marginally.
Do not add link!



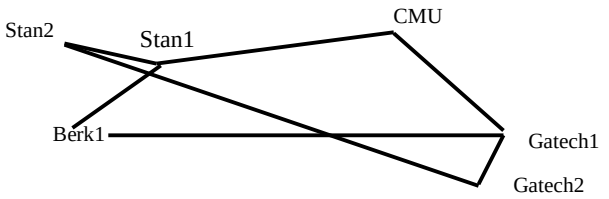

Delay improves to CMU, Gatech1 and significantly.
Add link!



## Example




Used by Berk1 to reach only Gatech2 and vice versa: Drop!!



## Simulation Results

- Simulations
  - Group of 128 members
  - Delay between 90% pairs < four times the unicast delay
  - No link caries more than 9 copies
- Experiments
  - Group of 13 members
  - Delay between 90% pairs < 1.5 times the unicast delay



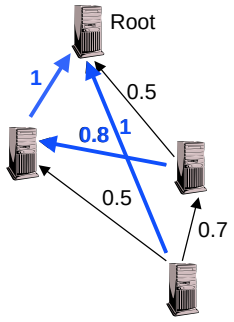
## Overcast

- Designed for throughput intensive content delivery
  - Streaming, file distribution
- Single source multicast; like Express
- Solution: build a server based infrastructure
- Tree building objective: high throughput

# Tree Building Protocol

- Idea: Add a new node as far away from the route as possible without compromising the throughput!

```
Join (new, root) {
   current = root;
   B = bandwidth(root, new);
   do {
      B1 = 0;
      forall n in children(current) {
         B1 = bandwidth(n, new);
         if (B1 >= B) {
           current = n;
           break;
         }
   } while (B1 >= B);
   new->parent = root;
}
```



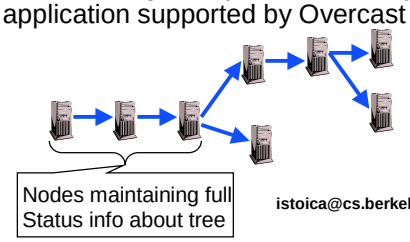



# Details

- A node periodically reevaluates its position by measuring bandwidth to its
  - Siblings
  - Parent
  - Grandparent
- The Up/Down protocol: track membership
  - Each node maintains info about all nodes in it sub-tree plus a log of changes
    - Memory cheap
  - Each node sends periodical alive messages to its parent
  - A node propagates info up-stream, when
    - Hears first time from a children
    - If it doesn't hear from a children for a present interval
    - Receives updates from children

## Details

- Problem: root → single point of failure
- Solution: replicate root to have a backup source
- Problem: only root maintain complete info about the tree; need also protocol to replicate this info
- Elegant solution: maintain a tree in which first levels have degree one
  - Advantage: all nodes at these levels maintain full info about the tree
  - Disadvantage: may increase delay, but this is not important for application supported by Overcast






## Some Results

- Network load < twice the load of IP multicast (600 node network)
- Convergence: a 600 node network converges in ~ 45 rounds

## Summary

- IP Multicast (1989) is not yet widely deployed: Why?
  - Scalability: per-group forwarding and control state → number of groups is a killer here
  - Difficult to support higher layer functionality → receiver heterogeneity is the killer here
  - Difficult to deploy, and get ISP's to turn on IP Multicast → no economic model
- Recently, a lot of work that try to get around these problems by pushing multicast functionality at the application level



## Summary

- End-system multicast (NARADA) : aimed to small-sized groups
  - Application example: chat
- Multi source multicast model
- No need for infrastructure
- Properties
  - low performance penalty compared to IP Multicast
  - potential to simplify support for higher layer functionality
  - allows for application-specific customizations

## Summary

- Overcast: aimed to large groups and high throughput applications
  - Examples: video streaming, software download
- Single source multicast model
- Deployed as an infrastructure
- Properties
  - Low performance penalty compared to IP multicast
  - Robust & customizable (e.g., use local disks for aggressive caching)



## Other Projects

- Scattercast (Chawathe et al, UC Berkeley)
  - Emphasis on infrastructural support and proxy-based multicast
  - Uses a mesh like Narada, but differences in protocol details
- Yoid (Paul Francis, Fastforward/ACIRI)
  - Uses a shared tree among participating members
  - Distributed heuristics for managing and optimizing tree constructions

# Conclusion

- Narada and Overcast demonstrate the flexibility of the application level multicast
  - I.e., the ability to optimize the multicast distribution to the application needs
- … but fragments the protocol space; inter-operability hard to achieve
- Questions
  - Is every application going to come with its multicast suite?
  - Are we going to end up with very few de facto standards for different categories of applications?