Scalable Application-Layer Multicast for Content Distribution

Bobby Bhattacharjee, Suman Banerjee

University of Maryland

http://www.cs.umd.edu/projects/nice

Scalable Wide-Area Content Delivery

- Wide-area content delivery is an important, emerging application

  Multicast *primitive* is certainly useful for scalability

  However, network layer multicast not widely deployed yet . . .

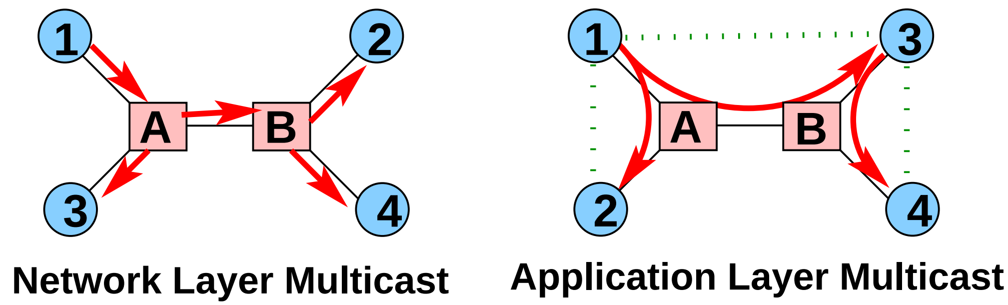
- Possible Solution: Implement multicast in the application layer

  Advantages: no change to infrastructure $\longrightarrow$ instant deployment

  Disadv.: Higher b/w usage, longer latency, more state at end nodes

Goal: Devise an app.-layer multicast protocol with “good” scalability and efficiency properties

"Good" Properties for App.-layer Multicast



- Low Stress — minimize copies of the *same* data sent over a link

  Requires level topology information

- Low Stretch — minimize overlay latency w.r.t. unicast shortest path latency

  If topology known, e2e stretch bounded by constant factor (UCB)

- Low Per-node state — ideally constant amount of state

  NICE

- Comparable robustness and security

  Node failures must be accounted for

Approaches for Building Overlay Trees

- Mesh-first:

  Creates a more densely connected structure first

  Data delivery path is a spanning tree of the mesh nodes

  Examples: Narada (CMU), Gossamer (UCB)

- Tree-first:

  The data delivery tree is created first

  Robustness via additional edges

  Examples: Yoid (ACIRI), ALMI (WU)

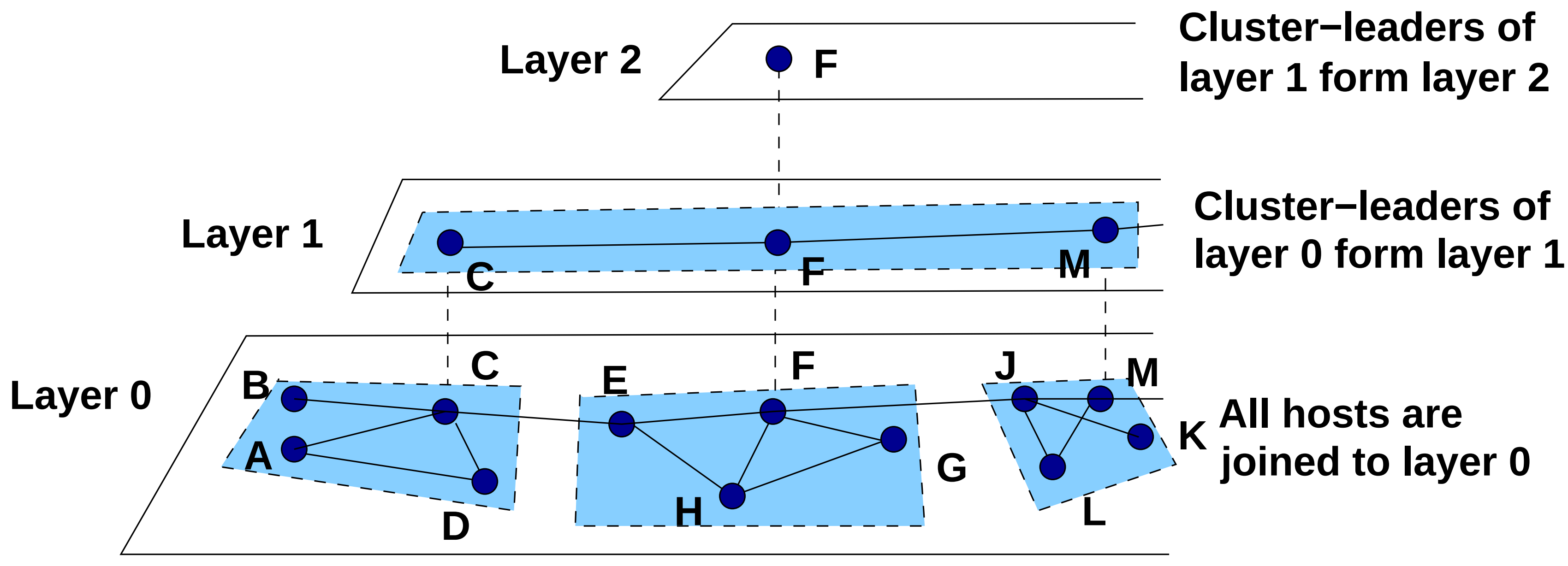
Narada Protocol (CMU)

- Canonical mesh-first scheme
    - New members choose random set of existing hosts as neighbors
    - Mesh quality is improved over time
    - Mechanism for recovery from mesh partitions
- Data delivery using source-specific trees
    - All members participate in a routing protocol over the mesh
    - Members forward data to other members using RPF check
- Requires O(num. of members) state and comm. at each member
- Simulated [Sigmetrics ’00] and implemented [SIGCOMM ’01]
    - Ideal for small groups

NICE Overlay Trees

- Consists of
  - A control topology
    - Structure with high connectivity
  - A data delivery topology
    - The control topology implicitly defines a base data delivery tree
    - However, the data tree can be independent of the control topology
- Main idea: Reduce state by using a hierarchy
  - End-hosts arranged in hierarchy of *layers* and *clusters*

NICE Hierarchy



- Structure Invariants

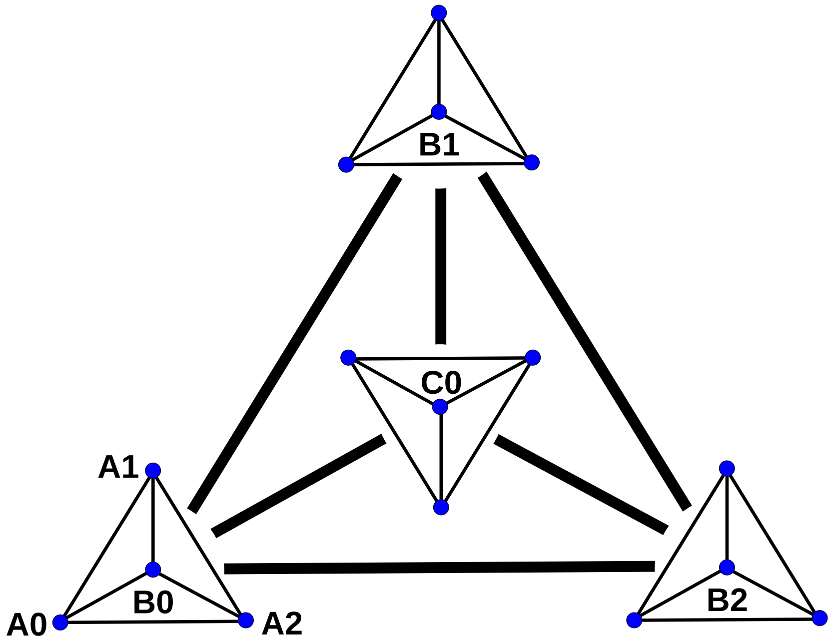
    An end-host belongs to a single cluster at any layer

    Cluster sizes have lower and upper bounds — between k and $2k$

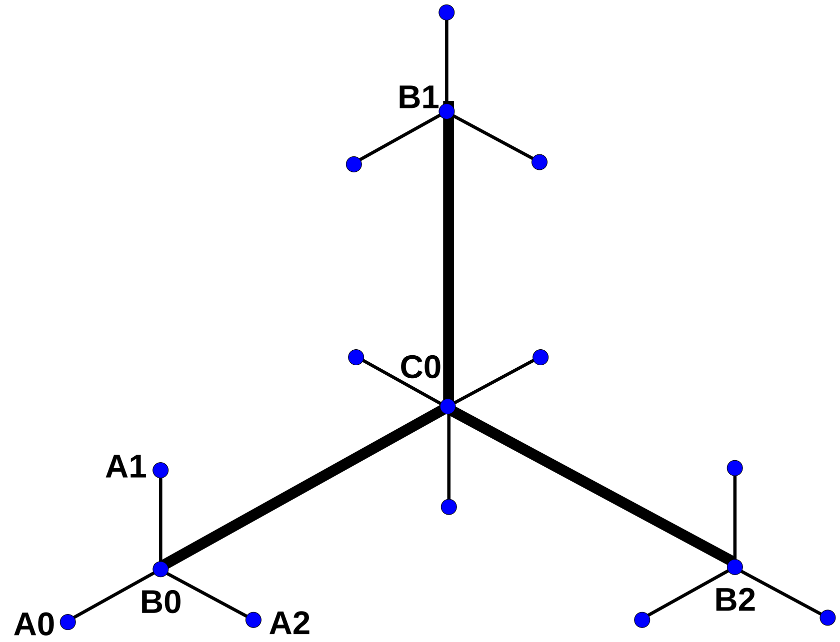
    The cluster leader is the *center* of the cluster

    Cluster leaders at a layer join a cluster in the next higher layer

Example Control and Data Paths



Control Path



Data Path

- Control path is the union of all the intra-cluster peerings.

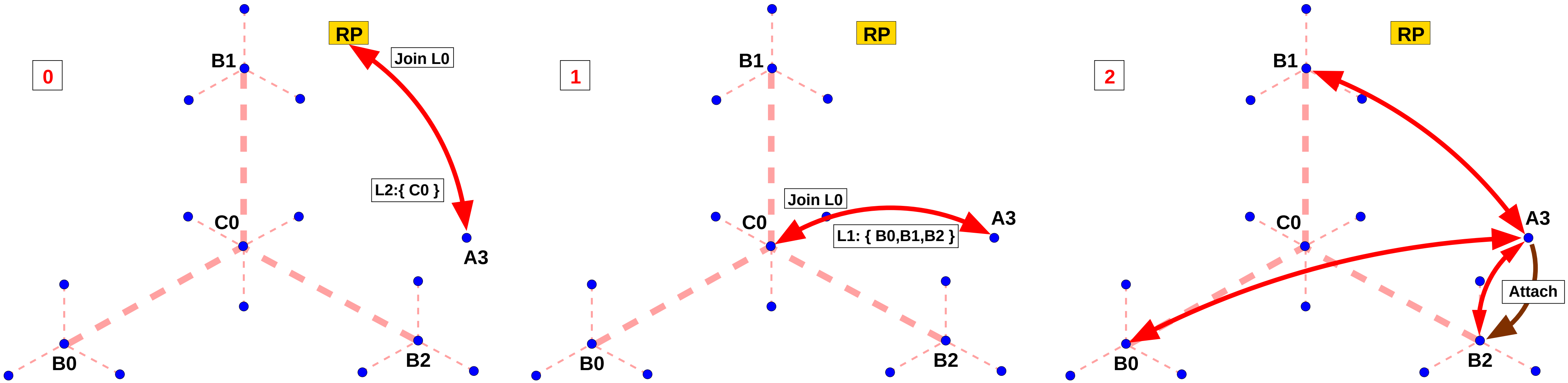
  Usually, within a cluster, connectivity is high

- The control topology *implicitly* defines a data delivery topology

  Possible to define other, better, data delivery trees

Join Procedure

- Assume a Rendezvous Point (RP)



- Join overhead: $O(\log N)$ RTTs and $O(k \log N)$ messages

  Some optimizations possible

Maintaining the Invariants

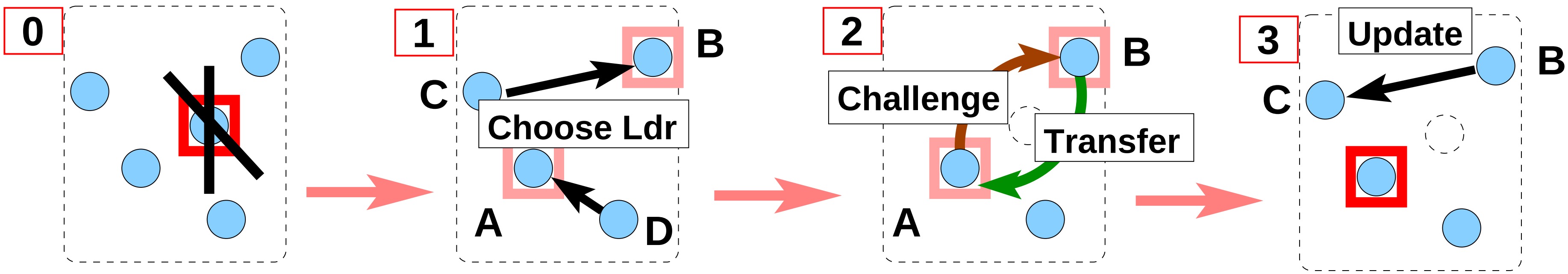
- Clusters split/merge to maintain size bounds
- Cluster Split:

  Leader partitions the cluster into two equal-sized clusters

- Cluster Merge:

  Small clusters merge with neighboring clusters at the same layer

Leader Elections



Leader lost

Each member independently chooses new leader

Competing leaders will reconcile

Single leader selected

- Heartbeat messages within each cluster
- Leader election protocol requires knowledge of all cluster members

State and Messages

- Members keep state for all members in each cluster to which they belong
- On average, state kept at each member is constant
- On average, control traffic overhead per member is constant

  In the worst case, both state and traffic overhaed is $O(k \log N)$

Simulation Study

- Packet-level simulations using 10 000 node TS graphs
- Hosts join and leave the multicast group arbitrarily
- Experiments with groups of size upto 2048
- Comparisons with NARADA protocol

Metrics

- Tree quality

  Stretch (Relative Delay Penalty)

  Stress

  Tree degree

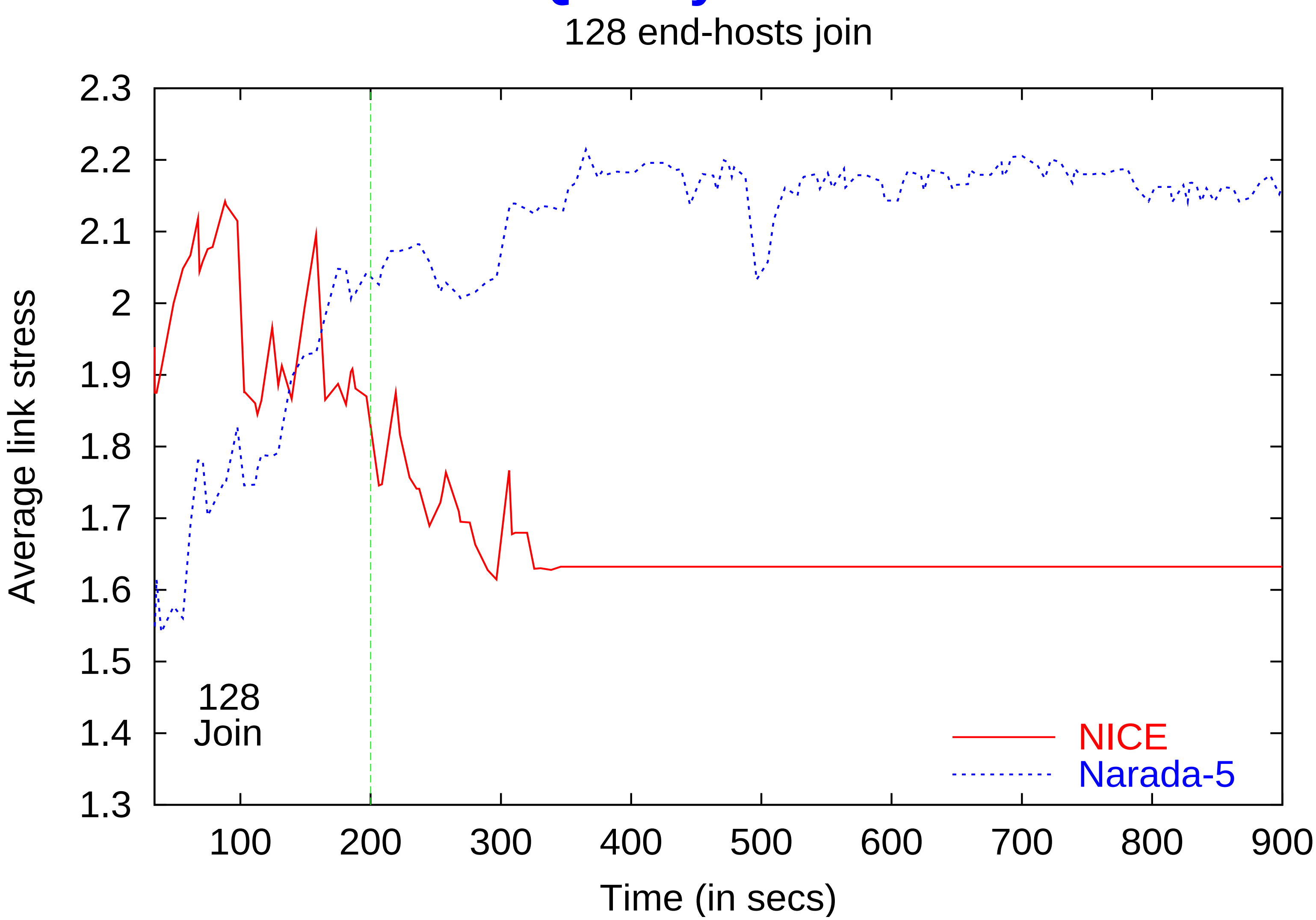
- Failure recovery

  Fraction of (remaining) members that receive a packet as end-hosts join (and leave) the group

- Protocol overheads

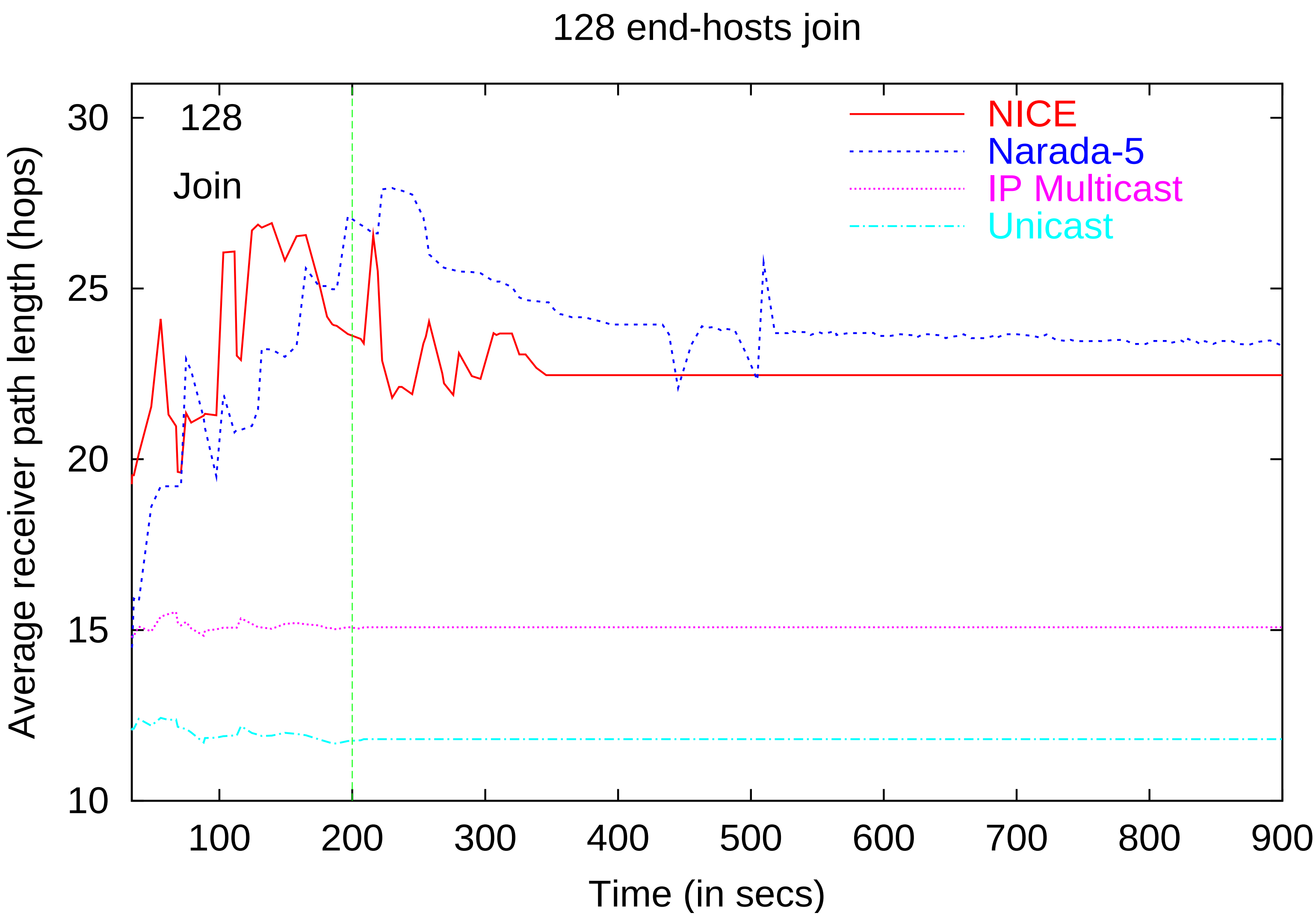
  Byte overheads at routers and end-hosts

Tree Quality: Stress

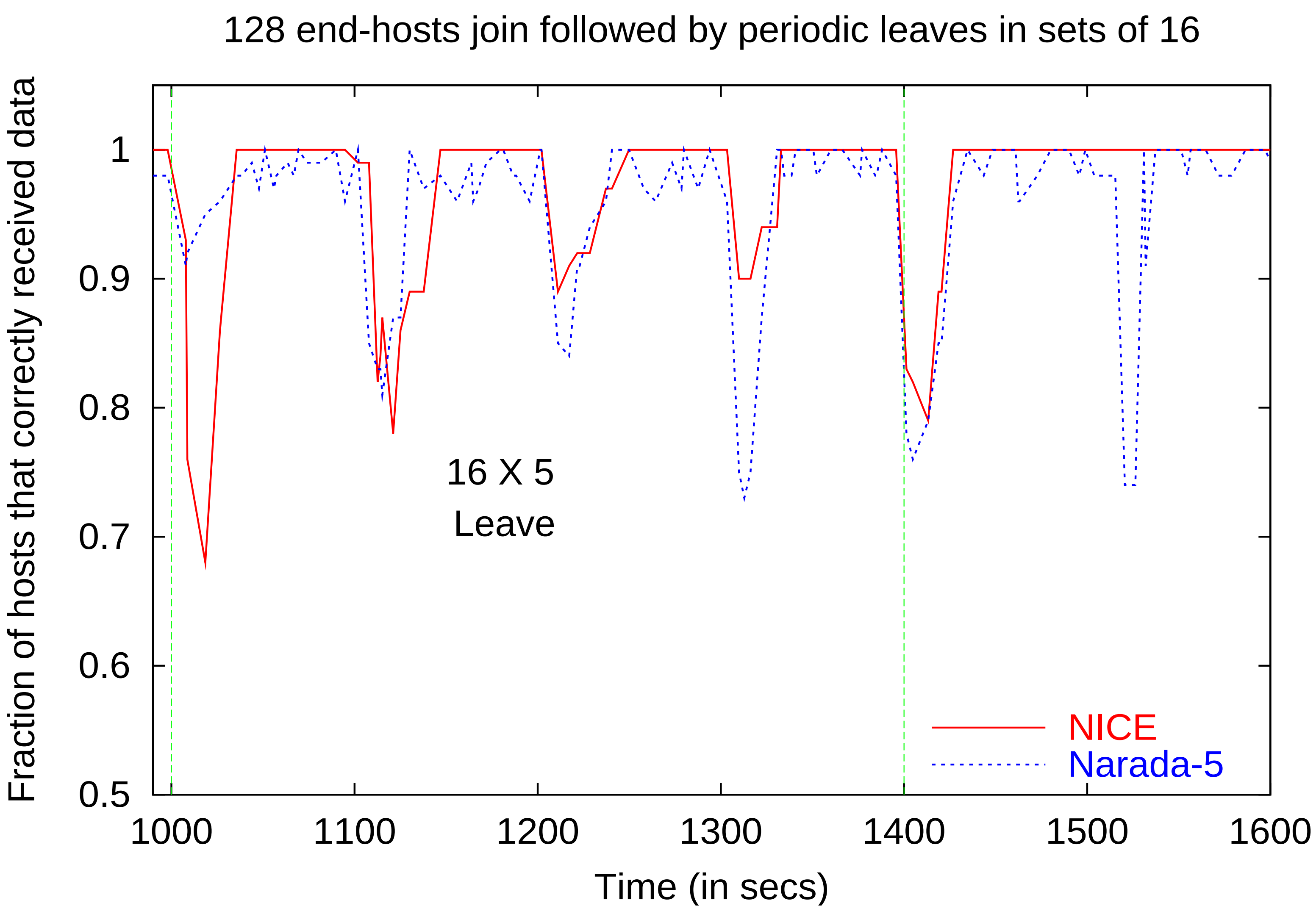


Join Phase: 128 members join in the first 200s; 5×16 members leave after time 1000s

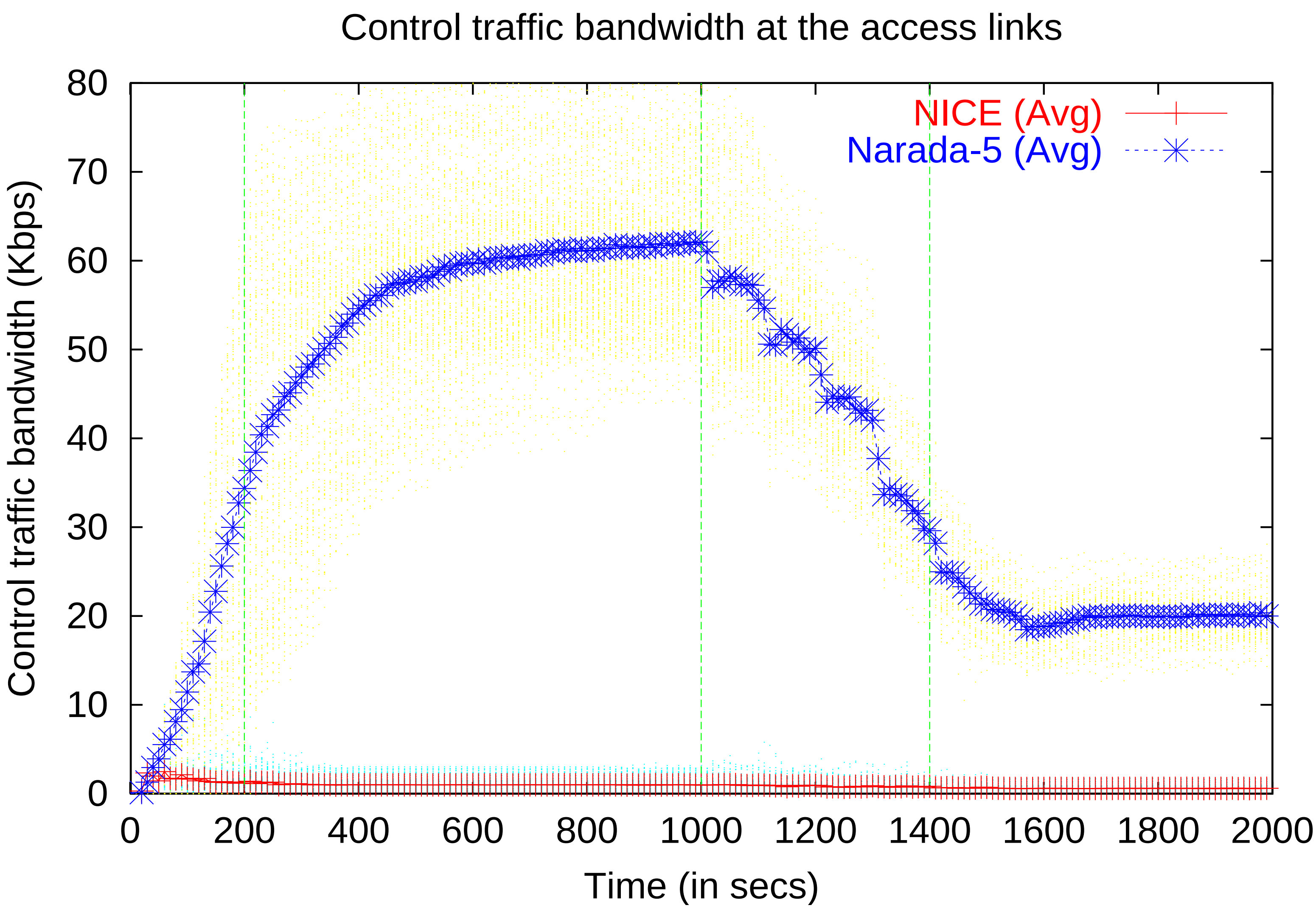
Tree Quality: Stretch



Failure recovery: Fr. of Group that receives data



Overhead



Control traffic bandwidth at the access links
Control traffic bandwidth (Kbps)
Time (in secs)
NICE (Avg)
Narada-5 (Avg)
0
10
20
30
40
50
60
70
80
0
200
400
600
800
1000
1200
1400
1600
1800
2000

Summary

Group Size	Router Stress Narada-5	Router Stress NICE	Link Stress Narada-5	Link Stress NICE	Path Length Narada-5	Path Length NICE	Overhead (KB) Narada-30	Overhead (KB) NICE
32	**2.13**	2.42	**1.54**	1.90	20.42	**17.23**	9.23	**1.03**
128	3.04	**2.36**	2.06	**1.63**	**21.55**	21.61	65.62	**1.19**
512	4.09	**2.34**	2.57	**1.62**	24.74	**22.63**	199.96	**1.93**
2048	-	2.92	-	1.93	-	24.08	-	5.18

- Path lengths and failure recovery similar for NARADA and NICE
- Stress (and variance of stress) is lower with NICE
- NICE has much lower control overhead

Current work

- Implementation

  Application: streaming-media delivery

- Interoperability with network layer multicast

- Incorporating security

  NICE security component

- An incentive based cooperation framework