

# Overlay Multicast/Broadcast

- Broadcast/Multicast Introduction
- Application Layer Multicast
- Unstructured Overlays
  - Centralised
  - Distributed
- Structured Overlays
  - Flooding: CAN & Prefix Flood.
  - Tree-based:  
Scribe/ SplitStream/ PeerCast  
Bayeux/BIDIR-SAM
  - Additional Design Mechanisms



# We need Multicast/Broadcast Services for ...

- Public Content Broadcasting
- Content Replication and Distribution
- Voice and Video Conferencing
- Collaborative Environments
- Gaming
- Rendezvous Processes / Neighbour Discovery
- Self Organisation of Distributed Systems
- ...

All of this seamless and ubiquitous!



# Broadcast

- ▶ Special mode of group communication: **all nodes**
- ▶ Operates **without active participation** of nodes
  - ▶ No signalling involved
  - ▶ Simple to map to lower layers (▶ shared media)
  - ▶ Potential of increased efficiency
  - ▶ Well suited for rendezvous processes
- ▶ Results in **flooding** – typically bound to limited domains (▶ locality)



# IP Multicasting

## Service for Transferring IP Datagrams to Host-Groups

- Originally: RFC 1112 (S. Deering u.a., 1989)
- Addresses a host-group by means of *one* group address
- Two types of Multicast:
  - Any Source Multicast (ASM)
  - Source Specific Multicast (SSM)
- Client protocol for group membership management (IGMP/MLD)
- Internet core left with complex Multicast Routing



# IP Mcast Deployment Issues

- ▶ Complexity versus Performance Efficiency
  - ▶ IP Multicast most efficient, but burdens infrastructure
- ▶ Provider Costs
  - ▶ Provisioning of knowledge, router capabilities & maintenance, Interdomain multicast routing problem
  - ▶ Business model: Multicast saves bandwidth, but providers sell it
- ▶ Security
  - ▶ ASM assists unrestricted traffic amplification for DDoS-attacks
- ▶ End-to-End Design Violation?
  - ▶ Service complexity objects implementation at lower layer
  - ▶ But for efficiency: Multicast favors lowest possible layer



# Multicast: Alternative Approaches

- ▶ Application Layer Multicast (ALM)
  - ▶ Solely built with end-user systems
  - ▶ Free of any infrastructure support (except unicast)
  
- ▶ Overlay Multicast
  - ▶ Built on fixed nodes / proxies
  - ▶ Nodes connect to local proxies
  - ▶ Proxies responsible for routing



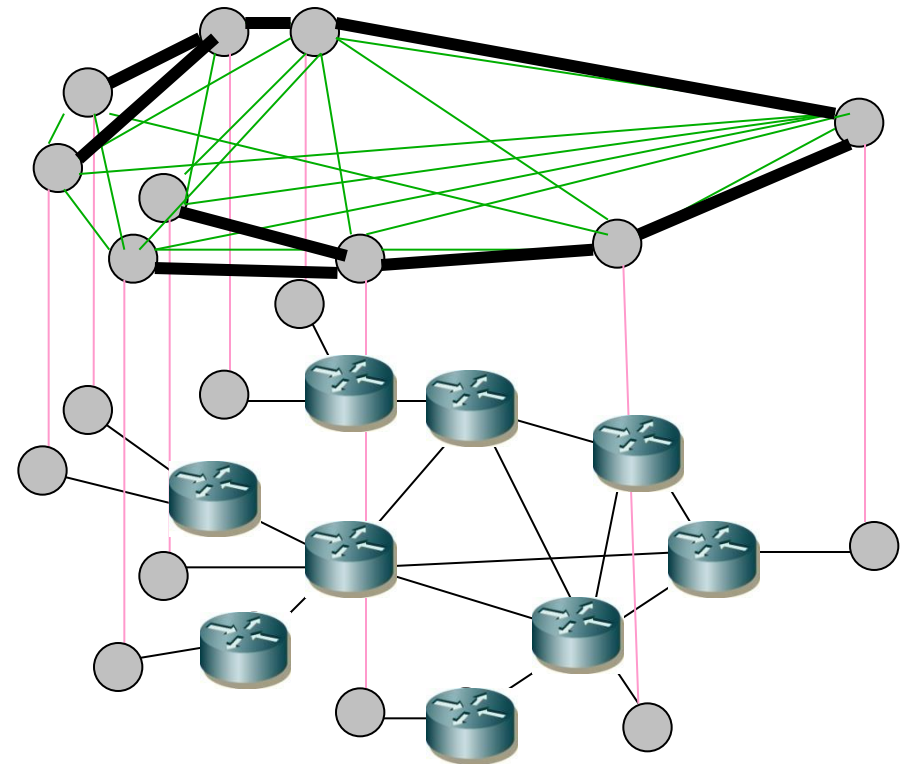
# Application Layer Multicast

## Advantages:

- Easy to deploy

## Disadvantages:

- High control overhead
- Low efficiency
- Degradation by end system instability



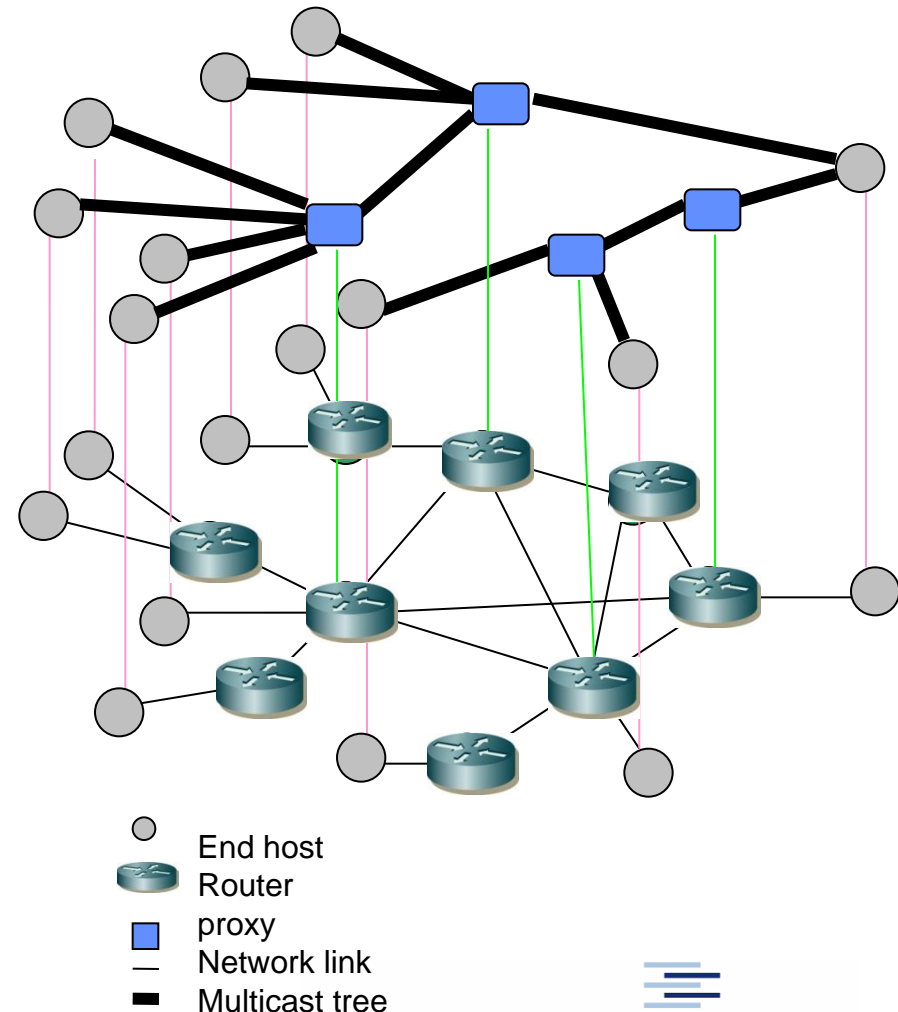
# Overlay Multicast

## Advantages:

- Improved efficiency in tree management
- Enhanced scalability
- Reduced control overhead

## Disadvantages:

- Deployment complexity





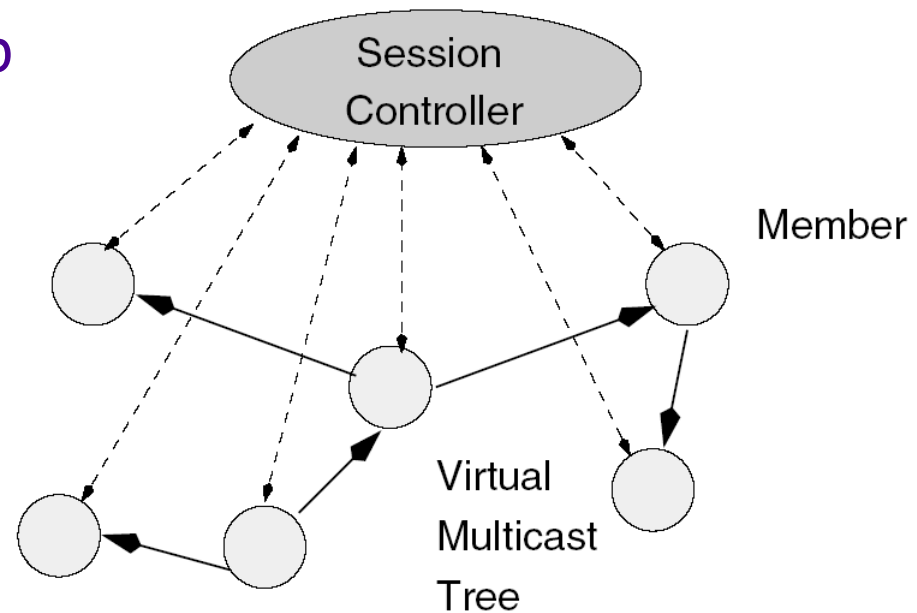
# Approaches to ALM/OLM

- ▶ Mesh first
  - ▶ Group members establish an (unstructured) mesh
  - ▶ Data distribution according to **tree built on top of the mesh** or **data driven** (pull mechanism)
- ▶ Tree first
  - ▶ Group members establish a **distribution tree**
  - ▶ Sender driven (push mechanism)
- ▶ Randomized / epidemic dissemination
  - ▶ Group members broadcast to selected neighbors (Gossip)



# Unstructured ALM: ALMI

- ▶ Relies on Session Controller
  - ▶ Dedicated server or group member node
  - ▶ Computes minimal spanning distribution tree
  - ▶ Assigns tree neighbours
- ▶ Controller unicasts messages per member



# ALMI Self Organisation

## Node Arrival:

- ▶ New node sends *JOIN* to controller, in response receives its ALM ID + parent location → **tree membership**
- ▶ Node submits *GRAFT* to request data from parent → **data forwarding**

## Node Departure:

- ▶ Departing node sends *LEAVE* to controller, which then updates tree neighbours

## Overlay Maintenance:

- ▶ Group members probe on others and report to controller



# ALMI: Summary

- ▶ Early, elementary approach

## *PROs:*

- ▶ Tree building easy adaptable to local requirements

## *CONs:*

- ▶ Scalability and reliability problems due to centralized controller
- ▶ Scalability issue of maintenance: Mutual neighbour probing requires up to  $\Theta(n^2)$  messages



# Unstructured, distributed: End System Multicast/ Narada (Chu et al. 2000)

- ▶ Construct overlay tree from a mesh
  - ▶ Overlay nodes first organize in a redundantly meshed graph
  - ▶ Source specific shortest path trees then constructed from reverse paths
- ▶ Group management equally distributed on all nodes
  - ▶ Each overlay node keeps track of all group members
  - ▶ Periodic heartbeat broadcasts of all members
- ▶ Regulates node fan-out degree to balance load



# Narada Components

## ► Mesh Management:

- Ensures mesh remains connected in face of membership changes

## ► Mesh Optimization:

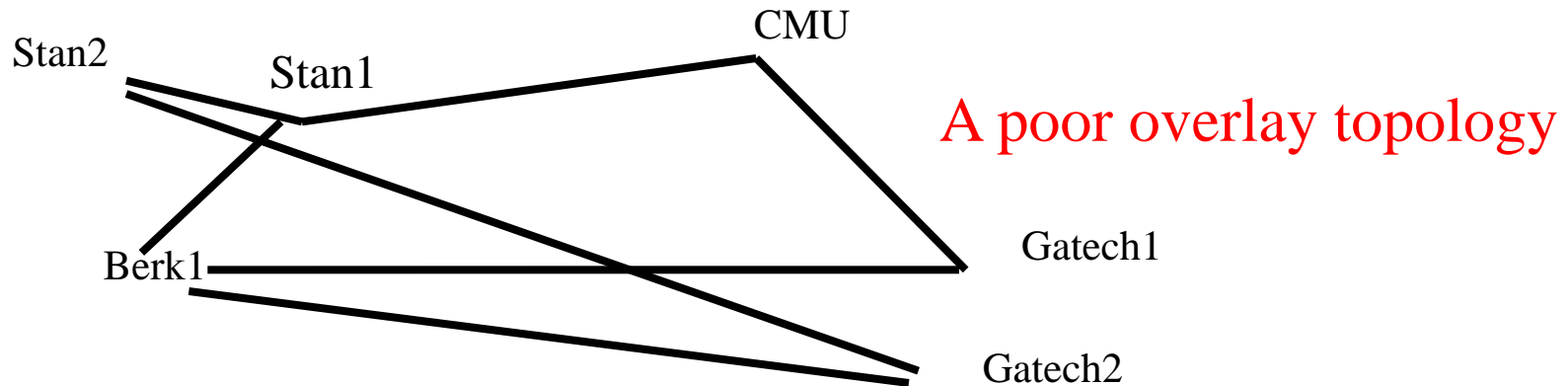
- Distributed heuristics for ensuring shortest path delay between members along the mesh is small

## ► Spanning tree construction:

- Routing algorithms for constructing data-delivery trees
- Distance vector routing, and reverse path forwarding
- Discovery and tree building analogue to DVMRP



# Optimizing Mesh Quality



- Members periodically probe other members at random

- New Link added if

Utility Gain of adding link  $>$  Add Threshold

- Members periodically monitor existing links

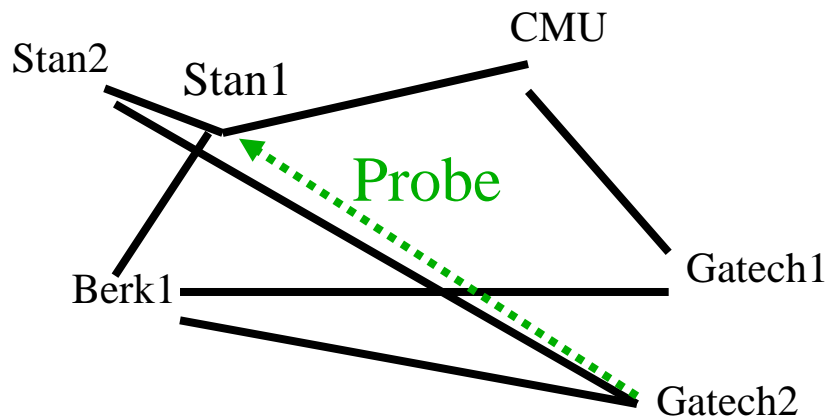
- Existing Link dropped if

Cost of dropping link  $<$  Drop Threshold



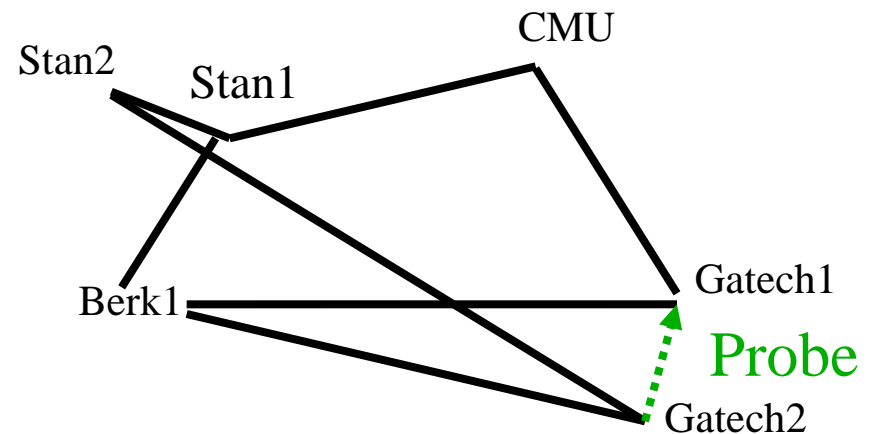
# Desirable properties of heuristics

- **Stability:** A dropped link will not be immediately readded
- **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!**

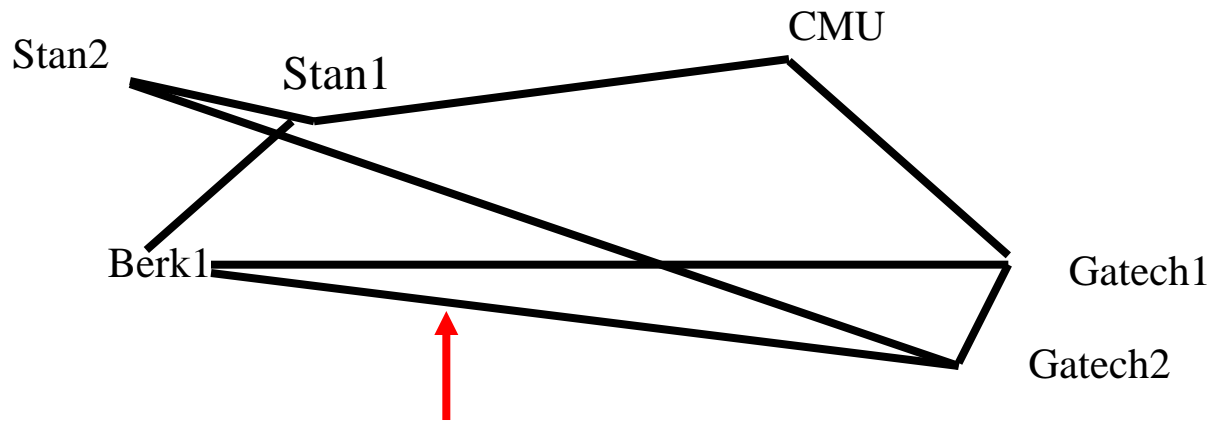


Delay improves to CMU, Gatech1 and significantly.

**Add link!**

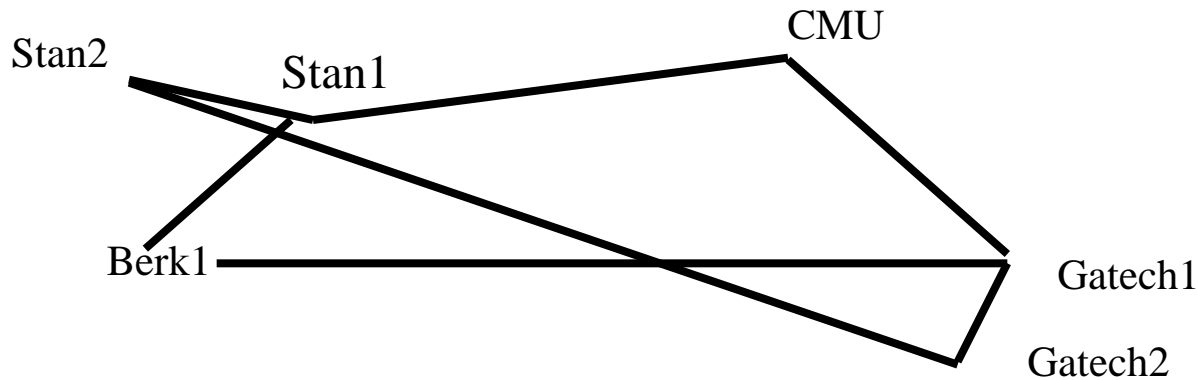






Used by Berk1 to reach only Gatech2 and vice versa.

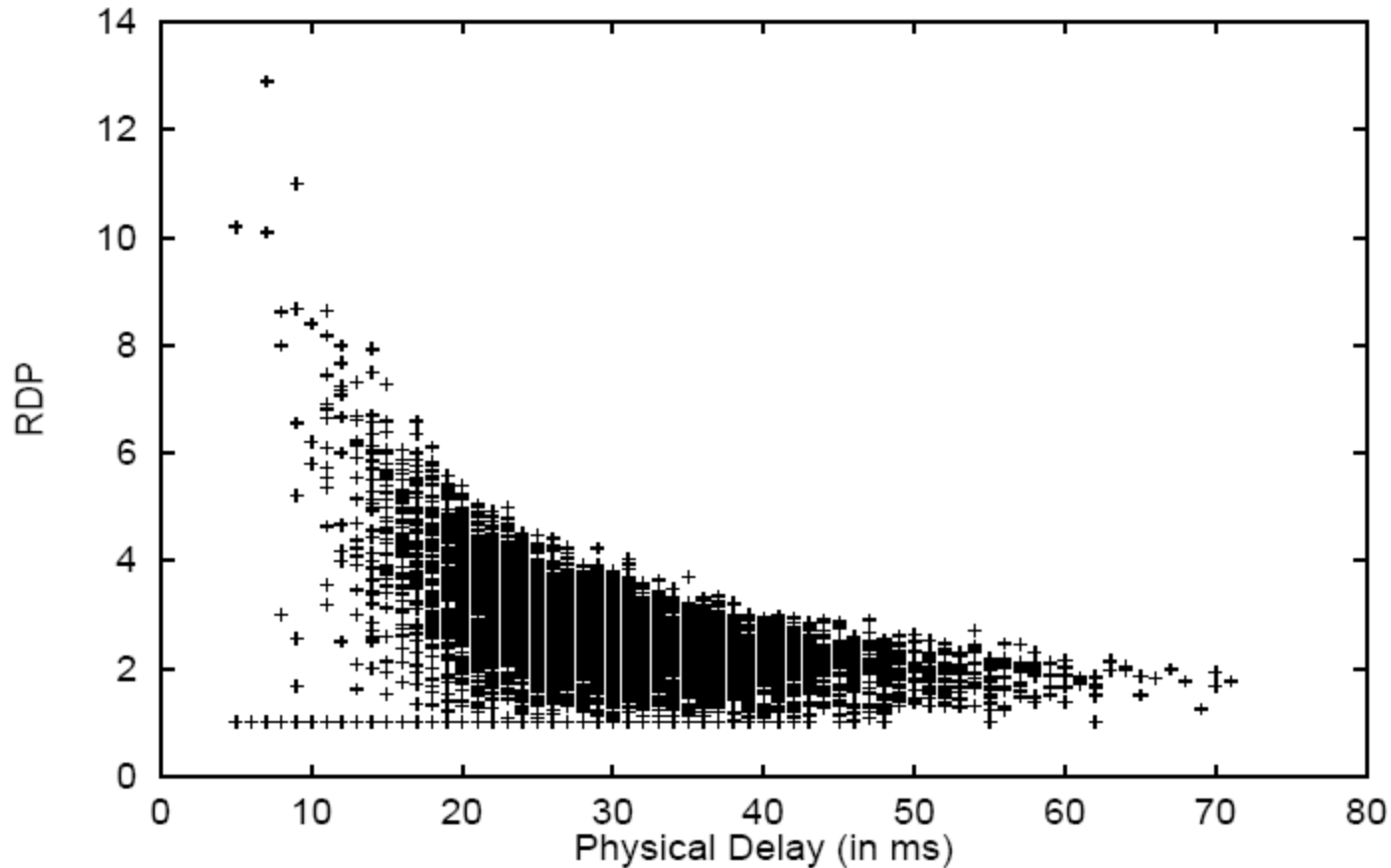
**Drop!!**



**An improved mesh !!**



# Evaluation: Relative Delay Penalty



128 Group Members within 1024 Nodes with 3145 Links



# Narada Summary

- ▶ Elementary mesh-centric approach
  - ▶ Topology inherited from mesh management

## *PROs:*

- ▶ Mesh organization easily adapts to underlay characteristics
- ▶ Decentralized group management, independent of individual nodes
- ▶ Fan-out adaptation

## *CONs:*

- ▶ Meshes do not adapt well to proximate environments
- ▶ Flooding & pruning inefficient, but required whenever mesh changes
- ▶ Scalability issues in group management: Heartbeat and tracking required

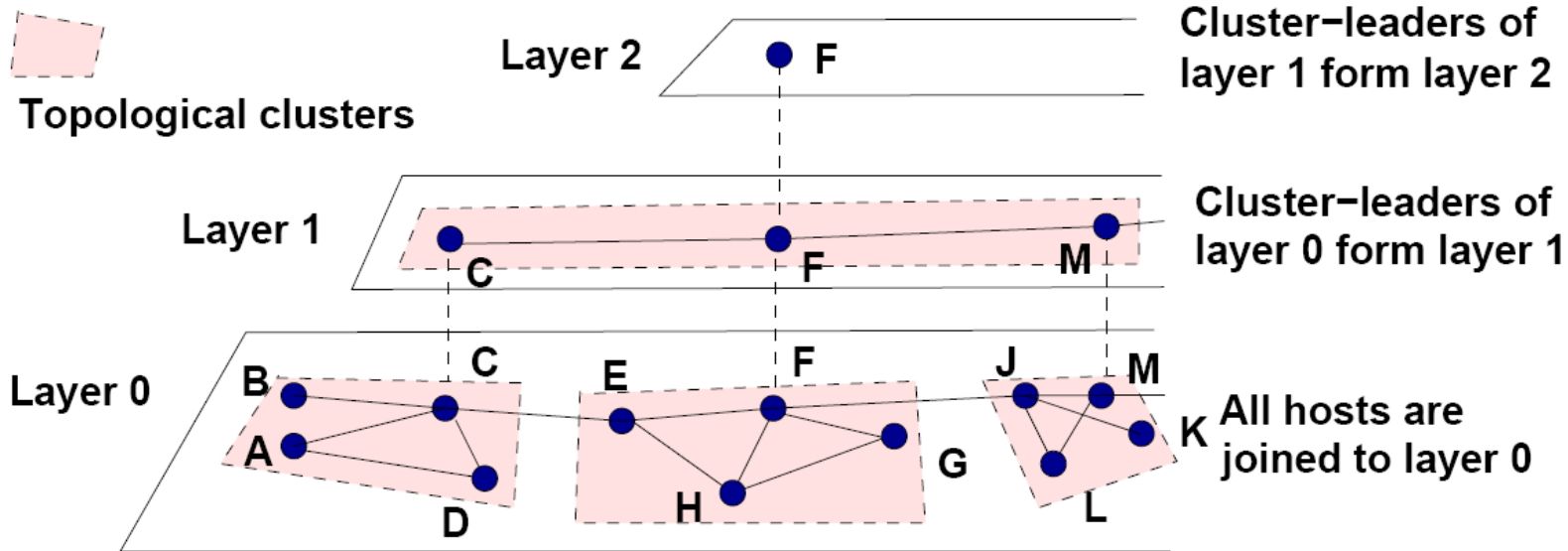


# Unstructured Scalable: NICE (Banerjee et al. 2002)

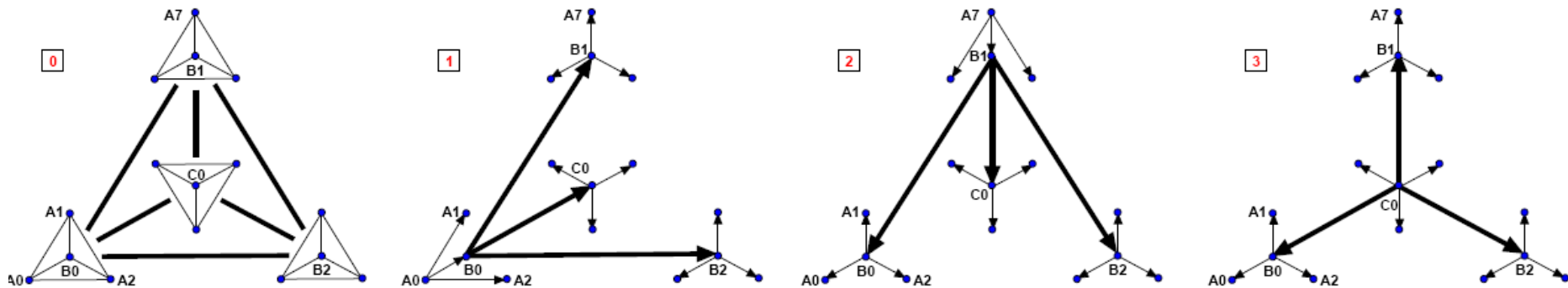
- ▶ Cluster-based approach: topologically close nodes are combined in clusters of approx. equal size
- ▶ Hierarchies are formed from clusters:
  1. All nodes are in some cluster at layer 0
  2. Each cluster determines a leader, leaders form next layer
  3. Layered clustering continues until leader set sizes match cluster size
  4. Last leader is root
- ▶ Cluster-Hierarchy generates trees



# Cluster to Tree

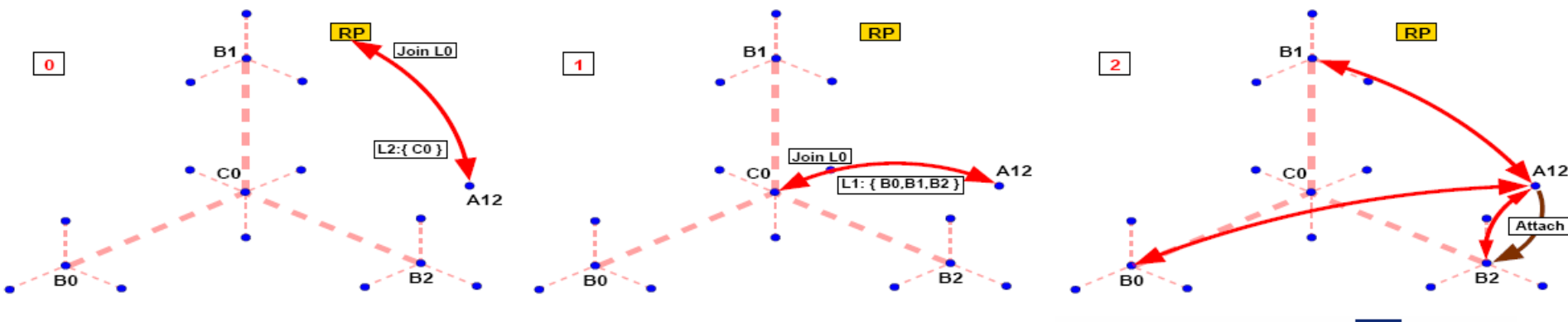


## Control- and data forwarding trees (source-specific shared trees):



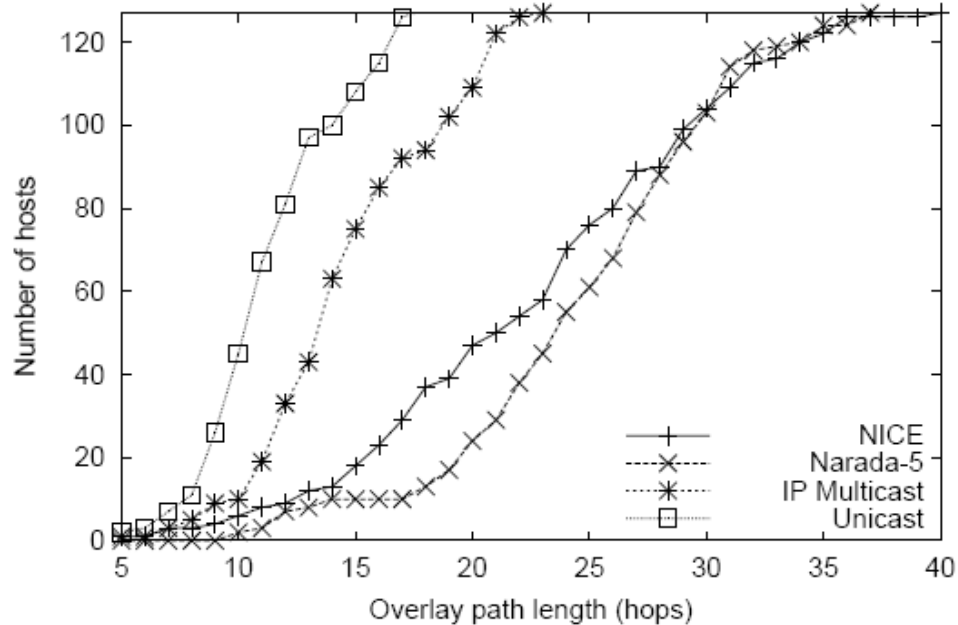
# Group Management

- ▶ One cluster hierarchy per group, a well known RP is assumed
- ▶ Joining node contacts RP and learns root node
- ▶ Joining node descends hierarchy to find appropriate cluster in layer 0

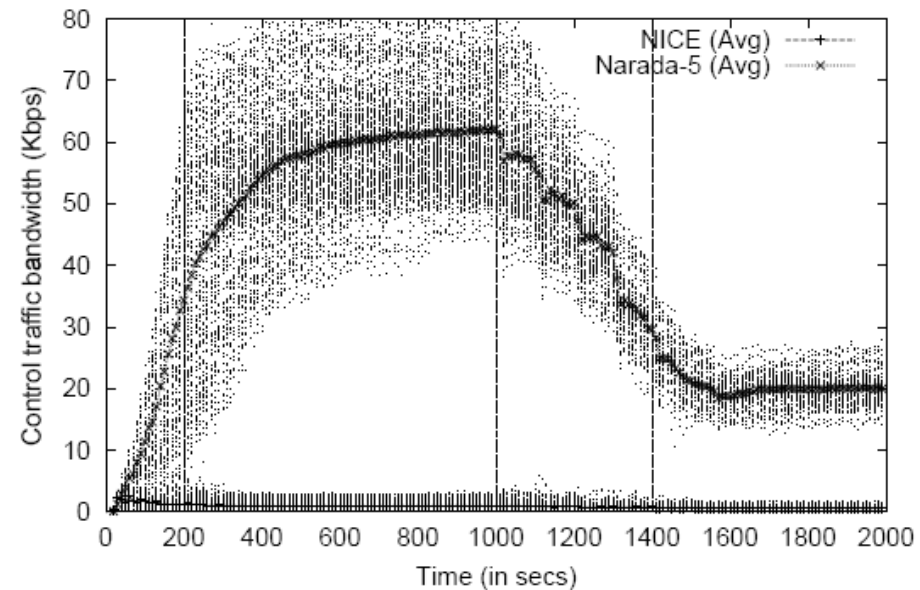


# Nice Performance

Cumulative distribution of data path lengths after overlay stabilizes



Control traffic bandwidth at the access links



# Nice Summary

- ▶ Unstructured cluster-centric approach
  - ▶ Topology reflected by clusters

## *PROs:*

- ▶ Scales logarithmically in hops and control overhead
- ▶ Replication load (fan-out) bound by a constant
- ▶ Constant state per node

## *CONS:*

- ▶ Topological knowledge is assumed, as well as known RP
- ▶ Clusters need maintenance after node arrival and departure





# Structured Overlay Multicast

- ▶ Flooding-based approaches
  - ▶ Packet broadcasts within a structured overlay
  - ▶ Selective broadcast (multicast) by group-specific DHT
    - ▶ Multicast on CAN & Prefix Flooding
- ▶ Tree-based approaches
  - ▶ Shared trees: Routing via group-specific rendezvous point
    - ▶ Scribe/Splitstream
  - ▶ Source-specific trees: Construction of source-specific shortest path trees after source announcements
    - ▶ Bayeux, BIDIR-SAM



# Multicast on CAN (Ratnasamy et al 2001)

- ▶ Within a previously established CAN overlay members of a Group form a “mini” CAN
  - ▶ Group-ID is hashed into the original CAN
  - ▶ Owner of the Group key used as bootstrap node
- ▶ Multicasting is achieved by flooding messages over this mini CAN
- ▶ Number of multicast states is limited by  $2d$  neighbours – independent of multicast source number!
- ▶ Can Multicast scales well up to very large group sizes
  - ▶ Replication load limited to neighbours ( $2d$ )
  - ▶ But tends to generate packet duplicates

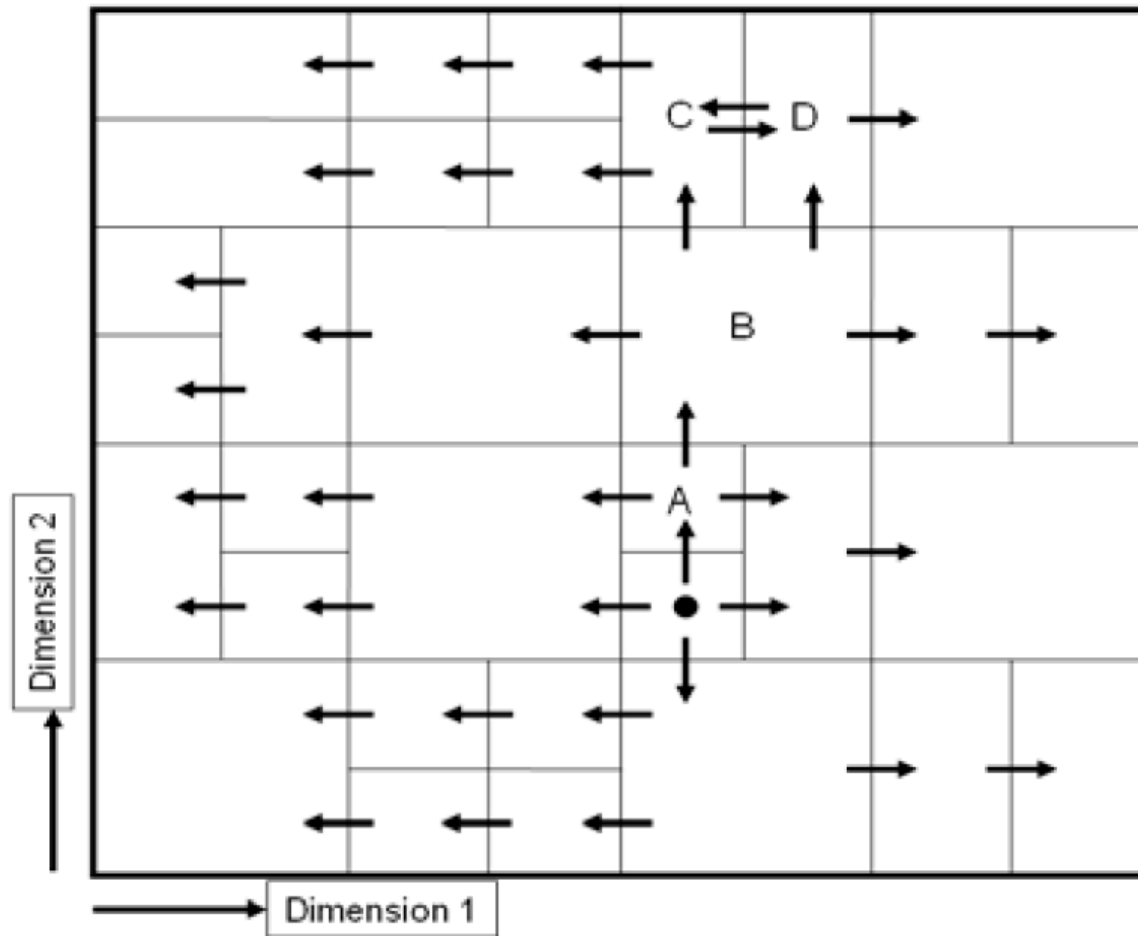


# Improved Flooding

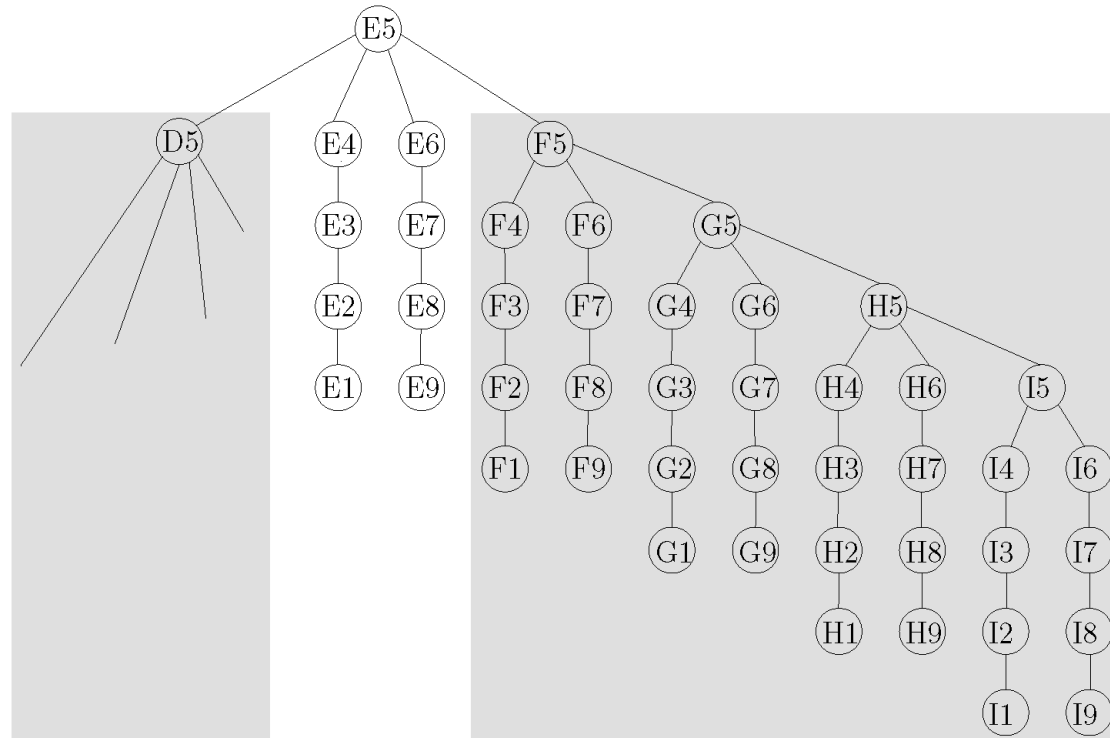
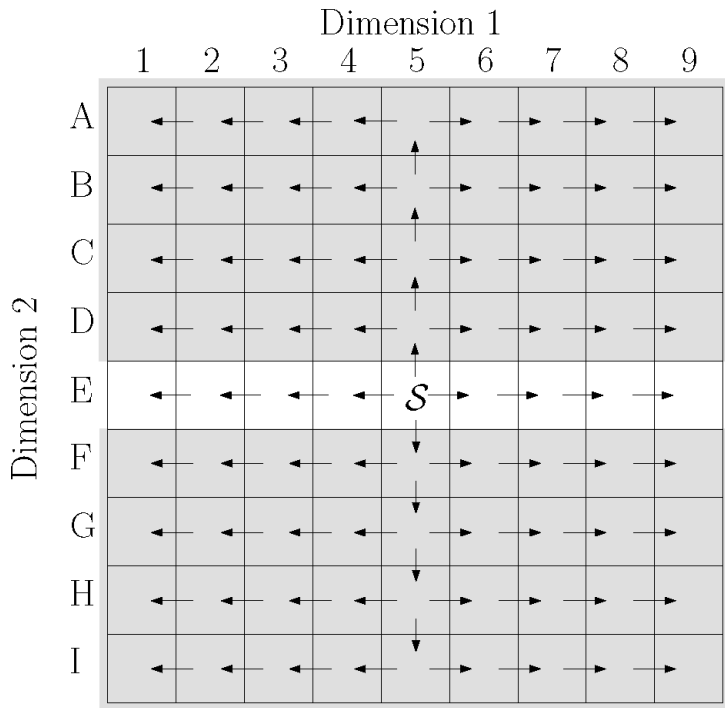
- ▶ Source of a messages forwards it to all neighbours
- ▶ Receiver of a message (from dimension  $i$ ) only forwards along dimensions lower than  $i$  and along  $i$  in opposite direction
- ▶ A node does not forward to a dimension, where the message has already travelled half way from source coordinate
- ▶ Nodes cache sequence numbers already forwarded to prevent duplicate forwarding



# Can Forwarding



# Forwarding in Idealized CAN

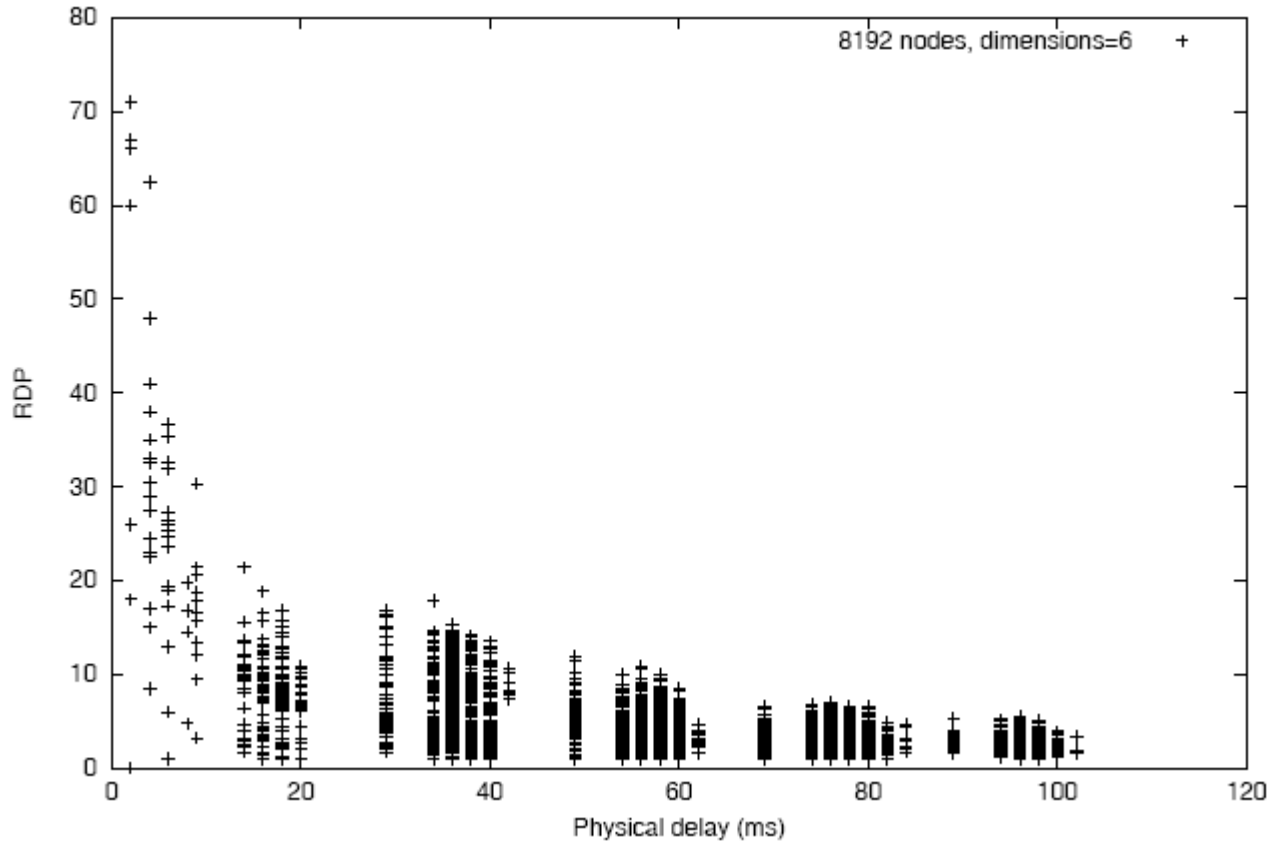


Even HyperCube

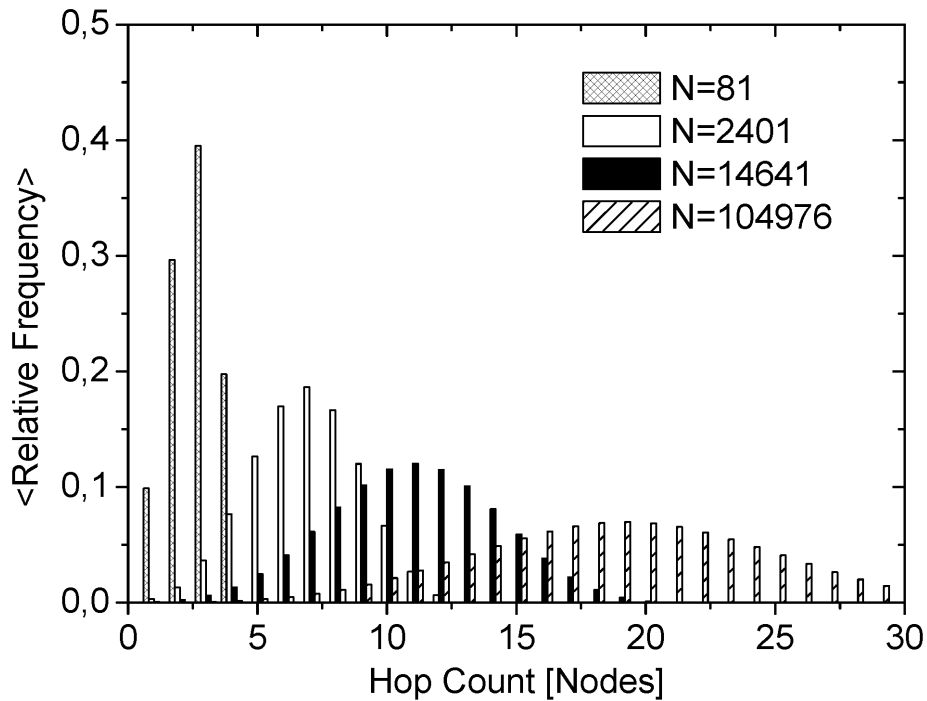
Corresponding Tree



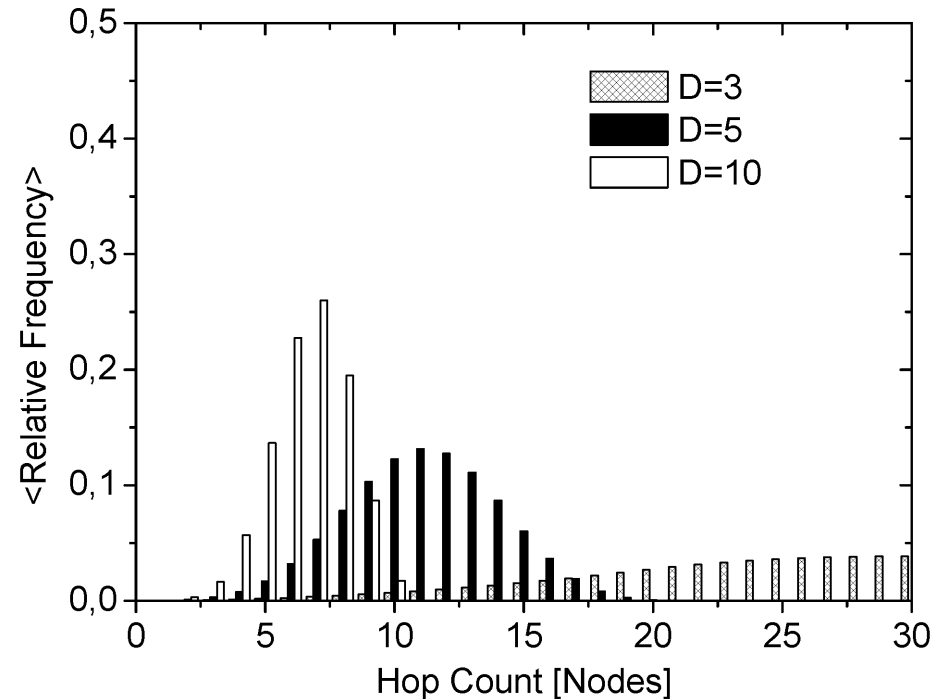
# Evaluation: Relative Delay Penalty



# Hopcount Distribution



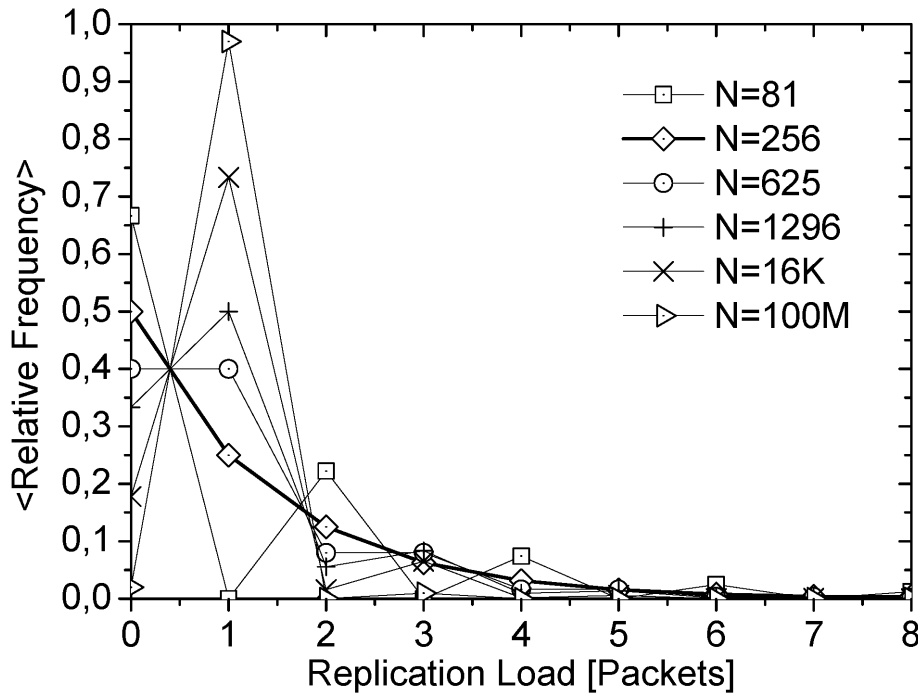
Dimension = 4



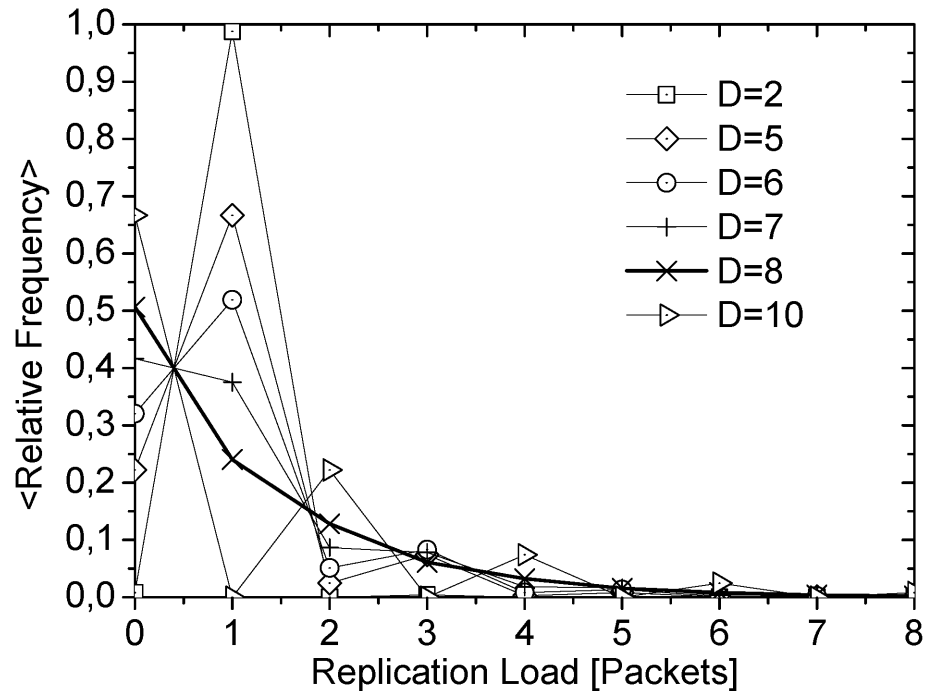
Network Size = 59049



# Replication Load



Dimension = 4



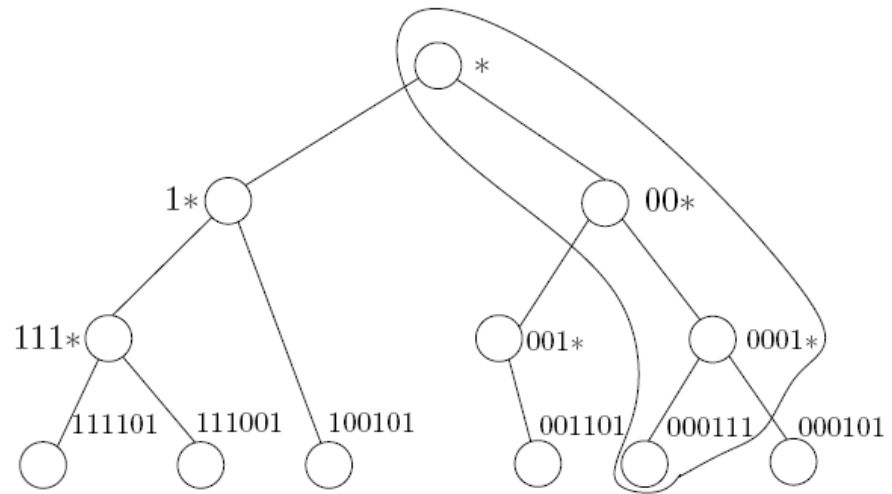
Network Size = 59049





# Prefix Flooding

- DHT Nodes are identified by hash codes
- Idea:
  - Arrange IDs in a prefix tree
  - Flood prefix neighbours (w.r.t. longest common prefix - LCP)
- Defines broadcast for any DHT, Multicast per mini-DHT analogue to CAN
- Packet delivery unique: no duplicates
- Particularly well suited for proximity-aware prefix routing like in Pastry



# Prefix Flooding Algorithm

Routing requires:

- Destination prefix  $\mathcal{C}$  for on-tree context
- Proactive routing maintenance: prefix neighbour entries needed for forwarding

## PREFIX FLOODING

- ▷ On arrival of a packet with destination prefix  $\mathcal{C}$
- ▷ at a DHT node

```
1 for all  $\mathcal{N}_i$  IDs in prefix neighbor set
2   do if ( $LCP(\mathcal{C}, \mathcal{N}_i) = \mathcal{C}$ )  ▷  $\mathcal{N}_i$  dntree neighbor
3     then  $\mathcal{C}_{new} \leftarrow \mathcal{N}_i$ 
4     FORWARD PACKET TO  $\mathcal{C}_{new}$ 
```

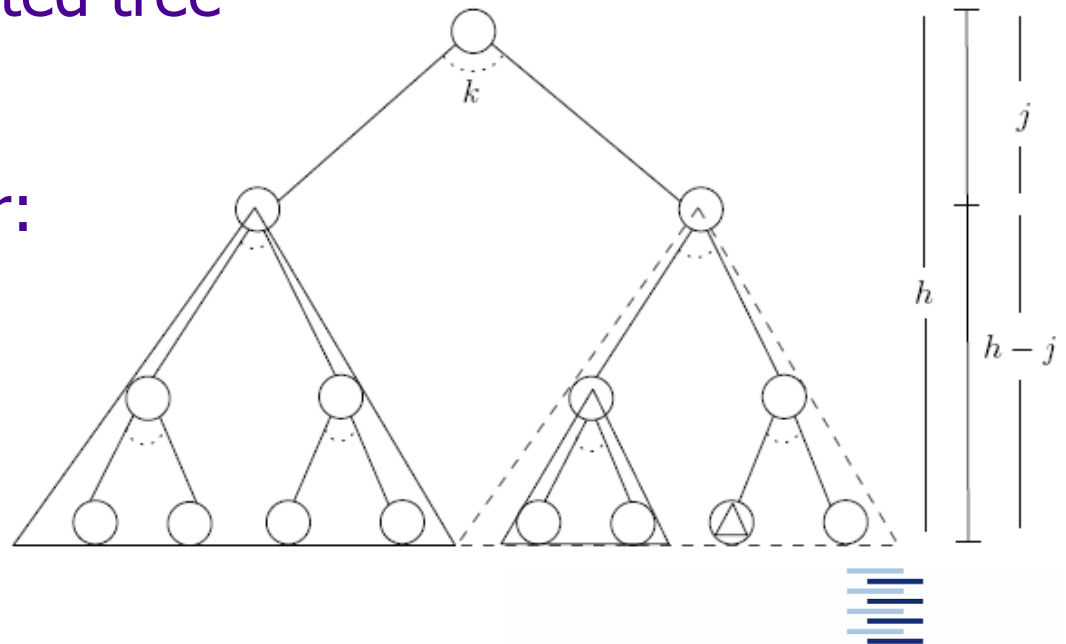


# Analysis of Prefix Flooding

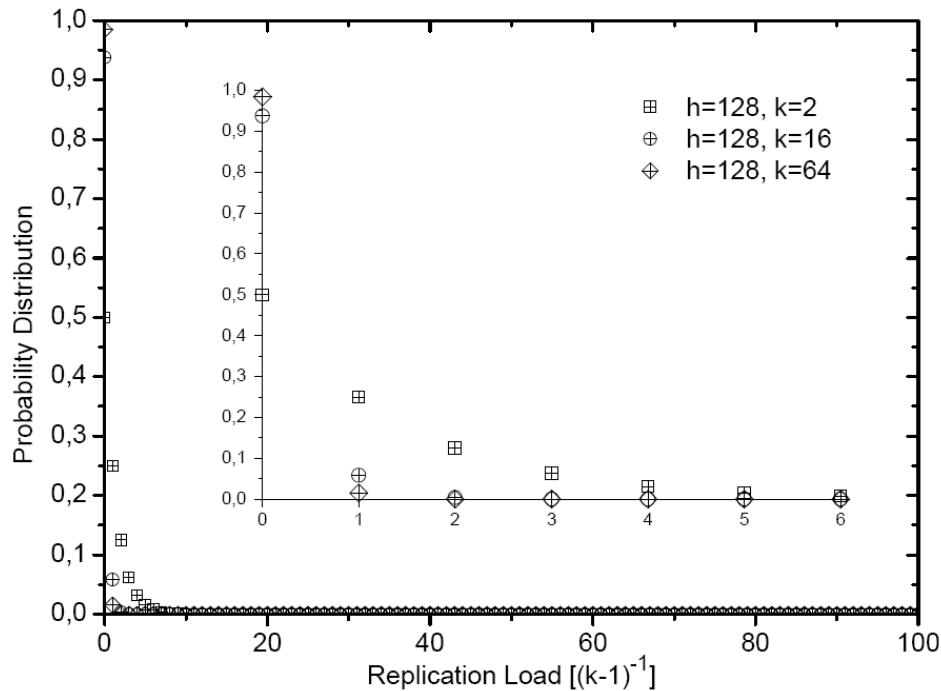
- Structural analysis relatively simple due to the recursive nature of k-ary trees
- Distinguish between fully and sparsely populated tree

Closed expressions for:

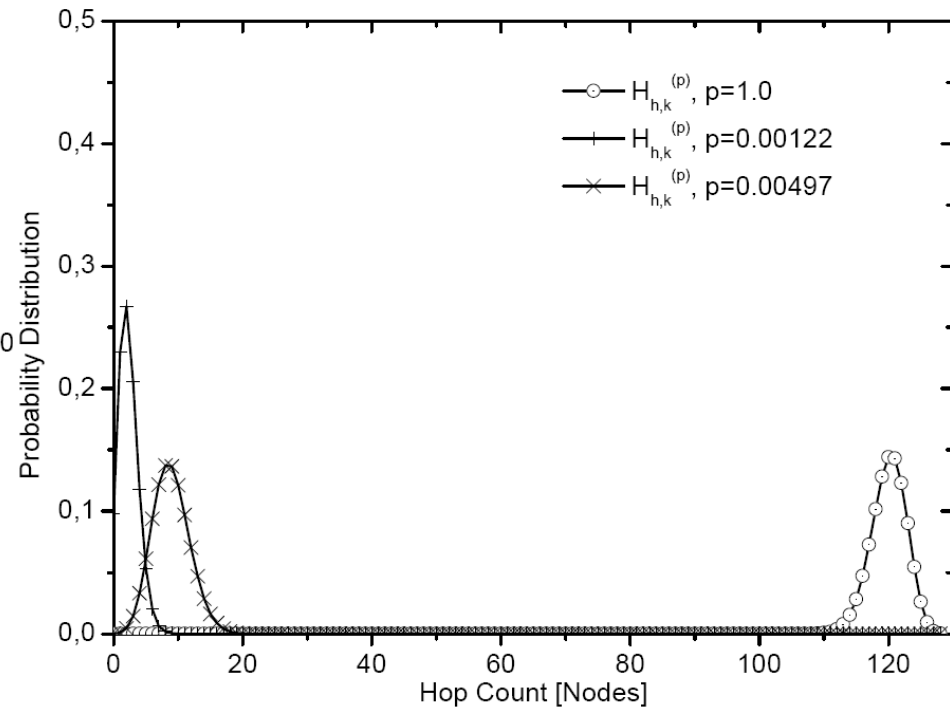
- Replication Load
- Hop Count



# Performance Values



p is sparseness parameter



# Summary on Flooding Approaches

- ▶ Defines a natural broadcast mechanism on the KBR
  - ▶ Transparent for sources & receivers:  
no signalling, no additional states
  - ▶ Problem of CAN: Duplicates & efficiency,  
solved with Prefix Flooding over Pastry
- ▶ Multicast requires construction of sub-DHTs
  - ▶ Group management based on DHT membership  
management
  - ▶ Tedious & slow – high overheads when updating  
routing tables



# Shared Distribution Tree: Scribe (Castro et al 2002)

- ▶ Large-scale distribution service based on Pastry
- ▶ Rendezvous Point chosen from Pastry nodes
  - ▶ Choice according to group key ownership
  - ▶ RP roots shared distribution tree (analogue PIM-SM)
- ▶ Shared tree created according to reverse path forwarding
  - ▶ Nodes hold *children tables* for forwarding
  - ▶ New receiver routes a *SUBSCRIBE* towards the RP
  - ▶ *Subscribe* intercepted by intermediate nodes to update children table, reverse forwarding done, if node not already in tree

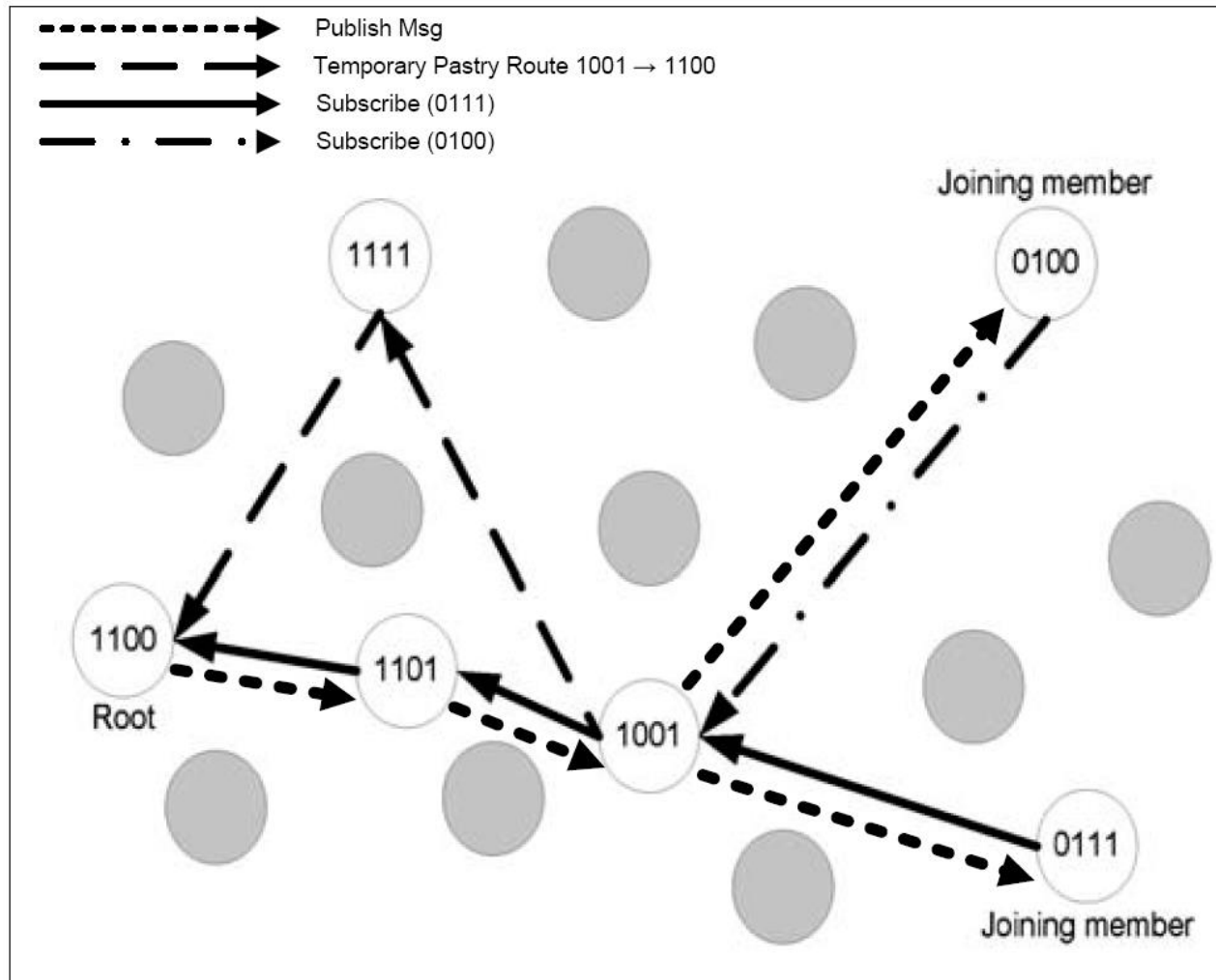


# Scribe API

- ▶ **Create (credentials, topicID):** Creates a group identified by a unique topicID (hash of textual description+creatorID), credentials administrative
- ▶ **Subscribe (credentials, topicID, eventHandler):** Initiates a local join to group, asynchronously received data passed to the eventHandler
- ▶ **Unsubscribe (credentials, topicID):** Causes a local leave of group
- ▶ **Publish (credentials, topicID, event):** Multicast source call for submitting data (event) to group

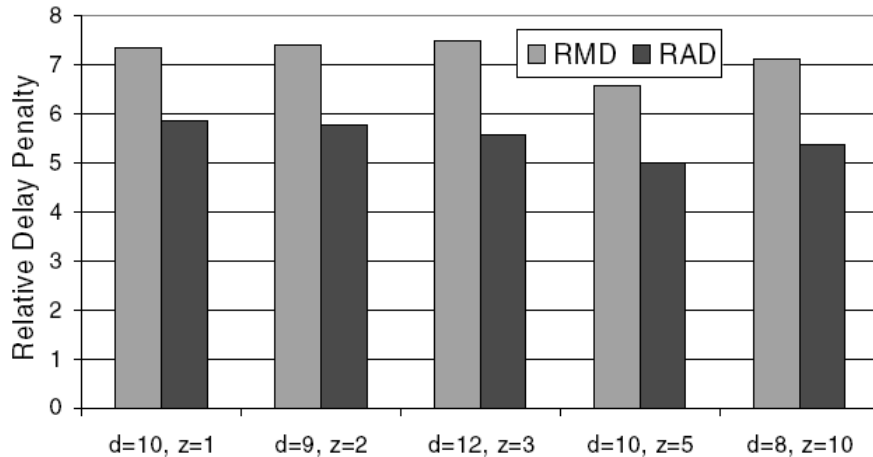


# Scribe Tree Construction

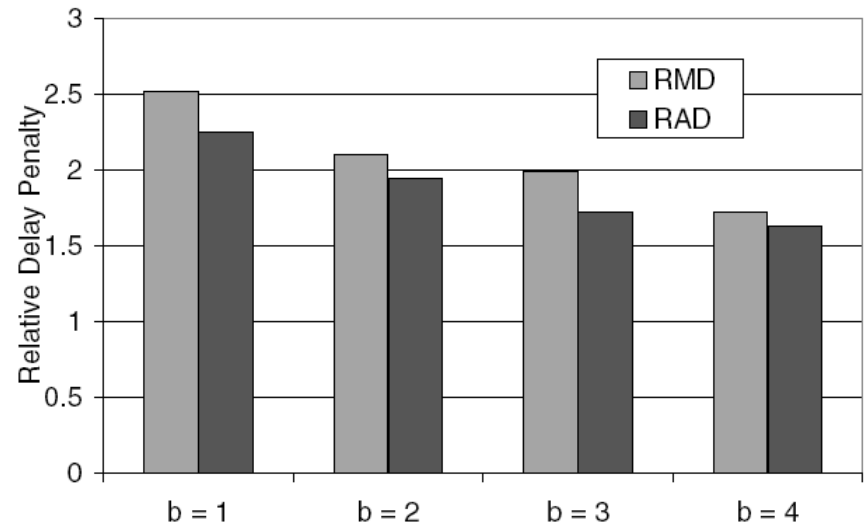




# Can versus Scribe: Delay Penalty



(a) CAN



(b) Scribe

RMD: Relative Delay Maximum

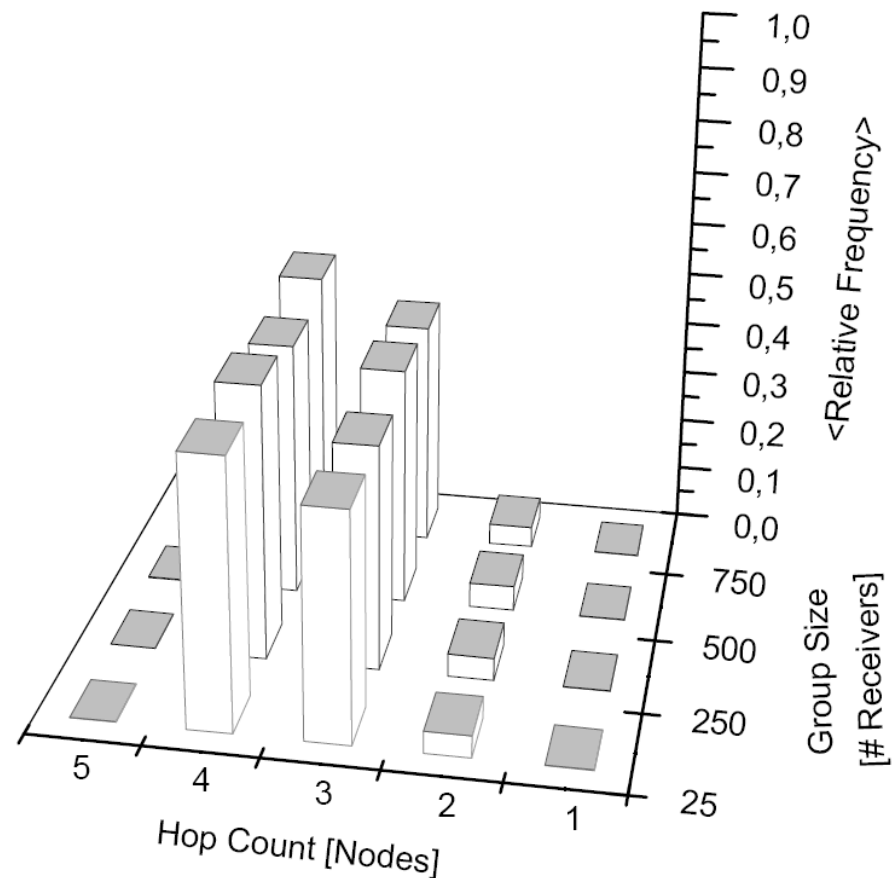
RAD: Relative Average Delay

CAN may be configured to provide higher network efficiency

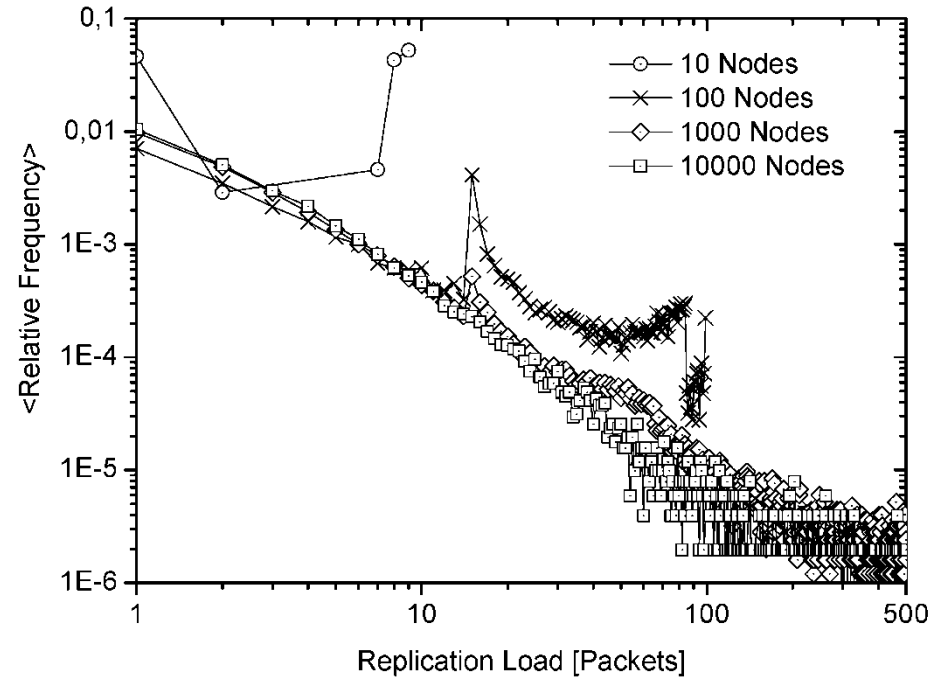
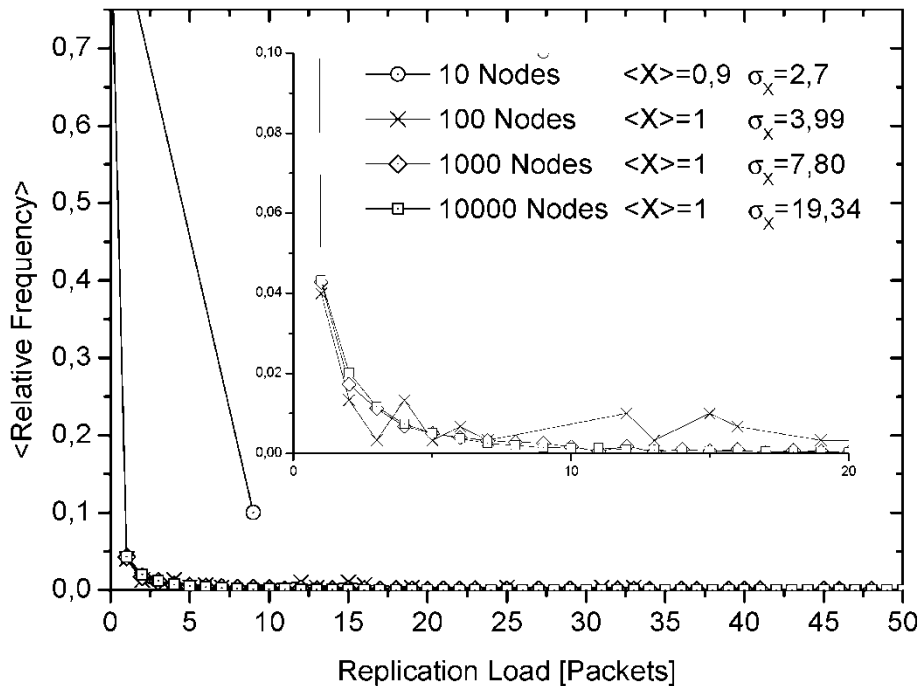


# Scribe Performance: Hop Count

- ▶ Simulation in OverSim network simulator
- ▶ 1.000 Pastry nodes
- ▶ Hop Count evaluated for varying group sizes



# Tree Characteristics in Scribe



- ▶ Almost all branches arise from Rendezvous Point
- ▶ Scribe foresees „manual“ load balancing



# Improvement: SplitStream (Castro et al. 2003)

- ▶ Focus on media data distribution
- ▶ Idea: Split media streams into slices and distribute sliced streams via disjoint trees
- ▶ Disjoint trees created by modifying prefix initial
  - ▶ Pastry leads to disjoint prefix routes
  - ▶ Scribe distribution trees according to prefix routes
  - ▶ All group members are leaves in all trees
- ▶ Accounts for member bandwidth constraints
- ▶ Problem: Jitter explosion



# Summary on Scribe/SplitStream

- ▶ Conventional approach to build ASM shared trees on the KBR (Key-based routing) layer

## *PROs*

- ▶ Autonomous identification of RP via keyspace
- ▶ Efficient group and tree management

## *CONS*

- ▶ Distribution trees lack efficiency because of the RP triangle and RPF at asymmetric unicast routing
- ▶ Highly unbalanced replication load at nodes
- ▶ High delay and jitter values



# PeerCast (Zhang et al. 2004)

- Multicast distribution service enhancing SCRIBE
  - Variation of PASTRY
  - Rendezvous-Point-based shared distribution tree
- Overlay structure adaptive to node capacities
- Landmark signatures to map proximity into key space
- Dynamic, passive replication scheme for reliable multicast distribution
- Two-tier approach:
  - ES Multicast Management
  - P2P Network Management



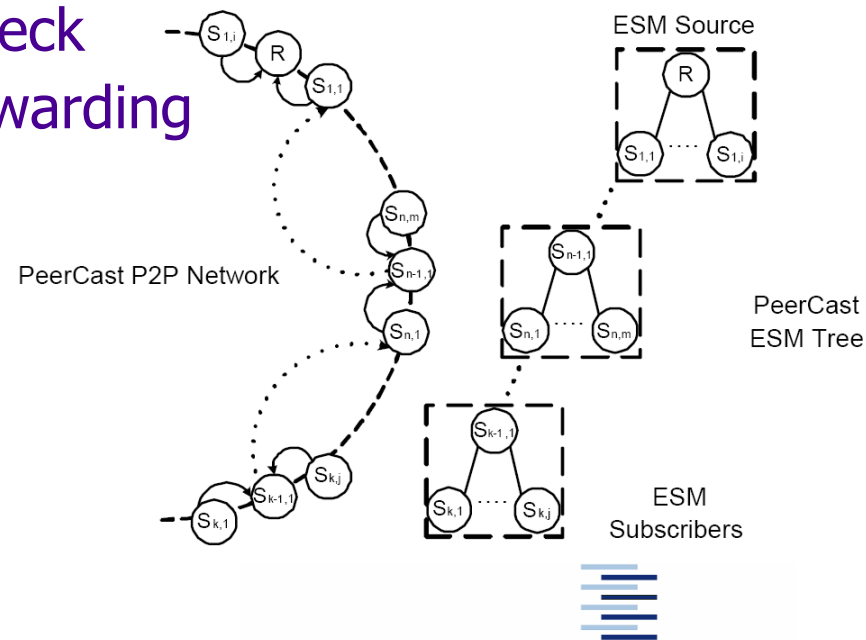
# PeerCast: P2P Management

- ▶ Proximity-aware DHT using landmarking
  - ▶ Landmark signature generated from distances to fixed landmark nodes
  - ▶ Landmark signature then substitutes a substring of each key identifier at the same "Splice Offset"
  - ▶ Neighbouring peers then clustered into "buckets"
- ▶ Accounting for node capabilities
  - ▶ Each node generates a multitude of keys, thus encountering multiple presence in the DHT ring
  - ▶ Key quantities are chosen according to node capabilities



# PeerCast: ES Multicast Management

- Rendezvous Node chosen as group key owner
- Shared tree created according to reverse path forwarding
- Improvement – **Neighbour Lookup:**
  - Subscribers + forwarders check their neighbours prior to forwarding subscription request
  - If any neighbour has already joined the group, a 'shortcut' is taken





# Performance of PeerCast

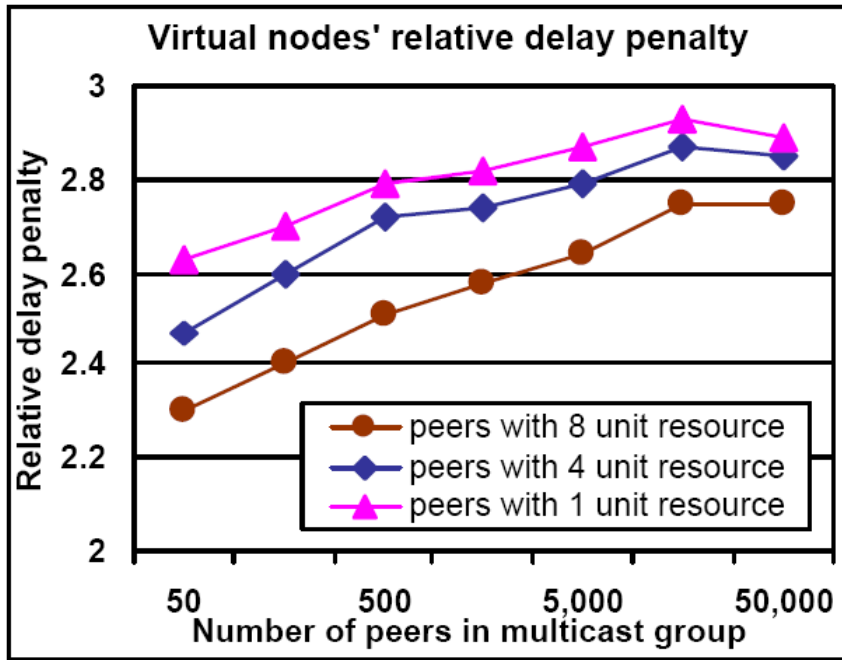


Figure 17: Relative delay penalty,  $r = 8$   
peers number = 50,000

$r$  is heterogeneity measure

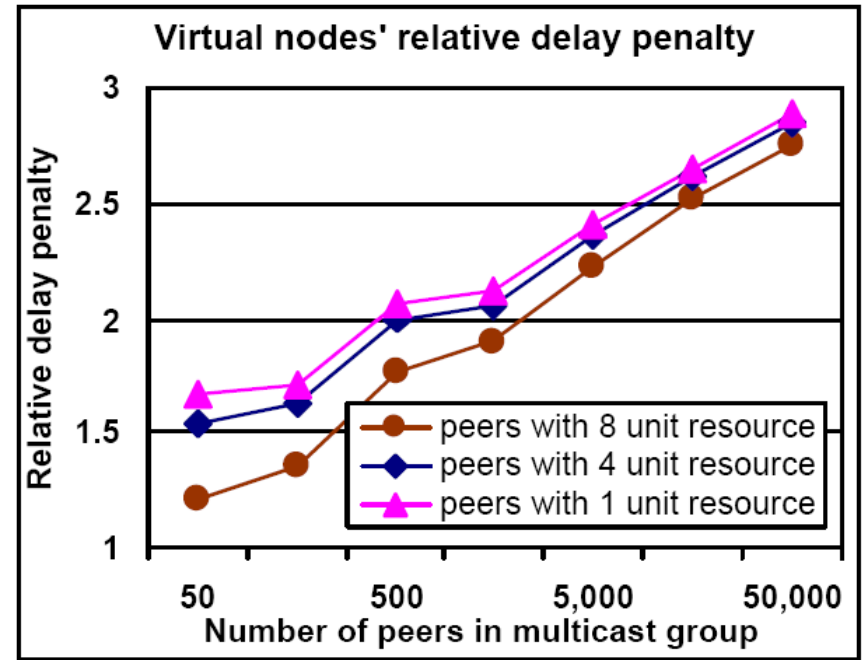


Figure 18: Relative delay penalty,  $r = 8$   
peers number = multicast group size



# Summary on PeerCast

- ▶ Interesting optimization of structured multicast
  - ▶ Introduces node capacity and neighborhood shortcuts

## *PROs*

- ▶ Improved ways of adaptation

## *CONs*

- ▶ Distribution trees still detour the RP triangle and use RPF at asymmetric unicast routing
- ▶ Unstable delay and jitter values



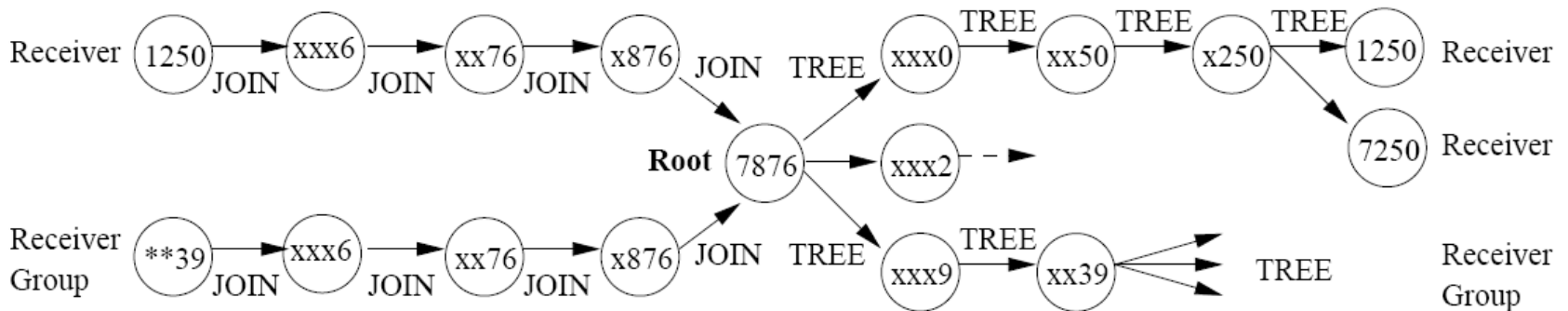
# Source Specific Distribution Tree: Bayeux (Zhuang et al, 2001)

- Based on Tapestry
- Creates a group by placing an empty file named by the hashed group ID
  - ▶ Announced by Tapestry location service
- Receivers learn about group ID and perform source-specific subscriptions
- Subscriptions are routed to the owner of the file, acting as the source & central controller
- Source (and intermediate branch nodes) perform full receiver tracking



# Bayeux Group Management

- Distribution tree is built according to (forward) pushed TREE messages

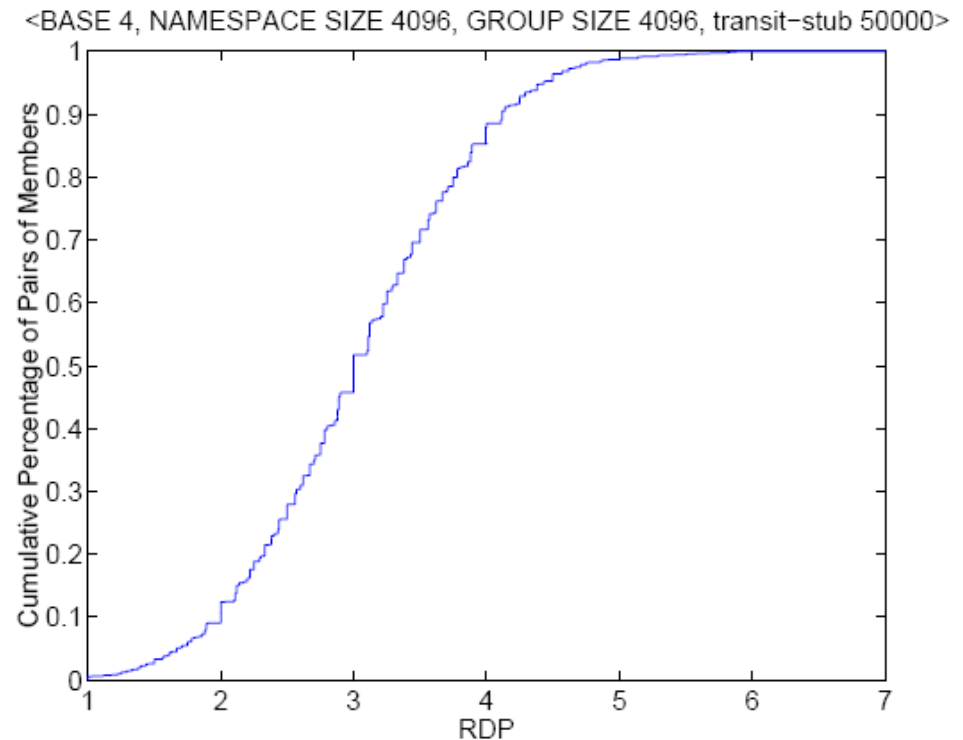


- Leaves are routed to the source and trigger a PRUNE message



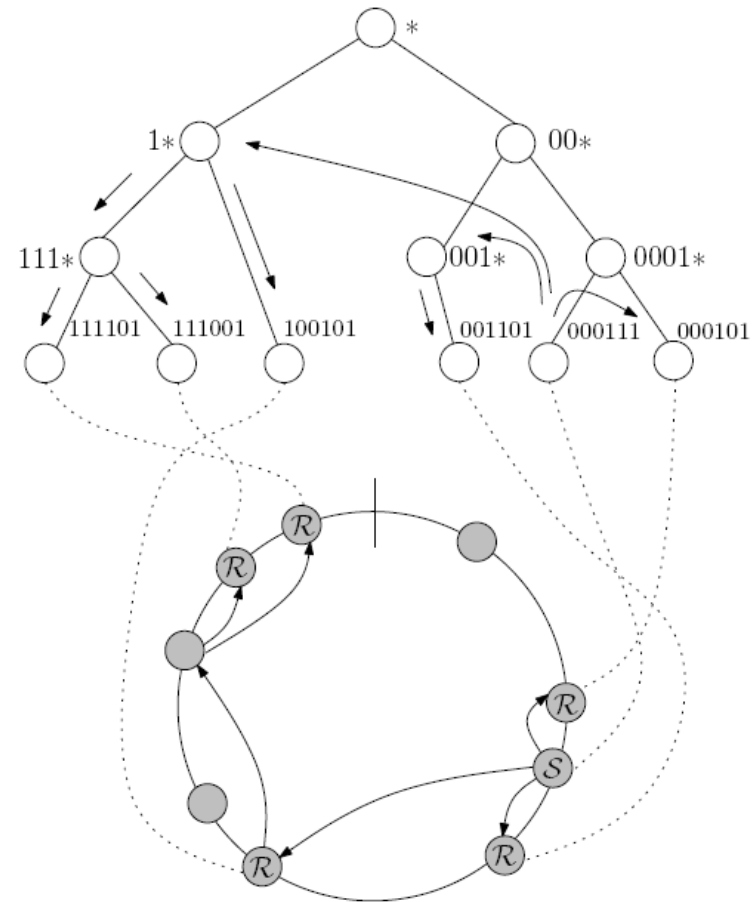
# Bayeux Performance

- ▶ Bayeux suffers from scaling problems due to the central controller
- ▶ Improvements are proposed to cluster receivers (hybrid) and to replicate via several roots

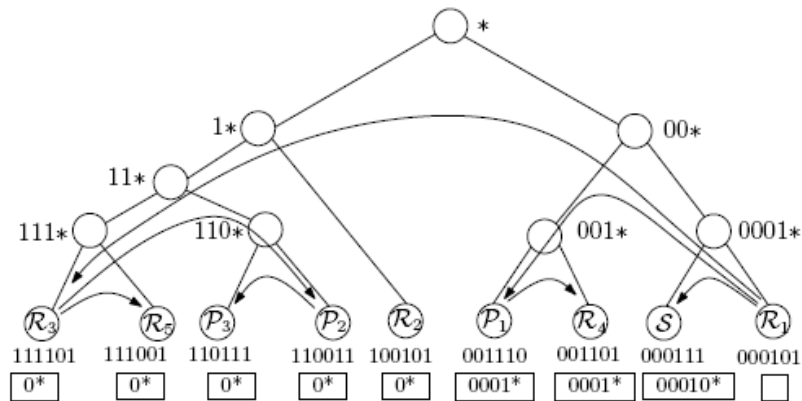


# Bidirectional Scalable Adaptive Multicast - BIDIR-SAM (Wählisch et al. 2007)

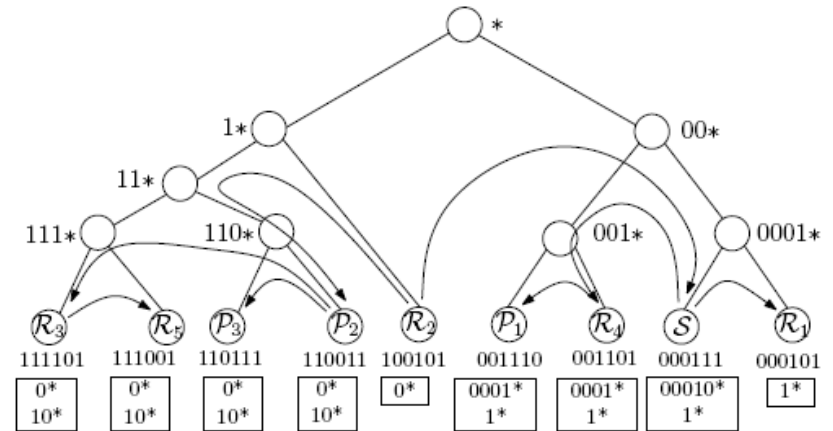
- Idea to build multicast in the key-based routing layer: Group distribution in a prefix overlay (on top of KBR)
- Nodes are represented in prefix trees (analogue to prefix flooding)
- Group management: State dissemination in prefix space
- Constructs source-specific shared trees (like Nice)



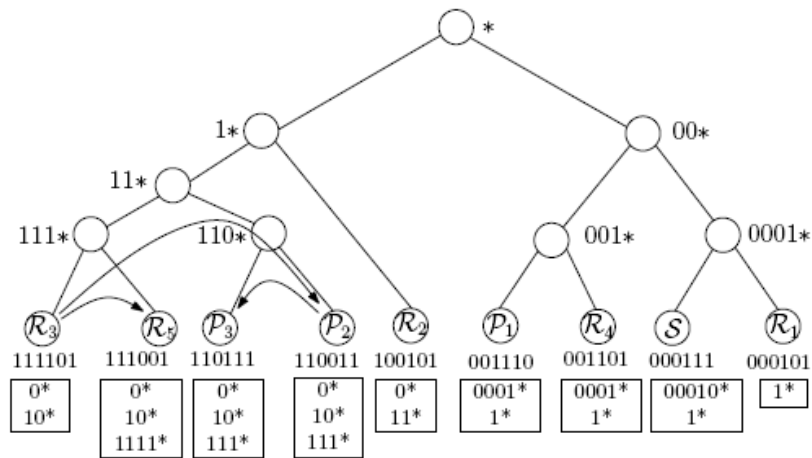
# Group Management



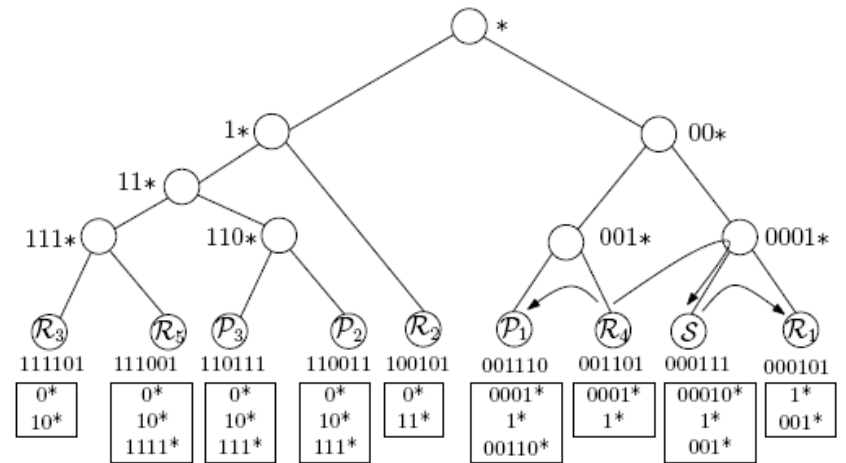
(a)  $\mathcal{R}_1$  joins  $G$



(b)  $\mathcal{R}_2$  joins  $G$



(c)  $\mathcal{R}_3$  joins  $G$



(d)  $\mathcal{R}_4$  joins  $G$

# Forwarding Along Virtual Prefix Tree

## BIDIR-SAM FORWARDING

- ▷ On arrival of packet with destination prefix  $\mathcal{C}$
- ▷ for group  $G$  at DHT node of ID  $\mathcal{K}$

1 **for** all  $\mathcal{N}_i$  IDs in  $MFT_G$

2       **do if**  $LCP(\mathcal{C}, \mathcal{N}_i) = \mathcal{C}$

- ▷  $\mathcal{N}_i$  is dountree neighbor

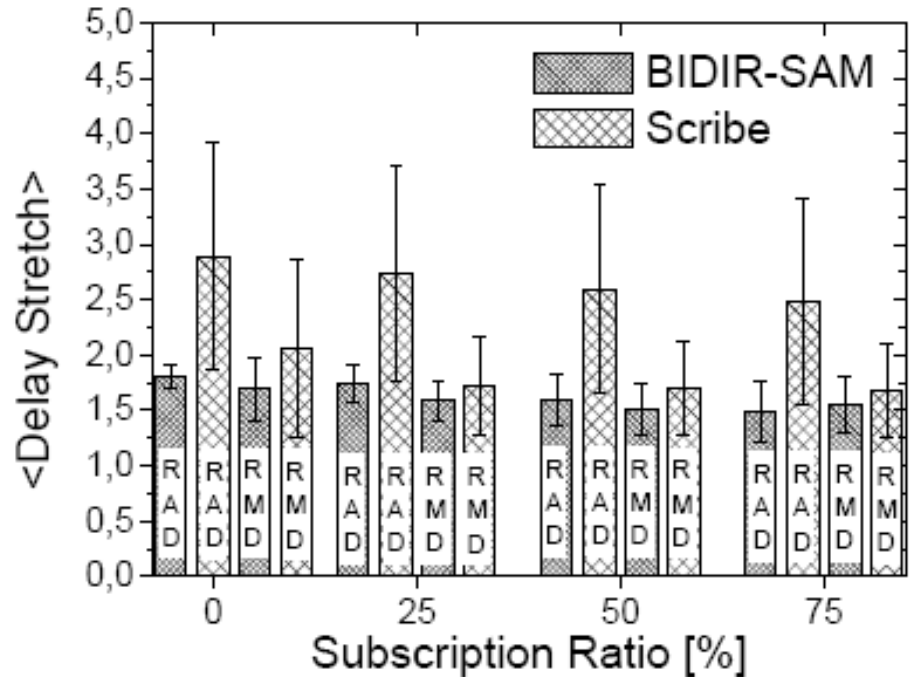
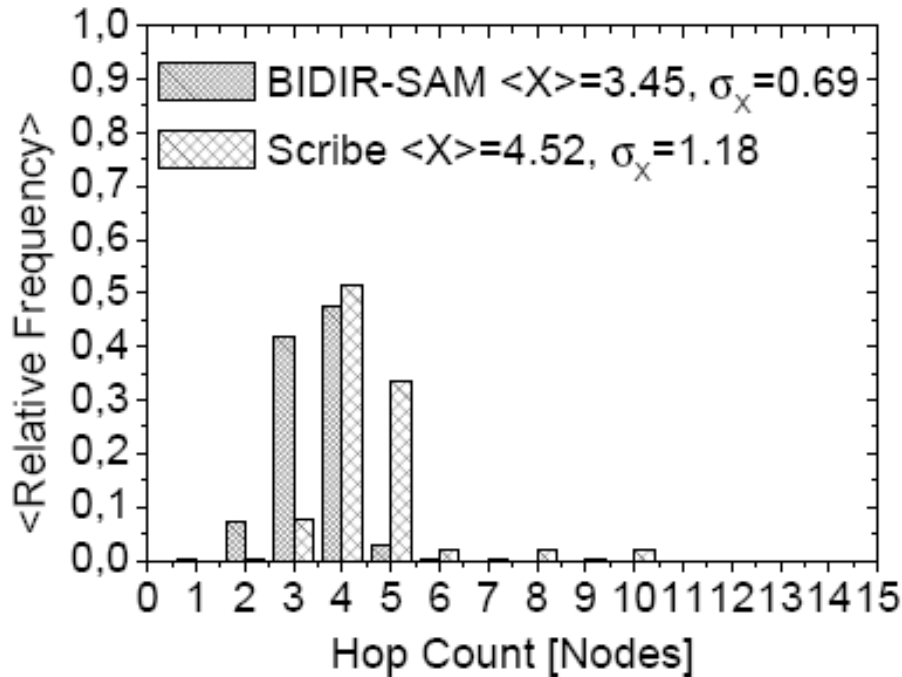
3       **then**  $\mathcal{C}_{new} \leftarrow \mathcal{N}_i$

4                               FORWARD PACKET TO  $\mathcal{C}_{new}$

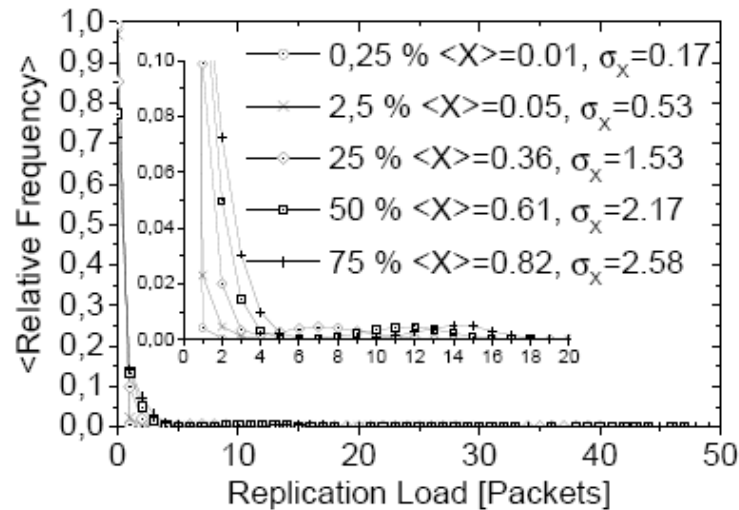




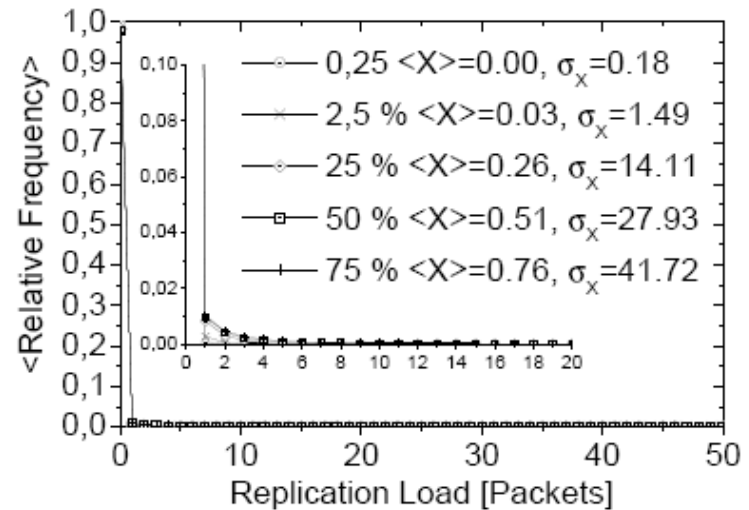
# BIDIR-SAM Performance



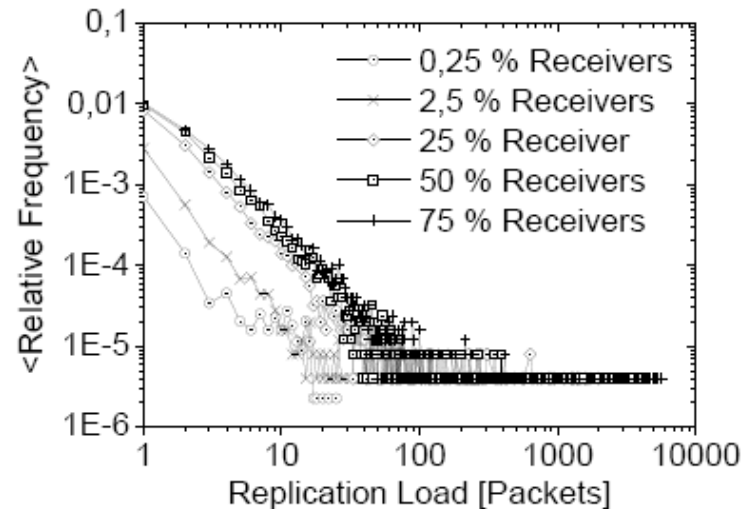
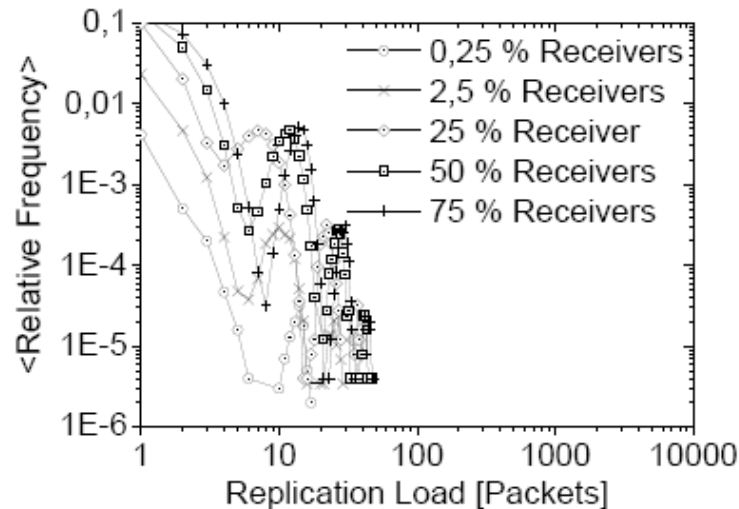
# BIDIR-SAM Performance



(a) BIDIR-SAM

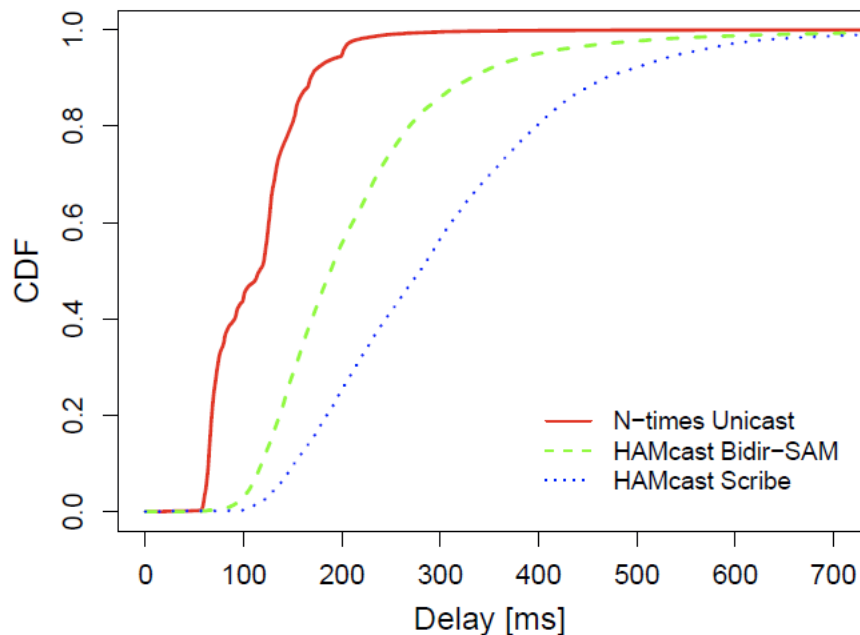


(b) Scribe

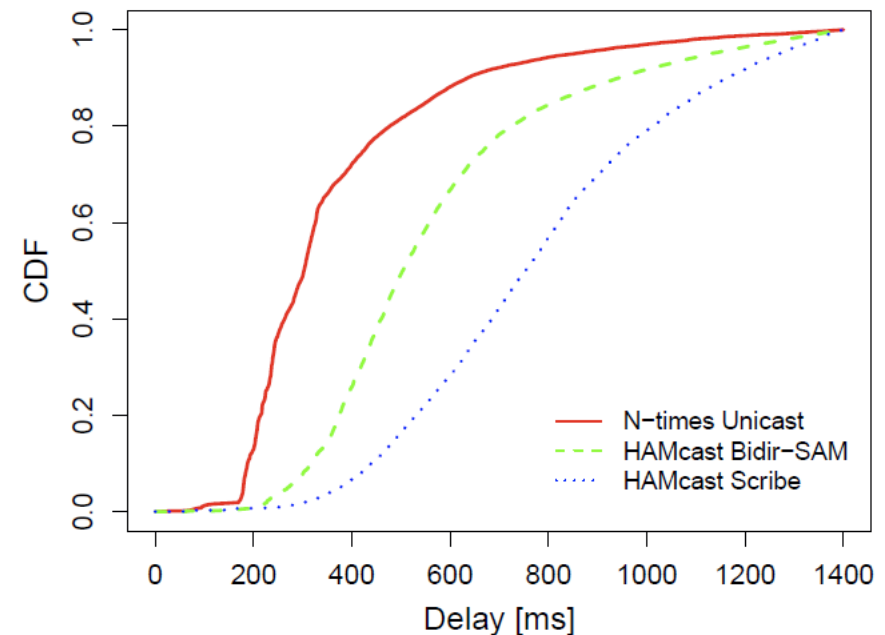


# Large-scale Measurements: Globally Distributed Delay Space

(a) One-Way Average Delays



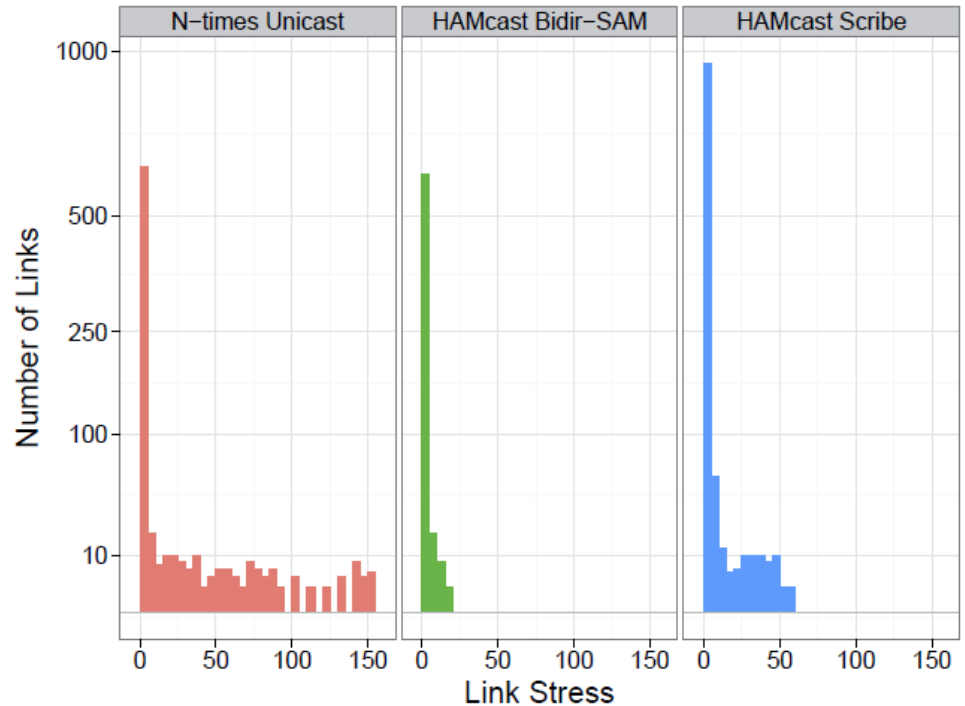
(b) One-Way Maximum Delays



► 250 Nodes in Planet-Lab on all continents



# Large-scale Measurements (cont.)



# Additional Design Mechanisms

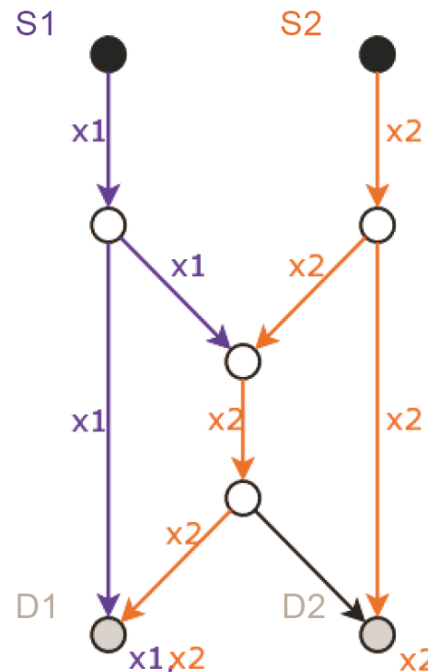
Two core problems arise in wide-area broadcast/multicast distribution:

- ▶ Reliability and redundancy without retransmission
  - ▶ In particular for file distribution: all blocks are needed
  - ▶ Promising approach: Network Coding
- ▶ Flow control / flow adaptation in heterogeneous environments
  - ▶ Data streams may meet network bottlenecks
  - ▶ Promising approach: Selective dropping after Backpressure Control

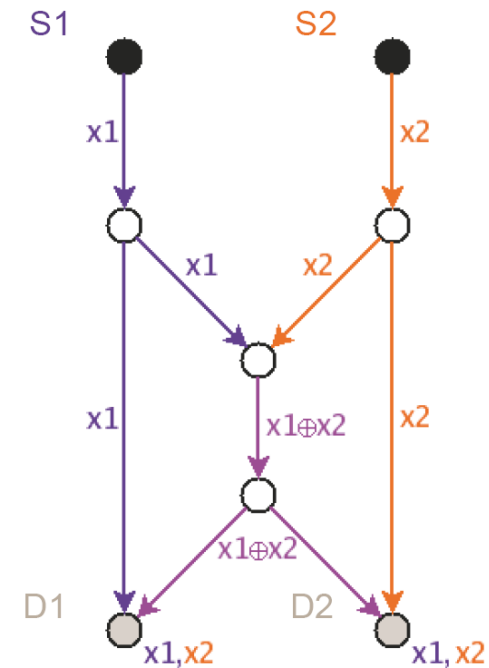


# Network Coding (Li, Yeung, Cai, 2003)

- Original idea: network efficiency can be enhanced by linear combination of packets
- Useful in Wireless transmission to enhance efficiency
- In Overlay Multicast mainly to add „universal“ redundancy

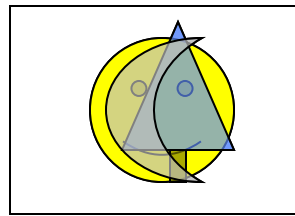


Max. broadcast rate = 1.5



Max. broadcast rate = 2

# Network Coding Simplified

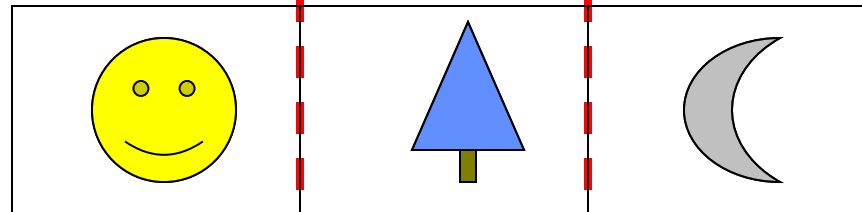


Encoding

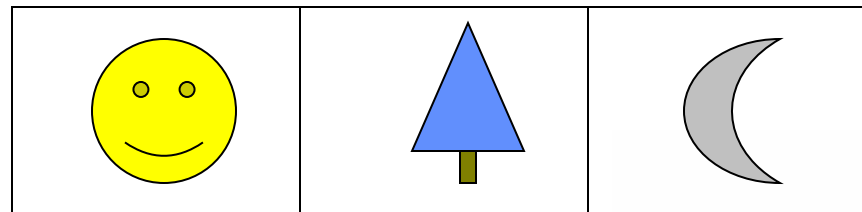
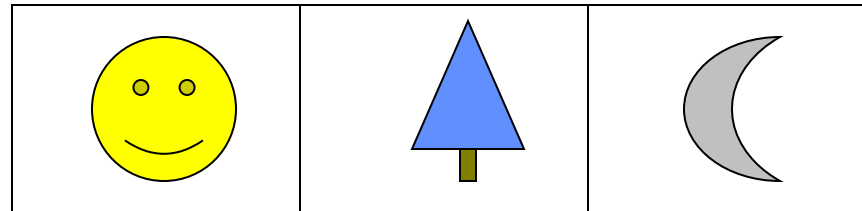
Block 1

Block 2

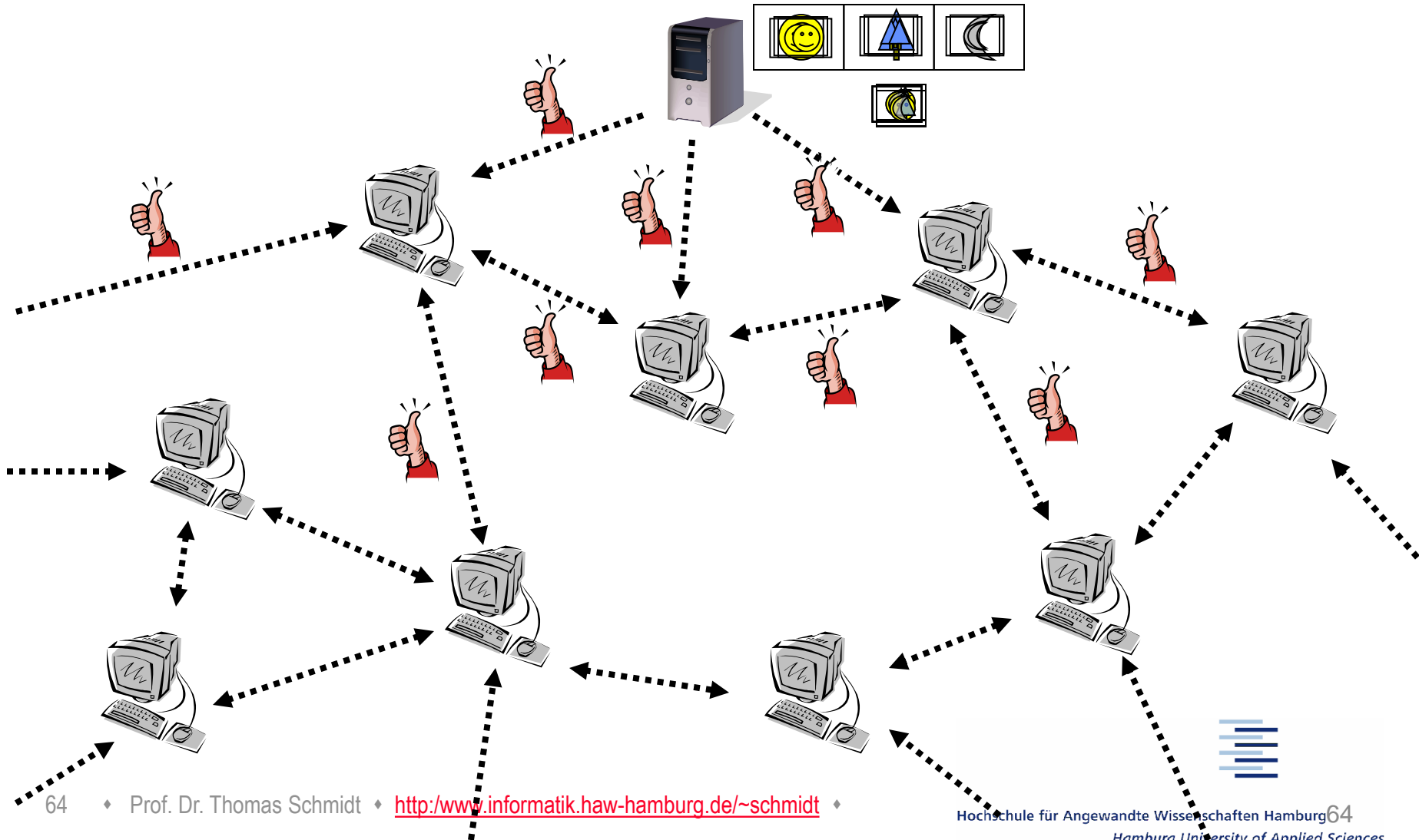
Block 3



File to Transfer



# With Network Coding



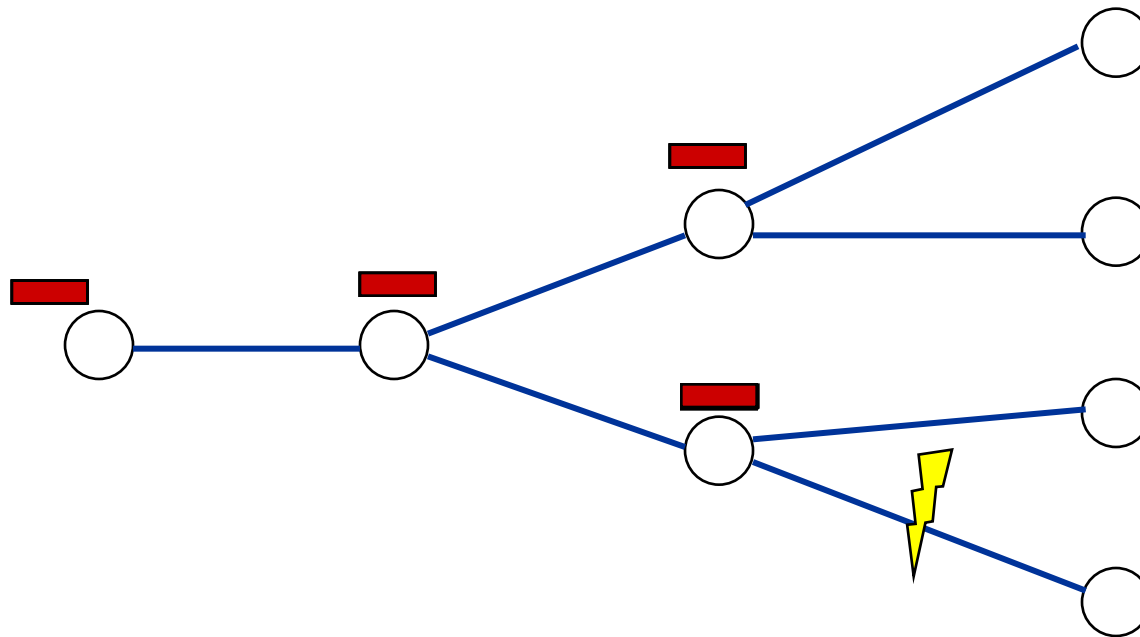


# Problem of Flow Control

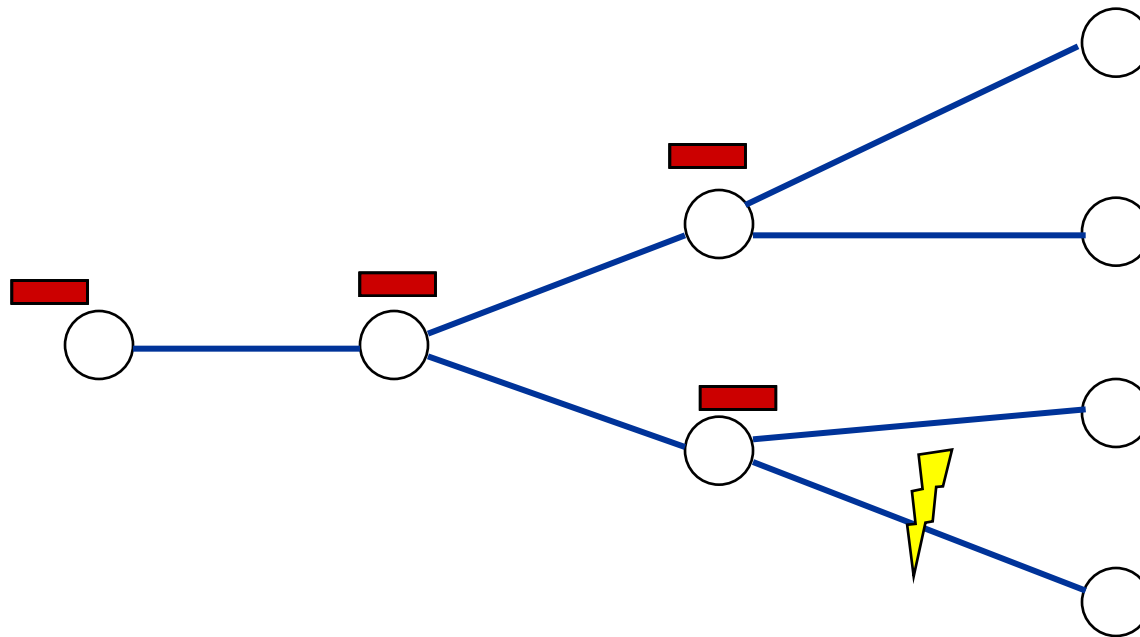
- ▶ In a distribution system (e.g., Tree) there may occur at some part
  - ▶ Heterogeneous link transitions
  - ▶ Congestions
  - ▶ Fluctuating link conditions
- ▶ Problems
  - ▶ Long-range (e.g., receiver) feedback prevents scaling
  - ▶ How to decide locally on efficient flow forwarding (omit forwarding packets that are discarded later)?



# Group Distribution without Flow Control



# Backpressure Multicast: Simple Flow Control



- Intermediate Node can decide about dropping or delaying



# Programming: Unique Interface

draft-irtf-samrg-common-api

## ► Send and receive calls

```
createMSocket(out SocketHandle h, [in enum Interface  
i])
```

```
join(in SocketHandle h, in URI g, [in Interface i])
```

```
leave(in SocketHandle h, in URI g, [in Interface i])
```

```
srcRegister(in SocketHandle h, in URI g, [out  
Interface i])
```

```
send(in SocketHandle h, in URI g, in Message msg)
```

```
receive(in SocketHandle h, out URI g, out Message  
msg)
```

## ► Service calls

## ► Socket option calls

# URI-based Naming Scheme

**scheme "://" group "@" instantiation  
":" port "/" sec-credentials**

- ▶ **scheme**: specification of assigned ID
- ▶ **group**: identifies the group
- ▶ **instantiation**: ID of the entity that generates the instance of the group (SSM source, RP, overlay node)
- ▶ **port**: ID of a specific application at a group instance
- ▶ **sec-credentials**: optional authentication

## Examples:

**ham:opaque:news@cnn.com/auth-value**

**ham:ip:224.10.20.30@1.2.3.4:5000/groupkey**



# Research Issues

- ▶ Joined / combined / hybrid solutions for a global group communication layer
- ▶ Redundancy & robustness enhancements by Network Coding
- ▶ Multipath transport without jitter explosion
- ▶ Proximity under mobility – Constructions of distributions trees efficient w.r.t. the underlay topology
- ▶ Stability under mobility – Construction of efficient multicast distribution trees, which are robust
- ▶ QoS improvements & flow control, measures and guaranties to provide real-time capabilities
- ▶ Security & Robustness against malicious node behaviour

# References

- K. Katrinis, M. May: *Application-Layer Multicast*, in Springer LNCS 3485, 2005.
- Y. Chu, S. G. Rao, and H. Zhang: *A Case for End System Multicast*, Proceedings of ACM SIGMETRICS, Santa Clara, CA, June 2000.
- S. Banerjee, B. Bhattacharjee, C. Kommareddy: *Scalable Application Layer Multicast*, SIGCOMM Pittsburgh, PA 2002.
- S. Ratnasamy, M. Handley, R. Karp, S. Schenker: *Application-Level Multicast using Content-Addressable Networks*, Proc. 3<sup>rd</sup> Intern. Workshop on Networked Comm., London, Nov. 2001.
- M. Castro, P. Druschel, A. Kermarrec, A. Rowstron: *SCRIBE: A large-scale and decentralized application-level multicast infrastructure*, IEEE Journ. Select. Areas in Comm., 20 (8), Oct 2002.
- M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron and A. Singh: *SplitStream: High-bandwidth multicast in a cooperative environment*, SOSP'03, Lake Bolton, New York, October, 2003.
- J. Zhang, L. Liu, C. Pu, M. Ammar: *Reliable End System Multicast with a Heterogeneous Overlay Network*. CERCS Technical Report git-cercs-04-19, Georgia Institute of Technology, April 2004.



# References

- S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz: *Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination*, in NOSSDAV '01: ACM, 2001, pp. 11-20
- M. Wählisch, T. C. Schmidt, G. Wittenburg: *BIDIR-SAM: Large-Scale Content Distribution in Structured Overlay Networks*, Proc. of the 34th IEEE Conference on Local Computer Networks (LCN), pp 372—375, IEEE Press, October 2009.
- M. Wählisch, T. C. Schmidt, G. Wittenburg: *On Predictable Large-Scale Data Delivery in Prefix-based Virtualized Content Networks*, Computer Networks, 55 (2011) 4086–4100.
- M. Wählisch, T. C. Schmidt: *Multicast Routing in Structured Overlays and Hybrid Networks*, in Shen, Yu, Buford: *Handbook of Peer-to-Peer Networking*, pp. 897--932, Springer, January 2010.
- S. Meiling, T. C. Schmidt, M. Wählisch: *Large-Scale Measurement and Analysis of One-Way Delay in Hybrid Multicast Networks*, Proc. of the 37th IEEE Conference on Local Computer Networks (LCN), IEEE Press, October 2012.
- M. Wählisch, T. C. Schmidt, S. Venaas: *A Common API for Transparent Hybrid Multicast*, IRTF Internet Draft -- work in progress, No. 10, October 2013.