Scalable Distributed Systems
Application-Level Multicast

Davide Frey
ASAP Team
INRIA

Group Communication

Common and useful communication paradigm

Disseminating information within a group sharing interest

- Consistency of replicated data
- Publish/Subscribe systems

Studied a lot in local area networks

- Group management (join, leave, send)

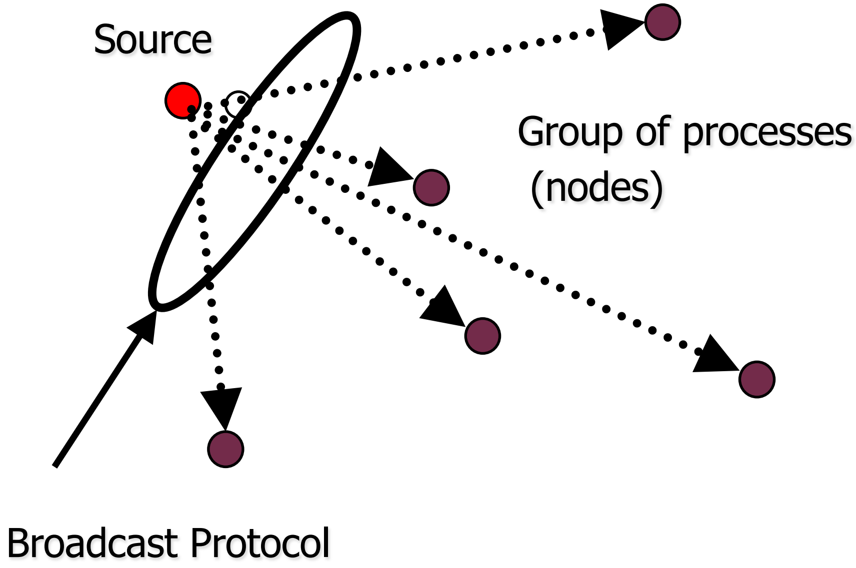
More scalability needed

- Application-level multicast (for medium-size groups) not scalable
- Network-level multicast not fully deployed

Group communication

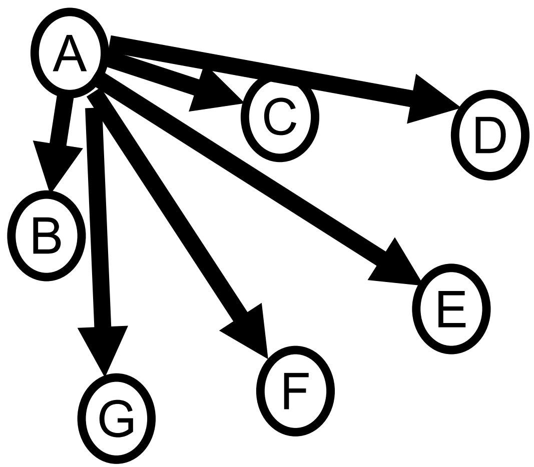
- Important functionality of distributed systems
  - Failure detection
  - Membership management
  - Coherence management
  - Event notification systems

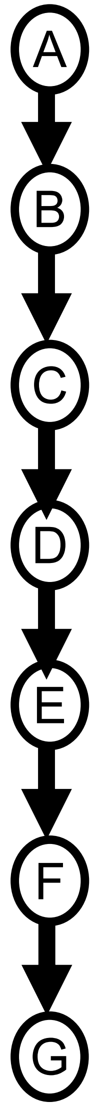
- Crucial Features
  - Reliability
  - Scalability
    - System size
    - Group size



Broadcast protocols

- Centralized versus decentralized protocols
  - Load balancing
  - Performance
- Evaluation metrics
  - Delay from source to each destination
  - Network traffic
  - Node load
  - Failure resilience





Large-scale broadcast/multicast

Application-level multicast (ALM)

1. Structured peer to peer networks (today)
   - Flooding
   - Tree-based
2. Content streaming (later)
   - Multiple Trees
   - Mesh
   - Gossip

Structured overlay networks

Scalability

- O(logN) hops routing with a O(logN) state
- Load balancing

Self-* properties (organizing, healing, …)

- P2P overlay network automatically repaired upon peer joins and departures
- Automatic load re-distribution

Attractive support for large-scale application-level multicast

ALM on structured overlay networks

- Overlay network used for
  - group naming
  - group localization
- Flooding-based multicast [CAN multicast]:
  - Creation of a specific network for each group
  - Message flooded along the overlay links
- Tree-based multicast [Bayeux, Scribe]
  - Creation of a tree per group
  - Flooding along the tree branches

Flooding-based multicast

- Group members join the network associated with a group
- Messages sent over all links of the P2P overlay
- Specific mechanism to get rid of duplications

- Example:

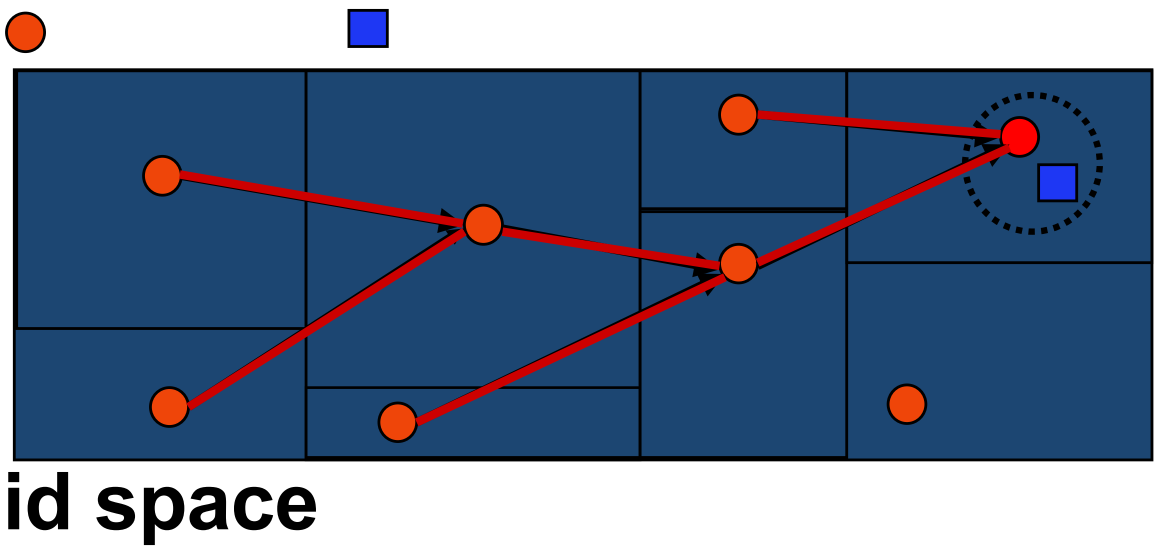
  message *m* in Pastry
  - on receiving <*flood, m, i*>
  - *i=0* for original message sender
  - for each routing table row *i'* (*i'* greater than *i*)

    send <*flood, m, i'*> to nodes in row

Tree-based multicast

Creation of a tree per group

- The tree root is the peer hosting the key associated with that group
- The tree is formed as the union of routes from every member to the root



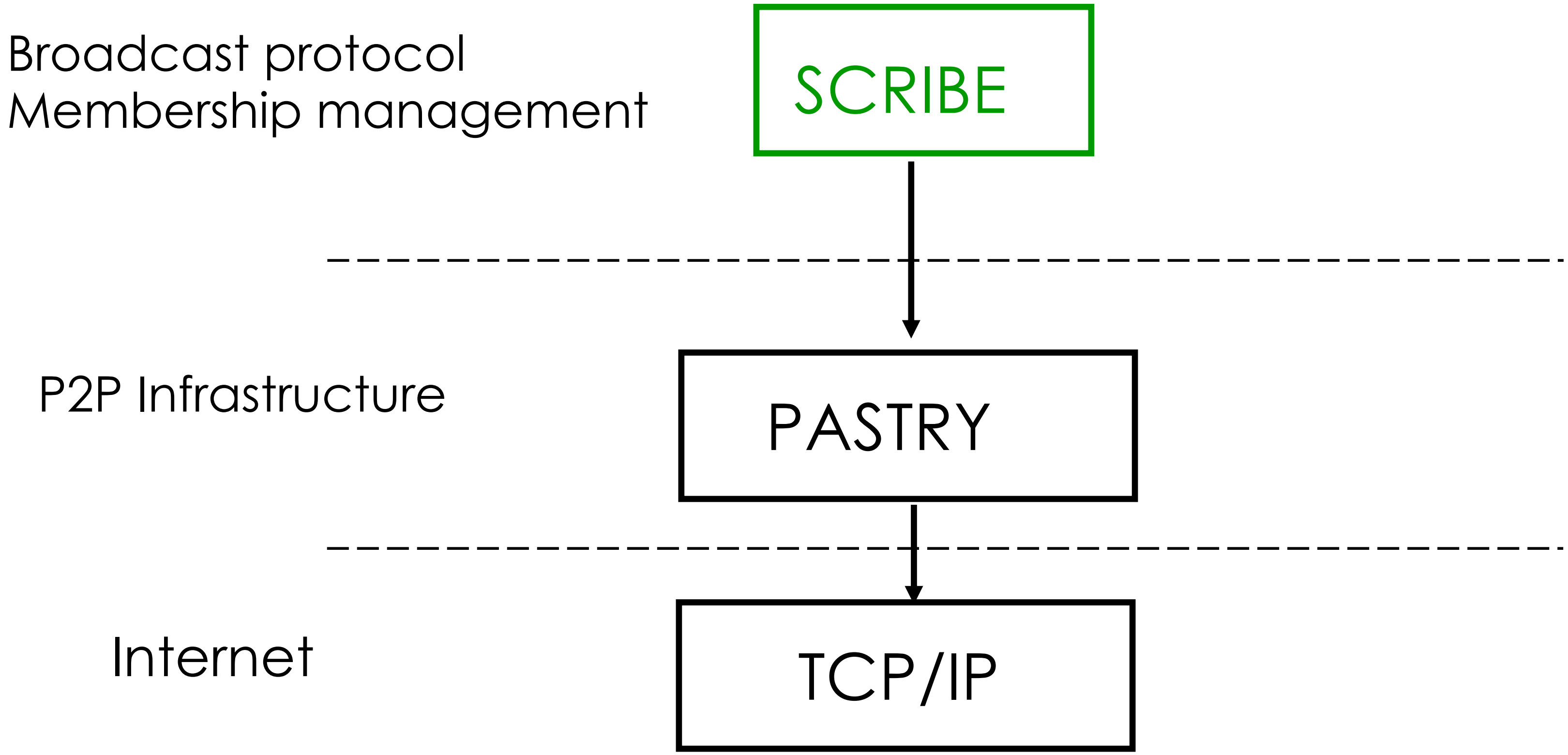
The two original examples:

- Scribe
  - Tree on Pastry
- CAN Multicast
  - Flooding on CAN

Scribe

- Multiple groups on a p2p prefix-matching infrastructure (Pastry, Tapestry,…)
- Support several applications on a single infrastructure
  - Instant Messaging
  - Information dissemination (stock alerts)
  - Diffusion lists (Windows updates)
- Properties
  - Scalability
  - Efficiency: low latency, low network link stress, low node load
  - Reliability: application-specific

Scribe


Broadcast protocol
Membership management
SCRIBE
P2P Infrastructure
PASTRY
Internet
TCP/IP

Scribe: interface

Goals

- Group creation
- Membership maintenance
- Messages dissemination within a group

Operations

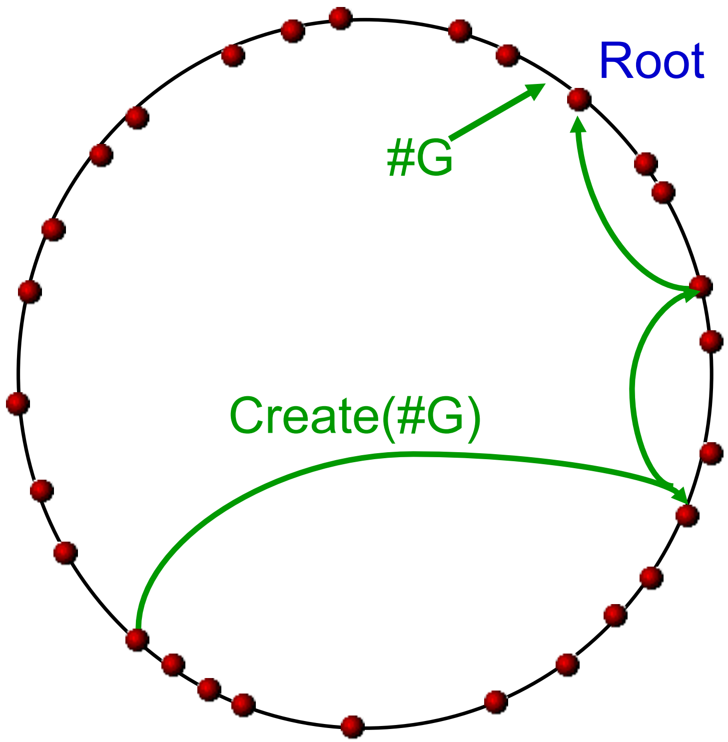
- Create(group)
- Join(group)
- Leave(group)
- Multicast(group,m)

Scribe Design

Use pastry-like P2P infrastructure

- Group creation and join protocol
  - Construct Multicast Tree
  - Establish reverse path forwarding
- Message dissemination
  - Flood messages along tree branches

Scribe: group creation



- Each group is assigned an identifier *groupId = Hash(name)*
- Multicast tree root : node whose nodeId is the numerically closest to the groupId
- **Create(group):** P2P routing using the *groupeId* as the key

Scribe: tree creation

join(group) : message sent through Pastry using

groupeId as the key

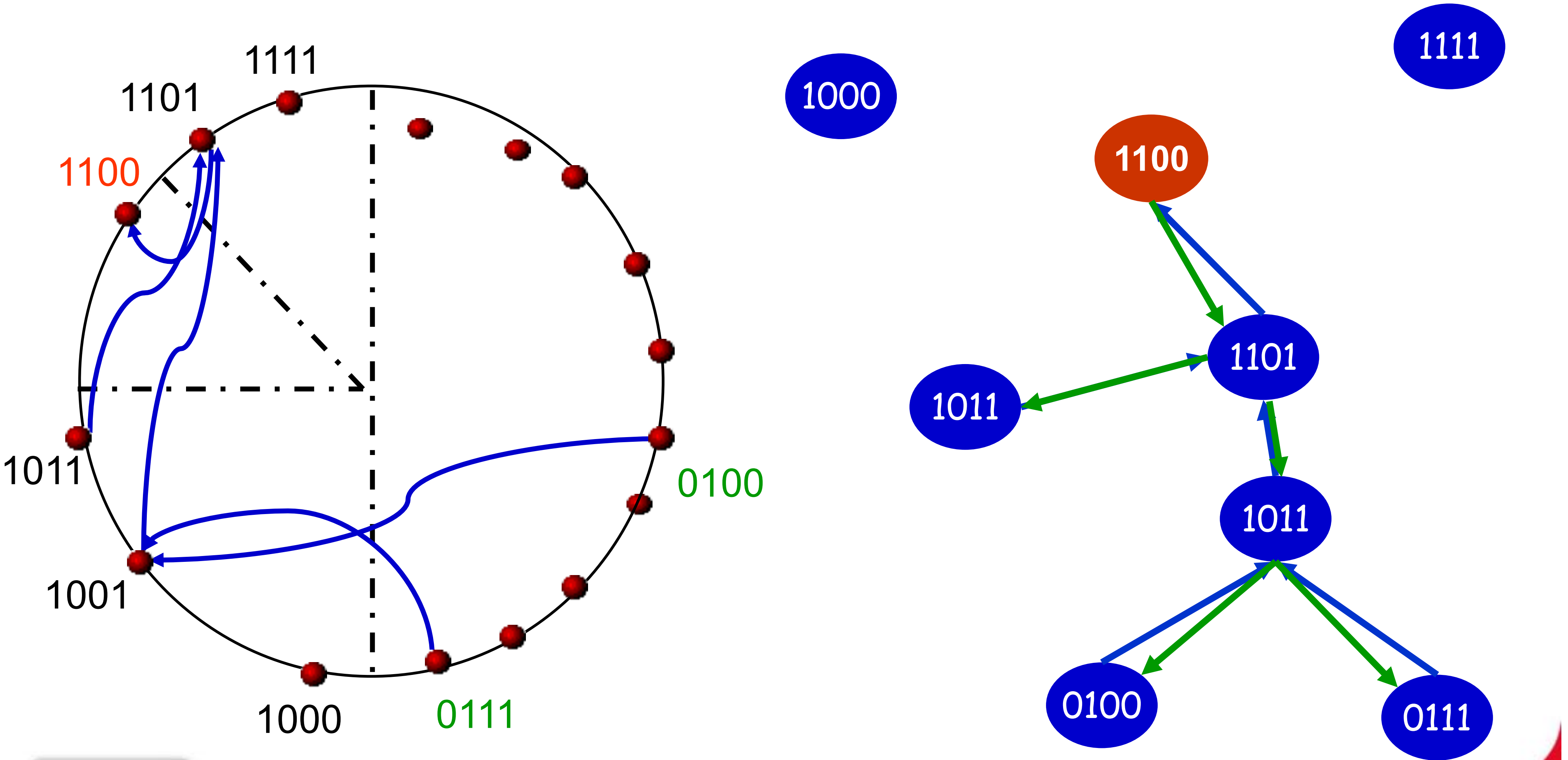
Multicast tree : union of Pastry routes from the root to

each group

- Low latency: leverage Pastry proximity routing
- Low network link stress: most packets are replicated

low in the tree

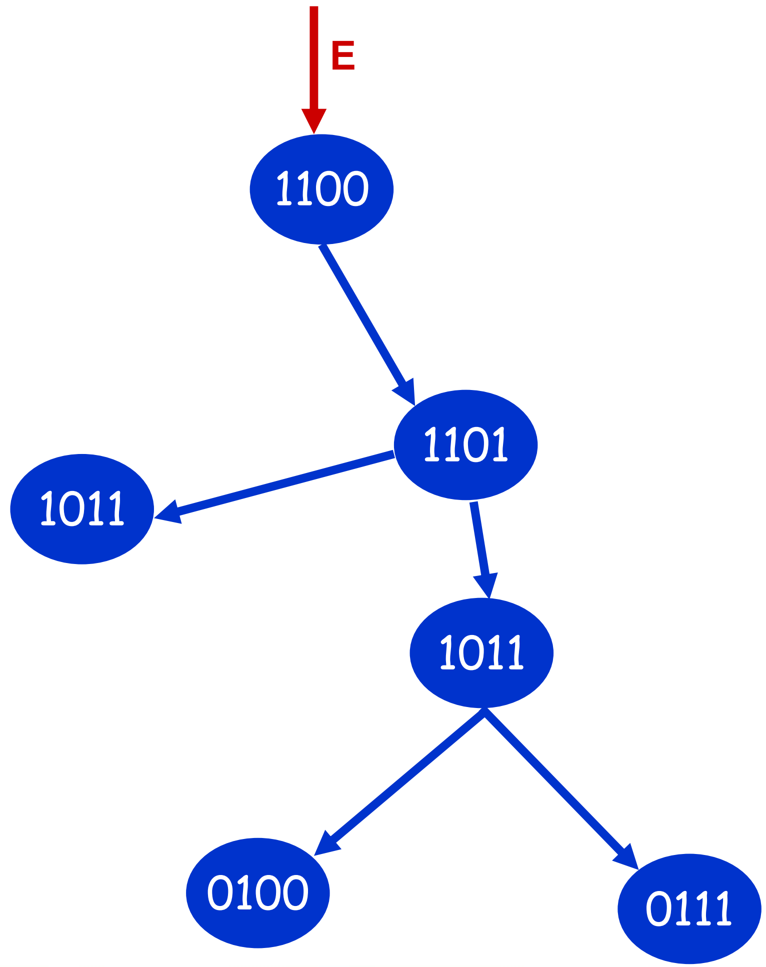
Scribe : join(group)



Scribe: message dissemination

Multicast(group, m)

- Routing through Pastry to the root key=*groupeId*
- Flooding along the tree branches from the root to the leaves



Reliability

« best effort » reliability guarantee

- Tree maintenance when failures are detected
- Stronger guarantee may also be implemented

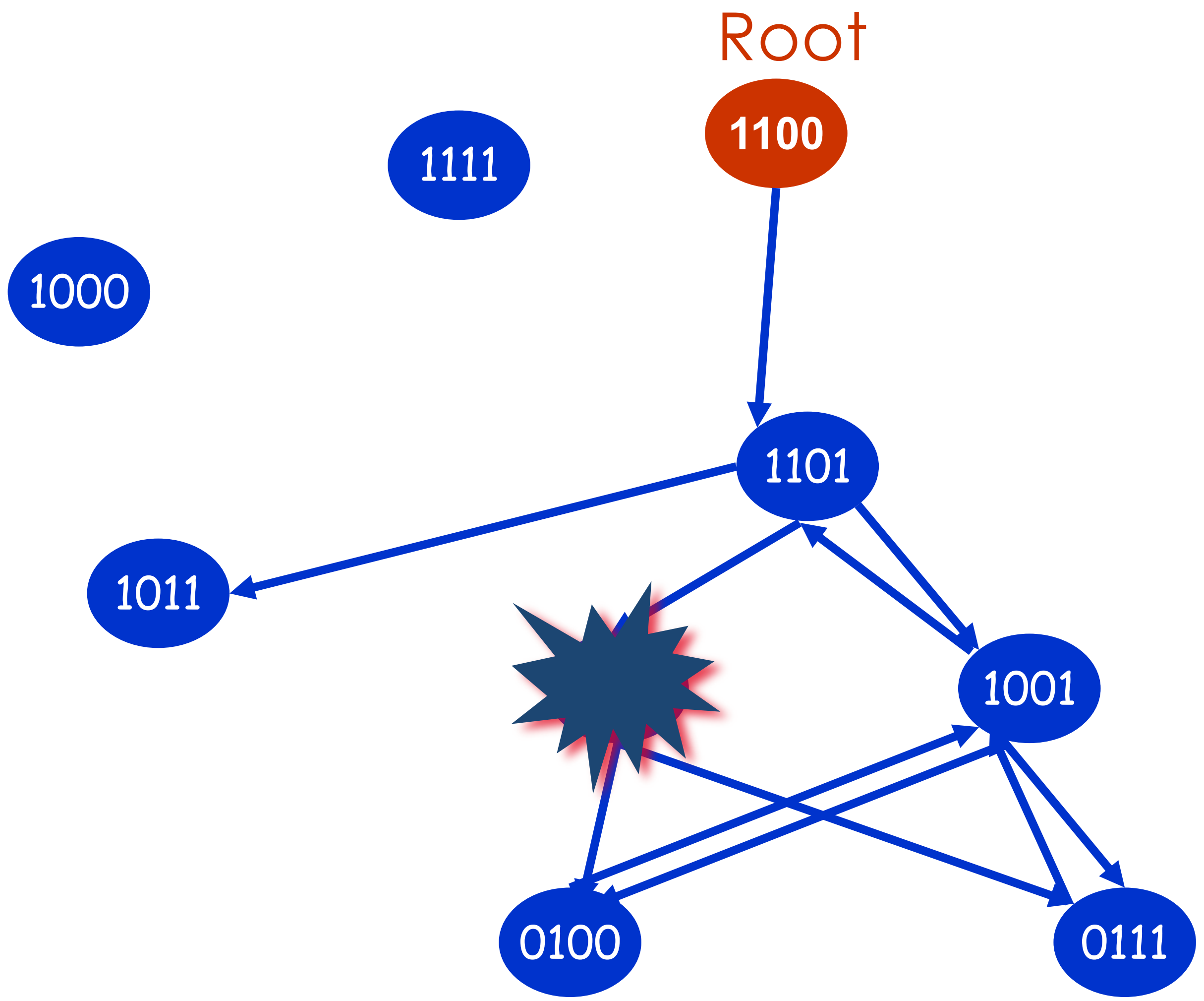
Node failure

- Parents periodically send heartbeat messages to their descendants in the tree
- When such messages are missed, nodes join the group again

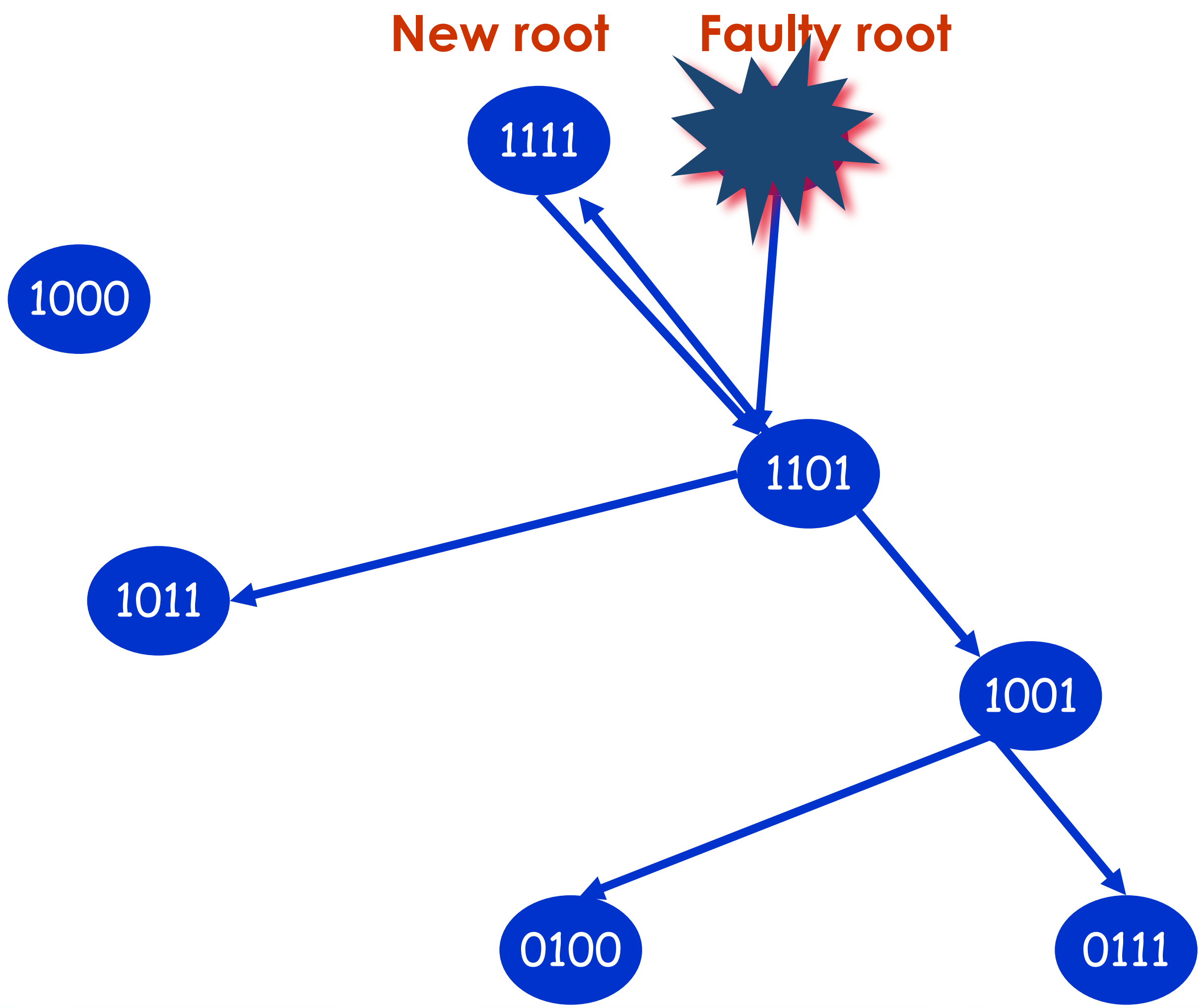
Local reconfiguration

Pastry routes around failures

Tree maintenance



Tree maintenance



Load balancing

- Specific algorithm to limit the load on each node
  - Size of forwarding tables
- Specific algorithm to remove the forwarders-only peers from the tree
  - smaller groups

Scribe performance

Discrete event simulator

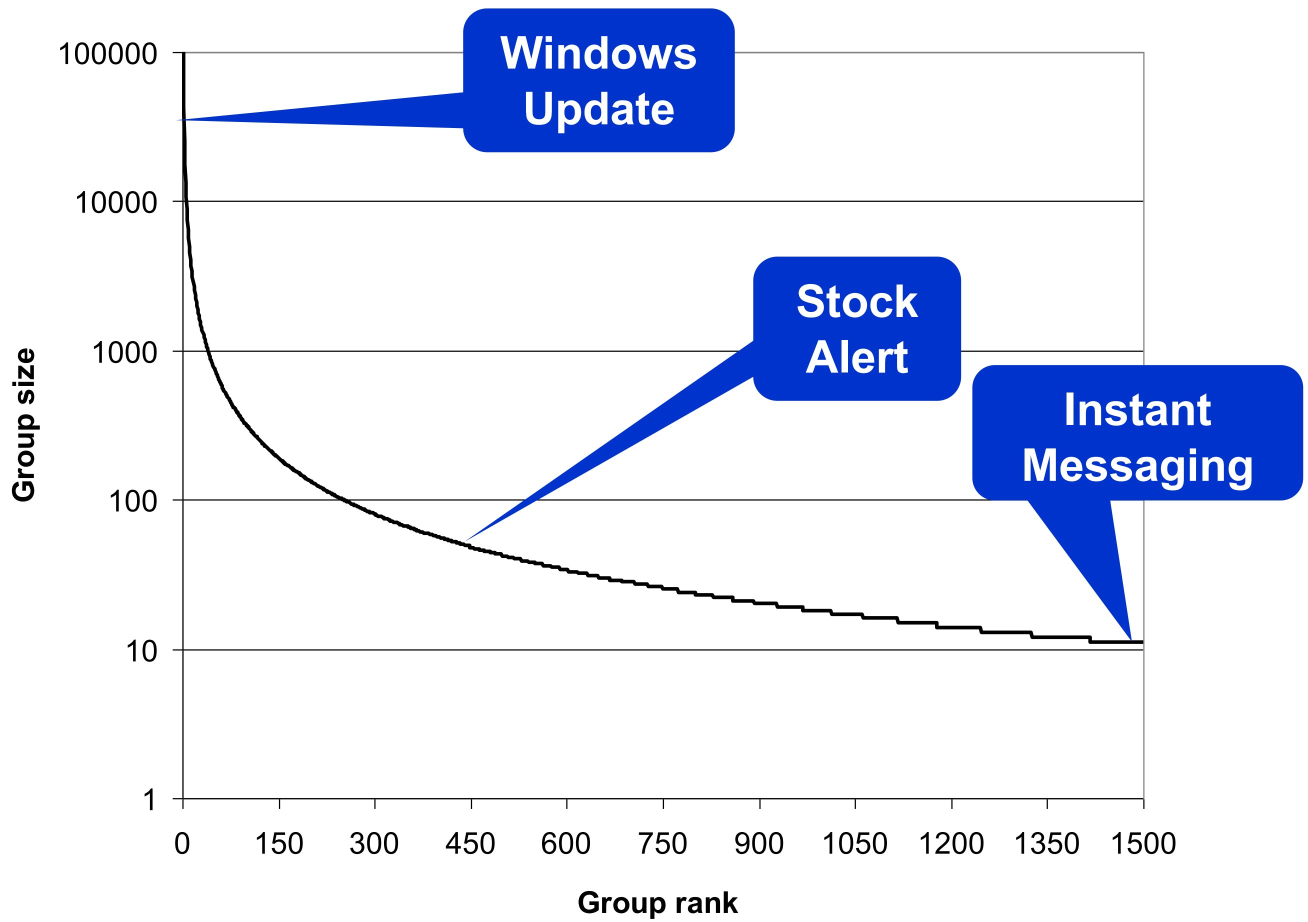
Evaluation metrics

- Relative delay penalty
  - RMD: max $delay_{app-mcast}$ / max $delay_{ip-mcast}$
  - RAD: avg $delay_{app-mcast}$ / avg $delay_{ip-mcast}$
- Stress on each network link
- Load on each node
  - Number of forwarding tables
  - Number of entries in the forwarding tables

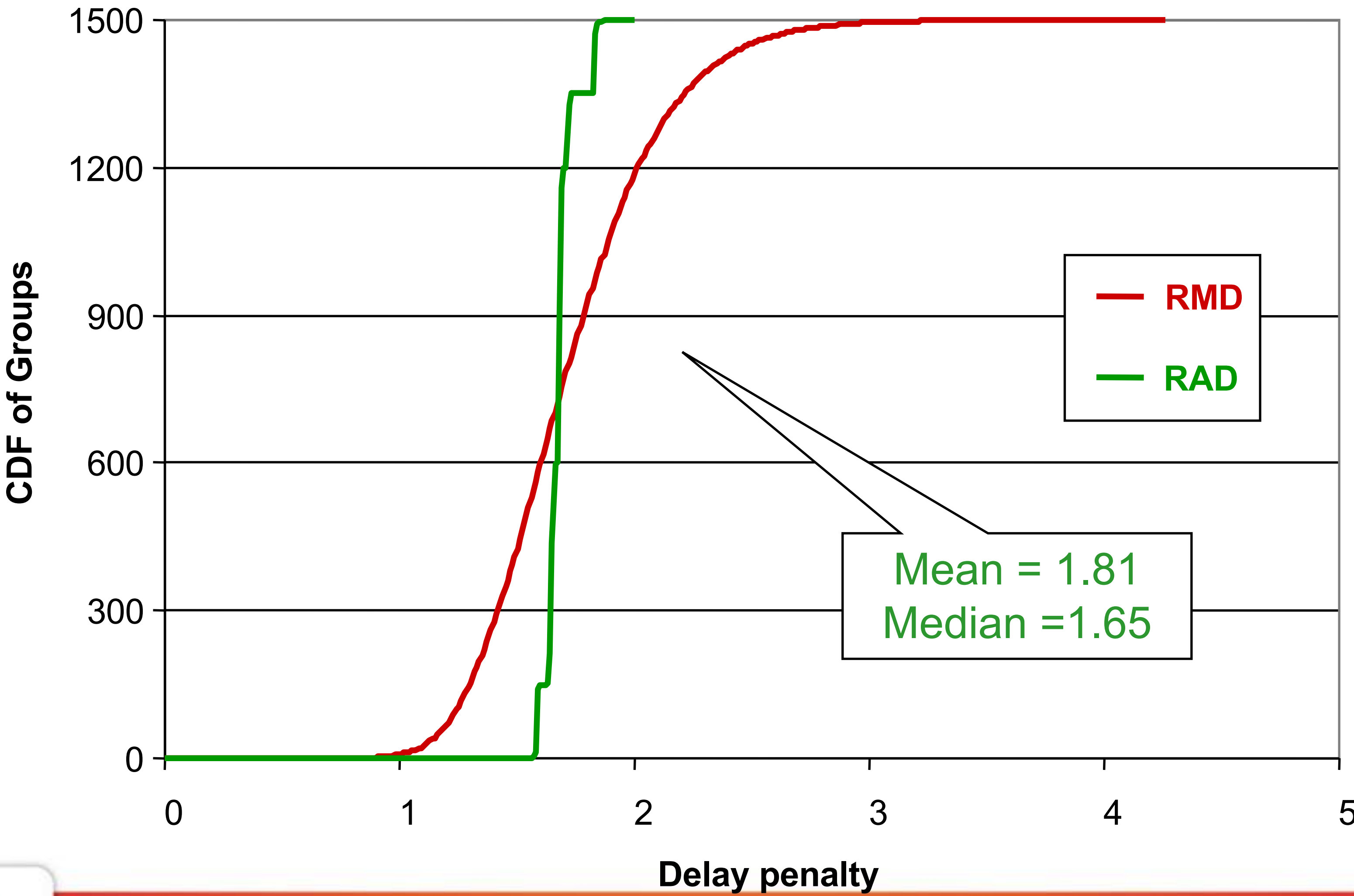
Experimental set-up

- Georgia Tech *Transit-stub* model (*5050 core routers*)
- 100 000 nodes chosen at random among 500 000
- Zipf distribution for 1500 groups
- Bandwidth not modeled

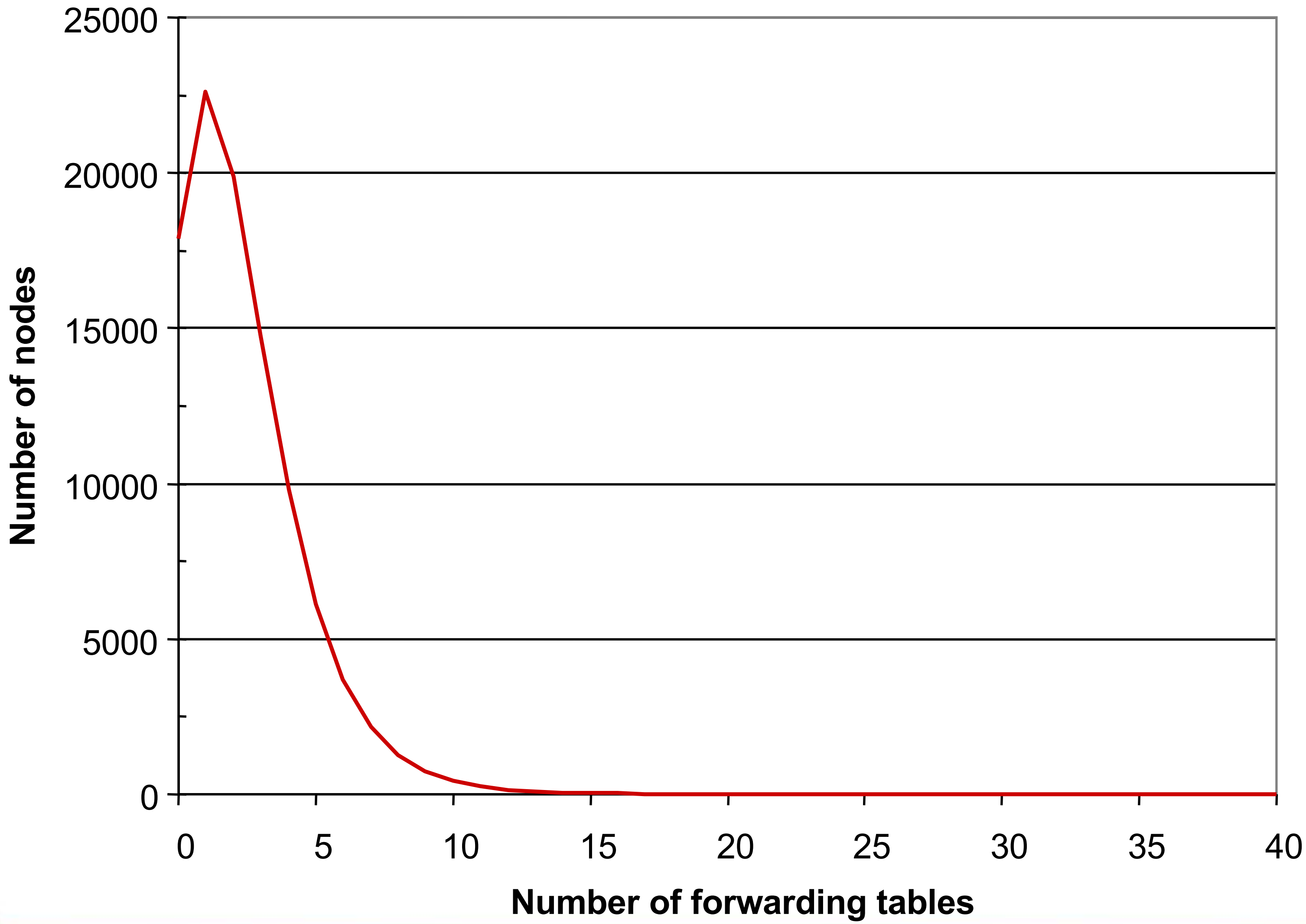
Group distribution



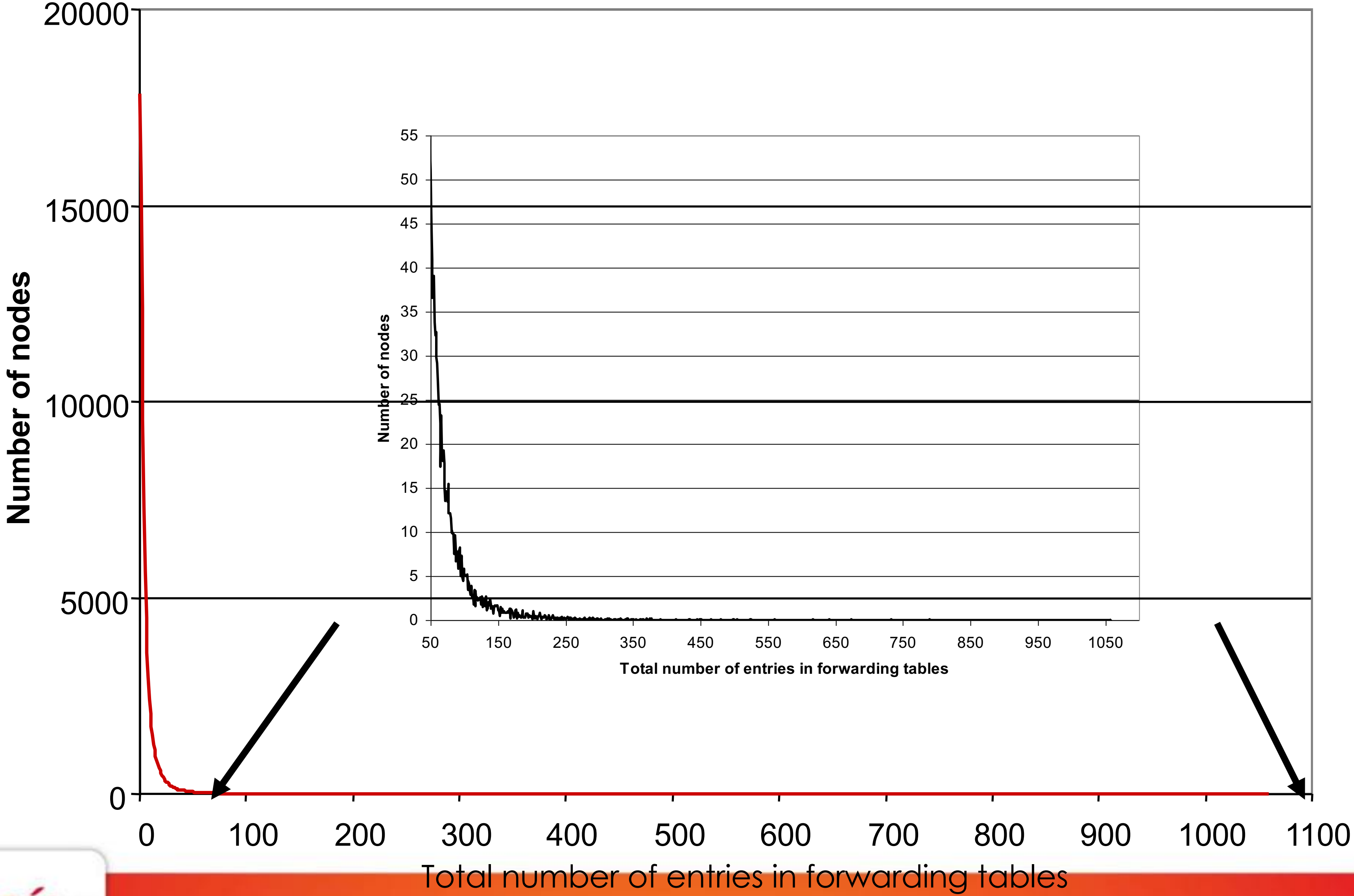
Delay/IP


1500
1200
900
600
300
0
CDF of Groups
0
1
2
3
4
5
Delay penalty
RMD
RAD
Mean = 1.81
Median =1.65

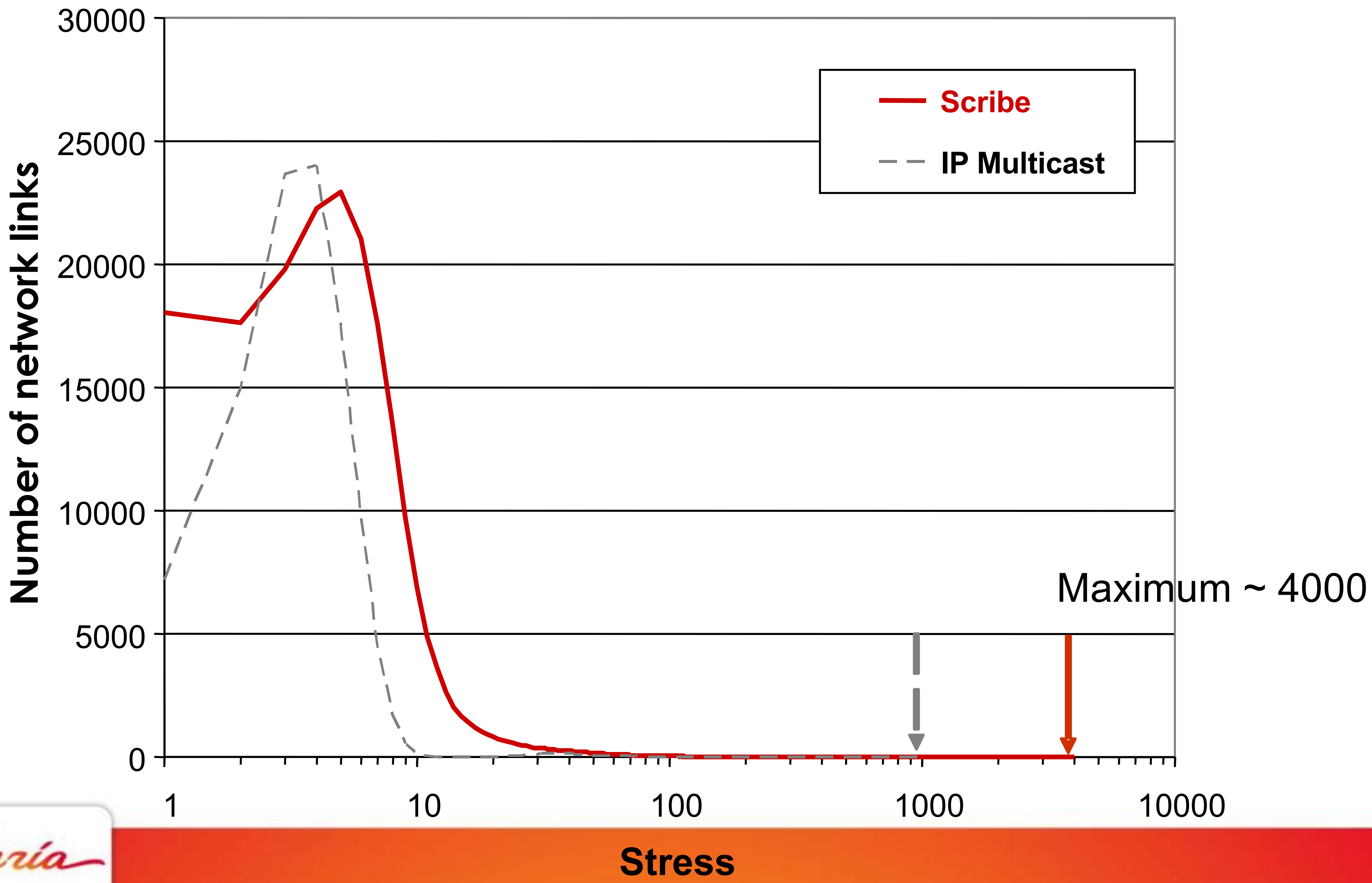
Load balancing



Load balancing



Network load



Summary

Generic P2P infrastructures

- Good support for large-scale distributed applications
- ALM Infrastructure

Scribe exhibits good performances/IP multicast

- Large size groups
- Large number of groups
- Good load-balancing properties

CAN Multicast

Flooding in a CAN network

- Either

  All CAN members are group members

- Or

  Mini CAN overlay creation/group

CAN multicast: group formation

Subset of CAN network members forms a mini-CAN

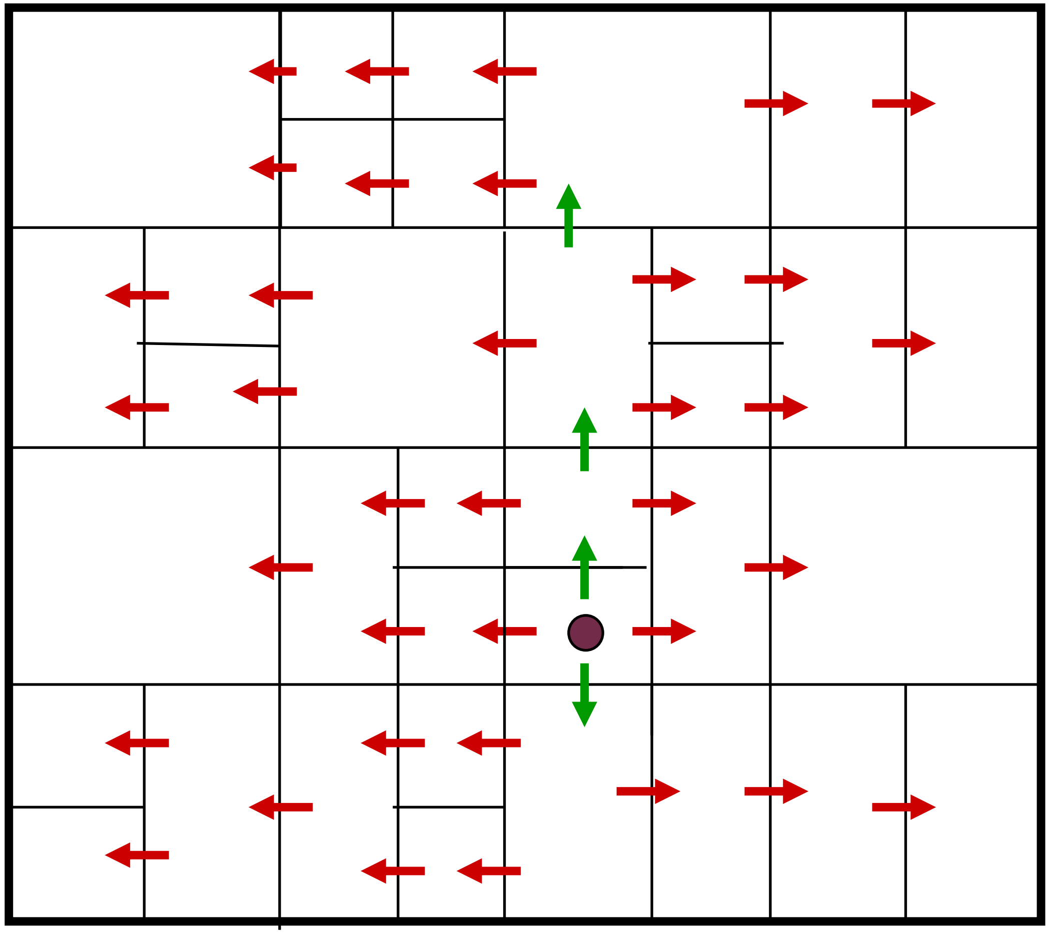
- Group identifier associated with a point *(x,y)* in the CAN space.
- (x,y) is the bootstrap node for the mini-CAN
- Group join = mini-CAN join

  Same as standard CAN join protocol

CAN multicast : message diffusion

- CAN network with *d* dimensions*: 1….d*
  - Each node maintains at least *2d neighbours*
- Diffusion
  - Source node sends the message to all its neighbours
  - A node receiving a message from dimension i
    - Forwards the message to its neighbours along the dimensions 1…(i-1)
    - Forwards the message to neighbours of dimension i in in the opposite direction (from the one it receives the message)
  - A node does not forward the message along a given dimension if the message has already traversed half of that dimension
  - A node does not forward an already received message

Example



Can multicast : Performance

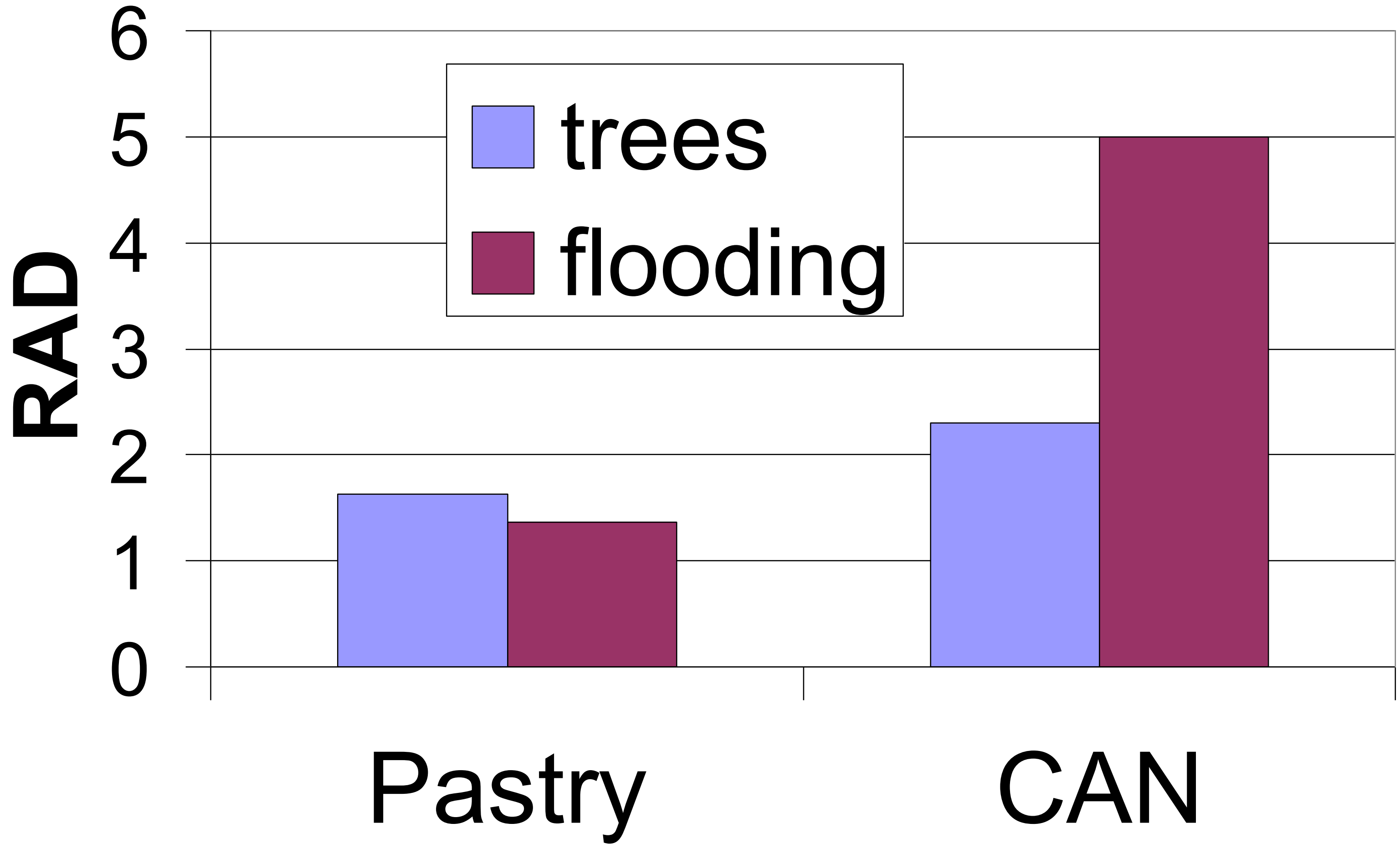
CAN: 6 dimensions, group of 8192 nodes, transit-stub topology

Relative delay penalty (RDP)

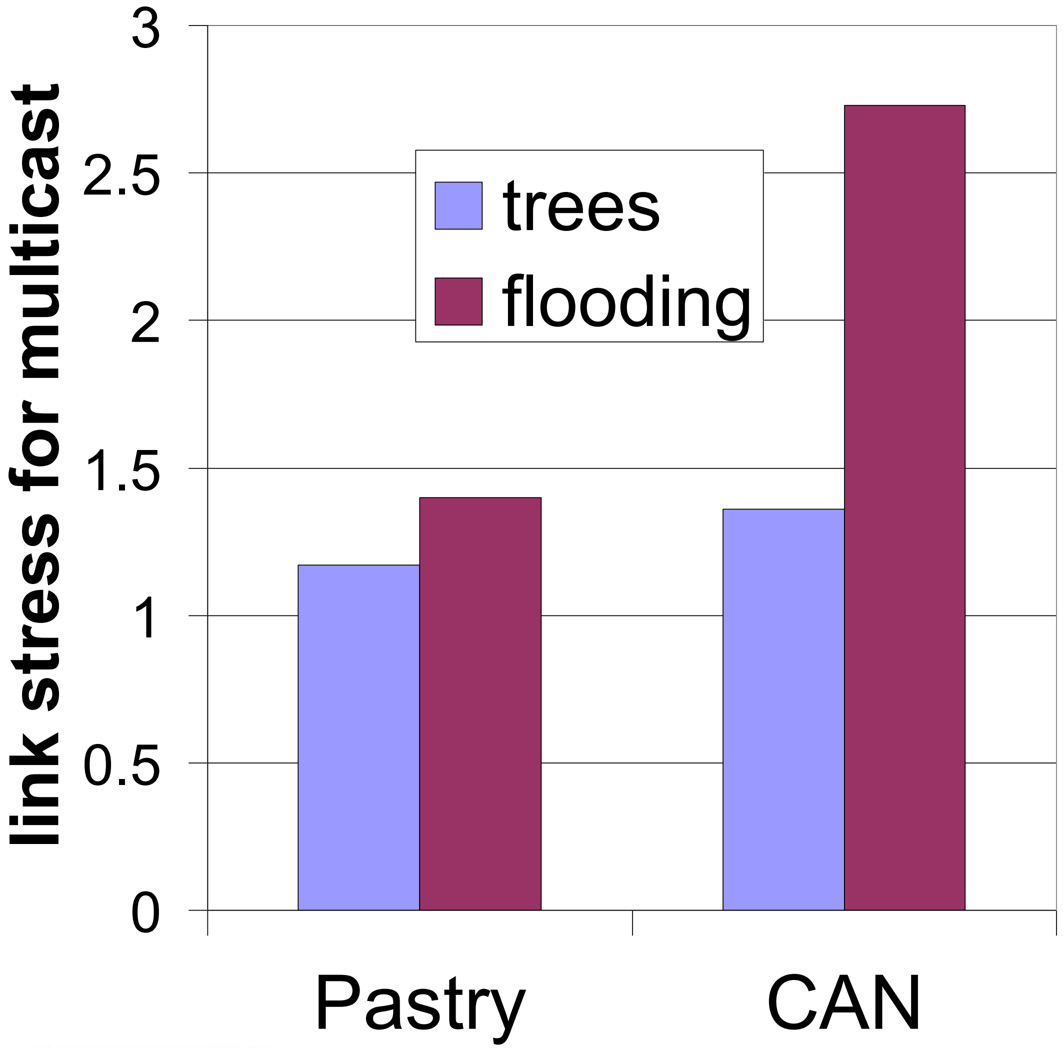
- 5-6 for the majority of group members

More details in the comparison

Comparison: delay penalty/IP



Comparison: average (physical) link stress



link stress for joining:

- identical for trees
- much larger for flooding
    - example: 281 on CAN

Trees versus flooding

Tree-based multicast is more efficient

- Lower delay and network stress during the multicast
- Huge difference in the network trafic during group creation
- Main drawback: some peers may be forwarders-only