# Implosion control in multipoint transport protocols

**Christos Papadopoulos**

*<christos@dworkin.wustl.edu>*

**Guru Parulkar**

*<guru@flora.wustl.edu>*

**IPP Progress Review**

**Jan 11, 1995**

This work is sponsored by:

- ARPA
- National Science Foundation
- Ascom Enterprise Networks
- Bell Northern Research
- Goldstar Information and Communication
- NTT
- NEC America
- Southwestern Bell
- Bay Networks (Synoptics)
- Tektronix

# Motivation

**Multicast connections in the future:**

- Hundreds or more participants (e.g. news distribution, stock-market information update)
- Senders may need feedback from receivers (e.g. error, flow control)
- Application latency requirements are critical

**What is implosion?**

The uncontrolled transmission of a large number of messages from endpoints in a multicast connection, that swamp other endpoints or the network

**Why implosion occurs (in spite of network support for multicast)?**

Networks achieve optimal delivery from source to destinations

However: Network optimal message delivery from multiple destinations to source (by combining messages) is hard:

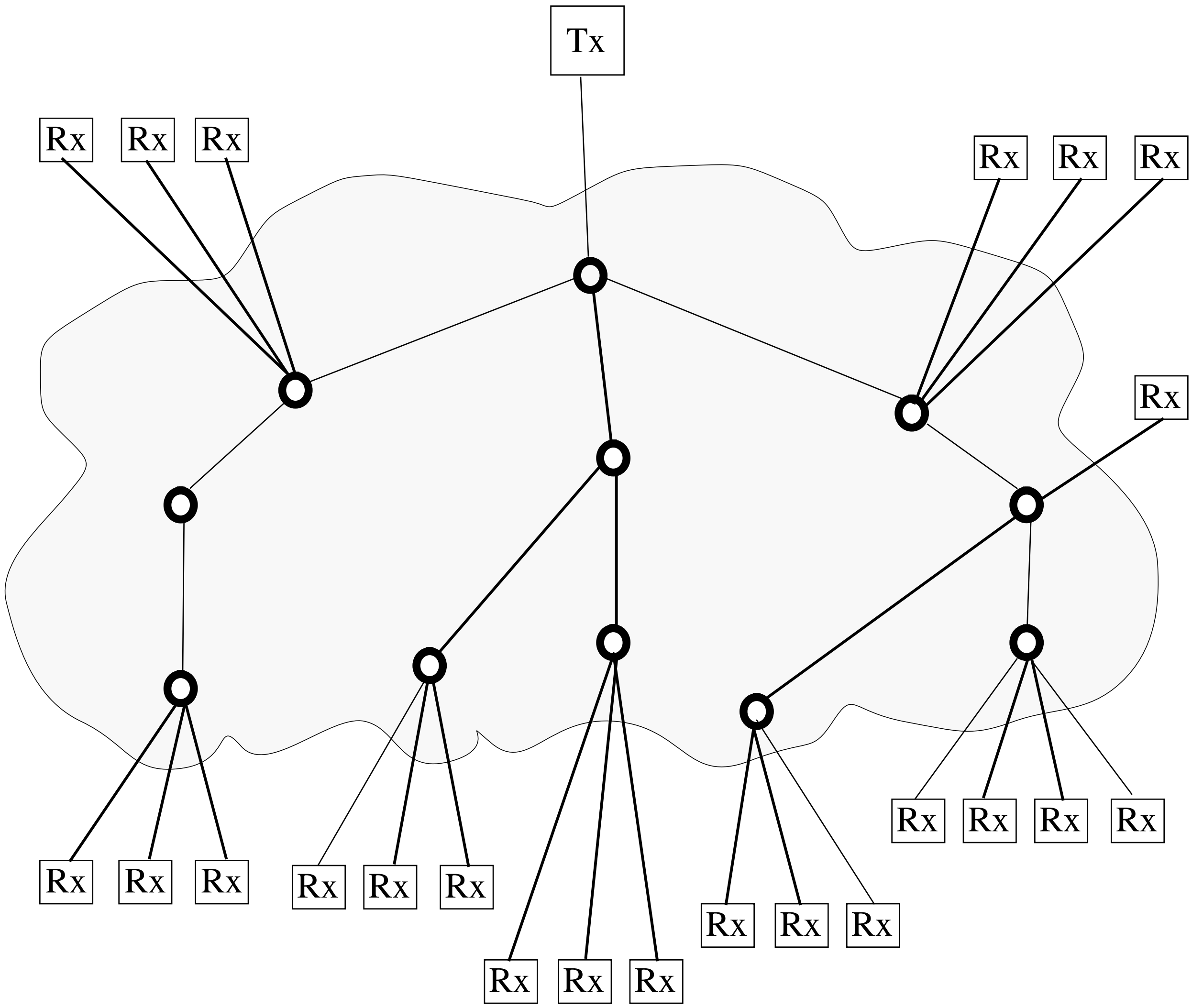
- synchronization
- software processing to combine messages

... and sometimes not desirable:

- security



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

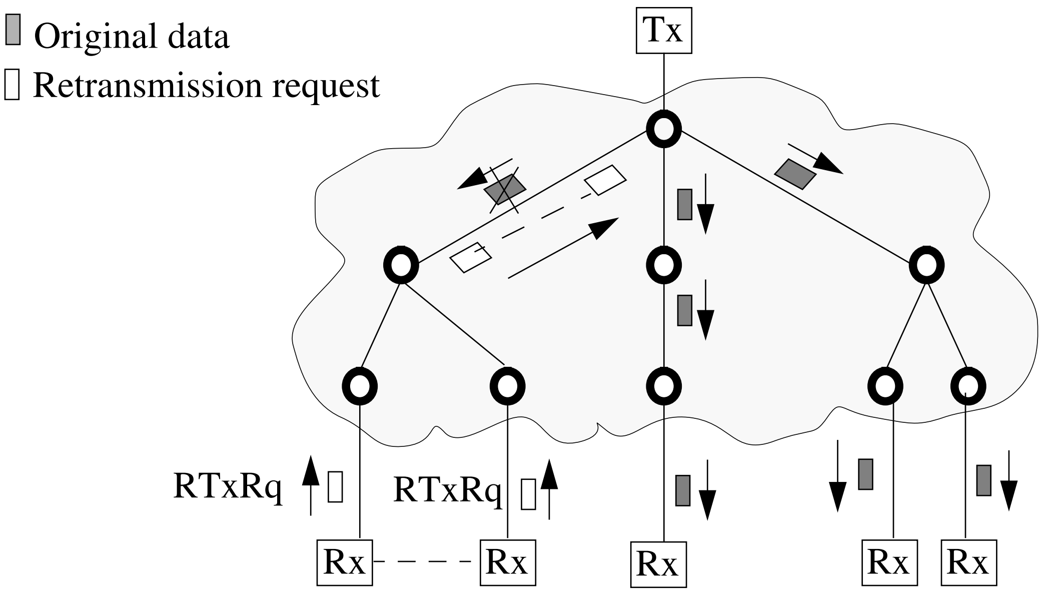





WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Implosion in error control

## Example:




A message on the left sub-tree is lost

All receivers in the left sub-tree send retransmission requests to the sender

## Possible consequences:

- The sender is swamped with messages and loses replies
- Severe latency is introduced at the sender due to processing of retransmission requests
- Network bandwidth requirements hard to predict

## Estimating implosion is very complex:

- loss probability, topology, location of loss

# Cost of implosion in error control

Assume a multicast tree where each switch has equal fan-out. Assume each internal node to be a switch and receivers to be the leaves

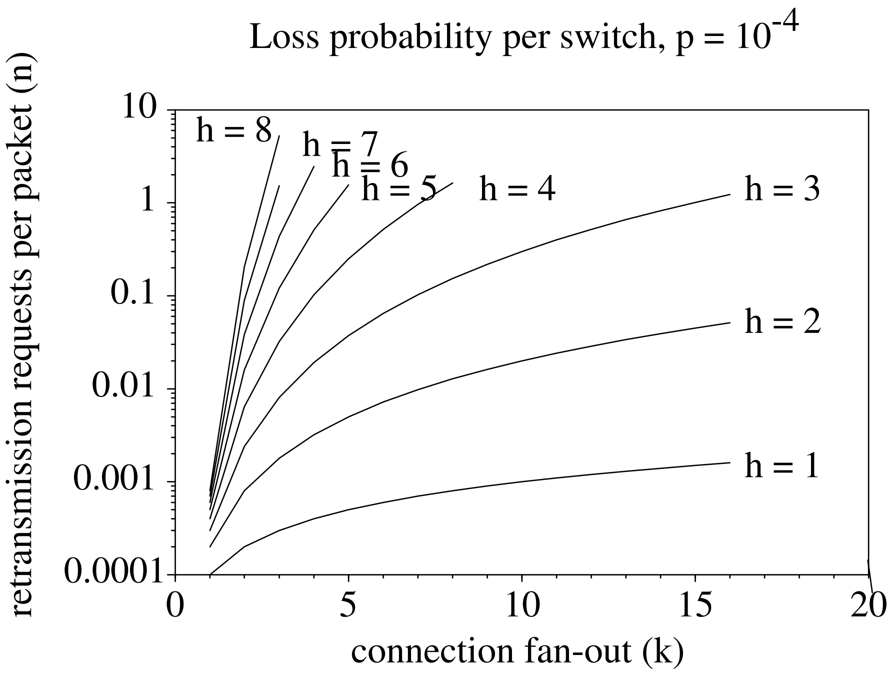
k: fan-out of each switch

h: height of the tree

p: probability a packet is lost at a switch

n: retransmission requests received for each data packet sent

$$n = k^h \left(1 - (1-p)^h\right)$$




Numerical example: for k=4, h=4 (256 receivers), p=$10^{-4}$, n=0.1

3 orders of magnitude increase in packets and interrupts compared to point-to-point case

# Optimal implosion control criterion

**Optimality criterion:**

optimal solution should resemble point-to-point recovery both in terms of messages and latency

**Messages:**

1. Recovery-related messages reach *only* endpoints experiencing loss and the retransmitting entity
2. Only one retransmission request is generated per subtree
3. Only one retransmission reaches each endpoint that experienced loss

**Latency:**

4. A retransmission request is sent by the endpoint that discovers the loss first
5. A retransmission request is sent immediately upon discovery of loss
6. A retransmission is sent by endpoint closest to the requesting endpoint
7. A retransmission is sent immediately upon reception of request

**Additional condition:**

Participation in error recovery is optional. Recovery should be completely transparent to endpoints that do not participate



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Designing an optimal solution is hard

- problem is distributed
- requires implicit mechanisms (to avoid more messages)
- processing messages distributed in time from multiple receivers can be ambiguous

However, optimal solution can be used as a *metric* for other simpler but sub-optimal solutions.



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Implementation of implosion control

**Attention must be paid to:**

1. Implosion-control processing: on par with current implementations
    - not more than a O(100) of instructions per packet
    - optimized critical path
    - overhead negligible during normal protocol operation

2. Distribution of processing: avoid bottlenecks, improve scalability

3. State information: resist temptation to increase state
    - avoid large and/or per-endpoint state
    - avoid frequent searching and update

4. Timers: avoid or minimize the use of timers
    - avoid per-packet and minimize or eliminate per-endpoint timers
    - timers are expensive: they involve interrupts + other OS overhead

5. Minimize number of connections



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Network participation

**Can we assume network participation?**

The idea is very attractive, but:

- Packets (headers) must be reassembled before examined. Very difficult for switches that switch cells (ATM)
- Additional processing and state at the switching entities. New resources must be shared and managed
- Entity examining packets must interpret protocol headers. At least part of the protocol must reside on that particular entity. Protocol updates difficult
- Switches must be programmable to perform application required implosion control services

For now we assume no network participation - implosion control is the job of the *Transport Protocol*

We assume internet protocol with multicast capabilities (e.g. IPV6)

However, future-generation switches may be programmable and thus able to participate in implosion control



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Implosion control design taxonomy

**Sender-controlled implosion**

- All responses from receivers go to sender only
- Sender tells the receivers what to do
- Centralized control
- Requires communication between sender and each receiver

**Receiver-controlled implosion**

- All responses from receivers go to all other receivers and sender
- Implosion control decisions are taken by receivers; sender may or may not participate
- Distributed control
- Requires broadcast communication between receivers and sender

**Hierarchical implosion control**

- Receivers are structured in a tree hierarchy, where children can send responses only to their parent
- Implosion control decisions are taken by receivers; sender may or may not participate
- Distributed control
- Requires communication between each child and its parent

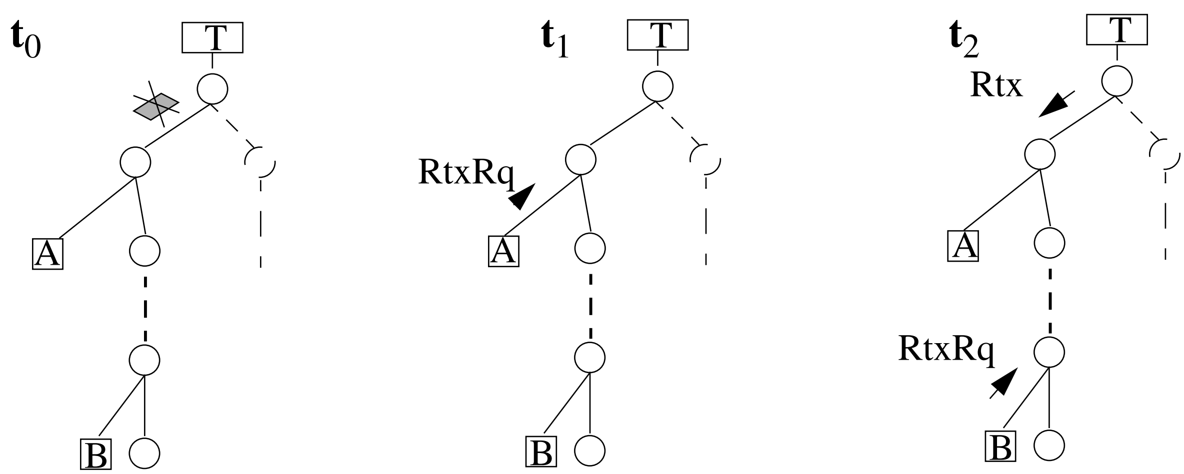
# Sender controlled implosion

Appropriate for:

- One-to-many connections over connection-oriented networks (ATM)
- Receivers must be isolated from each-other (performance, security)

Ignored by the Internet community (receiver-controlled assumed)

**Basic idea:**




Note: loss detection based on seq num gaps

**Key observations**

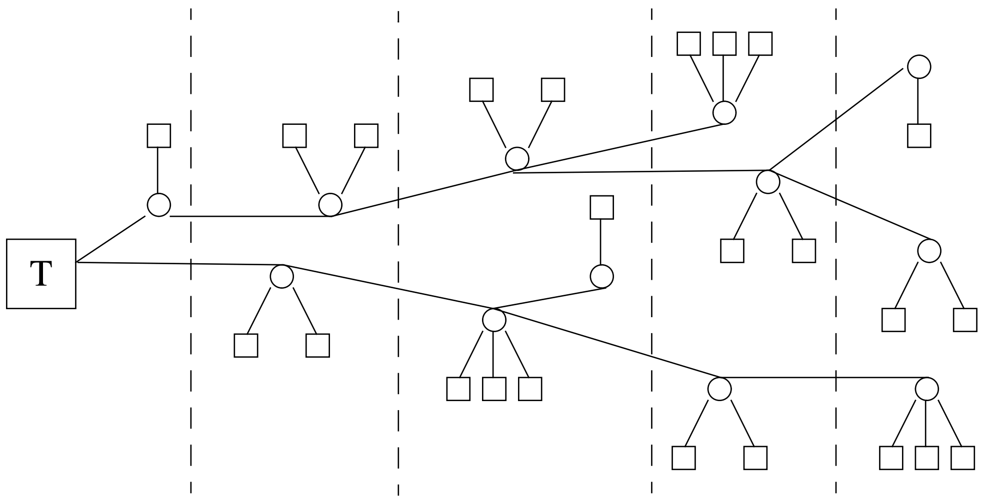
- Receivers closer to sender discover loss sooner
- Each receiver should back off for interval proportional to RTD from sender
- Receivers can find out about other receiver’s actions only through sender



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Example

One-to-many connection

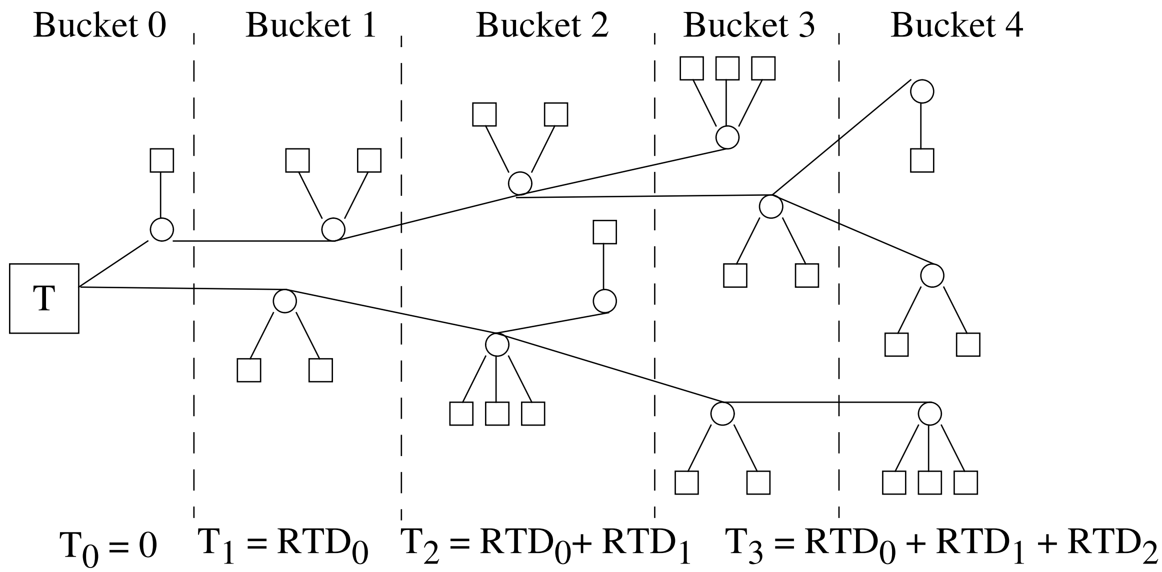



- Receivers divided into groups
- On detecting a loss, each group assumes that the previous group has detected the loss
- Assume the previous group has initiated a request and wait for a retransmission
- If none arrives, initiate a retransmission request



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# A heuristic algorithm




Each bucket b is associated with $RTD_b$ = MAX (RTD of any receiver in bucket) and back-off timer: $T_b = \sum_{k=0}^{b-1} RTD_k$

**Sender state:**

- Number of buckets
- back-off timer for each bucket
- histogram of receivers according to their RTD values

**Receiver state (relayed by sender):**

- Bucket ID
- Value of back-off timer
- Max back-off time (last bucket)

# Algorithm steps

**Receiver:**

- When receiver detects loss, it starts the back-off timer. If retransmission is received before timer expires, the request is cancelled
- If the timer expires and no retransmission is received, the receiver sends a request and sets its timer to max back-off
- If a retransmission fails to arrive again, another request is sent and the timer value is doubled. After N iterations, sender or network failure is assumed

**Sender:**

- On reception of first retransmission request, send retransmission
- Ignore other requests from the same bucket

Implosion: O (bucket size)

Latency: O (number of buckets)



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Controlling the bucket size

Sender controls the bucket size to avoid implosion from a bucket grown too large

**To split a bucket, the sender:**

1. Creates two new buckets by dividing number of receivers in bucket in two
2. Determines values of back-off timers
3. Multicasts the id of the bucket to be split, the RTD threshold, the new bucket ids and the new back-off timer values
4. Receivers belonging to the split bucket update their state and join the new bucket
5. Receivers in buckets further away increase their timer values by the back-off time of the new bucket

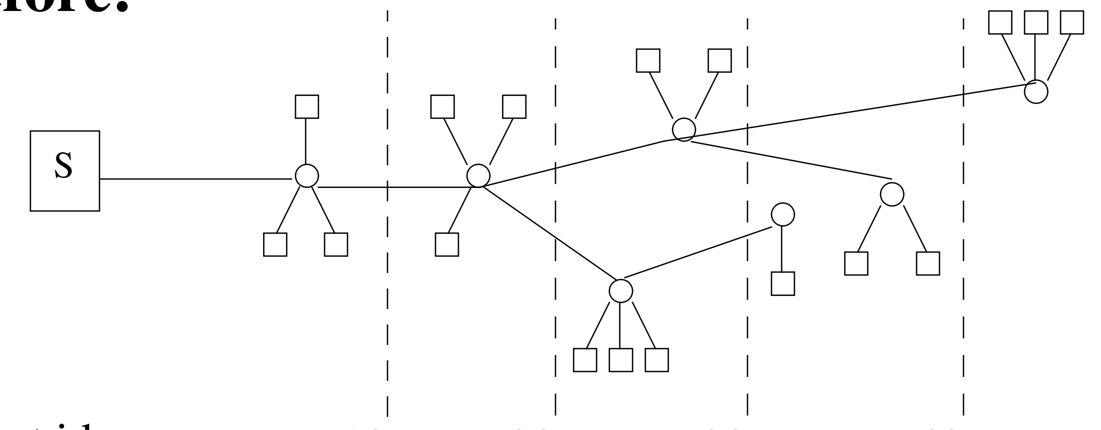
Sender must unite adjacent buckets when membership drops, to reduce latency

**To unite two adjacent buckets, the sender:**

1. Determines the new value of back-off timer (min of two buckets)
2. Multicasts message containing ids and timer values of united buckets
3. Receivers in united buckets update their timer values (if necessary)
4. Receivers in buckets further away reduce their timer values by the max of united buckets

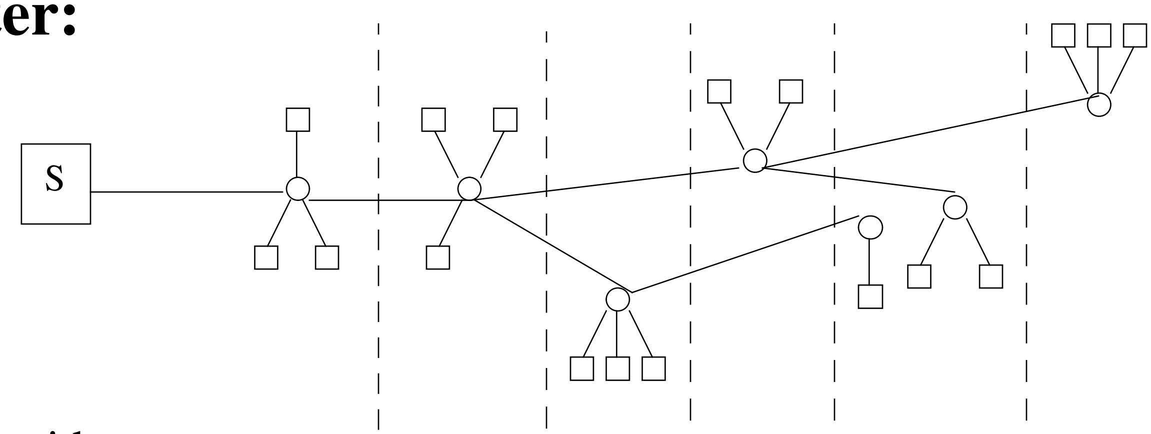
# Bucket splitting

## Before:




| Bucket id | 10 | 20 | 30 | 40 |
|---|---|---|---|---|
| RTD | 1 | 2 | 3 | 4 |
| Back-off delay | 0 | 1 | 3 | 6 |

## After:




| Bucket id | 10 | 20 | 30 | 35 | 40 |
|---|---|---|---|---|---|
| RTD | 1 | 2 | 2.5 | 3 | 4 |
| Back-off delay | 0 | 1 | 3 | 5.5 | 8.5 |



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Features/Trade-offs

**Features:**

1. Suitable for ATM one-to-many connections
2. Sender has direct control on implosion
3. Minimal implosion-related processing at receivers
   - no sending/receiving messages from other receivers
   - no buffering for retransmissions
4. Processing, state and buffering for retransmissions located at the sender
   - limits scalability, uneven load distribution
5. Latency and back-off delays high because messages must go through sender

Meets optimally criteria 3, 4 and 7

**Trade-offs:**

- Implosion is reduced as the number of buckets increases
- Latency is increased as the number of buckets increases
- Sender must select the desirable implosion/latency compromise based on # of receivers, loss rate, RTD



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Evaluation plan

1. Create a simulation environment to evaluate performance

**Parameters:**

- number of endpoints
- number of buckets
- topology and RTD
- loss rate

**Measure:**

- latency (compare with optimal and with other solutions)
- number of messages received by sender
- network bandwidth requirements

2. Specification of protocol (packet formats, state machines)

3. Create an implementation in SunOS or NetBSD over local ATM testbed to evaluate complexity
   - instruction count
   - use of timers
   - processing requirements and scalability
   - compare with simulation results for example topologies

# Receiver-controlled implosion

Favored by the Internet community

Basic mechanism first appeared in the XTP design: Although implemented, no known systematic evaluation exists

Enhanced design presented by Van Jacobson during his Sigcomm '94 tutorial in the context of IP Multicast - no known implementation yet

Requirements:

- inter-receiver communication (many-to-many connections)
- receivers able to buffer and retransmit lost data

Features:

- latency potentially better than sender-controlled
- even processing distribution
- receivers more complex that sender-controlled
- chance of implosion not eliminated
- no receiver isolation
- adding/deleting receivers transparent to the sender

# Performance and trade-offs

## Performance:

- best case: optimal implosion control
- worst case: no implosion control

## Trade-off

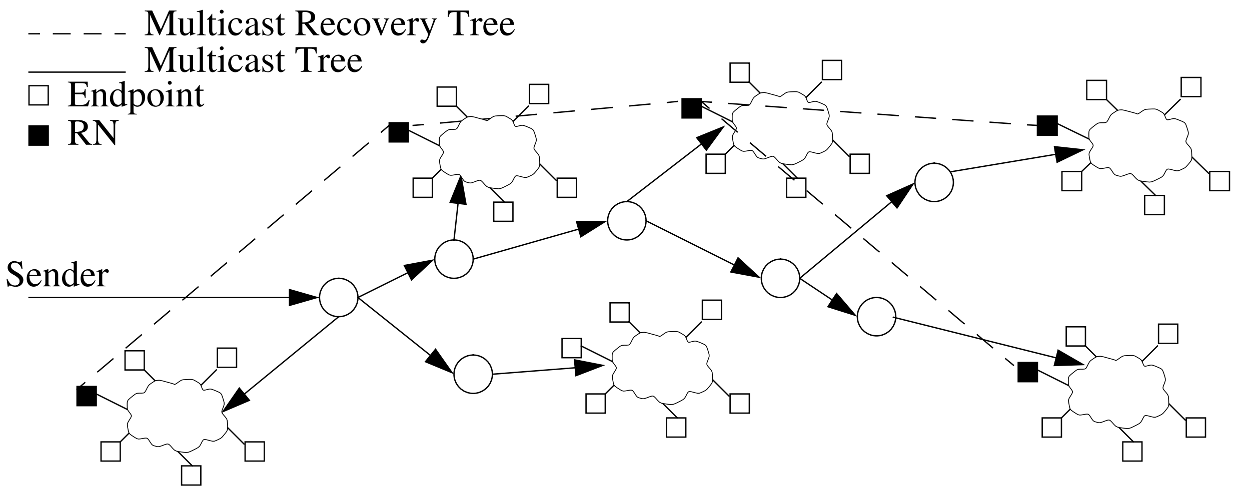
- Latency v.s. implosion
- evaluation plan similar to sender-controlled



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Hierarchical implosion control

**Basic ideas:**




1. Sender establishes a unidirectional multicast tree (MT) with all receivers

Receivers form a Multicast Recovery Tree (MRT) as follows:

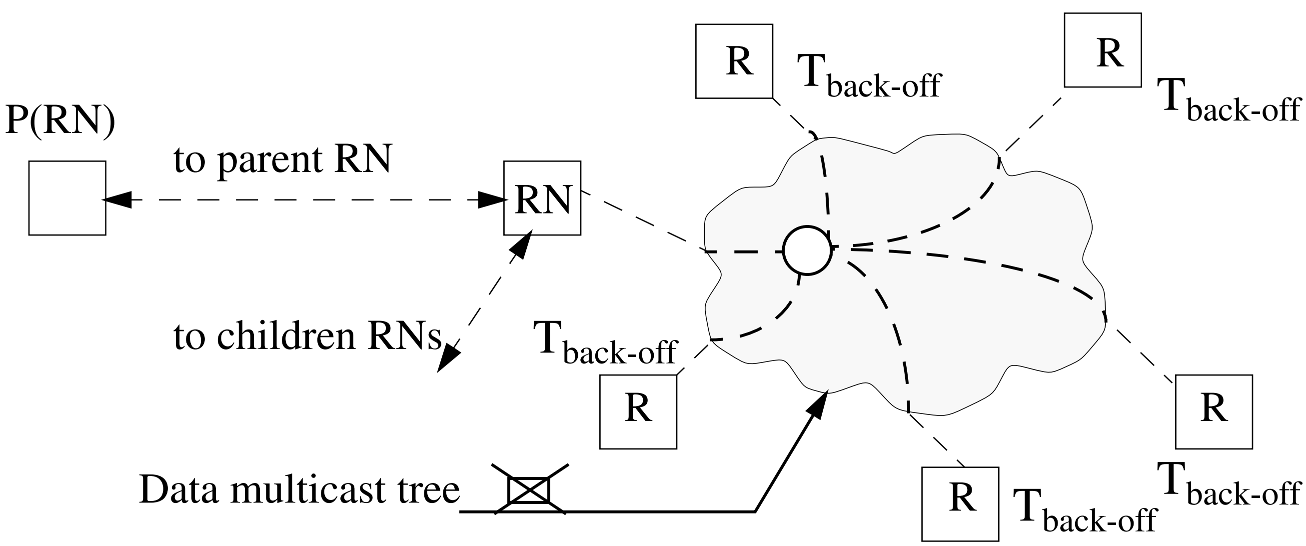
2. Receivers group themselves in local groups
3. Each local group selects a relay node (RN)
4. A bidirectional connection between receivers and RN is created (each group is free to choose the connection type according to requirements and the local underlying network)
5. RNs connect to parent RNs via a point-to-point connection

**Key observation: MRTs *need not* follow same topology as MTs**

# Heuristic algorithm

## A. All R, RN detect loss

1. All endpoints (R, RN) detect a loss
2. RN immediately (a) sends retransmission request to parent RN, and (b) forwards a copy to local group
3. Rs start local back-off timer $T_{back\text{-}off}$ >= propagation delay between R and RN
4. If notification from RN arrives before $T_{back\text{-}off}$ expires, Rs keep quiet
5. If P(RN) has data, it forwards it to RN.
6. If P(RN) does not have data, it has already sent its own request. P(RN) marks child RN as "expecting" data
7. When retransmission arrives to any RN, it is forwarded to local group and any marked children RNs






WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

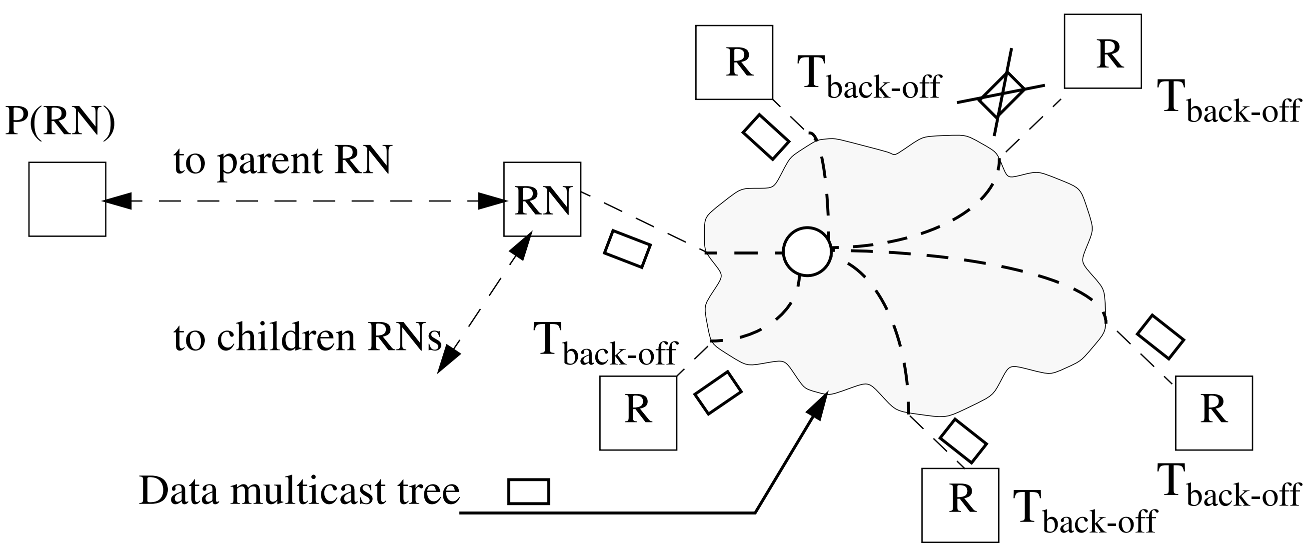
# Heuristic algorithm

**B. Some R detects loss, but RN has data**

1. R detects a loss and sets $T_{back-off}$ as before
2. When $T_{back-off}$ expires and a notification from RN fails to arrive, a request is send to RN
3. RN transmits missing data

**C. RN detects loss, but some R has data**

1. Notification from RN triggers a retransmission from R of missing data to RN (if Rs can retransmit)
2. Or: P(RN) retransmits to RN (if Rs do not retransmit)

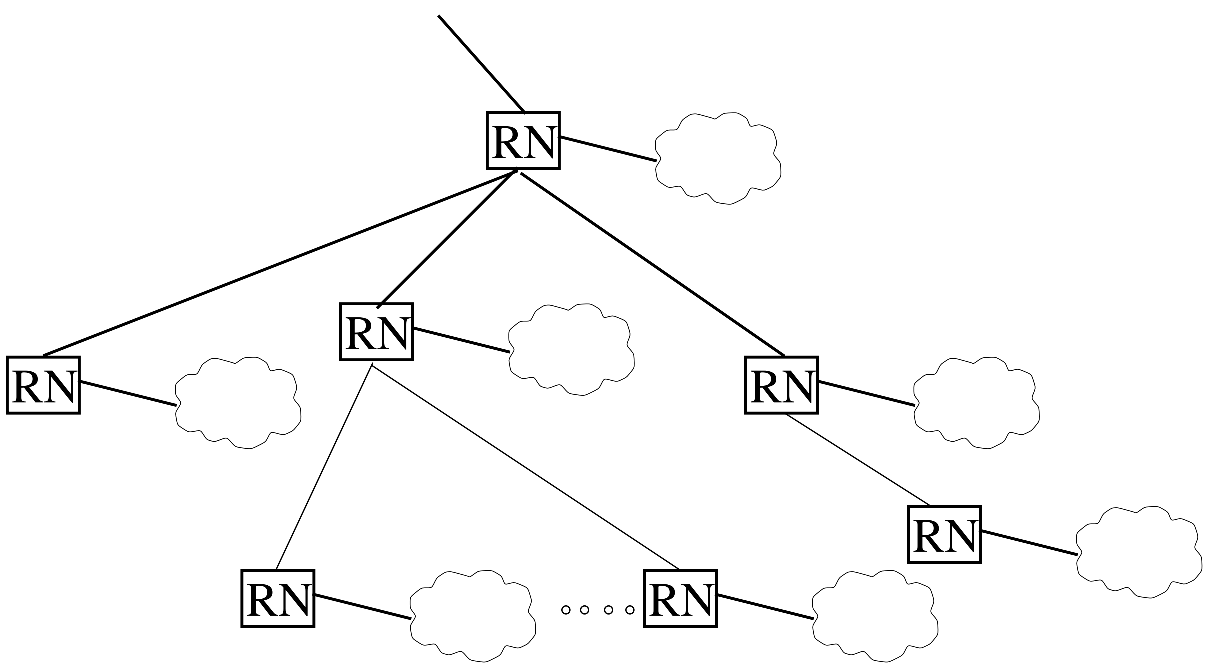





WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Features/trade-offs

- local groups use sender-controlled, receiver-controlled implosion or some sender/receiver combination
- near-optimal performance: meets *all* optimality criteria in most cases regardless of topology
- low latency: low or no back-off delays
- distributed processing: good load balancing, excellent scalability
- receivers more complex than with sender-controlled implosion: implosion processing and buffering retransmission
- general complexity higher than sender/receiver controlled implosion: hierarchical structure requires additional network connections and maintenance






WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Multicast Recovery Tree Protocols

1. Local group formation/reformation protocol
   - group membership (addition/deletion) and scope
   - notifies receivers of the RN identity
   - reformation protocol reforms group after failure.

2. Local group communication protocol
   - communication between receivers and RN
   - reliable delivery of data to all group members
   - detects failed or malfunctioning receivers (including RN)

3. MRT formation/reformation protocol
   - topology and membership operations
   - reformation protocol reforms MRT after failure

4. RN tree communication protocol
   - communication between RNs
   - point-to-point reliable protocol.



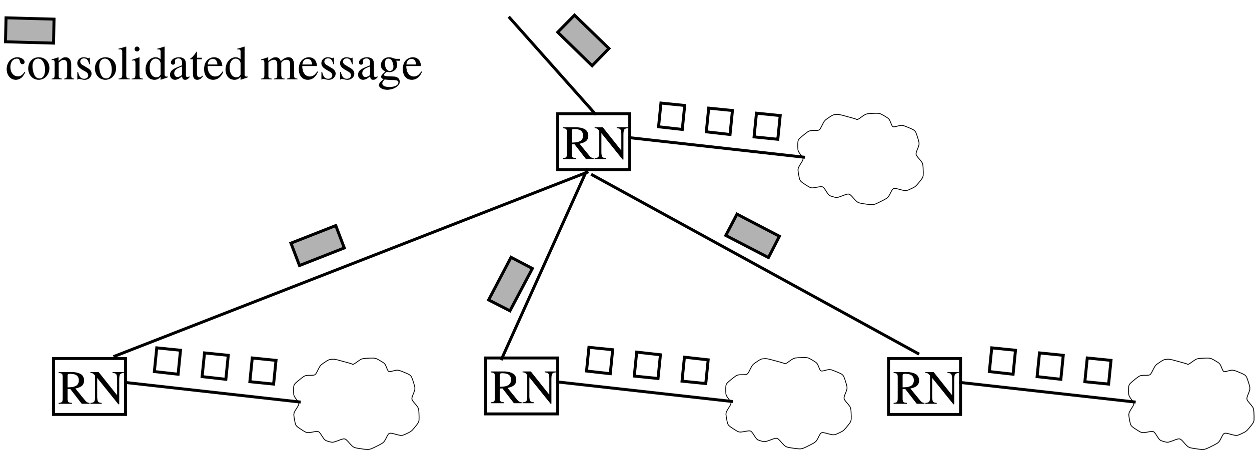
WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Consolidation

## Why do we need consolidation?

Consolidation is critical for reliable multicast protocols with large number of endpoints

- reliable protocols need to send positive acks
- positive acks are not redundant - must be delivered to sender
- positive acks pose a bigger implosion threat






WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Consolidation messages and operations

| Collection method | Explanation |
|---|---|
| Periodic | wait for time T, consolidate all collected messages and forward up the tree |
| Message per receiver | collect one message from every receiver |
| Sender initiated | sender requests message collection explicitly, specifying method |

| Operation | Example |
|---|---|
| MIN $(x_1, x_2, ... x_n)$ | highest packet sequence number received by all receivers |
| MAX $(x_1, x_2, ... x_n)$ | maximum window size advertized by any receiver |
| OR $(b_1, b_2, ... b_n)$ | combine error bitmaps from all receivers to determine packets lost by all receivers |
| AND $(b_1, b_2, ... b_n)$ | combine error bitmaps to determine which packets have been received by all receivers |



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Plan of action

1. Examine algorithms for group formation, maintenance, RN election and recovery from failure
   - From distributed computing

2. Examine location and interaction of group-related protocols
   - Coexist with, or be part of transport protocol
   - Part of signalling protocol (ATM)
   - Network daemons

3. Clearly define consolidation requirements
   - operations
   - state
   - messages

4. Create a simulation environment to evaluate performance

**Parameters:**

- number of endpoints
- hierarchy levels
- topology and RTD
- loss rate



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

**Measure:**

- latency (compare with optimal and sender/receiver controlled implosion)
- number of messages received by sender
- network bandwidth requirements

5. Create an implementation in SunOS or NetBSD over local ATM testbed to evaluate complexity
   - interaction of group-related and data transmission protocols
   - instruction count
   - use of timers
   - processing requirements and scalability

6. Understand impact of using:
   - sender-controlled implosion
   - receiver-controlled implosion
   - combination of sender and receiver controlled implosion

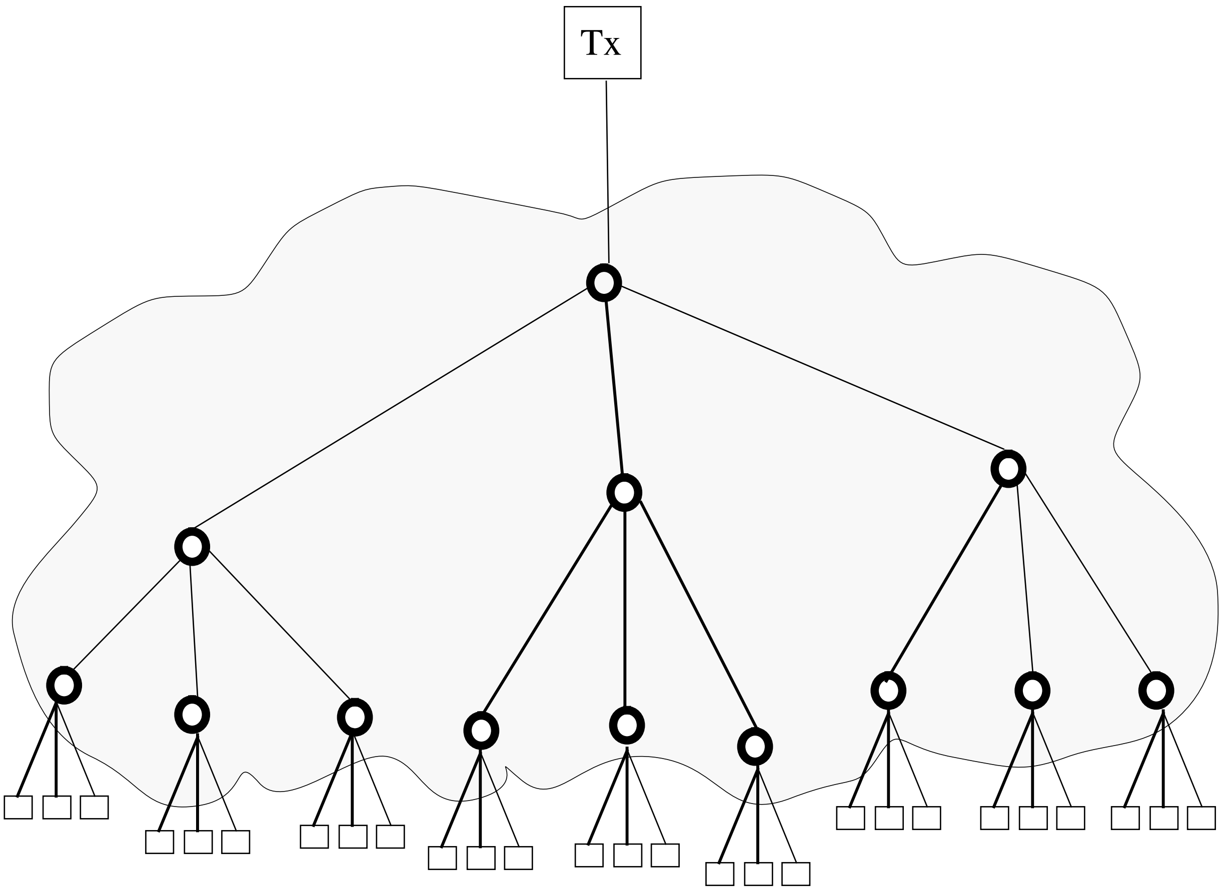
   in the local groups on the overall hierarchical scheme



WASHINGTON UNIVERSITY
ONE BROOKINGS DRIVE
ST. LOUIS MISSOURI

# Conclusions

1. Implosion is a serious problem in multipoint communication with large number of endpoints, which has not been adequately addressed yet
2. Implementing an optimal solution is hard, but several options exist for sub-optimal solutions which are worth exploring
   - sender-controlled implosion
   - receiver-controlled implosion
   - hierarchical implosion control
3. The trade-offs of these solutions vary widely, and must be systematically evaluated via both simulation and implementation
4. Implementation and evaluation of receiver-controlled implosion may be taking place in the Internet community, but not in ATM environment




k = 3
h = 3
27 endpoints