An Error Control Scheme for Large-Scale Multicast Applications

Christos Papadopoulos

http://dworkin.wustl.edu/~christos

Guru Parulkar

http://www.arl.wustl.edu/~guru

George Varghese

http:/dworkin.wustl.edu/~varghese

Supported by: NTT, Tektronix, NIH, NEC, SWBT TRI, Samsung, Sun

Outline

1. Problem
2. Solution
3. Related Work
4. Evaluation + Implementation
5. Conclusions

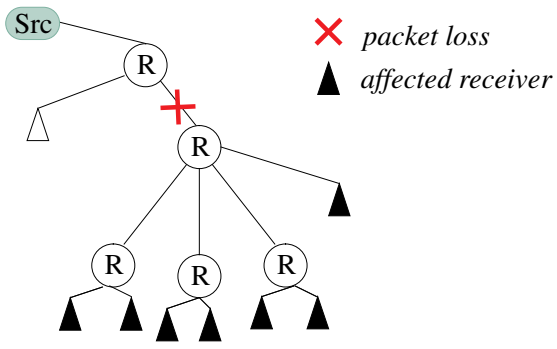
Work dates back to summer 1996.

Submitted but not accepted for Sigcomm '97.

Currently in review for Infocom '98.

1. Problem: Loss in Multicast

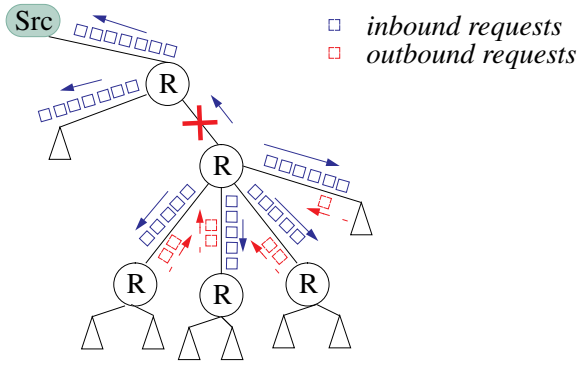
In a multicast environment, a single packet loss may affect anywhere from *one* to *all* receivers.



This leads to two main problems:

- **Duplicates**
- **Exposure**

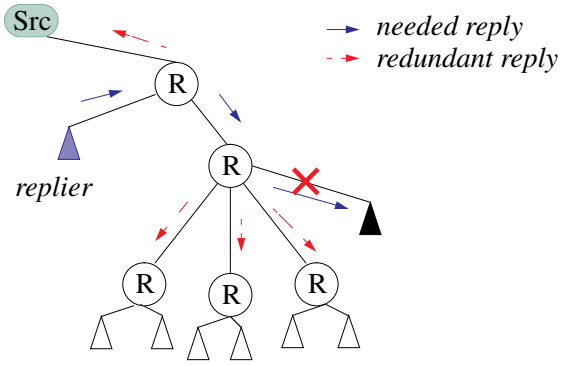
Problem 1: Duplicates (Implosion)



Loss of a packet triggers a flurry of **duplicate requests**, as shown above.

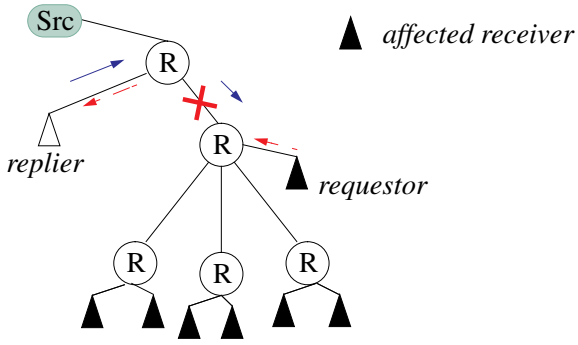
Requests may trigger a flurry of **duplicate replies**, depending on who is retransmitting.

Problem 2: Exposure



Requests and replies reach everyone in the group, regardless of whether they need them or not.

Ideal Recovery Scenario



STEP 1: A single request is sent from closest requestor below the loss to the closest replier above the loss (call them the *requestor/replier pair*).

STEP 2: Replier sends a single reply, which is multicast on the affected link.

No duplicates, no exposure, minimal latency.

The Need for Router Assistance

We need router assistance to:

- Find the requestor/replier pair closest to the loss.
- Locate the root of the loss subtree.
- Multicast reply to loss subtree.

SRM requires no router assistance, but pays a penalty in latency, duplicates and isolation.

NOTE: we are **not** asking routers to do error detection, buffering, or retransmissions.

Routers are a precious resource: we need to use them wisely.

2. Solution: Extend Router Forwarding

Three Core Ideas

Enhance routers to:

- Select a replier interface
- Guide requests to repliers, and in doing so, discover the root of the loss subtree
- Help repliers multicast replies to loss subtree

Impact on Routers

CLAIM: Extensions are simple and easy to integrate into existing router forwarding code:

-> No changes in routing

-> No packet storing

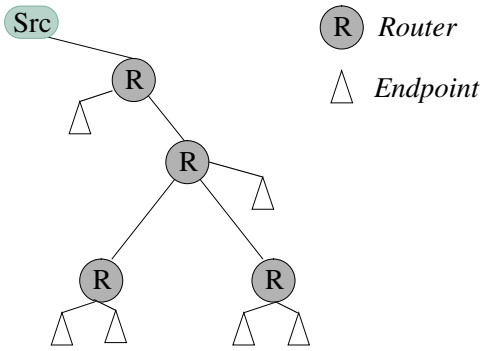
-> No per-packet state

-> No touching of user data

-> Small added state (4 bytes)

-> Slightly modified forwarding path **for requests**

Existing Forwarding State (DVMRP)



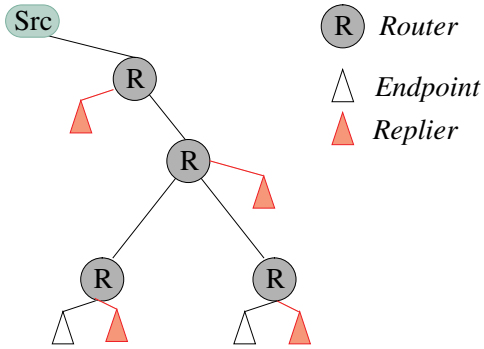
Routers create *per-source* multicast trees and maintain *per-source* state, including:

- Upstream link
- List of downstream links

Will show how to deal with routing protocols with no per-source state (e.g., PIM-SM) later.

Add a Replier to each Router

Each router picks a *Replier Link* from its list of downstream links:



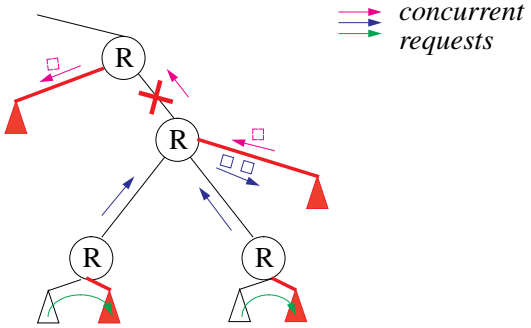
Router State (per-source tree) now becomes:

- Upstream link
- List of downstream links
- Replier link identifier

Forwarding Requests

When loss occurs *ALL* endpoints multicast requests containing a special IP option.

Routers forward requests to repliers, except replier's request, which is forwarded upstream.

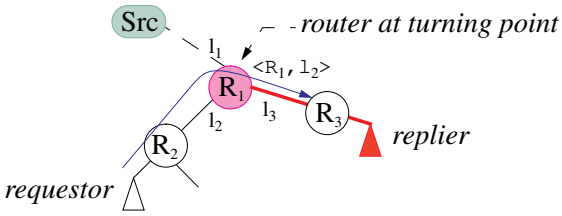


Result: No Implosion - only one request per subtree is sent upstream

Request Forwarding at the Routers

A *Turning Point* defines a loss subtree.

A turning point occurs when a router forwards a request received from downstream to its replier.



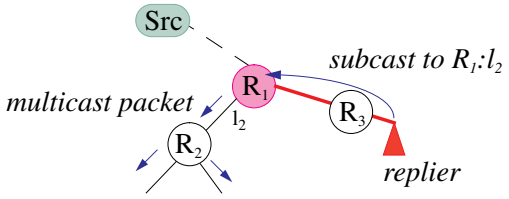
At the turning point the router adds to request:

- Router address (R_1)
- Link id of the link the request arrived on (l_2)

Request carries R_1:l_2 to the replier

Sending Replies with a SUBCAST*

A subcast allows an endpoint to unicast an encapsulated multicast packet to a router, where it is decapsulated and multicast on a specified link.

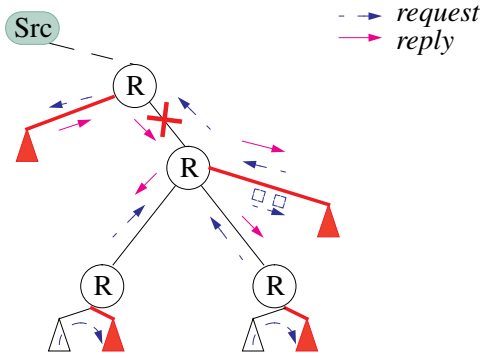


After subcast, packet appears as normal multicast to downstream routers and endpoints.

Result: subcast limits replies to the loss subtree.

* SUBtree multiCAST: term coined by Adam Costello. Mechanism also present in RMTP.

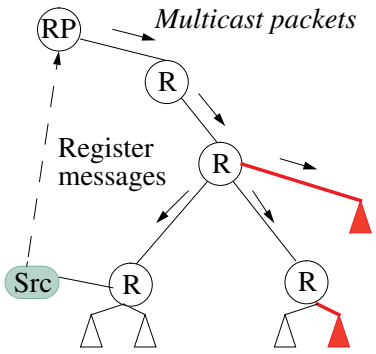
Recap:



After a loss is detected:

- All endpoints send requests
- Routers forward requests to repliers
- A single replier responds with a subcast

Dealing with PIM

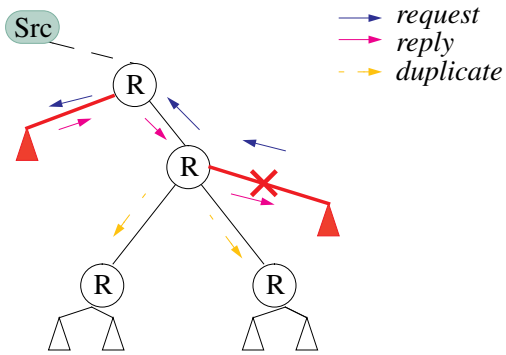


With register messages, a router with a (*, G) entry elects repliers by treating RP as the source. If a request reaches RP, it is unicast to the real source.

If a SPT was established, repliers are elected as before.

Duplicate Replies

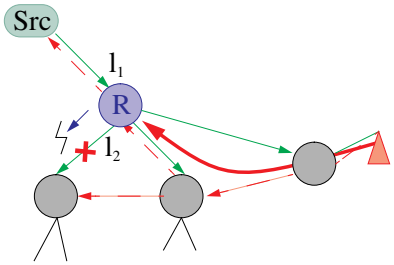
Duplicate replies will be generated if loss occurs on the replier branch:



Problem minimized by selecting the replier experiencing the least loss.

Per-source Multicast Trees*

With per-source multicast trees, a reply may not follow the same tree as packets from original source

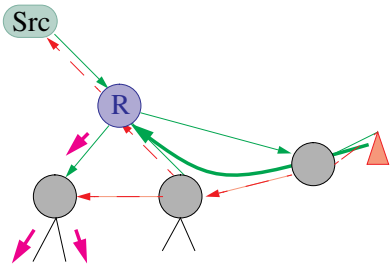


A subcast from replier to R is rejected since link l_2 is not part of replier's tree.

*Problem also mentioned in RMTP paper

Solution: Subcast with Source Spoofing

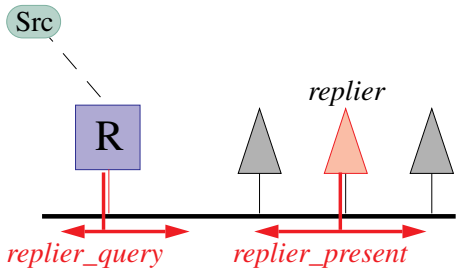
Repliers are allowed to "spoof" the original source address when using subcast.



Subcast contains multicast packet with address of *Src*. Subcast is accepted on tree.

For security, routers may insert option in multicast packet, identifying the real source.

Details: Repliers on LANs



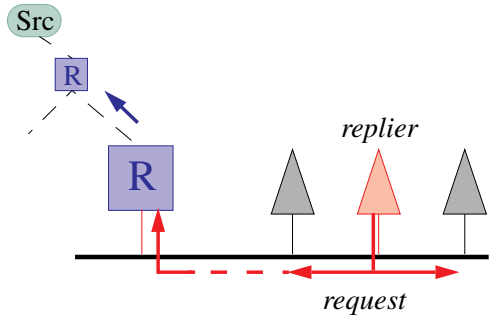
Router periodically multicasts "replier_query" messages.

Replier responds with "replier_present" message.

If replier leaves group or crashes, remaining endpoints elect new replier.

Router is not involved in election.

Sending Requests on a LAN

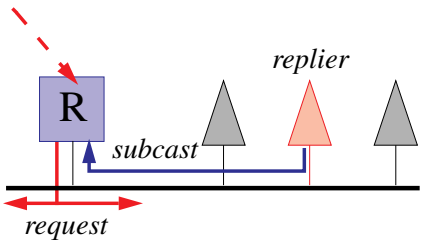


On loss, replier immediately multicasts request.

Replier's request cancels requests from other endpoints.

Router forwards replier's request upstream.

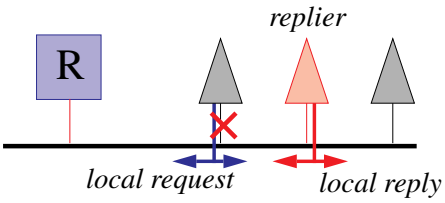
Request Arriving on a LAN



Router multicasts arriving request on the LAN.

Replier responds to request with a subcast; other endpoints ignore it.

Local Recovery on a LAN



Endpoint other than replier experiences loss.

After its timer expires, endpoint multicasts a local request.

Replier responds with local reply.

3. Contrast with Existing Work

PROS:

- Very few, or no duplicate requests and replies in most cases
- No back-off timers (SRM, TMTP)
- No need to estimate neighborhood size, round trip delays, TTL values (SRM)
- No fixed topology (RMTP, LBRRM))
- No need for parent discovery (TMTP)
- Highly dynamic (adjusts instantly to topology changes)

CONS:

- Requires router modification

("no problem", say Active Networks people)

4. Evaluation

The scheme was evaluated via:

1. Simple numerical results
2. Simulation
3. Implementation

Numerical Results: Binary Trees

$$\textbf{Exposure} = \frac{\#\ \textit{receivers that received a reply}}{\#\ \textit{receivers that should have received the reply}}$$

Tree Height	No Local Recovery	Our Scheme
1	2	1
2	4	1.5
3	8	2
4	16	2.5
...	...	...
h	2^h	O(h)

Simulation Results - Binary Trees

$$\textbf{Nuisance} = \frac{\#\ \textit{unwanted replies received by } R}{\textit{total number of losses}}$$

Latency = *loss recovery time - loss detection time*

Tree Height	Nuisance	Normalized latency (latency / unicast rtt)
3 (8 receivers)	0.11	0.92
4 (16 receivers)	0.10	0.93
5 (32 receivers)	0.08	0.95
6 (64 receivers)	0.06	0.96

Implementation

Highlights:

- Implemented in NetBSD Unix
- About 250 lines of new C-code for request and subcast processing
- Two new IP options (request, subcast)
- Request/subcast processing < normal multicast packet processing
- Replier info carried in IGMP messages
- Control info exchange between kernel and application via `sendmsg`/`recvmsg` syscalls (as proposed for IPv6)

Implementation Status

- Request forwarding code: completed
- Subcast code: completed
- Enhancing IGMP and `mrouted`: almost there

Most C-code already available (see paper). Rest will be available soon.

Additional ongoing work:

- Moving simulation into ns
- Creating user level protocol to use these services
- Designing scheme to deploy over existing routers in MBone

5. Conclusions

- Highly scalable scheme: no implosion, very little duplicates, rapid local recovery
- Impervious to group or topology changes
- Very simple forwarding changes to routers
- Forwarding overhead lower than normal multicast forwarding
- Easy integration with existing infrastructure
- Incremental deployment?