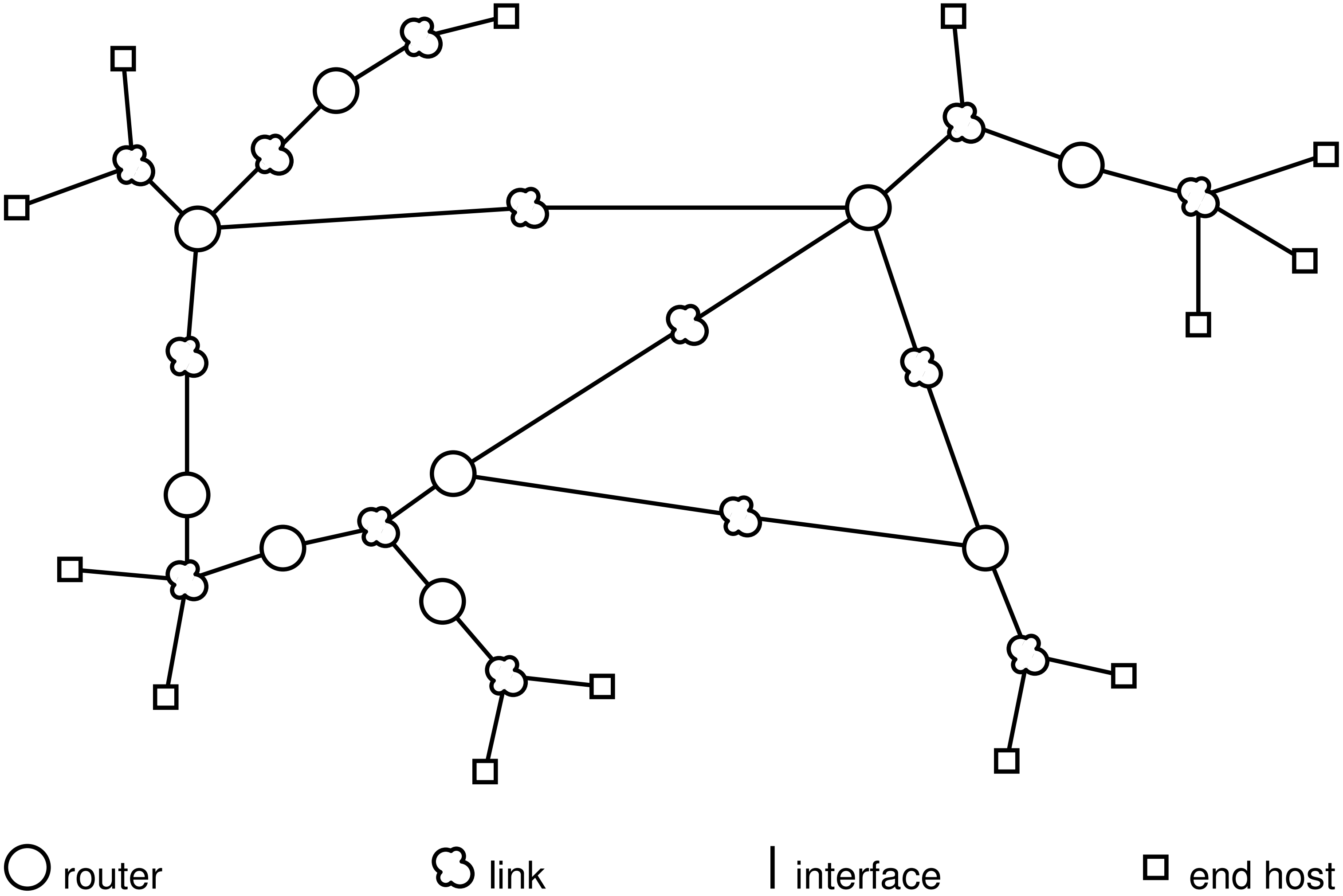
Joint Design of Multicast Loss Recovery and Forwarding Service Extensions

Qualifying Examination

Adam M. Costello

1999-May-06

Internet architecture

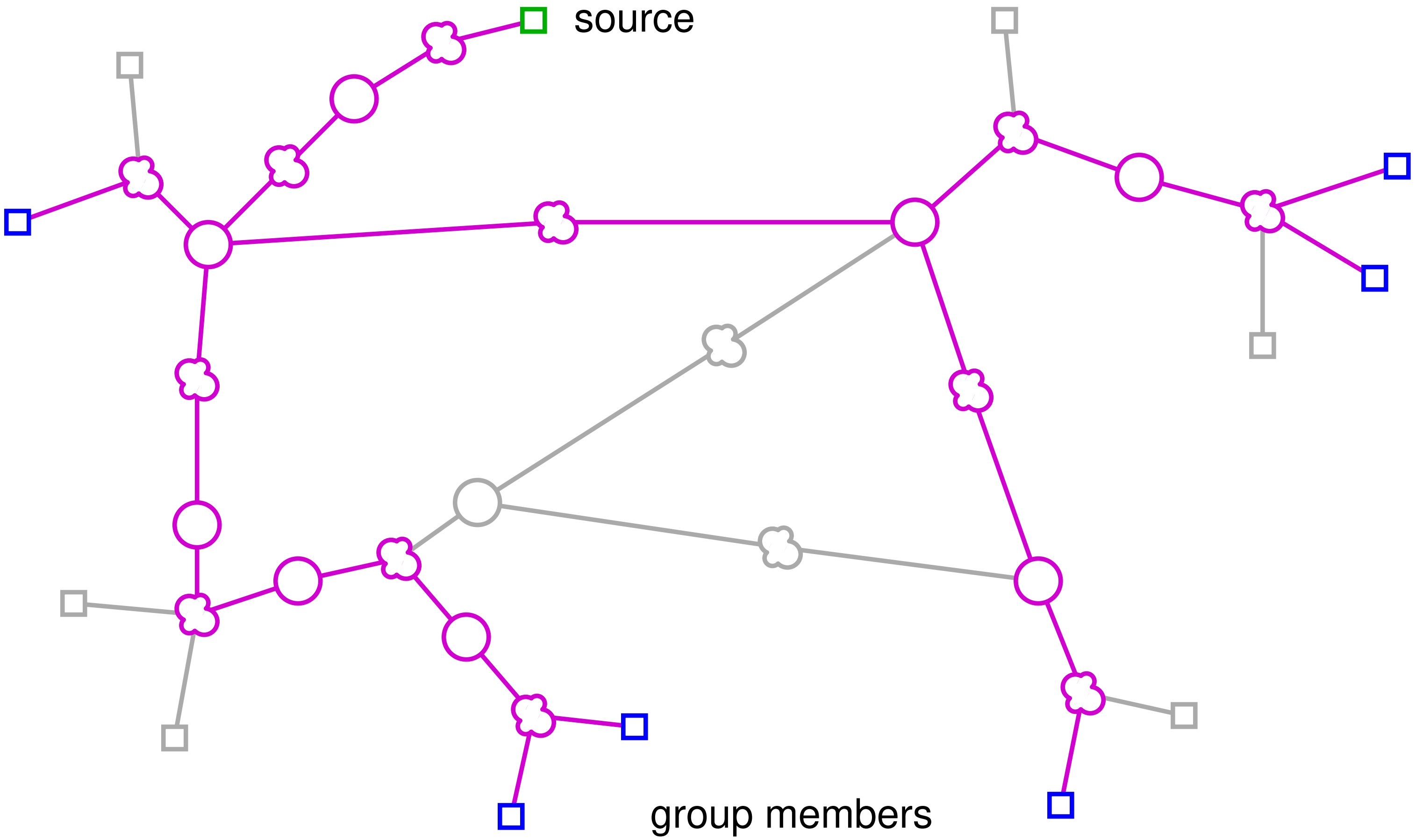


Packet delivery is *best-effort*.

Lost packets can be recovered *end-to-end* via feedback/repair.

Multicast

End hosts join a *group*, network constructs a *distribution tree*:



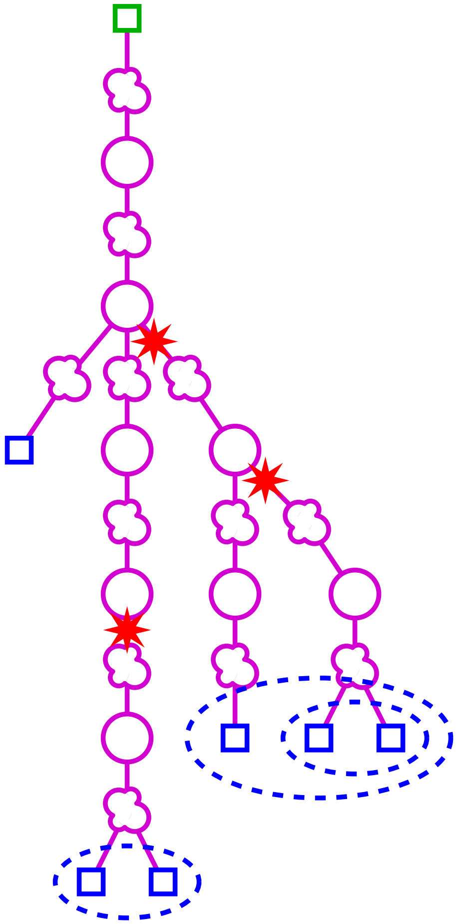
A packet loss affects members in the *loss subtree*.
Unicast repairs from source? Multicast? Neither scales.

Outline

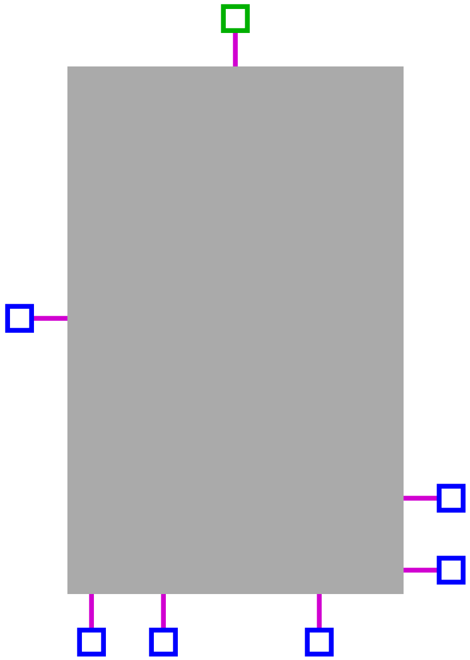
- approaches to multicast loss recovery
- specific schemes using new forwarding services
- general framework (taxonomy)
- research plan

Local groups

Members join multiple groups for common loss subtrees:

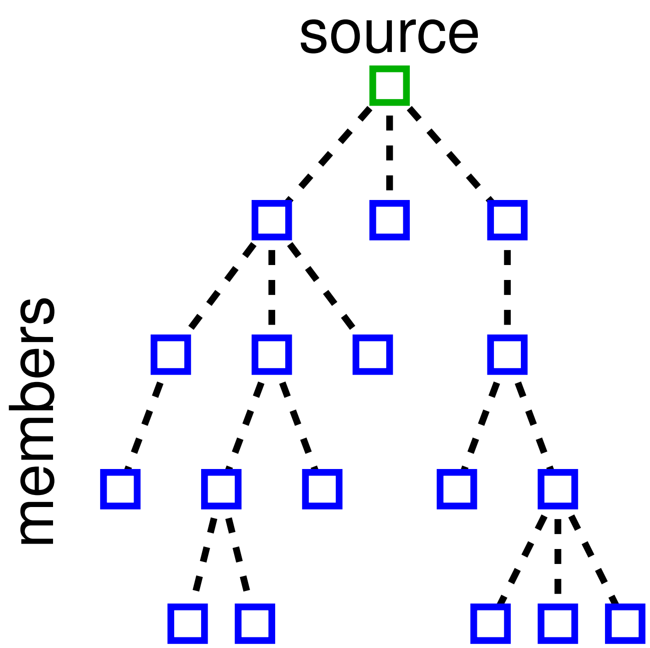


But the internet model is a black box:

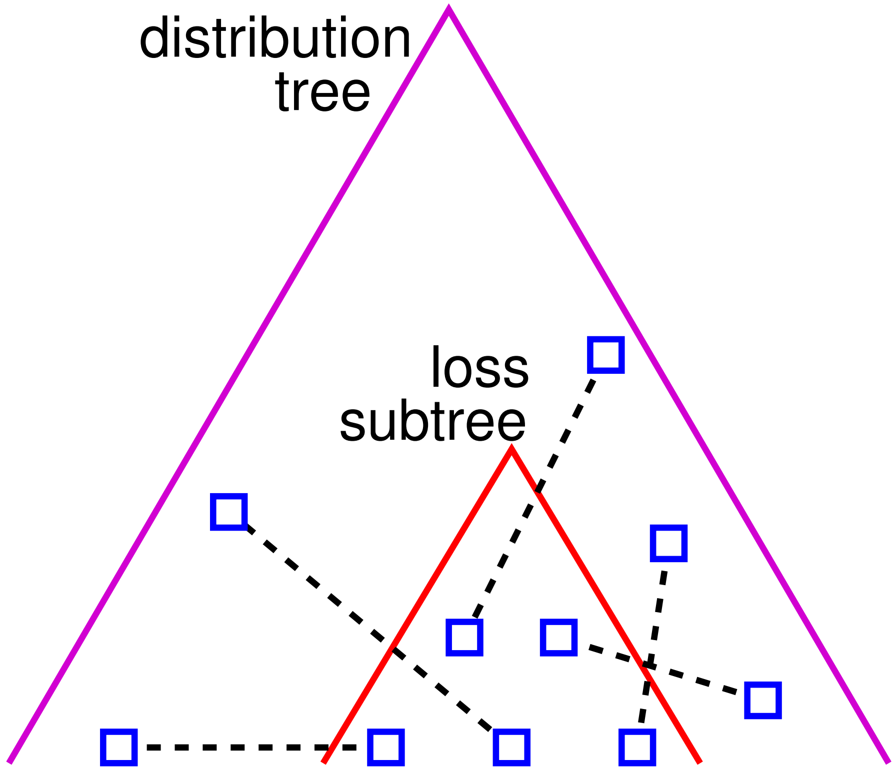


Hierarchy

Send feedback to parent member, repairs to children:



But without knowledge of topology, repairs converge:



Solution options

- **end hosts learn topology**
  e.g. *mtrace*, loss patterns
  departure from internet architecture
  but no changes to today's routers

- **transient subgroups** (breadcrumbs, PGM)
  per-packet state changes in routers

- **new forwarding services** ⇐ we like this one
  e.g. subcast, parentcast, randomcast
  lightweight, preserves layering, but still changes routers

network-layer part: new forwarding services
end-to-end part: feedback/repair protocol

Questions

- Which forwarding capabilities are worth supporting?

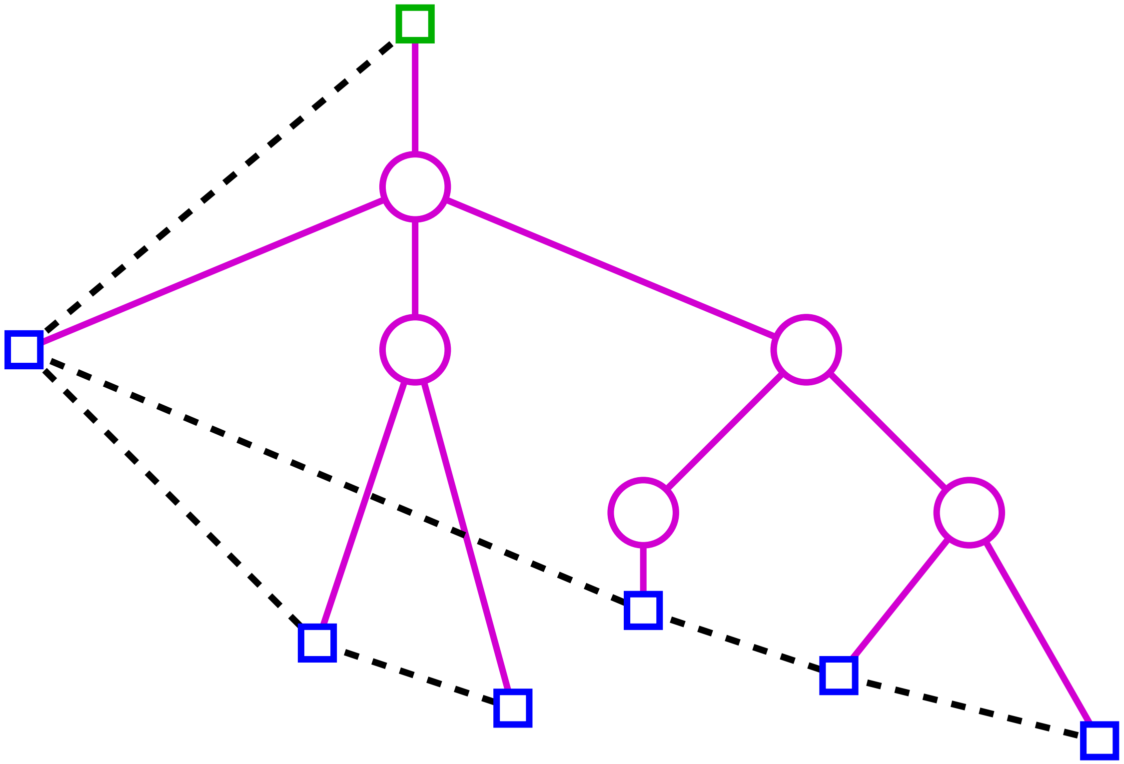
- Given a set of forwarding services and application requirements, what is the best loss recovery protocol?

Outline

- approaches to multicast loss recovery
- specific schemes using new forwarding services
- general framework (taxonomy)
- research plan

Subcast, RMTP [LP96]

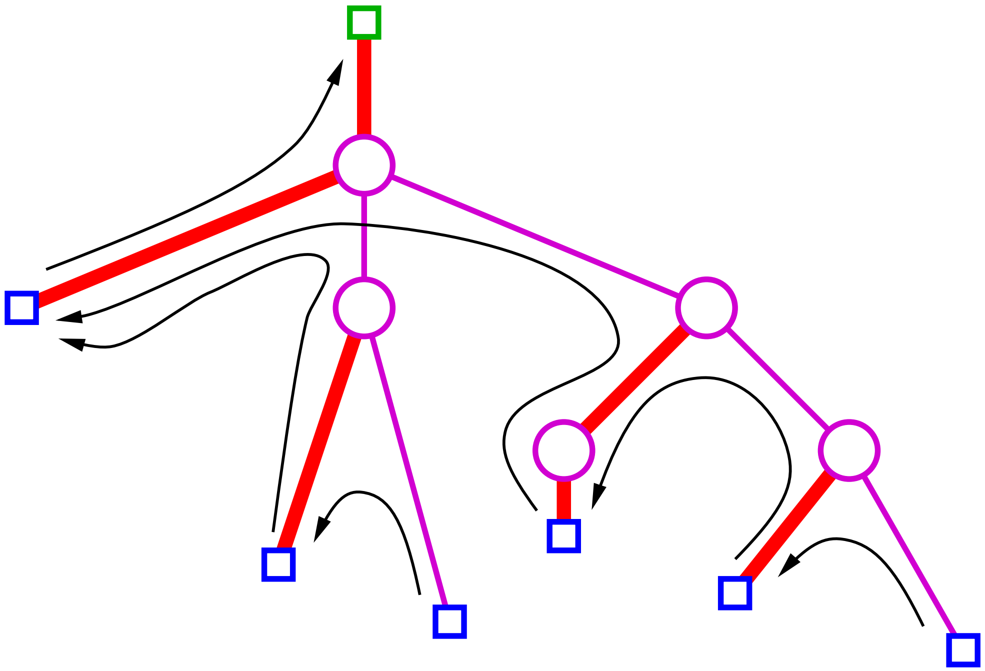
Assume hierarchy matches topology: member $+$ descendents $=$ subtree



subcast $=$ forward to the subtree below some point.

Parentcast, LMS [PPV98]

Network builds the hierarchy—each node flags one downward edge:



parentcast = forward up through non-flagged edge, down flagged edges, record non-flagged edge (*turning point*) in packet.

LMS: parentcast requests, subcast repairs below turning point.

Exactly one request escapes the loss subtree!

Other previous schemes

- **OTERS** [LC98]

  Like LMS, but instead of parentcast, end hosts use mtrace and subcast to construct the hierarchy.

- **AIM** [LG98]

  General forwarding service, can emulate parentcast and subcast.

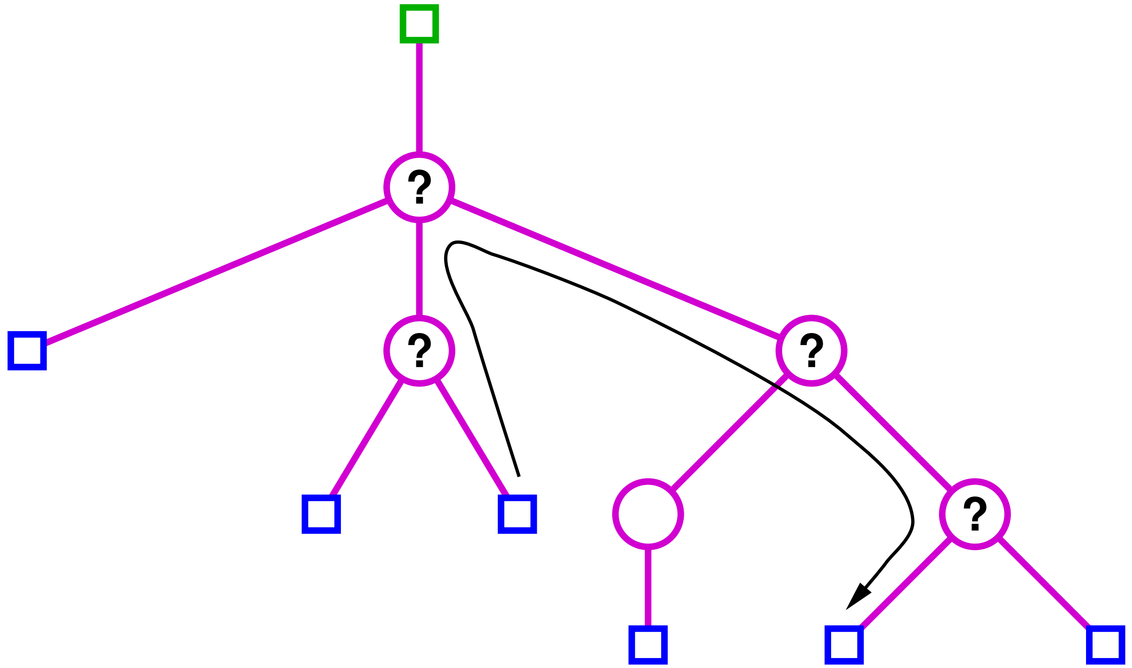
  Refers to points in the distribution tree by *positional labels* (path from root to point). Bidirectional shared trees can be re-hung from any point.

  RMA is like LMS, but on top of AIM.

New service: Randomcast [CM99]

With a hierarchy, every subtree depends on two members.
Can we do without a hierarchy?

randomcast = random walk in distribution tree without backtracking:



(optionally record turning point)

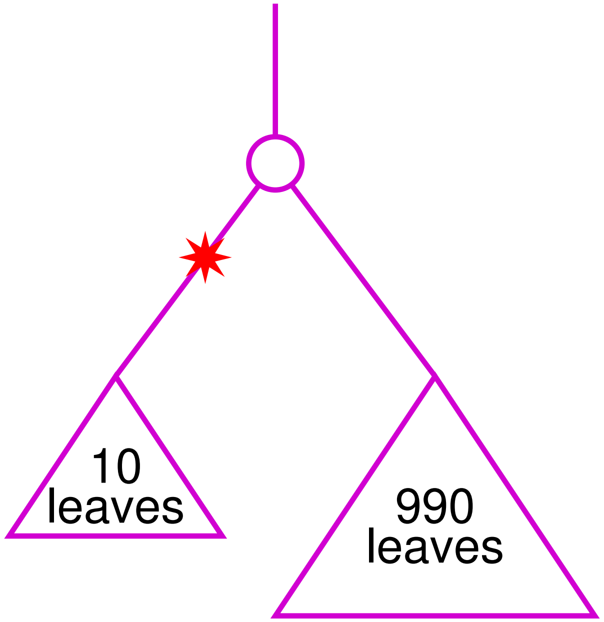
parentcast: one escapee always
randomcast: one escapee on average (with sensibly weighted coins)

Randomcast: Up or down?

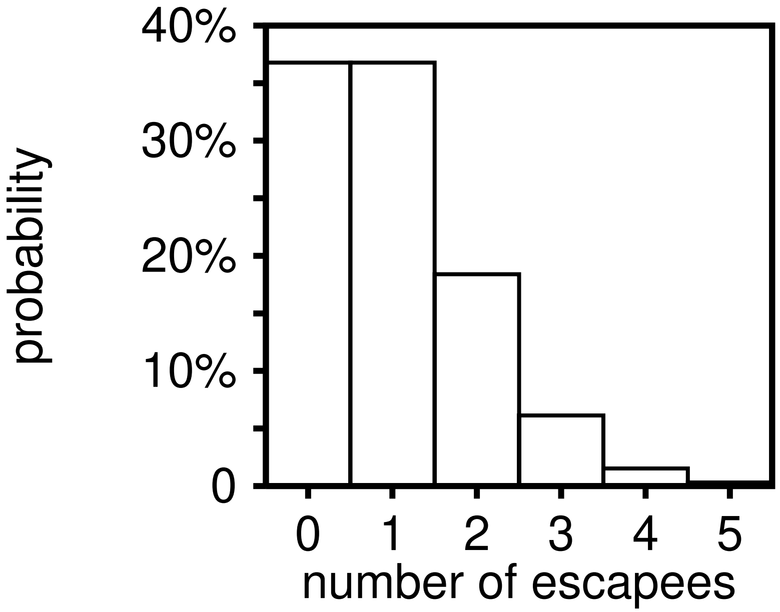
uniform: all packets have equal chance to go up
weighted: by number of members below arrival edge

Weighted requires knowledge of populations, but has advantages:

For subcast responses, smaller audience:



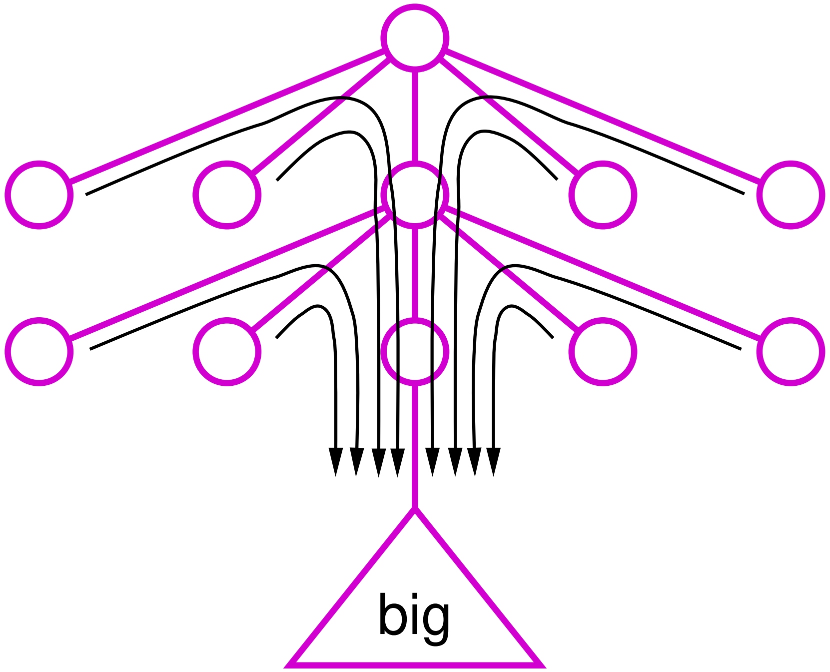
Number of escapees has known distribution (binomial ≈ Poisson):



Randomcast: Which downward edge?

uniform: each edge has equal chance
weighted: by number of members below the edge

Weighted can concentrate traffic:

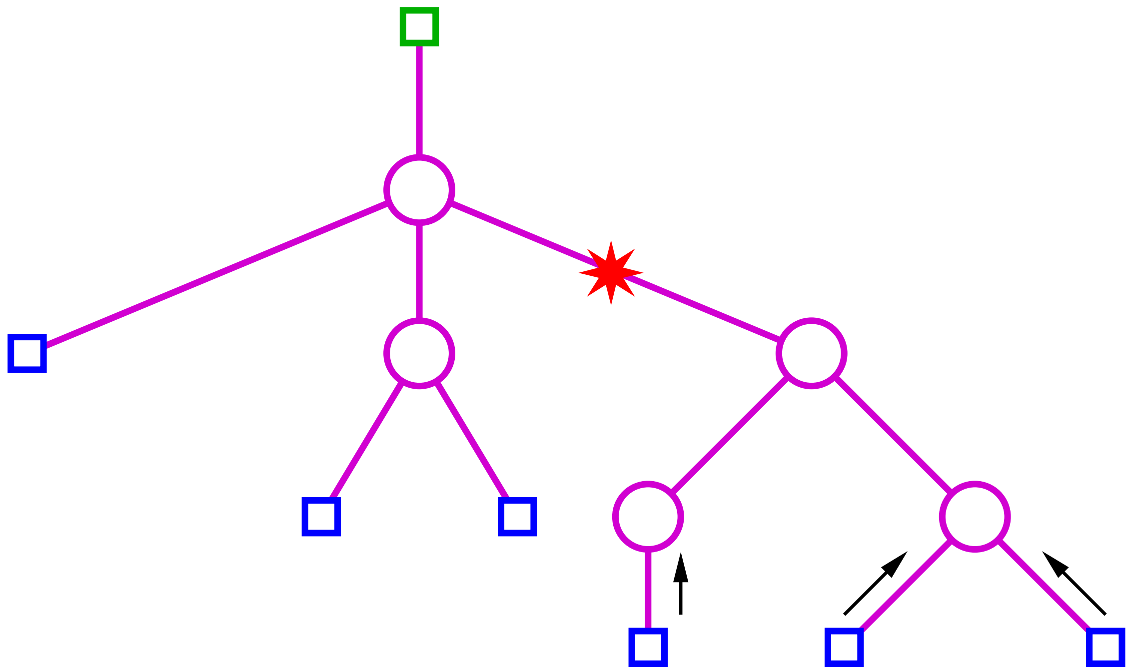


Populous subtrees tend to be deeper $\Rightarrow$ slower response time.

Search Party [CM99]

After a loss, members randomcast requests at rate μ (Poisson).

Responses are subcast using the turning point, like in LMS.



Responsibility is diffuse (no member is important).

Search Party: Analysis

Let X_i = node at height i above member M.
Let L_i = the number of leaves below X_i.
$P(M$'s request exceeds height $i) = \frac{1}{L_i}$.

If the number sent by each member in subtree below X_i is Poisson with mean m, the number that escape is Poisson with mean $\frac{m}{L_i} L_i = m$.

If X_i receives requets at rate $< 2\mu$ from above and has $d > 1$ downward edges, each one draws $< \frac{\mu}{d-1}$ from each other and $< \frac{2\mu}{d}$ from above, which is $< 2\mu$ total.

Let r = request/response round-trip time.
Expected responses per loss is $x = 1 + r\mu$ (1 for the first, plus during its round-trip another expected $r\mu$ requests are sent per member).

$P($no response before time t after loss detection$) = 1$ for $t < r$, or $e^{(r-t)\mu}$ for $t \geq r$, so expected recovery time is $y = \int_{\infty}^{0} t\, d\,(p(t)) = r + \frac{1}{\mu}$.

So $(x-1)(y-r) = r$ (extra responses $\times$ extra delay = round-trip time).

For expected response recipients (exposure), see UCB//CSD-98-1011.

Search Party: Performance

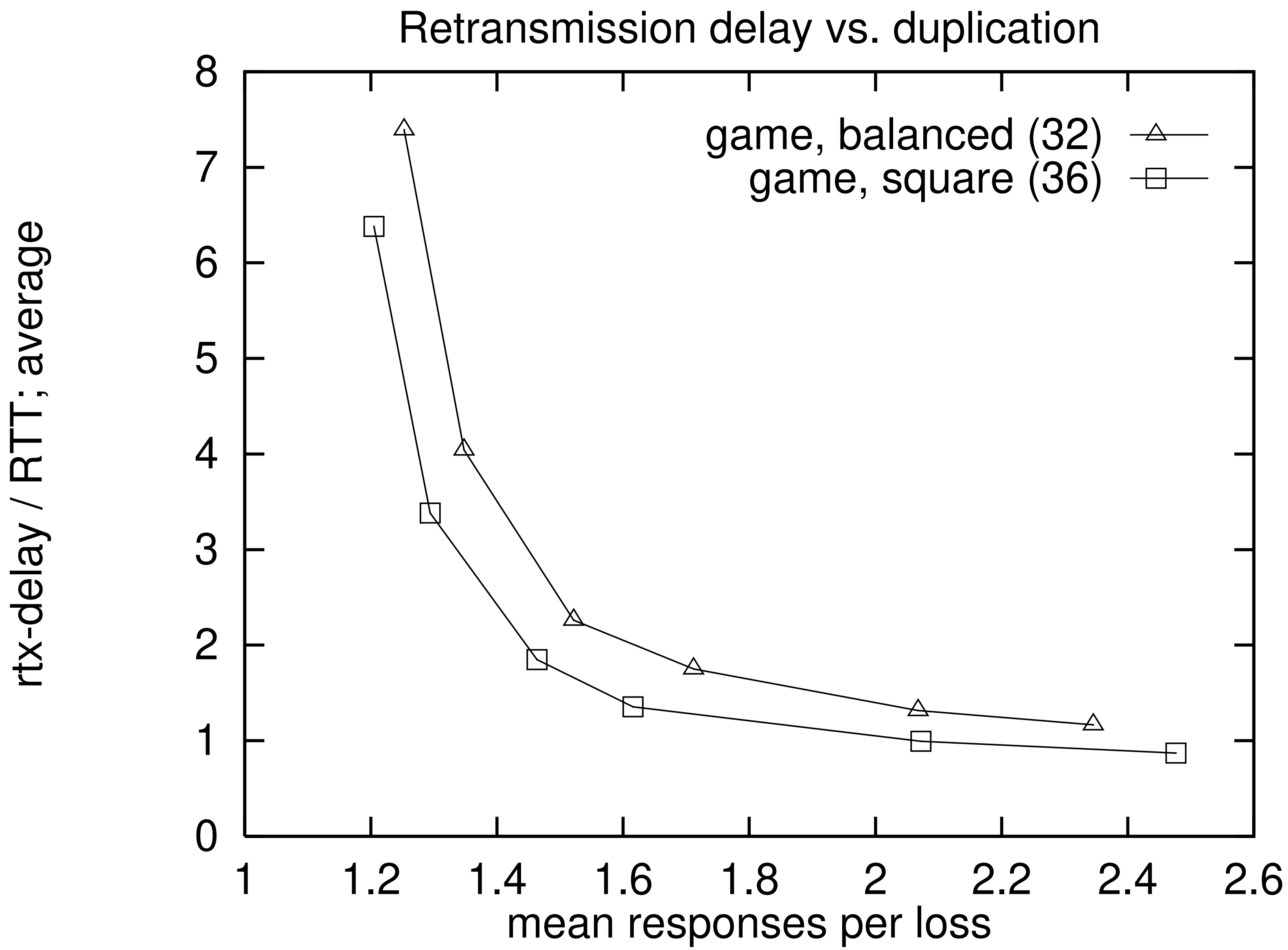
- request traffic (mean):
  outside loss subtree: μ total
  inside, upward: μ on each edge
  inside, downward: $< 2\mu$ on each edge

- *exposure* = $\dfrac{\text{response subtree population}}{\text{loss subtree population}}$

  LMS: average exposure $\leq$ group size
  Search Party: average exposure $\leq 1+$ ln(group size)

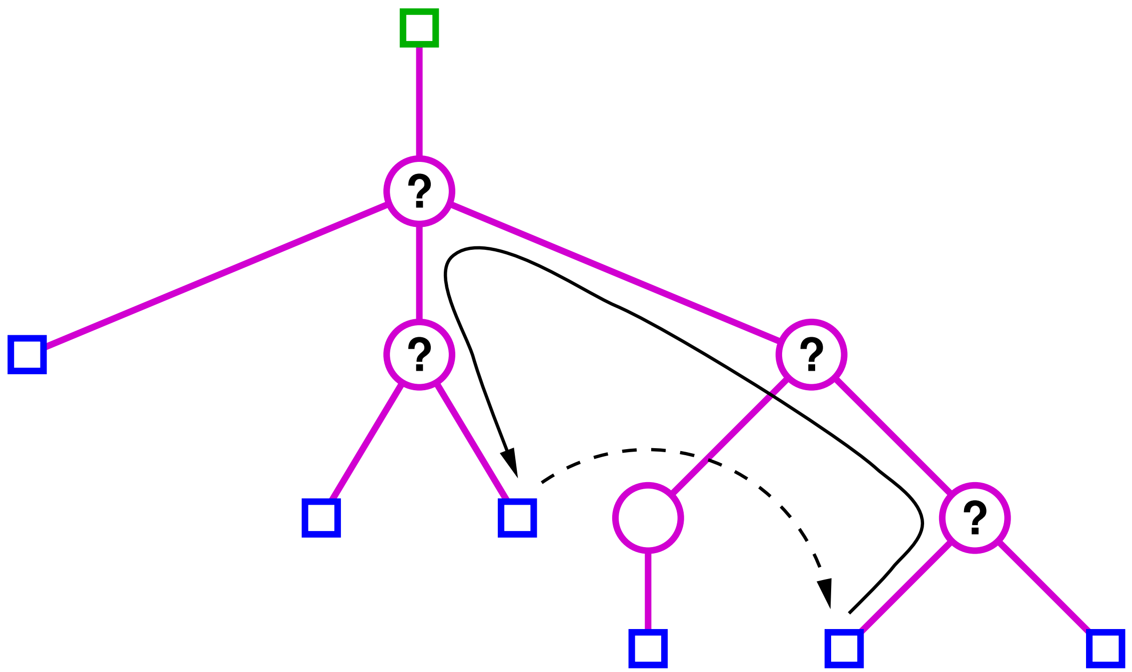
- recovery time and response duplication:
  depend on round-trip times and μ
  not on group size or loss subtree population

Search Party: Delay/overhead trade-off



Rumor Mill

Requests use *trivial randomcast* (uniform up/down, no turning point).
Responses are unicast back to the requestor.



- Data spreads like a rumor from member to member.
- Delay and overhead are worse than Search Party by a factor of about log(loss subtree population); trade-off is still tunable.
- exposure $= 1$
- Members do not need permission to send to multiple destinations.

Outline

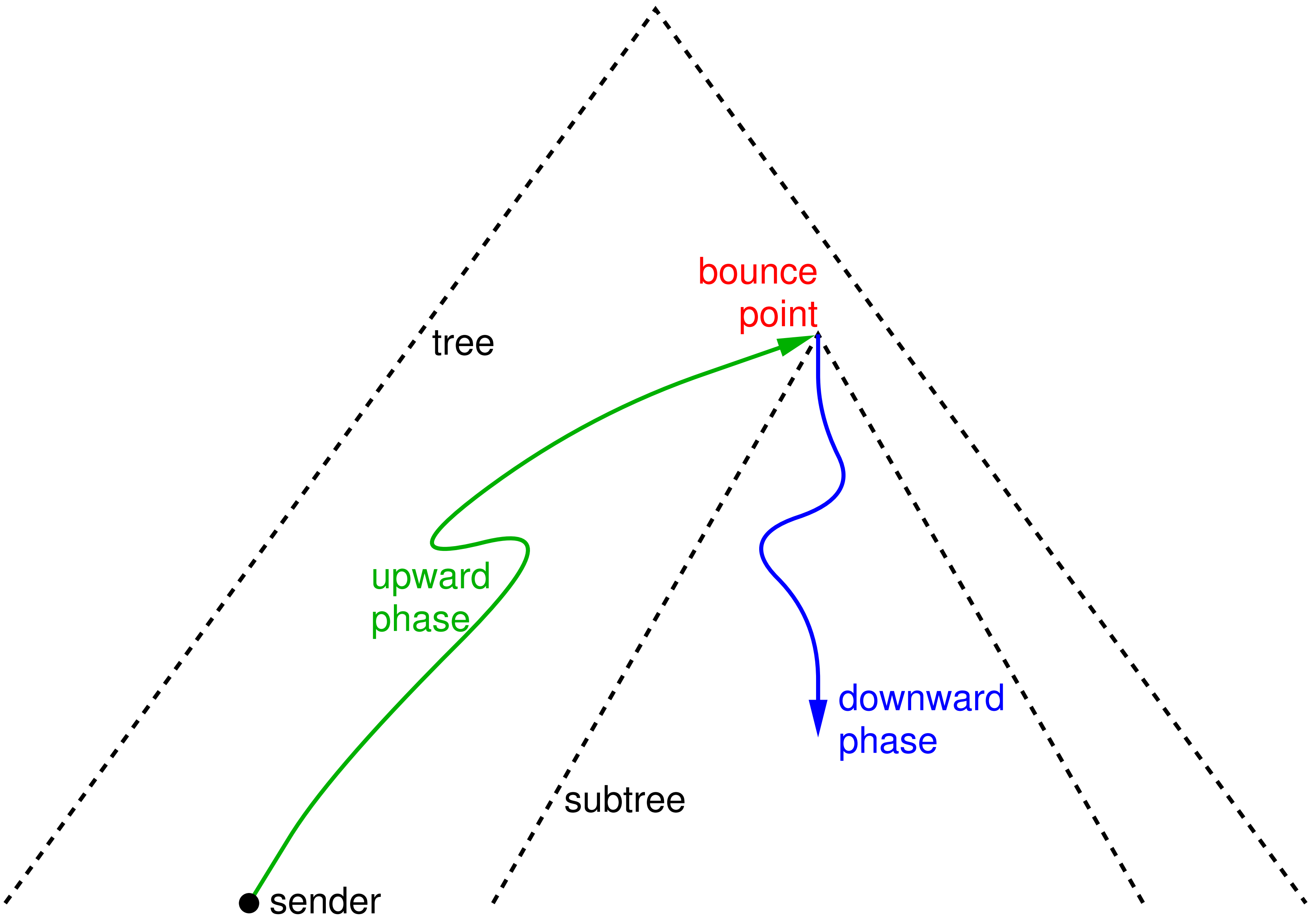
- approaches to multicast loss recovery
- specific schemes using new forwarding services
- general framework (taxonomy)
- research plan

Framework

- network-layer part: *treecast*

- end-to-end part: component architecture

Treecast


bounce
point
tree
upward
phase
downward
phase
subtree
sender

Treecast mechanisms

up_rule	bounce_rule	down_rule
root	entrance_address	all
address *address*	top_address	
position *label*	position	
height *distance*	remaining_hops	
depth *distance*	depth	
random *distribution*	parity	random *distribution*
hierarchy *metric*		hierarchy *metric*

Treecast examples

	up_rule	down_rule
subcast	address, height, depth	all
parentcast	hierarchy	hierarchy
randomcast	random	random
pachinkocast	address, height, depth	random
???cast	hierarchy	random
.		
.		
.		

Component architecture

The end-to-end part of loss recovery schemes can be decomposed.

Components shared by all protocols:

- **network endpoint** (provides unicast, multicast, treecast)
- **core** (buffers data, detects losses, demuxes packets. . .)

Components whose behavior defines the protocol:

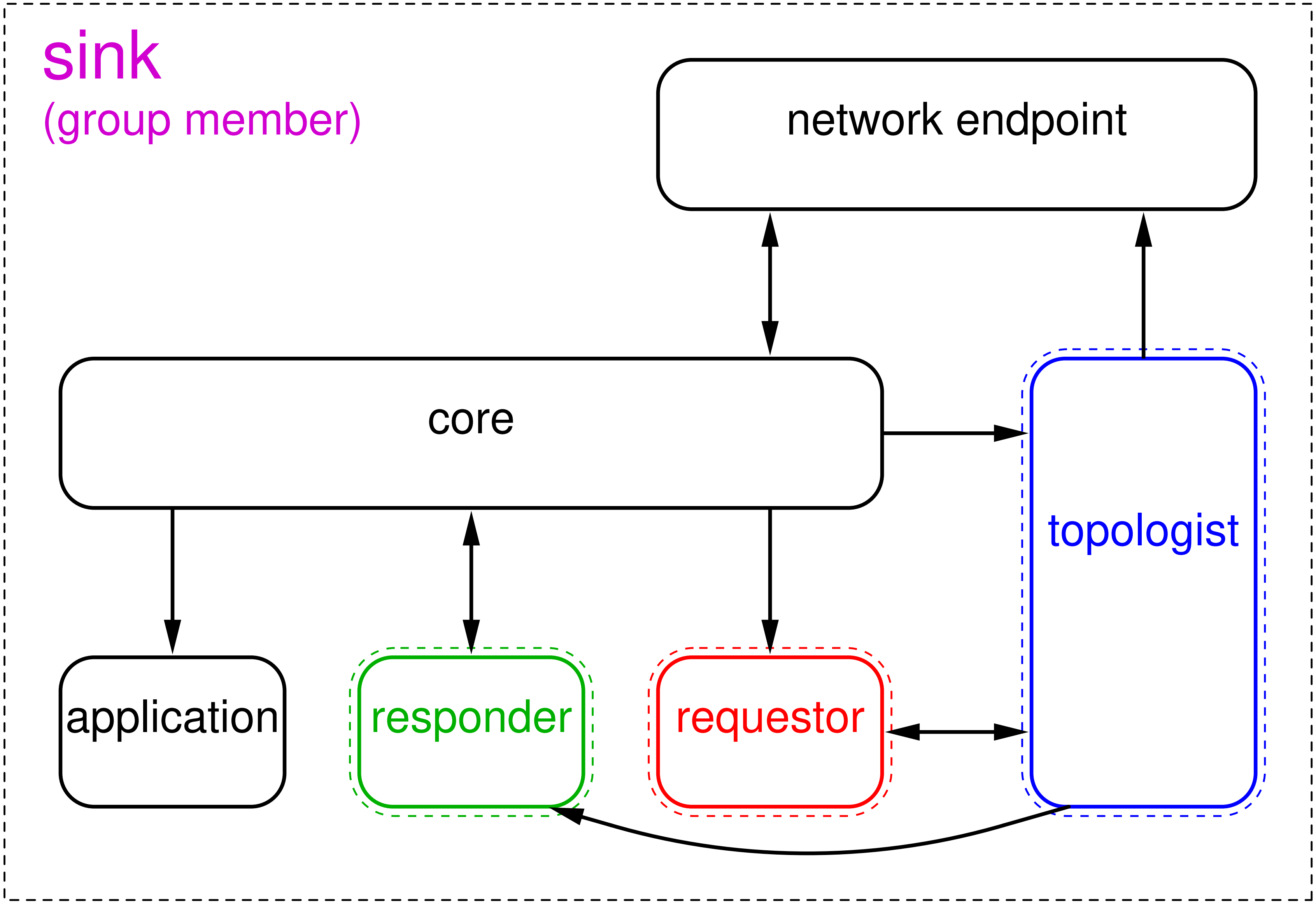
- **requestor** (determines what/when to request)
- **topologist** (determines request destinations)
- **responder** (determines response destinations)

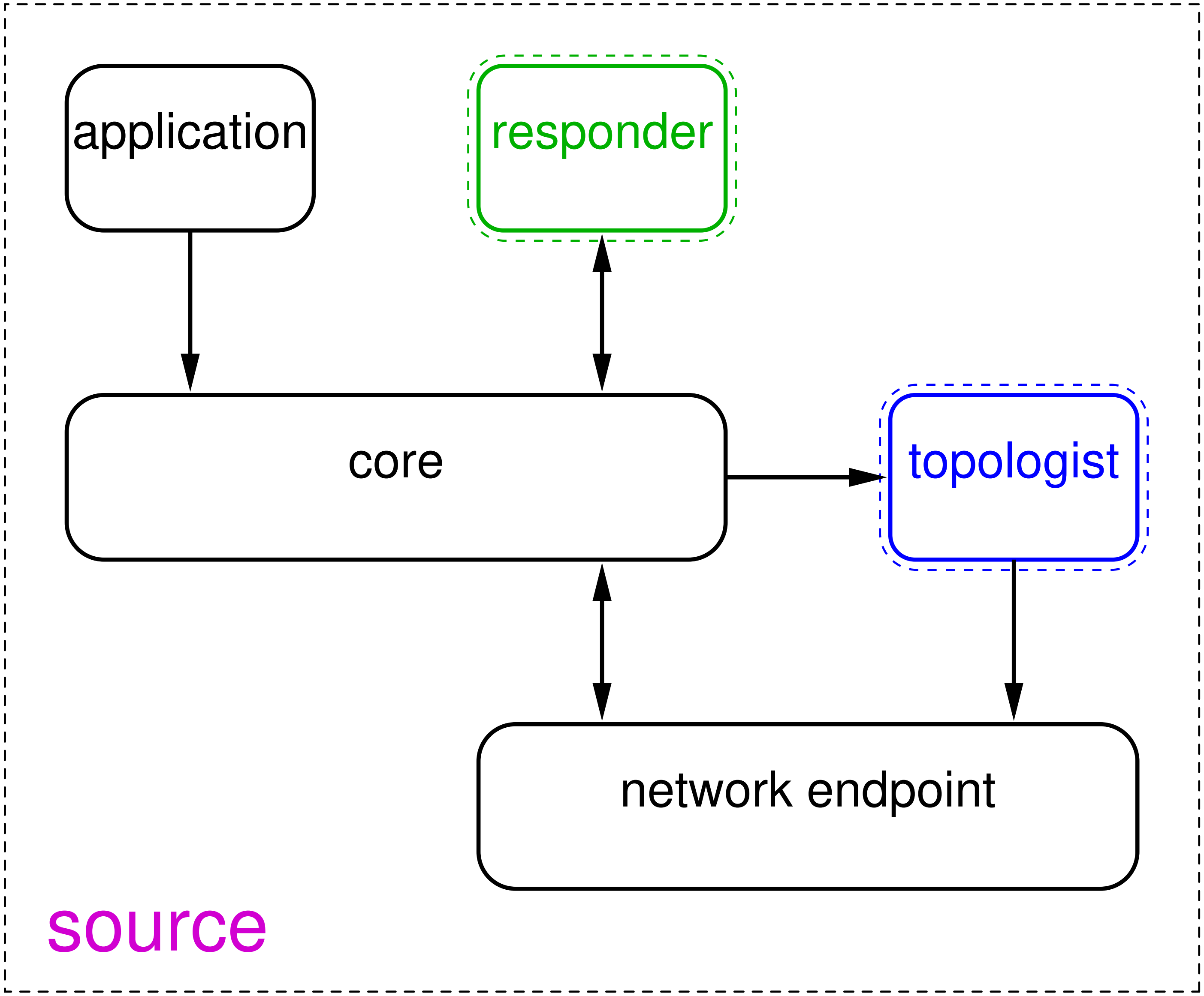
Pluggable components . . .

. . . ease implementation through code-reuse.

. . . facilitate fair comparisons between protocols.

. . . in new combinations generate new protocols.


sink
(group member)
network endpoint
core
topologist
application
responder
requestor


application
responder
core
topologist
network endpoint
source

Framework examples

	requestor	responder	topologist
LMS	adapt to RTT	subcast	parentcast
OTERS	adapt to RTT	subcast	build hierarchy
Search Party	adapt & tune	subcast	randomcast
Rumor Mill	adapt & tune	unicast	trivial randomcast
newer ideas	deferrable requests; aggregate requests	subcast & unicast; relay responses	parentcast & unicast; vary hierarchy building method

Research plan

1999-May,Jun
Finish Rumor Mill & submit.

1999-Jul,Aug
Work for Sony CSL in Japan (unrelated).

1999-Sep,Oct,Nov
Build framework as simulation using **ns**.
Implement existing & new protocols.

1999-Dec, 2000-Jan
Use simulation to investigate:
Best way to build hierarchy? Best way to use a hierarchy?
How well can we do without a hierarchy?
Which treecast mechanisms are most worthwhile?

Research plan (cont.)

2000-Feb,Mar,Apr

Rebuild framework as a testbed:

Overlay a treecast backbone (Tbone) on the wide-area internet.

Maybe from scratch with dummy applications; maybe modify libsrm and inherit existing applications.

2000-May,Jun

Reexamine the simulation results using the testbed.

2000-Jul,Aug,Sep,Oct

Write the dissertation.

Conclusion

- IP multicast service model is insufficient for loss recovery.
- One solution is to add forwarding services.
- General framework:

  network-layer part: treecast

  end-to-end part: component architecture

  - taxonomy of existing schemes
  - roadmap of the design space
- We will build the framework as a simulation & testbed in order to . . .

  . . . fairly compare schemes.

  . . . explore design options in the network and end hosts.

  . . . identify the most useful forwarding mechanisms.

  . . . determine the best way to use the forwarding mechanisms.