Search Party and Rumor Mill:
Using Randomcast for Reliable Multicast with Local Recovery

Adam M. Costello

1998-Jul-15-Wed

Problem: Multicast error recovery that scales to *arbitrarily* large groups.

- requirements
  - avoid NAK implosion (easy)
  - local recovery (trickier)
- challenges
  - only routers know topology
  - no one knows where loss occurred
  - avoid duplicate retransmissions
  - low retransmission delays

Problem **not** addressed: congestion.

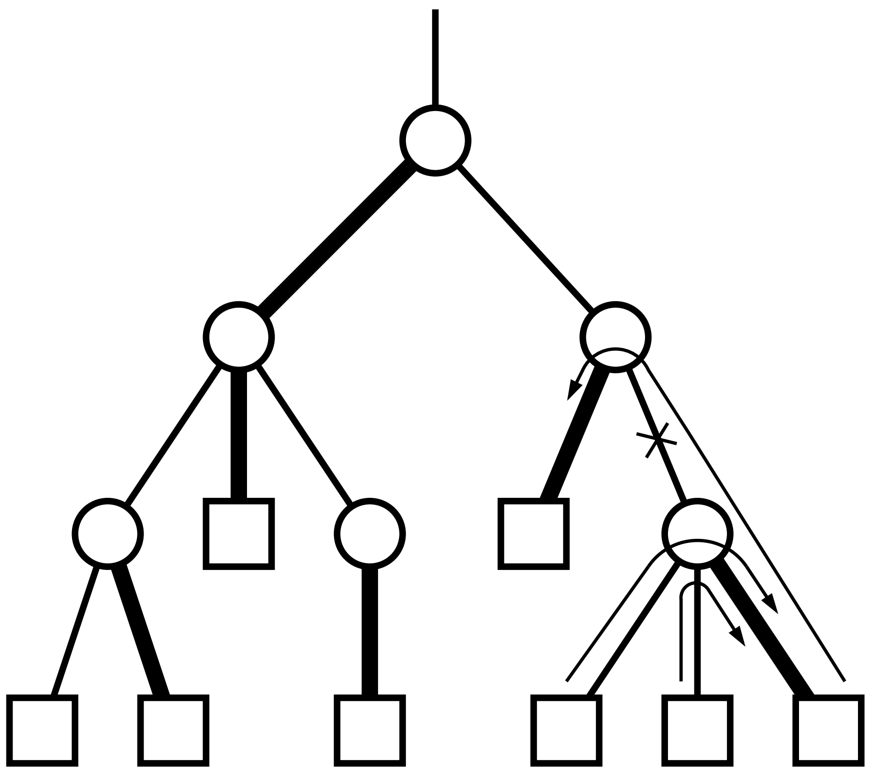
Outline

- LMS (Papadopoulos)
- Trivial Randomcast
- Randomcast
- Search Party
- Rumor Mill
- Future Work
- Conclusions

LMS (Papadopoulos et. al. Infocom'98)

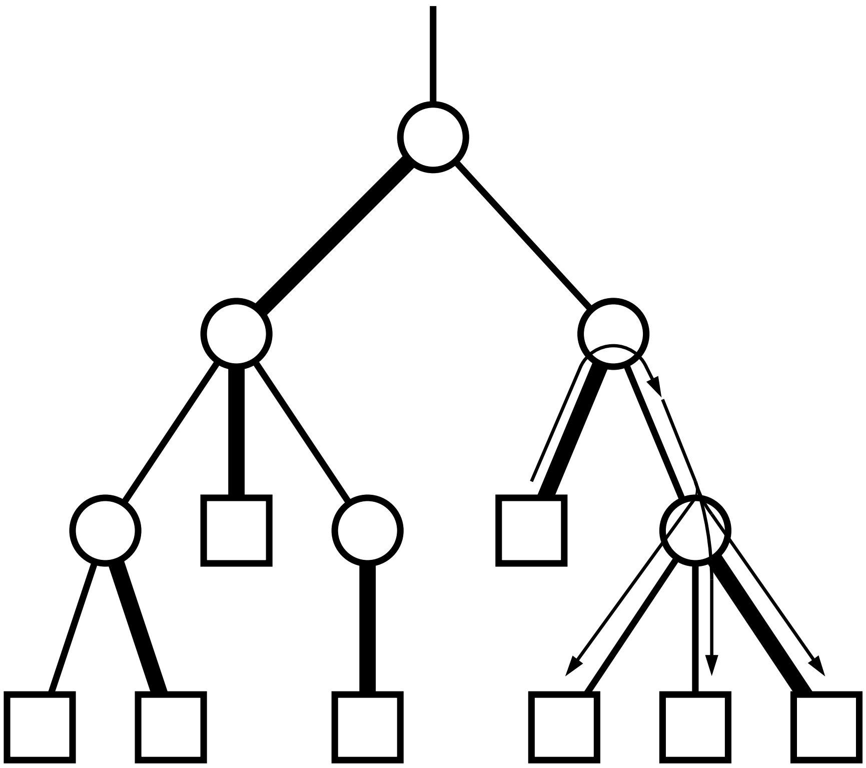
New forwarding service for requests:

Router chooses one downward *replier link*.
Forward requests to replier links, except:
Forward requests from replier links upward.
Insert *turning point* link ID into request.



LMS (cont.)

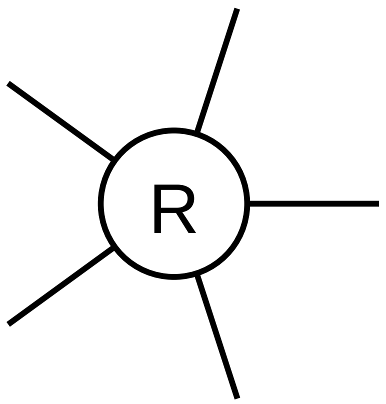
New forwarding service for responses:
Response goes to subtree below recorded link.



Hosts need not know topology.
Routers need not know protocol.
Many requests, but always one response!
Only one round-trip.
But responsibility is concentrated.
What if no response comes?

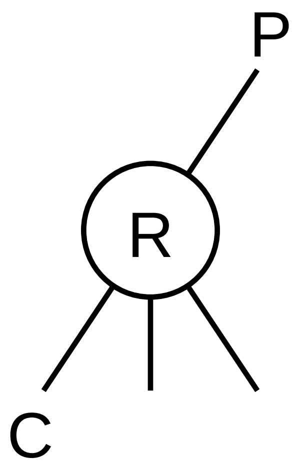
Trivial Randomcast

New forwarding service: random tree-walk



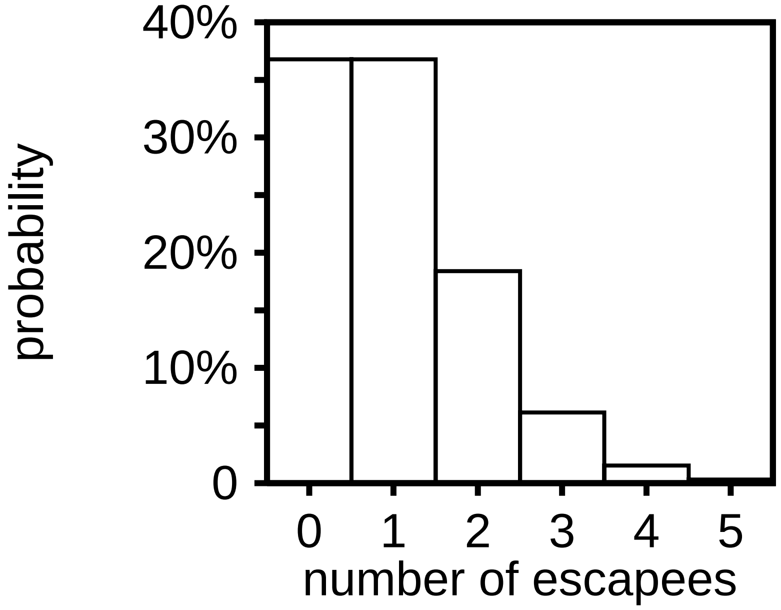
Forward to a random (uniform) link, but don't backtrack.

Randomcast

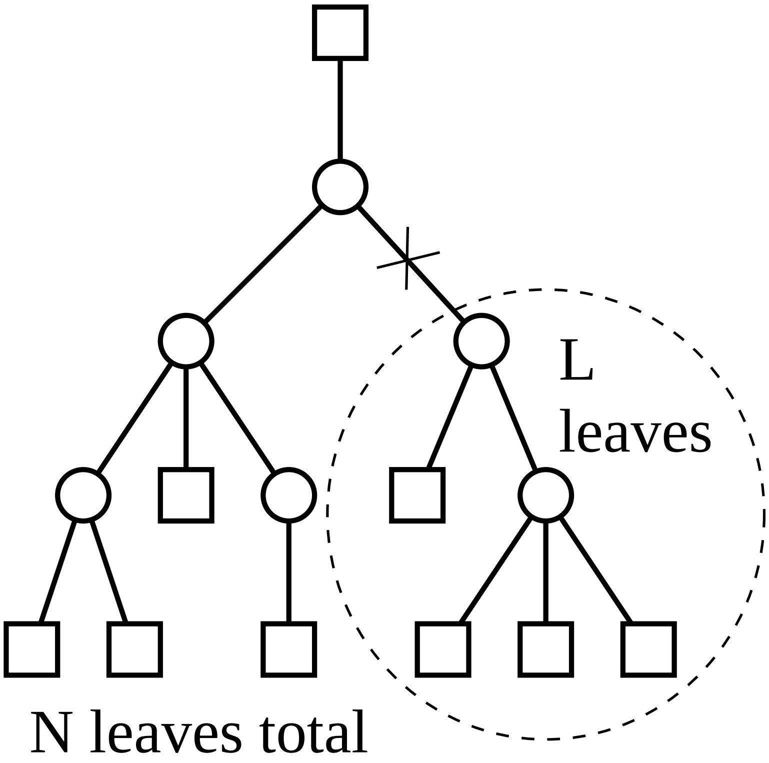


From C ⇒
Forward to P w/prob size(C)/size(R)
else forward to any other child (uniformly)
and write location info into packet.

From P ⇒ Forward to any child (uniformly).



Search Party



- randomcast requests at rate μ (Poisson)

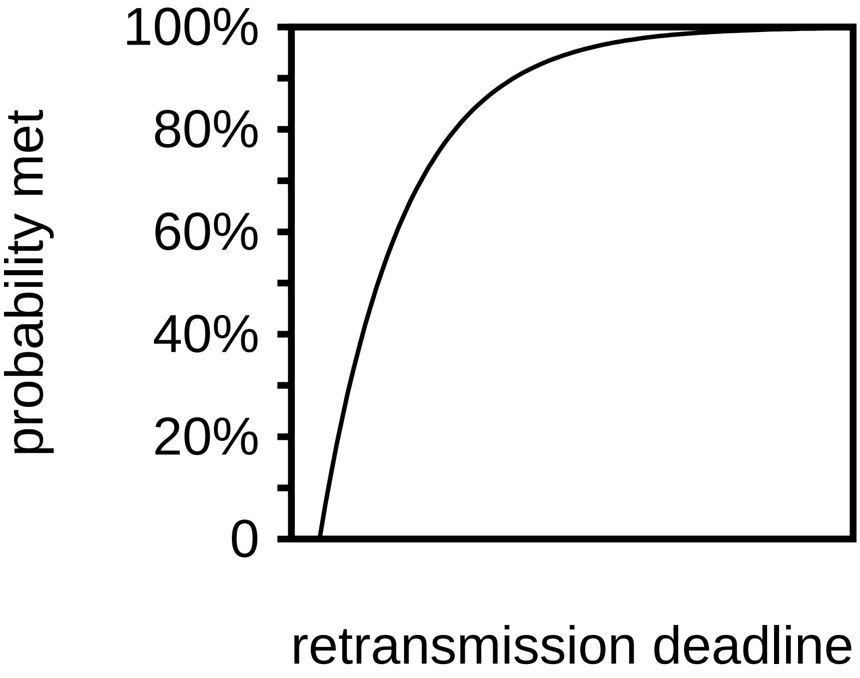
- subcast responses, like LMS

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

- response locality: exposure $\leq 1 + \ln \frac{N}{L}$

Search Party (cont.)

- retransmission delay:
  depends on RTT, μ
  not on L, N



- tunable delay/overhead tradeoff via μ

- responsibility is diffuse (no one is important)

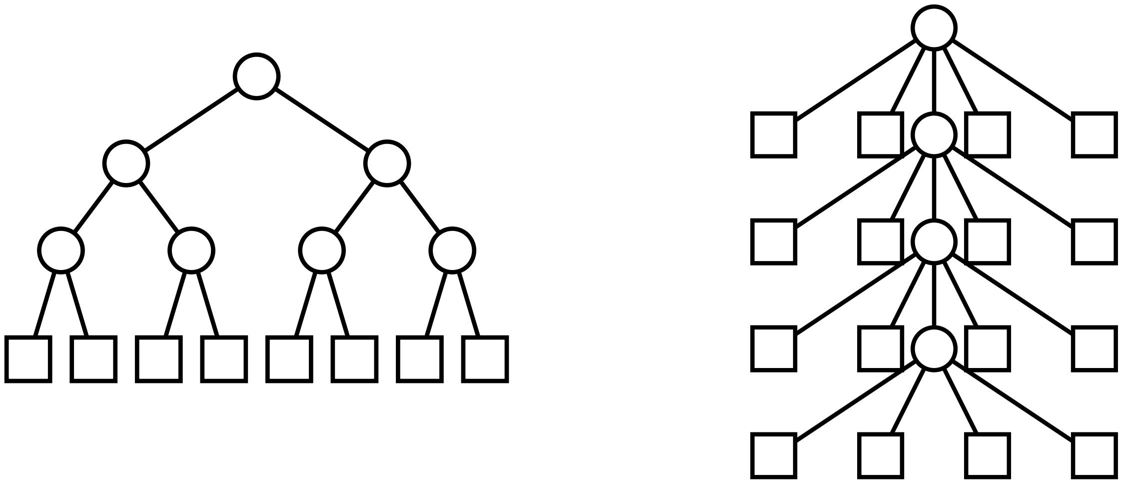
Analysis

http://www.cs.berkeley.edu/~amc/research/search-party/

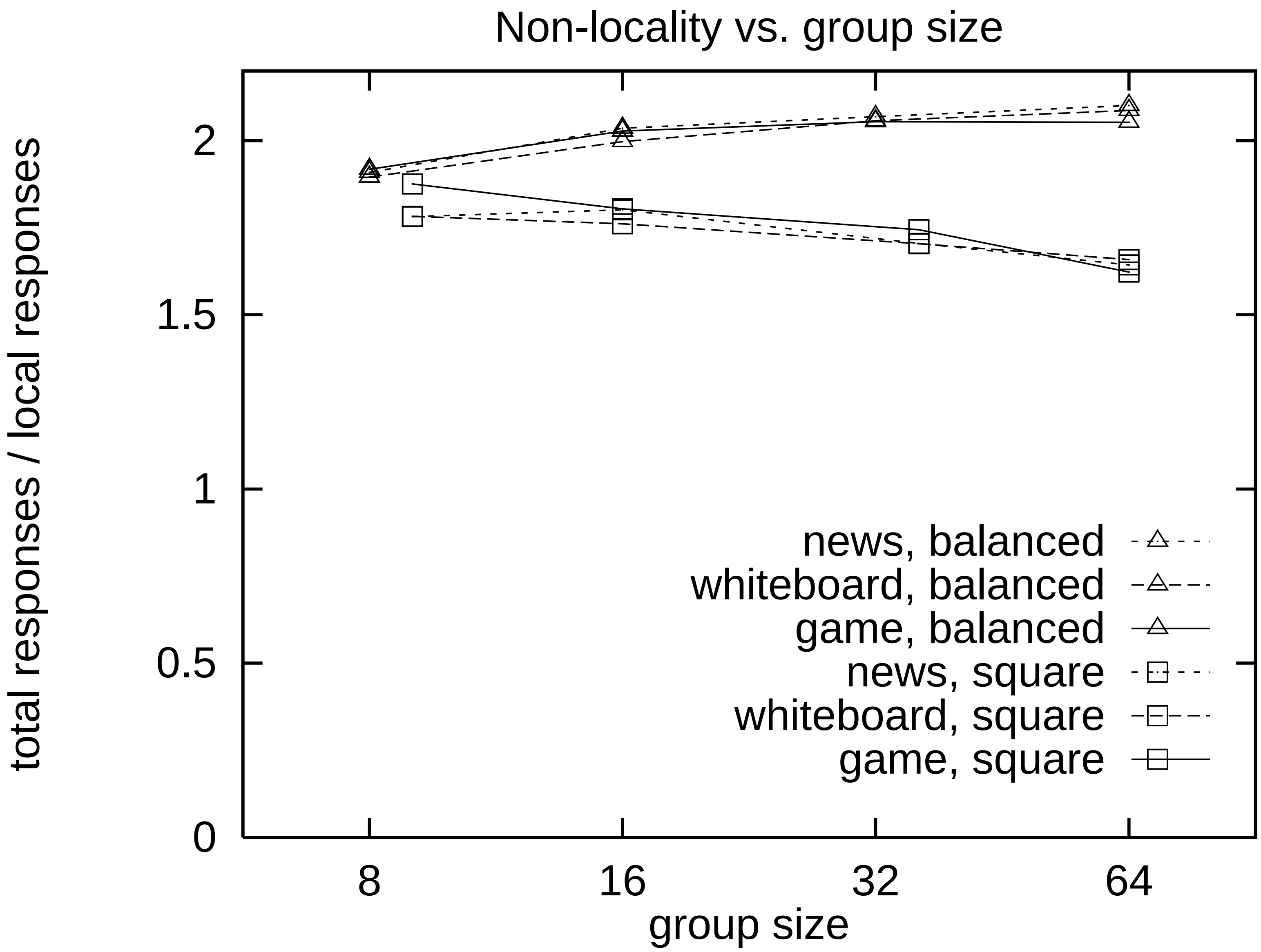
Simulation (**ns**)

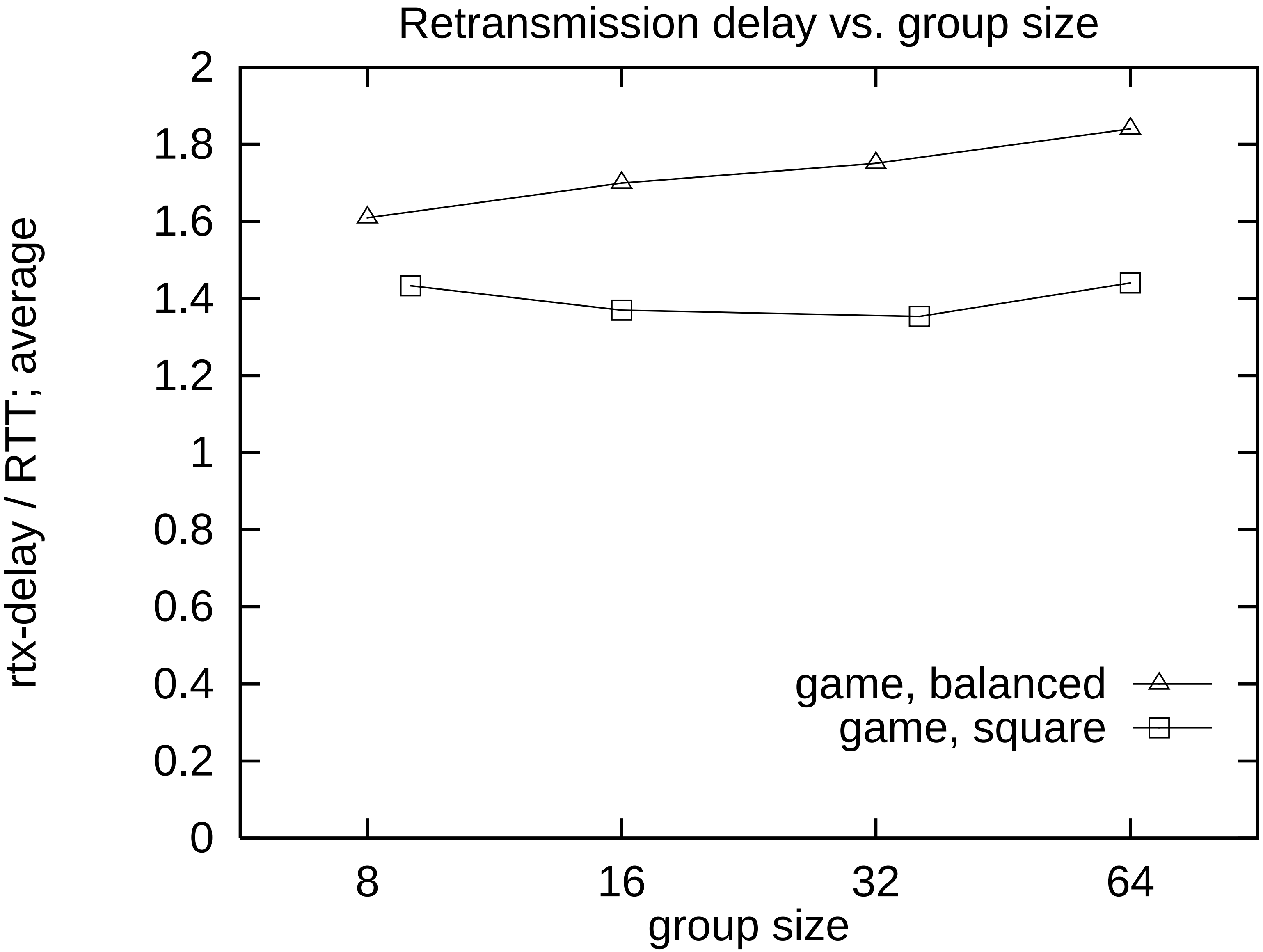
- links are all the same:
  10 ms propagation delay
  background Poisson traffic causes
  2.2% loss rate
  queue delay: 20 ms mean, 15 ms std.dev.

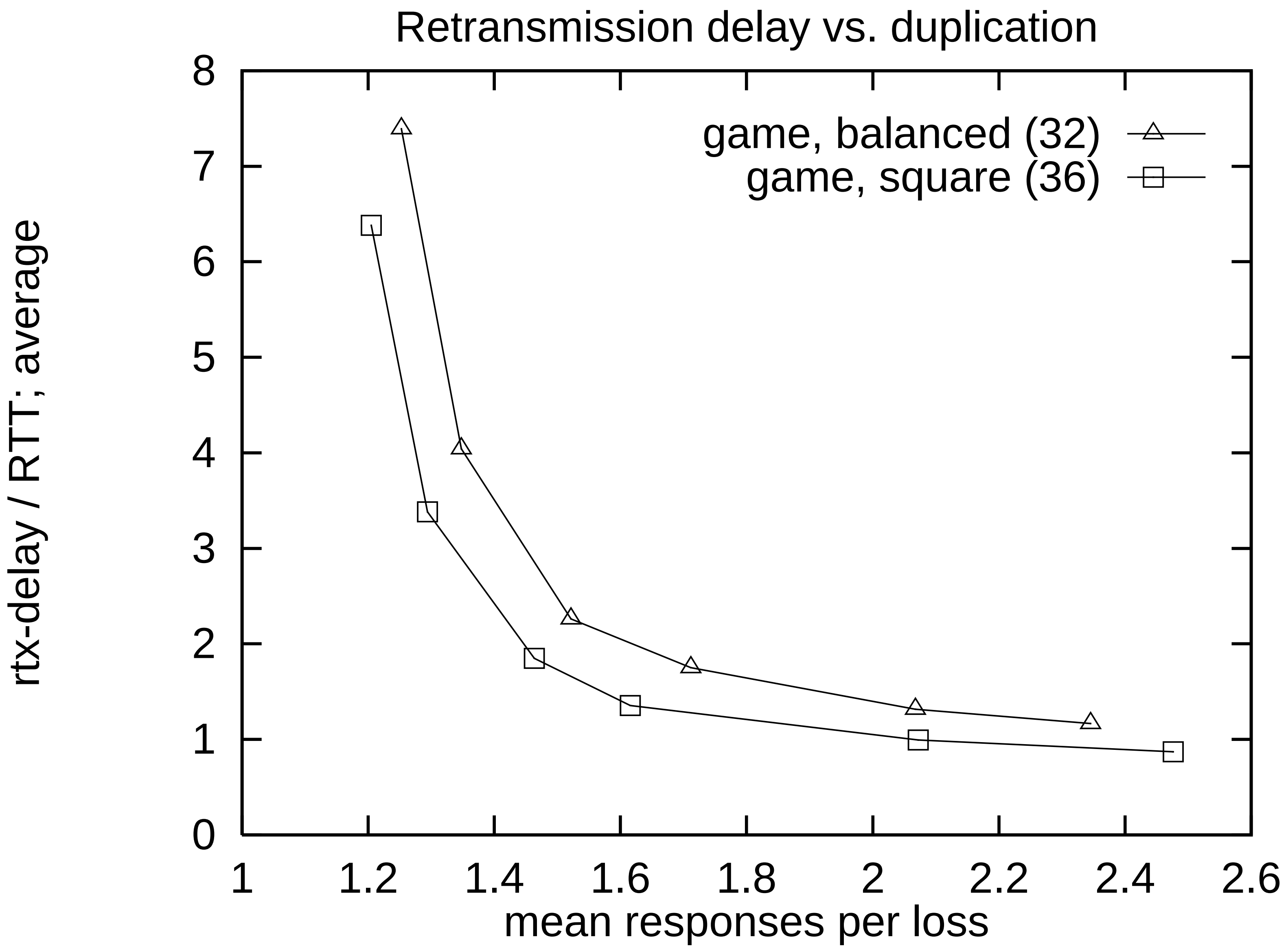
- topologies:



- applications (μ tuning goals):
  news: rtx delay $>$ 10 min w/prob 10^{-9}
  whiteboard: rtx delay $>$ 5 sec w/prob 10%
  game: repair ASAP but avg responses < 4


Non-locality vs. group size
total responses / local responses
0
0.5
1
1.5
2
8
16
32
64
group size
news, balanced
whiteboard, balanced
game, balanced
news, square
whiteboard, square
game, square


Retransmission delay vs. group size
rtx-delay / RTT; average
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
8
16
32
64
group size
game, balanced
game, square


Retransmission delay vs. duplication
game, balanced (32)
game, square (36)
rtx-delay / RTT; average
mean responses per loss
0
1
2
3
4
5
6
7
8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6

Rumor Mill

- trivial-randomcast requests at rate μ until response

- *unicast* responses (no subcast!)

- request traffic: μ per link (up & down) inside loss subtree

- response locality: perfect

- retransmission delay: $O(\text{RTT} \cdot logL)$

- tunable delay/overhead tradeoff via μ

- responsibility is diffuse (no one is important)

Future Work

- simulate thousands of nodes in C
- simulate Rumor Mill
- analyze Rumor Mill for particular topologies?

Conclusions

- Randomized forwarding is useful for local recovery.

- Search Party is like LMS in structure, complexity, and performance, but uses randomcast to improve robustness.

- Rumor Mill has higher latency, but is much simpler and respects source-specific joins.