Background

- P2p multicast: Bullet, SplitStream [SOSP’03]
- Existing systems rely heavily on cooperation
- Most incentivized solutions not suitable
  - Storage: Auditing [IPTPS’03], Samsara [SOSP’03]
  - Bandwidth: BitTorrent [P2pEcon’03]
- Do not map onto multicast fairness
  - Rely on tit-for-tat
  - Multicast trees are usually static
SplitStream Concept

Exploit that the forest trees can be interior-node disjoint

Stripe 1

Stripe 2
SplitStream Reliability

Peer failure only affects a single stripe
Freeloading Model

- Assume rationality
  - Selfish, but not malicious, freeloaders
- Nodes can refuse to accept children
- Nodes can refuse forwarding data

I can't take any more children.
I can't receive any data, so I can't forward anything to you.

No problem! I will accept you.

A, can you be my parent, please?

F

A
Design Overview

- Distinguish nodes with selfish behavior
  - Reduce the quality of service of selfish nodes
- Goal: freeloaders should not receive more data than they send
- Make judgment only by observing behaviors
  - Avoiding many thorny trust issues
Design Overview (cont.)

- Periodic tree reconstructions
  - Avoid suffering forever
  - Potentially reversing parent-child relationships
- Measure various metrics of other nodes
  - Combine to form a robust policy
Pairwise Debt
Ancestor Rating (Confidence)
Parental Availability (PA)

……

A made itself available as parent

A can you be my parent, please? No, I already have enough children.
Reciprocal Request

I should ask F to be my parent next time

Ok, F is accepting children
Experiments

- On SplitStream, part of FreePastry
- Stochastic model for node proximity
  - 500 nodes randomly distributed on a plane
- Each node subscribe to 16 trees
- Good nodes accept up to 16 children
Tree Reconstruction Cost

- Reconstruct one tree
- Unsubscribe one tree

*64 byte/msg, reconstruct 16 trees every 2 min, 128Kbps stream → 1.71% overhead*
Parental Availability (PA)

PA can be very low

Prob. the prospective parent becomes (in)direct parent
Debt Level

Cannot distinguish selfish nodes from normal nodes!
Confidence

Effectively distinguish selfish nodes
Enforcing Policy

- A simple policy using the above schemes
- Two types of selfish nodes
  - Refuse to accept children
  - Accept children but refuse to forward data
- Different start time to freeload
  - Begin cheating immediately
  - Start only after time 32
The Policy

- Not to use debt level
- Normal nodes will not serve those with
  - Confidence < -2
  - PA < 0.44 and confidence < 0.2
- Positive confidence/PA decay over time
- Allow preemption if 0.1 higher in PA
- Reciprocal requests if requests are 8 times more
Result from Enforcing Policy

- Normal nodes
- Refuse to forward data after 32
- Refuse to accept children after 32
- Always refuse to accept children
- Always refuse to forward data
More on Policy

- Increasing selfish nodes reduces the reception of normal nodes
  - 4% selfish nodes → 90% reception

- Can use encoding
  - Receive above a certain fraction of data to decode anything at all
  - Freeloaders get no service, would probably leave
Related Work

- Media streaming [Habib & Chuang, IWQoS’04]
  - Choosing peer to serve through scoring
- Focus on request-stream model
- Rely on trust system
Concluding Remarks

- Mechanism effective by tracking only first-hand observed behavior
- Low network and computation overhead
- Future work:
  - Robustness against more freeloaders
  - Learn parameters using Bayesian approach
  - Study dependence on multicast application, p2p substrate, and network topology