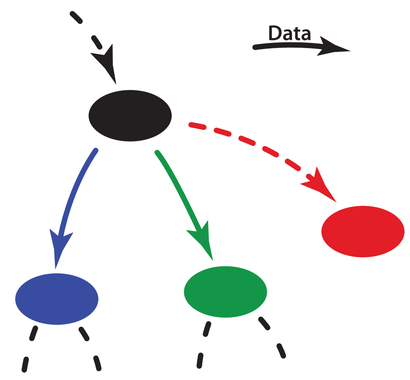
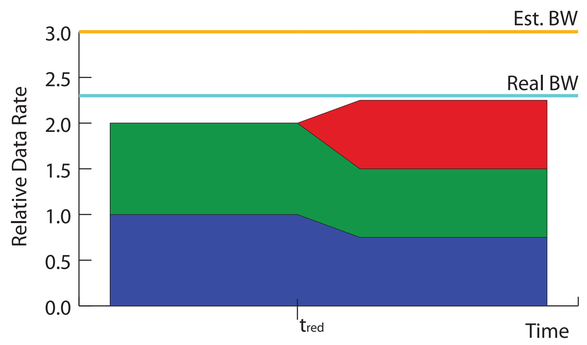
GateKeeper – Enabling Bandwidth Adaptation in Overlay Networks

Simon Schubert (EPFL) – Frank Uyeda (UCSD) – Dejan Kostić (EPFL) – Antony Rowstron (MSR)

Challenges in Streaming Overlays

- Host-based **overlays** are popular for **real-time media streaming**
- A significant number of hosts are behind **asymmetric links**
- Aggregate outbound bandwidth ≪ aggregate inbound bandwidth
- Users often under-/**overestimate** the available outbound bandwidth
- **Challenge:** manage the available bandwidth resources in the overlay

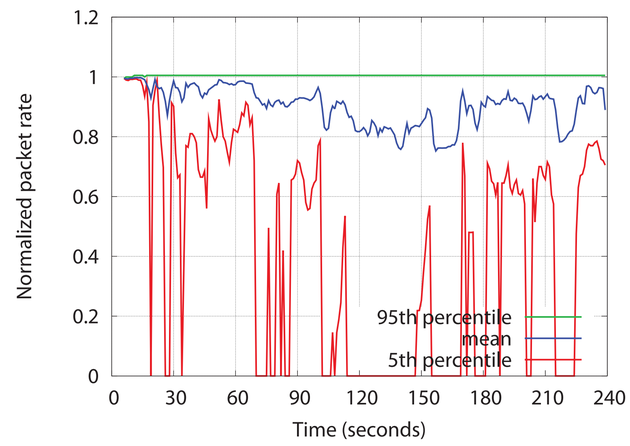




The **black** node successfully forwards data to its children, the **blue** and **green** nodes.
However, because **black** overestimates its outgoing bandwidth, it accepts **red** at t_{red}.
Not only **red** receives a reduced data rate, but **blue** and **green** suffer as well.
This in turn propagates to their children.

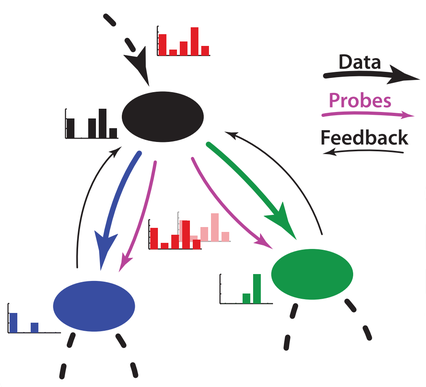
Experimental evaluation

Performance of a streaming overlay when a fraction of the nodes under- or overestimate their outbound bandwidth.
For a large fraction of nodes the receive rate is seriously impaired.



Bandwidth Adaptation with GateKeeper

- Each overlay node runs an instance of GateKeeper
- Overlay exposes information about peers and stream rate to GateKeeper
- GateKeeper maintains and exposes **local** and **global** bandwidth info to the overlay
- This enables the overlay to **adapt** to the underlying bandwidth availability.



GateKeeper

GateKeeper maintains an **estimate** of the consumed and **available outbound bandwidth** using a lightweight background **monitoring and probing process.**
Local bandwidth information is aggregated in the form of **histograms.**
These histograms are **convergecasted** up a control-channel tree and then **multicasted** to each node.

GateKeeper Benefits

- Not engineered into an overlay, but works in tandem
- Provides mechanism that can easily support adaptation policies
- Enables policies to be combined in different ways

Possible Policies enabled by GateKeeper

Dynamic stream-rate scaling
. . . accommodates as many users as possible.

Admission control
. . . prevents significant performance degradation for a large fraction of users.

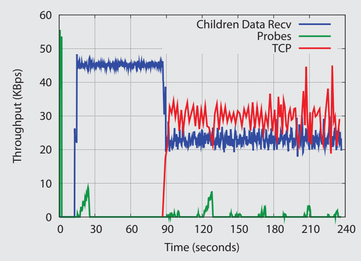
Dynamic contribution control
. . . saves cost & energy and can balance the contribution skew.

Experimental Evaluation

Local Probing and Interaction with TCP

- GateKeeper correctly estimates available outbound bandwidth
- Given the information from GateKeeper, the overlay decides to back off in the presence of TCP

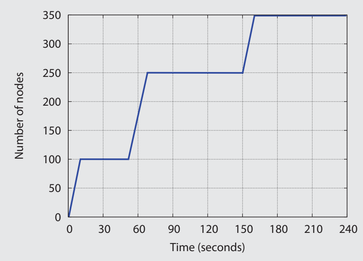
Timeline:
Time 0: Initial probing
Time 10: Node joins the overlay
2 children join
Additive probing attempted
Time 90: TCP flow begins
1 child is orphaned
Additive probing runs periodically



Experiment Setup

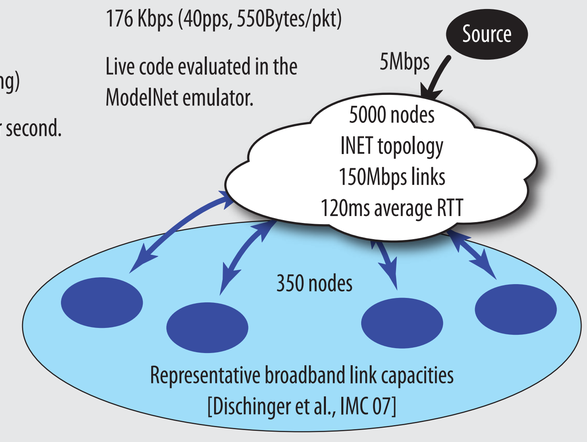
Timeline:
Time 0: 100 contributing clients
Time 50: 150 freeriding clients
Time 150: 100 clients (50 contributing, 50 freeriding)

For each flashcrowd, clients arrive at a rate of 5 per second.



Default Stream Rate:
176 Kbps (40pps, 550Bytes/pkt)

Live code evaluated in the ModelNet emulator.

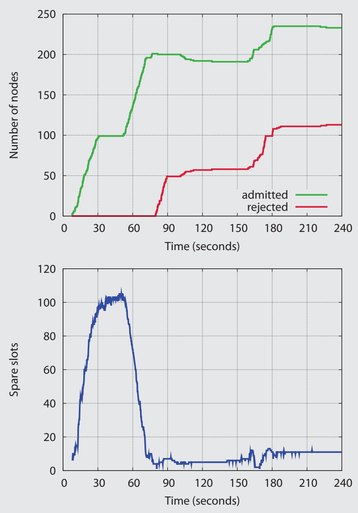


Admission Control

When outbound bandwidth is scarce, nodes use the histograms to explicitly reject new joins.

Configuration:
- Nodes receive global capacity histograms
- When < 5% available capacity, nodes explicitly reject joiners
- At t=100s, 10% of contributing nodes start a TCP upload

Results:
- Nodes rejected when capacity is low
- Excellent performance for admitted nodes

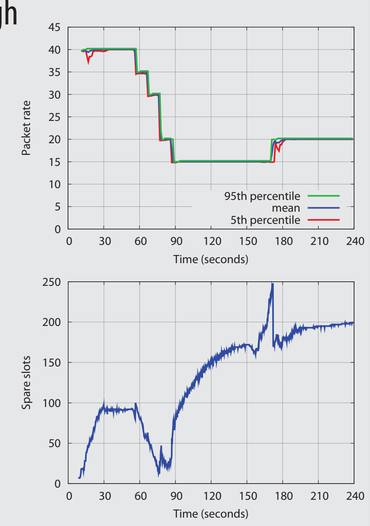


Dynamic Stream Rate Scaling

Using the histograms, the source dynamically scales the bitrate so that there are enough free slots for all nodes to join.

Configuration:
- The source tries to maintain 20% spare capacity
- The source can scale the stream rate in 5 packet steps

Results:
- Source scales down stream rate when capacity is low
- All nodes are able to join the overlay
- All nodes receive the stream



Dynamic Contribution Control

Per-node contribution depends on the spare capacity in the system and its own contribution relative to other nodes.

Configuration:
- 49 contributing nodes configured with 5Mbps in/out bandwidth
- Initial max. contribution level set to 2 children for all nodes
- When < 10% spare capacity, nodes probabilistically increase their max. contribution level if they can

Results:
- Reduces contribution skew
- All nodes are able to join the overlay
- All nodes successfully receive the stream