SAAR: A Shared Control Plane for Overlay Multicast

Animesh Nandi

Aditya Gangam, Peter Druschel, T.S. Eugene Ng, Ion Stoica, Hui Zhang, Bobby Bhattacharjee

(MPI for Software Systems, Rice University, CMU, UC Berkeley, University of Maryland)
Overlay Multicast

Endsystems cooperatively disseminate content
Applications

- Webcast special events
  e.g. ESM, TMesh ...

- P2P-Internet Television
  e.g. Coolstreaming, PPLive, Sopcast ...
Why another Overlay Multicast paper?

Architectural insight:
- separate overlay, optimized for control
- shared among many data overlays
- benefits single-tree, multi-tree, mesh
Data dissemination

Overlay structure optimized for efficient data dissemination
Control Mechanism

Employs control mechanisms to build/repair overlay structure
ESM’s Data/Control

Data overlay optimized for latency/bandwidth

Gossip to distribute membership info

Probing to select parents

Does not scale to large groups and high membership churn

Summary: efficient data plane structure but unscalable control
Idea: Decouple control/data into separate overlays

Dataplane optimized for data dissemination

Separate overlay provides efficient control

Summary: Using separate overlays avoids efficiency tradeoffs
Summary: Sharing reduces control overlay churn.
Summary: Neighbor acquisition is a fundamental control task.
**Anycast Primitive for Neighbor Acquisition**

**anycast** (groupId $g$, constraint $p$, objective function $o$)

(Amongst the members of $g$ satisfying $p$, chooses the one that maximizes $o$)

**Summary:** Anycast primitive used to build different overlays
Question: Can the control overlay efficiently implement this anycast primitive?
Idea: Use structured overlay

Data planes optimized for efficient data dissemination

Shared structured overlay provides efficient anycast service

Pastry key space

Animesh Nandi
SAAR: Per-channel control tree

Control tree’s structure is optimized for control and independent of data overlay structure.
Control tree formation

Tree structure formed efficiently

root (GID)

channel member

member of other channel
SAAR: Aggregation

Control plane view

State variable \( \text{cap} := \# \text{sparse capacity slots} \)

- \( \sum_{\text{cap}} = 0 \)
- \( \sum_{\text{cap}} = 3 \)
- \( \sum_{\text{cap}} = 2 \)
- \( \sum_{\text{cap}} = 5 \)

- channel member
- member of other channel

Animesh Nandi
SAAR: Aggregation

State variable \( \text{dep} := \) advertised tree depths

Control plane view

- channel member
- member of other channel
SAAR: Efficient search

Aggregation enables efficient Depth-First-Search of control tree
Evaluation

- Modelnet
- 350 nodes
- heterogeneous forwarding bandwidths
- mean stay time of 5 minutes (exponential distribution)
- Metrics: channel switching delay, streaming quality, control overhead
SAAR enables low join delays
SAAR enables low join delays

Initial single-tree join when not in SAAR control overlay
SAAR enables low join delays

- **Multitree**
- **Singletree**
- **ESM**

Initial single-tree join when not in SAAR control overlay
SAAR enables low join delays

Initial single-tree join when not in SAAR control overlay
SAAR: good streaming quality
SAAR: low control overhead

![Graph showing cumulative fraction against control overhead for Singletree, Mesh, Multitree, and ESM.](image)
SAAR supports complex overlays

Nodes start leaving/rejoining
All nodes have joined data overlay

Multi-tree, 5 stripes, 350 nodes, 2-min mean churn (exp.), RI=1.23
More results in the paper

- Flash crowd conditions
- Higher churn (2 min mean, exp. dist)
- Control churn
- Planetlab experiments
Conclusions

- Control efficiency is key to performance of overlay multicast systems, especially tree-based systems.

- Anycast based on structured overlays enables powerful, efficient control for tree-based systems.

- Fast channel switching can be achieved with tree-based overlay multicast systems.
Planetlab: Join delays

(a) Join delay (ms)
Planetlab: Streaming Quality

Cumulative Fraction

(b) Continuity Index

Native-ESM
SAAR-ESM

Animesh Nandi
Flash crowd: Join Delays

![Graph showing cumulative fraction against join delay (ms)]
Flash crowd: Tree Depths

(b) Tree Depth

Cumulative Fraction

SAAR-ESM
Native-ESM
eg. Scribe’s Control/Data

Scribe has efficient control but inefficient data paths.

Pastry’s Key space

Prefix-based Routing path

Low bandwidth

Subscribers

Match prefix with GId
Summary: SAAR provides uninterrupted control in the face of control overlay churn.

Animesh Nandi
Enabling multi-tree data plane

- Attempts to build $k$ interior-node disjoint stripe trees.
- Node anycasts to establish parents in each stripe, but joins only one anycast group corresponding to its primary stripe.
- Resource based primary stripe selection: `groupAggregateRequest (groupId, state-variable)`
- Same predicate and objective function as in single-tree for each stripe tree anycasts
- **Summary**: multi-tree policy expressed easily.
Enabling block-based data plane

Node anycasts for indegree-neighbors and employs swarming

Group’s state variables – $g_{capacity}$, $g_{load}$, $g_{loss}$, $g_{buffermap}$, $g_{coordinate}$,

Predicate: $(load < capacity) \& \& (loss < threshold)$  // not overloaded, low loss

Objective function:

$$\text{cardinality}(\text{missingset} \cap g_{buffermap})$$  // recover missing blocks
$$1/g_{loss}$$  // establish a good mesh neighbor

Summary: block-based policy expressed easily
Graphs from paper
Planetlab: Join delays

(a) Join delay (ms)
Planetlab: Streaming Quality

![Graph showing cumulative fractions against continuity index](image)
Multi-tree: Resource Balance

Nodes start leaving/rejoining
All nodes have joined data overlay

Forwarding Resources

Time (sec)

Stripes Resource Max
Stripes Resource Min
Group Size

Animesh Nandi
Multi-tree: #stripes recvd

Nodes start leaving/rejoining
All nodes have joined data overlay
Locality-awareness

![Graph showing cumulative fraction against proximity of prospective peer in milliseconds.](image)

[Animesh Nandi](#)
Load-awareness

Cumulative Fraction

Anycast Response Time (ms)

CENTRAL-NPS
SAAR RI=1.01
SAAR RI=1.23

Animesh Nandi
Depth-Optimization

![Graph showing cumulative percentage vs tree depth for CENTRAL-NPS, SAAR, and SAAR-NO-DepthOptimization]
Single-tree: Join Delays

![Graph showing cumulative fraction over time for different systems.](image-url)

- **SAAR-ESM**
- **SAAR-ESM-Unshared**
- **Native-ESM**
- **Scribe**

Animesh Nandi
Single-tree: Continuity
Single-tree: tree-depths
Single-tree: Nodestress

(d) Node Stress (msgs/node/sec, control only)
Multi-tree: Join delays

(c) Join Delay (ms)
Multi-tree: Streaming Quality

(d) Continuity Index
Mesh-based: Streaming Quality

(a) Continuity Index
Mesh-based: Path lengths

Cumulative Fraction

(b) Block Dissemination Hops