

Cross-Tree Adjustment for Spatialized Audio Streaming over Networked Virtual Environment

Ke Liang, Roger Zimmermann

Dept. of Computer Science
School of Computing
National University of Singapore

June 5, 2009

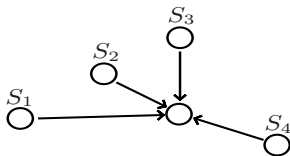
Introduction

Why provide spatialized audio service over NVEs?

- Interactive communication - Typing is boring
- Immersive experience - What you hear is what you see



Snapshot in Second life



Receiving 4 streams
rendered at different directions

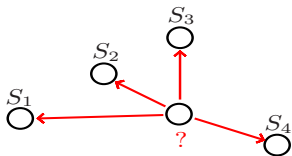
Introduction

Why provide spatialized audio service over NVEs?

- Interactive communication - Typing is boring
- Immersive experience - What you hear is what you see



Snapshot in Second life



Can it send its audio
to 4 receivers simultaneously?

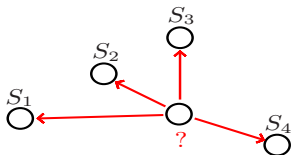
Introduction

Why provide spatialized audio service over NVEs?

- Interactive communication - Typing is boring
- Immersive experience - What you hear is what you see



Snapshot in Second life



How to build efficient dissemination topology?

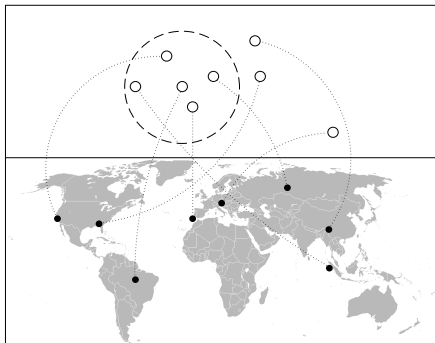
Motivation

Quickly deploy multicast trees in the overlay network for all speakers with **high reachability** and **low latency** respecting to upload bandwidth limits

Outline

- Assumptions and definitions
- Problem formulation
- Procedure of the proposed approach
- Results

Assumptions and definitions



Definitions:

- Area of interest (AoI): circular area centered at the user' position
- Degree: number of audio streams a node can send/forward

Assumptions:

- Any node can speak anytime, anywhere
- Nodes' capacity (degree) is heterogeneous and usually insufficient
- Every node periodically exchanges information with its neighbors
- Audio dissemination is restricted to be within the AoI of the speaker

Problem formulation

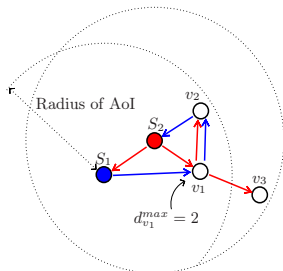
Forest of degree bounded shortest path trees problem (NP-complete)

Objectives:

- Maximize number of total receivers
- Minimize average latency

Subject to:

- Arbitrary degree bound at each node (user)



Heuristic solution:

Achieve a compromise between the two objectives via maximizing the number of receivers that have the minimum latency

Challenge:

How to allocate upload bandwidth of nodes which residing in multiple multicast trees?

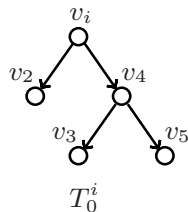
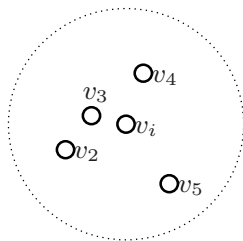
Algorithm - Multicast tree construction and adjustment

Multicast tree construction

- Construct a temporal **shortest path tree** (SPT) T_0^i for the new speaker i without considering its neighbor nodes' degree bounds
- Apply **post-order tree traversal** to 1) compute the weight^a of each edge, 2) locate *conflict node*^b and 3) partition all nodes into sets from bottom to top

^aNo. of descendent nodes

^bUpload bandwidth required for the node exceeds its maximum capacity



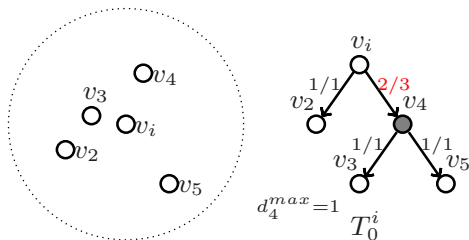
Algorithm - Multicast tree construction and adjustment

Multicast tree construction

- Construct a temporal **shortest path tree** (SPT) T_0^i for the speaker i without considering its neighbor nodes' degree bounds
- Apply **post-order tree traversal** to 1) compute the weight^a of each edge, 2) locate *conflict node*^b and 3) partition all nodes into sets from bottom to top

^aNo. of descendent nodes

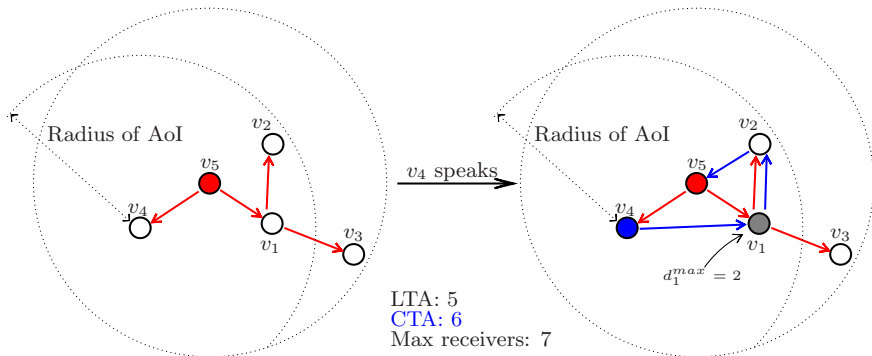
^bUpload bandwidth required for the node exceeds its maximum capacity



Algorithm - Multicast tree construction and adjustment

Multicast tree adjustment

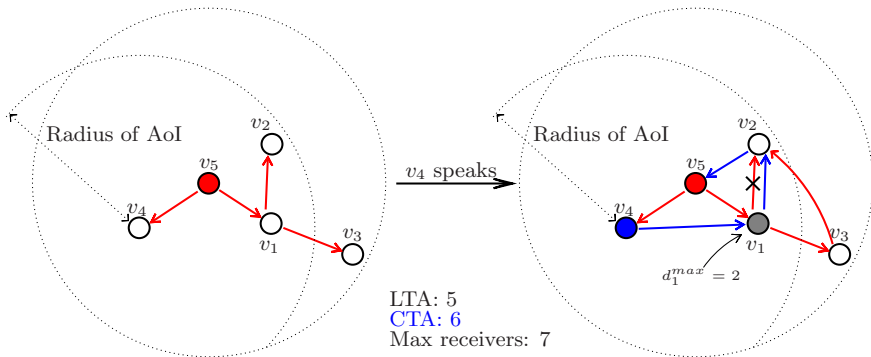
- Prune edges within the local multicast tree (LTA) or cross multiple trees (CTA)
- Attach orphan nodes (brought by edge pruning) to the pruned SPTs



Algorithm - Multicast tree construction and adjustment

Multicast tree adjustment

- Prune edges within the local multicast tree (LTA) or cross multiple trees (CTA)
- Attach orphan nodes (brought by edge pruning) to the pruned SPTs



Edge pruning at conflict nodes

LTA: pruning edges in local multicast tree

- Compute edge weight from **bottom to top** respecting to **available degree**
- Prune edges in the local multicast tree from **top to bottom** respecting to their weight

CTA: pruning edges cross multiple multicast trees

- Compute and **recompute** edge weight from bottom to top respecting to **degree bound**
- Prune edges in multiple trees from top to bottom respecting to their weight

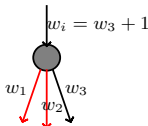
Why compute edge weight from bottom to top?

Edge pruning in SPT with CTA - edge weight computing

Data structures for edges:

- Global weight set W_i^g : the set of weight of edges (in existing multicast trees) starting from node i . $\max(|W_i^g|) = d_i^{max}$.
- Local weight set W_i^l : the set of weight of edges (in new multicast tree) starting from node i . For conflict node i , $|W_i^g| + |W_i^l| > d_i^{max}$
- Descendants set D_i^j : the set of descendants of node i in SPT T^j

$$W_i^g = (w_1, w_2), W_i^l = (w_3) \\ w_1 < w_3 < w_2$$



Trivial case

When the existing pathes^a starting from conflict node i don't have other conflict nodes

^aPathes in other mulicast trees

Put the descendant nodes of edge with smallest weight to ready-to-remove set (a set of tuples defined as (i, D_i^{red})).

Edge pruning in SPT with CTA - edge weight computing

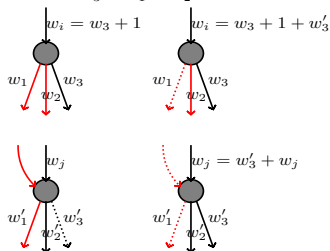
Case 1

When the existing paths starting from conflict node i have another conflict node j which **has been processed** and is still **in the local SPT**

$$W_i^g = (w_1, w_2), W_i^l = (w_3)$$

$$w_1 < w_3 < w_2$$

$$w'_3 < w'_1 < w'_2$$



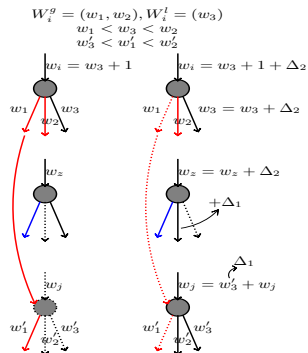
After processing:

- (Local tree): Local weight set of nodes in the path $i \rightarrow j$ is updated
- (Local tree): Descendant node set of nodes in the path $i \rightarrow j$ is updated
- (Other trees): Descendant node set of node i is **supposed** to be updated (put (i, D_i^{red}) into ready-to-remove set).

Edge pruning in SPT with CTA - edge weight computing

Case 2

When the existing paths starting from conflict node i have another conflict node j which **has been processed** but is **NOT** in the local SPT



After processing:

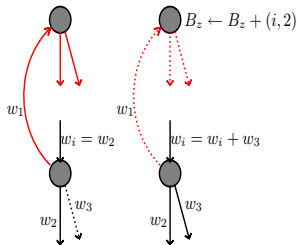
- (Local tree): Local weight set of nodes in the path $i \rightarrow j$ is updated
- (Local tree): Descendant node set of nodes in the path $i \rightarrow j$ is updated
- (Other trees): Descendant node set of node i is **supposed** to be updated (put (i, D_i^{red}) into ready-to-remove set).

Edge pruning in SPT with CTA - edge weight computing

Case 3

When the existing paths starting from conflict node i have another conflict node z which has **NOT** been processed

$$W_i^g = (w_1), W_i^l = (w_2, w_3) \\ w_1 < w_3 < w_2$$



After processing:

- (Local tree): Local weight set of nodes in the path $i \rightarrow j$ is updated
- (Local tree): Insert a tuple $(i, 2)$ to **backup list** of node z B_z
- (Other trees): Descendant node set of node i is **supposed** to be updated (put (i, D_i^{red}) into ready-to-remove set).

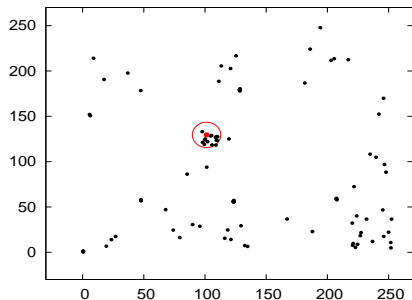
Edge pruning in SPT with CTA - edge pruning

- Adjust other existing multicast trees according to the ready-to-remove set and the adjusted new SPT
- Attach orphan nodes to those SPT members which have additional degrees.
- Merge the local weight set into global weight set

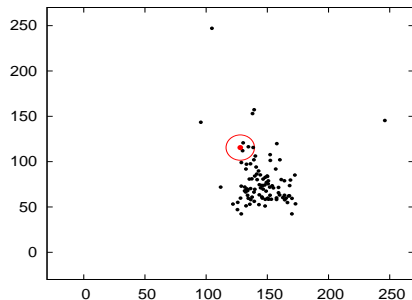
Simulation setup

Parameters input

- Latency data from Planetlab
- Bandwidth data from dsreport.com
- 25% of upload bandwidth is used to send/forward audio streams
- Avatar mobility data collect from Second life

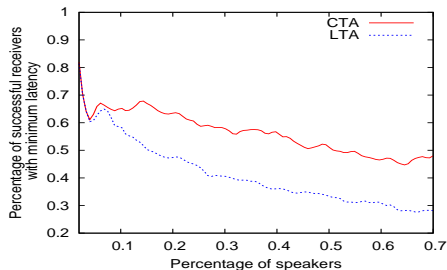
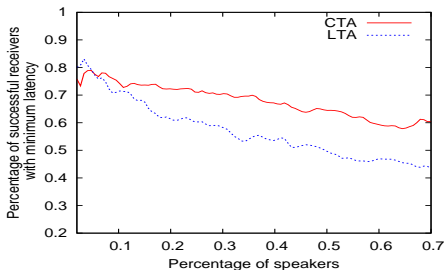
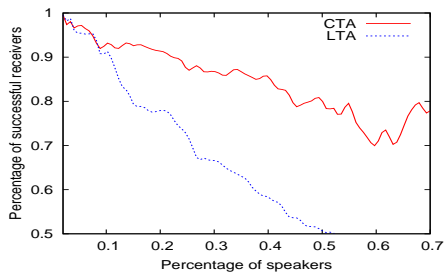
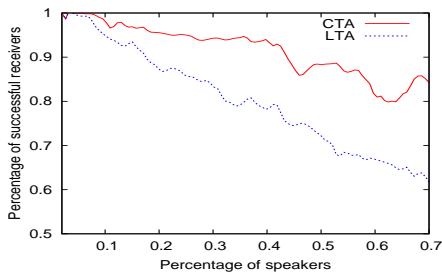


Freebies land



Pharm land

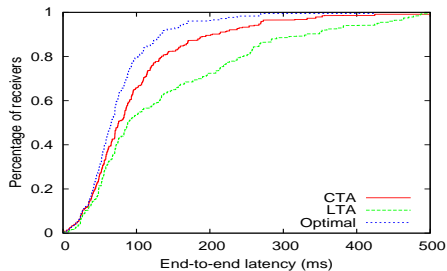
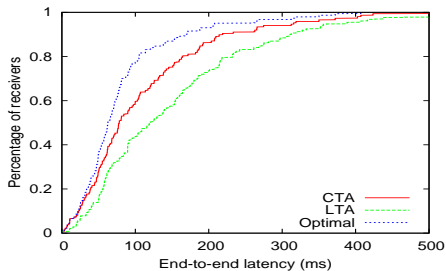
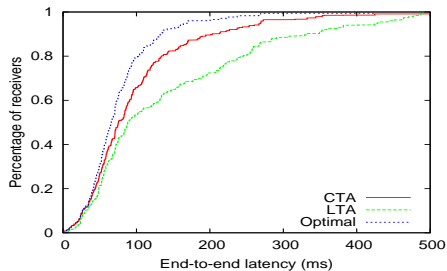
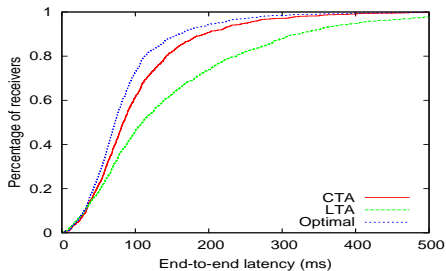
Simulation results - Number of receivers



Freebies land

Pharm land

Simulation results - Latency



Freebies land

Pharm land

Thanks!