A Study of Distributed Low Latency Application Layer Multicast Tree Construction

Su-Wei Tan, Gill Waters, John Crawford
Computing Laboratory, University of Kent

Abstract

We are interested in constructing low latency data delivery trees for application layer multicast (ALM). A salient feature of ALM is that it uses an overlay topology built on top of existing unicast services to bypass the need for the network layer multicast support. However, ALM protocol hosts are normally end systems that have constrained access bandwidth, processing capability and topological knowledge. These limitations suggest that the overlay structure used for data delivery must be degree-bounded, and need to be constructed in a decentralised manner using limited topology information. In this paper, we report a comparative study of a wide range of existing distributed ALM overlay construction techniques that adhere to the above requirements.

1 Introduction

Over the past few years, application layer multcast (ALM) has received much attention. This is mainly because it offers an alternative to realising group communication applications over the global Internet, which has been hindered by the sporadic deployment of IP multicast infrastructure. In ALM, all multicast functionalities such as packet replication and membership management are implemented directly at end systems like desktop/laptop computers, proxy servers, etc. These end systems are organised into an overlay topology for multicasting. The edges in the overlay are unicast tunnels between the end systems. Hence, the key to the efficiency of an ALM solution lies in the overlay structure used.

We are interested in the problem of creating low latency degree-bounded data distribution trees for point-to-multipoint applications. The degree constraint is due to the limited available bandwidth and processing capability at the end systems. This type of tree is particularly useful for real-time applications such as live webcasting and critical event notification. Specifically, we look at the following optimisation problem:

Minimum maximum-latency degree-bounded spanning tree problem: Given a data source, s and a set of receivers, and the degree constraint for each of these nodes, construct a degree-bounded overlay spanning tree rooted at s such that the maximum delay from s to the receivers is minimised.

A key issue in creating an efficient overlay structure is the end systems' lack of knowledge of the underlying topology and its performance metrics, e.g. delay and bandwidth. To obtain such information, an ALM protocol often uses active end-to-end measurement techniques. As such a technique consumes substantial network bandwidth, it is impractical to gather the metrics for all node-pairs — for an n-node system, $O(n^2)$ measurements will be needed. This indicates that a practical solution needs to work with limited topological knowledge. In addition, the solution must limit the interaction between the members so as to scale to large group sizes. Unfortunately, the above problem is NP-hard [1] even under a centralised computational model.

We believe that a practical solution for the tree construction problem needs to be fully distributed. The aim of this work is to understand the properties of existing solutions, which serves as a basis for the design of a new protocol. In this paper, we compare the quality of the overlay trees generated by several existing ALM protocols. The chosen protocols encompass a wide spectrum of overlay construction techniques.

2 Selected Distributed ALM Trees Construction Protocols

In general, a distributed overlay construction protocol consists of two phases: i.) *joining phase*; and ii.) *maintenance phase*. The joining phase deals with the creation of a new group and bootstrapping of new members into a running session, i.e. attaching to the existing overlay. Once attached to the overlay, members participate in maintenance functions such as overlay connectivity maintenance, overlay optimisation, and membership management. The overlay optimisation process normally involves periodical reconfiguration of the overlay structure. A reconfiguration operation (e.g. add/delete an overlay link) is done if it will improve the quality of the overlay.

In the following discussion, we will focus only on the overlay construction and optimisation techniques; other aspects of the protocols (e.g. loop detection/avoidance/etc.) are beyond the scope of this paper.

The rest of this section briefly reviews the chosen protocols. We select representative protocols that are designed to be scalable and are able to produce degree-constrained trees. Some of these are modified slightly to give known improvements.

HMTP [2]: HMTP is designed to create low cost trees, where the cost of a tree is defined as the sum of delays on the tree links. To construct a low cost tree, each node (except the root) periodically attempts to look for and switch to a new parent that is nearer to itself than its existing parent. HMTP uses a depth-first search technique to explore a branch of the tree at a time when looking for a potential parent.

Switch-trees [3]: Switch-trees defines a set of scope limited switch-parent operations to reconfigure the overlay trees. A switch operation is performed when a node finds that the switch target (i.e. potential parent) provides better performance (e.g. lower delay from the tree root) than its current parent. The proposed scopes include nodes that are h hops away from a node that tries to perform a switch. In this paper, we consider the case where $h = 2$, which is a practical and yet efficient solution compared to other cases proposed in the paper. In [4], the authors proposed HostCast for creating low latency overlay trees. HostCast also adopts the switch-parent operation for nodes within a local region. The local region for a node x defines its potential parents, which include x's grandparent and uncles. In addition, HostCast allows a parent-child pair to swap position to achieve better performance. In our 2-hop switch-trees implementation, we include the swap operation. As the 2-hop scope is the superset of HostCast's local region, we expect that our version of switch-trees to perform better than HostCast.

AOM [5]: AOM can be viewed as a hybrid of the above-mentioned protocols: it provides a set of conditions that combine the efforts to reduce the tree cost and delay from the tree root. The conditions are weighted by two parameters (i.e. α and p, where $0 < \alpha, p < 1$): these two values can be tuned to choose a new parent that is sufficiently close and also provides low delay to the tree root. AOM uses a local region that is similar to HostCast. Its delay properties have been shown to out-perform those of HMTP [5].

Banerjee et al.'s scheme [6]: This scheme was originally designed to create minimum-average latency trees for a proxy-based architecture. Extension were suggested for building minimum maximum-latency trees. It defines a set of local transformation operations (e.g. switch-parent, swapping) for nodes within two levels of each another. Besides the information about the delay from the tree root (as in switch-trees, HostCast and AOM), a node also keeps the maximum delay for the subtree rooted at itself. A transformation is made if it improves the objective function and does not increase the subtree delay.

TBCP [7]: TBCP is a generic tree building protocol. It uses a localised central arrangement scheme to place a node into the delivery tree. Basically, when a node x tries to join to a node y, x measures the distance from itself to y and y's children. Based on this information, and the distance information from previous rounds, y knows the complete distance matrix for itself, its children and x. It then selects the best configuration to place x onto the subtree rooted at itself, where the goodness of a configuration is judged by a score function. In this paper, we use a score function proposed in [8], which has been shown to out-perform the original function. In particular, the function tries to minimise the maximum stretch in the configuration.

Scribe [9]: Scribe constructs ALM trees on top of overlays built with Pastry [10], an efficient peer-to-peer object location protocl. Basically, Scribe utilises Pastry's good routing properties to attach newcomers to nodes that are topologically close to them. Instead of implementing the whole Pastry-Sribe protocol set, we simulate Scribe trees in the following manner. When a newcomer, say x, joins a session, we attach x to the nearest on-tree node, say y. If y found that adding x violates its degree bound, y executes the "bottleneck remover algorithm" as suggested in Scribe. Specifically, y drops its farthest child, which will be redirected to one of y's remaining children. Once attached to the tree, a node periodically uses the above procedures to improve its on-tree position.

3 Performance Evaluation and Discussion

We use a custom written event-driven simulator for the experiments. The experiments run on topologies generated by two well-known models: i.) *transit-stub*; and ii.) *power-law*. We report representative results from a 10100-node transit-stub network, and a 5000-node power-law network. For all experiments, each end system is randomly attached to one of the routers. Each of them is assigned a maximum out-degree which is uniformly distributed between 2 and 10. For each simulation scenario, we conduct 50

independent runs and report the average. Fig. 1 and Fig. 2 illustrate the quality of the delivery trees built with different techniques, in terms of four widely used metrics, observed under the transit-stub network and the power-law network respectively.

Tree Cost Ratio (TCR) [2]: Tree cost is calculated as the sum of delays on the overlay tree's links. It provides a simplified view of the total network resource consumption of a tree. We calculate TCR as the ratio between the costs of the overlay tree and the network layer shortest path tree. From Fig. 1 (a), we can see that Scribe and HMTP have the smallest TCR, which is then followed by AOM, switch-trees, TBCP and Baneerjee et al.'s scheme. The low TCR observed for Scribe and HMTP is because they try to place nodes that are topologically close together, and thus produce low cost tree. Scribe shows smaller TCR than HMTP as our implemented version exploits the knowledge of the underlying topology. Other protocols try to minimise the delay properties, which often result in a tree that has a *compact* structure. In other words, longer links may be added into the tree, and result in high tree cost. We can also observe a rather similar trend for results obtained from the power-law network (see Fig. 2 (a)). One major difference is that the TCR values are relatively smaller compared to the transit-stub network.

RMP: RMP is defined as the ratio between the maximum latency from the root, s to other nodes using the overlay tree and the maximum latency from s to other nodes using direct unicast connections. Therefore, it represents the objective of the low delay degree-constrained tree problem mentioned in Section 1. Fig. 1 (b) shows that Scribe and HMTP perform worse than the other protocols. This is because they produce low cost tree that often have long overlay paths between the nodes, and consequently results in a taller tree. Between these two protocols, Scribe which shows better TCR has higher RMP than HMTP.

For other protocols that specifically try to minimise the root delay, we found that Banerjee et al.'s scheme performs the best, which is followed by switch-2hop, TBCP and AOM. As Banerjee et al.'s scheme utilises more flexible tree reconfiguration technique, and uses additional information (i.e. subtree delay), its better performance is expected. The results for the power-law network (see Fig. 2 (b)) show a similar trend, except that the RMP for trees generated by AOM increases rapidly with the group size, which eventually show higher RMP than HMTP and Scribe. We note that the performance of AOM can be tuned with two configurable parameters, and we chose to use the best setting suggested in [5], i.e. $\alpha = 0.9$ and $p = 0.2$. However, the above observations suggest that the optimal setting of the parameters is very sensitive to the underlying topologies used. This is an undesirable property as the best model for the Internet is still an open question.

RAP: RAP is defined as the ratio between the average latency from s to other nodes using the overlay tree and the average latency from s to other nodes using direct unicast connections. From Fig. 1 (c), we can see that the comparison among the protocols are quite similar to that for RMP, for the same network model. One observable difference is that TBCP provides the lowest RAP. From Fig. 2 (c), we can observe that TBCP again provides the lowest RAP.

Link Stress: Link stress is defined as the number of duplicated copies of identical packet flows over a single link. Hence, it represents the redundant traffic injected in to the network by an ALM solution. From Fig. 1 (d) and Fig. 2 (d), we can observe that the performance trend roughly follows those of the TCR. In other words, protocols that produces low cost trees also have lower traffic redundancy.

Comparing the maximum stress values observed for both network models, we can see that the transit-stub network always result in higher stress (as well as TCR). In a transit-stub network, each stub domain attaches to a transit domain via limited stub-transit links, which results in traffic concentration on these links. On the other hand, the power-law model creates a flat topology, which allows traffic to be distributed more evenly to all links.

Summary: The above results show that Banerjee et al.'s scheme can produce trees with the smallest radius, i.e. the maximum delay from the tree root to the other nodes. However, as low delay trees tend to be more compact in structure, this results in high tree cost (i.e. resource usage) and link stress. As the compact structure often causes nodes that are topologically close to be placed farther apart, the *average* delay to all nodes may suffer. For example, this can be viewed from the better RAP performance of TBCP, which tries to minimise the stretch between the nodes. On the other hand, protocols that produces low tree cost (i.e. Scribe and HMTP) show the worst delay performance. Overall, this proves that there is a trade-off between delay and tree cost (and link stress), i.e. minimising tree cost results in high end-to-end delay; minimising delay results in high tree cost and link stress. Between these two extremes, there is a wide spectrum of other trees that provide varying cost and delay values. By using two different network models, we also observe some topological effects on the performance of the protocols.

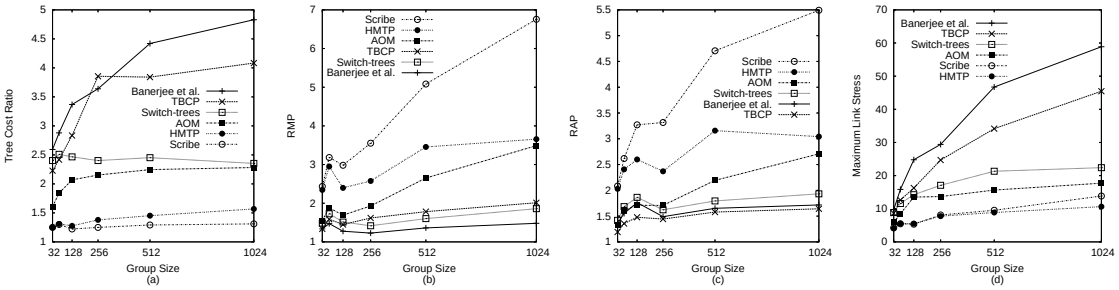

Figure 1: Results for Transit-stub Network: (a) TCR, (b) RMP, (c) RAP and (d) Max. Link Stress

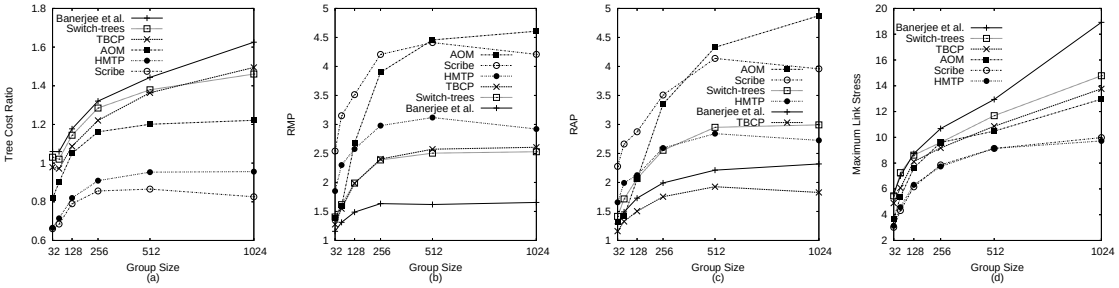

Figure 2: Results for Power-law Network: (a) TCR, (b) RMP, (c) RAP and (d) Max. Link Stress

4 Concluding Remarks

We are interested in building low latency degree-bounded application layer multicast trees for single source real-time applications. This work is carried out as the first step towards enhancing/designing protocols for such applications. In this paper, we report a simulation comparison for the following existing overlay tree construction protocols: a scheme by Banerjee et al., HMTP, Scribe, TBCP, switch-trees, HostCast and AOM. The results show that Banerjee et al's scheme can produce trees with the smallest worst-case delay from the tree root to the other nodes. However, this sometimes results in poorer average delay to the nodes. In addition, such low delay trees often have high traffic redundancy and network resource usage. On the other hand, protocols like Scribe and HMTP try to connect nodes that are topologically close together, and yield low traffic redundancy and resource usage. However, the resultant trees often have high end-to-end delay.

Based on the above observations, we have designed a new scheme called MeshTree [11], which considers both the delay and the closeness of the overlay nodes. The resultant trees show good delay, traffic redundancy and resource usage properties, compared to Baneerjee et al.'s scheme.

References

[1] S. Y. Shi, J. S. Turner, and M. Waldvogel. Dimensioning server access bandwidth and multicast routing in overlay networks. In *NOSSDAV*, Port Jefferson, New York, USA, 2001. ACM.

[2] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *IEEE INFOCOM*, pages 1366–1375, New York, USA, 2002.

[3] D. A. Helder and S. Jamin. End-host multicast communication using switch-trees protocols. In *Workshop on Global and Peer to Peer Computing on Large Scale Distributed System (GP2PC)*, 2002.

[4] Z. Li and P. Mohapatra. Hostcast: A new overlay multicast protocol. In *IEEE ICC*, Anchorage, Alaska, USA, 2003. IEEE.

[5] Shuju Wu, Sujata Banerjee, Xiaobing Hou, and R. A. Thompson. Active delay and loss adaptation in overlay multicast tree. In *IEEE ICC*, Paris, France, 2004.

[6] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *IEEE INFOCOM*, San Francisco, USA, 2003.

[7] L. Mathy, R. Canonico, and D Hutchison. An overlay tree building control protocol. In *Networked Group Communication*, pages 78–87, London, UK, 2001.

[8] S. W. Tan and Gill Waters. Building low delay application layer multicast trees. In *4th Annual PostGraduate Symposium (PgNet)*, Liverpool, UK, 2003.

[9] M. Casto, P. Druschel, A. M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralised application-level multicast infrastructure. *IEEE JSAC*, 20(8), 2002.

[10] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms*, 2001.

[11] S. W. Tan, Gill Waters, and J. Crawford. MeshTree: A delay-optimised overlay multicast tree building protocol. Submitted conference paper under submission, Computing Laboratory, University of Kent, 2004.