Efficient P2P Data Dissemination in a Homogeneous Capacity Network Using Structured Mesh

Thinh Nguyen
thinhq@eecs.oregonstate.edu
Oregon State University
EECS Department
Corvallis, OR 97330

Duc Tran
duc.tran@notes.udayton.edu
University of Dayton
Computer Science Department
Dayton, OH 45469

Sen-ching Cheung
cheung@engr.uky.edu
University of Kentucky
EECS Department
Lexington, KY 40506

Invited Paper

Abstract

Efficient data dissemination from a single source node to multiple receiving nodes on the Internet is crucial for many applications such as P2P streaming. Existing data dissemination schemes are typically accomplished using the overlay multicast trees. These overlay multicast trees, however, do not achieve the full bandwidth capacity since the leaf nodes do not contribute their bandwidth to the system. On the other hand, all the nodes in a properly constructed topology can contribute their bandwidth, resulting in high overall system throughput. In this paper, we define the notion of throughput efficiency to measure the performance of different data dissemination schemes from a single source node to multiple destination nodes. Using the proposed throughput efficiency, we propose an algorithm for constructing an overlay structured forest that enables high-bandwidth data dissemination in a network with roughly homogeneous capacity. The proposed structured forest and the associated data dissemination algorithm are designed to achieve the following: (1) end-to-end delay from the source node to any node is small ($O((logN)^2)$), (2) the out-degree of any node is small ($O(C)$), and (3) bandwidth usages of all the nodes are optimal.

1 Introduction

Many Internet applications such as video multicasts rely on the network topology and protocols for efficient data dissemination from a single source node to a large number of destination nodes. A well-known example of data dissemination on the Internet is the IP multicast [1]. The primary motivations of IP multicast are (a) to avoid wasted bandwidth incurred in point-to-point data transfer and (b) to scale with the number of receivers. IP multicast, however is not widely deployed due to the compatibility issues among the autonomous systems (AS) in the Internet. This has led to a number of overlay multicast systems [2] where end hosts themselves form a multicast tree for delivering data. The advantage of overlay multicast is that physical routers do not need to support complex multicast operations. Instead, packet routing and forwarding are logically done at the application layer, which lead to easy deployment across different AS(es). However, the overlay multicast techniques are sup-optimal since identical packets may travel on the same physical links. In addition, overlay multicast is not optimal in term of throughput since the leaf nodes do not contribute their bandwidth to the system.

Traditional data dissemination algorithms place bandwith constraints on individual links. This model does not reflect modern network such as P2P or wireless network, where there are limited upload bandwidth but with little restriction on the download bandwidth. In this paper, we study the data dissemination problem in which, bandwidth constraint is associated with the node's upload bandwidth and no constraint on the node's download bandwidth. For example, in a lightly loaded P2P network [3][4], the bandwidth constraint of a DSL subscriber is its upload physical bandwidth, e.g. 250 kbps. A peer may decide to have multiple connections to other peers, but the sum of all sending rates cannot exceed the upload physical bandwidth. That said, we would like to construct a network topology and the associated data dissemination algorithm to result in (1) end-to-end delay from the source node to any node is small ($O(log(N))^2$), (2) the out-degree of any node is small ($O(C)$), and (3) bandwidth usages of all the nodes are optimal.

The rest of the paper is organized as follows. in Section 2, we define the notion of throughput efficiency. In Section 3, we will show how to construct structured mesh that maximizes throughput efficiency, and at the same time, has small delay and out-degree. In Section 4, we show simulation results for a number of scenarios, demonstrating the advantages of our proposed structured mesh. In Section 5, we list a few representative work similar to ours. Finally, we summarize our contributions and suggest future direction in Section 6.

2 Throughput Efficiency

To measure the performance of different data dissemination schemes, we define the following throughput efficiency:

Definition 1: Throughput efficiency is defined as

$$E \triangleq \frac{\sum_{i=0}^{i=N} S_i}{min(\sum_{i=0}^{i=N} C_i, NC_0)} \tag{1}$$

where 0 denotes the source node, $i = 1...N$ denote N destination nodes, S_i and C_i are the actual, useful sending rate and the sending capacity of node i, respectively.

The *useful* sending rate S_i means that all the data sent directly from node i to node j are completely disjoint with the data received at node j from all other nodes $k \neq i$. Hence, the actual sending rate S_i is dictated by the topology, the algorithms for data dissemination, and the rate partition at each node as exhibited in the previous ad-hoc algorithms. The numerator in the definition 1 is the total actual sending rate of all the nodes, while the denominator is the minimum of the two quantities: (a) total maximum sending capacity of all the nodes and (b) the maximum receiving capacity.

Proposition 1: $E \leq 1$ for any data dissemination topology and algorithm.

Proof:

Case 1: Assume $\min(\sum_{i=0}^{i=N} C_i, NC_0) = \frac{\sum_{i=0}^{i=N}}{\sum_{i=0}^{i=N} C_i}$, then since $C_i \geq S_i$, we have $E = \frac{\sum_{i=0}^{i=N} S_i}{\sum_{i=0}^{i=N} C_i} \leq 1$.

Case 2: Assume $\min(\sum_{i=0}^{i=N} C_i, NC_0) = NC_0$, then $E = \frac{\sum_{i=0}^{i=N} S_i}{NC_0}$. Now, we observe the following. A destination node cannot receive the information at a rate faster than the information rate being injected into the network. Since the source node injects the maximum data rate of C_0 into the topology, maximum total receiving rate of useful data for all N destination nodes is NC_0 bps. Since the total sending rate $E = \sum_{i=0}^{i=N} S_i$ and the total receiving rate must equal to each other, and therefore is less than or equal to the maximum total receiving rate of all the nodes NC_0. Hence, $E = \frac{\sum_{i=0}^{i=N} S_i}{NC_0} \leq 1$. $\square$

Clearly, throughput efficiency relates directly to the average receiving throughput of all the nodes. Efficiency of 1 means all nodes are sending data at their capacities, and therefore results in highest efficiency. Since the total sending capacity of all the nodes may be larger than the allowable receiving rate as this rate is dictated by the injected data rate by the source, the min term in the denominator ensures that the throughput efficiency is not reduced for a network topology with large capacity but having small injected data rate.

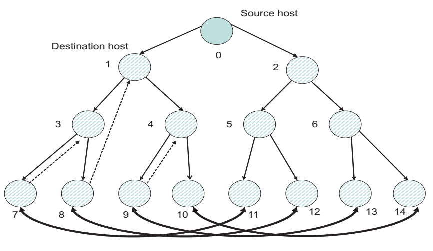


Figure 1: *Illustration of balanced mesh construction.*

3 Solutions

3.1 Balanced Mesh

We now proceed to consider a special scenario in which, all the nodes have the same upload bandwidth C. Furthermore, the nodes must be able to form a balanced mesh of degree b. We seek to build a data dissemination topology that results in large throughput efficiency, small delay, and the out-degree of each node is no more than b. We construct such a topology as a balanced mesh. A balanced mesh is first constructed as a balanced tree with a source node at the root. The leaf nodes of the tree are then connected together, and also connected up to their ancestors in a systematic manner to result in an efficiency data dissemination mesh. Figure 1 show an example of a balanced mesh with $b = 2$.

In this example, the source node partitions the data into two distinct groups and sends them down onto the left and right trunk of the tree. Thus, all the leaf nodes under the same trunk receive identical data. In order for the leaf nodes to receive the complete data, each leaf node is to connect to one other leaf node in the other trunk. For example, Figure 1 shows pairs of nodes 7 and 11, 9 and 12, 9 and 13, 10 and 14 connecting together. As a result, all the leaf nodes now receive the complete data. However, the internal nodes, e.g. nodes 3 and 4 in the left trunk are currently not receiving the data from the right trunk. To overcome this problem, the leaf nodes can forward the data received from the other trunk to their ancestors. As constructed so far, each leaf node currently sends data at only half of its capacity since it is connected to only one other leaf node. To fully utilize the bandwidth, each leaf node first forwards the data received from the other trunk to its parent. If its parent already receives the data from its other sibling, the leaf node forward the data to its grandparent. The process continues until all the nodes in the mesh receive the complete data. For example, node 7 forwards data from the right trunk to its parent (node 3). Since node 3 already receives that data from node 7, node 8 forwards its data from the right trunk to its grandparent (node 1). Node 9 forwards the data to its parent (node 4) while node 10 does not forward any data to its ancestor since there is no node in need to data in its trunk. We now present the general algorithm for constructing a b-balanced mesh. The algorithm ensures that all (a) the nodes in the balanced mesh receive the complete data, (b) no node has out-degree of more than b, number of hops from the source node to the destination nodes is $O(\log_b N)$ where N is total number of nodes, and (c) throughput efficiency $E = 1$.

To describe the algorithm, we first label the nodes as shown in Figure 1. In particular, nodes are labeled from low to high in a breadth-first manner. Within a level, the node labels increase from left to right.

Algorithm for constructing the balanced mesh is as follows.

1. Construct a balanced tree with each internal node having out-degree of b with the source node being the root.

2. Assuming the tree has i levels, each leaf node j in the leftmost trunk is then connected to $b-1$ other leaf nodes in each of different $b-1$ trunks. In particular, node j in the leftmost trunk is connected to nodes $k = j + b^{i-1}m$ where $m = 1, 2, ...b - 1$.

3. Each leaf node within a trunk from left to right except the rightmost node with respect to a particular parent, is connected back to its parent. This rightmost leaf node is connected to the grandparent. By this construction, each parent node will have b incoming connections, 1 connections from its parent and $b - 1$ connections from $b - 1$ of its children. The grandparent will also have b incoming connections, 1 connections from its parent and $b - 1$ connections from $b - 1$ of its grandparents. The reason for this is that there is exactly one grandchild from each of $b - 1$ parents that connects to the grandparent. Next, the grandchild of the rightmost parent is then connected to its grand-grand parent. The process continues until all the internal nodes or ancestors have exactly b incoming connections. Note that by construction, there will be exactly one rightmost leaf node within each trunk, e.g. nodes 10 and 14 in Figure 1, that will not connect to any ancestor.

Given the balanced mesh, the data dissemination algorithm is as follows.

1. The source node partitions the data into b distinct groups and send them down onto b trunks of the mesh at the rate of C/b bps per trunk. Each internal node in turn broadcasts the data down to its children also at the rate of C/b bps per link.

2. Since each leaf node is connected to $b-1$ other leaf nodes in other trunks, a leaf node can forward its data to $b - 1$ other leaf nodes in other trunks at the rate of C/b bps . As a result, each leaf node receive the complete data from $b - 1$ different leaf nodes and its parent.

3. By the construction, a parent is connected to $b - 1$ children, and therefore, $b - 1$ children forward $b - 1$ different partitions to their parents at the rate of C/b bps. This is possible because all the children, i.e. leaf nodes have the complete data. As a result, a parent node is able to receive the complete data. Similarly, all the ancestry nodes can receive the complete data since they all have b incoming connections from the leaf node with each leaf node forward a different data partitions.

Proposition 2: Out-degree for each node is at most b.

Proof: By construction, each internal node has exactly4 b out-connections to b children. With the exception of the rightmost leaf nodes, each leaf node has $b - 1$ out-connections to other leaf nodes, and one out-connection to its ancestor (e.g. parent, grandparent, ...). Thus all nodes have out-degree of b, except the b rightmost leaf nodes which have out-degree of $b - 1$.□

Proposition 3: The maximum node delay D is $log_b((b - 1)N + b) + 1$ where N is the number of destination nodes.

Proof: We omit the full proof, and note that the result can be obtained using geometric sum. □

Proposition 4: The throughput efficiency $E = 1$.

Proof: By construction, within a trunk, there is exactly one rightmost leaf node which does not forward its data to any of its ancestors. This rightmost leaf node, however, forwards its data to $b - 1$ leaf nodes at the rate of C/b bps. The rest of the "fully active" nodes within each trunk forward data at the rate of C bps. Since there are b trunks in a b-balanced mesh, the total sending rate of the entire mesh equals to sum of the sending rates of the source node, $N - b$ "fully active" nodes, and b rightmost leaf nodes, i.e. $\sum_{i=0}^{i=N} S_i = C + (N - b)C + b(b-1)C/b = NC$ bps. The denominator of E equals to $\min((N+1)C, NC) = NC$. Hence $E = NC/NC = 1$. □

3.2 Cascaded Balanced Mesh

In the previous case, the total number of nodes must be of the form $(b^i - 1)/(b - 1)$ where $i, b \in 0, 1, ...$ We now show an algorithm for constructing a mesh with arbitrary number of nodes, and still preserve desired properties. The main idea of the algorithm is to cascade a series of the balanced meshs in order to accommodate arbitrary number of nodes.

We note that there is exactly C bps of the remaining capacity after constructing a b-balanced mesh. In addition, there are b rightmost leaf nodes, each with only $b - 1$ out-connections. Hence, we can construct a new balanced mesh with the new root connected to b rightmost leaf nodes of the previous balanced mesh. The

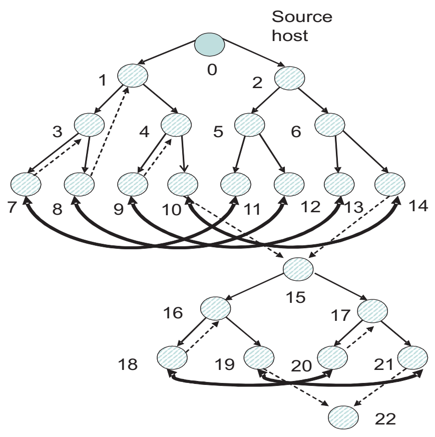


Figure 2: *A cascaded 2-balanced mesh.*

data is then sent from these nodes to the new root at the rate of C/b bps each, or a total rate of C bps. The new root then disseminates data to all the destination nodes in the same manner as that of the previous balanced mesh. The number of balanced meshs depends on the number of nodes in the mesh. Figure 2 shows an example of cascaded 2-balanced mesh consisting of 23 nodes. As seen, the remaining two nodes 10 and 14 of the previous balanced mesh have spare capacity to send their data to the new root node. Similarly, the nodes 19 and 21 send the data to the final node 22, the root of a new mesh without any children. The general algorithm for construction of a cascaded b-balanced mesh consisting of N destination nodes is as follows.

1. Construct a b-balanced mesh with the depth $i = \lfloor log((b-1)N + b\rfloor - 1$. This step constructs the deepest b-balanced mesh without exceeding the number of nodes. If there exists a previous b-balanced mesh, connect the b rightmost leaf nodes with extra bandwidth to the root of a newly created balanced mesh.

2. Set $N = N - (b^{i+1}-1)/(b-1)$. This is the number of remaining nodes.

3. If $N = 0$, stop. Otherwise, go back to step 1.

Since the construction of the cascaded balanced mesh is based on that of a balanced mesh, the properties of the cascaded balanced mesh are similar. In particular, the out-degree and throughput efficiency are exactly identical, i.e. out-degree is limited to b and the throughput efficiency is exactly 1.

Proposition 5: The delay in a topology of cascaded b-balanced mesh is $O((log_b N)^2)$.

Proof (outline): At each iteration of the algorithm, we construct the deepest b-balanced mesh without exceeding the number of nodes. Therefore, the remaining number of nodes after constructing a b-balanced mesh of maximum depth i cannot be greater than b^{i+1}. Otherwise, we can construct a b-balanced mesh of depth $i+1$ which contradicts the maximum possible i. Next, since the number of nodes in a b-balanced mesh of depth i is $(b^{i+1}-1)/(b-1)$, the maximum number of meshes of depth i that can cover the remaining nodes without exceeding the number of possible nodes is therefore $b^{i+1}(b-1)/(b^{i+1}-1) \leq b$. Therefore, we can construct at most b meshes of depth i before moving to the meshes of depth $j < i$. Hence, after the algorithm terminates, we have at most bi meshes with i being the depth of the first mesh. Since each mesh has depth of $O(i)$,the total delay is therefore $O(i^2)$, or equivalently $O((log_b N)^2)$. $\square$

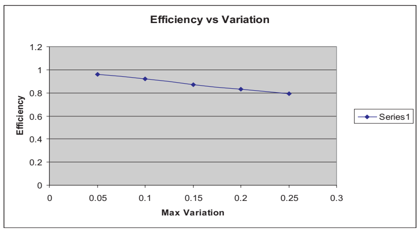
4 Simulation Results

We now present the simulation results for our proposed structured mesh. Figure 3(a) shows the throughput efficiency for our structured mesh vs maximum variation on capacity v. In particular, node capacities are uniformly generated between $C(1+v)$ and $C(1-v)$ where C is the mean capacity. As seen, the efficiency reduces as the capacity variation increases since an internal node may have small capacity which creates a bandwidth bottleneck for all its children. However, even when $v = 0.25$, the throughput efficiency are still 0.8%. Similar results are obtained when node capacity is normally distributed.

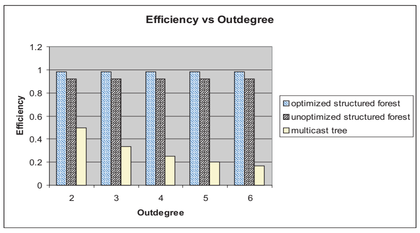
Figure 3(b) shows the throughput efficiency vs the outdegree for three different schemes: traditional multicast tree, unoptimized structured mesh, and optimized structured mesh. For optimized structured mesh, nodes with lower capacities are moved to the leaves to reduce bottleneck for other nodes. As seen, throughput efficiency is 98% for optimized structured mesh, 92% for unoptimized one. For the multicast tree, the throughput efficiency is small and decreases as the outdegree increases since the number of inactive nodes (leaf nodes) increases in this topology.

5 Related Work

We list a few representative work on data dissemination on the Internet. Similar to our approach, Byers and et.al. [5] propose to partition the data and make use of the peers to increase the throughput of the system. In this approach, each node randomly sends different partitions on different links. Data reconciliation techniques [6] are then used to reduce the data redundancy sent between the nodes. To address the transient and asynchrony issues of node joining and leaving the network, the paper advocates Forward Error Correction (FEC) approach in which a node can successfully recover the entire file using a fraction of the received packets. Similar work has also been done in [7]. In this work, the goal is to distribute data to a set of nodes in an overlay multicast tree in such a way to result in the



(a)



(b)

Figure 3: *(a) Efficiency vs variation; (b) Efficiency vs out-degree for different data dissemination schemes.*

approximate disjoint data set at these nodes. These individual nodes then can establish concurrent connections to other nodes in order to increase the download speed. The data reconciliation techniques similar to [5] are used to reduce overlapped data. Both of the aforementioned work focus on protocols and techniques for dynamically exchanging information between the nodes. On the other hand, our work focus on constructing the good topology with emphasis on throughput efficiency, node out-degree, node delay, and bandwidth fairness. In addition, unlike the randomized file partition approaches in [5] and [7], our file partition algorithm is simple, deterministic, and uses only a small number of partitions. Therefore, no data reconciliation is needed. Authors in [8] also propose to use multiple overlay multicast trees to stream multiple descriptions of the video to the clients. Each multicast tree contains a description of the video. When a large number of descriptions are received, the higher quality of video can be achieved. Unlike ours, the focus of this paper is on reliability and video quality. Most similar to our work is SplitStream [9]. In [9], the authors construct multiple multicast trees with the property that an internal node of one tree has to be the leaf node in the others for reliability issues. Data are then partitioned and sends on to different multicast trees. Unlike our work, SplitStream relies Scribe [10] and Pastry [11] infrastructure for tree construction without regarding for the constraints on out-degree and capacity of each node.

6 Conclusions and Future Work

We conclude our paper with the highlights of our contributions. First, we define the notion of throughput efficiency to measure the performance of different data dissemination topology and algorithms. Second, we devise algorithms for constructing a topology that outperform traditional multicast tree significantly, and at the same time, achieve the following properties: (1) end-to-end delay from the source node to any node is small ($O((logN)^2)$), (2) the out-degree of any node is small ($O(C)$), and (3) bandwidth usages of all the nodes are optimal. In the current proposed structured mesh, a node enters or leaves may affect a large number of nodes as the mesh may have to be rebuilt. This is unsuitable in a dynamic environment. We plan to investigate other topologies that enable node joining and leaving incrementally while preserving high throughput efficiency, small delay and outdegree. We also plan to extend our work to network with heterogeneous node capacity by first clustering of nodes with the same capacities and applying the proposed techniques to each cluster.

References

[1] S. Deering et al., "The pim architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 153–162, April 1996.

[2] 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*, 2003.

[3] http://www.gnutella.com.

[4] http://www.kazaa.com.

[5] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed content delivery across adaptive overlay networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 5, October 2004.

[6] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set reconciliation with nearly optimal communication complexity," in *IEEE International Symposium on Information Theory*, 2001.

[7] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *SOSP*, October 2003.

[8] V.N. Padmanabhan, H.J. Wang, P.A. Chou, and K. Sripanidkulchai, "Distributed streaming media content using cooperative networking," in *ACM NOSSDAV*, Miami, FL, May 2002.

[9] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in a cooperative environment," in *SOSP*, October 2003.

[10] A. Rowstron, A-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The design of a large-scale event notification infrastructure," in *NGC*, November 2001.

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