Rainbow: A Locality-aware Peer-to-Peer Overlay Multicast System*

Yang Chen, Bei-xing Deng, Xing Li
Department of Electronic Engineering, Tsinghua University, Beijing, China
chenyang04@mails.tsinghua.edu.cn

Abstract

In this paper, we propose Rainbow, a mesh based locality-aware P2P overlay multicast system. Nodes first self-organize into a two-layer overlay. A simple and accurate network coordinate system is used for more reliable node clustering. Pull-based method is used in data packet delivery. According to our simulation, the ARDP of Rainbow ranges from 63% to 76% of that in Binning overlay. Our real-world experiment on PlanetLab shows that the average packet loss rate of Rainbow is about 50% of that in ChainSaw.

1. Introduction

Existing overlay multicast protocols can be categorized into tree based and mesh based protocols. A common approach to P2P streaming is to organize participating peers into a tree-structured overlay over which the content is pushed from the source towards all peers. However, it also suffers from two major problems. Firstly, any data loss near the root node affects every node below it. Secondly, interior nodes are responsible to distribute data to all the children, while the leaf nodes do not upload at all. Now mesh based overlay networks are popular solutions for large scale P2P streaming.

If we do not consider the network locality in the construction of the overlay, nearby hosts in the overlay network may actually be far away in the underlying network. This may waste too much network resources and, therefore, degrade performance significantly. One important aspect of constructing an efficient overlay network is how to exploit network locality in the underlying network [1][2].

In this paper, we propose Rainbow, a mesh based locality-aware P2P overlay multicast system. Nodes self-organized into a two-layer hierarchical unstructured overlay. We use a simple and accurate network coordinate system, achieves more reliable node clustering. We use pull-based method in the data packet delivery. According to our simulation, the ARDP of Rainbow ranges from 63% to 76% of that in Binning overlay. Our real-world experiment on PlanetLab shows that the average packet loss rate of Rainbow is about 50% of that in ChainSaw.

The rest of this paper is organized as follows. First we review the related work in Section 2. Then we present our design of Rainbow in Section 3 and evaluate its performance in Section 4. We conclude the whole paper with Section 5.

2. Related Work

Ratnasamy *et al.* propose a binning scheme [2] to help the locality-aware overlay construction, which will be introduced in section 2.1. Tian *et al.* propose a hierarchical architecture, named HONet [3], will be introduced in section 2.2. We will introduce ChainSaw [4] in section 2.3, which is a mesh-based P2P overlay multicast system.

2.1. Topologically-aware construction of unstructured overlays

In Ratnasamy's binning scheme, nodes partition themselves into bins such that nodes will fall within a given bin are relatively close to one another in terms of network latency. Their scheme requires a set of well known landmark machines spread across the Internet. Each node measures its round-trip-time to each of these landmarks and orders the landmarks in order of increasing RTT. Thus, based on its delay measurements to the different landmarks, every node has an associated ordering of landmarks. This ordering represents the "bin" the node belongs to.

Binning scheme will be useful in the construction of locality-aware P2P overlay. In Rainbow, we will use a simple and accurate network coordinate system to

* This work is supported by the National Science Foundation of China (No.60473087)

Proceedings of the Fifth International Conference on
Grid and Cooperative Computing Workshops (GCCW'06)
0-7695-2695-0/06 $20.00 © 2006 IEEE

realize this. And we will compare the performance of binning scheme and our network coordinate in section 4.

2.2. HONet

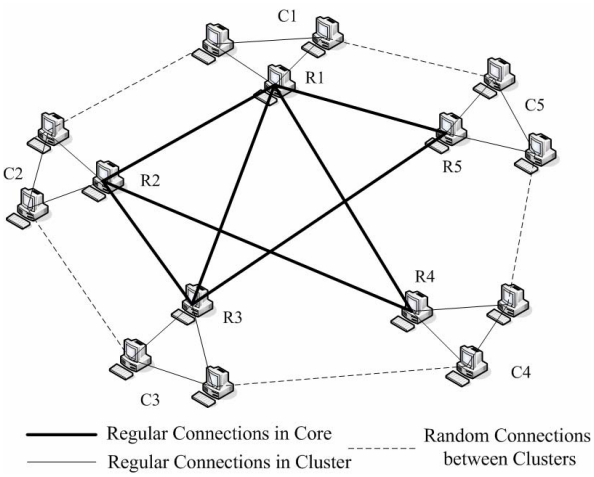

Figure 1. HONet Architecture

For network locality, nodes form clusters and each cluster provides a root node. All root nodes form a backbone network for inter-cluster traffic. In addition, random connections between members across clusters serve as shortcuts to reduce network delay and bandwidth consumption. As shown in Figure 1, HONet is organized in a two-layer hierarchy. The lower layer consists of many clusters with a root node for each cluster. The cluster root nodes form the upper layer, the backbone network. The backbone network and each cluster are constructed as structured overlays with independent naming spaces.

Because structured overlays are more expensive to maintain, in Rainbow, the backbone network and each cluster are constructed as unstructured overlays. We believe this will reduce the overlay maintain overhead and improve the flexibility.

2.3. ChainSaw

Pai *et al.* propose Chainsaw, a P2P overlay multicast system that completely eliminates trees. Chainsaw is a pull-based system that uses a randomly constructed graph with a fixed minimum node degree. Data is divided into finite packets and disseminated using a simple request-response protocol. Peers are notified of new packets by their neighbors and must explicitly request a packet from a neighbor in order to receive it. This way, duplicate data can be eliminated.

Though ChainSaw achieves good performance in real-world experiment on PlanetLab [4], we would like to see the performance improvement of using a locality-aware overlay instead.

3. Rainbow Architecture

In our design, Rainbow also uses a self-organized two-layer hierarchical architecture like HONet. The lower layer consists of many clusters with a root node for each cluster. The cluster root nodes form the upper layer, the backbone network. The backbone network and each cluster are constructed as unstructured overlays with a fixed minimum node degree. In addition, random connections between members across clusters serve as shortcuts to reduce network delay and bandwidth consumption. We propose a simple and accurate network coordinate system, achieve more reliable node clustering. We use pull-based method in data packet delivery.

3.1. Network Distance Prediction

An important step in Rainbow is node clustering, which requires the knowledge of nodes location. We employ a simple coordinate system to identify a node's network location, which uses a set of round-trip time from each node to a group of well-known landmark nodes. Any stable nodes which are able to response ICMP ping message can be chosen as landmark nodes.

After getting the network coordinate, we can predict the distance between two nodes. There are many distance predicting methods based on network coordinates, such as IDMaps [5], triangulated heuristic [6], GNP [6] and Vivaldi [7].

Rainbow uses triangulated heuristic to predict the network distance for both simplicity and accuracy. The triangulated heuristic predicts network distance by assuming shortest path routing. First N nodes in a network are selected to be landmark nodes Bi, i=0, 1, …, N. Then, node H is assigned coordinates which are simply the N-tuple of distances between H and the N base nodes, i.e. $(d_{HB_1}, d_{HB_2}, ..., d_{HB_N})$. Given two nodes H1 and H2, assuming the triangular inequality holds, the lower bound of the distance between H1 and H2 is

$$L = max_{i \in \{1,2,...,N\}} \left(\left| d_{H_1 B_i} - d_{H_2 B_i} \right| \right) \qquad (1)$$

And the upper bound is

$$U = min_{i \in \{1,2,...,N\}} \left(\left| d_{H_1 B_i} + d_{H_2 B_i} \right| \right) \qquad (2)$$

Various weighted averages can then be used as distance functions to estimate the distance between H1 and H2.

3.2. Node Clustering in Rainbow

After getting the network distance between nodes, we cluster the nodes. In real network applications, nodes can join or leave the network at any time, which will cause a heavy burden on centralized system. Therefore, we use a distributed clustering algorithm.

To join the overlay, a node first identifies its coordinates in the network. Then it uses triangulated heuristic to find the closest cluster root. In our design, if the new node cannot locate nearby cluster roots, or its distance to the nearest cluster root is larger than a threshold T, this node joins the backbone network as a new cluster root and announces its coordinates in the backbone network. Otherwise, the node joins the cluster led by its nearest cluster root.

Because we design our algorithm in a distributed way, no end-to-end measurement is performed between all pairs of nodes. This design reduces the overhead for large scale network and provides Rainbow with high scalability.

In triangulated heuristic, shortest path routing is assumed. But due to the inefficient routing behavior in the Internet, this assumption is not always true. For example, in [8], for all the triangular closed loop paths (a,b), (b,c) and (a,c) that they measured, 7% of the ratios $(a,c)/((a,b)+(b,c))$ are greater than 1. This error hurts the accuracy of the network distance prediction.

Base on our experiment on more than 150 PlanetLab nodes [9], the average distance between each pair of nodes in the same cluster is 23ms and the average distance between each pair of nodes in different clusters is 193ms. So the result of our node clustering is reliable, keeping the intra-cluster distance small and inter-cluster distance large.

3.3. Rainbow Overlay

There are three types of nodes in Rainbow overlay: one *Login Sever*, one *Data Server* and all the other nodes are *ordinary peers*.

We summarize the procedure that a new ordinary peer A joins a Rainbow overlay. Host A first contacts the rendezvous service provided by the Login Server. After obtaining a list of cluster roots from the Login Server, Host A joins a nearest cluster or starts a new cluster. After joining the overlay, Host A participates in the root election in its cluster. If A is selected as the root of the cluster, it will join the backbone network and create connections to root nodes of other clusters; otherwise, it will create random connections to nodes in other clusters. To construct random connections easily and adaptively, we use the random walk algorithm proposed in [3]. Though nodes can join and leave the overlay at any time, cluster will make corresponding adjustment as soon as possible, not only when the root node departs unexpectedly but also when there is a more appropriate node to serve as root node. The root election, random connections creation and cluster improvement tasks may be done in parallel. To detect node failure, root nodes exchange "heartbeat" messages, so do members in each cluster.

3.4. Data Packet Delivery

After the construction of the overlay, stream packets which are generated by the Data Server will be deliver to all ordinary peers through the overlay. Every peer connects to a set of nodes that we call its neighbors. Peers only maintain state about their neighbors. The main piece of information they maintain is a list of packets that each neighbor has. When a peer receives a packet it sends a NOTIFY message to its neighbors. The seed obviously does not download packets, but it sends out NOTIFY messages whenever it generates new packets. Every peer maintains a *window of interest*, which is the range of sequence numbers that the peer is interested in acquiring at the current time. It also maintains and informs its neighbors about a *window of availability*, which is the range of packets that it is willing to upload to its neighbors.

For every neighbor, a peer creates a list of desired packets, i.e. a list of packets that the peer wants, and is in the neighbor's window of availability. It will then apply some strategy to pick one or more packets from the list and request them via a REQUEST message. Some apply earliest packet first strategy, which will increase delay by not picking new packets. Some apply random strategy, which there may be some packets not get picked from the Data Server for a long time.

Currently, we use a pull-based packet picking strategy similar with Chainsaw [4]. Ordinary peers use random strategy, and the Data Server is smarter. If the Data Server has unsent packets and it receives a request for a previously sent packet, it answers the request with an unsent packet instead. This algorithm ensures that at least one copy of every packet is uploaded quickly, and the seed will not spend its upload bandwidth on uploading packets that could be obtained from other peers unless it has spare bandwidth available.

4. Performance Evaluation

We both use simulation and experimental study on PlanetLab to evaluate the performance of our architecture. In Section 4.1 and 4.2, we evaluate the

Rainbow overlay through extensive simulation. In Section 4.3 and 4.4, we evaluated Rainbow on PlanetLab test-bed [11].

4.1. Simulation Setup

We use GT-ITM to generate transit-stub network topologies for our simulation. We generate 5 topologies each with about 9600 routers and 57000 edges. We assign different distances to the edges in the topologies (with values from [10]): The distance of intra-stub edges is 1; the distance of the edges between transit node and stub node is a random integer in [5; 15]; and the distance between transit nodes is a random integer in [50; 150]. Five runs are performed on each network topology and the average value reported.

The main performance metric we adopt in our simulation is the Average Relative Delay Penalty (ARDP). Here we distinguish the latencies in an overlay, which we call overlay latencies, from the unicast latencies, which are the latencies in the underlying physical network. Relative Delay Penalty (RDP) is the ratio of the overlay latency between nodes i and j to their unicast latency. ARDP is the average RDP among all node pairs:

$$ARDP = \frac{1}{N} \sum_{i,j(i \neq j)}^{N} \frac{D'_{i,j}}{D_{i,j}} \qquad (3)$$

Where N is the number of node pairs in the overlay. Smaller ARDP indicates that most overlay latencies are close to the respective unicast latencies. If the overlay is a full mesh, where all members are directly connected, the ARDP is 1.

4.2. Simulation on Generated Topologies

We now compare the ARDP of the following three overlays: Random Overlay, Binning Overlay and Rainbow Overlay. In Random Overlay, each node picks k neighbors at random. In Binning Overlay, each node picks k/2 neighbors at random from its own bin and picks the remaining k/2 neighbors at random in other bins. In our Rainbow overlay, each node picks k/2 neighbors at random from its own cluster and picks the remaining k/2 neighbors at random in other clusters.

With each topology, we scale the overlay size by adding nodes to the stub (leaf) nodes in the underlying topology. The delay of the link from the end-host node to the stub node is set to 1ms. Thus, in scaling the overlay size from 2,000 to 32,000 nodes, we're scaling the density of the graph without scaling the backbone (transit) domain. In our simulation, k is set as 6 and we choose 8 landmark nodes both for Binning Overlay and Rainbow Overlay randomly. Five runs are performed on each network topology and the average value reported.

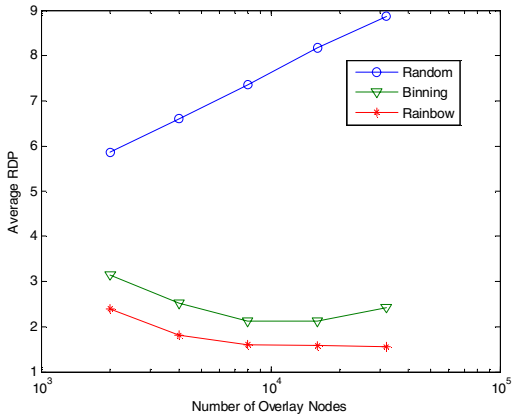

Figure 2. Average RDP of Random / Binning / Rainbow Overlay

Figure 2 shows the comparison of the ARDP among the three overlays. Our simulation result shows that both Binning Overlay and Rainbow Overlay perform much better than Random Overlay, which indicates that taking nodes' locality into consideration will reduce the ARDP much. In addition, comparing Rainbow Overlay with Binning Overlay, we find that Rainbow performs better than Binning, especially in larger scale. When the overlay size is 2,000 nodes, the ARDP in Rainbow is 76% of that in Binning. When the overlay size grows up to 32,000 nodes, the ARDP in Rain-bow is 63% of that in Binning. So Rainbow archives a better scalability.

4.3. Experiment Setup

Our experiment is conducted on PlanetLab [11]. PlanetLab is an open platform for developing, deploying and accessing planetary-scale services. Currently PlanetLab has 691 machines in 335 sites all over the world. We evaluated Rainbow on almost all the active nodes of PlanetLab, about 430 PlanetLab nodes in March 2006. Each PlanetLab node runs a copy of Rainbow program, acting as a node in the overlay. The Login server node is located in United States (planetlab4.lcs.mit.edu). The Data Server node is also located in United States (planetlab1.cs.uiuc.edu). We chose 15 landmark nodes, five in Asia, six in North America and four in Europe. T is set as 50ms. Our default stream rate is 530kbps and each packet contains one second of the stream. Each node maintains a window of availability of 50 seconds of the streaming data. We run each experiment six times independently. For comparison, we also run another program which using the same overlay construction and packet

delivery strategy with ChainSaw. We now compare the performance of Rainbow with Chainsaw. The main performance metric we adopt in our real-world experiment is the *packet loss rate*, is defined as the number of packets that does not arrive before or on playback deadlines over the total number of the packets.

4.4. Performance Evaluation on PlanetLab

Round	ChainSaw	Rainbow
1	4.21	2.35
2	3.99	2.16
3	4.83	2.15
4	4.50	2.43
5	4.11	2.36
6	4.39	2.32
Average	4.34	2.29

Table 1. Packet Loss Rate of ChainSaw / Rainbow

According to Table 1, average packet loss rate of Rainbow is 2.29%; average packet loss rate of ChainSaw is 4.34%. Our experiment result indicates that the average packet loss rate of Rainbow is about 50% of that in ChainSaw.

5. Conclusion and Future Work

In this paper, we propose Rainbow, a mesh based locality-aware Peer-to-Peer Overlay Multicast system. According to our simulation, the ARDP of Rainbow Overlay is not only much better than Random Overlay, but also better than Binning Overlay. The ARDP of Rainbow is 63% to 76% of that in Binning overlay. According to our real-world experiment on PlanetLab, the average packet loss rate of Rainbow is less than 50% of that in ChainSaw.

We currently focus on taking IP-layer information into account in our overlay construction. We also consider designing a smarter packet delivery strategy.

6. References

[1] Xin Yan Zhang, Qian Zhang, Zhensheng Zhang, *et al.* "A Construction of Locality-Aware Overlay Network: mOverlay and Its Performance". *IEEE Journal on Selected Areas in Communications*, Vol.22, No.1, Pages 18-28, 2004.

[2] Sylvia Ratnasamy, Mark Handley, Richard Karp, Scott Shenker. "Topologically-Aware Overlay Construction and Server Selection". *In Proc of IEEE INFOCOMM'02*, 2002.

[3] Ruixiong Tian, Yongqiang Xiong, Qian Zhang, Bo Li, Ben Y. Zhao, Li Xing. "Hybrid Overlay Structure Based on Random Walk". *In Proc of the 4th International Workshop on Peer-To-Peer Systems (IPTPS'05),* 2005.

[4] Vinay Pai, Kapil Kumar, Karthik Tamilmani, *et al.* "Chainsaw: Eliminating Trees from Overlay Multicast". *In Proc of the 4th International Workshop on Peer-To-Peer Systems (IPTPS'05)*, 2005.

[5] Paul Francis, Sugih Jamin, Cheng Jin, *et al.* "IDMaps: A Global Internet Host Distance Estimation Service". *IEEE/ACM Transactions on Networking, v9n5*, Pages 525-540, 2001.

[6] T. S. Eugene Ng and Hui Zhang. "Predicting Internet network distance with coordinates-based approaches". *In Proc of IEEE INFOCOMM'02*, 2002.

[7] Frank Dabek, Russ Cox, Frans Kaashoek, *et al.* "Vivaldi: A Decentralized Network Coordinate System". *In Proc of ACM SIGCOMM'04*, 2004.

[8] Paul Francis, Sugih Jamin, Vern Paxson, *et al.* "An architecture for a global Internet host distance estimation service". *In Proc of IEEE INFOCOMM'99*, 1999.

[9] Yang Chen, Bei-xing Deng and Xing Li, "Experimental study on network coordinate based node clustering". *Journal of Dalian University of Technology*, Vol.45, Supp.1, Pages 41-43, 2005.

[10] The pinger project. http://www-iepm.slac.stanford.edu/pinger/.

[11] PlanetLab. http://www.planet-lab.org/.