Design and Deployment of Locality-aware Overlay Multicast Protocol for Live Streaming Services[⋆]

Xuping Tu, Hai Jin, Dafu Deng, Chao Zhang, and Quan Yuan

Cluster and Grid Computing Lab
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

Abstract. This paper presents the design and deployment of a locality-aware overlay multicast protocol called *Anysee*. The key idea of *Anysee* is to use the geometrical information of end hosts to construct the locality-aware overlay data delivery tree such that nearby users in the underlying network can be organized into nearby subtrees. The prototype of *Anysee* has been widely used in CERNET. Logging traces obtained from broadcasting 2004 Athens Olympic Games over 16 days have shown that the performance of *Anysee*, such as end-to-end delay and absolute data delivery delay, significantly outperforms that of randomly constructed overlay multicast.

1 Introduction

Network-level IP multicast [1] [7] was proposed over a decade ago. It seems (or, at least, was designed) to be the idea solution for efficiently disseminating real-time media content over Internet. However, the lack of high level features such as reliability, *quality of service* (QoS) control, and security, as well as the necessary of changes at the Internet infrastructure level make it very difficult to be widely deployed.

As a result, application level multicast protocols [2] [4] [5] [8] [9] [12] have gained tremendous momentum in recent years. In particular, for high-quality video streaming service, routing overhead is a key performance metric for the overlay video data disseminating tree since each stream tends to consume large amount of underlying bandwidth. If the overlay tree is constructed randomly, e.g. Coopnet [11], nearby hosts in the overlay tree may actually be far away in the underlying network. In this method, the QoS requirements for media data delivery is very difficult to be guaranteed.

End System Multicast (ESM) and its extension [4] [5] [6] give out *Narada* protocol and deployment for broadcasting video conference streams to a small (or moderate) group users. The main idea of ESM is that end-hosts exclusively exchange group membership information and routing information, build a mesh, and finally run a DVMRP-like(Distance Vector Multicast Routing Protocol) protocol to construct a overlay data delivery tree. ESM focuses on the out-going bandwidth limits of end hosts and the reduction of source to user latency. However, it does not address on large-scale issues.

Other schemes, such as Overcast [9], NICE [2], Zigzag [13], Scattercast [3], and TAG [10], present different optimization methods to extend the system to larger-scale

[⋆] This work is supported by National Science Foundation of China under grant No.60433040.

cases under different conditions. However, as to our understand, all of them have not been widely deployed.

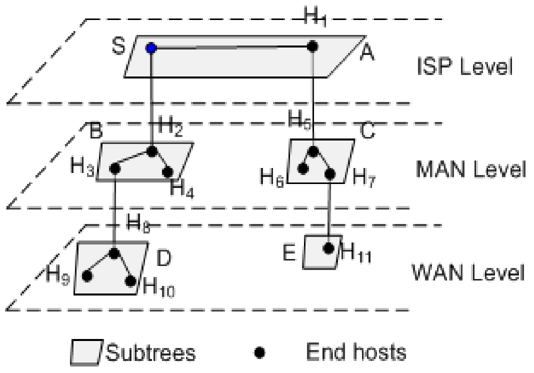
In this paper, we give out the design and deployment of a locality-aware multicast protocol called *Anysee*. *Anysee* is tailored to broadcast high-bandwidth video streams to a lager amount of users with low latency. The rest of this paper is organized as follows. Section 2 describes the locality-aware multicast protocol of *Anysee*, together with the prototype of *Anysee*, and gives out its performance analysis. Section 4 ends with conclusions.

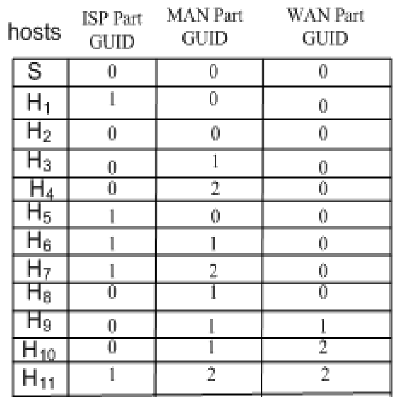
2 Locality-aware Overlay Multicast

The key idea of *Anysee* is to use the geometrical information of end hosts to construct the locality-aware overlay data delivery tree such that nearby users in the underlying network can be organized into nearby subtrees. It also supports *Network Address Translater* (NAT) traversals.

2.1 Tree Organization

To organize the overlay multicast tree into the locality-aware fashion, *Anysee* uses a L-parts *Global Unique Identify* (GUID) to identify the network position of an end host, where each part of the GUID value corresponds to the network and geometrical information. For example, we can statically divide the entire Internet to five-levels: the inter-country level, inter-ISP (*Internet Service Provider*) level, MAN (*Metropolitan-Area Network*) level, WAN (*Wide-Area Network*) level, and LAN (*Local-Area Network*) level, respectively. For each level, a corresponding part of GUID value of an end host is generated by the geometrical or network (i.e. ISP) information of that host.





hosts	ISP Part GUID	MAN Part GUID	WAN Part GUID
S	0	0	0
H_1	1	0	0
H_2	0	0	0
H_3	0	1	0
H_4	0	2	0
H_5	1	0	0
H_6	1	1	0
H_7	1	2	0
H_8	0	1	0
H_9	0	1	1
H_{10}	0	1	2
H_{11}	1	2	2

Fig. 1. An example of 3-layer hierarchy of subtrees

Corresponding to the L-part GUIDs, the overlay multicast tree is also organized as a L-level hierarchy of subtrees. A subtree at level i is comprised of end hosts whose i^{th}

part of GUID values are different from each other, and their $j^{th}(i < j \leq L-1)$ part GUID values are the same as each other. Each host at layer $i(0 < i \leq L-1)$ must reserve one out-degree for severing another host with the same i^{th} part GUID value.

Fig.1 illustrates an example of 3-level hierarchy of subtrees. In this case, hosts H_2, H_3, H_4, and S have the same inter-ISP level GUID value (i.e. H_2, H_3, H_4, and S are severed by the same ISP). Since they are located at different cities and have different MAN-level GUID values, they are organized into a subtree B at the MAN-level layer.

2.2 Tree Management

Initially, the entire tree contains a single subtree at the highest layer, consisting of a source host. The layer number of the source node is initialized as the serial number of the highest layer. To effectively construct and manage the overlay tree, an end host should maintain a small amount of state information–IP addresses and port number, layer number, connectivity constraints, and the GUID value of itself, its parent, grandparent, source host and children hosts.

New Host Joins: In our solution, we simply use the absolute difference value between GUIDs of any two hosts to predict their network distance. Given a new host X, it begins its join process by sending the ”*Join*” message to the source node, where the ”*Join*” message contains its GUID value and connectivity constraints information. Once an existing host Y at layer $i(0 < i \leq L-1)$ receives the ”*Join*” message sent by the new host, it uses the following rules to admit X and determine its level number. 1) If Y is the nearest host to X (comparing with its children), X will be admitted as a child of Y. In this case, the layer number of X is determined by the i^{th} part GUID of X and Y. If the i^{th} part GUID value of X is equal to that of Y, the layer number of X is assigned to $i-1$. Otherwise, the layer number of X is equal to i. 2) If Y is not the nearest host but it has enough upload bandwidth to serve X, X will be admitted as a child at current level. 3) If Y has not enough remaining upload bandwidth to sever X and it has a child Z which is the nearest one to X, it sends ”*Redirect*” message to the new host to redirect it to Z. 4) If X receives a ”*Redirect*” message, it resents the ”*Join*” message to the redirected host. The process repeats until X finds out its nearest parent. An except holds when the redirected one is a *freerider*. In this case, the new host is inserted between the existing host and the nearest *free-rider*. If the ”*Join*” message is sent to a host located at layer 0, it simply uses the *First-Come-First-Sever* (FCFS) with randomly redirecting method to admit the new host.

Fig.2 shows an example of a join process. In this figure, the ISP-part, MAN-part, and WAN-part GUID values of the new host is 1, 2, and 3, respectively. It first contacts the source host S to initialize the join process. S finds out that the new host and its child H_1 are in the same ISP network. Thus, it redirects the new host to H_1. Based on the nearest parent selection principle, the new host will be redirected to H_5 and H_7, respectively, until it finds out the nearest parent H_{11}.

Host Departs: Host departure due to purposely leave or accidently failure can be detected by its children since the video data stream will be interrupted. Children of the departed host send ”*Parent-Leave*” messages to their original grandparent to launch a recovery process. If unfortunately, the grandparent leaves at the same time, children should send ”*Join*” message to the source node to rejoin the overlay tree.

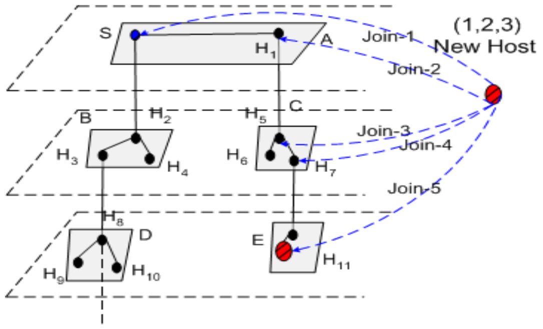


Fig. 2. An example of join process

Recovery from the departure of a host at layer 0 is trivial. The parent of the departure host randomly admits a grandchild to be its new child and redirects others to the admitted one. We propose the recovery process from the departure of a host at high layers. In this case, the parent of departed host sends a "*Probe*" message to that child. A non-leaf host forwards the message while the leaf node responses an "*Probe-Response*" message. Finally, the parent redirects other children of departed host to the promoted one.

Freeriders Supports: *Freeriders* can be detected by the *rendezvous point* (RP) when it requests the IP address and Port number of the source host. In particular, users behind NATs can be detected by comparing their public IP addresses and their private IP addresses (the private IP address is contained in their request message). This will be beneficial to admit more NAT-users since hosts behind the same NAT can be grouped into the same branch.

2.3 Prototype

The entire system is comprised of four components: a *rendezvous point* (RP), media sources, a monitor, and end systems. Source hosts are responsible for receiving the encoded video stream and promulgating them to end hosts. Each end system first accesses the RP machine to obtain the IP address and port number of source host and detect the connectivity constraints (i.e. whether the end host is behind NAT and firewall). Then, it joins the overlay network and periodically reports performance information to the monitor. The monitor is responsible for logging the performance information of joined hosts.

We implemented *Anysee*, and released the first version (v.1.0Beta) on August 12, 2004. This version runs on CERNET[1]. Each copy of end system software has combined a pre-built IP-to-GUIDs database that contains all class **C** and class **B** IP addresses in CERNET.

The system has been used by HUSTOnline (http://www.hustonline.net) for broadcasting high-quality TV streams (near 512kbps bit-rate) to students. We have analyzed

[1] CERNET stands for China Education and Research Network. It covers over 1000 colleges (or institutions) in China. More information can be found at http://www.cernet.edu.cn/

logging traces gathered from 13/8/04 to 29/8/2004. During this period, the 2004 Athens Olympic Games is broadcasted via four source hosts. Each of them corresponds to a unique TV channel.

2.4 Metrics

The metrics we are interest in the performance analysis are:

Control Overhead: This is measured by the time consumption of join processes, and time consumption of recovery processes.

Quality of Data Path: We evaluate the network proximity performance via the *End-to-End Delay* (EED) and the *Absolute Delay Penalty* (ADP) in the overlay multicast tree. EED is the *round-trip time* (RTT) between a parent and a child in the overlay multicast tree. It reflects QoS issues when a considerable end system buffer (20 seconds) has been used. ADP is defined as the cumulative latency to promulgate a data packet along the overlay path from the source host to an end host.

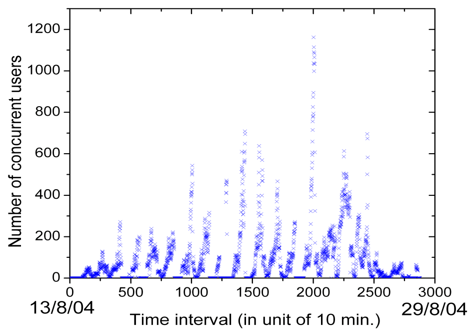


Fig. 3. The tree size vs. Time interval

2.5 Analysis Methodology

Logging traces shows that almost 7200 users distributed among 40 colleges in 14 cities have enjoyed our contributions. The number of maximum concurrent users supported by the entire system is 3749, and the number of maximum concurrent users supported by a single overlay multicast tree is 1162. We use the tree with size of 1162 users for performance analysis. Fig.3 shows the changes of the number of maximum concurrent users over 16 days. We choose 10 different time intervals (length of each time interval is 40 minutes) to reconstruct the overlay multicast tree for detail analysis, while the tree sizes (i.e. numbers of concurrent users) at these time intervals are between 100 and 1162.

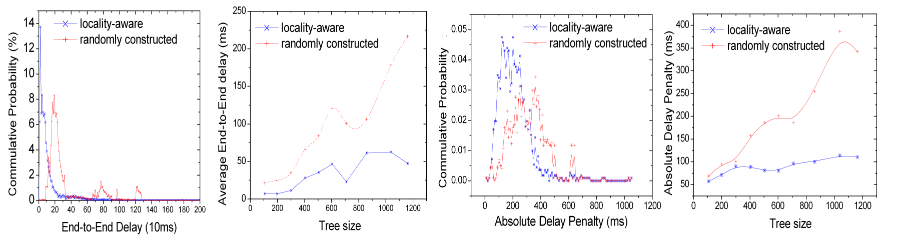
We choose the random construction method for comparison. In each time interval, we first compute the average end-to-end RTT and the average data promulgating delay between any two intra-region hosts and between any two inter-region hosts based on the measured latency information. Then, according to the real-life user accessing sequence, we reconstruct the overlay tree in the random fashion, where *free-riders* are processed by the method described in section 2. Finally, corresponding to the geometrical information of joined hosts, we assign the average end-to-end RTT value and the average data promulgating delay to neighbor hosts on randomly constructed trees to evaluate its data path quality.

Table 1. Basic performance results, where m/n in the NAT column represents that m NAT-users come from n different NATs

Size	Date	Height	Mean join	Mean recovery	EED (ms)	ADP (ms)	NAT	Firewall
			overhead (ms)	overhead (ms)			users	users
106	$14/8/04$	5	690	2,112	7.3	57.2	$21/18$	28
198	$15/8/04$	5	693	1,966	7.0	71.2	$22/14$	37
301	$17/8/04$	6	1,027	1,312	11.3	90.2	$33/28$	46
405	$18/8/04$	6	1,050	2,004	28.1	88.6	$35/32$	46
504	$19/8/04$	7	1,058	1,897	35.4	80.9	$29/23$	56
604	$22/8/04$	7	719	876	46.4	80.5	$35/29$	48
709	$20/8/04$	7	519	935	23.0	96.6	$89/65$	64
859	$21/8/04$	7	520	900	61.4	100.2	$95/70$	79
1038	$24/8/04$	7	1,018	2,338	62.7	114.3	$108/76$	103
1162	$26/8/04$	7	658	1,031	47.4	110.6	$123/85$	120

2.6 Performance Results

Table 1 shows basic performance results of *Anysee* system. From this table, it can be seen that the join and recovery overhead of *Anysee* system is very low. Users can quickly find out their nearest parents for requesting video data. Theoretically, if the average number of hosts in a subtree of *Anysee* system is m and all joined hosts are uniformly distributed in different subtrees, we have $\sum_{i=1}^{L} m^i = N$, where N is total number of joined hosts. Thus, $m \leq \sqrt[L]{N}$. Since the average height of a subtree is in order of $O(\log m)$, the amortized height of the entire overlay tree is in order of $O(L \log \sqrt[L]{N})$. Consider that the "*Join*" message and "*Probe*" message will traverse the branch with maximum path length (i.e. tree height) in the join algorithm and the recovery algorithm, respectively. Thus, the time complexities of both join and recovery procedure are upper-bounded by $O(L \log \sqrt[L]{N})$. As shown in Table 1, the mean time for join processes and recovery processes is less than 2 seconds. And the latter is larger than that for join processes. The main reason of this scenario is that lots of users tends to leave the overlay in a short time interval when the interested media content has finished. In this case, children of leave hosts will rejoin the overlay tree.



(a) CDF of End-to-End delay (Tree size: 1162)

(b) End-to-End Delay **vs.** Tree size

(c) CDF of Absolute Delay Penalty (Tree size: 1162)

(d) Absolute Delay Penalty **vs.** Tree size

Fig. 4. Comparison of data-path quality between locality-aware overlay multicast and randomly constructed overlay multicast

Fig.4 (a) shows the comparison of cumulative distribution function of end-to-end delay between the locality-aware overlay and the randomly constructed overlay, with the overlay tree size 1162 users. In this figure, for locality-aware multicast, the end-to-end delays for almost 95% parent-to-child pairs are less than 100 ms. However, for randomly constructed overlay tree, almost 93% end-to-end delay values are within range $[100ms, 300ms]$. Fig.4 (b) shows clearly that the average end-to-end delay in locality-aware overlay tree is far less than that in randomly constructed tree.

As shown in Table 1, heights for locality-aware multicast trees with size between 106 and 1162 users are larger than or equal to 5. Obviously, the height of randomly constructed overlay tree is less than $\log_5 N$, where 5 is the out-degree of an end host and N is the amount of joined hosts. Thus, for randomly constructed overlay trees with size less than 1162, the corresponding height is also less than $log_5 1162 \leq 5$. Clearly, the larger height of locality-aware overlay multicast tree is resulted from which some out-degrees of inner-hosts are reserved for severing future nearest hosts and not used in the practice.

However, as shown in Fig.4 (c) and (d), the absolute delay penalty of locality-aware overlay tree is far less than that of randomly constructed overlay tree. In Fig.4 (c), for the tree with size 1162 users, the absolute delay penalties for most users in the locality-aware overlay tree are between 100ms and 400 ms, while the absolute delay penalties for most users in the randomly constructed overlay tree are between 200ms and 600ms. Fig.4 (d) shows a clear comparison of average absolute delay penalty between locality-aware overlay multicast tree and randomly constructed overlay multicast tree. From this figure, it can be seen that the average absolute delay penalty for randomly construction method is as three times as that for locality-aware construction method when the tree size is larger than 1000 users.

Network Address Translator (NAT) partially solves the address exhaustion problem of IPv4 and firewall gives out a solution for security issue. However, they create many challenges to peer-to-peer overlay applications since hosts behind NAT gateways or firewalls are often restricted to serve as receivers only, not suppliers. As our log indicates (shown in Table 1), near $20\% \sim 45\%$ *Anysee* users are behind NATs or firewalls. In our implementation, we use the inserting mechanism to make hosts behind NAT or firewalls as leaves of the overlay multicast tree. In additional, users behind same NAT gateway can be organized into the same subtree.

3 Conclusion

In this paper, we present a live streaming system called *Anysee*. *Anysee* uses a locality-aware overlay multicast protocol to broadcast high-quality video streams to large amount of users. It also supports NAT(or firewall)-traversals. We have studied the performance of *Anysee* based on logging traces of broadcasting 2004 Athens Olympic Games on CERNET.

References

1. T. Ballardie, P. Francis, and J. Crowcroft, ”Core Based Trees (CBT): An Architecture for Scalable Multicast Routing”, In *Proc. of ACM Sigcomm*, 1995.
2. S. Banerjee, B. Bhattacharjee, and C. Kommareddy, ”Scalable application layer multicast”, In *Proc. of ACM Sigcomm*, Aug. 2002.
3. Y. Chawathe, ”Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastruture Service”, PH.D. Thesis, University of California, Berkeley, Dec. 2000.
4. Y.-H. Chu, S. G. Rao, and H. Zhang, ”Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture”, In *Proc. of ACM Sigcomm*, Aug. 2001.
5. Y.-H. Chu, S. G. Rao, and H. Zhang, ”A Case for End System Multicast”, In *Proc. of ACM SIGMETRICS*, June 2000.
6. Y.-H. Chu, S. G. Rao, and H. Zhang, ”Early deployment experience with an overlay based Internet Broadcasting System”, In *Proc. of USENIX Annual Technical Conference*, June 2004.
7. S. Deering and D. Cheriton, ”Multicast Routing in Datagram Internetworks and Extended LANs”, In *ACM Transactions on Computer Systems*, May 1990.
8. P. Francis, ”Yoid: Extending the Multicast Internet Architecture”, *White paper*, http://www.aciri.org/yoid/, 1999.
9. J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O’ Toole, ”Overcast: Reliable Multicasting with an Overlay Network”, In *Proc. of the 4th Symposium on Operating Systems Design and Implementation*, Oct. 2000.
10. M. Kwon and S. Fahmy, ”Topology-aware Overlay Networks for Group Communication”, In *Proc. of ACM NOSSDAV’02*, May 2002.
11. V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, ”Distributing Streaming Media Content Using Cooperative Networking”, In *Proc. of NOSSDAV’02*, USA, May 2002.
12. D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, ”ALMI: An Application Level Multicast Infrastructure”, In *Proc. of 3rd Usenix Symposium on Internet Technologies & Systems*, March 2001.
13. D. A. Tran, K. A. Hua, and T. T. Do, ”A peer-to-peer architecture for media streaming,” In *IEEE J. Select. Areas in Comm.*, Vol. 22, Jan. 2004.