Anysee2: An Auto Load Balance P2P Live Streaming System with Hybrid Architecture (Work-in-Progress)

Qi Huang, Hai Jin, Ke Liu, Xiaofei Liao, Xuping Tu
Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

ABSTRACT
Nowadays, Peer-to-Peer Technology has been widely used in live streaming applications and many related systems are proposed. However, their single overlay design and unbalanced scheduling methods lead to some inefficiency including high control overhead and bad playback experience. This paper mainly discusses how to address these certain problems. We introduce a hybrid architecture to solve the locality problem and reduce the control overhead. We also propose a scheduling method to achieve load balance. All solutions are implemented in Anysee2.

Categories and Subject Descriptors
C.2.4 [**Distributed Systems**]: Distributed applications.

General Terms
Algorithms, Design.

Keywords
Peer-to-Peer; Live Streaming; Hybrid Overlays; Load Balance

1. INTRODUCTION
There are three key problems in P2P live streaming system, including constructing an efficient overlay, managing the buffer and keeping the load balance. Former researchers have proposed different approaches. However, since they have not solved all the problems above, there still exists some inefficiency.

Structured/Tree-based system like PeerCast [3] is scalable. But parent peers in it suffer a heavy transferring burden. Moreover, complex tree adjustment in the dynamic network often affects the Qos (Quality of Service). Mesh-based system like Coolstreaming [6] is popular for the adaptability of network fluctuation. But gossip based protocol limits their scalability. And in a network resource constraint environment, observation shows that load imbalance would result in bad streaming experience [4].

Anysee [5] and Bullet [2] tried to optimize the system by transferring compensatory data through multi overlays. However, they still have to take a tradeoff between control and data overhead in any single overlay. This paper proposes an advanced version of Anysee called Anysee2 to address these problems. Anysee2 takes a hybrid architecture to divide control and data into different independent overlays. Anysee2 also proposes a new scheduling policy to achieve the load balance.

INFOSCALE 2007, June 6-8, Suzhou, China
Copyright © 2007 ICST 978-1-59593-757-5
DOI 10.4108/infoscale.2007.894

2. DESIGN AND ALGORITHMS
2.1 Tree-Mesh Hybrid Architecture
Anysee2 proposes a hybrid architecture to transfer the control message and media data in different overlays. Control Tree guarantees the scalability and efficiency of control messages transfer. Data Mesh guarantees the good QoS in dynamic network.

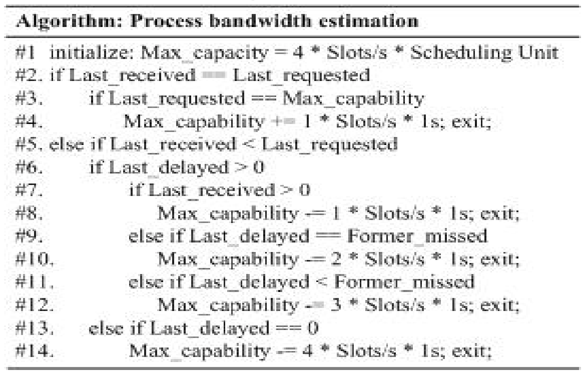
In Anysee2, every peer has its own GUID (Global Unique Identifier) denotes as [ISP, city, postcode, public IP, private IP, type, extend]. It is generated by bootstrapping server based on the information of IP database and used as the landmark for Control Tree construction.

Control Tree is the tree-based control overlay of Anysee2 system. It separates all the control messages from media data transfer. Therefore, mechanisms of peer join; peer leave and data supplier selection can be solely optimized in this structured overlay. Moreover, concerning about the tradeoff between control flow and data transfer efficiency is not needed anymore. The Broadcaster serves as the root and peers form the tree-based overlay layer by layer as the order of fields in GUID. The first layer children are from different ISP, the second are from different city under the same ISP and etc. Therefore, adjacent peers are placed in the same branches of the control tree, which can be called as the Swarm.

Data Mesh is another overlay of Anysee2. It is a mesh-based overlay for media data transfer. Through the Control Tree, peers can easily find their neighbors from the swarm. Moreover, no need of gossip messages makes the startup delay much lower. Why we choose the mesh instead of tree for data transfer is based on the reason that mesh-based overlay does not need some strong parents and have great churn tolerance.

In Anysee2, data buffer is synchronized by Time-Driver method instead of Packet-Driven method, since in latter condition media player would stop playing when network traffic is heavy. Control Tree can be used to broadcast the synchronizing information.

2.2 Bandwidth Estimation and Auto Load Balance Scheduling



```
Algorithm: Process bandwidth estimation
#1  initialize: Max_capacity = 4 * Slots/s * Scheduling Unit
#2. if Last_received == Last_requested
#3.       if Last_requested == Max_capability
#4.             Max_capability += 1 * Slots/s * 1s; exit;
#5. else if Last_received < Last_requested
#6.       if Last_delayed > 0
#7.             if Last_received > 0
#8.                   Max_capability -= 1 * Slots/s * 1s; exit;
#9.             else if Last_delayed == Former_missed
#10.                  Max_capability -= 2 * Slots/s * 1s; exit;
#11.            else if Last_delayed < Former_missed
#12.                  Max_capability -= 3 * Slots/s * 1s; exit;
#13.      else if Last_delayed == 0
#14.            Max_capability -= 4 * Slots/s * 1s; exit;
```

Figure 1. Pseudocode of Bandwidth Estimation.

In a P2P overlay, it is very difficult to achieve load balance for two reasons. One is that bandwidth of Internet varies from time to time. Another is the traditional streaming dissemination makes peers near the source popular and load heavier. Anysee2 uses bandwidth estimation based on the actual data transfer, and an auto load balance scheduling method to achieve load balance.

In every scheduling cycle, the peer will request and adjust the supplier's service capacity by data retrieval. Every partner's service capability is initialized as *Max_capability*. Algorithm in Figure 1 shows the *Max_capability* should be adjusted closer to practical available bandwidth in different conditions.

With estimated bandwidth, load balance scheduler of every peer would request data from best suppliers. The urgent data should be requested directly from Broadcaster to assure the startup QoS. The sequence of common data request is that the scarcest data in the overlay should be requested first, which is the same as that in Coolstreaming. Peers in the network those have spare capability will prefetch further data from Broadcaster with a probability corresponding to the size of the network.

3. SIMULATIONS

Time unit T is introduces as the logical scheduling unit in our simulations and one T is about 2 seconds in real-life. We use BRITE [1] to generate a topology with a set of 1000 router nodes and assign 2-8 terminal hosts to each of them. The join process follows poisson distribution. Assume the delay between two hosts is the link delay along the shortest path (omitting the queue time of each router), the link delay between each router follows uniform distribution automatically set to 4-15 segments per T in BRITE. We set the delay and bandwidth between hosts and routers to 20 segments per T respectively. Each node maintains a 256-segment capacity buffer, and it is connected with 3-8 partners based on the experiment result described in [6]. Assuming that the Broadcaster produces 8 new segments each T, each peer is assigned to run 2000 Ts before stop over 5 times. At last, we collect the simulation logs for our analysis.

Control Tree is introduced not only to achieve peers' contiguity in the overlay, but also to reduce control overhead. The first is obvious due to the construction, and the latter can be proven by comparing with Coolstreaming. The result shows that the control overhead of Anysee2 (about 0.9 percent) much lower than that of the CoolStreaming (about 1.4 percent).

Buffer full percentage is an important metrics of quality of live streaming. We have tested the average buffer full percentage of peers in the overlay which size changes from 50 to 2000. From the result we can conclude the full percentage of Anysee2 (about 55%) is much bigger than that of CoolStreaming (about 35%).

4. CONCLUSION

From the simulation, Anysee2 has been proven effective. Hybrid architecture design reduces the control overhead while keeping great scalability and stability. Auto load balance algorithm make peers guarantees the good QoS.

5. REFERENCES

[1] A. Medina, A. Lakhina, I. Matta, and J. Byers, "*BRITE: An Approach to Universal Topology Generation*", *Proceedings of IEEE MASCOTS'01*, August, 2001, Cincinnati, Ohio, USA.

[2] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "*Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh*", *Proceedings of the 19th ACM SOSP, October 19-22, 2003*, Bolton Landing, NY, USA.

[3] PeerCast. http://www.peercast.org.

[4] S. Saroiu, P. Gummadi, and S. Gribble. "*A measurement study of peer-to-peer file sharing systems*", *Proceedings of Multimedia Computing and Networking (MMCN02)*, January 2002, San Jose, CA, USA.

[5] X. Liao, H. Jin, Y. Liu, L. M. Ni, and D. Deng, "*AnySee: Inter-Overlay Optimization based P2P Live Streaming*", *Proceedings of IEEE INFOCOM*, April 2006, Barcelona, Spain.

[6] X. Zhang, J. Liu, B. Li, and T.-S. P. Yuan, "*CoolStream-ing/DONet: A Data-driven Overlay. Network for Peer-to-Peer Live Media Streaming*", *Proceeding of IEEE INFO-COM'05*, March 2005, Miami, FL, USA.