MoDast: A MoD System Based on P2P Networks

Yu Chen and Qionghai Dai

Broadband Network & Multimedia Research Center,
Graduate School in Shenzhen, Tsinghua University, Shenzhen, Guangdong, China, 518055
{cheny,daiqh}@mail.sz.tsinghua.edu.cn

Abstract. This paper presents MoDCast, a MoD servicing system based on P2P network. The hosts in MoDCast cache the media data receiving from streaming media server and forward them to other hosts. In this way, the MoD servicing system can accommodate more users concurrently, which improve the servicing capacity, and reduce the servicing cost.

1 Introduction

Streaming Media on-Demand (MoD) over Internet is one of the main Internet services, and how to improve the servicing capacity of MoD system is a hot topic. IP multicast improves server and networks' resource utilization remarkably, but hasn't been deploy wildly because of the problem of management, dependability, security, and servicing time, etc [1]. Application layer multicast makes hosts to forward data they receive to other host, and needn't dedicated support provided by lower layer. Typical application layer multicast protocols include Narada [2], Gossamer [3] and NICE [4], etc. Based on NICE, Tran, et al, develop Zigzag [5] for streaming media broadcast with lower control overhead.

This paper presents a MoD servicing system based on P2P networks, MoDCast. The hosts in MoDCast cache the media data receiving from streaming media server and forward them to other hosts that make the MoD system accommodate more users and reduce the servicing cost. Section 2 introduces the model of MoDCast. Section 3 analyzes its caching strategy. Section 4 analyzes its application layer protocol. In the last section, we give out conclusion.

2 The Model of MoDCast

A basic MoDCast system is composed of a WEB server (*ws*), streaming media server (*ms*) that stores a group of clips, and a group of hosts that access those clips. h_1, h_2, h_3 and h_4 are hosts that access clip c_1, and the request of h_1 arriving *ws* firstly. *ws* redirects h_1 to *ms*. Because h_1 can't get service from other hosts, it gets service from *ms* directly. After this, instead of receiving data from *ms*, h_2, h_3 and h_4 will receive media data from h_1. In the same way, h_5 get c_2 from *ms*, and forward data to h_6 and h_8. h_7 receives media data from h_6. Logically, hosts accessing same clip construct a host tree, and *ms*, is the public root (*r*) of all host trees. We call the host sending data parent node, and the host receiving data descendant. Host tree is look like application layer multicast tree, but parent node in host tree could send different data to different descendants at same time, and parent node in application layer multicast tree must send same data to different descendants at same time.

The hosts in a host tree could belong to different sub-network. We confine a basic MoDCast system in an autonomous system, such as a campus network, or a network belong to a ISP to control end-to-end delay between hosts, and reduce transmission overhead on backbone.

Web server is the rendezvous of host tree, but not a part of the tree. Besides redirecting hosts to streaming media server, wed server maintains and issues the basic information of the clips, such as clip's introduction, price, etc.

3 Caching Strategy

Caching several seconds or one minute of data makes host be able to tolerate the variation of QoS on network and server, and playback the clip smoothly, but can't satisfy the requirement of MoDCast. Due to the unpredictability of the request' arrival time, the duration time of media data cached on host must long enough to increase the utilization of cached data. Obviously, the longer the data stay on host, the higher the probability of the data used by other hosts. The duration time of cached data is related to the host's cache space. The larger the cache space is, the longer the data's duration time is, but large cache space will occupy host's resource too much. Besides this, MoDCast system hopes the hosts to run as a parent node as long as possible to keep the cached data available to other hosts. But, host has the trend of leaving host tree as soon as possible at the end of playback. So, we design a cache strategy to take advantage of hosts' shared resource efficiently.

Let the duration time of clip is f units, and host can cache l units, $f>l$. Host caches and forwards the data received from parent node. Due to $f>l$, after cache is filled, the oldest data in cache space is given up to reclaim resource for the new arrival data. Let cache space of host h_i is l, and h_i begin to receive data from server at t. Host h_j choose h_i as parent node, and begin to get data from h_i at t'. If $t'-t\leq l$, h_j can get all of the data h_i caches.

Normally, if the interval between the arrival times of two consecutive requests that accessing clip c is always smaller than the duration time that media data cached on a host, one host tree can satisfy all of the users that access clip c. Though it's impossible to assure above statement. Making the host tree as deep as possible can make use of the resource of host sufficiently. But along with the increase of depth of the tree, the number of hosts influenced by the failure of intermediate host in host tree will increase remarkably. So, we define a constant time l for host tree. An host tree created at t when a host become the immediate descendant of MoD server, and other host can only join this tree before $t+l-a$, a is the interval between the time host send request and the time host join host tree. $l-a$ is the duration time that media data cached on a host. MoD server records the deadline of each host tree. For every clip, at any time, at most one host tree available for hosts to join. In this way, we can control the weak consistency of the host tree.

4 Application Layer Protocol for MoDCast

When request is redirected from WEB server to MoD server, MoD server will find whether the user sending this request can join a host tree. If can't, MoD server create a new tree for the user. Hosts in a host tree can be divided into two classes, peer host and selfish host. Peer host forwards the cached data to other host, and selfish host only receive data. New host chooses its parent node from peer hosts in the host tree.

A host tree has several layers. MoD Server is the unique node at the first layer, and second layer only has one host, *r*. The immediate descendants of *r* are composed of layer three, H_{L3}, and the immediate descendants of H_{L3} are composed of H_{L4}, etc.

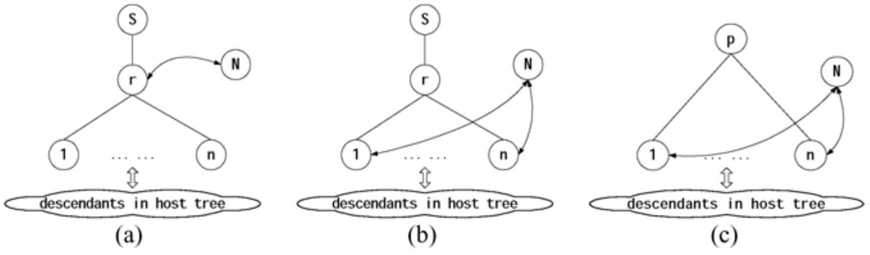

Fig. 1. Join Operation

When host h_n, wants to join in a host tree, server will send the address of *r* to h_n. h_n contacts with *r* (in figure 1a). If *r* has enough shared resource to connect with h_n, h_n become *r*'s immediate descendant. Otherwise, *r* sends the member address of H_{L3} to h_n. h_n probes the distance between the member of H_{L3} and itself, and chooses the host closest to itself as parent node (in figure 1b). Let *p* is chosen, but *p* hasn't enough shared resource, then h_n will choose parent node from *p*'s immediate descendant in H_{L4} (in figure 1c). h_n repeats above operation until find a suitable parent node, or exit because couldn't find parent node. If there are two or more hosts have same distance to h_n, h_n choose the host with lower overhead as parent node.

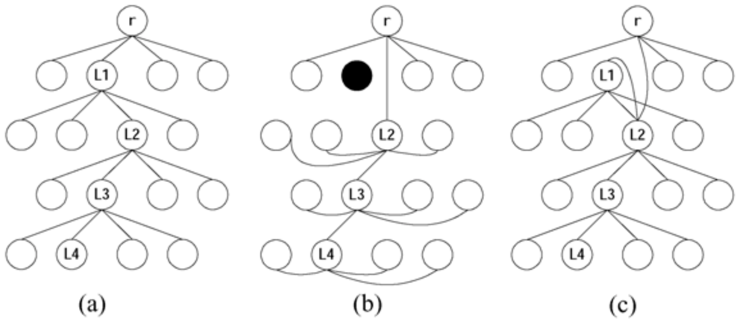

Fig. 2. A example of recovery protocol

We use an example to explain MoDCast's recovery protocol. Figure 2a is a host tree in normal state; *L1*, *L2*, *L3* and *L4* are mediate host in the host tree. *r* is the only host in H_{L2}, and receive data from MoD server. In figure 4b, *L1* is failed. *L2* informs its sibling hosts that it can't get media data, and its sibling response with the same message. So, a new parent node should be selected from *L1*'s immediate descendants to replace *L1*. The new parent host should be the first host that chooses *L1* as parent host, because the data cached in this host can satisfy all other siblings' requirement. In figure 2b, *L2* is the new parent node, and receives media data from *r* directly. So, *L2* is lifted from H_{L4} to H_{L3}. *L2*'s siblings become its descendants, and only one of *L2*'s original descendants continue connect with *L2*. In this example is *L3*, others become *L3*'s descendants.

When *L1* can't receive data from *r*, *L1* exchange information with its siblings to find out whether its parent node is failure or the link to its parent node is failure. If other hosts can still receive data from parent node, the link between *L1* and *r* is failed. *L1* chooses a relay node in its immediate descendants to recover from link failure. As shown in figure 2c, *L2* is the relay node between *L1* and *r*, which receives the date needed by *L1* from *r*, and forward them to *L1*. At the same time, *L2* gets data from *L1* to playback.

5 Conclusions and Future Works

In this paper, we introduce MoDCast's architecture, and analyze its cache strategy and host tree protocol. We have done some simulation about MoDCast, and the results of simulation demonstrate that, when working in an autonomous system, MoDCast can improve the servicing capability significantly with low control overload. Besides this, we can conclude from simulation that, 1) sharing peer host's resource can reduce the reject ratio, and improve resource utilization, 2) cache more data can improve system's accommodation, but at the cost of increasing hosts' overhead, 3) the number of selfish hosts hasn't significant influence on system performance.

References

1. C. Diot, B. N. Levine and B. Lyles, et al, Deployment Issues for the IP Multicast Service and Architecture. IEEE Network, Jan 2000.
2. Y. Chu, S. Rao, and H. Zhang. A Case For End System Multicast. In Proc. ACM Sigmetrics, June 2000.
3. Y.CHAWATHE, S. MCCANNE and E. BREWER, An architecture for internet content distribution as an infrastructure service. http://www.cs.berkeley.edu/ yatin/papers, 2000.
4. S. Banerjee, B. Banerjee and C. Kommareddy, Scalable Application Layer Multicast, ACM SIGCOMM 2002.
5. D. A. Tran, K. A. Hua and T. T. Do, Zigzag: An Efficient Peer-to-Peer Scheme for Media Streaming, IEEE INFOCOM, 2003.