A Video Broadcast Architecture with Server Placement Programming

Lei He[1], Xiangjie Ma[2], Weili Zhang[1], Yunfei Guo[2], and Wenbo Liu[2]

China National Digital Switching System Engineering and Technology Research Center (NDSC), P.O. 1001, 450002 Zhengzhou, China
{hl.helei,clairezwl}@gmail.com,
{mxj,gyf,lwb}@mail.ndsc.com.cn

Abstract. We propose a hybrid architecture MTreeTV to support fast channel switching. MTreeTV combines the use of P2P networks with dedicated streaming servers, and was proposed to build on the advantages of both P2P and CDN paradigms. We study the placement of the servers with constraints on the client to server paths and evaluate the effect of the server parameters. Through analysis and simulation, we show that MTreeTV supports fast channel switching (<4s).*

Keywords: Internet video broadcast; mesh; fast channel switching; locality; pastry.

1 Introduction

With the widespread penetration of broadband accesses, multimedia services are getting increasingly popular among users and have contributed to a significant amount of today's Internet traffic. The sparse deployment of IP Multicast [1] have limited broadcast to only a subset of Internet content publishers. Both CDN-based and P2P based architectures have their advantages and disadvantages, and each architecture alone does not provide a cost-effective and scalable solution to streaming media distribution.

There has been considerable work in the area of peer-to-peer live-video streaming [3]-[11]. There are two key drivers making the approach attractive. First, such technology does not require support from Internet routers and network infrastructure, and consequently is extremely cost-effective and easy to deploy. Second, in such a technology, a participant that tunes into a broadcast is not only downloading a video stream, but also uploading it to other participants watching the program. Consequently, such an approach has the potential to scale with group size. However, the P2P architecture has the performance problem and suffers from long setup and source to end delay. e.g., CoolStreaming [7] is one of the first data-driven systems, and its setup delay is longer than 60 seconds. PPLive [8], a popular commercial peer-to-peer streaming system, its source to end delay is between 20 and 60 seconds.

* This work is partially supported by National Basic Research Program of China (973 Program) with No.2007CB307102.

P. Primet et al. (Eds.): GridNets 2008, LNICST 2, pp. 130 – 137, 2009.
© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2009

To reduce the delay, we propose a novel hybrid architecture MTreeTV that combines the use of P2P networks with dedicated streaming servers, and was proposed to build on the advantages of both paradigms. We emphasize on the effect of placing servers in overlay networks. Through analysis and simulation, we show that MTreeTV supports fast channel switching (<4s), it can provide high playback continuity (>98%) with little control overload (<2%).

The remainder of this article is organized as follows. In Section 2, we present the overview of hybrid architecture MTreeTV. We solve the server placement problem in Section 3. In Section 4, we evaluate the performance of MTreeTV. Finally, Section 5 concludes the paper and offers potential future research directions.

2 Hybrid Architectures

We consider a live video streaming system using overlay multicast. The video stream, originated from the source to multiple tree roots and then the other nodes, is divided into equal-length segments. The clients that are interested in the video form an overlay mesh by joining multiple trees simultaneously, and each overlay node, except for the source, acts both as a receiver and, if needed, an application-layer relay that forwards data segments. In addition, these nodes can join and leave the overlay at will, or crash without notification.

Our hybrid video broadcast architecture MTreeTV is shown in Fig.1. The details of MTreeTV overlay construction and scheduling algorithm is described in ref. [12]. In this architecture, the two streaming technologies complement each other: The video stream will be first pushed to streaming servers; then clients that are interested in the video form a P2P overlay and pull the streaming data from the near peers.

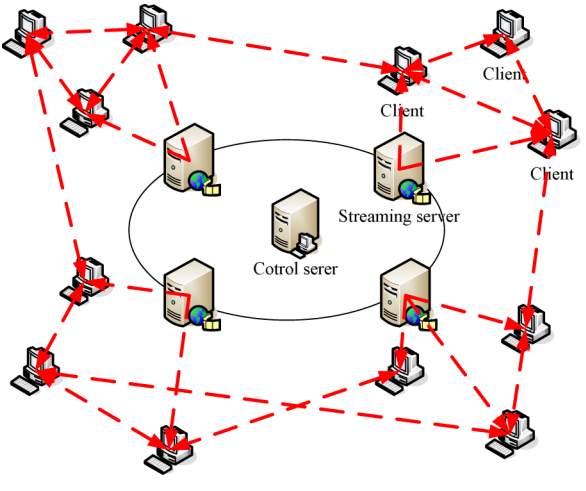

Fig. 1. The hybrid video broadcast architecture MTreeTV. Dotted lines show the directions of stream data transmission.

The main entities of MTreeTV are:

- Control Server. The role of the Control server is to maintain client/channel information. When a client starts, it first registers to the control server, then it retrieves the information of channel/streaming-server from the control server; finally, it joins the overlay and gets the video playing.
- Streaming Server. It holds a copy of all channels and is responsible for streaming. It is also responsible for processing the data requests of direct connected clients. We assume a zero downtime for the servers.
- Client. The set of user machines participating in the streaming system. It essentially constructs a mesh out of the overlay nodes, with each node having a small set of neighbors to exchange data.

We compared the performance of MTreeTV with CoolStreaming and the Tree. Tree was a single tree of MTreeTV. We used several metrics to evaluate the performance. **Setup delay** is the time from the user first receives a segment to the video is visible. **Source to end delay** is the delay between the content originator and the receiver, also known as playback delay. **Playback continuity** is the percentage of received data packets before the deadline. **Control overhead** is Control message volume/Video traffic volume at each node. **Failure percent** is the ratio of the failure node number to all node number in an time interval T.

3 The Server Placement Problem

In this section, we study the placement of the streaming servers with constraints on the client to server paths, reflecting the quality of service within the regional access networks. We envision that this imposition of service quality constraints on server to client paths is essential for the newer network services to achieve better service quality in order to attract and retain customers.

Given the network and their estimated service parameters, how many servers are needed and where should an overlay service provider locate them to ensure a desired service quality to all its clients? To answer the above question, we transform the placement problem to the set cover problem and solve it using 0-1 programming, linear programming (LP) relaxation and greedy heuristics.

3.1 Formal Definitions

The server placement maps to the set cover problem as follows: an element corresponds to the network location of an client; the base element set contains all clients; a set represents a potential server placement at one of the router locations, each set includes all the network locations that are within the service range from the server location represented by the same set. Given a base set of elements and a family of sets that are subsets of this base set, find the minimum number of sets such that their union includes all elements in the base set [10]. In Fig.2, the points are denoted by numbers and the sets of points are denoted by names. The goal is to "cover" all the points by choosing the minimum number of sets, e.g. we can choose S1, S3 and S4.

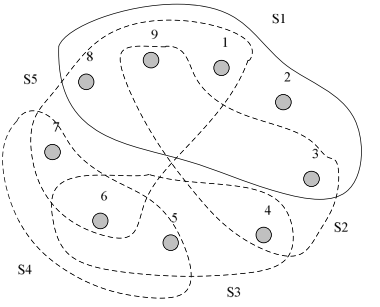

Fig. 2. A set covering instance

Given a set of networks and their interconnections, as well as a specific routing policy, we can compute an end-to-end path for every pair of nodes in the networks. For each node n_i, we compute a set S_i which includes all the nodes reachable from n_i within the server service range C. Let $S_1, S_2, \ldots, S_m$ be all the sets computed. The LP formulation of the set cover problem is:

$$\text{Objective: } \min \sum_{j=1}^{j=m} x_j \tag{1}$$

$$\text{Subject to: } \sum_{j=1}^{j=m} a_{ij} x_j \geq 1, \text{ for i=1 … n} \tag{2}$$

$$x_j \in \{0,1\}$$

where x_j is the selection variable of S_j , a_{ij} is 1 if $n_i \in S_j$ and 0 otherwise. A variation of the problem is to allow one primary and one backup server to cover each node. A backup server can cover twice the distance of the primary server. Let T1, T2, … , Tm be all the backup sets, and $b_{ij} = 1$ if $n_i \in T_j$ and 0 otherwise. The objective here is still to minimize the number of selected sets but with the additional constraints of:

$$\sum_{j=1}^{j=m} b_{ij} x_j \geq 2, \text{ for i=1 … n} \tag{3}$$

Since all nodes in the primary set are also in the backup set centered at the same server, $b_{ij} = 1$ if $a_{ij} = 1$; but we can not have a primary server also service the same node as the backup server — the constraint in (3) ensures the selection of a different server as the backup.

3.2 LP Based Algorithms

The above formulation can be solved by the 0-1 programming (also call binary integer programming). 0-1 programming is a special type of integer programming, and x_j is 0 or 1. An efficient method to solve Integer programming is the branch-and-bound

method. Matlab provides functions to solve the 0-1 integer programming problem and the linear programming problem.

The solution for the 0-1 programming naturally provides a lower bound = $\sum_j x_j^*$ for the set cover problem, since it is an optimal solution and the 0-1 programming is a super set of the set cover problem. We will use this lower bound to evaluate the quality of the solutions produced by other algorithms.

The above formulation can also be approximated by first solving the LP relaxation of the problem optimally and then rounding the fractional values to integers. The LP relaxation of the problem is to allow the selection variables xj to take fractional values between [0, 1]. The LP relaxation can be solved in polynomial time and the rounding can be done in O(n). Reference [13] introduced a rounding algorithm which is a p-approximation algorithm, where $p = \max_i \{\sum_j a_{ij}\}$ is the maximum number of sets covering an element. We refer to the rounding algorithm in [14] as the fixed-rounding (FR) algorithm:

1. Solve the LP relaxation of the problem and let $\{x_j^*\}$ be the optimal solution;
2. Output $sets = \{ j \mid x_j > \frac{1}{p} \}$.

It is easy to see that the FR algorithms can still have redundant sets in the final solution. To prune these unnecessary extra sets, we use a simple pruning algorithm as the final step to complete the set selection:

1. Sort all selected sets in increasing order of set size;
2. Starting from the smallest set, check if it can be removed without leaving any of its nodes uncovered.
3. Repeat Step 2 until all sets are checked.

3.3 Greedy Heuristics

The basic greedy attribute of the algorithm is to select a set at every step that contains the maximum number of uncovered elements. For the backup problem variant, we extend the algorithm by treating any node that has not satisfied the constraints of (2) and (3) as equally uncovered. At each step, we select a set that has the largest number of remaining uncovered nodes and repeat till there are no more uncovered nodes. The greedy heuristic is:

1. Sort all unselected sets in decreasing order of uncovered node number;
2. Picking up the largest set, and changing the elements' state of the set to covered;
3. Repeat Step 1 and 2 until all nodes are covered.

Using simulation, we show that this rounding technique and the greedy heuristic approach the lower bound very closely; in fact, it reaches the lower bound for many configurations. Meanwhile, the greedy heuristic also provides good performance in all instances with significantly less computation complexity.

4 Simulation Results

We have performed a series of experiments on our segment-level, event-driven simulator. Our simulation topologies has 100 routers and 1000 hosts, it is generated by the GT-ITM topology generator using the transit-stub model. Message delays are determined using the resulting distance metric. 1000 end nodes that were randomly assigned to routers in the core of each topology with uniform probability. Each end system was directly attached by a LAN link to its assigned router. The delay of each LAN link was 1ms and the average delay of core links (computed by GT-ITM) was approximately 216ms.

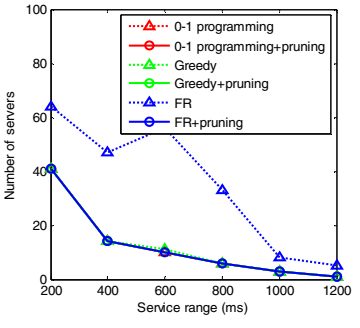

Fig. 3. Performance comparison of the FR, Greedy and 0-1 programming algorithms with variation on server service range.

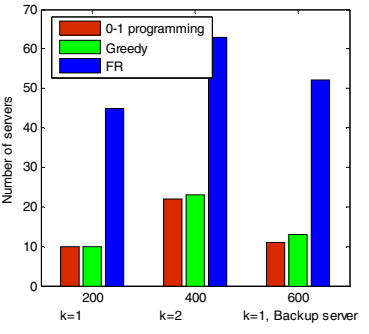

Fig. 4. Variation on the placement strategy (service range =500ms)

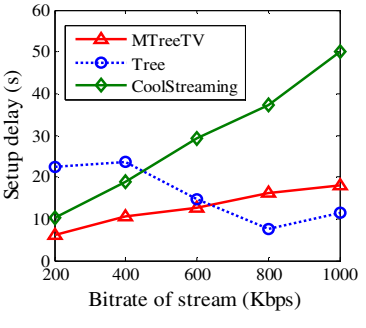

Fig. 5. Setup delay as a function of the bitrate of stream.

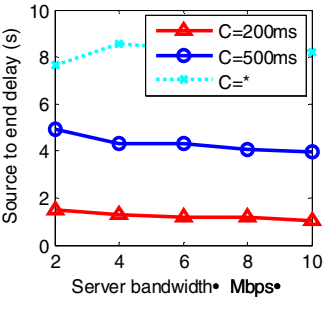

Fig. 6. Souce to end delay as a function of the time of server bandwidth and service range.

We compare the greedy algorithm with the fixed rounding (FR) algorithm. The results are further compared with the lower bound obtained as the optimal solution from the 0-1 programming method. The programming and greedy algorithm solver we used, is called Matlab7.0.1 which is an advanced computation and simulation package. The simulations reveal the impact of different server and client parameters (service range, server bandwidth, client buffer time) on the system performance. All the nodes join the overlay initially, and the tree roots begin streaming from the time

zero. We used different random join sequence for each run. All results are the average values of 10 times run.

In this section, we compare the fixed rounding (FR) algorithm and the greedy algorithm with the 0-1 programming method. Given the network, we also use the previous metrics to evaluate the effect of the server parameters (e.g., service range, bandwidth). With this routing policy, we can compute a routing table for each node n_i and the cost of each routing path $c(n_i, n_j)$, which is the sum of the hop distances along the path. For each router r_i, we compute a set S_i which includes all the nodes reachable from r_i within the routing cost of C, C is the service range of this server. This is to say that if a server is placed at the location of router r_i, then all the nodes within this service range can be serviced by this server.

Fig.3 shows the number of required servers when varying service range. In general, both the FR (with pruning) and the greedy algorithm perform closely with the lower bound. With increasing service range, the number of servers needed drops significantly. Fig.4 shows the performance against the different server placement strategies. We perform simulations on the following three scenarios: (a) k = 1, with only one primary server required to cover each node; (b) k = 2, with two primary server required to cover each node; (c) k = 1 and requiring one backup server. The backup server range is twice that of the primary server for both networks. Fig.4 shows that two primary servers will significantly increase the number of servers. And the addition of the backup servers only increases the number of total servers slightly.

MTreeTV has the shortest source to end delay (From Fig. 5~6). Fig. 5 shows that the high bitrate of stream will increase the setup Delay. When the number of nodes is 200, B=10Mbps, buffertime=16s, C=200ms, the SETUP delay of MTreeTV is about 4s (Fig. 6). With the optimization of hybrid architecture optimize, the setup delay of MTreeTV is only 1/4 of CoolStreaming's. From the simulation results, MTreeTV also gets high playback continuity (>98%) and little control overload (<2%).

5 Conclusion and Future Work

In the very near future, video may become the dominant type of traffic over the Internet, dwarfing other types of traffic. Among the three video distribution modes: broadcast, on-demand streaming, and file download, broadcast is the most challenging to support due to the strong scalability and performance requirements.

In conclusion, MTreeTV as a whole outperforms other systems under various environments. Through analysis and simulation, we show that MTreeTV can support fast channel switching, it provide high playback continuity with little control overload.

As part of our future work, we plan to further explore this class of application. We will look at the extreme peer dynamics or flash crowd, access bandwidth scarce regimes, incentives and fairness. We are going to simulate more system behaviors to further improve the system's performance.

Acknowledgment

Lei He would like to thank Prof. Julong Lan, Prof. Dongnian Cheng of NDSC for their helpful comments.

References

[1] Deering, S.: Multicast Routing in Internetworks and Extended LANs. In: Proceedings of the ACM SIGCOMM (August 1988)

[2] Danzig, P., et al.: A case for caching file objects inside internetworks. In: Proc. ACM SIGCOMM Conference (SIGCOMM 1993), pp. 239–248 (September 1993)

[3] Banerjee, S., Bhattacharjee, B., Kommareddy, C.: Scalable applicationlayer multicast. In: Proc. ACM SIGCOMM 2002, Pittsburgh, PA (August 2002)

[4] Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstronand, A., Singh, A.: SplitStream: High-bandwidth multicast in cooperative environments. In: Proc. ACM SOSP 2003, New York, USA (October 2003)

[5] Chu, Y.h., et al.: A Case for End System Multicast. In: Proc. 2000 ACM SIGMETRICS Int'l. Conf. Measurement and Modeling of Comp. Sys. (2000)

[6] Rowstron, A., et al.: Scribe: The design of a large-scale event notification infrastructure (June 2001)

[7] Zhang, X.: CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In: Proc. INFOCOM 2005, Miami, FL, USA (March 2005)

[8] Hei, X., et al.: A Measurement Study of a Large-Scale P2P IPTV System. Tech. rep., Dept. of Comp. and Info. Sci., Polytechnic University (2006)

[9] Venkataraman, V., Francis, P., Calandrino, J.: ChunkySpread: Multitree unstructured peer-to-peer multicast. In: Proc. The 5th International Workshop on Peer-to-Peer Systems (February 2006)

[10] Shi, S., Turner, J.: Placing Servers in Overlay Networks. In: SPETS (July 2002)

[11] "Bittorrent"[EB/OL], `http://www.bittorrent.com`

[12] Lei, H., et al.: A Peer-to-Peer Internet Video Broadcast System Utilizing the Locality Properties. In: HPCC, China (September 2008)

[13] Shi, S., Turner, J.: Placing Servers in Overlay Networks. In: SPETS (July 2002)

[14] Hochbaum, D.: Approximation Algorihtms for NP-Hard Problems. Brooks/Cole Publishing Co. (1996)