2012 4th International Conference on Computer Research and Development
IPCSIT vol.39 (2012) © (2012) IACSIT Press, Singapore

Overlay Optimization for Mesh-based Peer-to-peer Streaming Applications

Yusuo Hu

Gooogle Inc., Mountain View, CA, U.S.A.

yusuo.hu@gmail.com

Abstract. Mesh-based peer-to-peer (P2P) streaming has proven to be a robust and scalable way for delivering live video streams over the Internet. In this paper, we investigate the overlay optimization problem for mesh-based peer-to-peer streaming. By modeling the peer-to-peer overlay as a directed graph, we propose a new overlay optimization method that imposes a strict partial order on the P2P overlay. The partial order can be defined according to different capability metrics of the heterogeneous peers. We have also designed overlay construction and optimization protocols that can promote large capability nodes to higher levels while keeping the overlay robust and efficient. Experimental results show that our optimization method can efficiently promote the high performance peers and improve efficiency in the P2P overlays. Compared with the non-optimized case, the P2P streaming applications that use our method have reduced average latency by 25-40% and increased acceptance rate up to 125%.

Keywords: peer-to-peer streaming; overlay optimization; mesh; directed graph; ordering

1. Introduction

Recently, P2P networking has proven to be a promising method for delivering live video streams to a large number of clients over the Internet. Much effort has been devoted to the research and development of the live P2P streaming systems. Some deployed systems have shown that the P2P systems can scale to millions of online users in a single session [9][11][16][17].

In the design of the P2P streaming systems, one of the most important problems is how to organize the dynamic and heterogeneous peers into an efficient overlay. The structure and behavior of the overlay greatly affect the efficiency of many modules such as peer discovery, data transmission delay, and ultimately the final streaming performance perceived by end users.

In this paper, we propose a flexible optimization method by imposing a strict partial order on the P2P overlay. We formulate the mesh-based P2P overlay as a directed graph and design a set of protocols to optimize the P2P overlay in a proactive way. To verify the efficiency and robustness of our protocols, we conduct extensive experiments on a packet-level simulator. We find that the performance of P2P streaming systems can be significantly improved by incorporating some global optimization procedures, especially when there are not enough resources to satisfy all the participants. And our optimization protocols can keep the overlay compact and efficient. For live streaming applications, our method can reduce the delay by 25-40% and increase the acceptance rate up to 125%.

The paper is organized as follows: We first propose our system model, which is a description of the current overlay in section II. In sections III and IV, we present our overlay construction and optimization algorithms. We also describe some protocols and implementation there. In section V, we verify our optimization method by simulation and conclude in section VI.

2. Model and Definitions

2.1 Directed Graph Model

We use a directed graph model to describe our P2P streaming overlay. The graph consists of the streaming source node r and a number of other nodes representing the peers that participate in the same P2P streaming session.

A directed arc from node A to node B, denoted by $A \rightarrow B$, represents that the peer A has peer B as its children. The parent node periodically advertises the indication of the available packets, namely buffer map, to all its children nodes. And the child nodes schedule the missing packets to be pulled from its parents through some packet scheduling strategy.

We do not allow duplicate arc with the same direction between two nodes. However, there may be two oppositely directed arcs between two nodes. If two nodes are both the parent/child of each other, they will exchange their buffer maps and packets.

To describe the different capacity of the peers, we set some constraints on each node's incoming and outgoing degrees. Each node has a maximal out degree, denoted by $d_o(p)$ which stands for the maximal number of child nodes that it can support. We call node p is *underloaded* if p's out degree is smaller than $d_o(p)$ and *overloaded* if p's out degree is larger than $d_o(p)+c$, where c is a small constant. Meanwhile, each node has a desired in-degree, denoted by $d_i(p)$, which stands for the number of parents it tries to find. We assume that all nodes have a constant desired in-degree, except for the source node whose desired in-degree is zero.

2.2 Hop Count

We define the *Hop Count* of a node p, $H(p)$, to be p's shortest distance to the source node on the directed graph. The hop count of the source node r is always 0, i.e.

$$H(r) = 0;$$

$$H(p) = \min_{q:q \rightarrow p} H(q) + 1, \forall p \neq r;$$

The hop count represents the directed distance from the source peer to the node, i.e. the length of the shortest path which consists of a number of consecutive directed arcs from the source to the peer.

Sometimes we use the concept of "level" to describe the peer's position in the overlay. We say a peer with a smaller hop count is located at a higher level in the overlay.

2.3 Ordering

To compare the capability of the peers, we introduce a strict partial order $\succ$ on the directed graph. Here, a "strict partial order" means that $\succ$ is a binary relation which is

irreflexive: $\neg\, p \succ p$ and, transitive: $p \succ q,\ q \succ r \rightarrow p \succ r$

We use the strict partial order to describe whether a node is superior to another node with respect to their streaming capability. The order can be defined according to different application metrics. E.g., the order may be defined as: "Given two peers A and B, $A \succ B$ if and only if A has larger outgoing bandwidth than B and A's connectivity is not worse than B".

When comparing two the capability of two peers, sometimes it is unnecessary (or impossible) to judge which one is superior to the other. Therefore, it is more reasonable to define a partial order instead of a total order on the graph.

The goal of the overlay optimization strategy is to organize the overlay so that the nodes with larger capability will be located at a higher level of the streaming overlay. To do this, we try to align the order with the hop count, i.e. if node p and q satisfy $p \succ q$, then $H(p)$ should be larger than $H(q)$. If all the nodes in the directed graph satisfy the above condition, we call such a graph is a *well-ordered* graph.

We first need to ensure the ordering does not conflict with the general goal of P2P streaming applications. We call a directed P2P graph is *satisfied*, if each node has get its desired incoming degree and there is not any overloaded graph. It is easy to see that a satisfied directed P2P graph can be turned into another satisfied well-ordered graph if the partial order satisfies the following condition:

$\forall\, p, q$, if $p \succ q$, then $d_o(p) > d_o(q)$

The above statement means that the partial order should be consistent with the ascending order of the maximum out-degree. Based on this principle, we can define several kinds of strict partial order and will compare their performance in the experimental part.

3. P2P Streaming System

In this section, we describe the mesh-based P2P streaming system that we have implemented. The basic components are similar to most of the other mesh-based P2P streaming implementations, except that we have leveraged the hop count information in the peer selection. Our experimental results show that the hop-count based peer selection can effectively keep the overlay compact and prevent the overlay from becoming so-called "long-shaped overlay".

3.1 Membership Protocol

We adopt a push-pull gossip protocol [13] to maintain a unified random partial view on each peer. In this protocol, each peer periodically picks a node by random and sends it the partial view along with its own node record. The contacted gossip target will reply with its partial view and its own record. After receiving the gossip message, both peers update their partial views by merging the node records. They may truncate their partial views to keep the view size below a constant threshold.

We assume that a well-known bootstrapping node is deployed to help new nodes to join the overlay. When a new peer joins the overlay, it first contacts the bootstrapping node to start the gossiping process. It also requests the bootstrapping to get a number of under-loaded nodes. The new peer then starts to contact these under-loaded nodes to find enough parents. Later, the peer will rely on its own partial view to find under-loaded nodes and select parent candidates from them.

3.2 Hop count calculation

The hop count of each peer represents the length of the shortest path from the source peer to it on the directed graph. The hop count of the source peer is always 0. A new peer first initializes its hop count to be ∞ (which can be represented by a very large number). Then the hop count is calculated by monitoring its parent nodes' status. Each peer tracks its parent with the minimal hop count H_m and set its own hop count to $H_m + 1$. If $H_m = \infty$, it will set its own hop count to ∞ too.

When a peer finds its own hop count changed, it will notify all its child peers. The child node will re-calculate its hop count when it receives notification, or when it adds/removes a parent node.

In very rare cases, some partitioned peers may form separated loops and cause the hop count to increase infinitely. To prevent this situation, we set a threshold for the maximal hop count value. Any time a peer observes that its hop count exceeds this threshold, it will set its hop count to ∞. However, we did not observe any occurrence of this phenomenon in our experiments.

3.3 Parent Selection Policy

Each peer will try to find enough parent nodes to satisfy its desired in-degree. When joining the overlay, the peer will query the bootstrapping node to get some under-loaded peers as its parent candidates. After joining the overlay, the peer periodically picks the under-loaded nodes from its own partial view and caches them as parent candidates.

Among the collected parent candidates, a peer needs to decide which nodes it should contact. Here we use a parent selection policy based on the hop counts: The peer will select the parent nodes in the ascending order of their hop counts, i.e. the peer prefers choosing those nodes that are nearer to the source node on the directed graph. The peer then sends a subscribing request to the selected parent candidate. If the contacted peer is really under-loaded at that time, it will accept the peer as a child node. Otherwise, it declines the request and sends back its current status. The requesting peer will continue to contact other parent candidates until it has found enough parents.

4. Overlay Optimization

In this section, we propose a distributed optimization algorithm that proactively optimizes the P2P overlay based on a defined strict partial order on the directed graph.

4.1 Protocol

We assume that the peers in the overlay are cooperative, i.e. no peer lies on its own status information and all the peers follow the same optimization policy. We also assume that, the node status, which is gossiped in the membership protocol, has conveyed enough information of the peer for capability comparison with others.

Suppose *p* is a peer that has joined the overlay. It periodically checks its membership cache and tries to find the peer *q* (except the source node) that satisfies the following condition:

(1) $H(p) > \mathrm{H}(q)$
(2) $p \succ q$

The above inequalities show that the overlay order between *p* and *q* does not align with the desired partial order. Thus, *p* should be promoted to a level which is higher than that of *q*.

If *p* finds more than one candidate that satisfies the above conditions, it will choose the node with the minimal hop count. If there are more than one candidate that have the same hop count, *p* will choose the one which is not superior to any other node in the partial order.

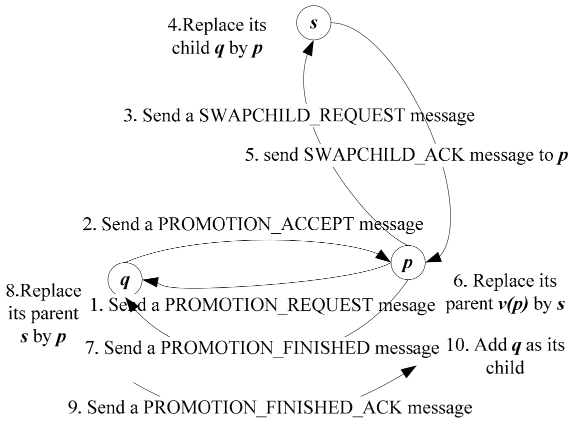
4.Replace its child **q** by **p**
s
3. Send a SWAPCHILD_REQUEST message
5. send SWAPCHILD_ACK message to **p**
2. Send a PROMOTION_ACCEPT message
p
q
8.Replace its parent **s** by **p**
1. Send a PROMOTION_REQUEST mesage
6. Replace its parent **v(p)** by **s**
7. Send a PROMOTION_FINISHED message
10. Add **q** as its child
9. Send a PROMOTION_FINISHED_ACK message

Figure 1: Promotion Procedure

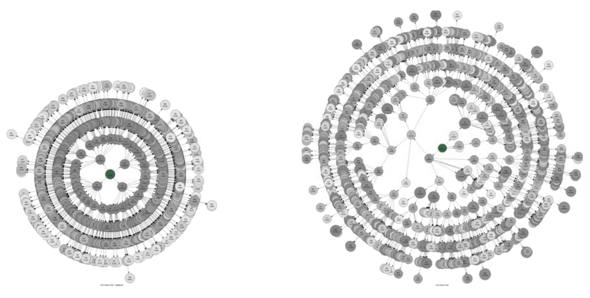

Figure 2: Snapshots of the peer-to-peer overlay graph. (The darker circles represent the nodes with larger out degree)

To promote itself to a higher level, the node *p* first sends a PROMOTION_REQUEST message to *q*, which includes *p*'s current status. After receiving the request, *q* will compare *p*'s status with its current status and judges whether the promotion condition still holds. If not, *q* will decline the request and sends back its current status. Otherwise, it accepts the request and sends back the address and the status of the node *s*, which is *q*'s parent node with the smallest hop count.

If the promotion request is accepted, *p* will connect to the peer *s* and send a SWAPCHILD_REQUEST to *s*. Upon receiving the SWAPCHILD_REQUEST from *p*, node *s* replace the child *q* by *p* and sends back a SWAPCHILD_ACK message to *p*.

After receiving the SWAPCHILD_ACK message, the node *p* adds *s* as its parent. If *p*'s parent number exceeds the required incoming degree, *p* will select its parent *v(p)* that has the largest hop count and unsubscribe from it. At the same time, the node *p* sends a PROMOTION_FINISHED message to *q*. When *q* receives the message, it replaces its parent node *s* by *p*. Finally, *q* sends a PROMOTION_FINISHED_ACK message to *p* and *p* adds *q* as its child after receival. The procedure of a successful promotion procedure is illustrated in Fig. 1.

During the promotion procedure, nodes may leave the overlay and there may be message loss or connectivity problem that breaks the procedure. Thus, each request-acknowledgement message pair is monitored with a time-out value. Meanwhile, each node will check its own status. If a node finds itself overloaded, it will discard the child node that has the largest hop count.

5. Experimental Study

We implemented our streaming protocols and simulated them on a packet-level simulator[26]. For synthesized data, we use BRITE [25] to generate hierarchical network topologies and model peer behavior as Weibull distributions.

In the streaming experiments, we assume that each peer knows its own incoming and outgoing bandwidth. The video bit-rate is set to 256K bps. We simulate a high dynamic case, where the average session time of the peers is only 80 seconds. The peer's lifetime is modeled as lognormal distribution. We simulated the situation of 1000 peers joining and leaving the overlay within 300 seconds.

To illustrate the efficiency of our optimization procedure more visually, we draw the snapshots of the P2P overlays with and without the optimization procedure by Graphviz [14] in Fig. 2. At the sampled time we draw the breadth-first spanning tree of the directed graph which is rooted at the source node. In the figure, the source node is placed in the center and the other nodes are placed on the circles whose distances are proportional to their hop counts.

We observe that the optimized overlay has a much short diameter than the non-optimized one. And, as we expected, most of the high-capability nodes have been placed at high levels in the graph. Such an optimized overlay performs much better than the non-optimized random overlay, as we will show in the next section.

To investigate how different ordering criteria affect the streaming performance of the P2P overlay, we choose three representative metrics to define the partial order: (i) the outgoing degree of the peer, denoted by $O(.)$, represents the outgoing bandwidth level of the peer; (ii)the peer's life-time, denoted by $L(.)$; (iii)the RTT latency between the peer and the source node, denoted by $R(.)$.

The following three ordering strategies are used in our experiments:

- Order *A*: Bandwidth only
  $p \succ q$, if and only if $O(p) > O(q)$
- Order *B*: Bandwidth and lifetime
  $p \succ q$, if and only if $O(p) > O(q)$ and $(1+\alpha)L(p)>L(q)$.

Here $\alpha(>0)$ is a parameter to ensure that a late arriving peer with a larger bandwidth can surpass an early arriving peer only if its life-time is large enough.

- Order *C*: Bandwidth and latency
  $p \succ q$, if and only if $O(p) > O(q)$ and $R(p) < R(q) + d$.

Here d is a relaxation constant for smoothing out the small differences in latency measurement.

We describe the bandwidth resource of the P2P streaming session by the resource index, which is defined as the ratio of the total uploading bandwidth to the total required incoming bandwidth. We change the resource index from low to high and run the simulation to compare the performance. Under each setting, we run the simulation for five times to reduce randomness.

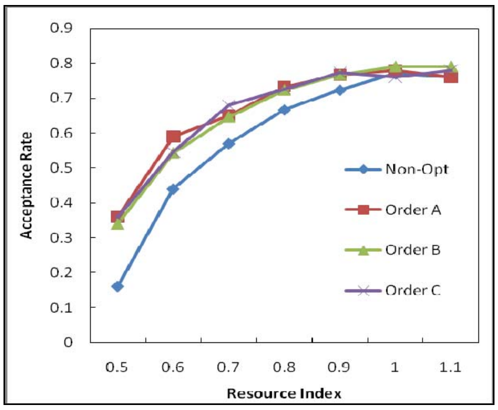

Figure 3: Acceptance Rate

The first performance metric is the service acceptance rate, which is defined as the ratio of the successful request number to the total request number. Here a peer's request is successful if the peer can get the data within a specified time after joining the overlay.

Fig. 3 shows that, our optimized overlay can better utilize the bandwidth and all the three optimizing strategies have shown a significant improvement over the non-optimized one, especially when there are not enough bandwidth resources. The largest increase in acceptance rate is about 125% when the resource index is 0.5.

Note that when the resource index is very low, the bandwidth ordering performs much better, and when the resource index goes to a high level, the lifetime ordering strategy works the best. It is because that the bandwidth is the most important issue when resources are scare. When there are enough resources, the stability of the peers becomes the main influencing factor.

Fig. 4 shows the average transmission latency from the source node to every peer. We observe that the average latency of three optimized overlays is much smaller than that of the non-optimized case. Note that when the resource index is extremely low, the average latency of the non-optimized overlay also decreased greatly. This is because that the non-optimized overlay only served a small portion (about 16%) of the peers at that resource level. However, in the other cases, all the three optimized overlays significantly outperform the non-optimized overlay. The decrease in average latency is about 25-40%. We did not observe much decrease in the latency using the latency ordering strategy. This is because the number of application–level hops is a more dominating factor than the network distance in the packet transmission time.

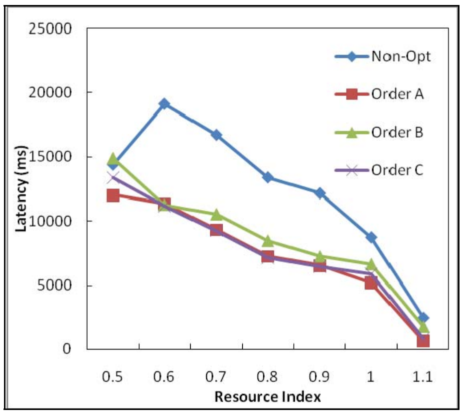

Figure 4: Packet Latency

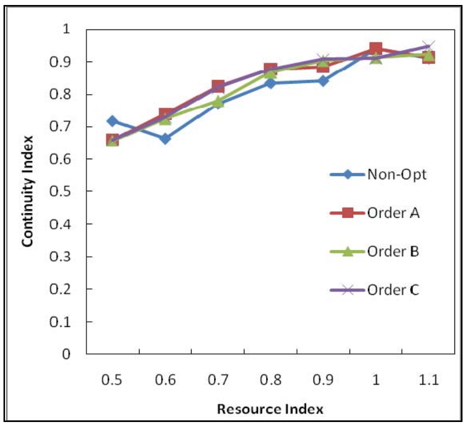

Figure 5: Continuity Index

Finally, we compare the continuity index of the peers. The continuity index is defined as the number of packets that arrived before playback deadline over the total packet number. Figure 5 show the continuity index at different resource levels. We find that, in most cases, the optimized overlays have a slightly higher continuity index than that of the non-optimized one, except for the case where resource index is very low. Therefore, while the optimization procedure improves the streaming performance metrics, it does not hurt the stability or robustness of the mesh-based P2P streaming system.

6. Conclusion

Most mesh-based P2P streaming applications rely on random overlays to transmit video. In this paper, we investigated the problem of optimizing the overlay structure of the underlying unstructured P2P overlay. We found that, through proactive optimization, the overlay structure can be well optimized and the streaming performance can be improved significantly.

7. References

[1] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, DONet/CoolStreaming: A data-driven overlay network for live media streaming, Technical Report, June 2004.

[2] Reza Rejaie, Shad Stafford, “A framework for architecting Peer-to-Peer receiver-driven overlays”, NOSSDAV’ 04, pp 42-48..

[3] Y. Chu, S. G. Rao, and H. Zhang, "A case For end system multicast", Proceedings of ACM SIGMETRICS 2000, Santa Clara, CA, June 2000, pp 1-12.

[4] V Venkataraman, P Francis, “Chunkyspread: Multi-tree unstructured Peer-to-Peer multicast”, IPTPS 2006.

[5] YW Sung, M Bishop, S Rao, “Enabling contribution awareness in an overlay broadcasting system”, SIGCOMM 2006.

[6] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh", SOSP, 2003.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments", SOSP, 2003.

[8] J Liang, K Nahrstedt, “DagStream: locality aware and failure resilient peer-to-peer streaming”, Proceedings of SPIE, 2006

[9] Chuan Wu, Baochun Li, Shuqiao Zhao, Magellan: Charting large-Scale Peer-to-Peer live streaming topologies, ICDCS 2007..

[10] Nazanin Magharei, Reza Rejaie, “PRIME: P2P receiver-driven mesh-based Streaming”, NOSSDAV’06

[11] X Hei, C Liang, J Liang, Y Liu, KW Ross, “A measurement study of a large-scale P2P IPTV system”, IPTV Workshop, International World Wide Web Conference, 2006.

[12] Nazanin Magharei, Reza Rejaie, Yang Guo, “Mesh or Multiple-Tree: A comparative study of live P2P streaming approaches”, Proceedings of IEEE INFOCOM, Anchorage, Alaska, May 2007.

[13] Mark Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, Maarten van Steen, “The peer sampling service: experimental evaluation of unstructured gossip-based implementations”, Proc. Middleware, October 18-22, 2004, Toronto, Canada

[14] Graphviz: http://www.graphviz.org/ .

[15] Shi Zhou, Guoqiang Zhang and Guoqing Zhang , “Chinese Internet AS-level Topology”, to appear in IET Communications..

[16] PPLive: http://www.pplive.com/

[17] UUSee: http://www.uusee.com/

[18] FixedOrbit: http://fixedorbit.com/

[19] China ASN list: http://bgpview.6test.edu.cn/bgp-view/cur_ana/ipv4cn/china_asnlist.shtml

[20] Speedtest: http://www.speedtest.net

[21] Broadmeter: http://www.linkwan.com/gb/broadmeter/SpeedAuto/stats

[22] CNNIC, “19th statistical survey report on Internet development in China”, Jan. 2007.

[23] Cymru: http://www.cymru.com/BGP/asnlookup.html.

[24] V. Vishnumurthy and P. Francis. “On heterogeneous overlay construction and random node selection in unstructured P2P networks”, Proceedings of IEEE INFOCOM, Barcelona 2006.

[25] BRITE: http://www.cs.bu.edu/brite/

[26] GTNets:http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/