

A Peer-to-Peer Network for Live Media Streaming – Using a Push-Pull Approach¹

Meng Zhang, Jian-Guang Luo, Li Zhao, and Shi-Qiang Yang

Department of Computer Science and Technology
Tsinghua University, Beijing, China, 100084

{zhangmeng00, luojg03}@mails.tsinghua.edu.cn, {zhaoli, yangshq}@tsinghua.edu.cn

ABSTRACT

In this paper, we present an unstructured peer-to-peer network called GridMedia for live media streaming employing a push-pull approach. Each node in GridMedia randomly selects its neighbors in the overlay and uses push-pull method to fetch data from the neighbors. The pull mode in the unstructured overlay which is inherently robust can work well with the high churn rate in P2P environment while the push mode can efficiently reduce the accumulated latency observed at user nodes. A practical system based on this framework has been developed. And the performance evaluation of our system which is established on PlanetLab [8] demonstrates that the pull-push method in GridMedia achieves good qualities even in high group change rate. Furthermore, our system was adopted by CCTV to broadcast the Gala Evening for Spring Festival 2005 through the Internet and attracted more than 500,000 users all over the world at that night with the incredibly maximum concurrent users of 15,239.

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed System – Distributed Applications; H.3.5 [Information-Storage and Retrieve]: Online information Service – Data Sharing

General Terms

Design, Experimentation, Performance

Keywords

Peer-to-Peer, Streaming, Push-Pull, Delivery Ratio

1. INTRODUCTION

With the explosive growth of multimedia services and applications over the Internet, there has been much work in recent years on the topic of streaming video to a large population of users. Due to the limited deployment of IP multicast, application layer multicast based P2P media streaming have attracted more and more research interests and efforts. However, the scalability, robustness to high churn rate of nodes and the fluctuation of network bandwidth are all challenges of P2P streaming. In the past few years, a number of P2P multicast architectures were proposed [1, 2, 3]. As

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM'05, November 6–11, 2005, Singapore.

Copyright 2005 ACM 1-59593-044-2/05/0011...\$5.00.

the tree-based approaches are vulnerable with dynamic group variation [5, 7], gossip based unstructured framework has recently become popular owing to its inherent robustness, such as PRO [4], DONet [5], PRM [6]. In our framework, we also adopt the gossip based approach for the construction of the overlay. Moreover, we use a push-pull mechanism to not only keep robust with the nodes participating churn but also efficiently reduce the latency observed at user nodes. We evaluate the performance of our approach over PlanetLab. Almost all active PlanetLab nodes across five continents are involved in our experiments. To our knowledge, very few literatures have utilized so many nodes as in our experiments over PlanetLab. Our system was adopted by CCTV (the largest TV station in China) to broadcast the Gala Evening for Spring Festival 2005 through the Internet. Over 500,000 users all around the world enjoyed it. The peak simultaneous online number reached 15,239, achieved by only one common streaming server with an encoding rate of 300 Kbps. Only 200 nodes directly connected with the server. So far as we know, systems to support users of a par scale at a rate of 300 Kbps by only one server have been seldom published in both academic and industrial world.

2. GRIDMEDIA STRUCTURE OVERVIEW

In GridMedia, we use an unstructured overlay to accommodate the high churn rate in live streaming application. A well-known rendezvous point (RP) is deployed to assist the construction of the overlay. As a startup, a participating node first contacts the RP to get a list of part of the nodes already in the overlay, called a login process when the local clock of the new node should be synchronized with RP. Then the participating node will randomly select some nodes in this list as its neighbors. In the latest version of GridMedia, the overlay organized in a distributed way. Every node maintains a *member table* in which each item represents an active member in the current overlay. Each item has a field named *life-time* denoting the elapsed time since the last message received from this member. When the life-time of a member exceeds a threshold, it will be removed from member table. Meanwhile, member tables are exchanged between neighbors periodically. Once a neighbor of a node quits or fails, the node selects a new neighbor from its member table. Other details of the overlay construction can be found in [7], a brief introduction. Furthermore, each node uses a push-pull mechanism to fetch data from its neighbors. The pure pull mode in GridMedia system is interpreted in Section 3. Section 4 presents the employed push-pull method.

¹Supported by the National Natural Science Foundation of China under Grant No. 60273008 and No. 60432030

3. STUDY OF PURE PULL METHOD

DONet [5] takes a data-driven streaming approach upon a multi-partner overlay. Each node in DONet periodically exchanges buffer map (BM) of media segments with partners, and then retrieves the absent segments from partners which reports to own the segments. This pure pull method will bring tremendous latency from root to each node, i.e., the huge delay between the sampling time at the server and the playback time at users. We define this delay as an **absolute delay**, and the playback time as **absolute playback deadline**. Pay attention that the absolute delay defined here is different from the deadline used in [5] which refers to the elapsed time after receiving the first packet at each node. However, a number of applications are delay-sensitive, such as interactive IP TV, distance education, etc. Streaming multicast using data-driven approach may not meet these demands. To exactly evaluate the quality performance, we define **delivery ratio** to represent the number of packets that arrive at each node before or on absolute playback deadline over the total number of packets. Apparently, a greater (at least no smaller) delivery ratio can be reached at a node with the absolute delay increasing. Another term **α -playback-time** should also be defined here to denote the minimum absolute delay at which the delivery ratio is larger than α , where $0 \leq \alpha \leq 1$. In this section, we study the delivery ratio as a function of absolute delay when using a pure pull method.

3.1 Simple Analysis of the Pull Method

We first review the details of the pull mode in GridMedia. The pull component in GridMedia is similar to the data-driven approach in DONet. But we use one UDP packet as a unit of transmission rather than a segment containing one second media content which is convenient for synchronization as mentioned in [5]. Since RTP transport protocol is employed, sequence number field in RTP packet can be used for buffer management, while the time stamp field is used to synchronize among nodes. In addition, we adopt UDP with TFRC for transmitting between nodes.

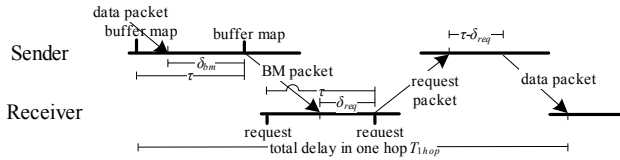


Figure 1. One hop delay using pure pull method

In the pull mode of our system, when a packet goes from one node to another, the following three steps are done as depicted in Figure 1: first, the sender informs the receiver that the packet has already been in its buffer; second, if the receiver needs this packet, request for the packet is sent to the receiver; third, the sender will deliver all the requested packets to the receiver at an smooth rate because of the control of TFRC. Intuitively, at least three end-to-end delays are involved in these steps. Furthermore, for the efficiency of information exchange between nodes, buffer map and requests will only be sent periodically so that dozens of packets can be mapped into a single packet. As a consequence, the delivery of most packets will have extra delays in one hop. More formally, we denote the interval between two buffer map packets and two request packets is τ . In the first step, most arrivals of packets will not be notified to the receiver until the next buffer map packet is sent. We use δ_{bm} to denote this waiting time. In the second step, the notified packets will not be requested as immediately as the buffer map packet is received. Let δ_{req} denote this waiting time. In

the third step, packets also have to wait due to the smooth control. Since the packet with the largest waiting time in the second step will be sent first and the requested packets should be finished transmitting in one cycle if the bandwidth is sufficient, the waiting time is $\tau - \delta_{req}$. Apparently, δ_{req} and δ_{bm} are independent, and their average values are both $\tau/2$ if the packet rate is invariable. Thus, the total average latency for a packet transmitted in one hop T_{1hop} can be computed as $\delta_{bm} + \delta_{req} + (\tau - \delta_{req}) + 3\overline{\delta_{EED}} = 3\tau/2 + 3\overline{\delta_{EED}}$, where $\overline{\delta_{EED}}$ is the average end-to-end delay between nodes.

In the pull mode of GridMedia, each node maintains a packets owner table. Each item in this table records the owner neighbors of a packet in the current slide-window. The table will be updated once a buffer map was received from a neighbor. If the buffer map contains packets with larger time stamps than the largest one in local window, the slide-window moves forward. In our system, the absent packets with larger sequence number will be preemptively contained in the request packet. This aims to reduce the transmission delay. As a result, each packet will be first fetched from the neighbor who possesses it earliest. When the streaming transmission becomes steady, it can be imagined that the transport paths through which the streaming is delivered tends to form a shortest sub-tree of the random-built mesh. We know that if the degree of each node is limited to n , a tree whose internal nodes have $n-1$ children except for the root node has n children is the shortest one. Hence, we can employ this tree structure to estimate the upper bound of the average absolute delay in the pull mode.

Following the simple analysis above, we can compute the approximate average delivery ratio \bar{r} at any specified absolute delay. Assuming that the group size is N and each node has at most n neighbors, we let the streaming transport through a shortest tree described previously. Let T_{1hop} denote the average delay of each packet transmitted in one hop. Thus, a time of iT_{1hop} ($i \geq 0$) after a packet was sent out from the root node, the nodes whose depth are less than or equal to i will receive it if no packets is lost. We denote the number of these nodes as N' . Therefore, at this absolute delay of iT_{1hop} , the average delivery ratio is the ratio of N' to the group size N . The result is written formally as below, where the average delivery ratio at the other absolute delay can be approximately calculated by linear interpolation:

$$\bar{r}(iT_{1hop}) = \begin{cases} \sum_{k=0}^i \lfloor n^k/N \rfloor & i = 0, 1 \\ (1+n)/N + \sum_{k=1}^{i-1} \lfloor n(n-1)^k/N \rfloor & 2 \leq i \leq \lceil \log_{n-1}(N/n) \rceil \\ 1 & i > \lceil \log_{n-1}(N/n) \rceil \end{cases}$$

3.2 Experiment of the Pull Method

An experiment has also been established on PlanetLab for the performance of the pull method. In this experiment, only the result in static environment is presented in order to compare with the analytical result. The total nodes number or the group size is 310 ($N=310$), which are almost all the available nodes that can be utilized on PlanetLab (details of our experiments on PlanetLab can be found in Section 5). Each node has 5 neighbors at most ($n=5$). The cycle of exchanging buffer map packet and request packet is 1 second ($\tau=1sec$). The average packet rate is 36 packets/sec and the bit rate is about 360 Kbps. Simple RTT measurement shows that the average RTT between nodes on PlanetLab is approximately 120ms, as a result $\overline{\delta_{EED}} \approx 60ms$. In this experiment,

only the pull mode of the system is turned on. In order to obtain the average delivery ratio at several absolute delays, each node will synchronize its local clock with RP including the root node when login. A log server collects all the report packets that contain the delivery ratio information. Figure 2 illustrates the analytical upper-bound and experimental result of the average delivery ratio as a function of absolute delay. Attention should be paid that both analytical and experimental results reveal the pure pull method causes a striking latency between the sampling time and the playback time. The α -playback-time ($\alpha=0.97$) of the experimental result is about 10 seconds – a tremendous delay.

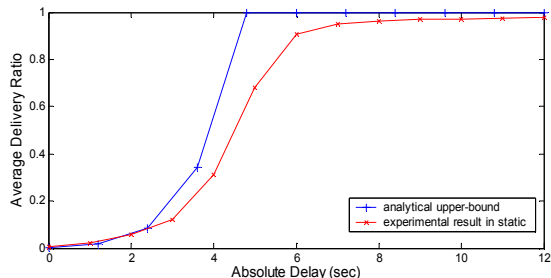


Figure 2. Delivery ratio as a function of absolute delay by analysis and experiment, group size 310

4. PUSH-PULL MECHANISM

Although the pure pull or data-driven method works well with high churn rate and diverse network conditions, it can't meet the demands of delay-sensitive applications because of the striking latency accumulated hop by hop. Additionally, strong buffer capacities at each node are needed to store the exchanging data. We propose a simple push-pull streaming mechanism to greatly reduce the latency and inherit most good features (such as simplicity and robustness) of the pure pull method. Concretely, our push-pull mechanism can obtain the same delivery ratio at a much smaller absolute delay than the pure pull method.

4.1 Packets Scheduling

In brief, push-pull streaming mechanism can be described as follow: each node uses the pull method as a startup, and after that each node will relay a packet to its neighbors as soon as the packet arrives without explicit requests from the neighbors. More detailedly, we classify the streaming packets into pulling packets and pushing packets. Just as the name implies, a pulling packet of a node is delivered by a neighbor only when the packet is requested, while a pushing packet is relayed by a neighbor as soon as it is received. Each node works under pure pull mode in the first time interval when just joining. After that, based on the traffic from each neighbor, the node will subscribe the pushing packets from its neighbors accordingly at the end of each time interval. We use a simple roulette wheel selection scheme to allocate pushing packets in the next time interval to each neighbor. The selection probability of a neighbor is equal to the percentage of traffic from that neighbor in the previous time interval. Meanwhile, the lost packets induced by the unreliability of the network link or the neighbors failure will be pulled as well from the neighbors, where the roulette wheel selection scheme is also used to select the suppliers of each packet from neighbors. Thus, most of the packets received will be pushing packets from the second time interval. With the purpose of controlling easily, the start of time interval on each node should be synchronized. Therefore in our system each node synchronizes with RP when first login.

4.2 Pushing Packets Map and Tracker

Since each neighbor of a node should push different packets, we use a *Pushing Packets Map* (PPMAP) to represent all the pushing packets in the next time interval. First, we partition the streaming into P parts (numbered by $0, \dots, P-1$), and each packet is hashed into only one part. As mentioned previously that an RTP packet has a field of sequence number, we use simple hashing function (mod P function) to map each packet to a part of streaming. In the PPMAP, every bit represents one part of streaming. And all the packets which can be hashed into any part of the streaming contained in the PPMAP should be pushed. For each neighbor of a node, there are two relevant PPMAPs – incoming PPMAP and outgoing PPMAP, where the incoming PPMAP represents the pushing packets from this neighbor to local node while the outgoing PPMAP represents the ones from local node to this neighbor.

In a peer-to-peer environment, the packets received by a node can not be expected to arrive in order. Hence a problem appears that how a node decides whether a packet is lost in the transmission or will be pushed from one of the neighbors later. It can be imagined that if a node do not make a correct decision, some packets will be either missing or received more than once. With the aim of avoiding such a packet loss or redundancy, a simple solution is employed. In our system, each sender should record the packet with the maximum sequence number up-to-date pushed to each neighbor. When the sequence number of a just-received packet is far behind the largest sequence number up-to-date by a threshold, it will not be relayed. We call this threshold the *maximum lag gap* at sender, denoted by l_{sender} . Likewise, each receiver use a *Pushing Packet Tracker* to record the packet with the maximum sequence number up-to-date pushed from each neighbor. When the Pushing Packet Tracker detects a packet whose sequence number falls behind the maximum one up-to-date by a threshold is still not in buffer, it will request for this packet from a neighbor in the next request cycle, i.e., the packet will be pulled from a neighbor. This threshold is the maximum lag gap at receiver, denoted by $l_{receiver}$. Moreover, when the Pushing Packet Tracker detects that no packets has been pushed from a neighbor for a timeout, all the packets which should have been pushed by this neighbor will be pulled from other neighbors in the current time interval, which is against the failure of neighbors. Apparently, if $l_{receiver} \geq l_{sender}$, there will be no redundant packets pulled.

5. PERFORMANCE EVALUATION ON PALNET-LAB

All of our experiments are established on PlanetLab [8], where the active nodes that can be utilized ranges from 300 to 350. Additionally, with the purpose of controlling all the nodes efficiently, we clustered them into 12 sub-groups in terms of the nodes location, and each group has a leader. We use the node at planetlab1.csail.mit.edu to command all the leaders so that all the active nodes on PlanetLab can be controlled. Meanwhile, several tools were written for mastering the remote nodes to join or depart freely. Most of the parameters are the same with the ones in Section 3. The time interval in subscribing pushing packets is 10sec.

We first show the convergence of the delivery ratio when the nodes join in the multicast group. Figure 3 illustrates the delivery ratio as a function of elapsed time when the 310 nodes take part in the group one by one in an initialization period. After that, they will persist in the life time (typically an hour in our experiment). We call this a static environment. The dotted line in Figure 3

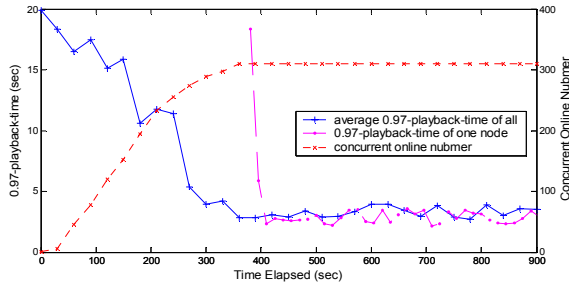


Figure 3. The convergence of the α -playback-time where $\alpha = 0.97$ in the push-pull method

shows the concurrent online number at different time. The solid line represents the average α -playback-time ($\alpha=0.97$) of all nodes with the elapsed time. We can observe that the average α -playback-time goes down to about 2 to 3 seconds almost as immediately as all the nodes finish participating. Moreover, the dashed line in Figure 3 depicts the α -playback-time of the node joining the group at last that is considered to be far away from the root node. The α -playback-time ($\alpha=0.97$) of this node converges to 2~3 seconds within only about 20 seconds.

Figure 4 indicates the comparison between the pure pull method and the push-pull method in both static and dynamic environment. In dynamic environment, 335 nodes on PlanetLab are utilized. Each of them will repeatedly join and depart the group. The online and offline duration of each node per time are exponentially distributed with an average of 100 and 10 seconds respectively, which is indeed a high group change rate in practice. The online number ranges from 300 to 320. We note that using the push-pull method can reach an equivalent α -playback-time at a much smaller absolute delay than the pure pull method as shown in Figure 4. The two solid lines show that the α -playback-times ($\alpha=0.97$) of the pure pull method and the push-pull method are 10 and 3 seconds separately in static environment while the two dotted lines illustrate that the α -playback-times ($\alpha=0.95$) of the two methods in dynamic environment are 22 and 13 seconds separately. Additionally, all the experiments show that the control overhead of our system is always no more than 2%. More details about the experiments can be found at my homepage [9].

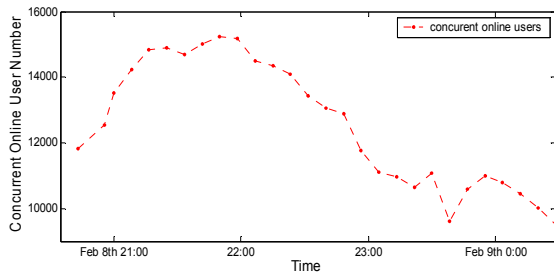


Figure 5. Number of users over time in Feb 8th.

6. GRIDMEDIA: SERVICE ON INTERNET

GridMedia system was adopted by CCTV to broadcast the Gala Evening for Spring Festival through the Internet and attracted more than 500,000 users all over the world at that night with the maximum concurrent users of 15,239. Figure 5 gives the statistical result of online user number on the night of 8th Feb. 2005. The video encoding rate in the broadcast was 300 Kbps. Users from China, North America, Europe and other continents had enjoyed it.

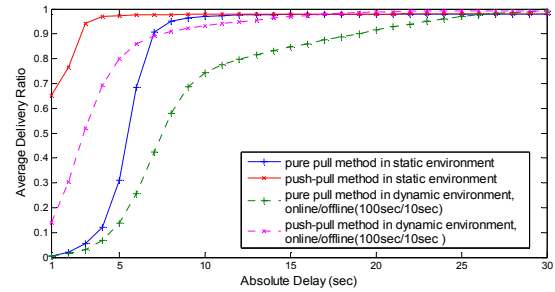


Figure 4. Comparison between pure pull and push-pull method in both static and dynamic environment

The peak online number 15,239 was a result just supported by one single common server. Systems that are capable to sustain users of a par scale at a bit rate of 300 Kbps by only one server have seldom been published before as much as we know.

7. CONCLUSION AND FUTURE WORK

In this paper, we first make a simple analysis of pure-pull method in P2P streaming multicast and reveal the tremendous latency observed at user node when using the pull method. Then we propose an unstructured protocol with a novel push-pull mechanism to greatly reduce the latency and inherit most good features (such as simplicity and robustness) from the pull method. After that the performance of our approach is evaluated on PlanetLab. Finally, we present a preliminary statistical result of our practical system over the Internet. Our system indicates that peer-to-peer technology is really an efficient way for multimedia content delivery. Supporting the group in which most nodes have very limited uploading bandwidth and the optimal selection of the maximum lag gap in push-pull mechanism should be well studied. Besides, we need to evaluate some other important performance (such as link stress, etc.) on PlanetLab. The experience of our service over the Internet is also very interesting to further interpret.

8. REFERENCES

- [1] Y. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, June 2000.
- [2] S. Banerjee, B. Bhattacharjee, and Christopher Kommareddy. Scalable application layer multicast, In *Proceedings of ACM SIGCOMM*, August 2002.
- [3] Duc A. Tran, Kien A. Hua and Tai Do. ZIGZAG: An efficient peer-to-peer scheme for media streaming. In *Proceedings of IEEE INFOCOM*, April 2003.
- [4] Reza Rejaie, Shad Stafford. A framework for architecting peer-to-peer receiver-driven overlays. In *Proceedings of NOSSDAV*, June 2004.
- [5] Xinyan Zhang, JC Liu, Bo Li, and Tak-Shing Peter Yum. CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming. In *Proceedings of IEEE INFOCOM*, March 2005.
- [6] Suman Banerjee, Seungjoon Lee, Bobby Bhattacharjee, and Aravind Srinivasan. Resilient multicast using overlays. In *Proceedings of ACM SIGMETRICS*, June 2004.
- [7] Li Zhao, Jian-Guang Luo, Meng Zhang, and Shi-Qiang Yang. Gridmedia: A practical peer-to-peer based live video streaming system. Submitted to IEEE MMSP 2005.
- [8] PlanetLab website: <http://www.planet-lab.org/>
- [9] My homepage: <http://media.cs.tsinghua.edu.cn/~zhangm>