

Understanding the Power of Pull-based P2P Live Streaming and Doing Even Better*

Meng ZHANG
Tsinghua University
Student
zhangmeng00@
mails.tsinghua.edu.cn

Qian ZHANG
Hong Kong University of
Science and Technology
Associate Professor
qianzh@cs.ust.hk

Lifeng SUN, Shiqiang
YANG
Tsinghua University
Assistant Professor, Professor
{sunlf,yangshq}@tsinghua.edu.cn

1. INTRODUCTION

The Internet has been witnessing the success of Peer-to-Peer (P2P) live streaming during the past three years. A number of commercial P2P streaming software has emerged on Internet (such as PPLive to name a few). Almost all these P2P streaming systems are based on pull-based (or called “data-driven”/“swarming”) protocol [4, 2]. In this type of protocol, the live media content is divided into segments and every node periodically notifies its neighbors of what packets it has. Then each node explicitly requests the segments of interest from its neighbors according to their notification. This protocol is very similar to that of BitTorrent, and the major difference from BitTorrent is that each node only requests the latest generated segments in a much smaller window (called *request window*). The well-acknowledged advantages of pull-based protocol are its robustness and simplicity. However, we would like to report a graceful characteristic of pull-based protocol that has not been paid enough attention: with appropriate protocol design and parameter settings, the simplest pull-based protocol without any intelligent scheduling and bandwidth measurement is nearly optimal in bandwidth utilization and system throughput, that is, as long as the total upload capacity supply is a little higher than the minimum bandwidth demand, all the peers can obtain nearly 100% playback quality. On the other hand, the most outstanding drawback of pull-based streaming is its tradeoff between control overhead and delay [3]: to minimize the delay, each node notifies its neighbors of packet arrivals as soon as it receives the packet and the neighbor should request the packet immediately after receiving the notification, resulting in a remarkable control overhead; while, to diminish the overhead, a node can wait until dozens of packets arrived and inform these packets to its neighbors once in a packet meanwhile the neighbors can also request a bunch of packets once, which obviously leads to a considerable delay. An interesting question is whether there exists any protocol that can not only hold the near optimality in upload capacity utilization but also achieve both much lower delay and smaller overhead. Fortunately, a simple protocol called hybrid pull-push protocol we proposed has all these features.

2. THE NEAR OPTIMALITY OF PULL-BASED PROTOCOL

We use both event-driven packet-level simulation and real-world PlanetLab experiment to investigate the near optimal-

ity of pull-based protocol. In simulation, replying on a 8G-memory machine, we are able to simulate 10,000-node sessions. And in PlanetLab, we totally use 409 nodes all over the world. Term *capacity supply ratio* is defined to represent the bandwidth supply “tightness”, which is defined as the ratio of the total upload capacity among all peers to the raw streaming rate (300kbps) times the receiver number, i.e., the ratio of bandwidth supply to the minimum bandwidth demand. We use three types of nodes whose upload capacities are 1Mbps, 384kbps and 128kbps respectively and download capacities are greater than 300kbps. By adjusting the fraction of each node type, we can derive different capacity supply ratio. For example, we set the fractions of the three types to 0.1, 0.5 and 0.4, then we can derive capacity supply ratio 1.15. We define the *deliverable rate* of a node as the available streaming rate received by the node (not involving redundant streaming packets and control packets), and the *delivery ratio* of a node as the ratio of its deliverable rate to the packetized streaming rate (the raw streaming rate plus 3% header, i.e., 309kbps) encoded at the source node. For delay performance, we define 0.99-playback delay here. As the longer time the packets are buffered at a node, with higher possibility the better playback quality (i.e. delivery ratio) may be achieved, we call the minimum buffered time when the delivery ratio reaches 0.99 as the 0.99-playback time. This delay is the time from the packet is sent out from the source node until the packet is played back at the end node.

Our pull-based protocol uses purely random neighbor selection and random packet assignment among neighbors. When a requested packet does not arrived after a timeout, it will be requested again. Streaming packet size is 1250bytes. Each node randomly selects other 15 nodes as its neighbors and the upload capacity of source node is set to 2Mbps. We assume the bottlenecks are only at the last mile. As shown in Fig. 1 (a), we simulate a static session with 10,000 participants (i.e., no nodes quit). Note that when the capacity supply ratio is only 1.15, the deliverable rate can achieve the best packetized streaming rate (309kbps) (as shown by the dotted horizontal line). And when the capacity supply ratio is smaller than 1.15, the average download/upload rate is equal to the upload capacity, that is, the capacity utilization is nearly 100%. In PlanetLab experiment (Fig. 1 (b)), we select 409 PlanetLab nodes and use the user behavior trace of GridMedia to drive the experiment (dynamic environment). The capacity supply ratio is limited to 1.2. We see that average delivery ratio is very close to 1 all the time. Both simulation and real-world experiment show that the pull-based protocol is nearly optimal in upload capacity utilization and system throughput.

*Supported by the National Natural Science Foundation of China under Grant No.60432030, 973 Program under Grant No. 2006CB303103 and 863 Program under Grant No. 2006AA01Z321

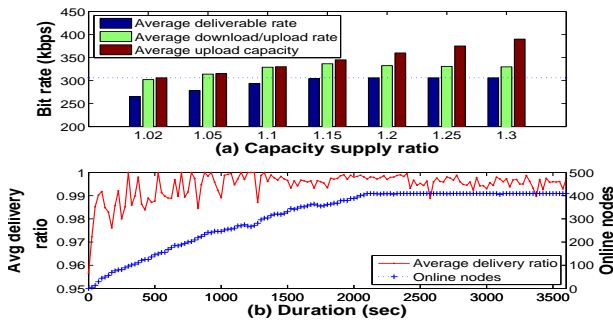


Figure 1: Pull-based protocol (a) Deliverable rate by simulation with 10,000 nodes (b) PlanetLab experiment with 409 nodes driven by GridMedia trace

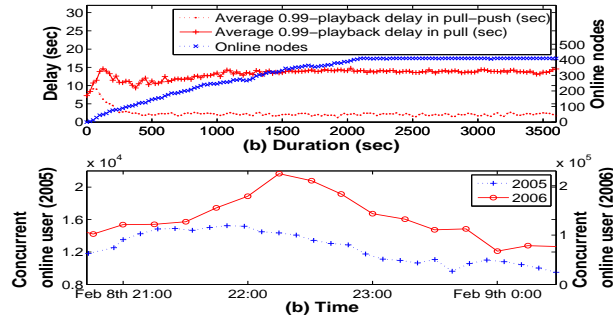


Figure 3: (a) Hybrid pull-push protocol on PlanetLab experiment (b) Real deployed system - GridMedia

Why does the simplest pull-based protocol even without any heuristic strategy have so graceful characteristic? We build stochastic model to explain the near optimality in bandwidth utilization and system throughput of pull-based protocol. Our mathematical model demonstrates that a purely random pull-based protocol can achieve 99% delivery ratio after 10 hops. Briefly speaking, the key reason for the graceful performance of pull-based protocol is its attempting to request a packet again and again when the packet is in the request window. And the parameters that have the most impacts on the performance are the request window and the request interval. Our analysis shows that when the request windows is larger than 20 sec and the request interval is between 400ms and 1 sec, the upload capacity utilization and system throughput will be very close to the optimal.

3. HYBRID PULL-PUSH PROTOCOL

In the proposed hybrid pull-push protocol, the stream is split into multiple sub streams. When a node has just joined the session or some of its neighbors just quit and cause bad quality, it will start to pull packets from all the neighbors. Once a packet is successfully pulled from a neighbor, it will ask that neighbor to push/relay all the packets in the sub stream to which this packet belongs directly in the future. Therefore, when the sub streams are pushed directly, there is no need to exchange buffer map and packet requests, resulting in a much smaller control overhead. The key point of pull-push protocol is to utilize the highly efficient bandwidth-aware multicast routing ability of pull-based protocol to build packing spanning trees. As shown in Fig. 2 (a), the hybrid pull-push protocol can reach the best deliverable rate when the capacity supply ratio is only 1.1, which is even more effective than pull-based protocol. This is because the hybrid pull-push protocol has much smaller overhead compared to

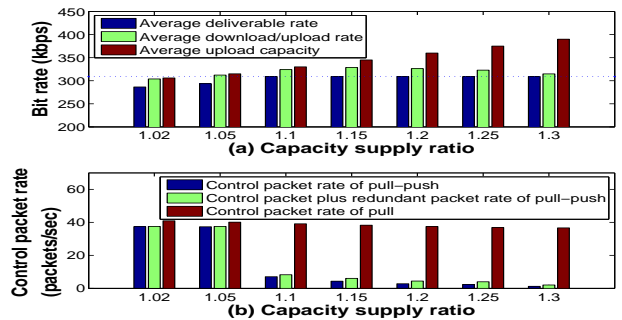


Figure 2: (a) Deliverable rate of hybrid Pull-Push protocol (b) Control packet rate comparison

pull-based protocol as shown in Fig. 2 (b). Furthermore, Fig. 3 (a) shows that, in our trace-driven PlanetLab experiment, the playback delay of hybrid pull-push protocol (about 2.5 sec) is much smaller than that of pull-based protocol (14 sec). Our simulation of 10,000-node session also gives similar result: the playback delays of the two protocols are about 5 sec and 24 sec respectively. Our experiments show that in static and dynamic environment, about 99% and 87% packets are respectively relayed immediately after it is received by a node, which results in a much smaller playback delay.

4. DEPLOYMENT ON INTERNET

The pull-push hybrid protocol is fully implemented in our real-deployed system - GridMedia [1]. This system was adopted by CCTV (the largest TV station in China) to live broadcast TV program since Jan. 2005. And particularly, GridMedia system helped CCTV to broadcast Spring Festival Gala show in 2005 and 2006 through global Internet and respectively supported at most 15,239 and 224,453 concurrent users in a 300kbps streaming rate by only one server in the two years as shown in Fig. 3 (b). And the server bandwidth costs are respectively 60Mbps and 240Mbps for the two events.

5. CONCLUSION AND FUTURE WORK

We first show that the pull-based protocol is near optimal in capacity utilization and system throughput, and then indicate that the proposed hybrid pull-push protocol not only even more effective in system throughput but also has both much smaller overhead and lower delay. In fact, our findings also potentially imply that there is little room for network coding to further promote P2P streaming throughput. For future network, we would like to examine whether the near optimality of pull-based and hybrid pull-push protocol holds under more actual Internet environment (considering the NAT/firewall impacts and cross-ISP bottlenecks) and also build mathematical model for hybrid pull-push protocol.

6. REFERENCES

- [1] Gridmedia: <http://www.gridmedia.com.cn/>. 2006.
- [2] N. Magharei and et al. Prime: Peer-to-peer receiver-driven mesh-based streaming. In *IEEE INFOCOM 2007*.
- [3] V. Venkataraman and et al. Chunkspread: Multi-tree unstructured end system multicast. In *IEEE ICNP 2006*.
- [4] X. Zhang and et al. Coolstreaming/donet: A data-driven overlay network for efficient media streaming. In *IEEE INFOCOM 2005*.