Push/Pull Protocols for Streaming in P2P Systems

Alessandro Russo, Renato Lo Cigno
DISI - University of Trento
38100 Trento, Italy
russo@disi.unitn.it

Abstract—P2P approaches are being adopted in more and more applications, exploiting the intrinsic scaling properties of P2P systems. Streaming multicasting applications in particular seems to benefit most from the P2P approach. In this context, systems have been divided between those that push information from parent to child and those that pull it from child to parent. In this work we explore the generalization to push/pull protocols, where each peer can dynamically push or pull, playing alternately on short time frames, the role of parent or child.

I. INTRODUCTION

Multimedia applications are dominating the Internet in terms of bandwidth usage (at least if we include in this traffic also the download of video files). IPTV to VoIP and video conference applications are among the fastest growing applications, and those based on P2P communications are starting to dominate the lot (e.g., PPLive, Amnet, Coolstreaming, Skype). The main goals in the design of a video streaming system are i) provisioning information delivery within given time bounds, and ii) make an efficient use of the available bandwidth. In addition the system should be simple and resilient to churn and overlay dynamism, as well as scaling well with the number of users and the length of the stream. Traditional approaches for P2P streaming systems propose streaming on either structured or unstructured overlays. Normally in structured overlays the information is pushed from parents to children (e.g. [1]), while in non-structured overlays each peer pulls the information from their neighbors and, depending on the strategies, the parent/child relationship is not as clear as in structured overlays ( [4], [5] are examples). Indeed, some recent papers [6], [7] use pushing in non-structured overlays, hinting to the fact that the structure of the overlay and the information distribution process are not necessarily coupled. Two recent works [2], [3] proposed a system called Interleave where nodes are synchronized and alternate regularly push and pull phases where push is used to disseminate fresh content to target nodes while pull is used to retrieve missing information. The proposal is appealing because of good asymptotic performances, and because it reduces almost to zero the need for signaling. However, all the analysis is based on asymptotic assumptions and synchronized, cycle-based operation.

In this work we present a new class of asynchronous Push/Pull protocols inspired to Interleave using both push and pull for information distribution. We study the fundamental properties of this new class of Push/Pull protocols for information streaming in unstructured overlays.

II. HYBRID PUSH/PULL PROTOCOLS

Push/Pull protocols alternate push and pull operations, each node is autonomous, independent and not synchronized either with the source or with other peers. During Push a node actively sends information (organized in chunks) to a known peer, selected in its neighborhood with any suitable peer selection algorithm. During Pull the node tries to recover chunks needed locally, once more using any suitable peer selection strategy, not necessarily the same used in Push periods. In this phase of the research we are interested in the fundamental properties of the protocol, so we restrict the analysis to random selection of peers in both Push and Pull phases, associated with the idea (shared with the original Interleave proposal) of Pushing new, recently received information and pulling old, missing information. We assume that peers have no information about the status of neighbors (limiting signaling overhead), so that also in presence of high churn the performance should be almost unaffected. In other words, we propose and evaluate a system with elementary characteristics only, so that the performance obtained can be used as a benchmark (hopefully a lower bound) for the performance of more sophisticated systems based on the same principle. Fig. 1 reports the pseudocode of a peer running this protocol. The source is a normal peer with the same resources of the other nodes. This is

This work is supported by the EU IST Napa-Wine project.
very important since other studies assume that the source is a super-peer with more upload bandwidth, and in general, more resources than other nodes. This assumption affects (indeed improves) performance results non marginally: a source that has \( n \) times the bandwidth needed for the stream can inject the information \( n \) times in the system, practically “seeding” \( n \) parallel distribution processes. We analyze the impact of both upload and download bandwidth on streaming performance, using a priority-sharing bandwidth management system. Focusing on streaming, the simple idea of pushing recent information (generated continuously in a stream) and pulling old one seems winning: push operations open the receiver window timeframe (or chunk trading window) with new chunks, and pull operations try to fill holes left by non contiguous pushes. We remark that nodes have no information on target nodes’ status and signaling is limited accepting or refusing a transfer. To explore the importance of choices, we also introduce a small window choice in chunks offers and requests: nodes propose \( S_w \) chunks that can be pushed or that wish to pull.

### III. Early Results and Future Work

Initial investigation is focused on:
- \( \hat{T}_{\text{max}} \) is the maximum propagation delay after a generic chunk is diffused in the entire overlay;
- \( \hat{T}_{90} \) is the 90-th percentile of the transfer delay after every nodes has received the 90% of chunks;
- finally the histogram of the chunks diffusion delay as an approximation of the pdf of the chunks diffusion delay;

We have implemented our protocol on PeerSim adding the priority-sharing bandwidth management mechanism. We consider many parameters: network size, number of chunks, upload and download bandwidth, minimum and maximum one-way delay, number of uploads and downloads and finally the maximum number of chunks offered in either push or pull. The chunks size is 122 kbit and we remark that source has the same resources of other peers. Fig. 2 shows a comparison between \( \hat{T}_{\text{max}} \) and \( \hat{T}_{90} \) varying the upload bandwidth without parallel download. Fig. 3 reports the histogram of chunks’ diffusion delay varying \( S_w \). Fig. 2 shows that with upload bandwidth 1.5 times larger than the stream rate \( \hat{T}_{\text{max}} \) is less than 20s and obviously \( \hat{T}_{90} \) is lower than the last one, which are values allowing VCR-like streaming. Increasing the upload bandwidth both values decrease. We recall that this study is exploring a basic version of this protocol which retrieves all chunks, we are now exploring solutions discarding packets to better understand delay percentiles. Both \( \hat{T}_{\text{max}} \) and \( \hat{T}_{90} \) can be used for tuning buffering for different quality desired, in particular \( \hat{T}_{90} \) is useful for dimensioning MDC or FEC codes. Fig. 3 shows the system sensitivity to parallel downloads and possible choice on chunks selection. Choice helps improving performance, but not dramatically. This basic version of a Push/Pull protocol is coupled, intentionally, with random selection of peers. In these same hypotheses for scheduling pure push or pure pull protocols simply cannot sustain streaming, meaning that the delay continuously grows with the number of chunks, independently from the available bandwidth. Thus Push/Pull seems to have as basic mechanism interesting potential to support streaming in P2P systems. Future work includes heterogeneous networks, investigating churning impact, dynamic neighborhood construction mechanism, buffering techniques and skipping policies, as well as, obviously, considering intelligent scheduling policies.

### REFERENCES


