Deterministic Near-Optimal P2P Streaming

Shaileshh Bojja Venkatakrishnan
Department of Electrical and Computer Engineering
University of Illinois Urbana-Champaign
bjvnkt2@illinois.edu

Pramod Viswanath
Department of Electrical and Computer Engineering
University of Illinois Urbana-Champaign
pramodv@illinois.edu

ABSTRACT
We consider live-streaming over a peer-to-peer network in which peers are allowed to enter or leave the system adversarially and arbitrarily. Previous approaches for streaming have either used randomized distribution graphs or structured trees with randomized maintenance algorithms. Randomized graphs handle peer churn well but have only probabilistic connectivity guarantees, while structured trees have good connectivity but have proven hard to maintain under peer churn. We improve upon both approaches by presenting a novel distribution structure with a deterministic and distributed algorithm for maintenance under peer churn. The algorithm has a constant repair time for connectivity, and near optimal delay. As opposed to order results, the guarantees provided by our algorithm are exact and hold for any network size.

Categories and Subject Descriptors
C.2.1 [Computer-Communication networks]: Network Architecture and Design—Distributed networks, Network topology; C.2.4 [Distributed Systems]: Distributed Applications

Keywords
Peer-to-peer, Streaming, Deterministic, One-to-many content distribution

1. INTRODUCTION
In peer-to-peer (P2P) live-streaming, a low-capacity server uploads live-content to a small number of clients which, together with the other clients (a total of $n$ peers), then exchange the content among themselves. Peers are often limited in their resources such as bandwidth, number of connections they can make to other peers, and available memory. Further, they are not guaranteed to stay in the system and can arrive or depart from the system at any time (also called peer churn). With these constraints, the problem now is how to construct, maintain and schedule the P2P network in order to effectively utilize the available upload capacity for a high rate and to ensure playback with a small delay.

Several approaches have been presented towards the management of P2P networks in the above setting, such as using a centralized controller [2], distributed lookup protocols [3] or random sampling [1]. Our main result is the design of a distribution structure and algorithm that is (i) distributed, (ii) deterministic and (iii) has constant repair time to ensure connectivity under peer arrival and departure. As far as the authors are aware, no other algorithm in the literature has all of the above properties. The tradeoff is that the algorithm has a sub-optimal rate and delay, however we show that the losses incurred are only small. Let us now present the system model.

2. SYSTEM MODEL
We model the P2P overlay network as a directed node capacitated graph $G$. Each peer has an upload capacity of $C$, a constant degree bound of $\Delta$ and a total memory of $M$. We incorporate peer churn by allowing peers to arrive or depart from the system arbitrarily. In particular, we do not require the peers to announce their arrival or departure a priori. Whenever a peer departs, the node and all edges connected to it are lost immediately; only the neighbors of the departing peer are aware of this event. We also assume communication happens as a flow. In addition to the upload capability, we let the nodes be able to communicate control messages through the edges. The terms peer, node and client have been used interchangeably.

3. ALGORITHM AND PERFORMANCE
In the following we let $C = 1$ and consider streaming at a rate of $R \leq 1$. To do this we divide the stream, and distribute it over $m = [R/(1-R)]$ distribution graphs. Let $G_1, G_2, \ldots, G_m$ denote the $m$ distribution graphs with each distribution graph carrying a rate of $R/m$. Figure 1 shows the steady-state distribution graphs $G_1, G_2$ and $G_3$ for a network with 11 users, $R = 3/4$ and $m = 3$.

Steady-state Topology: The structure of the distribution graphs in the steady-state can be described as follows. Each of the $G_i$ comprises of a tree $T_i$ (rooted at the server and spanning all the nodes) and additional edges $U_i$ that connect the leaves of the tree to its interior nodes. For example, the distribution graph $G_1$ shown in Figure 1 can be decomposed into a tree and redundant edges as shown in Figure 2(a) and 2(b) respectively. The latter edges are redundant because they carry streaming packets that have been
potentially already received at the other end. Structurally the tree \( T_1 \) consists of a balanced binary tree of roughly \( n/m \) nodes, plus additional nodes forming chains (i.e. line graphs) of length roughly \( m - 1 \) that hang from the leaves of the binary tree. This has also been illustrated in Figure 2(a), where the nodes hanging from the leaves of the binary tree are of depth 2 (\( = m - 1 \) for \( m = 3 \)). Now, for any degree two node let the left (resp. right) edge and child be called the primary (resp. secondary) edge and child respectively of that node. Then, the redundant edges \( U_i \) consist of edges that go from the left to the right sub-tree of each degree 2 node – the rightmost leaf of the left sub-tree is connected to the secondary child of the degree 2 node. For the \( G_1 \) of Figure 1(a) this has been illustrated in Figure 2(b).

Now, given a distribution graph with the above structure (say \( G_1 \)), the other \( m-1 \) distribution graphs \( G_2, \ldots, G_m \) can be deterministically constructed from \( G_1 \) as follows. Consider any degree 2 node in the binary tree portion of \( T_1 \). The secondary child of that degree 2 node receives a redundant edge from a leaf node of \( T_1 \). Now consider the chain of degree 1 nodes (in \( T_1 \)) to which that leaf belongs. We associate the degree 2 node to that chain of degree 1 nodes. Now, to construct the other substream graphs \( G_2, \ldots, G_m \), we simply swap a member in the degree 1 chain of \( T_1 \) with its associated degree two node. For example, in Figure 1(a) nodes 3,4 are associated with 2, nodes 5,6 with 1 and nodes 8,9 with 7. As such, in \( G_2 \) we have node 3 taking the position of node 2, while node 2 becomes part of the degree 1 chain. Similarly, node 5 takes the place of node 1, while node 1 becomes part of the corresponding degree 1 chain.

Thus, our P2P overlay (i) has redundancy and (ii) is symmetric – features that facilitate very fast and robust maintenance of the network under churn. Whenever a peer departs, the child of this peer in any \( G_i \) can connect to his parent in that \( G_i \). If the departing peer has a degree 2 in any \( G_i \), then the primary child of the peer can connect to his parent, while the secondary child continues to receive the stream via the redundant edge. Similarly, whenever a peer enters the system, it can simple break a single edge to insert itself between two nodes in each of the \( G_i \). Thus connectivity is maintained at all times. With this guarantee, balancing of the \( G_i \)’s to ensure small delay can now be performed via the use of node labels. The delay provided by our algorithm is given by the following.

\[
\text{Theorem 1. In the steady state with } n \text{ peers in the system, the streaming delay is bounded by } \\
\log_2(n+1) + \frac{2R}{C-R} + \log_2(1 - \frac{R}{C}) - 2 \text{ for a rate } R \in (0, C).
\]

Hence, by using a slightly sub-optimal rate we are able to greatly improve the churn handling capability while incurring only a small penalty on the delay (here each edge is assumed to contribute 1 unit of delay). In our second result, we show that for the amount of redundancy used, the delay guarantee of the algorithm is order optimal.

\[
\text{Theorem 2. For structured streaming in which multiple spanning graphs each carry partial flows, the maximum delay across the substream graphs is at least } \\
\log_2(n+1) + \frac{2R}{C-R} + \log_2(1 - \frac{R}{C}) - c', \text{ where } c' = (\Delta - 2)\log_2 \left( \frac{\Delta}{\Delta-1} \right) + \log_2 \left( \frac{\Delta}{\Delta-1} \right) - 1 + 2, \text{ for a rate } R, \text{ degree bound } \Delta \text{ and } n \geq \frac{2R}{C-R}, \text{ if the partial flow graphs have enough capacity redundancy to handle arbitrary node departures.}
\]

Thus, we claim that the \( R/(C-R) \) term in the delay is fundamental for all algorithms guaranteeing continuity of playback.

4. ACKNOWLEDGMENTS

This work was supported by Army grant W911NF-14-0220 and NSF grant CCF-1409106.

5. REFERENCES

[1] J. Kim and R. Srikant. Real-time peer-to-peer streaming over multiple random hamiltonian cycles. Information Theory, IEEE Transactions on, 59(9):5763–5778, 2013.

[2] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient peer-to-peer streaming. In Network Protocols, 2003. Proceedings. 11th IEEE International Conference on, pages 16-27. IEEE, 2003.

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM ’01, pages 149–160, New York, NY, USA, 2001. ACM.