The Methodology of Mesh-Cast Streaming in P2P Networks

Yi Ma
Department of Computer Science
Virginia Polytechnic Institute and State University
may05@vt.edu

Ramazan S. Aygün
Computer Science Department
University of Alabama in Huntsville
raygun@cs.uah.edu

Abstract

Peer-to-Peer (P2P) streaming supports distributed data transfer over Internet. It claims high resource utilization and better streaming performance. In this paper, we define mesh-cast *concept to model "many-to-many" streaming in P2P networks. The main origin of network congestion in mesh-cast is the articulation point/edge in inferred network topology. We resolve the congestion using the peripheral articulation node. We provide two possible approaches, optimistic and pessimistic, to solve network congestion in mesh-cast, both of which use a novel P2P structure, G-Super-Peer backbone.*

1. Introduction

Peer-to-Peer (P2P) networks, like Napster, Gnutella, and Edutella [6], have accommodated tremendous users over the Internet. It is considered as a distributed computing model and competitive with Client/Server model. Several applications in P2P networks further proved its advantages in the environment of Internet. The major advantages are marked as better scalability, reliability, and availability. P2P model also motivates exciting applications that may lead novel methods on the utilization of the Internet in our daily life.

P2P networks bring both opportunities and challenges to multimedia world since many multimedia applications need considerable computing resources. For example, multimedia streaming will occupy significant bandwidth in network and buffer space on the host. It is natural to distribute these workloads by the use of P2P model. P2P streaming has been studied in [3], [4], and [9]. In PROMISE [4], the authors developed "CollectCast" service to optimize the streaming from many senders to one receiver. According to the experimental results, the topology-aware selection in PROMISE improved the streaming performance significantly. CollectCast or many-to-one streaming is likely to be widespread in applications such as P2P multimedia digital libraries [6]. However, when considering more than one simultaneous receiver of different videos, the streaming plan computed by PROMISE might cause significant network congestion.

The general assumption in previous P2P strategies is that the satisfaction of peer requests would not interfere with each other. In this paper, we analyze the problem of arrival of new requests and provide solution on how to handle new peer requests. We define a *mesh-cast* problem to model many-to-one streaming in P2P network and show that it is indeed "many-to-many" streaming when considering multiple concurrent requests. We find the articulation points and edges that play a key role in generating network congestion. Then we propose two possible ways to avoid such congestion by using additional resources in P2P network. We also provide a so-called Geographical-Super-Peer [6] backbone structure to support our solution.

The rest of the paper is organized as follows. The following section explains the problem of many-to-many broadcasting. Section 3 discusses our method, mesh-cast, and explains the structure. The solutions are presented in Section 4. The last section concludes our paper.

2. The Many-to-Many Streaming

We illustrate the origin of the problem in Figures 1 and 2. In Figure 1, each circle node is a peer and each rectangular node is a virtual router inferred from the network to approximate network topology by using network tomography techniques.

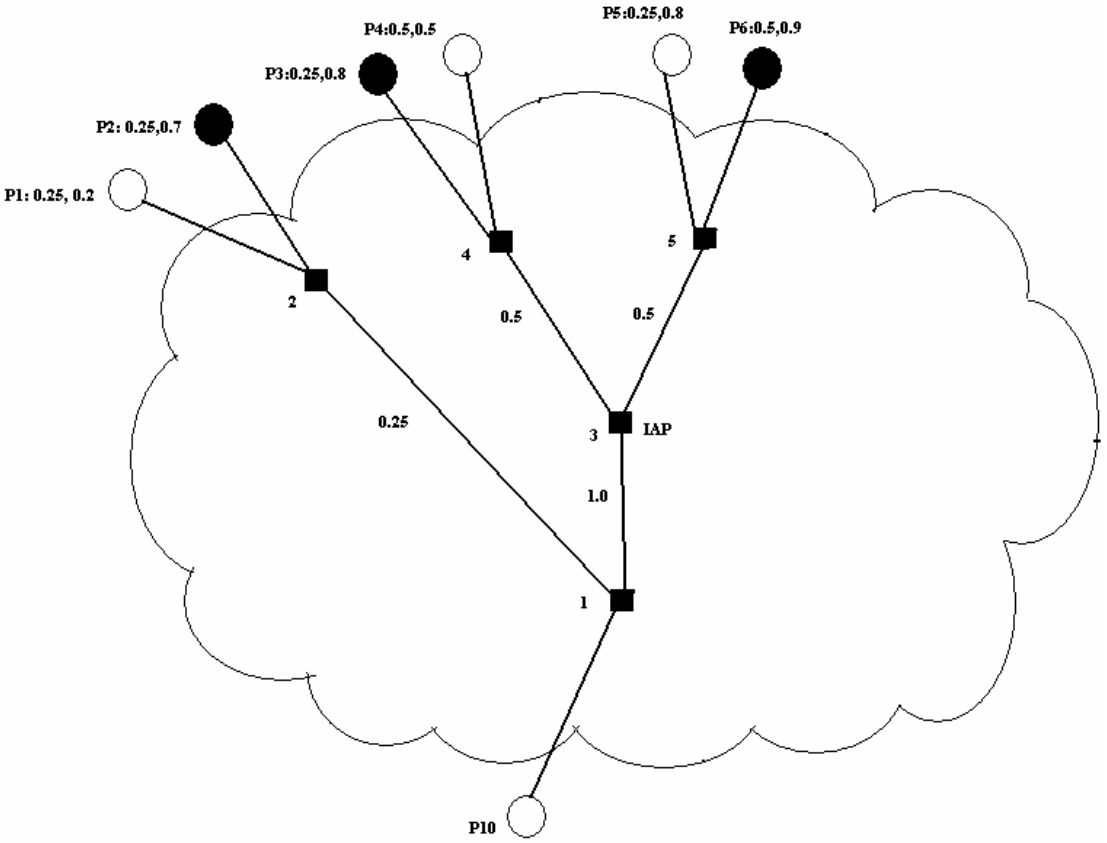

Figure 1. Topology-aware selection in PROMISE

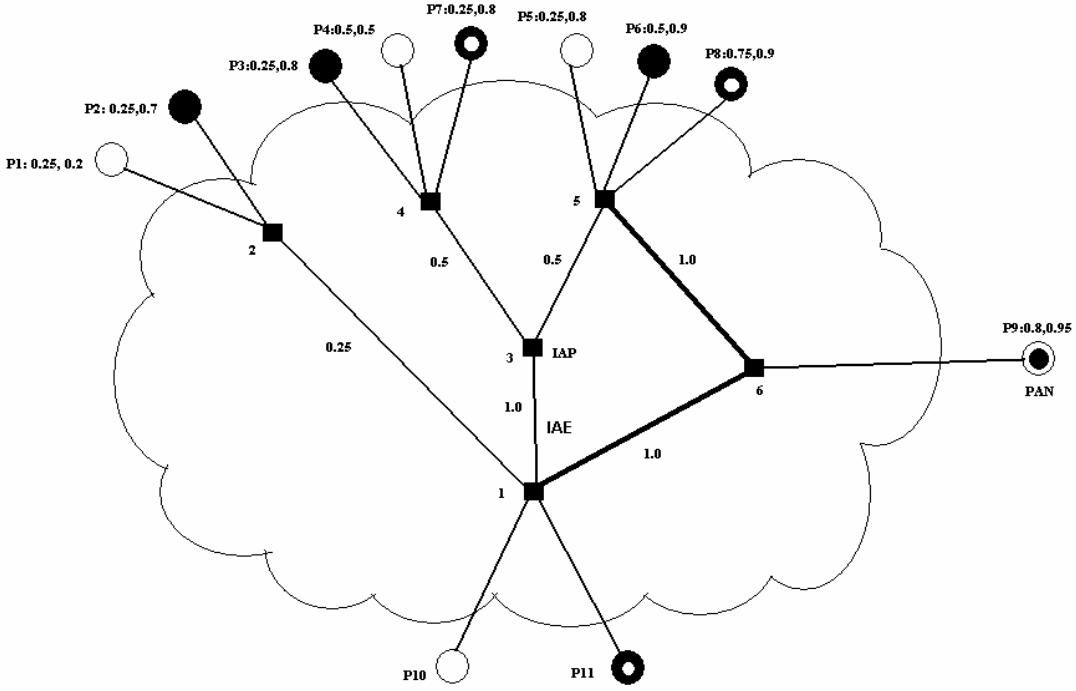

Figure 2. Mesh-cast streaming in a P2P network

Each edge between two nodes denotes the network link between them. The number assigned to the edge is the available bandwidth. The two numbers, from left to right, assigned to each peer is its offer rate and availability respectively. In PROMISE, if there is a request of video A from P_{10}, the copies of A on $P_1 \sim P_6$ are taken into consideration.

Based on topology-aware selection in PROMISE, $\{P_2, P_3, P_6\}$ are selected to streaming data to P_{10}. However, as shown in Figure 2, when there is another request occurs right after we have assigned streaming load to $\{P_2, P_3, P_6\}$, the later request encounters network congestion problem. In Figure 2, P_{11} commit a request for video B that exists on $\{P_7, P_8\}$. The minimum data rate of the request is also 1MBPS. Since the offer rate of either P_7 or P_8 can't satisfy P_{11} individually (0.25, 0.75 < 1.0), we have to use both of them in streaming data to P_{11}. However, the available bandwidth of <3,5> is now 0.25 since half of its 0.5 bandwidth has been assigned to the first request. Note that, based on the study in [10], we assume that the Internet path between P_8 and P_{11} remain unchanged in a significant time period. At the same time, we cannot force underlying network to choose other path. Thus, the later request could not be fulfilled immediately.

From above example, we conclude that the actual streaming manner is many-to-many when there are concurrent requests. Furthermore, if we consider each request individually, CollectCast might yield more congestion in future requests. Although the problem seems to be a typical routing problem in TCP/IP, we could find a simple solution in P2P network. Before we give our solution, it is helpful to give some basic analysis of the problem.

3. Mesh-Cast in P2P Networks

We need to clarify three basic assumptions before we define our problem. First, there exist multiple copies of the same content in the P2P network. Second, a peer in the P2P network has redundant in/out-bound bandwidth and buffer space that are available to other peers in the same P2P network. Finally, the end-to-end Internet path will remain stable during a significant time period, or, even remain unchanged during a streaming session [2], [7]. Before providing solutions, we provide the following definitions.

(1) Inferred Edge and Intra-graph: Let V = $\{P_1, P_2 \ldots P_n\}$ be the set of host peers that have the requested video. Let R= $\{r_1, r_2 \ldots r_n\}$ be the set of peers that generate data request. Let VR = $\{vr_1 \ldots vr_m\}$ be the set of virtual routers that are obtained through topology inference and path merging [4], that connect V and R. We call connections between any two nodes in V, R, or VR as an *inferred edge* (IE). Actually, a peer could only connect with a virtual router, but a virtual router could connect with either a peer or another virtual router. The graph that consists of V, R, VR and their IEs is called an *intra-graph* (IG). Note that, only sender or receiver peers could be involved in an IG. We call streaming sessions that involve peers in an IG as the IG's streaming session. Based on the study in [1] and [4], for each single streaming session (|R| = 1), the resulting IG is a tree rooted at the receiver. However, when we consider multiple receivers, the IG will be graph with cycles.

(2) *Inferred Articulation Edge/Point*: Let $e = <v, w>$ be an IE where v is the sender peer and w is the receiver peer. If e is an articulation edge of IG then e is called an *inferred articulation edge* (IAE). Similarly, a virtual router v in IG is an articulation point then v is an *inferred articulation point* (IEP). IAE and IAP represent the connection degree of IG. Figure 1 depicts a tree where each edge is an IAE and each internal node is an IAP. If an articulation edge/point is removed from IG, IG becomes disconnected. An edge/point is an articulation edge/point if it exists in every path between two points.

(3) *Peripheral Articulation Node*: Let G = <E, V+R+VR> be an IG. Let e is an IAE and v is an IAP. Then a peer u that is not involved in streaming sessions of IG, is a *Peripheral articulation node* (PAN) of e or v, if we add u in inferring IG, then e or v will no longer be an IAE or IAP. Clearly, each IAE or IAP may have multiple PANs. For example, in Figure 2, if we add P_9 into IG, we will also include virtual router 6 in the resulting IG. Now, edges <1,3> and <3,5> are no longer IAE. And, 3 is also not an IAP.

Mesh-Cast Problem: Let C = $\{c_1, c_2, \ldots c_n\}$ is a set of movies that are requested by different peers. Note that c_i and c_j (i != j) might be the same video and for each c_i there exists only single requesting peer. Let R = $\{r_1, r_2 \ldots r_n\}$ is the set of peers that request for content c_i. Based on our assumption, for each c_i in C, there exist multiple copies on $V_i = \{v_{i1}, v_{i2} \ldots v_{in}\}$, where v_{ij} denotes a peer. Then we call $V = \{V_1, V_2, \ldots, V_n\}$ be the *host set* of C. We define $C_i' = \{c_{i1}', c_{i2}' \ldots c_{in}'\}$ be a *content distribution* of c_i such that each c_{ij}' is part of c_i and all the c_{ij}' add up to c_i. Note that some of these c_{ij}' could be of zero size. Then $C' = \{C_1', C_2' \ldots C_n'\}$ is the content distribution of C. Let G = <E, V+R+VR> be an IG. The streaming sessions that transfer data in C from V to R are streaming sessions of G. Then, in P2P

network with the aforementioned assumptions, the actual streaming model will be the dataflow from V, through VR, to R. We call this model as *mesh-cast*. The main problem in mesh-cast is network congestion.

We define the *statistic mesh-cast problem* as following: Find a distribution of C such that data could be transferred from V to R without congestion. Clearly, when |R| = 1, static mesh-cast is exactly the same problem solved by CollectCast. However, the actual streaming model in P2P network is dynamic instead of static. That is, C, V, R, and VR will change with time. So, we define *mesh-cast problem* as following: Find a distribution of C such that we could transfer data from V to R, while at the same time, maintain the P2P network less prone to congestion in future streaming.

4. Solving Mesh-Cast

Before we give possible solutions to mesh-cast problem, we need to analyze the origin of it. In Figure 2, when streaming A, we get intra-graph, G_1, consisting of P_1~P_8, P_{10}, and virtual routers 1~5. When streaming B, we get intra-graph, G_2, consists of P_7, P_8, P_{11}, and virtual router 1, 3, 4, 5. So the intra-graph of both A and B is the union of G_1 and G_2 and denoted as G. The second request encounters congestion because all edges in G_2 are IAEs and there is no other path from P_8 to P_{11}. Since we cannot control the data flow among routers directly, we cannot find a solution without additional resources. However, if we consider P_9 as part of the resulting intra-graph, virtual router 6 should also be included into the final intra-graph G'. Although 3 is an IAP in G, it is not in G'. So, P_9 is a PAN in G'. If we buffer data of P_8 on P_9, and let P_9 relay data to P_{11}, a new path consists of <5,6> and <6,3> is added into streaming session, and thus we will avoid the congestion. So, the key to solve mesh-cast congestion is buffer data on PANs. The choice of when, what, and where to buffer data require a separate paper to explain. We would introduce the basic approaches of possible solutions in this paper.

The actual approach is to use other peers to force data flow to change to other Internet paths. Now, we can say that the congestion in mesh-cast is caused by IAE/IAP and it can be resolved by using appropriate PANs. Also, we need to note that we must process different requests one by one; otherwise there exists mutual exclusion problems when assigning a path to a streaming session. This is prone to network congestion.

Based on above analysis, we could find there are only two possible approaches: optimistic and pessimistic. The essential idea of these solutions, based on the assumption of significant stability of Internet paths, is to use PANs to force data flow change to other idle connections and thus distribute streaming load. In pessimistic approach, we try to avoid future congestion when we compute streaming plan (not necessarily be topology-aware selection as in PROMISE). That is, we try to use as more as possible data hosts. When there is congestion, we try to find PANs for IAEs/IAPs and use PANs to relay data. The disadvantage is that this may generate too many PANs that make the plan difficult to handle. In optimistic approach, we first use topology-aware selection to find optimized content distribution without considering future request. Then, when congestion happens, we try to find appropriate PANs to re-distribute data and thus the streaming loads.

In further consideration, it is impossible to find PANs in P2P network if we only compute the inferred topology for each streaming request individually. So, we need some mechanism to maintain topology information among all peers in a relatively long time. Based on the assumption of the stability of Internet paths, this is possible to maintain such information.

Here we propose using HyperCuP [8] structure to support optimistic solution. As shown in Figure 3, a node in each cub is a Super-Peer that maintains topology information in the format of IG. We call them Geographical-Super-Peers (GSP). Each GSP is connected to a set of peers V and maintain the information of IG = <E, V>. A GSP is elected by all peers in V. When a new peer joins the P2P network, it registers itself to a nearby GSP. Then the GSP will update its own IG. GSP also updates its IG when a peer leaves P2P network. When a peer requests a video, it consults with the GSP to get a part of IG consists only itself and content hosts. Then the receiver computes an optimized content distribution by using methods like CollectCast. If it encounters congestion, it consults with the GSP again and GSP will find appropriate PANs. Then the receiver can use PANs to avoid congestion. Since there is a tremendous number of peers and virtual routers in the P2P network, we need to divide the whole network into several segments. IG of each segment is maintained by a single GSP—1^{st} level GSP. A higher layer of GSP connects 1^{st} level GSPs together, and organizes them into a 2^{nd} level HyperCube. The higher-level GSP abstracts a more condensed IG by merging them together. There may be several layer GSPs and they construct a G-Super-Peer backbone. Since HyperCuP structure is easy to maintain and less prone to dynamic characteristics of P2P network, we consider it a reasonable way to maintain topology information. For robustness reasons, a backup node is selected for GSPs in case of their failure.

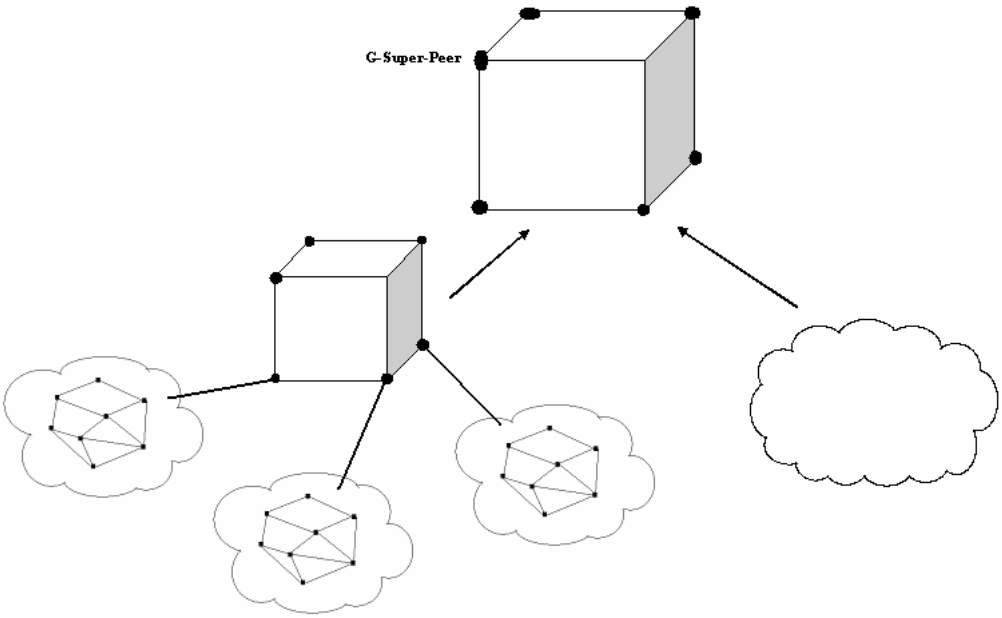

Figure 3. G-super-peer backbone

5. Conclusion

This work presents our first phase on multimedia streaming in P2P networks. In this paper, we analyzed the mesh-cast problem. Mesh-cast problem is resource management when there are multiple streamings at an instant. We identify the reasons of the problem and provide a sample solution. We emphasize that P2P systems have to consider multiple requests at a time. In this paper, we have provided a theoretical analysis of the problem and the solution. As future work, we are planning to validate our results by performing experiments.

6. References

[1] Bestavros, J. Byers, and K. Harfoush. *Inference and labeling of metric-induced network topologies*. In Proc. of IEEE INFOCOM'02, New York, NY, USA, June 2002.

[2] K. Calvert, M. Doar, and E. Zegura. *Modeling Internet topology*. In IEEE Communications Magazine, pages 35:160–163, 1997.

[3] Yang Guo; Kyoungwon Suh; Kurose, J.; Towsley, D., *A peer-to-peer on-demand streaming service and its performance evaluation*. Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on , Volume: 2 , 6-9 July 2003 Pages:II - 649-52 vol.2

[4] Mohamed Hefeeda, Ahsan Habib, Boyan Botev, Dongyan Xu, Bharat Bhargava. *PROMISE: PeertoPeer Media Streaming Using CollectCast.* ACM MM'03, November 2–8, 2003, Berkeley, California, USA.

[5] Yi Ma, Ramazan. S. Aygün. *P2P Based Multimedia Digital Library*. 37th IEEE Southeastern Symposium on System Theory, March 2005, pages 130-134.

[6] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmr and T. Risch *EDUTELLA: a P2P Networking Infrastructure based on RDF*. 11th International World Wide Web Conference Hawaii, USA, May 2002

[7] S. Ratnasamy and S. McCanne, *Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements*, in Proceedings of IEEE INFOCOM 1999, New York,NY, March 1999.

[8] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. *HyperCuP—Hypercubes, Ontologies and Efficient Search on P2P Networks*. In Intl. Workshop on Agents and Peer-to-Peer Computing,2002.

[9] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. *On peer-to-peer media streaming*. In Proc. of IEEE ICDCS'02, Vienna, Austria, July 2002.

[10] Y. Zhang, N. Duffield, V. Paxon, and S. Shenker. *On the constancy of Internet path properties.* In Proc. of ACM SIGCOMM Internet Measurement Workshop, San Francisco, CA, USA, November 2001.