This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Collaborative ISP-CP Live Streaming

C.-H. Him Cheng S.-H. Gary Chan
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
Email: {himcheng, gchan} @cse.ust.hk

Guangyu Shi Jian Chen Hongbo Zhang
Huawei Technologies Co., Ltd
Shenzhen, 518129, China
Email: {shiguangyu, jchen, zhanghongbo888} @huawei.com

***Abstract*—When a content provider (CP) provides peer-to-peer live streaming service, routing decisions with the knowledge of underlay traffic can often lead to much better performance (such as user delay). On the other hand, if the Internet Service Providers (ISPs) provide underlay traffic information to the CP, their overall network cost due to routing inefficiencies can be reduced. There is hence incentive for ISP-CP collaboration. In this paper, we study protocol design to reduce network cost for such collaboration. We consider two ways of collaboration: one on complete information sharing (i.e., the case of ISPs playing the role of CP), and the other one on limited information sharing by means of peer ranking (i.e., the case of P4P framework through Oracle/iTracker).**

We first formulate the problem of minimizing network cost with a certain delay target, and show that it is NP-hard. For complete information sharing, we propose a centralized heuristic which achieves low network cost. For limited information sharing, we present a simple distributed algorithm called Coppice (Collaborative ISP-CP live streaming service) which achieves low network cost. Simulation results show that indeed there is a strong benefit for ISP-CP collaboration, and substantial performance can be gained if ISPs can share complete network information with CP.

I. Introduction

With the penetration of broadband networks, video streaming over the Internet has become popular recently. In order to overcome the limitation of server bandwidth, peer-to-peer (P2P) technologies can be used. In P2P streaming, peers connect with each others to aggregate a full stream. In this way, the uplink bandwidth of peers is utilized to achieve user scalability. Using this technique, many applications on live streaming have been developed and deployed. The popularity of P2P applications is evident from the studies that P2P traffic contributes more than 50-70% of the overall Internet traffic, with an estimated extra annual network investment cost in excess of €500M world-wide [1], [2].

In traditional P2P live streaming deployed by content providers (CPs), peers establish their overlay connections in a rather ad-hoc manner, without taking into much consideration of underlay topology and network conditions. In order for a P2P protocol to be adaptive to peer churns and bandwidth fluctuations, the overlay connections are continuously changed upon detecting network anomalies. Such transient connectivities often leads to much inefficiency and possibly oscillations in bandwidth utilization in the underlay links. Even though the protocol may implement some network measurements to reduce such oscillations, the flows in the resultant overlay topology often suffer from much underlay collisions, adversely affecting stream continuity and overall user delay.

The underlay-oblivious nature of many P2P protocols also present much challenge on traffic engineering to Internet Service Providers (ISPs). As the volume of P2P streaming traffic increases, the relatively short-lived P2P connections make the traffic engineering difficult. The rather random nature of the connections often leads to excessive and unnecessary traffic crossing network boundaries, creating link bottlenecks between ISPs. Such inefficiency not only increases network management and inter-AS (Autonomous System) traffic cost, but also adversely affects other non-P2P traffic (such as web service). In an effort to reduce cost, many ISPs have deployed traffic policing and inter-AS traffic shaping to curb the P2P flows. This, however, greatly violates the network neutrality principle.

As clear from above, it is beneficial for the ISPs and CPs to work together to achieve better streaming with lower network cost. In this collaboration, ISPs feed back underlay network information to CPs. Based on such information, better decision can be made on overlay constructions to achieve lower delay. Because the overlay is constructed reflecting ISP traffic engineering, the P2P routes do not need to make frequent random and transient P2P connectivities (i.e., achieving lower flow volatility). This results in reduced network congestion, management cost and inter-AS traffic.

We show in Figure 1 the collaborative ISP-CP live streaming network under consideration. The CP sets up a video source in one of the ISPs. In order to serve distributed users better, the CP also sets up proxies in different ISPs. These proxies, which are rather stable, exchange streams with each others using a P2P protocol so that a full stream can be aggregated at each proxy. A user joins one of the proxies in his/her ISP to get the stream. *Within* an ISP network, users may take advantage of their network locality, layer-3 multicast capability, and low-cost and abundant bandwidth to achieve effective streaming. There is a certain cost associated with traffic flowing *across* ISPs. Because of the relatively low bandwidth cost within an ISP, in this paper we will focus only on the proxy protocol

This work was supported, in part, by the General Research Fund from the Research Grant Council of the Hong Kong Special Administrative Region, China (611209) and the Cisco University Research Program Fund, a corporate advised fund of Silicon Valley Community Foundation (SVCF08/09.EG01&GMGS08/09.EG05).

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

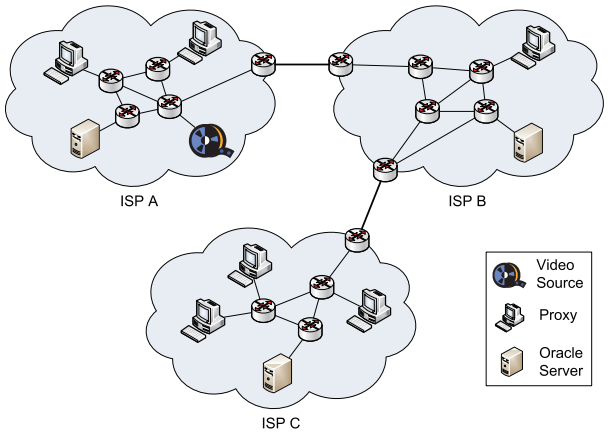

Fig. 1. A network for collaborative ISP-CP live streaming.

to reduce streaming cost *across* ISPs. In this context, "peers" are referred to as a proxies, as they collaborate to achieve streaming across ISPs.

In order to communicate with the CP on underlay information, ISPs set up and maintain some trusted servers such as Oracle or iTrackers in their ASes [3]. These servers exchange with each other their network information, and communicate such information to the proxies for their overlay network construction. The proxies have diverse uplink bandwidth and each of them has to aggregate a full stream to serve their local users. In order to meet a certain service quality, there is a target maximum delay (i.e., the delay from the source to proxy) which should not be exceeded. Given heterogeneous (transmission) cost between the ISPs, our goal is to design proxy-level P2P protocol minimizing the network cost. We also study how to support dynamic joins and leaves of proxies. We consider the following two ways to share information depending on the willingness of the ISPs to make known topology and cost information to the CP:

- *Complete sharing*, where the complete network information and conditions are shared with the proxies. This is the case when the ISPs also play the role of CP, and hence a centralized design is possible. We first formulate the streaming problem with minimum cost, and show that it is NP-hard. Then we propose a heuristic which achieves very good performance. The performance of our heuristic shows the value of centralized coordination and complete information in ISP-CP streaming;
- *Partial sharing*, where the complete network information is not shared to the proxies, i.e., the ISPs do not want to reveal underlay information to the proxies (for some privacy reasons). In this case, the ISPs set up independent Oracle servers and the protocol has to be designed in a distributed manner. Each proxy presents a list of possible neighbors to its oracle server, which returns its ranked list. We propose and study a distributed algorithm as run in proxies called *Coppice* (**Co**llaborative I**S**P-**C**P li**ve**-streaming servi**ce**), which achieves full streaming rate with low network cost.

We study our proposed algorithms using simulations. Our results show that there is a substantial benefit for ISP to collaborate with CP, and significant performance can be gained if ISPs are also CP in providing streaming services (i.e., the complete sharing case).

We briefly review previous work here. There has been some work on ISP collaboration with P2P applications [4], [5]. They study the risks and drawbacks of ISP-P2P collaboration. This body of work is general and orthogonal to our work, which addresses discussed how a live streaming protocol should be designed to achieve the trade-off between ISP cost and CP delay. There has also been work on ISP collaboration for P2P file downloading [6]. The results show that locality can be better taken care of to achieve higher throughput. However, the work cannot be straightforwardly extended to live streaming case, where sustained bandwidth with low delay is an important concern. To the best of our knowledge, this is the first work addressing the cost-delay issue and protocol design for ISP-CP collaboration to provide live streaming service.

Another body of work is on topology-aware streaming, where the focus is on network design given underlay topology (see, for examples, [7]). However, they have not addressed how P4P framework can be used in streaming, and how to achieve the trade-off between inter-ISP cost and delay bound in live streaming. To know underlay topology, one may use network inference techniques (see, for examples, [8] and the reference therein). These techniques are usually centralized in nature, and require some expensive network measurements (and hence relatively slow). With a P4P framework, the underlay topology can be obtained more directly and accurately, and some metrics such as network cost can be reflected more appropriately in the routing design.

The rest of this paper is organized as follows. We present the problem formulation, the Minimum Cost Streaming Mesh (MCSM) problem and its complexity, in Section II. We discuss the centralized heuristics and the distributed *Coppice* in Section III. Illustrative simulation results and comparisons are presented in Section IV. We conclude in Section V.

II. Problem Formulation and its Complexity

We model the network as a connected directed graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices representing the participating source, proxies and underlay routers, and $\mathcal{E}$ is the set of links between them. For any edge $(i, j) \in \mathcal{E}$, there is a link latency $d_{ij} \in \mathcal{R}^+$ (in second) and a bandwidth $b_{ij} \in \mathcal{Z}^+$ (we assume that bandwidth is in some integral units, where each unit is, say, 50 kbits/s).

We further let $G_o = (\mathcal{V}_o, \mathcal{E}_o)$, where $\mathcal{V}_o \subseteq \mathcal{V}$ is the set of the overlay nodes consisting of the proxies and source S, and $\mathcal{E}_o$ is the overlay links between them. Denote $\mathbb{P}_{mn}$ the underlay paths from m to n, where $m, n \in \mathcal{V}_o$.

The video is divided into s substreams, each of which is one unit of bandwidth. Each unit of stream is delivered to all the nodes in $\mathcal{V}_o$ by a spanning tree rooted at S, hence there are exactly s spanning trees in total. Denote T_k the spanning

tree of the kth substream and $D_m(T_k)$ as the source-to-end delay of proxy m in spanning tree T_k. If proxy n is the parent of proxy m in T_k, we clearly have

$$D_m(T_k) = D_n(T_k) + \sum_{(u,v)\in\mathbb{P}_{nm}} d_{uv}.$$

Given that the mesh delay of a proxy is the maximum delay of all its substreams, the delay of proxy m is hence

$$D_m = \max_{1\le k\le s} D_m(T_k).$$

We denote s_{ij} the total streaming rates from proxy i to proxy j, i.e.,

$$s_{ij} = \sum_k 1_{(i,j)\in T_k},$$

where $1_{(i,j)\in T_k} = 1$ if $(i,j) \in T_k$, 0 otherwise. For any proxy m in $V_o \backslash S$, if it gets an aggregate stream of s units, we call it *fully served*. In our streaming network, because all proxies have to be fully served, we need

$$\sum_n s_{nm} = s, \quad \forall m. \tag{1}$$

In other words, if node m receives streams from all s spanning trees, it is *fully served* and can play back the video smoothly. For every underlay links, the aggregation of streaming rates passing through it cannot exceed its bandwidth, i.e.,

$$s_{ij} \le b_{ij}, \quad \forall (i,j) \in \mathcal{E}. \tag{2}$$

Given a streaming rate s_{mn} from proxy m to n, there is an associated cost function $c_{mn}(.)$ depending on s_{mn}, i.e., $c_{mn}(s_{mn})$. We denote $\widehat{D} \in \mathcal{R}^+$ the delay target from the source to any proxy, i.e.,

$$D_m \le \widehat{D}, \quad \forall m \in V_o. \tag{3}$$

Minimum Cost Streaming Mesh Problem (MCSM problem): The MCSM problem is to find a mesh which minimizes the total traffic cost, i.e.,

$$\min C = \min \sum_k \sum_{(m,n)\in T_k} c_{mn}(s_{mn}), \tag{4}$$

subject to Equations (1), (2) and (3).

Claim 1. *The MCSM problem is NP-hard.*

Proof: Traveling salesman problem (TSP) is reducible to our MCSM problem in polynomial time. An input to TSP is a weighted, undirected complete graph $G(\mathcal{V}, \mathcal{E})$ and a vertex $S \in \mathcal{V}$. Recall that the TSP is to find a tour of minimum cost through all vertices exactly once (i.e., a Hamiltonian cycle) such that S is both the starting point and ending point.

To prove that TSP can be reduced to MCSM problem, the polynomial time transformation is as follows. Let $G'(\mathcal{V}', \mathcal{E}')$ be the graph of a TSP instance. We transform $G'(\mathcal{V}', \mathcal{E}')$ into $G''(\mathcal{V}'', \mathcal{E}'')$ by adding a vertex S_{sink} and edges from all the vertices to S_{sink} with cost 0. In this way, the vertices in $\mathcal{V}''$ represent peers and the weight on the edges are the delay between the two adjacent peers. Consider the special case that the uplink bandwidth of each peer is 1, S_{sink} has zero uplink bandwidth, the bandwidth in the network core and the delay target are sufficiently large. Consider also the streaming rate is 1. In this way the resulting overlay topology must be a chain starting at S and ending at S_{sink}. The total delivery cost, C, equals to the sum of all cost in that chain. It is obvious that C in G'' is minimum if and only if the cost of a tour in G' is minimum. Therefore, TSP is polynomial reducible to MCSM problem. Since MCSM problem is at least as hard as TSP problem, MCSM problem is NP-Hard. ■

III. A Centralized Heuristics and Distributed Approach

In this section, we first present a centralized heuristic to solve the problem based on complete information sharing (Section III-A), followed by the distributed approach, Coppice (Section III-B).

A. A Centralized Heuristics

In order to achieve overall low delivery cost, the paths with lower costs while meeting the delay target are preferred. For a proxy, a good heuristic should balance the delay in all the spanning trees formed by substreams. To achieve this, we process each of the s spanning trees in a round-robin manner. For each round, a proxy with the lowest cost whose delay is within the target and has enough bandwidth joins the tree. This process is repeated until all s trees spanning all the proxies are formed.

We iteratively construct s spanning trees. Starting with the source node S as the root of the tree, we take turns to push the nodes in $\mathcal{V}_o$ into each of the spanning trees using Prim's minimum spanning tree algorithm. The algorithm ends when all s spanning trees are constructed.

Regarding its running time complexity, the time needed to construct a spanning tree is $O(|\mathcal{E}_o| \log |\mathcal{E}_o|)$, because each path $(m,n) \in \mathcal{E}_o$ is pushed into the priority queue at most once. Each of them takes $O(\log |\mathcal{E}_o|)$ time to push into the priority queue and $O(\log |\mathcal{E}_o|)$ time for popping and reordering the queue. Therefore, the total time complexity of the algorithm is $O(s|\mathcal{E}_o| \log |\mathcal{E}_o|)$. In other words, the complexity is $O(s|\mathcal{V}_o|^2 \log |\mathcal{V}_o|)$, because $|\mathcal{E}_o| \le |\mathcal{V}_o|^2$ in G_o.

B. Coppice: A Distributed Approach for Low-Cost Streaming

If ISPs do not want to share complete network information to CP, we propose in this section a simple, efficient, adaptative and distributed protocol called *Coppice* (**Co**llaborative IS**P**-C**P** l**i**ve streaming servi**ce**). In order to work with CP without revealing full network information, the ISP can make use of Oracle service, which presents to the CP ISP's preference of a proxy in the form of a ranking list.

A tracker server called Oracle server is set up in each of the ISPs. It contains the following information: 1) all topology information (such as bandwidth and traffic engineering) of the ISP; 2) the traffic cost for streaming packets towards a given destination; and 3) Border Gateway Protocol (BGP) information of other ISPs. For a given list of proxies, the

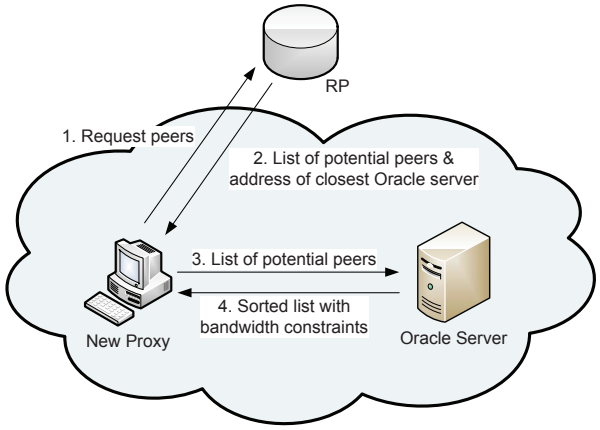

Fig. 2. Parent selection in Coppice.

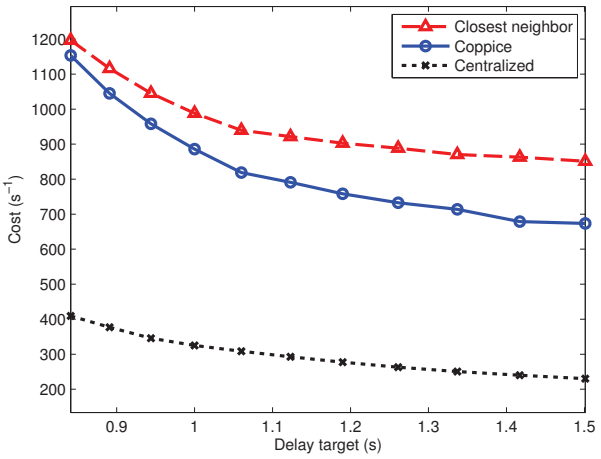

Fig. 3. Trade-off between cost and delay

Oracle server retrieves the routes and costs to each of the proxies and then sorts the list according to the costs. In order not to inject traffic to a bottleneck link, the Oracle server also ranks the list based on residual bandwidth of the links by contacting other Oracle servers.

A joining proxy first contacts a Rendezvous Point (RP) to get a few potential parents and the address of its Oracle server. Then, it forwards a partial list of proxies that it knows of to the Oracle server for sorting. Figure 2 shows the procedures of parent selection in *Coppice*.

Upon receiving the rank list, the joining proxy goes through the rank order and connects to the one after verifying the delay being within the target and the residual bandwidth being enough. This process repeats until the proxy is fully served for each substream.

If the proxy cannot be *fully served* within the delay target, the proxy will try connecting to parents which stream data to itself with the lowest delay. Furthermore, if after visiting all proxies the newcomer still cannot be fully served, the newcomer requests the neighbor of those proxies and sends them to Oracle server for further ranking. Then the above process is repeated till it is fully served.

When a proxy is about to leave, it initiates a LEAVE message to its parents and to all its children, which look for new parents in the mesh using the join process. Proxies also periodically exchange KEEP-ALIVE messages with their parents and children. When a parent failure is detected, a proxy seeks for new parents in the mesh as in the join process.

IV. Illustrative Simulation Results

In this section, we first present our simulation environments and metrics (Section IV-A), followed by illustrative simulation results in Section IV-B.

A. Simulation Environment and Metrics

We develop an event-driven simulation to study the algorithms. We use BRITE to generate different two-level top-down hierarchical topologies [9]. Each topology consists of 100 ASes, each of which with 100 routers. BRITE also provides us link latency in millisecond. Proxies are attached to the routers randomly. The bandwidths of underlay intra-AS links, b_{intra}, is assumed to be abundant (some large value). The inter-AS link bandwidths, b_{inter}, are exponentially distributed within a range with rate λ_1/unit. The proxy bandwidths, b_{proxy}, are uniformly distributed. We consider only the inter-AS traffic costs, denoted by c_{inter}, as the traffic cost within an AS is negligible. Unless otherwise stated, we use the following baseline parameters: $s = 5$ units, number of peers $= 500$, number of RP list $= 1/10$ of the number of peers, $b_{intra} = 50s$ (units), $10s \leq b_{inter} \leq 50s$ (units), $1s \leq b_{proxy} \leq 5s$ (units), $\lambda_1 = 1/15$, c_{inter} is uniformly distributed between $1/s$ and $3/s$ (per unit), $\widehat{D} = 1$s. The proxy holding time is exponentially distributed with mean 10 hours, and the inter-arrival time is exponentially distributed with mean 72s.

We use the following evaluation metrics:

- *Delivery cost*, which is the total traffic cost needed to stream media from the source to all the proxies.
- *Delay*, which is the time of the stream from the source to all the proxies.

We compare the performance of our centralized heuristic and distributed *Coppice* with the traditional closest neighbor approach. In closest neighbor, for each substream, new arrival considers parents whose delay is within the target and tries to build connections to the closest one. If it cannot be fully served, new arrival will connect to proxies which make the resulted delay to be minimum. Closest neighbor is better than random selection as it captures locality of the peers. It is simple to implement and therefore quite a number of streaming systems nowadays adopt an approach similar to it.

B. Illustrative Simulation Results

To illustrate the trade-off between delivery cost and delay bound, we show in Figure 3 the average cost per second for streaming against delay targets given different algorithms. Clearly, we see that the cost decreases when the delay target increases. The cost first decreases quite sharply and then settles

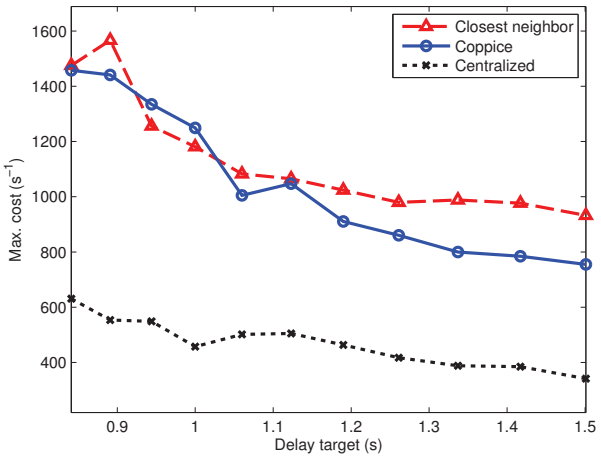

Fig. 4. Trade-off between maximum cost and delay

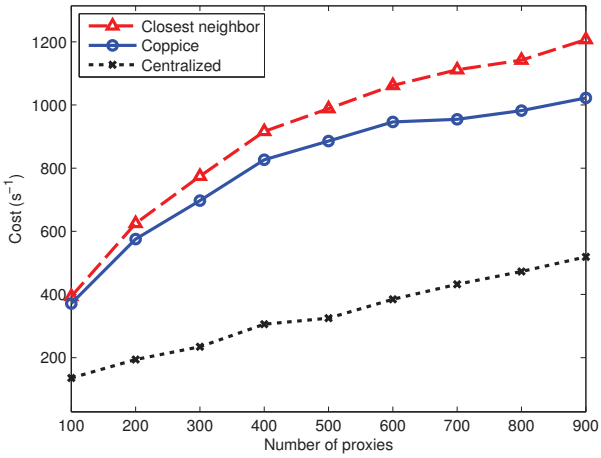

Fig. 5. Distribution cost with different number of proxies.

to some value as the target is relaxed. This is reasoned because proxies would not connect to parents far away when the target is relaxed in order to save network cost. On the other hand, when the delay target is small, proxies may need to build direct connections with each others in order to meet the target, hence spanning many ISPs and increasing the cost. The centralized scheme performs substantially the best due to complete sharing of the network information. Coppice performs substantially better than the closest neighbor scheme because ISPs and CP collaborate through Oracle servers. This illustrates the effectiveness of collaboration between ISPs and CP, especially with complete sharing. The traffic cost also does not depend sensitively on delay target beyond a certain value (e.g. beyond 1.5s). Figure 4 shows the maximum cost of the schemes versus delay targets. The result is similar to the average one.

In Figure 5, we plot the delivery cost of various schemes versus number of proxies. The cost of closest neighbor increase relatively faster than Coppice and centralized heuristic. This shows the weakness of traditional closest neighbor approach, which does not take ISP-localization in parent selection into consideration while our schemes minimize the unnecessary inter-AS traffics. Furthermore, as the number of proxies increases, the cost does not increase proportionally, especially for the collaborative cases. This shows that traffics are better contained within an ISP.

V. Conclusion

In this paper, we investigate ISP-CP collaboration to provide live streaming. The CP puts proxies in different ISPs, and P2P protocols are used among the proxies to aggregate full stream from a certain number of substreams. There is a certain delay target to meet for a good service quality. The ISPs collaborate with the CP by reviewing either partial or complete network information to the CP, so that routing decision can be made more optimally to reduce cross-ISP traffic cost.

We study protocols on reducing network cost across ISPs while meeting a certain delay target. We formulate the problem as Minimum Cost Streaming Mesh (MCSM) problem, which is to form a low-cost mesh meeting a certain streaming rate and streaming delay requirement. We have shown the problem is NP-hard and presented a centralized heuristic for the ISPs to collaborate with the CP with complete information sharing. We then propose a distributed protocol called *Coppice*, which is based on limited information sharing by making use of servers (Oracles/iTrackers) to rank peers presented by the proxies.

We have conducted simulation to study the performance of the schemes, and compare them with the common closest neighbors approach. The results show that our algorithms achieve significantly lower cost given a certain delay target. If ISPs are CP as well, or ISPs can share with CP complete network information, network cost can be significantly reduced (by an order of magnitude in our simulations). Using Oracle/iTracker, the streaming cost can also be reduced substantially. ISP-CP collaboration is beneficial to both ISPs and CP.

References

[1] R. Steinmetz and K. Wehrle, "P2P systems and applications," in *Springer Lecture Notes in CS*, 2005.

[2] T. Karagiannis, A. Broido, N. Brownlee, kc claffy, and M. Faloutsos, "Is P2P dying or just hiding?" in *Global Telecommunications Conference*, vol. 3. IEEE, Nov. 2004, pp. 1532–1538.

[3] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P users cooperate for improved performance?" in *ACM SIGCOMM Computer Communication Review*. ACM, Jul. 2007, pp. 29–40.

[4] Y. Liu, H. Zhang, W. Gong, and D. Towsley, "On the interaction between overlay routing and traffic engineering," in *IEEE INFOCOM*, vol. 4, Mar. 2005, pp. 2543–2553.

[5] A. Kaya, M. Chiang, and W. Trappe, "P2p-isp cooperation risks and mitigation in multiple-isp networks," in *Proc. IEEE GLOBECOM 2009*, Nov. 2009.

[6] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 363–374, Aug. 2008.

[7] D. Ren, Y.-T. H. Li, and S.-H. G. Chan, "Fast-mesh: A low-delay high-bandwidth mesh for peer-to-peer live streaming," *IEEE Transactions on Multimedia*, vol. 11, no. 8, pp. 1446–1456, 2009.

[8] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient algorithms for large-scale topology discovery," in *Proc. ACM SIGMETRICS*, Jun. 2005.

[9] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," in *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS)*, Cincinnati, Ohio, Aug. 2001.