ISP-Friendly Peer Matching without ISP Collaboration

Cheng-Hsin Hsu
School of Computing Science
Simon Fraser University
Surrey, BC V3T 0A3, Canada

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Surrey, BC V3T 0A3, Canada

Abstract

In peer-to-peer (P2P) systems, a receiver needs to be matched with multiple senders, because peers have limited capacity and reliability. Efficient peer matching can reduce the cost on Internet Service Providers (ISPs) for carrying the P2P traffic. We study the following peer-matching problem: given a set of potential senders, find the best subset of them that will minimize the transit cost on ISPs. This problem is fairly general and the proposed algorithms for solving it can be used in many P2P systems. We propose two ISP-friendly algorithms for solving this problem: *ISPF* and *ISPF-Lite*. These two matching algorithms leverage public available information, such as BGP tables, to infer the network topology, and to minimize the cost on ISPs. The inference algorithms, however, are fairly complex, and we propose optimization techniques to reduce the inference time and to lower the memory requirement. We use trace-driven simulations to show that the proposed algorithms outperform other popular matching algorithms by a large margin. Between the two proposed algorithms, *ISPF* results in better matching, but incurs higher complexity. Hence, we recommend *ISPF* if resources are not stringent, otherwise *ISPF-Lite* is recommended.

1 Introduction

Many peer-to-peer (P2P) file-sharing systems have gained tremendous popularity in the recent years. BitTorrent, eMule, and Gnutella, for example, attract millions of users and generate tera bytes of Internet traffic every day. P2P video streaming (live and on-demand) has also seen wide deployment and has been used by numerous users around the globe. In addition, there have been several proposals for designing peer-assisted content distribution networks (CDNs) [18, 16, 15]. In all of these systems, a receiving peer needs to be *matched* with multiple sending peers, because peers have limited capacity and reliability. Peer matching is critical to the success of P2P systems for two reasons. First,

finding senders closer to the receiver can shorten download times in file-sharing systems and enhance video quality in streaming systems. Second, and more importantly, finding senders closer to the receiver can reduce the cost of carrying the P2P traffic within an Internet Service Provider (ISP) and across various ISPs. We refer to the former cost as the *Intra-ISP* cost and the latter cost as the *Inter-ISP* cost.

Despite the importance of peer matching, many current P2P systems randomly match peers. Unfortunately, random peer matching may increase the intra-ISP and the inter-ISP costs. For example, a previous study showed that random peer matching unnecessarily increases the bandwidth consumption on inter-ISP links [3], which are costly for ISP operators [11]. Increasing the cost on ISPs may (actually does, in many cases) lead them to react adversely by shaping the P2P traffic using devices such as [13, 14], or even blocking it. These adverse reactions will degrade the performance of P2P file-sharing and streaming systems, and will make it harder for peer-assisted CDNs to provide high-quality, commercially viable, distribution services. Therefore, it is in the best interest of P2P systems to employ an ISP-friendly peer matching algorithm.

In this paper, we study the following peer-matching problem: given a set of potential senders for an object requested by a receiver, find the best subset of senders that will minimize the transit cost on ISPs. This problem is fairly general and our algorithms for solving it can be used in many P2P systems. For example, in BitTorrent-like networks, a tracker can use our algorithms to return matching senders for a requesting client. In peer-assisted CDNs, where peers help a few dedicated servers in distributing content, servers performing sender-receiver matching can benefit from our algorithms. In Gnutella-like networks, where object lookup is done in a distributed manner, the receiver can employ our algorithms to choose good senders from a list of potential senders who replied with query hits to the query issued by the receiver.

We propose two ISP-friendly algorithms for solving this problem: *ISPF* and *ISPF-Lite*. They first minimize the inter-ISP cost, and then minimize the intra-ISP cost. We give the inter-ISP cost higher precedence because the internal traffic is less expensive to ISPs as they *own* their internal links. We have implemented both proposed peer matching algorithms in a trace-driven simulator. We compared them against two other popular algorithms. Our experimental re-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ROADS'08, December 9, 2008, Madrid, SPAIN

sults show that the proposed matching algorithms outperform the currently used matching algorithm by a large margin: as high as 6 times. Among the two proposed matching algorithm, ISPF results in better matching, but at the expense of higher complexity. Hence, we recommend ISPF if resources are not stringent, otherwise ISPF-Lite is recommended.

The remaining of this paper is organized as follows. In Section 2, we summarize the related work. We present our algorithms in Section 3. We evaluate the algorithms in Section 4. We conclude the paper in Section 5.

2 Related Work

The work in [3] is probably the closest to ours, in which the authors propose to select the majority of senders from the same AS as the receiver to reduce inter-ISP traffic amount. Their work is specific to BitTorrent and only gives priority to senders in the same AS as the receiver, but resorts to random selections for senders outside that AS. In contrast, our algorithms is more general in the sense that we consider the ISP and the PoP topologies when selecting senders in ASes other than the AS the receiver is in. A recent work proposes to use CDNs for finding senders that are likely to be closer to the receiver [4]. In their system, each peer resolves a DNS name, backed by CDNs such as Akamai, for several times, and reports the distribution of DNS redirections to a centralized server that provides the peer matching service. The authors of [4] hypothesize that peers observing similar redirection distribution are likely close to some replica server and thus to each other. The so-called “closeness” is defined by CDNs mainly for minimizing latency between a chosen replica server and the requesting client, which is may be different from reducing the inter-ISP traffic. In contrast, our algorithms defines precise objective, minimizing the cost on ISPs, and does not rely on CDNs, which are companies with completely different interests than ISPs. Moreover, we consider both the inter-ISP and the intra-ISP costs.

The P4P (provider portal for P2P) architecture [17] is a recent proposal that advocates the cooperation between ISPs and P2P applications. P4P requires the deployment of portal servers, called iTrackers, in various ISPs. The wide deployment of P4P may take years as pointed out by the authors of [17]. We take a different approach than P4P, where we leverage publicly available information and network topology inference algorithms. This enables us to provide efficient peer matching service *now* with *no costs* due to new infrastructure.

3 ISP-friendly Peer Matching Algorithms

3.1 Overview

To solve the matching problem for minimizing the cost on ISPs, we roughly divide the cost into two parts: inter-ISP cost and intra-ISP cost. We discuss these two costs in the following.

The inter-ISP cost refers to the amount of money a customer ISP pays its provider ISPs for transiting Internet traffic. Although ISPs hold different commercial agreements, the common charge model is based on the 95-percentile of the peak bandwidth usage of the inter-ISP links. Therefore, reducing inter-ISP traffic amount in general minimizes the

Figure 1. An illustrative example on network topology.

inter-ISP cost. In this paper, we do not consider the inter-ISP costs on individual ISPs. This is because ISPs consider their commercial agreements and network status as sensitive information, and will not reveal them. Therefore, we consider the inter-ISP cost on all ISPs as a whole. Furthermore, we consider costs, but ignore incomes due to the inter-ISP traffic. Indeed, provider ISPs *want* their customer ISPs to consume as much bandwidth as possible for higher revenues. However, the provider ISPs are financially much stronger. Therefore, a charge that is significant to customer ISPs is relatively negligible to provider ISPs. Hence, we believe that reducing the cost on customer ISPs is more critical.

To minimize inter-ISP traffic amount, the matching algorithms should return senders that are closer to the receiver in terms of *ISP distance*, which is the number of ISPs the data need to cross from the senders to reach the receiver. To illustrate, consider the topology in Fig. 1, in which receiver *R* needs to be matched with one of the potential senders: *S*₁ and *S*₂. Observe that sending a file from *S*₁ to *R* means every bit crosses inter-ISP links once, while sending a file from *S*₂ to *R* means every bit travels inter-ISP links twice. Clearly, matching *R* with *S*₁ reduces inter-ISP traffic amount by half and lowers the inter-ISP cost.

The intra-ISP cost refers to the amount of money an ISP spends to deploy and maintain its internal links. As shown in Fig. 1, an ISP (e.g., ISP B) may have multiple ASes, which are connected by sibling-to-sibling (s2s) links in the AS graph, while each AS (e.g., AS I) may have several PoPs (point-of-presents), which are connected by internal network links. A PoP is a set of routers owned by the same AS, and are located in the same geolocation, such as a city [6]. Both of these links are internal to ISPs. We believe that reducing the geographic distance the data have to travel from senders to the receiver in general minimizes the intra-ISP cost. This is because longer geographic distance means longer network links and more intermediate routers, which incur higher deployment and maintenance costs on ISPs. In Sec. 4, we experimentally show that some ASes have diameters close to 9,000 km. Considering such an AS as a single node clearly is not accurate. Therefore, we use PoP level network topologies to estimate the geographic distance between two peers and refer to this distance as *PoP distance*. We emphasize that data do not travel in a straight line from the sender to the receiver. Instead, a legitimate network path needs to be followed when computing PoP distance. To minimize the intra-ISP cost, our peer matching algorithms should return

Figure 2. ISP-friendly peer matching algorithms.

senders that are closer to the receiver in terms of the PoP distance. To illustrate, consider the topology in Fig. 1, in which receiver R needs to be matched to either S_2 or S_3 . While S_2 and S_3 are equally good in terms of ISP distance, selecting S_2 incurs higher intra-ISP costs than selecting S_3 , because sending a file from S_2 to R means every bit travels coast-to-coast. Clearly, matching R with S_2 reduces the distance the data have to travel and lowers the intra-ISP cost.

To develop ISP-friendly matching algorithms, we first need to infer the ISP and the PoP distance between any two peers. There are several methods to infer the distances, including sending probing packets and analyzing routing information. While sending probing packets might reveal some details on networks, it incurs high traffic overhead and is not scalable. Hence, we do not send probing packets. Instead, we leverage publicly available BGP and geolocation information to infer the ISP and the PoP distance between any two peers. We give details on the inference algorithms in Secs. 3.2 and 3.5.

These inference algorithms, however, are computationally complex, thus can not run in real-time. To cope with this excessive complexity, we propose a two-step approach as shown in Fig. 2. The peer matching algorithm builds *offline* a distance oracle that enables fast distance lookups between any two peers. Then, it uses this distance oracle *online* to efficiently match peers. In graph theory, a distance oracle refers to an intermediate data structure, constructed from a given graph, that supports very efficient distance lookups between any two vertices. There are several distance oracle designs in the literature [20]. However, these distance oracles are not suitable for estimating Internet path distance. This is because the Internet routes are not plain shortest routes: they rather follow certain known practice of BGP policy routing. For example, to find AS paths that conform to the routing policy, specialized AS path inference algorithms, such as [9], are required. Hence, we design our own distance oracle in Sec. 3.4.

Proposed matching algorithms. Given the inferred ISP and PoP distances, we propose two peer matching algorithms, ISPF and ISPF-Lite, to minimize the cost on ISPs. For every matching request, ISPF sorts the potential senders on their ISP distance to the given receiver, so that potential senders closer to that receiver are placed earlier in the list. On this sorted list, several potential senders may have the same ISP distance. To break these ties, ISPF sorts potential senders with the same ISP distance on their PoP distance, and places those with smaller PoP distance earlier than others with the same ISP distance. Once the sorted list is completed, the algorithm returns the first n senders in the

list, where n is a configurable parameter. We should mention that inferring PoP distance is computationally demanding because there are too many PoPs in the Internet. To ease this burden, ISPF-Lite sorts the tied potential senders on the length of IP prefix they share with the receiver. The rationale is that, the longer the shared IP prefix between two peers, the more likely that they are within the same network. We perform this longest prefix matching using a binary trie data structure, somewhat similar to the ones used for IP lookup in core routers. The main advantage of the matching based on IP prefixes is that it is almost *free*: it does not rely on any external information and runs fast.

3.2 Inferring ISP Distance

We estimate the ISP distance between two given peers using the following three steps. (i) We map the IPs of these two peers to their AS numbers. (ii) We infer the AS path between these two ASes. (iii) We count the number of inter-ISP links along the inferred AS path for the ISP distance. We present details on each step in the following.

First, we explain how we map an IP to its AS number. We achieve this by collecting public BGP tables, in which each entry contains a destination IP prefix and an AS path to that prefix. We map each IP prefix to the last AS on the AS list. The last AS is called the origin AS of the IP prefix. The mapping enables us to find the AS given any IP, and is also used in previous works such as [19].

To infer AS paths, we need to construct an AS graph. An AS graph consists of inter-AS links, where each link is annotated with an AS relationship that abstracts the commercial agreement between the two ASes. The AS relationships are classified into three categories: customer-to-provider (c2p), peer-to-peer (p2p), and sibling-to-sibling (s2s) [7, 5]. There are several inference algorithms, such as [5], that construct an AS graph using information about BGP tables and traceroute results. Given an AS graph, the AS relationship of each link describes the routing policy between the two ASes. An AS path that conforms to these routing policies is called *valley-free* [7, 8]. In this work, we propose optimization techniques to accelerate the modified Dijkstra (MD) algorithm [9, 8] that returns AS paths conforming to the policy routing practices. The proposed techniques are general enough and can work with other algorithms.

Last, we comment on how to derive the ISP distance given an AS path between two peers. We compute the ISP distance by counting the number of c2p and p2p links on a given AS path. We do not consider s2s links, because these links connect ASes belonging to the same ISP. We need to do this because large ISPs may have too many ASes, while transiting traffic among them incurs no inter-ISP cost.

3.3 Efficient AS Path Inference

Since there are currently more than 25,000 ASes in the Internet, the time complexity of calculating all-pair AS paths will be high because MD algorithm has time complexity $O(|V|^3)$ as proved in [8], where V represents all ASes. To reduce this complexity, we propose optimization techniques for efficient AS path inference.

We first observe that, according to the CAIDA AS relationship data, many ASes (83%) have only one or a few (c2p)

Figure 3. Paths are computed using comparisons.

Figure 4. Steps to infer all-pair AS paths.

links to their providers. We refer to these ASes as *stub* ASes, and other ASes as *transit* ASes. Traffic to/from a stub AS must go through one of its provider ASes, which is a transit AS. Therefore, given ISP distance between any two transit ASes, the ISP distance from any stub AS to any transit AS can be computed using simple comparisons, as shorter AS paths are preferred in BGP routing. To illustrate, Fig. 3 shows a stub AS S that has two provider (transit) ASes: T_1 and T_2 . The distance from T_1 to S equals the distance from T_2 to S . To compute the distance from another transit AS T_0 to S , we simply compare the *total* distances of crossing T_1 and T_2 . Since $3 + 1 < 5 + 1$, we infer that the distance from T_0 to S is 4. Notice that, in reality the comparisons are even simpler than in this illustrative example: we observe that stub ASes only have 1.76 provider ASes on average.

-
1. // Use BFS find the shortest uphill paths, which only
 1. // consist of c2p and s2s links.
 2. for each $s, t \in \mathbf{T}$, find the shortest uphill path
 3. // Use uphill path to compute valley-free path
 4. for each $s, t \in \mathbf{T}$, compute valley-free path
 5. // Compute valley-free path for all ASes
 6. for each $s \in \mathbf{T}$ and $t \in \mathbf{S}$ {
 7. compute valley-free path of s and t by finding u to
 7. $\min_{u \in s.\text{provider}} \{\text{valley-free distance of } u \text{ and } t\} + 1$
 8. }
 9. for each $s \in \mathbf{S}$ and $t \in \mathbf{S}$ {
 10. compute valley-free path of s and t by finding u to
 10. $\min_{u \in t.\text{provider}} \{\text{valley-free distance of } s \text{ and } u\} + 1$
 11. }
-

Figure 5. An efficient AS path inference algorithm.

This observation enables us to develop an efficient AS path inference algorithm as follows. Let $\mathbf{T}$ be the set of transit ASes, and $\mathbf{S}$ be the set of stub ASes. We infer all-pair AS paths in three steps as outlined in Fig. 4. We first compute the AS path between any two transit ASes by considering valley-free paths. We then consider the AS path between any two ASes, where exactly one of these two ASes is a transit AS. Finally, we consider the AS path between any two stub ASes.

The pseudocode is illustrated in Fig. 5. The time complexity of the loop in lines 2 and 4 is dominated by line 4, and is $O(|\mathbf{T}|^3)$. Notice that in lines 7 and 10, the degree of

ASes s and t can be considered as a constant, as we showed that the degrees of stub ASes are low. The time complexity of loops start from lines 6 and 9 are $O(|\mathbf{T}||\mathbf{S}|)$ and $O(|\mathbf{S}|^2)$, respectively. Thus, the time complexity of our AS path inference algorithm is $O(|\mathbf{T}|^3 + |\mathbf{T}||\mathbf{S}| + |\mathbf{S}|^2)$, which is considerably smaller than the time complexity $O(|\mathbf{V}|^3)$ of the MD algorithm, as $\mathbf{T}$ only consists of 18% of the ASes out of $\mathbf{V}$. In fact, we experimentally compared the running times of the proposed algorithm against that of the algorithm in [9] using a commodity PC. We observed a dramatic running time reduction: from almost two days to three hours.

3.4 Concise Distance Oracle

While the proposed AS path inference algorithm is quite efficient, it still can not run in real-time. To cope with this issue, we pre-compile an intermediate data structure offline, called distance oracle, to enable fast, $O(1)$ time, online ISP distance lookups. A simple distance oracle design is to employ a complete two-dimensional ISP distance matrix for *all* ASes in the Internet. This simple distance oracle, however, incurs high space overhead. For example, as there are more than 25,000 ASes in the Internet, allocating a complete distance matrix takes 625 MB of memory, assuming every distance is stored as a single byte. To mitigate this overhead, we propose to *only* store ISP distance between any two transit ASes. We then extend the ISP distance for stub ASes in on-demand basis while performing peer matching. This can be done by simple comparisons such as the illustrative example showed in Fig. 3. The comparisons incur a small, constant, overhead, because stub ASes have small degrees. Using the distance oracle enables us to reduce the memory consumption to about 10 MB of memory (a reduction of 97%) while still achieving $O(1)$ lookup time.

3.5 Inferring PoP Distance

To derive PoP distance between any two peers, we must infer the PoP topology. There are a few existing works that combine the traceroute utility and the reverse DNS lookups for inferring the PoP topology [6]. These algorithms, however, are not very accurate because they rely on the non-standardized DNS naming conventions. Moreover, these algorithms send probing packets, and thus incur too much traffic overhead. We propose a new PoP topology inference algorithm that combines several public information sources, including BGP tables, IP to geolocation databases, and BGP updates.

This new inference algorithm consists of the following steps. (i) It collects all IP prefixes from BGP tables, and organizes them based on their ASes. (ii) It uses an IP to geolocation database to cluster IP prefixes of each AS into one or more PoPs. Several PoPs of different ASes may exist at the same geolocation. (iii) It analyzes BGP updates to infer the connectivity among PoPs in the same AS. That is, it constructs a PoP topology for each AS. (iv) It combines the PoP topology with the AS graph derived in Sec. 3.2 for a complete, worldwide, PoP topology.

Due to space limitations, we only present an overview of the PoP inference algorithm. The inference algorithm uses the single-linkage clustering, an efficient pair-wise clustering method, to cluster prefixes of each AS into PoPs

based on their geolocations derived from an IP to geolocation database. It then constructs a PoP topology for every AS by analyzing the BGP updates to connect the PoPs that are frequently updated together. We note that the authors of [2] analyze BGP updates to cluster IP prefixes into a binary tree, while we combine BGP updates and IP to geolocation databases to construct the PoP topology. We are currently designing rigorous experiments to validate the accuracy of the proposed PoP topology inference algorithm.

4 Evaluation

We evaluate the ISP-friendly matching algorithms using trace-driven simulations. We have implemented ISPF and ISPF-Lite algorithms. For comparison, we have also implemented two other popular algorithms: the random algorithm (denoted as *Random*) that returns random senders and the AS-biased algorithm (denoted as *AS*) that chooses most of the senders within the same AS as the receiver [3]. We have also implemented an IP prefix matching (denoted as *Prefix*) that matches senders that share the longest IP prefix with the receiver. We note that we cannot compare our algorithms with Ono [4], because it requires modifications of the client software. In the experiments, we consider two performance metrics: the ISP distance and the PoP distance.

We collected real IPs from both live P2P networks and multimedia streaming servers. For P2P networks, we wrote a crawler to download more than 5,000 torrent files from Iso-Hunt, which is a famous torrent site. For each torrent file, we used a script to periodically request for potential senders during June 23–25, 2008. We then aggregated all the IPs together. We also got access logs of multimedia streaming servers from the CBC/Radio-Canada, which is the largest content provider in Canada. We constructed two sets of IPs: 160,543 and 147,872 unique IPs from CBC/Radio-Canada and BitTorrent, respectively. We repeated our experiments on these two sets of IPs.

We simulate multi-sender download sessions for files of different popularities. The popularity of a file is adjusted by controlling the number of peers that have that file. For a given file, we vary the number of peers storing it from 1% (rare file) to 10% (fairly popular file). For each download session, we randomly choose senders based on the chosen file popularity and a receiver from all considered IPs. Then n number of senders are chosen by peer matching algorithms. We compute the average ISP distance and PoP distance between the chosen senders and the receiver. We vary n between 10 and 100 to cover wide ranges of P2P systems, and for each value of n we repeat the experiment 100 times and report the average across them.

We plot sample results of the ISP distance in Fig. 6, and other results are similar. This figure shows that *Random* results in considerably higher ISP distance than other algorithms. In addition, *AS* performs worse than the ISP-friendly algorithms. This is because although *AS* selects most of the senders in the same AS as the receiver, it also randomly chooses senders outside the receiver’s AS. ISPF and ISPF-Lite algorithms leverage on the knowledge of the network topology and thus can achieve lower ISP distance. We note that the ISPF and ISPF-Lite dramatically decrease the

ISP distance to less than one. This is because they return peers within the receiver’s ISP in most download sessions. Therefore, it significantly reduces the load on the expensive inter-ISP links. Moreover, ISPF and ISPF-Lite also yield better application-level performance because close-by peers usually have short and less variable round trip delays. Finally, we observe that *Prefix* outperforms *Random* by almost two times, which is nontrivial considering *Prefix* is almost free.

Next, we plot a sample result of the PoP distance in Fig. 7, and other results are similar. This figure shows that using ISPF can significantly reduce the PoP distance of *Random* by up to 7,000 km, while ISPF-Lite results in similar PoP distance as *AS*. We note that ISPF-Lite significantly outperforms *AS* in terms of the ISP distance (see Fig. 6), which is more critical than the PoP distance. Last, *Prefix* still outperforms *Random* in terms of the PoP distance.

To summarize, our experimental results show that *Prefix* outperforms *Random* at almost no extra expense. Both ISPF and ISPF-Lite perform better than *AS* in terms of the inter-ISP cost. While ISPF further reduces the intra-ISP cost, it incurs higher computational complexity. Therefore, we recommend ISPF when resources are not concerned. For more resource constrained systems, ISPF-Lite is recommended. Finally, if there is almost no additional resource available, *Prefix* can be used to boost performance over *Random*.

Last, we briefly comment on how we inferred the ISP and the PoP distances, due to space limitations. We used the GeoIP [10] database and the inference algorithms described in Sec. 3 to infer the ISP and the PoP distances. The BGP tables and updates on June 23, 2008 were downloaded from RouteViews [12] and RIPE-NCC [1]. We obtained the AS relationships from the CAIDA web site. With the proposed optimization techniques, we generated the distance oracle for ISP distance in 146 minutes on a commodity PC. This is the most time consuming part in the inference procedure: all other steps terminated in a few minutes.

5 Conclusions

In P2P systems, a receiving peer needs to be matched with multiple sending peers because peers have limited capacity and reliability. Efficient peer matching is important to both ISPs and P2P systems. We studied the following peer-matching problem: given a set of potential senders for an object requested by a receiver, find the subset of senders that will minimize the cost on ISPs. We proposed two ISP-friendly algorithms, ISPF and ISPF-Lite, that leverage publicly available information, such as BGP tables, to infer the network topology for optimizing peer matching.

We evaluated the proposed matching algorithms using trace-driven simulations. We considered two sets of traces: from BitTorrent networks and from CBC/Radio-Canada video streaming servers. In both cases, our experimental results indicate that a significant reduction on the number of ISPs that P2P traffic has to traverse from senders to the receiver is achieved. The proposed algorithms outperform the current random matching algorithm by a large margin: as high as 6 times. We are currently implementing the proposed matching algorithms in a real P2P network, which is

Figure 6. ISP distance under: (a) different number of chosen senders and (b) different object popularity.

Figure 7. PoP distance under different number of chosen senders. 10% popularity, BitTorrent.

a collaborative project between our group and CBC/Radio-Canada. This P2P network will be released to the public in the near future.

6 References

- [1] RIPE Network Coordination Project. <http://www.ripe.net/>.
- [2] D. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan. Topology inference from BGP routing dynamics. In *Proc. of ACM SIGCOMM Internet Measurement Workshop (IMC'04)*, pages 243–248, Marseille, France, November 2002.
- [3] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in BitTorrent via biased neighbor selection. In *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, pages 66–74, Lisboa, Portugal, July 2006.
- [4] D. Choffnes and F. Bustamante. Taming the Torrent: A practical approach to reducing cross-ISP traffic in peer-to-peer systems. In *Proc. of ACM SIGCOMM'08*, pages 363–374, Seattle, WA, August 2008.
- [5] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. H. K. Claffy, and G. Riley. AS relationships: Inference and validation. *ACM SIGCOMM Computer Communication Review (CCR'06)*, 37(1):29–40, January 2007.
- [6] B. Donnet and T. Friedman. Internet topology discovery: a survey. *IEEE Communications Surveys & Tutorials*, 9(4):56–69, 4th Quarter 2007.
- [7] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, December 2001.
- [8] L. Gao and F. Wang. The extent of AS path inflation by routing policies. In *Proc. of IEEE Global Telecommunications Conference (GLOBECOM'02)*, pages 2180–2184, Taipei, Taiwan, November 2002.
- [9] Z. Mao, L. Qiu, J. Wang, and Y. Zhang. On AS-level path inference. In *Proc. ACM SIGMETRICS'05*, pages 38–49, Banff, Canada, June 2005.
- [10] GeoIP Web Page. <http://www.maxmind.com/app/ip-location>.
- [11] W. Norton. The evolution of the U.S. Internet peering ecosystem. <http://www.nanog.org/mtg-0405/norton.html>, November 2003. White Paper.
- [12] U. of Oregon Route Views Project. <http://antc.uoregon.edu/route-views/>.
- [13] P-Cube Web Page. <http://www.p-cube.net/indexold.shtml>.
- [14] Packeteer Web Page. <http://www.packeteer.com/>.
- [15] P. Rodriguez, S. Tan, and C. Gkantsidis. On the feasibility of commercial, legal P2P content distribution. *ACM SIGCOMM Computer Communication Review (CCR'06)*, 36(1):75–78, January 2006.
- [16] Y. Tu, J. Sun, M. Hefeeda, and S. Prabhakar. An analytical study of peer-to-peer media streaming systems. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 1(4):354–376, November 2005.
- [17] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4P: Portal for (P2P) applications. In *Proc. of ACM SIGCOMM'08*, pages 351–362, Seattle, WA, August 2008.
- [18] D. Xu, S. Kulkarni, C. Rosenberg, and H. Chai. Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution. *ACM/Springer Multimedia Systems Journal*, 11(4):383–399, April 2006.
- [19] M. Zhang, Y. Ruan, V. Pai, and J. Rexford. How DNS misnaming distorts Internet topology mapping. In *Proc. of USENIX Annual Technical Conference (USENIX'06)*, pages 369–374, Boston, MA, May 2006.
- [20] U. Zwick. Exact and approximate distances in graphs—a survey. In *Proc. of Annual European Symposium on Algorithm (ESA'01)*, pages 33–48, Aarhus, Denmark, August 2001.