ALTO WG
Internet-Draft
Intended status: Experimental
Expires: April 20, 2011

Y. Yang
Yale University
R. Alimi
Google
Y. Wang
Yale University
D. Zhang
PPLive
K. Lee
China Telecom
October 17, 2010

Tracker-Based Peer Selection using ALTO Map Information
draft-yang-tracker-peer-selection-00.txt

Abstract

As ALTO core information starts to become available from some ISPs, how to effectively utilize such information by P2P applications has become a major issue. In this document, we discuss some techniques that a P2P application tracker can incorporate ALTO information in initial peer selection.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

Status of this Memo

Table of Contents

1. Introduction

ALTO provides information to network applications to improve network efficiency [2]. There are many ways that a network application can utilize ALTO information. For example, an application may choose to utilize only the Network Map, another may use both the Network Map and the Cost Map, and yet another may use the Endpoint ranking. It can be either a more centralized entity such as a tracker in a P2P application or the P2P clients that utilize ALTO information. One P2P application may choose to use the information at the P2P clients during piece selection or rate scheduling, while another P2P application may use it during peer selection. How to effectively utilize ALTO information is a challenge that P2P applications considering integrating with ALTO need to address.

In this document, we present example techniques of how to integrate ALTO information into the peer selection process at a P2P tracker. We first present some key challenges. We then present some techniques used in real trial examples that address the challenges.

2. Challenges

A P2P tracker selects a set of peers upon receiving a LISTING request of a peer. Since a tracker may receive a large number of such LISTING requests, it is important that the tracker can handle each request with high efficiency while achieving effectiveness in utilizing available information. The design of the data structures and algorithms at the tracker for peer selection is challenging and can have a major impact on the efficiency and effectiveness of peer selection.

Specifically, there are two challenges in integrating ALTO information into the peer selection of a P2P tracker.

o Scalability: A P2P developer may have a small number of trackers to handle a large number of channels (files) each with multiple peers. The peers might be distributed across multiple ISPs that provide ALTO information. Thus, the storage and processing overhead caused by using ALTO information must be considered in order to scale to the increasingly larger P2P applications. In addition, it may be necessary to scale the tracker of a particularly popular channel from a single machine to multiple machines. In practice, many P2P applications may use multiple physical P2P trackers for a single channel for fault tolerance (e.g., when one tracker crashes) and/or connectivity reasons (e.g., poor connectivity between networks).

- o Application-Network Information Fusion: When selecting peers, a tracker should consider not only ALTO information, but also peer properties known only to the application (e.g., instantaneous peer upload capacity) as well as application requirements. In particular, a key concern of a P2P application is that solely considering ALTO information may lead to degraded application performance (e.g., slower download rate in a P2P file sharing application.)

One of the simplest ways to implement peer selection is random peer selection using a single array storing all current peers. Upon receiving a LISTING request, the tracker picks a random position in the array, and returns a set of peers starting from the chosen position. A slightly different random peer selection algorithm is to repeatedly pick random numbers in the range of the size of the array to pick multiple random peers.

An advantage of the preceding algorithm is scalability. But it is lacking in network-application information fusion. It does not consider peer properties during peer selection. On the other hand, many P2P trackers already select peers considering peer properties. For example, one type of peer property often considered is peer upload capabilities. Another type of peer property is the playpoint of a peer, in particular, in an VoD setting. Also, in addition to using ALTO information, some existing P2P trackers already consider network location properties such as the ASN, the IP prefix, the geo location (e.g., city, country or latitude/longitude), or the set of nearest landmarks of a peer.

3. Peer Classification Data Structures

When peers are annotated with properties, we might envision that the peers are stored at a tracker in a table similar in format to Figure 1:

peer_id	IP	upld_cap	play_point	ASN	country	cty	..
...	...	...	...	...	...	...	..

Figure 1: Using a Table to Store Peers

A problem of a flat table is that it does not support peer classification to find peers with given properties. Just as many databases build indices, many P2P trackers build inverted data structures such as map/hash in order to index to the pool of peers with a given property.

In an abstract formulation, for each peer A requesting LISTING, the peer selection algorithm at the tracker determines a probability that any peer B will be returned to A, where the probability depends on the relative "match" between the properties of A and B. However, too much fine-grained tuning of the probabilities (there are O(N^2) such values, where N is number of peers) may not be necessary or feasible. Peer classification is a technique to aggregate peers into equivalent classes before peer selection to improve scalability.

3.1. One-Level Key Partitioning

Multiple examples exist in this category. In one example, the key is a category of the uploading capacity of a peer. For example, the tracker may classify peers as having high upload capacity, medium upload capacity, or low upload capacity. Then according to the property of the peer issuing the LISTING request, the tracker selects fractions of peers from each category.

In another example, the key can be the ASN or ISP name. When the tracker receives a LISTING request from a peer, the tracker looks up the ASN/ISP of the peer, and indexes to the ASN/ISP to select peers. To avoid partition of the P2P topology or when there are not sufficient numbers of peers in the ASN/ISP, the tracker may select peers from other ASNs/ISPs.

3.2. Hierarchical Partitioning

In addition to a single level flat map, some trackers classify peers using multiple attributes and/or build multiple levels of indexing, utilizing a hierarchical partitioning of peers according to peer properties.

One example is to use the hierarchical geo partitioning of peers first into country, then state/province, and then city.

Utilizing the Network Map of ALTO, the tracker can classify each peer into the PID of each ISP providing ALTO Network Map.

Extending the preceding examples of using one type of peer property, the partition keys at different levels can be from different categories. For example, at the first level, the tracker might use peer upload capacity, the next level uses ISP as the key, and the third level uses PID.

3.3. Comments

There are several comments. First, a tracker may partition peers from multiple perspectives. For example, each ISP providing ALTO Network Map provides a classification of peers into its set of PIDs. Thus, a single IP address may belong to different PIDs from different Network Maps of different ISPs. The tracker may build a classification tree for each ISP. It is possible that these multiple trees can be merged into a single tree with a dummy ROOT. We use a single classification tree as an example.

Second, the nodes of different non-overlapping branches may overlap regarding the sets of peers contained in them.

Second, it may not be straightforward or necessary to partition peers using certain properties. For example, landmarks may not lead to easy partitioning of peers.

4. Peer Selection Using Peer Classification

4.1. Overview of Scheme

We now look at a class of tracker peer selection techniques that utilize peer classification. Each peer is located at a leaf node of a classification tree. We consider the set of algorithms where the peer selection depends on the properties of the peer issuing the request.

We introduce a concept called the "home" node of a peer issuing the LIST request. The home node is identified first before peer selection.

The peer selection is specified in the following way. Associated with each "home" leaf node of the classification tree is an ordered list, where each element in the list contains two fields: the first is a pointer to a node in a peer classification tree, and the second indicates how and how much to select peers from the node. It is important to notice that the list is ordered as the tracker picks peers in order. It is straightforward to extend that the nodes may come from multiple classification trees.

Figure 2 is an example. We refer to the data structure containing the selected peers as the bucket. The example specifies that when a peer A with "home" leaf node at n4 (high capacity peers from ISP1) issues a LISTING request, first fill 50% of the bucket containing peers to be returned to A from n4 (high capacity peers from same ISP) or no more peers available from n4, then continue to fill the bucket by choosing peers from n5 (low capacity peers from same ISP) until the bucket is 80% full or no peers available from n5, then continue to fill the bucket by picking peers from n2 (peers from ISP2) so that the bucket can be 95% full, and finally fill the remaining of the bucket from n3. Note that the scheme intends that for n4, to pick more from the same ISP (80%), then ISP2 (15%), and then ISP3. It also tries to pair high capacity peers more with high capacity peers.

```
                       ROOT
                     /   |    \
                   /     |      \
            ISP1 /  ISP2 |  ISP3 \
               /         |        \
             n1          n2        n3
           /  |         /  \      /   \
         /    |        |    \     |    \
       HC     LC      HC    LC   HC    LC
       n4     n5      n6    n7   n8    n9

leaf n4: [n4, 50%] [n5, 80%] [n2, 95%] [n3, 100%]
leaf n5: [n4, 20%] [n5, 60%] [n2, 95%] [n3, 100%]
...
leaf n9: ...
```

Figure 2: Example: Peer Selection using Classification Tree.

4.2. Extensions and Issues

The preceding peer selection scheme is simple and flexible. It can be efficiently implemented. There can be multiple ways to extend the scheme.

There are two remaining issues:

o First, how to design the classification tree?

o Second, how to create the traversal list of each leaf node?

In addition to addressing the two preceding issues, this scheme may not be a general representation of some existing peer selection schemes. For example, when the network location of a peer is represented by its set of close-by landmarks, a straightforward partition tree may not exist. Instead, some other data structures and algorithms may be needed to pick the peers that are the closest measured by a special metric space measured by "closeness" of landmark sets.

5. Peering Matrix

5.1. Overview

The Peering Matrix approach is an instance of the scheme of Peer Selection Using Partition Tree. It has been used in several trials, including the Pando/Comcast trial [3]. The scheme has also been evaluated in the context of P2P Live Streaming. Below, we present more details on Peering Matrix. We also briefly summarize a set of results applying Peering Matrix to P2P Live streaming on the Planetlab.

5.2. Partition Tree

Recall that we name the peer issuing the LISTING request as A. There is a unique path Path(A) going up from the leaf node containing A to ROOT in the partition tree. We consider the class of tracker peer selection algorithms that specify a (upper bound) target fraction of peers to be selected at each node n along Path(A). The peer selection algorithm goes up along Path(A). To be consistent, the fraction value at a node n should be no larger than that of the parent of n named Parent(n).

Also, at each node n along Path(A), for each child c of n, the peer selection algorithm specifies how peers are distributed among the siblings of c.

Consider an example in Figure 3:

```
                         ISP1
                       / |   \  \
                     /   |    \   \
                   /     |     \    \
                 /       |      \     \
               PID1    PID2   PID3 ... PIDn
```

Figure 3: Example: Using ALTO Network Map for building a Classification Tree.

Specifically, Figure 3 is a two level classification tree for an ISP, and each second level node represents a PID of the ISP. Each PID is labeled with Fraction = 75%. The ROOT has a fraction of 100%. The sibling distribution of node PID1 node is 50%, 30%, 20% to PID2, PID3, and PID4 respectively. This means that when a peer A from PID1 asks for a list of peers, the tracker selects up to 75% peers at PID1, and fills the remaining (25%) at PID2 (up to 25% * 50%), PID3 (25% * 30%), and PID4 (25% * 20%).

During P4P trials, we used a three level partition tree for each ISP.

```
                            ROOT
                          / |   \  \
                        /   |    \   \
                      /     |     \    \
                    /       |      \     \
                  intra   ePID1  ePID2 ... ePIDn
               /   |    \
              /    |     \
           iPID1 iPID2 ...iPID
```

Figure 4: Three-Level Peer Classification.

One nice property of using per ISP classification tree is to implement a distributed tracker, where a tracker is responsible for the peers within a set of ISPs. A peer may request LISTING from multiple trackers (e.g., located at different ISPs) that together are responsible for the channel. The tracker hosting the "home" leaf of the peer uses peering matrix, while the other trackers return a small number of random peers for robustness.

5.3. Computing the Peering Matrix: Bandwidth Matching

To compute the sibling distribution at node intra and ROOT, the tracker estimates the aggregated upload capacity (a seed can use full upload capacity and a leecher achieves 70%) and demand of each PID and then conducts bandwidth matching as specified in [4].

Specifically, The following diagram shows how the information flow as well as how to transform ALTO Network Maps and Cost Maps into peering matrix, considering application states.

```
                   Network               [App-specific
      .----------.   Map  .-------------.   state]   .-------------.
      | ALTO     | <----- | Peering     | <--------- |             |
      | Services |        | Matrix      |            | Application |
      |          | -----> | Computation | ---------> |             |
      ‘----------’  Cost  ‘-------------’App-specific‘-------------’
                    Map                   or generic
                                           guidance
```

Figure 5: Information Flow to Compute Peering Matrix.

The interface to the Peering Matrix Computation Component, for a BitTorrent like file sharing application can be:

GetPeeringWeights: The request optionally includes swarm state information as a list of PIDs, and for each PID, the number of seeds and leechers and the aggregated download and upload capacity of clients within the PID. The response is a matrix of peering weights amongst the PIDs included in the request, as computed from the set of Costs currently pulled from the ALTO Server. If the request included swarm information, the returned weight matrix is tailored for the current state of the swarm.

5.4. Computing the Peering Matrix: Generic

Similar to the preceding, but instead of using estimated capacity and demand, it assumes that each PID has one peer.

5.5. Live Streaming Results Using Planetlab

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

This document does not evaluate security considerations. Multiple other documents in the ALTO working group considers the security perspective of using ALTO information.

8. References

8.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[2] Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-03 (work in progress), March 2010.

[3] Griffiths, C., Livingood, J., Popkin, L., Woundy, R., and Y. Yang, "Comcast's ISP Experiences in a Proactive Network Provider Participation for P2P (P4P) Technical Trial", RFC 5632, September 2009.

[4] H. Xie, Y.R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz., "P4P:", In SIGCOMM 2008.

Appendix A. Acknowledgments

This document benefits substantially from trials and designs involving P2P applications Pando (Laird Pasko), PPLive (David Zhang), Digimeld (Gene Qin), and Xunlei. The data structure designs and algorithms include the contributions of Hao Wang, Ye Wang, and Harry Liu. We appreciate their contributions. The design is quite similar to that of Xunlei, whose more details will be included in the updated China Telecom trial document.

Authors' Addresses

Y. Richard Yang
Yale University

Email: yry@cs.yale.edu

Richard Alimi
Google

Email: ralimi@google.com

Ye Wang
Yale University

Email: ye.wang@yale.edu

David Zhang
PPLive

Email: davidzhang@pplive.com

Kai Lee
China Telecom

Email: leek4u@gmail.com