Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: June 18, 2011

H. Asai
H. Esaki
U. Tokyo
T. Momose
Cisco Systems

December 15, 2010

A Solution Approach for AS Relationships-aware Overlay Routing
draft-asai-cross-domain-overlay-01

Abstract

This document provides an idea of cross-domain traffic control in overlay routing such as peer-to-peer content delivery networks to reduce transit traffic that costs more for Internet service providers. The simulation results in this document show the advantage of AS relationships-aware overlay routing and cross-domain cooperation. This document also proposes a solution approach to take into account AS relationships for overlay routing, with hiding confidential information as much as possible.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 18, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Peer-to-peer (P2P) technologies have been introduced into many systems such as content delivery networks (CDNs) and video streaming systems. These P2P technologies have enabled to avoid excessive server load and to achieve effective and high-quality communication (e.g., high throughput, fault tolerance). Today, the traffic generated by P2P applications become a significant amount of the Internet traffic [RFC5693]. Since P2P applications construct their own network topologies over the Internet without taking into account the network layer topology (i.e., layer 3 topology), these P2P applications frequently utilize a larger amount of network resources than network providers expect. Since cross-domain links, especially transit links, are generally expensive than intra-domain links, this document focuses on cross-domain traffic.

This document provides an idea of cross-domain traffic control in overlay routing to reduce transit traffic that costs more for network providers such as Internet service providers (ISPs). Our simulation results show that P2P CDNs that are unaware of commercial relationships between autonomous systems (ASes) [RFC1930] utilize transit links more, and consequently, it is required to take into account commercial relationships between ASes. This document also provides a solution approach to take into account commercial relationships between ASes, with hiding confidential information as much as possible.

1.1. Terminology

We use the following terms in this document.

1.1.1. AS

Autonomous System

1.1.2. AS Relationships

AS relationships represent commercial relationships between interconnected ASes. AS relationships are categorized into two major types: transit and peering.

1.1.3. Transit

Transit is a type of AS relationships. Transit relationships are also called provider-customer relationships. A customer AS purchases Internet access from its transit providers over transit links by paying some amount of money according to the actual bandwidth usage.

1.1.4. Peering

Peering is a type of AS relationships, and the relationships between two peering ASes are equal relationships. Traffic exchanged over peering links is free of charge.

1.1.5. Overlay Network

Overlay networks are constructed by application-layer nodes such as P2P application nodes over the Internet (i.e., IP network) that is operated by network providers. The topology and routing of overlay networks are controlled by applications that construct overlay networks but not by network providers.

1.2. 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 [RFC2119].

2. Problem Statement

The Internet consists of thousands of ASes operated by distinct network providers such as commercial ISPs, companies and universities. Each AS generally connects with multiple ASes, and there are distinct charging policies for each inter-AS link. These charging policies are roughly categorized into two major types of relationships; transit (with charge) and peering (without any charge). From the economical viewpoint, network providers want to reduce the traffic volume exchanged with transit providers as much as possible, and consequently, they manage the routing policies as explained in [Wang03].

However, overlay networks sometimes break these routing policies and cause problems with cross-domain traffic. We summarize the problems with overlay networks as follows.

- o The cross-domain traffic generated by applications are neither controlled nor optimized on overlay networks.
- o ASes hardly cooperate with each other in computing and fairly balancing cost when ASes provide some cost information to applications as a traffic optimization metric because charging policies are complicated and each AS operates its network autonomously.
- o Neither AS relationships nor charging policies for transit traffic can be disclosed.

2.1. Cross-Domain Traffic of Overlay Networks

Network providers cannot control nor optimize the cross-domain traffic generated by applications on overlay networks. This is because the traffic is controlled by a set of application-specific algorithms that determines overlay network topologies and traffic delivery paths such as peer/neighbor/path selection algorithms.

```
                    +------+ provider
                    | AS 1 |----------------------+
           provider +------+                      | transit
                       | transit                  |
                       v                          v
           customer +------+ peering +------+  +------+ customer
                    | AS 2 |<------->| AS 3 |  | AS 4 |
                    +------+         +------+  +------+
```

AS 2 purchases Internet access from AS 1 via a transit link. On the contrary, the link between AS 2 and AS 3 is peering, which is lower cost link from the viewpoint of AS 2 network operators.

Figure 1: An example of AS-level topology with AS relationships

We show an example of the problem with cross-domain traffic of overlay networks. An example of interconnections of ASes and their relationships is shown in Figure 1. Suppose server nodes, nodes that provide a certain content file, exist in both AS 3 and AS 4 and a client node, a node that downloads the file, in AS 2 is to retrieve the file from one of these server nodes, the client node should select a server node in AS 3 to reduce transit charge for both the client-node-side and server-node-side ASes, but today’s client nodes that are unaware of AS relationships often select other server nodes.

Moreover, on overlay networks, the connectivity of end-point nodes (i.e., peers) is provided by residential ISPs and most of them are not transit providers but transit customers. Therefore, it is significantly important to control the transit traffic not to increase their charge to their providers though these kinds of application-layer traffic are hardly controlled by ISPs.

[RFC5693] also claims this problem with cross-domain traffic in terms of transit cost as well as congestion in intra-domain networks.

2.2. Cross-Domain Cooperation

```
                    +------+ provider
                    | AS 1 |----------------------+
           provider +------+ 1                    | transit
                     5 | transit                  |
                    30 v                          v 10
           customer +------+ peering +------+  +------+ customer
                    | AS 2 |<------->| AS 3 |  | AS 4 |
                    +------+ 10   20 +------+  +------+

                    Each number represents egress cost.
```

Figure 2: An example of unfair cost setting

The ALTO Working Group has worked on application-layer traffic optimization, and it has proposed a protocol to provide end-to-end cost between peers [I-D.ietf-alto-protocol]. Cost computation of this protocol is based on P4P [Xie08] that is an oracle-based approach of application-layer traffic optimization. The P4P approach has achieved fair utilization of network resources by setting up priorities automatically computed from the configuration (e.g., ‘cost’ in OSPF) in routers to links.

However, there is a problem with these oracle-based approaches when they are applied to the Internet (i.e., multi-domain system). Charging policies for inter-AS links and exchanged traffic volume are so complicated that different ASes hardly cooperate with each other in computing and fairly balancing cost. This is because each AS aims to maximize its income and minimize its expense. This problem is similar to so-called hot-potato problems. For example, suppose egress cost of each inter-AS link is configured autonomously (i.e., each AS sets cost according to its own policies) as shown in Figure 2, then the accumulated cost of the path from AS 4 to AS 2 becomes larger than that of the path from AS 3 to AS 2 though the path from AS 3 to AS 2 seems to be better than the other. On the other hand, when we consider ingress cost setting, cost on the source is ignored. Thus, oracle-based approaches hardly achieve fair traffic optimization among multiple autonomous domains because the Internet is autonomously operated by each AS.

2.3. Non-disclosure AS Relationships

To enable AS relationships-aware overlay routing, applications should take into account AS relationships or charging policies among ASes. So, cross-domain cost is required to be unveiled or estimated, and provided. The ALTO protocol [I-D.ietf-alto-protocol] provides end-to-end cost based on P4P [Xie08], but it does not mention how to cooperate with each AS.

Interconnections between ASes are established by commercial contracts, and consequently, most ISPs cannot disclose their commercial relationships. Hence, there is a difficulty in applying the approach of cross-domain cost computation in P4P to the real Internet because issues on disclosing topology information such as confidential commercial contracts lie upon it. Even though ASes can exchange the cost of cross-domain links, the problem with cross-domain cooperation described in Section 2.2 still exists.

2.4. Problems with the ALTO Approach

	Problems
Cooperation	Require to set coordinated cost onto inter-AS links, but each AS is autonomously operated
Security	Require to exchange the cross-domain cost among ALTO services, i.e., require to disclose AS relationships, though AS relationships are non-disclosure ones

Table 1: Problems with the ALTO approach

The ALTO approach mainly defines intra-domain traffic optimization, and consequently, it does not focus on the cross-domain cooperation. We summarize problems with the ALTO approach in Table 1.

3. Simulation Results: Oracle-based Naive Approach

To point out the problems with cross-domain traffic and cooperation, we evaluate cross-domain traffic of a P2P CDN with a trace-driven simulation.

We had collected a list of peers from a tracker (http://bttracker.debian.org:6969/announce) every minute from 23/10/2009 to 19/12/2009 for the content: Debian Linux DVD image; debian-503-i386-DVD-1.iso (4.4GB). The collected list contains sets of peer’s IP address and port number. We generated a trace for the trace-driven simulation according to the method described in [Asai10-1]. By using a trace-driven simulator [Asai10-1] with this trace, we compute exchanged cross-domain traffic volume of ASes providing the Internet connectivity to peers. Note that the piece size is set to 1 in this simulation and other parameters follow [Asai10-1].

We evaluate five oracle-based peer selection algorithms in the P2P CDN; 1) Random, 2) AS hops, 3) Selfish, 4) Gentle, and 5) Cooperative. ‘‘Random’’ and ‘‘AS hops’’ are algorithms to randomly select a peer and to select a peer minimizing AS hops between source and destination, respectively. ‘‘Selfish’’ is an algorithm to select a peer minimizing expense of ASes accommodating download peers (based on a download-side policy); i.e., ‘‘intra-domain’’ is the highest priority, followed by ‘‘from customer’’, ‘‘from peer’’ and ‘‘from provider’’. ‘‘Gentle’’ is an algorithm to select a peer maximizing profit of ASes accommodating upload peers (based on an upload-side policy); i.e., ‘‘intra-domain’’ is the highest priority, followed by ‘‘to customer’’, ‘‘to peer’’ and ‘‘to provider’’. ‘‘Cooperative’’ is the intermediate between ‘‘Selfish’’ and ‘‘Gentle’’; i.e., to select a peer minimizing the summation of cost of both download- and upload-sides where the cost values of intra-domain, from/to provider, from/to peer, and from/to customer are 0, 3, 2, 1, respectively.

Algorithm	From providers	From customers	From peers
Random	96.8%	0.4%	2.7%
AS hops	90.2%	4.9%	4.9%
Selfish	89.3%	8.8%	1.9%
Gentle	96.5%	0.0%	3.4%
Cooperative	88.9%	5.6%	5.5%

Table 2: Simulation Results: Breakdown of total exchanged cross-domain traffic volume of ASes accommodating peers by types of AS relationships (Download traffic)

Algorithm	To providers	To customers	To peers
Random	61.0%	24.8%	14.2%
AS hops	62.0%	19.7%	18.3%
Selfish	63.6%	12.8%	23.6%
Gentle	7.4%	83.2%	9.4%
Cooperative	11.4%	79.4%	9.3%

Table 3: Simulation Results: Breakdown of total exchanged cross-domain traffic volume of ASes accommodating peers by types of AS relationships (Upload traffic)

We show the breakdown of total exchanged cross-domain traffic volume of ASes accommodating peers by types of AS relationships in Table 2 and Table 3. These results show that even the algorithm Selfish did not achieve to reduce transit traffic from providers much, and consequently, it is difficult to reduce much download transit traffic. On the other hand, for upload traffic, algorithms Gentle and Cooperative significantly reduced transit traffic to providers. These results also indicate that the algorithm Cooperative worked quite well for both download and upload traffic though algorithms Selfish and Gentle were not good for either download or upload traffic. Therefore, traffic control with cooperation between download- and upload-sides is required for transit traffic reduction.

Note that further evaluation and results (e.g., with other traces and evaluation parameters) should be given in future.

4. Solution Approach

This section describes an approach to solve the problems which have been figured out in Section 2 and Section 3, and the requirements. In this approach, the cost computation for AS paths between any two ASes consists of three services, as follows.

1. AS path provision service: This service provides AS paths between two arbitrary nodes (IP addresses) to applications, and it is provided by each AS.

2. AS relationships estimation service: This service provides AS relationships (i.e., cross-domain cost) of any specified inter-AS links to applications. This service is provided by its service providers which may not be ISPs but some other volunteer service providers.

3. Cost computation service: This service computes cost for an AS path according to an algorithm, and it is installed into each application. The computed cost for the AS path would be used for traffic optimization.

It is true that AS paths can be resolved from IP address-based paths, which can be retrieved by network management tools (e.g., traceroute). Hence, ASes do not have to provide AS paths because applications can resolve AS paths without support of ASes. However, it is an ongoing work to assign appropriate AS numbers to routers [Huffaker10]. Moreover, some ISPs block ICMP packets including ICMP time exceed messages, and consequently, IP address-based paths are not always resolved. Therefore, provision of AS paths by ASes is enough helpful to resolve AS paths and use them for AS relationships-aware application-layer traffic optimization. Thus, it is recommended for each AS to be implemented. A method for providing AS path to applications is described in Section 4.1.

This approach aims to hide information on AS relationships as much as possible; i.e., not to disclose AS relationships. So, it uses AS relationships estimated from publicly available information instead of AS relationships which are to be disclosed by ASes. This document provide one possible estimation method and the detailed description follows in Section 4.2.

Cost for a path which would be used for the traffic optimization is computed from the estimated AS relationships by a certain algorithm. This document does not define any specific algorithms but provides an example as an idea in Section 4.3.

Service	Service provider	Requirement level
* AS path provision	Each AS	RECOMMENDED
* AS relationships estimation	Volunteer service providers etc.	REQUIRED
* Cost computation	Each application	REQUIRED

Table 4: Requirements

These services and the requirements of this approach are summarized in Table 4. AS relationships estimation and cost computation services are REQUIRED ones for taking into account AS relationships. AS path provision service is not mandatory but recommended one because this function can be alternated by other mechanisms.

4.1. AS Path Provision Service

```
                  +--------------+
                  | Applications |
                  +--------------+
                       |    ^                <at customer's network>
     AS paths request  |    | AS paths response
       (src X, dst Y)  |    |   (AS path from X to Y)
     - -- -- -- -- --  | -- | -- -- -- -- -- -- -- -- -- -- -- -- -
                       v    |                               <at ISP>
            +----------------------------+
            | AS path provision service |
            |  +---------------------+  |
            |  | Interface w/ filter |  |
            |  +---------------------+  |
            |          |    ^           |
            |          |    |           |
            |          v    |           |                 _______
            |    +-----------------+    |  routing table   /       \
            |    | AS paths table  |<----------------------*  BGP   *
            |    | lookup function |    |  w/ AS paths    * router *
            |    +-----------------+    |                  \_______/
            +----------------------------+
```

Figure 3: System overview of AS path provision service

As described above, AS path provision by ASes helps applications to resolve AS paths. Here, note that AS paths can be easily retrieved from BGP routing tables at ASes’ BGP routers. The overview of the AS path provision system is shown in Figure 3. The requirements of AS path provision are listed below.

- AS path provision service discovery: A mechanism which enables applications to discover AS path provision servers is required. (This document does not define this service discovery protocol.)

- AS path provision service: A mechanism which enables applications to resolve AS path from an IP address belonging to the AS which provides the AS path provision service to another IP address belonging to an arbitrary AS via a certain protocol is required. Here, note that the source IP address of the request must belong to the AS providing the service because AS paths are retrieved from routing tables in BGP routers and a routing table has a spanning tree from the AS as root (i.e., the source AS). (This document does not define the protocol.)

- Filter: AS path provision services can deny some requests by a filter to hide their information.

4.2. AS Relationships Estimation Service

Several AS relationships inference or estimation algorithms have been proposed in the research field. There are two types of these algorithms; one is based on paths analysis [Gao01] and the other is based on differences in AS’ network size [Asai10-2].

The algorithms based on paths analysis have a difficulty in applying the inferred AS relationships to the cost computation because there are lots of missing links, which have not been inferred. The algorithms based on differences in AS’ network size first quantify the network size, then estimate the relationships. Therefore, the relationships can be estimated for almost all links because one BGP routing table contains almost all ASes though there exist lots of missing links, which are not contained in the routing table but would be possibly observed at other points. Thus, this document uses the algorithms based on differences in AS’ network size.

Here, we provide a possible estimation method. Degree, the number of neighboring ASes, has been commonly used as an indicator which represents AS’ network size. Degree for each AS is approximately counted from publicly available datasets such as public BGP routing tables (e.g., Route Views Project [RouteViews]) and Internet routing registries. If the degree of AS X is larger than that of AS Y, AS X is considered to be transit provider of AS Y. If the degree of AS X is nearly equal to that of AS Y, the link between AS X and AS Y is considered to be peering. Thus, the relationships (i.e., cost) are estimated from publicly available information. Note that [Asai10-2] has improved the accuracy of this estimation.

```
                          +--------------+
                          | Applications |
                          +--------------+
                               |    ^        <at customer's network>
      AS relationships request |    | AS relationships response
                  (AS X, AS Y) |    |   (AS relationships of X--Y)
     - -- -- -- -- -- -- -- -- | -- | -- -- -- -- -- -- -- -- -- -- -
                               v    |                        <at ISP>
            +-------------------------------------+
            | AS relationships estimation service |
            |       +---------------------+       |
            |       |      Interface      |       |
            |       +---------------------+       |
            |                |    ^               |
            |                |    |               |
            |                v    |               |
            |       +----------------------+      | +----------------+
            |       | AS relationships     |<-------| datasets       |
            |       | estimation algorithm |      | | (AS adjacency) |
            |       +----------------------+      | +----------------+
            +-------------------------------------+
```

Figure 4: System overview of AS relationships estimation service

Figure 4 shows the system overview of AS relationships estimation service. In this figure, the AS relationships estimation service calculates degree from AS adjacency datasets, which can be approximated from public BGP routing tables etc., and then it provides the estimated AS relationships from degree. Note that degree can be replaced by the magnitude defined in [Asai10-2].

4.3. Cost Computation Service

Cost for a path is computed from the estimated AS relationships by a certain cost computation algorithm. Cost computation services on applications compute the cost.

```
             +------+   +------+           +------+   +------+
             | AS S |-->| AS X |--> ... -->| AS Y |-->| AS T |
             +------+   +------+           +------+   +------+
```

Suppose AS S and AS T are the source AS and the destination AS, respectively, on this AS path. The cost computation algorithm takes into account only edge relationships, i.e., the relationships between AS S and AS X, and AS Y and AS T.

Figure 5: AS path and a cost computation algorithm

This document provides an example of cost computation algorithms with a paper in research field. There is a research on application-layer traffic optimization in content delivery networks for reducing transit traffic by taking into account the AS relationships with degree [Asai10-1]. In [Asai10-1], only the relationships between edge (i.e., source and destination) AS and their neighbors are considered for the cost computation. The cost for the AS path shown in Figure 5 can be computed in the following equation: cost = {log(degree-of(S)) - log(degree-of(X))} + {log(degree-of(T)) - log(degree-of(Y))}. Here, function ‘‘degree-of(X)’’ returns the degree of AS X, and AS relationships (i.e., cross-domain cost) for each inter-AS link (e.g., {log(degree-of(S)) - log(degree-of(X))} and {log(degree-of(T)) - log(degree-of(Y))} in Figure 5) are resolved via AS relationships estimation services described in Section 4.2.

[Asai10-1] has shown from a simulation that their method has reduced the percentage of high-cost transit traffic (i.e., traffic from/to provider) in inter-domain traffic on residential ASes by 8.46 percentage point compared to minimum AS hop selection though the total amount of inter-domain traffic has not been changed.

Cost computation services run on not any servers but applications, so the algorithms can be modified by applications. Note that application service providers can provide the cost computation service if they need to control the computation algorithm.

5. Discussion

This section discusses the cooperation with the ALTO approach. The proposed solution approach in this document is applicable only for cross-domain cost estimation. In another word, The proposed solution approach does not mention the intra-domain traffic optimization. This document figured out the problem with the ALTO approach for cross-domain cost estimation in Section 2. Hence, the proposed solution approach can be used as a complementary element of the ALTO approach to compute cross-domain cost.

6. IANA Considerations

No need to describe any request regarding number assignment.

7. Security Considerations

This document requests that all residential ISPs should provide AS paths in their routing tables. Some ISPs do not want to reveal the information on the AS paths because they consider that it can cause security problems. On the other hand, AS paths are probably resolved by network management tools such as ‘‘traceroute’’ though they sometimes fail. Therefore, AS path provision service can be OPTIONAL.

The requirement level of AS path provision should be discussed in greater detail by considering the trade-off between security and accuracy.

8. Informative References

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", BCP 6, RFC 1930, March 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

[I-D.ietf-alto-protocol] Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-06 (work in progress), October 2010.

[Asai10-1] Asai, H. and H. Esaki, "Towards Interdomain Transit Traffic Reduction in Peer-assisted Content Delivery Networks", 14th International Telecommunications Network Strategy and Planning Symposium, pp. 95-100, 2010.

[Asai10-2] Asai, H. and H. Esaki, "Estimating AS Relationships for Application-Layer Traffic Optimization", 3rd Workshop on Economic Traffic Management, LNCS Vol. 6236, pp. 51-63, 2010.

[Gao01] Gao, L., "On inferring autonomous system relationships in the Internet", IEEE/ACM Transactions on Networking, Vol. 9, No. 6, pp. 733-745, 2001.

[Huffaker10] Huffaker, B., Dhamdhere, A., and Fomenkov, M., "Toward Topology Dualism: Improving the Accuracy of AS Annotations for Routers", Passive and Active Measurement: 11th International Conference, PAM 2010, pp. 101-110, 2010.

[RouteViews] University of Oregon, "University of Oregon Route Views Project", <http://www.routeviews.org/>.

[Wang03] Wang, F. and L. Gao, "On Inferring and Characterizing Internet Routing Policies", IMC ’03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, pp. 15-26, 2003.

[Xie08] Xie, H., Yang, Krishnamurthy, A., Liu, and A. Silberschatz, "P4P: provider portal for applications", SIGCOMM ’08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication, pp. 351-362, 2008.

Authors' Addresses

Hirochika Asai
The University of Tokyo
7-3-1 Hongo
Bunkyo-ku, Tokyo 113-8656
JP

Phone: +81 3 5841 6748
Email: panda@hongo.wide.ad.jp

Hiroshi Esaki
The University of Tokyo
7-3-1 Hongo
Bunkyo-ku, Tokyo 113-8656
JP

Phone: +81 3 5841 6748
Email: hiroshi@wide.ad.jp

Tsuyoshi Momose
Cisco Systems G.K.
2-1-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo 163-0409
JP

Phone: +81 3 5324 4154
Email: tmomose@cisco.com

Peer-to-peer simulation frameworks: a survey
draft-gurbani-p2prg-simulation-survey-00

Abstract

Peer-to-peer (p2p) protocols, like all distributed protocols, are complex, and therefore harder to debug and study in the wild. This is more true of existing p2p protocols, where changing the behaviour of the protocol --- however minor the change may be --- may result in unknown manifestations on the dynamics of the swarm using that protocol. In lieu of the unintended consequences of perturbing a live swarm, researchers have resorted to simulation frameworks. However, simulation results obtained from one simulator are often hard to reproduce when using another simulation framework. This document surveys existing simulator frameworks prevalent in simulating p2p protocols today in order to quantify any assumptions and characteristics inherent in the simulator. This, we hope, will aid future researchers in choosing the right simulation framework for their abstraction.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 8, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Peer-to-peer (p2p) protocols, like all distributed protocols, are complex, and therefore harder to debug and study in the wild. This is more true of existing p2p protocols, where changing the behaviour of the protocol --- however minor the change may be --- may result in unknown manifestations on the dynamics of the swarm using that protocol.

Researchers contemplating on changing the behavior of an existing p2p protocol have to be careful still, least they inadvertently do more harm than good by introducing their changes. Furthermore, any changes to an existing p2p protocol or a newly developed p2p protocol must be tested and evaluated for validity and reproducibility by the research community. While analytical and mathematical modeling (fluid models, optimization and linear programming) is easily validated, it is harder to validate empirical experiments due to the dynamic nature of the networks, hosts, and interconnections between them. Simulation frameworks are attractive since they provide a controlled environment under which new behavior of p2p protocols can be studied and quantified.

The good news is that there is a plethora of simulation frameworks for p2p protocols available today, some of them are surveyed in Naicken et al. [naicken]. However, that survey is dated and does not include simulation frameworks like ns-3 [ns-3] and ProtoPeer [protopeer] that have become available since the survey was published.

The aim of this document is to update the state-of-art with respect to p2p simulation frameworks available today. We will survey simulator frameworks prevalent --- and actively used --- in simulating p2p protocols today in order to quantify any assumptions and characteristics inherent in the simulator. This, we hope, will aid future researchers in choosing the right simulation framework for their abstraction.

2. Terminology

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 [RFC2119].

3. Criteria for evaluating simulation frameworks

This is a non-exhaustive list of all criteria that we should evaluate when surveying a simulation framework.

- o Type of simulator: flow-level, message-level, or packet- level. Advantages and disadvantages of each.
- o Does the simulator specifically target p2p networks? Some like ns-3 are general purpose simulators, but p2p models can be constructed and evaluated over a general-purpose simulator.
- o Level of documentation (APIs, wiki, etc.)
- o Support for building models: script level, compiled language, through a visualization editor, etc.
- o System limitations imposed by the simulator framework, if any.
- o Learning curves associated with the simulator framework.
- o Support for trace-driven simulation (i.e., using live traces to inject events in the simulator queue.
- o Scalability of the simulator.
- o Whether or not the simulator framework supports distributed simulations synchronized on a common time source or event queue.
- o Support for transitioning from a simulation environment to actual system implementation (or, can the code developed for a simulator be used with minimal or no modifications in a real host)? See Galuba et al. [protopeer].
- o Support for modeling link-level (delay, latency, loss, data rate) and host-level characteristics (i.e., simulate both low-level events and application PDUs).
- o Support for interfacing real hosts that inject events into the simulator.
- o Support for collecting statistics and measurements from the models.
- o Visualization tools for creating topologies, viewing the simulation in action, etc.
- o Support for importing existing topologies (GT-ITM) and others.
- o Support for exporting topologies in a standard graph markup language.
- o Should we focus on only academic and research simulators or commercial simulators as well?
- o ...

4. List of simulation frameworks

A list of simulation frameworks that we can survey appears below (original list is in Naicken et al. [naicken], I have added a couple more simulators). This is a rather exhaustive list, however, going forward, we should focus on those frameworks that are: newer, actively in use today, and those frameworks that are actively used today and have been surveyed before, but could stand to be looked at again in light of hardware and software advances in the last few years (multi-cores, parallel programming, etc.):

- ns-3 [ns-3].
- ProtoPeer [protopeer].
- GPS.
- PeerSim.
- P2PSim.
- OverSim.
- DHTSim.
- PlanetSim.
- VPDNS.
- Narses.
- Neurogrid.
- GnutellaSim.
- myNS --- we could probably drop this in favor of ns-3.
- Overlay Weaver.
- Query-cycle Sim.
- GTNetS [gtnets] --- seems to be abandoned.
- ...

5. Security Considerations

This document does not introduce any new security considerations in p2p protocols.

6. IANA Considerations

This document does not require any IANA considerations.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

7.2. Informative References

[gtnets] "The Georgia Tech Network Simulator (GTNetS)", http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/.

[naicken] Naicken, S., Basu, A., Livingston, B., and S. Rodhetbhai, "A Survey of Peer-to-Peer Network Simulators", Proceedings of the Seventh Annual Postgraduate Symposium, Liverpool, UK, 2006.

[ns-3] "The ns-3 network simulator", http://www.nsnam.org.

[protopeer]
Galuba, W., Aberer, K., Despotovic, Z., and W. Kellerer, "ProtoPeer: A p2p toolkit bridging the gap between simulation and live deployment", Proceedings of SIMUTools, Rome, Italy, 2009.

Appendix A. Acknowledgments

Author's Address

Vijay K. Gurbani
Bell Labs, Alcatel-Lucent
1960 Lucent Lane, Rm 9C-533
Naperville, IL 60563
USA

Email: vkg@bell-labs.com

PPSP Y. Gu
Internet-Draft N. Zong
Intended status: Standards Track Huawei
Expires: September 12, 2011 Hui. Zhang
NEC Labs America.
Yunfei. Zhang
China Mobile
J. Lei
University of Goettingen
Gonzalo. Camarillo
Ericsson
Yong. Liu
Polytechnic University
Delfin. Montuno
Lei. Xie
Huawei
March 11, 2011

Survey of P2P Streaming Applications
draft-ietf-ppsp-survey-01

Abstract

This document presents a survey of popular Peer-to-Peer streaming applications on the Internet. We focus on the Architecture and Peer Protocol/Tracker Signaling Protocol description in the presentation, and study a selection of well-known P2P streaming systems, including Joost, PPlive, andother popular existing systems. Through the survey, we summarize a common P2P streaming process model and the correspondent signaling process for P2P Streaming Protocol standardization.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Toward standardizing the signaling protocols used in today’s Peer-to-Peer (P2P) streaming applications, we surveyed several popular P2P streaming systems regarding their architectures and signaling protocols between peers, as well as, between peers and trackers. The studied P2P streaming systems, running worldwide or domestically, include such as PPLive, Joost, Cybersky-TV, and Octoshape. This document does not intend to cover all design options of P2P streaming applications. Instead, we choose a representative set of applications and focus on the respective signaling characteristics of each kind. Through the survey, we generalize a common streaming process model from those P2P streaming systems, and summarize the companion signaling process as the base for P2P Streaming Protocol (PPSP) standardization.

2. Terminologies and concepts

Chunk: A chunk is a basic unit of partitioned streaming media, which is used by a peer for the purpose of storage, advertisement and exchange among peers [Sigcomm:P2P streaming].

Content Distribution Network (CDN) node: A CDN node refers to a network entity that usually is deployed at the network edge to store content provided by the original servers, and serves content to the clients located nearby topologically.

Live streaming: The scenario where all clients receive streaming content for the same ongoing event. The lags between the play points of the clients and that of the streaming source are small..

P2P cache: A P2P cache refers to a network entity that caches P2P traffic in the network, and either transparently or explicitly distributes content to other peers.

P2P streaming protocols: P2P streaming protocols refer to multiple protocols such as streaming control, resource discovery, streaming data transport, etc. which are needed to build a P2P streaming system.

Peer/PPSP peer: A peer/PPSP peer refers to a participant in a P2P streaming system. The participant not only receives streaming content, but also stores and uploads streaming content to other participants.

PPSP protocols: PPSP protocols refer to the key signaling protocols among various P2P streaming system components, including the tracker and peers.

Swarm: A swarm refers to a group of clients (i.e. peers) sharing the same content (e.g. video/audio program, digital file, etc) at a given time.

Tracker/PPSP tracker: A tracker/PPSP tracker refers to a directory service which maintains the lists of peers/PPSP peers storing chunks for a specific channel or streaming file, and answers queries from peers/PPSP peers.

Video-on-demand (VoD): A kind of application that allows users to select and watch video content on demand

3. Survey of P2P streaming system

In this section, we summarize some existing P2P streaming systems. The construction techniques used in these systems can be largely classified into two categories: tree-based and mesh-based structures.

Tree-based structure: Group members self-organize into a tree structure, based on which group management and data delivery is performed. Such structure and push-based content delivery have small maintenance cost and good scalability and low delay in retrieving the content(associated with startup delay) and can be easily implemented. However, it may result in low bandwidth usage and less reliability.

Mesh-based structure: In contrast to tree-based structure, a mesh uses multiple links between any two nodes. Thus, the reliability of data transmission is relatively high. Besides, multiple links results in high bandwidth usage. Nevertheless, the cost of maintaining such mesh is much larger than that of a tree, and pull-based content delivery lead to high overhead associated each video block transmission, in particular the delay in retrieving the content.

Hybrid structure: Combine tree-based and mesh-based structure, combine pull-based and push-based content delivery to utilize the advantages of two structures. It has high reliability as much as mesh-based structure, lower delay than mesh-based structure, lower overhead associated each video block transmission and high topology maintenance cost as much as mesh-based structure.

3.1. Mesh-based P2P streaming systems

3.1.1. Joost

Joost announced to give up P2P technology on its desktop version last year, though it introduced a flash version for browsers and iPhone application. The key reason why Joost shut down its desktop version is probably the legal issues of provided media content. However, as one of the most popular P2P VoD application in the past years, it’s worthwhile to understand how Joost works. The peer management and data transmission in Joost mainly relies on mesh-based structure.

The three key components of Joost are servers, super nodes and peers. There are five types of servers: Tracker server, Version server, Backend server, Content server and Graphics server. The architecture of Joost system is shown in Figure 1.

First, we introduce the functionalities of Joost’s key components through three basic phases. Then we will discuss the Peer protocol and Tracker protocol of Joost.

Installation: Backend server is involved in the installation phase. Backend server provides peer with an initial channel list in a SQLite file. No other parameters, such as local cache, node ID, or listening port, are configured in this file.

Bootstrapping: In case of a newcomer, Tracker server provides several super node addresses and possibly some content server addresses. Then the peer connects Version server for the latest software version. Later, the peer starts to connect some super nodes to obtain the list of other available peers and begins streaming video contents. Different from Skype [skype], super nodes in Joost only deal with control and peer management traffic. They do not relay/ forward any media data.

Channel switching: Super nodes are responsible for redirecting clients to content server or peers.

Peers communicate with servers over HTTP/HTTPs and with super nodes/ other peers over UDP.

Tracker Protocol: Because super nodes here are responsible for providing the peerlist/content servers to peers, protocol used between tracker server and peers is rather simple. Peers get the addresses of super nodes and content servers from Tracker Server over HTTP. After that, Tracker sever will not appear in any stage, e.g. channel switching, VoD interaction. In fact, the protocol spoken between peers and super nodes is more like what we normally called "Tracker Protocol". It enables super nodes to check peer status, maintain peer lists for several, if not all, channels. It provides peer list/content servers to peers. Thus, in the rest of this section, when we mention Tracker Protocol, we mean the one used between peers and super nodes.

Peers will communicate with super nodes in some scenarios using Tracker Protocol.

1. When a peer starts Joost software, after the installation and bootstrapping, the peer will communicate with one or several super nodes to get a list of available peers/content servers.

2. For on-demand video functions, super nodes periodically exchange small UDP packets for peer management purpose.

3. When switching between channels, peers contact super nodes and the latter help the peers find available peers to fetch the requested media data.

Peer Protocol: The following investigations are mainly motivated from [Joost- experiment], in which a data-driven reverse-engineer experiments are performed. We omitted the analysis process and directly show the conclusion. Media data in Joost is split into chunks and then encrypted. Each chunk is packetized with about 5-10 seconds of video data. After receiving peer list from super nodes, a peer negotiates with some or, if necessary, all of the peers in the list to find out what chunks they have. Then the peer makes decision about from which peers to get the chunks. No peer capability information is exchanged in the Peer Protocol.

```
                   +---------------+       +-------------------+
                   | Version Server|       |   Tracker Server  |
                   +---------------+       +-------------------+
                             \                       |
                              \                      |
                               \                     | +---------------+
                                \                    | |Graphics Server|
                                 \                   | +---------------+
                                  \                  |     |
   +--------------+        +-------------+        +--------------+
   |Content Server|--------|    Peer1    |--------|Backend Server|
   +--------------+        +-------------+        +--------------+
                                     |
                                     |
                                     |
                                     |
                              +------------+       +---------+
                              | Super Node |-------|  Peer2  |
                              +------------+       +---------+
```

Figure 1, Architecture of Joost system

The following sections describe Joost QoS related features, extracted mostly from [Joost- experiment], [JO2-Moreira] and [JO7-Joost Network Architecture].

For peer selection, Host Cache of a peer, which is refreshed periodically, stores a list of Joost super nodes IP addresses and ports. The selection strategy is influenced by the number of peers accessing the same content. Specifically, the number of candidate peers made available is proportional to the number of active peers. If there are a few of them, then Joost content server is made available to assist in the data delivery. Although there is no explicit consideration for peer heterogeneities in peer selection, low capacity peers tend to partner with low capacity peer. Peers under the same NAT also tend to serve each other preferentially [JO2-Moreira]. It may consider geographical locality but not have AS-level awareness or exploit topological locality and thus may have impact on the efficiency of video distribution.

To maintain the overlay networks, super nodes probe clients, clients probe clients and super nodes, and super nodes communicate with super nodes and servers. To make up for inadequate bandwidth and to be scalable, Joost forms groups of Joost Server Islands, each island consisting of one streaming control server controlling ten streaming servers. Moreover, STUN protocol enables a client to discover whether it is behind a NAT or firewall and the type of the NAT or firewall.

For data delivery, audio and video traffic are streamed separately to allow for multi-lingual programming. Content comes mostly from peers and occasionally from content server for !olong-tail!+/- content. As peers are assumed to contribute in a best-effort manner, infrastructures are needed to make up for insufficient bandwidth, including in the asymmetric scenario. However, super nodes are not part of the bandwidth supplying infrastructures as they only relay control traffic but not data traffic to clients. To support the P2P media distribution services, Joost uses an agent based peer-to-peer system called Anthill. Joost also employs Local Video Cache for later viewing and to avoid reloading but will still require authorization from Joost server when accessing the video file at a later time.

Joost provides large buffering and thus causes longer start-up delay for VoD traffic than for live media streaming traffic. It affords more FEC for VoD traffic but gives higher priority in delivery to live media streaming traffic.

For Joost, load-balancing and fault-tolerance is shifted directly into the client and all is done natively in the p2p code.

To enhance user viewing experience, Joost provides chat capability between viewers and user program rating mechanisms.

3.1.2. Octoshape

CNN has been working with a P2P Plug-in, from a Denmark-based company Octoshape, to broadcast its living streaming. Octoshape helps CNN serve a peak of more than a million simultaneous viewers. It has also provided several innovative delivery technologies such as loss resilient transport, adaptive bit rate, adaptive path optimization and adaptive proximity delivery. Figure 2 depicts the architecture of the Octoshape system.

Octoshape maintains a mesh overlay topology. Its overlay topology maintenance scheme is similar to that of P2P file-sharing applications, such as BitTorrent. There is no Tracker server in Octoshape, thus no Tracker Protocol is required. Peers obtain live streaming from content servers and peers over Octoshape Protocol. Several data streams are constructed from live stream. No data streams are identical and any number K of data streams can reconstruct the original live stream. The number K is based on the original media playback rate and the playback rate of each data stream. For example, a 400Kbit/s media is split into four 100Kbit/s data streams, and then k = 4. Data streams are constructed in peers, instead of Broadcast server, which release server from large burden. The number of data streams constructed in a particular peer equals the number of peers downloading data from the particular peer, which is constrained by the upload capacity of the particular peer. To get the best performance, the upload capacity of a peer should be larger than the playback rate of the live stream. If not, an artificial peer may be added to deliver extra bandwidth.

Each single peer has an address book of other peers who is watching the same channel. A Standby list is set up based on the address book. The peer periodically probes/asks the peers in the standby list to be sure that they are ready to take over if one of the current senders stops or gets congested. [Octoshape]

Peer Protocol: The live stream is firstly sent to a few peers in the network and then spread to the rest of the network. When a peer joins a channel, it notifies all the other peers about its presence using Peer Protocol, which will drive the others to add it into their address books. Although [Octoshape] declares that each peer records all the peers joining the channel, we suspect that not all the peers are recorded, considering the notification traffic will be large and peers will be busy with recording when a popular program starts in a channel and lots of peers switch to this channel. Maybe some geographic or topological neighbors are notified and the peer gets its address book from these nearby neighbors.

The peer sends requests to some selected peers for the live stream and the receivers answers OK or not according to their upload capacity. The peer continues sending requests to peers until it finds enough peers to provide the needed data streams to redisplay the original live stream. The details of Octoshape are (not?) disclosed yet, we hope someone else can provide much specific information.

```
            +------------+   +--------+
            |   Peer 1   |---| Peer 2 |
            +------------+   +--------+
                 |    \    /      |
                 |     \  /       |
                 |      \         |
                 |     / \        |
                 |    /   \       |
                 |  /      \      |
      +--------------+    +-------------+
      |     Peer 4   |----|    Peer3    |
      +--------------+    +-------------+

      *****************************************
                         |
                         |
                 +---------------+
                 | Content Server|
                 +---------------+
```

Figure 2, Architecture of Octoshape system

The following sections describe Octoshape QoS related features, extracted mostly from [OctoshapeWeb], [OC2-Alstrup] and [OC3-Alstrup]. As it is a closed system, the details of how the features are implemented are not available.

To spread the burden of data distribution across several peers and thus limiting the impact of peer loss, Octoshape splits a live stream into a number of smaller equal-sized sub-streams. For example, a 400kbit/s live stream is split and coded into 12 distinct 100kbit/s sub-streams. Only a subset of these sub-streams needs to reach a user for it to reconstruct the !(R)original!_ live stream. The number of distinct sub-streams could be as many as the number of active peers.

Therefore, even if the upload capacity of a peer is smaller than its download capacity, it would now be easier to contribute a sub-stream than a whole live stream. An Octoshape peer can then receive from each neighboring peer at least a distinct sub-stream. To make up for the bandwidth asymmetry, artificial end users are used to deliver additional bandwidth. Multi OctoServers are also available to guarantee no single point of failure [OC3-Alstrup].

Octoshape keeps peer!_s availability information in an address book. Each peer keeps a periodically updated stand-by list and passes it along with its transmitted sub-stream. With constant monitoring of the quality and consistency of each content source, the peer can switch partners in case of bottleneck and congestion to a better source.

Octoshape provides operator to control who should and should not receive certain video signal due to copyright restriction, to control access based in part on IP numbers, and to obtain real time statistics during any live events.

To optimize bandwidth utilization, Octoshape leverages computers within a network to minimize external bandwidth usage and to select the most reliable and !oclosest!+/- source to each viewer. It also chooses the best matching available codecs and players and scales bit rate up and down according to available internet connection.

Octoshape [OctoshapeWeb] claims to have patented resiliency and throughput technologies to deliver quality streams to the mobile and wireless edge networks. This throughput optimization technology also cleans up latent and lossy network connections between the encoder and the distribution point, providing a stable, high quality, stream for distribution. Octoshape also claims to be able to deliver true HD, 1280x720 30fps (720p) video over the Internet and to have advanced DVR functionalities such as allowing users to move seamlessly forward and back through the streams with almost no waiting time.

3.1.3. PPLive

PPLive is one of the most popular P2P streaming software in China. It has two major communication protocols. One is Registration and peer discovery protocol, i.e. Tracker Protocol, and the other is P2P chunk distribution protocol, i.e. Peer Protocol. Figure 3 shows the architecture of PPLive.

Tracker Protocol: First, a peer gets the channel list from the Channel server, in a way similar to that of Joost. Then the peer chooses a channel and asks the Tracker server for the peerlist of this channel.

Peer Protocol: The peer contacts the peers in its peerlist to get additional peerlists, which are aggregated with its existing list. Through this list, peers can maintain a mesh for peer management and data delivery.

For the video-on-demand (VoD) operation, because different peers watch different parts of the channel, a peer buffers up to a few minutes worth of chunks within a sliding window to share with each others. Some of these chunks may be chunks that have been recently played; the remaining chunks are chunks scheduled to be played in the next few minutes. Peers upload chunks to each other. To this end, peers send to each other "buffer-map" messages; a buffer-map message indicates which chunks a peer currently has buffered and can share. The buffer-map message includes the offset (the ID of the first chunk), the length of the buffer map, and a string of zeroes and ones indicating which chunks are available (starting with the chunk designated by the offset). PPlive transfer Data over UDP.

Video Download Policy of PPLive

1 Top ten peers contribute to a major part of the download traffic. Meanwhile, the top peer session is quite short compared with the video session duration. This would suggest that PPLive gets video from only a few peers at any given time, and switches periodically from one peer to another;

2 PPLive can send multiple chunk requests for different chunks to one peer at one time;

PPLive maintains a constant peer list with relatively small number of peers. [P2PIPTV-measuring]

```
         +------------+    +--------+
         |   Peer 2   |----| Peer 3 |
         +------------+    +--------+
                  |          |
                  |          |
                 +--------------+
                 |    Peer 1    |
                 +--------------+
                         |
                         |
                         |
                 +---------------+
                 | Tracker Server|
                 +---------------+
```

Figure 3, Architecture of PPlive system

The following sections describe PPLive QoS related features, extracted mostly from [PL3-Hei], [PL5-Vu], [PL6-Liu], and [PL7-Liu].

After obtaining an initial peer list from the member server, a peer periodically updates its peer list by querying both member server and partner peers. New peers are aggressively contacted at a fixed rate. In selecting peers as partners, a peer considers their upload-bandwidth and in part, their location information [PL6-Horvath] in selecting on a FCFS basis those that have responded [PL7-Liu].

For data distribution, PPLive, a data-driven or mesh-pull scheme [PL3-Hei], divides the media content into small portions called chunks uses and TCP for video streaming. Neighbor peers use a gossip-like protocol to exchange their buffer map that indicates chunks available for sharing. Peers obtain one or more their missing chunks from one or more peers having them. Available chunks may also be downloaded from the original channel server.

PPLive uses a double buffering mechanism consisting of TV Engine and Media Player for its stream reassembly and display [PL3-Hei]. The TV Engine is responsible for downloading video chunks from the PPLive network and streaming the downloaded video to the Media Player, which in turns displays the content to the user, after each buffer is filled up to its respective predetermined threshold.

PPLive is observed to have the download scheduling policy of giving higher priority to rare chunks and to chunks closer to play out deadline and to be using a sliding window mechanism to regulate the buffering of chunks.

To utilize available peer resources, peers in one subscribed overlay may also be harnessed to support peers in other subscribed overlays [PL5-Vu].

3.1.4. Zattoo

Zattoo is P2P live streaming system which serves over 3 million registered users over European countries [Zattoo].The system delivers live streaming using a receiver-based, peer-division multiplexing scheme. Zattoo reliably streams media among peers using the mesh structure.

Figure 4 depicts a typical procedure of single TV channel carried over Zattoo network. First, Zattoo system broadcasts live TV, captured from satellites, onto the Internet. Each TV channel is delivered through a separate P2P network.

```
      ------------------------------
      |   ------------------       |         --------
      |   |  Broadcast     |       |---------|Peer1 |-----------
      |   |  Servers       |       |         --------          |
      |   Administrative Servers   |                      -------------
      |   ------------------------ |                      | Super Node|
      |   | Authentication Server | |                     -------------
      |   | Rendezvous Server     | |                           |
      |   | Feedback Server       | |         --------          |
      |   | Other Servers         | |---------|Peer2 |----------|
      |   ------------------------| |         --------
      ------------------------------|
```

Figure 4, Basic architecture of Zattoo system

Tracker(Rendezvous Server) Protocol: In order to receive the signal the requested channel, registered users are required to be authenticated through Zattoo Authentication Server. Upon authentication, users obtain a ticket with specific lifetime. Then, users contact Rendezvous Server with the ticket and identify of interested TV channel. In return, the Rendezvous Server sends back a list joined peers carrying the channel.

Peer Protocol: Similar to aforementioned procedures in Joost, PPLive, a new Zattoo peer requests to join an existing peer among the peer list. Upon the availability of bandwidth, requested peer decides how to multiplex a stream onto its set of neighboring peers. When packets arrive at the peer, sub-streams are stored for reassembly constructing the full stream.

Note Zattoo relies on Bandwidth Estimation Server to initially estimate the amount of available uplink bandwidth at a peer. Once a peer starts to forward substream to other peers, it receives QoS feedback from other receivers if the quality of sub-stream drops below a threshold.

The following sections describe Zattoo QoS related features, extracted mostly from [ZT1-Chang].

For reliable data delivery, each live stream is partitioned into video segments. Each video segment is coded for forward error correction with Reed-Solomon error correcting code into n sub-stream packets such that having obtained k correct packets of a segment is sufficient to reconstruct the remaining n-k packets of the same video segment. To receive a video segment, each peer then specifies the sub-stream(s) of the video segment it would like to receive from the neighboring peers.

Zattoo uses Peer-Division Multiplexing (PDM) scheme for its data delivery topology setup. In this scheme, each new peer independently executes the Search and Join phases. In the Search Phase, a peer queries the members of the peer list for sub-streams availability; in response, receives additional prospective peers, sub-streams availability, quality indications, and sub-stream sequence numbers; and then selects, among the responses, partnering peers or quits after failing two search attempts.

In the Join Phase, a joining peer, having selected the candidate peers, requests to partner with some of them, spreading the load among them and preferring topologically close-by peers, if these peers have less capacity or carry lower quality sub-streams. Barring departure or performance degradation of neighboring peers, the established connections stay and the specified sub-stream packet of every segment continues to be forwarded without further per-packet handshaking between peers.

To manage stream efficiently for incoming and outgoing destinations, each peer has a packet buffer, called IOB (Input-Output Buffer). The IOB is referenced by an input pointer, a repair pointer, and one or more output pointers, one for each forwarding destination such as player, file, and other peer. The input pointer points to the slot in the IOB where the next incoming packet with sequence number higher than the highest sequence number received so far will be stored, and the repair pointer always points to one slot beyond the last packet received in order and is used to regulate packet retransmission and adaptive PDM (to be described later). A packet map and forwarding discipline is associated with each output pointer to accommodate the different forwarding rates and regimes required by the destinations. Note that retransmission requests are sent to random peers and not to partnering peers and they are honoured only if the requested packets are still in IOB and there is sufficient left-over capacity to transmit all the requested packets. To avoid buffer overrun, a set of two buffers is used in the IOB instead of a circular buffer.

Zattoo uses Adaptive Peer-Division Multiplexing scheme to handle longer term bandwidth fluctuations. In this scheme, each peer determines how many sub-streams to transmit and when to switch partners. Specifically, each peer continually estimates the amount of available uplink bandwidth based initially on probe packets to the Zattoo Bandwidth Estimation Server and later, based on peer QoS feedbacks, using different algorithms depending on the underlying transport protocol. A peer increases its estimated available uplink bandwidth, if the current estimate is below some threshold and if there has been no bad quality feedback from neighboring peers for a period of time, according to some algorithm similar to how TCP maintains its congestion window size. Each peer then admits neighbors based on the currently estimated available uplink bandwidth. In case a new estimate indicates insufficient bandwidth to support the existing number of peer connections, one connection at a time, preferably starting with the one requiring the least bandwidth, is closed. On the other hand, if loss rate of packets from a peer!_s neighbor reaches a certain threshold, the peer will attempt to shift the degraded neighboring peer load to other existing peers, while looking for replacement peer. When a replacement is found, the load is shifted to it and the degraded neighbor is dropped. As expected if a peer!_s neighbor is lost due to departure, the peer initiates the process to replace the lost peer. To optimize the PDM configuration, a peer may occasionally initiate switching existing partnering peers to topologically closer peers.

3.1.5. PPStream

The system architecture and working flows of PPStream is similar to PPLive. PPStream transfers data using mostly TCP, only occasionally UDP.

Video Download Policy of PPStream

1 Top ten peers do not contribute to a large part of the download traffic. This would suggest that PPStream gets the video from many peers simultaneously, and its peers have long session duration;

2 PPStream does not send multiple chunk requests for different chunks to one peer at one time;

PPStream maintains a constant peer list with relatively large number of peers. [P2PIPTV-measuring]

The following sections describe PPStream QoS related features, extracted mostly from [PS3-Li], [PS4-Jia] and [PS5-Wei].

PPStream is mainly mesh-based but to some extent it is layered in its data distribution topology. It uses geographic clustering to some extent based on geographic longitude and latitude of the IP addresses [PS4-Jia].

To ensure data availability, some form of chunk retransmission request mechanism is used and the buffer map is shared at high rate, although concurrent requests for the same data chunk is rare. Each data chunk, identified by the play time offset encoded by the program source, is divided into 128 sub-chunks of 8KB size each. The chunk id is used to ensure sequential ordering of received data chunk.

The buffer map consists of one or more 128-bit flags denoting the availability of sub-chunks and having a corresponding time offset. Usually a buffer map contains only one data chunk at a time and is thus smaller than that of PPLive. It also contains sending peer!_s playback status to the other peers because as soon as a data chunk is played back, the chunk is deleted or replaced by the next data chunk [PS5-Wei].

At the initiating stage, a peer can use up to 4 data chunks and on a stabilized stage, a peer uses usually one data chunk. However, in transient stage, a peer uses variable number of chunks. Although, sub-chunks within each data chunks are fetched nearly in random without using rarest or greedy policy, the same fetching pattern for one data chunk seems to repeat in the following data chunks [PS3-Li]. Moreover, high bandwidth PPStream peers tend to receive chunks earlier and contributes more than lower bandwidth peers.

3.1.6. SopCast

The system architecture and working flows of SopCast is similar to PPLive. SOPCast transfer data mainly using UDP, occasionally TCP;

Top ten peers contribute to about half of the total download traffic. SOPCast’s download policy is similar to PPLive’s policy in that it switches periodically between provider peers. However, SOPCast seems to always need more than one peer to get the video, while in PPLive a single peer could be the only video provider;

SOPCast’s peer list can be as large as PPStream’s peer list. But SOPCast’s peer list varies over time. [P2PIPTV-measuring]

The following sections describe SopCast QoS related features, extracted mostly from [SC1-Ali], [SC2-Ciullo], [SC4-Fallica], [SC5-Sentinelli], [SC6-Silverston], and [SC7-Tang].

SopCast allows for software update through (HTTP) a centralized web server and makes available channel list through (HTTP) another centralized server.

SopCast traffic is encoded and SopCast TV content is divided into video chunks or blocks with equal sizes of 10KB [SC7-Tang]. Sixty percent of its traffic is signaling packets and 40% is actual video data packets [SC4-Fallica]. SopCast produces more signaling traffic compared to PPLive, PPStream, and TVAnts, whereas PPLive produces the least [SC6-Silverston]. Its traffic is also noted to have long-range dependency [SC6-Silverston], indicating that mitigating it with QoS mechanisms may be difficult. [SC1-Ali] reported that SopCast communication mechanism starts with UDP for the exchange of control messages among its peers using a gossip-like protocol and then moves to TCP for the transfer of video segments. This use of TCP for data transfer seems to contradict others findings [SC4-Fallica, SC6-Silverston].

To discover candidate peers, a peer requests peer list from Tracker, or from neighboring peer using a gossip-like protocol. To retrieve content [SC4-Fallica], a new peer contacts peers selected randomly from the peer list it obtained from having queried the root servers (trackers). The process of contacting peers slows down after the initial bootstrap phase [SC3-Horvath, SC2-Ciullo]. The number of peers a node typically connects to for download is about 2 to 5 [SC5-Sentinelli] and there is no observed preference for peers with shorter paths [SC2-Ciullo]. Partner peers periodically advertise content availability and exchange sought content. In forming multiple parent and children relationships, a peer does not exploit peer location information [SC3-Horvath]. In general, parents are chosen solely based on performance; however, lower capacity nodes seem to be choosing parents that are closer to improve performance and to compensate for its bandwidth constraints [SC1-Ali]. When needed, a peer can download video streams directly from the Source Provide, a node that broadcasts the entire video [SC7-Tang]. In the process of data exchange, there is no enforcement of tit-for-tat like mechanisms [SC2-Ciullo].

Similar to PPLive, SopCast uses a double-buffering mechanism. The SopCast buffer downloads video chunks from the network, storing them, and upon exceeding a predetermined number of stored chunks, launches the Media player. The Media player buffer then downloads video content from the local web server listening port and upon receiving sufficient amount of content, starts video playback.

3.1.7. TVants

The system architecture and working flows of TVants is similar to PPLive. TVAnts is more balanced between TCP and UDP in data transmission;

The system architecture and working flows of TVants is similar to PPLive. TVAnts is more balanced between TCP and UDP in data transmission;

TVAnts’ peer list is also large and varies over time. [P2PIPTV-measuring]

We extract the common Main components and steps of PPLive, PPStream, SopCast and TVants, which is shown in Figure 5.

```
                    +------------+
                    |   Tracker  |
                   /+------------+
                  /
                 /    +------+
            1,2/     /|Peer 1|
              /     / +------+
             /     /3,4,6
        +---------+/              +------+
        |New Peer |---------------|Peer 2|
        +---------+\     4,6      +------+
        |5  |       \
        |---|        \ +------+
                3,4,6 \|Peer 3|
                       +------+
```

Figure 5, Main components and steps of PPLive, PPStream, SopCast and Tvants

The main steps are:

(1) A new peer registers with tracker / distributed hash table (DHT) to join the peer group which shares a same channel / media content;

(2) Tracker / DHT returns an initial peer list to the new peer;

(3) The new peer harvests peer lists by gossiping (i.e. exchange peer list) with the peers in the initial peer list to aggregate more peers sharing the channel / media content;

(4) The new peer randomly (or with some guide) selects some peers from its peer list to connect and exchange peer information (e.g. buffer map, peer status, etc) with connected peers to know where to get what data;

(5) The new peer decides what data should be requested in which order / priority using some scheduling algorithm and the peer information obtained in Step (4);

(6) The new peer requests the data from some connected peers.

The following sections describe TVAnts QoS related features, extracted mostly from [TV1-Alessandria], [TV2-Ciullo], and [TV3-Horvath].

TVAnts peer discovery mechanism is very greedy during the first part of a peer life and stabilizes afterwards [TV2-Ciullo].

For data delivery, peers exhibit mild preference to exchange data among themselves in the same Autonomous System and also among peers in the same subnet. TVAnts peer also exhibits some preference to download from closer peers. According to [TV3-Horvath], TVAnts peer exploits location information and download mostly from high-bandwidth peers. However, it does not seem to enforce any tit-for-tat mechanisms in the data delivery.

TVAnts [TV1-Alessandria] seems to be sensitive to network impairments such as changes in network capacity, packet loss, and delay. For capacity loss, a peer will always seek for more peers to download. In the process of trying to avoid bad paths and selecting good peers to continue downloading data, aggressive and potentially harmful behavior for both application and the network results when bottleneck is affecting all potential peers.

When limited access capacity is experienced, a peer reacts by increasing redundancy (with FEC or ARQ mechanism) as if reacting to loss and thus causes higher download rate. To recover from packet losses, some kind of ARQ mechanism is also used. Although network conditions do impact video stream distribution such as the network delay impacting the start-up phase, they seem to have little impact on the network topology discovery and maintenance process.

3.2. Tree-based P2P streaming systems

3.2.1. PeerCast

PeerCast adopts a Tree structure. The architecture of PeerCast is shown in Figure 6.

Peers in one channel construct the Broadcast Tree and the Broadcast server is the root of the Tree. A Tracker can be implemented independently or merged in the Broadcast server. Tracker in Tree based P2P streaming application selects the parent nodes for those new peers who join in the Tree. A Transfer node in the Tree receives and transfers data simultaneously.

Peer Protocol: The peer joins a channel and gets the broadcast server address. First of all, the peer sends a request to the server, and the server answers OK or not according to its idle capability. If the broadcast server has enough idle capability, it will include the peer in its child-list. Otherwise, the broadcast server will choose at most eight nodes of its children and answer the peer. The peer records the nodes and contacts one of them, until it finds a node that can server it.

In stead of requesting the channel by the peer, a Transfer node pushes live stream to its children, which can be a transfer node or a receiver. A node in the tree will notify its status to its parent periodically, and the latter will update its child-list according to the received notifications.

```
            ------------------------------
            |            +---------+      |
            |            | Tracker |      |
            |            +---------+      |
            |                  |          |
            |                  |          |
            |   +---------------------+   |
            |   |   Broadcast server  |   |
            |   +---------------------+   |
            |------------------------------
                  /                     \
                 /                       \
                /                         \
               /                           \
         +---------+                 +---------+
         |Transfer1|                 |Transfer2|
         +---------+                 +---------+
           /      \                     /      \
          /        \                   /        \
         /          \                 /          \
   +---------+  +---------+     +---------+  +---------+
   |Receiver1|  |Receiver2|     |Receiver3|  |Receiver4|
   +---------+  +---------+     +---------+  +---------+
```

Figure 6, Architecture of PeerCast system

The following sections describe PeerCast QoS related features, extracted mostly from [CVV1-Zhang], [CVV4-Chu], [CVV5-Chu], and [CVV6-Chu].

Each PeerCast node has a peering layer which is a layer between the application layer and the transport layer. The peering layer of each node coordinates among similar nodes to establish and maintain a multicast tree. Moreover, the peering layer also supports simple, lightweight redirect primitive. This primitive allows a peer p to direct another peer c which is either opening a data-transfer session with p, or has a session already established with p to a target peer t to try to establish a data-transfer session. Peer discovery starts at the root (source) or some selected sub-tree root and goes recursively down the tree structure. When a peer leaves normally, it informs its parent who then releases the peer, and it also redirects all its immediate children to find new parents starting at some target t.

The peering layer allows for different policies of topology maintenance. In choosing a parent from among the children of a given peer, a child can be chosen randomly, one at a time in some fixed order, or based on least access latency with respect to the choosing peer. There are also many choices of peers to start and limit the search. The different combinations are all the descendants of a leaving peer have to start searching from the root [root-All (RTA)]; only the children of a leaving peer have to start searching from the root [Root (RT)]; all the descendants of a leaving peer have to start searching from the parent of the leaving peer [Grandfather-All (GFA)]; and only the children of the leaving peer have to start searching from the parent of the leaving peer [Grandfather (GF)].

A heart-beat mechanism at the peer is available to handle failed peer. With this mechanism, a peer sends keep-alive messages to its parent and children. If a parent peer detects that a child has skipped a specified number of heart-beats, it deems the child as lost and tidies up. Similarly, a child peer starts its search for new parent once its current parent is deemed to have left.

PeerCast also proposes but has not evaluated a number of algorithms that use some cost function to optimize the overlay. Some of them are described next. If a parent is already saturated, a newly arrived peer replaces one of the costlier children than the newly arrived peer and the replaced peer tries to reconnect somewhere else [Knock-Down]. Newly arrived peer replaces the target peer and the target peer becomes its child [Join-Flip]. Unstable peers are pushed down to the bottom of the tree [Leaf-Sink]. Existing child and parent relationship is flipped [Maintain-Flip].

3.2.2. Conviva

Conviva[TM][conviva] is a real-time media control platform for Internet multimedia broadcasting. For its early prototype, End System Multicast (ESM) [ESM04] is the underlying networking technology on organizing and maintaining an overlay broadcasting topology. Next we present the overview of ESM. ESM adopts a Tree structure. The architecture of ESM is shown in Figure 7.

ESM has two versions of protocols: one for smaller scale conferencing apps with multiple sources, and the other for larger scale broadcasting apps with Single source. We focus on the latter version in this survey.

ESM maintains a single tree for its overlay topology. Its basic functional components include two parts: a bootstrap protocol, a parent selection algorithm, and a light-weight probing protocol for tree topology construction and maintenance; a separate control structure decoupled from tree, where a gossip-like algorithm is used for each member to know a small random subset of group members; members also maintain pathes from source.

Upon joining, a node gets a subset of group membership from the source (the root node); it then finds parent using a parent selection algorithm. The node uses light-weight probing heuristics to a subset of members it knows, and evaluates remote nodes and chooses a candidate parent. It also uses the parent selection algorithm to deal with performance degradation due to node and network churns.

ESM Supports for NATs. It allows NATs to be parents of public hosts, and public hosts can be parents of all hosts including NATs as children.

```
            ------------------------------
            |            +---------+      |
            |            | Tracker |      |
            |            +---------+      |
            |                  |          |
            |                  |          |
            |   +----------------------+  |
            |   |    Broadcast server |   |
            |   +----------------------+  |
            |------------------------------
                  /                     \
                 /                       \
                /                         \
               /                           \
         +---------+                  +---------+
         |  Peer1   |                 |  Peer2  |
         +---------+                  +---------+
          /      \                      /      \
         /        \                    /        \
        /          \                  /          \
   +---------+  +---------+     +---------+  +---------+
   |  Peer3  |  |  Peer4  |     |  Peer5  |  |  Peer6  |
   +---------+  +---------+     +---------+  +---------+
```

Figure 7, Architecture of ESM system

The following sections describe ESM QoS related features, extracted mostly from [CVV1-Zhang], [CVV4-Chu], [CVV5-Chu], and [CVV6-Chu], and the details of Conviva are not publicly available.

ESM constructs the multicast tree in a two-step process. It constructs first a mesh of the participating peers; the mesh having the following properties:

- o The shortest path delay between any pair of peers in the mesh is at most K times the unicast delay between them, where K is a small constant.

- o Each peer has a limited number of neighbors in the mesh which does not exceed a given (per-member) bound chosen to reflect the bandwidth of the peer!_s connection to the Internet.

It then constructs a (reverse) shortest path spanning trees of the mesh with the root being the source.

Therefore a peer participates in two types of topology management: a control structure in which peers make sure they are always connected in a mesh and a data delivery structure in which peers make sure data gets delivered to them in a tree structure.

To keep connected, each peer maintains communication with a small number of random neighbors and a complete list of members through a gossip-like algorithm. When a new node joins, it gets a list of group members from the source. To look for a parent, it sends probe request to a subset of the group members it obtained; evaluates them with respect to delay to the source, application throughput and link bandwidth; and then chooses from among them a candidate parent that is not a descendant and is not saturated. In addition to using RTT-probes, consisting of 1-Kbyte transfers to detect bottleneck bandwidth, performance history of previously chosen parent is also considered. The peer also avoids probing hosts with low bottleneck bandwidth.

When a peer leaves normally, it notifies its neighboring peers and the neighboring peers propagate the departing peer info. At the same time, the departing peer continues to forward packets for some time to minimize transient packet loss. When a peer leaves due to failure, active peers detect the departure of the peer through its non-responsiveness to their probe messages. Active peers that detected the loss then propagate the departed peer info. A departed peer list that is flushed after a sufficient amount of time has passed keeps track of leaving and failed peers. The list enables refreshes from an active peer and a leaving/failed peer to be distinguished.

Departing peers and failing peers could in some instance partition a mesh into two or more components. Mesh repair algorithm detects such occurrences by noticing split in the membership list and tries to repair by virtually linking between active members to one of the non-active members, trying one non-active member at a time.

To improve mesh/tree structural and operating quality, each peer randomly probes one another to add new links that have perceived gain in utility; and each peer continually monitors existing links to drop those links that have perceived drop in utility. Switching parent occurs if a peer leaves or fails; if there is a persistent congestion or low bandwidth condition; or if there is a better clustering configuration. To allow for more public hosts to be available for becoming parents of NATs, public hosts preferentially choose NATs as parents.

The data delivery structure, obtained from running a distance vector protocol on top of the mesh using latency between neighbors as the routing metric, is maintained using various mechanisms. Each peer maintains and keeps up to date the routing cost to every other member, together with the path that leads to such cost. To ensure routing table stability, data continues to be forwarded along the old routes for sufficient time until the routing tables converge. The time is set to be larger than the cost of any path with a valid route, but smaller than infinite cost. To make better use of the path bandwidth, streams of different bit-rates are forwarded according to the following priority scheme: audio being higher than video streams and lower quality video being higher than quality video. Moreover, bit-rates of stream are adapted to the peer performance capability.

3.3. Hybrid P2P streaming system

3.3.1. New Coolstreaming

The Coolstreaming, first released in summer 2004 with a mesh-based structure, arguably represented the first successful large-scale P2P live streaming. As the above analysis, it has poor delay performance and high overhead associated each video block transmission. After that, New coolstreaming[New CoolStreaming] adopts a hybrid mesh and tree structure with hybrid pull and push mechanism. All the peers are organized into mesh-based topology in the similar way like pplive to ensure high reliability.

Besides, content delivery mechanism is the most important part of New Coolstreaming. Fig.8 is the content delivery architecture. The video stream is divided into blocks with equal size, in which each block is assigned a sequence number to represent its playback order in the stream. We divide each video stream into multiple sub-streams without any coding, in which each node can retrieve any sub-stream independently from different parent nodes. This subsequently reduces the impact to content delivery due to a parent departure or failure. The details of hybrid push and pull content delivery scheme are shown in the following:

(1) A node first subscribes to a sub-stream by connecting to one of its partners via a single request (pull) in BM, the requested partner, i.e., the parent node.(The node can subscribe more sub-streams to its partners in this way to obtain higher play quality.)

(2) The selected parent node will continue pushing all blocks in need of the sub-stream to the requested node.

This not only reduces the overhead associated with each video block transfer, but more importantly, significantly reduces the timing involved in retrieving video content.

```
                   ------------------------------
                  |            +---------+      |
                  |            | Tracker |      |
                  |            +---------+      |
                  |                  |          |
                  |                  |          |
                  |   +---------------------+   |
                  |   |    Content server   |   |
                  |   +---------------------+   |
                  |------------------------------
                        /                     \
                       /                       \
                      /                         \
                     /                           \
               +---------+                   +---------+
               |  Peer1  |                   |  Peer2  |
               +---------+                   +---------+
                /      \                       /      \
               /        \                     /        \
              /          \                   /          \
         +---------+  +---------+     +---------+  +---------+
         |  Peer2  |  |  Peer3  |     |  Peer1  |  |  Peer3  |
         +---------+  +---------+     +---------+  +---------+
```

Figure 8 Content Delivery Architecture

The following sections describe Coolstreaming QoS related features, extracted mostly from [CS1-Bo] and [CS2-Xie].

The basic components of Coolstreaming consist of the source, bootstrap node, web server, log server, media servers, and peers. Three basic modules in a peer help it maintain a partial view of the overlay (Membership Manager); establish and maintain partnership with other peers with which Buffer Maps indicating available video content, are exchanged (Partnership Manager),; and manage data delivery, retrieval, and play out (Stream Manager).

In building the overlay topology, a newly arrived peer contacts the bootstrap node for a list of nodes and stores it in its own mCache. From the stored list, it selects nodes randomly to forms partnership and then parent-children relationship, where a partnership between two nodes exists when only block availability information is exchanged between them, and a parent-children relationship exists when, in addition to being partner, video content is also exchanged.

Video content is processed for ease of delivery, retrieval, storage, and play out. To manage content delivery, a video stream is divided into blocks with equal size, each of which is assigned a sequence number to represent its playback order in the stream. Each block is further divided into K sub-blocks and the set of ith sub-blocks of all blocks constitutes the ith sub-stream of the video stream, where i is the value bigger than 0 and less than K+1. To retrieve video content, a node receives at most K distinct sub-streams from its parent nodes. To store retrieved sub-streams, a node uses a double buffering scheme having a synchronization buffer and a cache buffer. The synchronization buffer stores the received sub-blocks of each sub-stream according to the associated block sequence number of the video stream. The cache buffer then picks up the sub-blocks according to the associated sub-stream index of each ordered block. To advertise the availability of the latest block of different sub-streams in its buffer, a node uses a Buffer Map which is represented by two vectors of K elements each. Each entry of the first vector indicates the block sequence number of the latest received sub-stream, and each bit entry of the second vector if set indicates the index of the sub-stream that is being requested.

For data delivery, a node uses a hybrid push and pull scheme with randomly selected partners. A node having requested one or more distinct sub-streams from a partner as indicated in its first Buffer Map will continue to receive the sub-streams of all subsequent blocks from the same partner until future conditions cause the partner to do otherwise. Moreover, users retrieve video indirectly from the source through a number of strategically located servers.

To keep the parent-children relationship above a certain level of quality, each node constantly monitors the status of the on-going sub-stream reception and re-selects parents according to sub-stream availability patterns. Specifically, if a node observes that the block sequence number of the sub-stream of a parent is much smaller than any of its other partners!_ by a predetermined amount, then the node concludes that the parent is lagging sufficiently behind and needs to be replaced. Furthermore, a node also evaluates the maximum and minimum of the block sequence numbers in its synchronization buffer to determine if any parent is lagging behind the rest of its parents and thus needs also to be replaced.

4. A common P2P Streaming Process Model

As shown in Figure 8, a common P2P streaming process can be summarized based on Section 3:

1) When a peer wants to receive streaming content:

   1.1) Peer acquires a list of peers/parent nodes from the tracker.

   1.2) Peer exchanges its content availability with the peers on the obtained peer list, or requests to be adopted by the parent nodes.

   1.3) Peer identifies the peers with desired content, or the available parent node.

   1.4) Peer requests for the content from the identified peers, or receives the content from its parent node.

2) When a peer wants to share streaming content with others:

   2.1) Peer sends information to the tracker about the swarms it belongs to, plus streaming status and/or content availability.

```
+---------------------------------------------------------+
|   +--------------------------------+                    |
|   |              Tracker           |                    |
|   +--------------------------------+                    |
|        ^  |                    ^                        |
|        |  |                    |                        |
|  query |  | peer list/         |streaming Status/       |
|        |  | Parent nodes       |Content availability/   |
|        |  |                    |node capability         |
|        |  |                    |                        |
|        |  V                    |                        |
|   +-------------+         +------------+                |
|   |    Peer1    |<------->|  Peer 2    |                |
|   +-------------+ content/+------------+                |
|                   join requests                         |
+---------------------------------------------------------+
```

Figure 8, A common P2P streaming process model

The functionality of Tracker and data transfer in Mesh-based application and Tree-based is a little different. In the Mesh-based applications, such as Joost and PPLive, Tracker maintains the lists of peers storing chunks for a specific channel or streaming file. It provides peer list for peers to download from, as well as upload to, each other. In the Tree-based applications, such as PeerCast and Canviva, Tracker directs new peers to find parent nodes and the data flows from parent to child only.

5. Security Considerations

This document does not consider security issues. It follows the security consideration in [draft-zhang-ppsp-problem-statement].

6. Acknowledgments

We would like to acknowledge Jiang xingfeng for providing good ideas for this document.

7. Informative References

[PPLive]   "www.pplive.com".

[PPStream]
"www.ppstream.com".

[CNN]   "www.cnn.com".

[OctoshapeWeb]
"www.octoshape.com".

[Joost-Experiment]
Lei, Jun, et al., "An Experimental Analysis of Joost Peer-to-Peer VoD Service".

[Sigcomm_P2P_Streaming]
Huang, Yan, et al., "Challenges, Design and Analysis of a Large-scale P2P-VoD System", 2008.

[Octoshape]
Alstrup, Stephen, et al., "Introducing Octoshape-a new technology for large-scale streaming over the Internet".

[Zattoo]   "http: //zattoo.com/".

[Conviva] "http://www.rinera.com/".

[ESM04]   Zhang, Hui., "End System Multicast, http://www.cs.cmu.edu/˜hzhang/Talks/ESMPrinceton.pdf", May .

[Survey]   Liu, Yong, et al., "A survey on peer-to-peer video streaming systems", 2008.

[draft-zhang-alto-traceroute-00]
"www.ietf.org/internet-draft/draft-zhang-alto-traceroute-00.txt".

[P2PStreamingSurvey]
Zong, Ning, et al., "Survey of P2P Streaming", Nov. 2008.

[P2PIPTV_measuring]
Silverston, Thomas, et al., "Measuring P2P IPTV Systems".

[Challenge]
Li, Bo, et al., "Peer-to-Peer Live Video Streaming on the Internet: Issues, Existing Approaches, and Challenges", June 2007.

[NewCoolstreaming]
Li, Bo, et al., "Inside the New Coolstreaming: Principles,Measurements and Performance Implications", Apr. 2008.

[JO2-Moreira]
Moreira, J, et al., "IEEE Network Operations and Management Symposium", Apr. 2008.

[JO7-Joost Network Architecture]
"Joost Network Architecture, http://scaryideas.com/content/2362/".

[OC2-Alstrup]
Alstrup, S, et al., "Octoshape "C a new technology for large-scale streaming over the Internet", 2005.

[OC3-Alstrup]
Alstrup, S, et al., "Grid live streaming to millions", 2006.

[PL3-Hei] Hei, X, et al., "Insights into PPLive: A measurement study of a large-scale P2P IPTV system", May 2006.

[PL5-Vu]   Vu, L, et al., "Understanding Overlay Characteristics of a Large-Scale Peer-to-Peer IPTV System", November 2010.

[PL6-Horvath]
"".

[SC3-Horvath]
""".

[TV3-Horvath]
Horvath, A, et al., "Dissecting PPLive, SopCast, TVAnt".

[PL7-Liu] Liu, Y, et al., "A Case Study of Traffic Locality in Internet P2P Live Streaming Systems".

[PS3-Li]   Li, C, et al., "Measurement Based PPStream client behavior analysis", 2009.

[PS4-Jia] Jia, J, et al., "Characterizing PPStream across Internet", 2007.

[PS5-Wei] Wei, T, et al., "Study of PPStream Based on Measurement", 2008.

[SC1-Ali] Ali, S, et al., "Measurement of Commercial Peer-to-Peer Live Video Streaming", Aug 2006.

[SC2-Ciullo]
"".

[TV2]   Ciullo, D, et al., "Network Awareness of P2P Live Streaming Applications: A Measurement Study", Aug 2010.

[SC4-Fallica]
Fallica, B, et al., "On the Quality of Experience of SopCast", Aug 2008.

[SC5-Sentinelli]
Sentinelli, A, et al., "Will IPTV Ride the Peer-to-Peer Stream?", June 2007.

[SC6-Silverston]
Silverston, T, et al., "Traffic analysis of peer-to-peer IPTV communities", 2009.

[SC7-Tang]
Tang, S, et al., "Topology dynamics in a P2PTV network", 2009.

[TV1-Alessandria]
Alessandria, E, et al., "P2P-TV Systems under Adverse Network Conditions: a Measurement Study", 2009.

[ZT1-Chang]

Chang, H, et al., "Live streaming performance of the Zattoo network", 2009.

[PC1-Deshpande]
Deshpande, H, et al., "Streaming Live Media over a Peer-to-Peer Network", August 2001.

[PC2-http]
"http://arbor.ee.ntu.edu.tw/archive/p2p/p2p/showDoc2.pdf".

[PC3-http]
"http://ilpubs.stanford.edu:8090/863/".

[CVV1-Zhang]
Zhang, H, et al., "End System Multicast", May 2004.

[CVV4-Chu]
Chu, Y, et al., "A Case for End System Multicast", June 2000.

[CVV5-Chu]
Chu, Y, et al., "Early Experience with an Internet Broadcast System Based on Overlay Multicast", June 2004.

[CVV6-Chu]
Chu, Y, et al., "Narada is a self-organizing, overlay-based protocol for achieving multicast without network support", Aug 2001.

[CS1-Bo] Li, B, et al., "Inside the New Coolstreaming: Principles, Measurements and Performance Implications", 2008.

[CS2-Xie] Xie, S, et al., "Coolstreaming: Design, Theory, and Practice", 2007.

Authors' Addresses

Gu Yingjie
Huawei
Baixia Road No. 91
Nanjing, Jiangsu Province 210001
P.R.China

Phone: +86-25-56624760
Fax: +86-25-56624702
Email: guyingjie@huawei.com

Zong Ning
Huawei
Baixia Road No. 91
Nanjing, Jiangsu Province 210001
P.R.China

Phone: +86-25-56624760
Fax: +86-25-56624702
Email: zongning@huawei.com

Hui Zhang
NEC Labs America.

Email: huizhang@nec-labs.com

Zhang Yunfei
China Mobile

Email: zhangyunfei@chinamobile.com

Lei Jun
University of Goettingen

Phone: +49 (551) 39172032
Email: lei@cs.uni-goettingen.de

Gonzalo Camarillo
Ericsson

Email: Gonzalo.Camarillo@ericsson.com

Liu Yong
Polytechnic University

Email: yongliu@poly.edu

Delfin Montuno
Huawei

Email: delfin.montuno@huawei.com

Xie Lei
Huawei

Email: xielei57471@huawei.com

P2PRG S. Kamei
Internet-Draft NTT Corporation
Intended status: Informational T. Momose
Expires: September 5, 2011 Cisco Systems
T. Inoue
NTT Communications
March 4, 2011

ALTO-Like Activities and Experiments in P2P Network Experiment Council
draft-kamei-p2p-experiments-japan-05

Abstract

This document introduces experiments to clarify how ALTO-like approach was effective to reduce network traffic made by a Council in Japan to harmonize P2P technology with the infrastructure. And this also provides some suggestions that might be useful for ALTO architecture learned through our experiments.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 5, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the BSD License.

Table of Contents

1. Introduction

An overlay network, which is used by P2P and other applications, offers the advantage of allowing flexible provision of services while hiding the lower layer network. The downside is that inefficient routes are often taken in the lower IP network, thereby increasing the network load. Several proposals have been made to build an overlay network that takes account of the information about the lower layer network. Since the management of the Internet is highly distributed, it is difficult to implement such proposals and thus optimize a network without the cooperation of network providers.

Recently, the controversy between the overlay network and the network providers have been rekindled. Under these circumstances, some researchers have studied overlay network control technology that takes account of the network topology information obtained from network providers.

One of the activities concerning this issue has been made by the P2P Network Experiment Council in Japan. This document reports on the issues addressed and experiments being made by the council, focusing on the experiments made from 2007 to 2008.

2. Background in Japan

2.1. P2P traffic

As of 2008, the world most popular P2P file sharing application Bittorrent isn’t widely deployed in Japan. Instead, other Japan specific file sharing P2P applications such as Winny [1], Share [2], PerfectDark, and so on, still occupy 40% of the Internet traffic in Japan even though many those P2P users were arrested for sharing illegal files with these P2P apps.

Each P2P file sharing application has their original protocol. Therefore, it is more difficult to control one by one unlike Bittorrent.

2.2. Impact on network infrastructure

One of the advantage of using P2P technology for content delivery is that peers exchange content directly among themselves. This reduces the load on servers. Also, P2P applications can reduce upstream traffic from an original content server. This is significant that the charge for upstream traffic is usually traffic-sensitive for content delivery services, and it is not negligible.

It is also known that server cost could be reduced with P2P technology. However, the story is quite different for network providers. From the viewpoint of network providers, the traffic that content servers generate has shifted to the edge network and the amount of traffic has not necessarily been reduced. Another problem for network providers that an extremely inefficient routing may be selected has been raised. It is because overlay network systems are configured without any regard to the structure of the lower layer network or network geometry.

In some cases, traffic on the Internet used to be limited by the capacity of servers. For those cases, the improvement in the scalability of servers has made it likely that network resources will be used up before server resources are. Using P2P applications increases the volume of traffic per user remarkably.

Faced with increase in the load on network infrastructure, network providers are compelled to take actions to overcome the sudden increase in facilities' cost. Representative actions include placing content in IXs or data centers, introducing bandwidth control, and raising the access fees[3].

In the future, video posting sites, which has been delivered using client-server applications, may adopt P2P system. The increase in traffic arising from such a shift will be a great threat to the network.

2.3. The object of P2P Network Experiment Council

In order to reduce Internet traffic and encourage legitimate use of P2P technologies, the Japanese government led to establish a new council called P2P Network Experiment Council conjunction with commercial P2P application vendors and ISPs in 2006.

Then the council had started to develop regulations that include several guidelines such like an advance notice to restrict bandwidth to heavy duty users. In accordance with the regulations, some ISPs introduced solutions that reduce traffic caused by P2P file sharing applications .

Besides this activity, the council also looked for new ways of commercial use P2P applications under conjunction with ISPs, carrier, contents providers and P2P system vendors. In this work, the council had experiments that introduced ALTO-like system and observed how the traffic was reduced by redirecting to proper peers on the real Internet in Japan.

This memo describes the overview of the experiments.

3. The details of the experiments

The council has already learned that the server cost could be reduced with using P2P technology for contents delivering by investigating data offered by the members of the council. For example, the data brought by the vendors shows as follows:

90% of traffic was reduced with UG Live by Utagoe Inc[4].

The costs of delivering to tens of thousand subscribers was reduced to 1/5 with BBbroadcast with TV Bank Corp.[5]

On the other hand, these reduced server costs may affect network load. One of the goals of our experiments are to visualize the impacts and propose an architecture to reduce network load caused by these new technologies.

To satisfy the above goals, the framework to be proposed should be well generalized as possible that doesn't rely specific P2P application behaviors because multi P2P application vendors join these experiments. In addition, the traffic should be captured beyond multi ISPs.

3.1. Dummy Node

As mentioned before, while the effect of delivery using P2P technology on reducing the traffic and the load on servers is well known, traffic behavior in the inter-ISP is not known. In Japan, there is a backbone traffic report cooperated with ISPs and IXes [6]. However, this measurement requires to capture packets on subscribers line to know end user's activity. It is not realistic to measure the behavior of P2P applications at user terminals connected to the Internet because that would require a large-scale arrangement for measurement, such as using Deep Packet Inspection (DPI) on aggregated lines.

To solve these problems, we put several nodes called 'dummy nodes' in the ISP's networks. The dummy nodes emulate an end user's PC and P2P applications are running on the nodes.

By introducing dummy nodes, we can observe and evaluate how much P2P applications have affected networks by measuring the traffic on dummy nodes. Since this method can't measure every subscriber's traffic, the accuracy would be less than other methods. But this make it possible to adapt to situations many different P2P applications coexist on a network. We can say this is suitable for these experiments.

A dummy node consists of Intel PC server, Linux(CentOS), VMWare and Windows XP works on VMWare. With this configuration, all packets can be captured without any impacts to the network, nodes and application behaviors. And it enable us to use different P2P applications for windows and evaluate them generally.

To see behaviors of the node, incoming and outgoing packets are captured on Linux because every packets are transmitted through it. In these experiments, we captured source/destination address, port number, amount of traffic and start/end time to see flow information.

60 Dummy nodes are put on access networks that are closest subscriber as possible in different 40 networks.

```
+----------------------+
|+--------------------+|
||+------------------+||
||| P2P Application  |||
|||    WindowsXP     |||
|||        +--+      |||
||+--------|N |------+||
||  VMware |e |       ||
|+---------|t |-------+|
|   Linux  |IF| capture|
+----------|  |--------+
           +--+
```

Dummy nodes

Figure 1

4. Hint Server ('08)

In Japan, bottleneck in IP networks will shift from access networks to backbone networks and equipments, such as bandwidth between ISPs and capacity in IXs, since FTTH has rapidly spread all over Japan. Under this situation, the Council proposed a less restrictive and more flexible cooperation between ISPs than existent P4P experiments [7]. The proposed method consists of the following elements: (1) P2P clients, (2) P2P control servers, and (3) a peer selection hint server, and a Hint Server. (1) and (2) are existing systems but whether (2) exists depends on each application. (3) is a server that provides a hint as to the selection of a peer, and plays a role equivalent to that of ALTO Server. Note that this proposal was based on results of experiments using dummy nodes. The results showed that it was possible to reduce unnecessary traffic that flows across the boundaries of geographical districts or ISPs through providing information about the physical network to P2P applications.

When a peer joins the network, it registers its location information (IP address) and supplementary information (line speed, etc.) with the Hint Server. The Hint Server makes a mapping of the new peer (P2P client) based on network topology information obtained from the ISP, generates a routing table in which peers are listed in the order of priority for selection, and returns the table to the peer.

If all information can be made public, the above procedure can produce a result which is close to overall optimization. However, some information held by ISPs can often be confidential. Besides, in some cases, the volume of calculation required to process all information can be excessive. To avoid these problems, it is planned to conduct experiments with a limited set of functions, analyze experiments results, and gradually expand the scope of optimization.

A control mechanism that makes use of all possible information is difficult not only technically but also difficulties to achieve coordination among providers. In consideration of these difficulties, the council has been limiting the implementation and experiments to the following scope since 2006.

Figure 2 shows an outline of the hint server.

```
   +---------+   GetLocation    +-------------GeoIP DB Server---------+
   |         |  +-----------+   |   +----------+     +-----------+   |
   |         |--|IP Address |-->|   | GeoIP DB |     |Quagga etc |   |
   |         |  +-----------+   |   +----------+     +-----------+   |
   |         |                  | +-------------+  +----------------+ |
   |         |  +-----------+   | |  District   |  |    Routing     | |
   |         |--|AS Code:   |---| | information |  |information(DGP)| |
   |         |  |Regional   |   | |             |  |                | |
   |P2P Peers|  |Information|   | |  Range of   |  |AS Code(origin) | |
   |   or    |  +-----------+   | | IP address  |  |                | |
   | Contro| |                  | +-------------+  +----------------+ |
   | Server  |                  +-------------------------------------+
   |         |                                 |      ^
   |         |  PeerSelection                  v      |
   |         |  +-----------+   +--------------------------------------+
   |         |--|IP Address |-->| +--Priority Node Selection System--+ |
   |         |  |    List   |   | |                                  | |
   |         |  +-----------+   | |     Peer candidate ranking       | |
   |         |  +-----------+   | |                                  | |
   |         |--|  Ranking  |-->| +----------------------------------+ |
   |         |  +-----------+   +--------------------------------------+
   +---------+
```

Peer selection hint server

Figure 2

The network information used by the Hint Server is not information solicited from individual ISPs but the AS number and district information, which are more or less already public. Routing tables are not generated. Instead, peers within the same ISP or the same district are selected with higher priority in order to confine traffic to within the same ISP or the same district.

When the Hint Server receives an IP address, it returns its attribute information, to achieve the above. A peer can select a peer based on the returned information. This operation is called GetLocation. However, in preparation for the time when it becomes necessary to hide topology information, an interface is provided through which a priority order is returned in response to an input of a list of candidate peers. This operation is called PeerSelection.

Although the priority node is selected based on the criterion that it is within the same ISP or the same district, this type of selection is not very effective if the number of participating peers is small. Table 1 shows ratio of peers within the same AS or the same prefecture calculated from the distribution of ASs and prefectures in the IP address space from one-day data on a Winny network.

Conditions	ratio
AS matches	6.70%
Prefecture matches	12.76%
Both match	2.09%
Neither match	78.45%

Table 1: AS and prefecture distributions

Since, in addition to the above, the presence/absence of content affects the result, the control of selecting a peer within the same district may be inadequate. Therefore, it is necessary to introduce the weight of a continuous quantity that reflects the physical distance or the AS path length as an indicator of the proximity of the areas involved.

In consideration of the above, the following two measures are used for the evaluation of proximity between peers in a Hint Server.

- AS path length (distance between ISPs)

  AS path length calculated from BGP full routes. Since a full routing table retrieved at an ISP can show only a best path, it may not get an accurate length if the AS hop of both ISPs is too large. To avoid this, we use multiple BGP information gotten at different ISPs and combine them. Based on this concept, we used BGP routing information’s offered by three ISPs operated by big telecommunication couriers and made a topology tree. Then it enables to calculate the shortest path between given two ASes.

- Geographical distance

  Distances between peers are measured using physical distance of prefectural capitals that target peers belong to. The distance between prefectural capitals is used to calculate physical distance. Distances between prefectural capitals are sorted into ascending order, and then into bands, with weights 1 to 15 assigned to them so that there are a more or less equal number of "capital pairs" in each band. If either of their location is indefinite, distance is equal to 15 and, if they are in the same prefecture, distance is equal to 0.

  Evaluation of distances between peers showed that the distribution of distances was almost uniform when distances between peers are normalized. This result suggests that using normalized distances expands the area where the control by a Hint Server is effective.

An example of the request and the response

o Request

```
POST /PeerSelection HTTP/1.1
Host: ServerName
User-Agent: ClientName
Content-Type: text/plain; charset=utf-8

v=Version number
[application=Application identifier]
ip=IP address of physical interface
port=Port number of physical interface
[nat={no|upnp|unknown}]
[nat_ip=Global IP address using UPnP]
[nat_port= Global port number using UPnP]
[trans_id=transcation ID]
[pt=Flag of port type]
[ub=upload bandwidth]
[db=download bandwidth]
```

o Response

```
HTTP/1.1 200 OK
Date: Timestamp
Content-Type: text/plain; charset=utf-8
Cache-control: max-age=max age
Connection: close

v=Version number
ttl=ttl
server=hint server name
...
trans_id=transaction ID
pt=Flag of port type
client_ip=Peer IP address observed from server
client_port=Peer port number observed from server
numpeers=number of respond peer
n=[src address] dst address / cost / option
```

5. High-Level Trial Results

5.1. Peer Selection with P2P

Table 2 shows the result of the analysis of communication in a node of an ISP installed in Tokyo, as an example of measurement results.

Conditions	Experiment 1	Experiment 2
*Peers selected within the same ISP	22%	29%
*Peers selected within the same district	19%	23%
*Peers selected within the same district and the same ISP	5%	7%

Table 2: Percentage of communication within the same ISP

The table shows that the probability of communication with peers in the same ISP is proportional to the number of population and the share of the ISP in each district. The data show that peers were selected at random. Note that the vendor of a P2P application used in these experiments explained that the mechanism of selection a peer using network information can be implemented. However, peer selection is normally based on past information because users often cannot actually perceive the effect of using network information.

5.2. Peer Selection with the Hint Server

Since the main objective of these experiments was to verify the operations of the Hint Server and P2P applications, the degree to which traffic in the network was actually reduced was not evaluated. However, the distances between a dummy node and a peer were obtained from data on the dummy nodes. An examination of the distances between a dummy node and a peer revealed that mean value of distance after the Hint Server was introduced was reduced by 10% and that 95% value of that was reduced by 5%.

6. Considerations

We clarified followings throughout our experiments.

1. Dispersed dummy nodes can figure out the behavior of peers and traffic between inter-ISP networks, which peers are selected by each peer. Therefore it proves that the importance of peer

selection control mechanism proposed in ALTO.

2. Using our peer selection control mechanism, called hint server, could achieve significant differences. Our hint server can lead each peer to select nearer peer.

In the experimental result of peer selection control, it is smaller in intra-ISP traffic than other experiments[8] We think that it is because there are smaller peers in each area of traffic control. When there are many peers in one ISP, it is easy to select peers in the same ISP. However, when there are small peers in one ISP, it is difficult to select peers in the same ISP. In the situation of our experiments, there are many ISPs of peers belonging, and there are relatively smaller peers exist in same ISP.

Moreover, we didn’t force P2P vendors to limit their implementation policy, therefore we can observe differences how each implementations weigh the information from the hint servers. Especially, in tree overlay topology P2P applications, such mechanism is very effective, on the other hand, in mesh overlay system, less effective.

6.1. Next steps

The experiments are on going as of 2011. Current experiments in 2011, we’ve changed the communication protocol to hint servers to ALTO based because it is nearly standardized. In our implementation, PIDs and the value of cost are mapped to ISP subnets, and ISP distance respectively. We also implement services for compatibility required by ALTO such as Service Capability and Map Services. But the Endpoint Cost Service is mainly used because of backward compatibility of our experiments.

We also study hierarchical hint server structure, in order to control in coarse inter-ISPs and in detail intra-ISP. It is also effective for limiting the area of information disclose.

6.2. Feedback to ALTO WG

This section describes what the authors learned with these experiments would be useful for the ALTO WG.

6.2.1. Hierarchical architecture for ALTO servers

In our experiments, we present the possibility of traffic control among multi-ISPs and multi-P2P applications using ALTO mechanism. On the other hand, we found several problems in ISP operations to adapt the mechanism. One is the granularity of network information. Among inter-ISP area, it is relatively easy to treat information for public purpose using BGP full route. On the other hand, among intra-ISP area, it may be difficult to disclose private information of each ISP. [9] propose some modification for ALTO protocol in order to hide ISP information. We propose hierarchical structures. From the viewpoint of cooperation between ISPs, fine-grained information is not necessarily required and moreover it is difficult to exchange the fine-grained information between ISPs. Considering this situation, the authors use only coarse-grained information to control backbone traffic in the experiments this year, though demand of controlling traffic within an ISP using fine-grained information will arise in the near future. Therefore it led us that introducing hierarchical structure into ALTO is necessary to cope with both situations. Actually, the authors plan to adapt a hierarchical control mechanism in the next steps, which include the following two steps.

- In the first step, coarse-grained information about whole the network is used to select ISPs.
- Next, fine-grained information within the ISP is used to select a peer.

6.2.2. Measurement mechanism

In the experiments, there were two difficulties as follows:

- Evaluating effect of introducing a Hint Server was difficult, since P2P applications had their own measurement mechanisms.
- How to treat priority orders of peers suggested by a Hint Server could not be predetermined for P2P applications.

From these experiences, the authors consider that clarifying requirements about measurement mechanisms for P2P applications are necessary also in ALTO.

7. Security Considerations

This document does not propose any kind of protocol, practice or standard.

8. IANA Considerations

No need to describe any request regarding number assignment.

9. Acknowledgments

Thanks to strong support by MIC (Ministry of Internal Affairs and Communications of Japanese government), the council was established. These experiments were performed under cooperation among P2P Network Experiment Council members, and DREAMBOAT co.,ltd., Bitmedia Inc., Utagoe. Inc. and Toyama IX have especially supported analyses of the experiments. The authors appreciate Tohru Asami, Hiroshi Esaki and Tatsuya Yamshita for their constructive comments.

10. Informative References

[1] "Winny on Wikipedia", <http://en.wikipedia.org/wiki/Winny>.

[2] "Share on Wikipedia", <http://en.wikipedia.org/wiki/Share_(P2P)>.

[3] Taniwaki, "Broadband Competition Policy in Japan", 2008, <http://www.smartireland.jp/en/forum/may-2009/>.

[4] Utagoe Inc., "UGLive technology introduction", http://www.utagoe.com/en/technology/grid/live/index.html, March, 2011.

[5] TVBank, "Live Delivery using ‘BB Broadcast’Achieving 96% Saving in Traffic!", http:.wwww.tv-bank.com/jp/20081031.html, 2008 (in Japanese).

[6] Cho, Fukuda, Esaki, and Kato, "The Impact and Implications of the Growth in Residential User-to-User Traffic", SIGCOMM2006, pp207-218, Pisa, Italy, September 2006.

[7] Open P4P, "P4P Field Tests: Yale-Pando-Verizon", http://www.openp4p.net/front/fieldests, 2009.

[8] "RFC5632: Comcast’s ISP Experiences in a Proactive Network Provider Participation for P2P (P4P) Technical Trial", September 2009.

[9] "ALTO H12,draft-kiesel-alto-h12-02 (work in progress)", March 2010.

Authors' Addresses

Satoshi Kamei
NTT Service Integration Laboratories
3-9-11, Midori-cho
Musashino-shi, Tokyo 180-8585
JP

Phone: +81-422-59-6942
Email: kamei.satoshi@lab.ntt.co.jp

Tsuyoshi Momose
Cisco Systems G.K.
2-1-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo 163-0409
JP

Phone: +81-3-5324-4154
Email: tmomose@cisco.com

Takeshi Inoue
NTT Communications
3-4-1, Shibaura
Minato-ku, Tokyo 108-8118
JP

Phone: +81-3-6733-7177
Email: inoue@jp.ntt.net