Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 5, 2009

Y. Wang
Microsoft Corp.
R. Alimi
Yale University
D. Pasko
Verizon
L. Popkin
Pando Networks, Inc.
Y. Yang
Yale University

March 4, 2009

P4P Protocol Specification
draft-wang-alto-p4p-specification-00.txt

Status of This Memo

Copyright Notice

and restrictions with respect to this document.

Abstract

Provider Portal for Network Applications (P4P) is a framework that enables Internet Service Providers (ISPs) and network application software developers to work jointly and cooperatively to optimize application communications. The goals of this cooperation are to reduce network resource consumption and to accelerate applications. To achieve these goals, P4P allows ISPs to provide network information and guidance to network applications, allowing clients to exchange data more effectively. This document specifies the P4P protocol operations and message formats. The goal is provide a formal specification for developers to create inter-operable implementations.

Table of Contents

1. Introduction 4
  1.1. Status of this Memo 4
  1.2. Terminology 4
  1.3. Protocol Overview 5
    1.3.1. Location Portal Service 5
    1.3.2. pDistance Portal Service 5
  1.4. Common Application Scenario 5
  1.5. Key Features 6
2. Conventions Used in This Document 6
3. Messages 6
  3.1. Definitions 6
    3.1.1. Basic Types 7
    3.1.2. IP Addresses 7
    3.1.3. Autonomous System Numbers 7
    3.1.4. Network Location Identifiers 8
    3.1.5. ISP Identifiers 8
    3.1.6. PIDs 8
    3.1.7. pDistance Values 8
    3.1.8. pDistance Endpoint 9
  3.2. Syntax 9
    3.2.1. Headers 9
    3.2.2. Content-Type 10
    3.2.3. GetPID and PID Messages 10
    3.2.4. GetPIDMap and PIDMap Messages 11
    3.2.5. GetpDistance and pDistance Messages 11
4. Protocol Operations 13
  4.1. Standard Definitions and Reserved Values 13
    4.1.1. PIDs 13
    4.1.2. pDistances 14
  4.2. Message Handling 14

4.2.1. Common Operations 15
4.2.2. GetPID 15
4.2.3. GetPIDMap 16
4.2.4. GetpDistance 17
4.3. Exception Handling 19
4.3.1. Invalid Request URI Path 19
4.3.2. Invalid Request Format 19
4.4. Timers 19
4.5. Message Exchange Examples 19
4.5.1. GetPID 20
4.5.2. GetPIDMap 22
4.5.3. GetpDistance 23
5. Discussions 24
5.1. Discovery 24
5.2. Delegation 25
5.3. Load Balancing Considerations 25
5.4. Caching P4P Information 25
5.5. Transport and Encoding Considerations 25
6. Security Considerations 26
6.1. Protecting P4P Information 26
6.1.1. Authentication 26
6.1.2. Encryption 26
6.2. ISPs 26
6.3. Clients 27
7. IANA Considerations 27
8. Conclusions 27
9. References 27
9.1. Normative References 27
9.2. Informative References 28
Appendix A. Contributors 28
Appendix B. Acknowledgments 29

1. Introduction

Provider Portal for Network Applications (P4P) [I-D.p4p-framework] is a framework that enables Internet Service Providers (ISPs) and network application software developers to work jointly and cooperatively to optimize application communications. The goals of this cooperation are to reduce network resource consumption and to accelerate applications. To achieve these goals, P4P allows ISPs to provide network information and guidance to network applications, allowing clients to exchange data more effectively.

This document specifies the P4P protocol operations and message formats. The goal is provide a formal specification for developers to create inter-operable implementations.

1.1. Status of this Memo

The goal of this specification is to provide a snapshot of the current P4P design and implementation. Please refer to the P4P Framework document [I-D.p4p-framework] for detailed description of the design rationale and architecture. As the P4P framework is still under field trials and active development, this document will be updated to track the progress of major milestones or releases of the P4P framework.

1.2. Terminology

A detailed description of the terminology can be found in [I-D.p4p-framework]. This section provides a short list of the terminology used in this specification.

- Network Location Identifier: IP address, IP prefix, or an autonomous system number (ASN).

- PID (Partition ID): an identifier for a set of Network Location Identifiers defined by ISPs for aggregation purposes under similar network characteristics; a PID can represent different network scopes such as subnet, groups of subnets, autonomous system (AS), etc. depending on the granularity desired by an ISP.

- pDistance: a metric representing network information or preference between PIDs or Network Location Identifiers. pDistances have (optional) attributes to indicate type (e.g., routing cost, hop count, geographical distance, etc) and their interpretation (e.g., numerical or ordinal ranking).

- Location Portal Service: (described in Section 1.3.1)

- pDistance Portal Service: (described in Section 1.3.2)

1.3. Protocol Overview

The P4P framework provides two services to applications, which correspond to the two sets of information defined in the P4P Protocol: the Location Portal Service and the pDistance Portal Service.

1.3.1. Location Portal Service

The Location Portal Service provides a lookup service for the mappings between PIDs and Network Location Identifiers. There are two interfaces defined in the Location Portal Service:

- GetPID: returns the PIDs corresponding to the Network Location Identifiers queried.

- GetPIDMap: returns the lists of Network Location Identifiers contained within the PIDs queried. This allows applications to locally perform the mapping from Network Location Identifiers to their corresponding PIDs without further querying the Location Portal Service.

1.3.2. pDistance Portal Service

The pDistance Portal Service: provides a lookup service for the pDistances between given PIDs. There is a single interface:

- GetpDistance: returns the pDistances between given PID pairs or between given Network Location Identifier pairs.

1.4. Common Application Scenario

A common usage scenario is for a network application, such as a peer-to-peer application, to use the P4P services to determine the order of communication preferences among a pool of available nodes that can provide the desired contents or services.

One possibility is for an application to rely on the pDistance Portal Service alone by using Network Location Identifiers directly in the query. The returned pDistances may then be used by the application to specify the order in which target nodes are contacted. This use case may raise privacy and scalability issues due to inclusion of private information in requests and frequent queries.

A second possibility is that the application queries the Location Portal Service to first obtain mappings between PIDs and network

nodes. These PID mappings may remain stable for a longer period of time. The application can then query the pDistance Portal Service to obtain pDistances between the target PIDs and its own PIDs, and rank the network nodes accordingly. The pDistance information may be refreshed at a smaller timescale than PID mappings.

The introduction of PIDs as an aggregation point can reduce redundant lookups among nodes belong to the PIDs where pDistances are known (from prior lookups). The separation of the Location Portal Service and the pDistance Portal Service provides a clean differentiation between the two basic types of information in P4P, which can be updated at different timescales.

1.5. Key Features

While the P4P Framework does not depend on any particular transport or message formats and encodings, the current P4P protocol is implemented primarily considering ease of application integration, caching of network information (Section 5.4), and authentication and encryption (Section 6.1). Also see Section 5.5 for further discussion.

2. Conventions Used in This Document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

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 [RFC2119].

3. Messages

This section formally specifies the P4P protocol messages that implement the P4P interfaces presented in Section 1.3. This section presents encodings for data types used in the messages, and then defines the messages themselves.

The current P4P protocol uses textual encodings for request and response messages. The following definitions use the Augmented BNF and Core Rules in [RFC4234] to specify the encodings.

3.1. Definitions

The P4P interfaces make use of data types such as IP addresses and PIDs. We first define the encodings used for these data types.

3.1.1. Basic Types

P4P data type definitions use some basic data types:

```
    ACHAR       = ALPHA                        ; Alphanumeric character
                / DIGIT
    SCHAR       = ACHAR                        ; Alphanumeric character
                / "-"                          ;   or hyphen

    ASTRING     = 1*ACHAR                      ; Alphanumeric string

    NZDIGIT     = %x31-39                      ; Non-zero digit

    UINT        = NZDIGIT *DIGIT               ; Unsigned integer
                / "0"
```

Note that when a particular definition requires an unsigned integer with a particular range (e.g., 16-bit unsigned integer with range 0 - 65535), its format is indicated as UINT-K where K is the size in bits (e.g., UINT-16).

3.1.2. IP Addresses

IPv4 addresses and prefixes use their standard textual representation:

```
    ipv4-addr   = UINT-8 3("." UINT-8)         ; IPv4 address
    ipv4-prfx   = ipv4-addr "/" UINT-5         ; IPv4 prefix
```

IPv6 addresses and prefixes use the standard textual representations as specified in Sections 2.2 and 2.3 of [RFC2373]. These representations are indicated in this document as 'ipv6-addr' and ipv6-prfx', respectively.

IP Addresses and Prefixes may either be IPv4 or IPv6:

```
    ip-addr     = ipv4-addr                    ; IP address
                / ipv6-addr
    ip-prfx     = ipv4-prfx                    ; IP prefix
                / ipv6-prfx
```

3.1.3. Autonomous System Numbers

Autonomous System numbers may either be 16-bit or 32-bit:

```
      asn16       = "AS" UINT-16               ; 16-bit ASN
      asn32       = "AS" UINT-16 "." UINT-16   ; 32-bit ASN

      asn         = asn16                      ; 16-bit or 32-bit ASN
                  / asn32
```

3.1.4. Network Location Identifiers

Network Location Identifiers can be either IPv4 or IPv6 addresses or prefixes, as well as Autonomous System numbers:

```
      netloc-id   = ip-addr                    ; Network Location
                  / ip-prfx                    ;   Identifier
                  / asn
```

3.1.5. ISP Identifiers

ISP identifiers follow standard domain name syntax:

```
      hostname    = ACHAR *(*SCHAR ACHAR)      ; Hostname
      tld         = 1*ACHAR                    ; Top-level Domain
      domainname  = 1*(hostname ".") tld       ; Domain name

      isp-id      = domainname                 ; ISP identifier
```

3.1.6. PIDs

PIDs use an indicator to specify whether they represent intradomain ("internal") or interdomain ("external") network locations:

```
      pid-ind-int = "i"                        ; Internal PID indicator
      pid-ind-ext = "e"                        ; External PID indicator

      pid-ind     = pid-ind-int                ; PID indicator
                  / pid-ind-ext
```

A PID name is fully-specified by a 16-bit integer, its indicator, and ISP identifier:

```
      pid         = UINT-16 "." pid-ind "." isp-id
                                               ; PID
```

3.1.7. pDistance Values

pDistance values are 16-bit unsigned integers:

```
      pdist       = UINT-16                    ; pDistance value
```

3.1.8. pDistance Endpoint

pDistances are configured between pDistance Endpoints, which may be PIDs or specialized to Network Location Identifiers:

```
    pdist-endp  = pid                      ; pDistance Endpoint
                / netloc-id
```

3.2. Syntax

This section formally defines the message formats used by the P4P interfaces. The P4P protocol operates over HTTP 1.0 [RFC1945] or 1.1 [RFC2616]. Thus, this specification defines the following components of the request and response messages:

- Request Method
- Request URI Path and Query String
- Request Data
- Response Data

3.2.1. Headers

In addition to the components of individual messages defined in this section, the P4P protocol defines additional headers.

3.2.1.1. PIDMap Version Tag Response Header

A PIDMap Version Tag (discussed in Section 4.2.1.1) is specified in response messages from a Portal Server using the header:

```
    X-P4P-PIDMap: <UINT-32>
```

where <UINT-32> is a string following the 'UINT-32' format.

3.2.1.2. pDistance Type Response Header

The Type of pDistances contained in a response from the pDistance Portal Service is specified using the header:

```
    X-P4P-pDistType: <ASTRING>
```

where <ASTRING> is a string following the 'ASTRING' format.

3.2.1.3. pDistance Mode Response Header

The Mode of pDistances contained in a response from the pDistance Portal Service is specified using the header:

```
    X-P4P-pDistMode: <ASTRING>
```

where <ASTRING> is a string following the 'ASTRING' format.

3.2.2. Content-Type

The data contained in the request and response messages MUST use a Content-Type of 'text/plain'. The standard HTTP mechanisms for encoding (e.g., Content-Encoding and Transfer-Encoding) the data MAY additionally be applied as indicated by the HTTP standard.

3.2.3. GetPID and PID Messages

3.2.3.1. GetPID Request

The GetPID message requests the PIDs corresponding to a set of Network Location Identifiers.

The format of the Request Data is:

```
    getpid-data     = *(netloc-id CRLF)
```

The GetPID message is then specified as:

```
    Request Method: POST (may be GET if Request Data is empty)
    Request URI:    /pid
    Request Data:   getpid-data
```

3.2.3.2. PID Response

The PID message is returned by a Portal Server in response to a GetPID request and provides the PID for the requested Network Location Identifiers.

The format of the Response Data is:

```
    pid-data        = 1*(netloc-id WSP pid CRLF)
```

The PID message is specified as:

```
    Response Data:  pid-data
```

3.2.4. GetPIDMap and PIDMap Messages

3.2.4.1. GetPIDMap Request

The GetPIDMap message requests the Network Location Identifiers contained within PIDs.

The format of the Request Data is:

```
getpidmap-data    = *(pid CRLF)
```

The GetPIDMap message is specified as:

```
Request Method: POST (may be GET if Request Data is empty)
Request URI:    /pid/map
Request Data:   getpidmap-data
```

3.2.4.2. PIDMap Response

The PIDMap message is returned by a Portal Server in response to a GetPIDMap request and provides the list of Network Location Identifiers for each of the requested PIDs.

Each line of the Response Data contains a PID and the Network Location Identifiers contained in the PID. The count of Network Location Identifiers in the list is also included to simplify processing. The format of the Response Data is:

```
pidmap-line     = pid WSP UINT-32 1*(WSP netloc-id)

pidmap-data     = *(pidmap-line CRLF)
```

The PIDMap message is specified as:

```
Response Data:  pidmap-data
```

3.2.5. GetpDistance and pDistance Messages

3.2.5.1. GetpDistance Request

The GetpDistance message requests pDistances of the specified type between pDistance Endpoints (i.e., a list of Source Endpoint -> Destination Endpoint pairs).

pDistance Type and Mode are optionally specified as a query string arguments in the Request URI.

Conceptually, the message data specifies a list of Source ->

Destination pairs in the Request Data. For efficiency, however, a more compact representation is used.

Each line of the Request Data encodes a request for the pDistances from a particular Source Endpoint to a list of Destination Endpoints. By specifying an "inc-reverse" option, the pDistances from the Destination Endpoints to the Source Endpoint may also be requested. The count of Destination Endpoints in the list is also included to simplify processing. The format of the Request Data is:

```
    getpdist-line   = pdist-endp
                        WSP ("inc-reverse" / "no-reverse")
                        WSP UINT-32
                        1*(WSP pdist-endp)

    getpdist-data   = *(getpdist-line CRLF)
```

The GetpDistance message is then specified as:

```
    Request Method: POST (may be GET if Request Data is empty)
    Request URI:    /pdistance?type=<ASTRING>&mode=<ASTRING>&direct
    Request Data:   getpdist-data
```

where <ASTRING> indicates a string following the 'ASTRING' format. The 'direct' parameter indicates that pDistance Endpoints are Network Location Identifiers instead of PIDs.

The 'type', 'mode', and 'direct' Request URI query string parameters are optional. Section 4.2.4 indicates the default behavior if not explicitly supplied.

3.2.5.2. pDistance Response

The pDistance message is returned by a Portal Server in response to a GetpDistance request and specifies the pDistances for the requested Source Endpoint -> Destination Endpoint pairs.

The encoding for the Response Data follows a similar pattern as the GetpDistance message Request Data.

Each line of the Response Data specifies the pDistances from a Source Endpoint to a list of Destination Endpoints. The pDistance to a Destination Endpoint is encoded directly following Destination Endpoint in the list. If the reverse option is "inc-reverse", a second pDistance is included indicating the pDistance from the Destination Endpoint to the Source Endpoint. The format of the Response Data is:

```
      dst-pdist       = pdist-endp 1*2(WSP pdist)

      pdist-line      = pdist-endp
                          WSP ("inc-reverse" / "no-reverse")
                          WSP UINT-32
                          1*(WSP dst-pdist)

      pdist-data      = *(pdist-line CRLF)
```

The pDistance message is specified as:

```
      Response Data:  pdist-data
```

4. Protocol Operations

The P4P Protocol is a simple request/response protocol. This section first discusses standard definitions such as well-known values, and then defines message and error handling.

4.1. Standard Definitions and Reserved Values

4.1.1. PIDs

4.1.1.1. Well-Known PIDs

Some PID names are well-known and used for specific purposes. These PIDs use ISP Identifier "pid.p4p".

4.1.1.1.1. Default Aggregation PID

Each Portal Server MUST define a PID which implicitly contains all Network Location Identifiers not contained by other Aggregation PIDs. This PID has the name:

```
      0.i.pid.p4p
```

4.1.1.2. Routing Cost PIDs

The pDistance Portal Service allows applications to query pDistances between PIDs. We use the term Routing Cost PID to refer to a PID for which Routing Cost pDistances are defined. As defined later in Section 4.2.4, a "default" request to the GetpDistance interface returns the Routing Cost pDistances between each pair of PIDs. The full set of PIDs contained in this response message is the full set of Routing Cost PIDs.

4.1.2. pDistances

4.1.2.1. Reserved pDistance Types

The pDistance Portal Service may define pDistances of multiple types. Specific pDistance types have reserved names beginning with "p4p".

4.1.2.1.1. Routing Cost pDistance

Each Portal Server MUST define the Routing Cost pDistance type. This type uses the name:

```
p4proutingcost
```

4.1.2.2. Reserved pDistance Modes

pDistances have an attribute, called a Mode, indicating how they should be interpreted. Modes for both numerical and ordinal pDistances have reserved names.

4.1.2.2.1. Numerical pDistances

Each Portal Server MUST reserve the following pDistance Mode to indicate numerical pDistances:

```
p4pnumerical
```

Numerical pDistances are defined such that a smaller pDistance value indicates a higher preference, while larger pDistance values indicates a lower preference.

4.1.2.2.2. Ordinal pDistances

Each Portal Server MUST reserve the following pDistance Mode to indicate ordinal pDistances:

```
p4pordinal
```

Ordinal pDistances are defined such that a smaller pDistance value indicates a higher preference, while a larger pDistance value indicates a lower preference.

4.2. Message Handling

This section further defines P4P interfaces by detailing the semantics applied to the P4P messages discussed in Section 3.2.

4.2.1. Common Operations

In addition to the specific message handling behavior discussed later in this section, certain common operations apply to all P4P interfaces.

4.2.1.1. PIDMap Version Tag

Recall that P4P information is separated into two services, and that information provided by the pDistance Portal Service may be dependent on current PID mappings provided by the Location Portal Service. Applications may query the services independently, but should also ensure that they use consistent information.

PIDMap Version Tags are opaque identifiers that allow an application to detect when previously-retrieved PID mappings are no longer valid. Conceptually, a Portal Server maintains a database, called the PIDMap, containing the mappings between Network Location Identifiers and PIDs. All responses from the Location Portal Service and pDistance Portal Service include the Version Tag of the PIDMap used to generate the response. If the Version Tag for pDistance information received by an application does not match the Version Tag for the stored PID mappings, the PID mappings should be updated from the Location Portal Service.

One way to implement the Version Tag is as an integer which is incremented when the PIDMap is changed at the Portal Server. The integer can wrap around to 0 if necessary.

Each non-error response message from a Portal Server MUST include a 'X-P4P-PIDMap' header with its value being the Version Tag of the PIDMap used to generate the response.

4.2.1.2. Successful Responses

A Portal Server MUST use HTTP Status Code 200 when replying to an operation that completed successfully. Note that this includes cases where the Portal Server responds with only a subset of the requested information, as discussed later in this section. The requesting application is expected to handle such cases if necessary.

4.2.2. GetPID

A P4P Portal Server MUST implement the GetPID interface.

The GetPID interface is defined to allow the Portal Server to directly return the PIDs for the supplied list of Network Location Identifiers. In the absence of an error condition specified in

Section 4.3, the Portal Server MUST respond with a PID message specifying a PID for each queried Network Location Identifier supplied in the GetPID request message.

4.2.2.1. Empty Requests

If the GetPID request message is empty (i.e., it contains a zero-length list of Network Location Identifiers), the Portal Server MUST interpret the request as if the list of Network Location Identifiers contained the IP address of the requestor as its only element.

This provides an easy mechanism for a client to lookup its own PID even when it is behind a NAT or has multiple network interfaces.

4.2.3. GetPIDMap

A P4P Portal Server SHOULD implement the GetPIDMap interface.

The GetPIDMap interface is defined to provide an application information such that it can locally map between Network Location Identifiers and PIDs. In the absence of an error condition specified in Section 4.3, the Portal Server MUST respond with a PIDMap message containing lists of Network Location Identifiers for at least the Routing Cost PIDs supplied in the GetPIDMap request message. If the request specifies a non-empty list of PIDs, the Portal Server MUST NOT respond with lists of Network Location Identifiers for PIDs not contained in the request.

4.2.3.1. Empty Requests

If the GetPIDMap request message is empty (i.e., it contains a zero-length list of PIDs), the Portal Server MUST interpret the request as if the list of PIDs were the full set of Routing Cost PIDs.

4.2.3.2. Non-Routing Cost PIDs

If the GetPIDMap request message contains PIDs that are not in the set of Routing Cost PIDs, the Portal Server MAY interpret the request as if the list of PIDs did not contain such PIDs.

4.2.3.3. Network Location Identifier Lists

The Network Location Identifiers returned by the Portal Server SHOULD, where possible, allow applications to locally obtain equivalent mappings between PIDs and Network Location Identifiers as would be obtained using the GetPID interface. If the list of Network Location Identifiers contains AS numbers, the Portal Server SHOULD ensure that this mapping can be done by applications with reasonable

accuracy with publicly-available information (e.g., public databases).

4.2.4. GetpDistance

A P4P Portal Server MUST implement the GetpDistance interface for pDistances amongst PIDs. A P4P Portal Server MAY implement the GetpDistance interface for pDistances directly between Network Location Identifiers.

The GetpDistance interface provides pDistances between PIDs defined by the Location Portal Service. It may also be used to directly query the pDistances between Network Location Identifiers. In the absence of an error condition specified in Section 4.3, the Portal Server MUST respond with a pDistance message containing pDistances of the requested type for all requested Source Endpoint -> Destination Endpoint pairs for which the pDistance type is defined. If the request specifies a non-empty list of Source Endpoint -> Destination Endpoint pairs, the Portal Server MUST NOT respond with pDistances for pairs not contained in the request.

4.2.4.1. Endpoint Types

If the GetpDistance request message does not specify the 'direct' query string parameter, the Portal Server MUST parse all endpoints in the Request Data as PIDs (and hence follow the 'pid' syntax). If the 'direct' query string parameter is specified, the Portal Server MUST parse all endpoints in the Request Data as Network Location Identifiers (and hence follow the 'netloc-id' syntax). If an endpoint in the request is found to not meet the expected format, the Portal Server MUST reject the request as being incorrectly formatted (see Section 4.3.2).

4.2.4.2. Invalid PID Pairs

If the GetpDistance request message contains PIDs that are not in the set of PIDs that define pDistances of the requested type, the Portal Server MAY interpret the request as if the list of Source PID -> Destination PID pairs did not contain pairs referring to such PIDs.

4.2.4.3. Network Location Identifier Endpoints

If the 'direct' query string parameter is specified, the the Portal Server MAY return customized pDistances instead of pDistances amongst the PIDs that contain the Network Location Identifiers.

If a Portal Server does not implement the GetpDistance query for Network Location Identifiers, it MUST reply with a HTTP 501 (Not

Implemented) status code.

4.2.4.4. Default pDistance Type

If the GetpDistance request message does not specify a pDistance type via a 'type' query string parameter, the Portal Server MUST interpret the message as if it specified the type as 'p4proutingcost'.

4.2.4.5. Unsupported pDistance Type

If the GetpDistance request message specifies a pDistance type that is not supported by the Portal Server, the Portal Server MUST reply with a HTTP 501 (Not Implemented) status code.

4.2.4.6. pDistance Type Handling

The pDistances encoded in the response message MUST be pDistances with the Type specified in the request message. If the pDistances encoded in the response message are not Routing Cost pDistances, the Portal Server MUST specify the returned pDistances' Type using the 'X-P4P-pDistType' header.

4.2.4.7. Default pDistance Mode

If the GetpDistance request message does not specify a pDistance Mode via a 'mode' query string parameter, the Portal Server MUST interpret the message as if it specified the mode as 'p4pnumerical'.

4.2.4.8. Unsupported pDistance Mode

If the GetPDistance request message specifies a pDistance Mode that is not supported, the Portal Server MUST reply with pDistances with either a Mode of 'p4pnumerical' or 'p4pordinal'. Thus, a Portal Server must implement at least one of 'p4pnumerical' or 'p4pordinal' pDistances, but it may choose which to support.

4.2.4.9. pDistance Mode Handling

The pDistances encoded in the response message SHOULD be pDistances with the Mode specified in the request message. If the pDistances encoded in the response message are not numerical pDistances, the Portal Server MUST specify the returned pDistances' Mode using the 'X-P4P-pDistMode' header.

4.2.4.10. Empty Requests

If the GetpDistance request message is empty (i.e., it contains no Source Endpoint -> Destination Endpoint pairs) and the 'direct' query

string parameter is not specified, the Portal Server MUST interpret the request as if the list of PIDs were the full set of PIDs that define pDistances of the requested type.

If the request message is empty and the 'direct' query string parameter is specified, the Portal Server MUST reject the request as being incorrectly formatted (see Section 4.3.2).

4.3. Exception Handling

This section specifies Portal Server behavior for specific error conditions that may be encountered. Standard HTTP status codes are returned by a Portal Server. The Portal Server MUST follow the HTTP protocol version in use for the current request for error conditions (e.g., indicating server overload conditions) not explicitly listed in this section.

4.3.1. Invalid Request URI Path

If the Path portion of the Request URI does not refer to a valid P4P interface, the Portal Server MUST return an HTTP 404 (Not Found) status code.

4.3.2. Invalid Request Format

If the Request Data or a Request URI Query String parameter is formatted incorrectly (i.e., it does not follow the syntax in Section 3.2 or it fails to meet additional requirements specified in Section 4.2), the Portal Server MUST return an HTTP 400 (Bad Request) status code.

4.4. Timers

The P4P protocol is simple request/response protocol and hence does not require any additional timers beyond those required by the underlying protocol (i.e., HTTP and TCP).

4.5. Message Exchange Examples

This section presents example message captures from the P4P protocol. Note that the message captures use HTTP chunked encoding for requests and responses. This is an implementation detail and does not imply that the P4P protocol must use chunked encoding.

The message exchange examples in this section use a Portal Server configured with the following simple, illustrative topology. Labels on arrows between PIDs indicate pDistances. The pDistance from a PID to itself is configured to be 1. Note that the Portal Server reports

end-to-end pDistances. The method and factors (including, but not limited to, algorithm and routing policy) for computing end-to-end pDistances is a policy decision implemented by the Portal Server, and is outside the scope of this document. These examples are only provided to illustrate message format.

```
    .-------------.
    | 4.e.isp.net |
    '-------------'
           ^
           | 60
           v
    .-------------.    8     .-------------.
    | 0.i.isp.net | <------> | 3.i.isp.net |
    '-------------'          '-------------'
           ^                        ^
           | 4                      | 4
           v                        v
    .-------------.    10    .-------------.  40  .-------------.
    | 1.i.isp.net | <------> | 2.i.isp.net | <--> | 5.e.isp.net |
    '-------------'          '-------------'      '-------------'
```

Each PID has a set of Network Location Identifiers configured:

```
    0.i.isp.net : 10.0.0.0/24 10.0.1.0/24
    1.i.isp.net : 10.1.0.0/16
    2.i.isp.net : 10.2.0.0/24 10.2.1.0/24
    3.i.isp.net : 10.3.0.0/24
    4.e.isp.net : 172.16.0.0/12
    5.e.isp.net : 192.168.0.0/16
```

4.5.1. GetPID

4.5.1.1. Request PID for Own IP Address

The following message exchange illustrates a client requesting its own PID from a Portal Server. The client uses an empty request, and the Portal Server responds with the client's IP address and the PID corresponding to the IP address.

```
      C: POST /pid HTTP/1.1
      C: Host: localhost:6671
      C: Accept: */*
      C: Transfer-Encoding: chunked
      C: Expect: 100-continue

      S: HTTP/1.1 100 Continue

      C: 2

      C: 0

      S: HTTP/1.1 200 OK
      S: Transfer-Encoding: chunked
      S: X-P4P-PIDMap: 1
      S: Cache-Control: max-age=604800
      S: Content-Type: text/plain
      S: Date: Tue, 24 Feb 2009 19:26:43 GMT

      S: 17
      S: 10.1.1.12 1.i.isp.net

      S: 0
```

4.5.1.2. Request PIDs for List of IP Addresses

The following message exchange illustrates a client directly asking the Portal Server to map a set of IP addresses into their corresponding PIDs.

```
      C: POST /pid HTTP/1.1
      C: Host: localhost:6671
      C: Accept: */*
      C: Transfer-Encoding: chunked
      C: Expect: 100-continue

      S: HTTP/1.1 100 Continue

      C: 1e
      C: 10.1.23.200
      C: 192.168.1.128

      C: 0

      S: HTTP/1.1 200 OK
      S: Transfer-Encoding: chunked
      S: X-P4P-PIDMap: 1
      S: Cache-Control: max-age=604800
      S: Content-Type: text/plain
      S: Date: Tue, 24 Feb 2009 19:26:47 GMT

      S: 34
      S: 10.1.23.200   1.i.isp.net
      S: 192.168.1.128 5.e.isp.net

      S: 0
```

4.5.2. GetPIDMap

4.5.2.1. Request PID Map

The following message exchange illustrates an application requesting the set of Network Location Identifiers contained within particular PIDs. Note that the application could also request for Network Location Identifiers in all Routing Cost PIDs by using an empty request.

```
       C: GET /pid/map HTTP/1.1
       C: Host: localhost:6671
       C: Accept: */*
       C: Transfer-Encoding: chunked
       C: Expect: 100-continue

       S: HTTP/1.1 100 Continue

       C: 1c
       C: 0.i.isp.net
       C: 2.i.isp.net

       C: 0

       S: HTTP/1.1 200 OK
       S: Transfer-Encoding: chunked
       S: X-P4P-PIDMap: 1
       S: Cache-Control: max-age=604800
       S: Content-Type: text/plain
       S: Date: Tue, 24 Feb 2009 19:26:55 GMT

       S: 4E
       S: 0.i.isp.net   2   10.0.0.0/24 10.0.1.0/24
       S: 2.i.isp.net   2   10.2.0.0/24 10.2.1.0/24

       S: 0
```

4.5.3. GetpDistance

4.5.3.1. Request pDistance Among PIDs

The following message exchange illustrates an application requesting Routing Cost pDistances between particular PIDs. Note that the application could also request pDistances amongst all Routing Cost PIDs by using an empty request.

```
      C: POST /pdistance HTTP/1.1
      C: Host: localhost:6671
      C: Accept: */*
      C: Transfer-Encoding: chunked
      C: Expect: 100-continue

      S: HTTP/1.1 100 Continue

      C: 98
      C: 0.i.isp.net   no-reverse  1   2.i.isp.net
      C: 1.i.isp.net   no-reverse  1   5.e.isp.net
      C: 2.i.isp.net   no-reverse  1   0.i.isp.net
      C: 3.i.isp.net   no-reverse  1   4.e.isp.net

      C: 0

      S: HTTP/1.1 200 OK
      S: Transfer-Encoding: chunked
      S: X-P4P-PIDMap: 1
      S: Cache-Control: max-age=7200
      S: Content-Type: text/plain
      S: Date: Tue, 24 Feb 2009 19:26:39 GMT

      S: A4
      S: 0.i.isp.net   no-reverse  1   2.i.isp.net 14
      S: 1.i.isp.net   no-reverse  1   5.e.isp.net 50
      S: 2.i.isp.net   no-reverse  1   0.i.isp.net 14
      S: 3.i.isp.net   no-reverse  1   4.e.isp.net 68

      S: 0
```

5. Discussions

5.1. Discovery

To make use of a P4P Portal Server, an application must first be able to identify the address and port on which the server is running. The discovery mechanism is not part of the P4P protocol specification as it can be provided as a modular component in the framework. Several existing protocols, such as DNS, DHCP, or IP multicast, can be used to discover the service locations of the P4P Portal Servers. This section briefly describes the discovery mechanism used by the current P4P implementation.

The P4P prototype is (as of this writing) deployed with a small number of active Portal Servers. Thus, a simple centralized discovery mechanism is used for clients that must discover a Portal Server. Manual configuration is used for tracker-based integrations,

by configuring Application Trackers with addresses of available Portal Servers. This mechanism is independent of the protocol messages exchanged between applications and Portal Servers, and hence can easily be replaced by another mechanism (e.g., as recommended by ALTO).

5.2. Delegation

During P4P field tests, ISPs have proposed the possibility of delegation, in which an ISP provides information for customer networks which do not wish to run Portal Servers themselves. A consideration for delegation is that customer networks may wish to explicitly configure such delegation.

5.3. Load Balancing Considerations

Due to a large volume of requests or fault tolerance concerns, it may an ISP may wish to provide multiple Portal Servers to serve requests. The current P4P protocol only requests information from Portal Servers, so it is straightforward to use existing load balancing techniques and/or providing redundant backup Portal Servers.

5.4. Caching P4P Information

P4P information can include parameters controlling the lifetime and caching options. In particular, the standard HTTP Expires (for HTTP 1.0 and 1.1) and Cache-Control (for HTTP 1.1) headers MAY be included in response messages from Portal Services. Portal Servers MUST NOT include Cache-Control headers enabling caching in responses to non-empty requests. The semantics applied to the Expires and Cache-Control headers follow the interpretation in the standard HTTP protocol.

Requests to Portal Services MAY include Cache-Control headers to serve as instructions to the Portal Server. The Portal Server MUST follow standard HTTP behavior in response to such headers. Note that this includes the possibility of ignoring the instruction and including a Warning header in the response message.

5.5. Transport and Encoding Considerations

The P4P framework does not depend on any particular message transport or encoding. However, the current P4P Protocol uses HTTP since it is widely-implemented and directly provides or integrates with much of the desired functionality:

o Authentication and Encryption: HTTP directly provides Basic and Digest authentication options. Existing implementations also

commonly integrate SSL/TLS which can also provide authentication and encryption. See Section 6.1 for further discussion.

- Caching: Network information may be easily cached to reduce load on a Portal Server. The current P4P protocol formats requests and responses (by specifying operations and parameters in the Request URI) such that they may be cached using existing HTTP cache servers. As discussed in Section 5.4, Portal Servers indicate the lifetime of P4P information, and the same caching parameters indicate to applications how long P4P information is valid before it should be refreshed.

Note, however, that other transports or message encodings may have benefits in certain (e.g., UDP for small messages).

6. Security Considerations

6.1. Protecting P4P Information

A Portal Server can optionally control access to P4P information to specific users or applications. Additionally, the transport of such information may be encrypted. This section discusses the authentication and encryption as they relate to the P4P protocol. Note that authorization is outside the scope of this document.

Note that the discovery mechanism may need to account for certain Portal Server capabilities (e.g., SSL/TLS).

6.1.1. Authentication

If a Portal Server wishes the P4P interfaces to be accessible to particular users or applications, it MAY use either a standard HTTP authentication techniques (e.g., Basic and Digest), or SSL/TLS.

6.1.2. Encryption

If a Portal Server wishes requests and responses to be encrypted, it MAY use standard SSL/TLS techniques.

6.2. ISPs

Additional security consideration for ISPs lies in the potential risk of disclosing network topology and provisioning information through PIDs and pDistances. ISPs must evaluate how much information to reveal and the associated risks. For example, if an ISP reveals extremely fine-grained information, it may be easier for attackers to infer network topology information. ISPs should also take into account that revealing overly coarse-grained information may not

provide benefits to either them or applications.

6.3. Clients

There are two possible security concerns for the clients: privacy and malicious P4P providers. First, clients can potentially disclose private information to the P4P Service Portals if either PIDs are extremely fine-grained or Network Location Identifiers are included directly in the query. In such a case, ISPs may be able to infer from the queries the communication patterns of a client. One possibility is for clients to only retrieve the full set of PIDs (via GetPIDMap) and pDistances (via GetpDistance).

Second, a malicious or ineffective P4P service provider could lead to bad application performance or, in extreme cases, denial of service. Clients may use other mechanisms to complement P4P information, or replace or ignore P4P information if it is ineffective.

7. IANA Considerations

The P4P protocol includes identifiers and well-known values that may be assigned by the IANA. However, as the P4P framework is still under field trials and active development, this current specification does not cover such policies. This document will be updated to include any IANA considerations at a later point.

8. Conclusions

The main contribution of the P4P Framework is to establish a communication channel between network applications and the infrastructure providers (ISPs). The current implementation focuses on providing the services to query PIDs for aggregation and pDistances for network information and preferences among PIDs for communication. This document provides a formal specification of the detailed operations and message formats for the base P4P protocol used in the P4P framework.

9. References

9.1. Normative References

[I-D.p4p-framework] Alimi, R., Pasko, D., Popkin, L., Wang, Y., and Y. Yang, "P4P: Provider Portal for P2P Applications", draft-p4p-framework-00 (work in progress), November 2008.

[RFC1945] Berners-Lee, T., Fielding, R., and H. Nielsen, "Hypertext Transfer Protocol -- HTTP/1.0",

RFC 1945, May 1996.

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

[RFC2373] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

9.2. Informative References

[SIGCOMM08] H. Xie, Y.R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz., "P4P: Provider Portal for (P2P) Applications", In ACM SIGCOMM. 2008.

Appendix A. Contributors

The P4P project includes contributions from many members of the P4P Working Group, hosted by DCIA.

The individuals involved in the design and P4P field tests include (in alphabetical order):

- Richard Alimi, Yale University
- Alex Gerber, AT&T
- Chris Griffiths, Comcast
- Ramit Hora, Pando Networks
- Arvind Krishnamurthy, University of Washington
- Y. Grace Liu, IBM Watson
- Jason Livingood, Comcast
- Michael Merritt, AT&T

- o Doug Pasko, Verizon
- o Reinaldo Penno, Juniper Networks
- o Laird Popkin, Pando Networks
- o Stefano Previdi, Cisco
- o Satish Raghunath, Juniper Networks
- o James Royalty, Pando Networks
- o Thomas Scholl, AT&T
- o Emilio Sepulveda, Telefonica
- o Stanislav Shalunov, BitTorrent
- o Avi Silberschatz, Yale
- o Hassan Sipra, Bell Canada
- o Haibin Song, Huawei
- o Oliver Spatscheck, AT&T
- o Jia Wang, AT&T
- o Richard Woundy, Comcast
- o Hao Wang, Yale University
- o Ye Wang, Yale University
- o Haiyong Xie, Yale University
- o Y. Richard Yang, Yale University

Appendix B. Acknowledgments

The authors would like to thank the members of the P4P Working Group for their collaboration, and the members of the p2pi mailing list for their comments and questions. We would like to think Marty Lafferty from DCIA, Erran Li, Jin Li, and See-Mong Tang for giving us excellent feedback.

We would also like to thank David Zhang from PPLive for identifying the need for PIDMap Version Tags.

Authors' Addresses

Yu-Shun Wang
Microsoft Corp.
One Microsoft Way
Redmond, WA 98052
USA

EMail: yu-shun.wang@microsoft.com

Richard Alimi
Yale University

EMail: richard.alimi@yale.edu

Doug Pasko
Verizon

EMail: doug.pasko@verizon.com

Laird Popkin
Pando Networks, Inc.

EMail: laird@pando.com

Y. Richard Yang
Yale University

EMail: yry@cs.yale.edu