

P4P Portal: Configuration Manual

Yale Laboratory of Networked Systems

April 3, 2010

Abstract

This document provides information about configuring the P4P Portal services for your network.

Contents

1	Introduction	2
1.1	Prerequisites	2
2	Portal Server Configuration	2
2.1	p4p-portal.conf	2
2.2	p4p-portal-intf.conf	3
3	Portal Service Configuration	3
3.1	Configuration Syntax	3
3.2	Starting the Shell	4
3.3	Configuration Walkthrough	4
3.3.1	ISP Identifier	4
3.3.2	PID-Level Network Topology	4
3.3.3	pDistances	5
3.4	Loading a Configuration File	7
3.5	Configuration Verification and Validation	7
3.5.1	Viewing Details	7
3.5.2	Configuration Analysis	8
3.5.3	Topology Visualization	8
3.5.4	Other Utilities	8
3.6	Converting Old Configurations	9
3.6.1	Convert 1.0 to 2.0	9
3.6.2	Convert 2.0 to 2.1	9

1 Introduction

The P4P Portal Server provides applications information to better utilize network resources. This manual begins by discussing how to run the P4P Portal Server, and then discusses how to configure the network information it provides to applications.

1.1 Prerequisites

Before reading this guide, you should be familiar with the P4P framework. The framework, including terminology and architecture, is discussed in:

<http://p4p.cs.yale.edu/projects/p4p/draft-p4p-framework.txt>

This guide currently assumes you are running Linux, and that you have installed at least the following P4P packages on your machine:

- `p4p-portal-server`
- `p4p-portal-shell`
- `p4p-portal-utils`
- `p4p-portal-docs`

2 Portal Server Configuration

After you have installed the Portal server, it must be configured for your system.

There are two configuration files used by the Portal server:

- `p4p-portal.conf`
- `p4p-portal-intf.conf`

The Portal server will search for these files in the following directories:

1. Same directory as the `p4p_portal` binary
2. The `../conf` directory, relative to the location of the `p4p_portal` binary
3. `/etc/p4p`

The next two subsections detail these two configuration files.

2.1 `p4p-portal.conf`

This file controls general runtime options for the Portal server. The Portal server accepts a number of command-line arguments. They can either be supplied when you start the server, or you may specify them in the `/etc/p4p/p4p-portal.conf` configuration file.

The configuration file is formatted as a set of `name = value` pairs such as the following example:

```
job-threads = 4
admin-txn-timeout = 20
daemon = 1
```

You may view the list of available options by running the command:

```
$ p4p_portal --help
```

2.2 p4p-portal-intf.conf

The Portal Server allows services to be exposed on different ports. Ports may either be defined for use by clients, or for administration purposes. In particular, client ports expose the Location Portal and pDistance Portal Services to clients. Administration ports expose specialized administration services that are used remotely to configure and control a running Portal Server. With such a setup, it is easy to provide network-level security for the administration services via firewall rules.

The `p4p-portal-intf.conf` file controls which services are exposed, and for each service, the port number through which it is accessed. For example, you may allow client services to be exposed on port 6671 and administration services to be exposed on port 6672.

Such a policy can be achieved with the following lines:

```
[Admin]
type = REST
port = 6672
threads = 2
allow-admin = true
```

```
[Client-REST]
type = REST
port = 6671
threads = 4
```

You may view the list of available options for each interface by running the command:

```
$ p4p_portal --interface-help
```

3 Portal Service Configuration

The P4P Portal services provided by your Portal Server expose P4P information of your network to P2P applications. This information must be loaded to the Portal Server through Portal service configuration.

We have provided an interactive shell through which you can load a Portal service configuration file and make manual updates, as well as monitor and debug the active configuration.

The Portal Shell may be run either on the same computer as the Portal Server or another computer with network access to a port enabled for administration (see Section 2.2).

3.1 Configuration Syntax

The shell uses a Cisco IOS-like command syntax for updating and displaying the running configuration of your Portal service.

Section 3.3 provides a brief walkthrough to illustrate how to use the shell to configure your Portal service with the network information. For further details, an annotated formal specification of the configuration syntax is installed at:

```
/usr/share/doc/p4p-portal-docs-<version>/portal-shell-v2.1-spec.txt
```

This file contains annotations discussing the usage of each command. Beyond the commands specified in this document, there are other useful commands. As we will see, **show** commands allow you to display and test the running configuration (see Section 3.5.1).

3.2 Starting the Shell

The Portal Shell is installed as `p4p_portal_shell`. After starting the shell, you must first specify the address and administration port for a running Portal Server:

```
$ p4p_portal_shell
p4p# server p4p.Internet2.edu:6672
```

Another way to specify the Portal Server is via command-line arguments:

```
$ p4p_portal_shell p4p.Internet2.edu:6672
```

After starting the shell, the `p4p#` prompt is displayed and you can begin typing commands. The `help` command displays the list of available commands and their syntax:

```
p4p# help
```

3.3 Configuration Walkthrough

A P4P Portal service configuration consists of an ISP Identifier, PID-level network topology, and pDistances. We will walk through the three parts of configuration using the Internet2 network as an example. The complete example configuration file is installed at:

```
/usr/share/doc/p4p-portal-docs-<version>/abilene.conf
```

3.3.1 ISP Identifier

First, you must define an ISP Identifier. This identifier should be globally unique, because it may be used by applications to distinguish your P4P Portal service from other network operators.

We recommend using a domain name as the ISP Identifier. For example:

```
$ p4p_portal_shell p4p.Internet2.edu:6672
p4p# isp default internet2.edu
```

3.3.2 PID-Level Network Topology

Network topology information is a major part of the configuration. It impacts the information provided to applications by the Location and pDistance services.

The network topology in P4P is specified as a PID-level “my-Internet view” of network locations and their connectivity. Conceptually, the global IP address space is subdivided into PIDs.

A PID is a network partition ID. Each PID consists of a collection of IP addresses (clients). Thus, a PID can represent a PoP, metropolitan area, interdomain peering point, set of clients with similar capabilities, or some other grouping of clients.

When configuring a PID node, you can specify the set of associated IP prefixes. The PID is also given a number, name, and type (*internal* or *external*). An internal PID should only be configured with IP prefixes within your network, while an external PID can be configured with IP prefixes outside of your network (interdomain). For example:

```
p4p# pid internal 0 newy-us-abilene prefixes 128.36.0.0/16 210.150.0.0/23
p4p# pid internal 1 chic-us-abilene prefixes 12.108.127.0/24 192.16.104.0/23 172.16.0.0/12
p4p# pid external 100 peering-to-nox prefixes 10.1.0.0/16
```

PIDs that have IP prefixes associated with them will be visible to clients. You may optionally define PIDs without any prefixes, which will not be visible to users. That is, they will not be included in the PID Maps or pDistance Maps returned by the Portal Server.

After creating PID nodes, their connectivity can be specified by the `pid link` command:

```
p4p# pid link newy-us-abilene chic-us-abilene link-newy-chic routing-weight 392
p4p# pid link chic-us-abilene newy-us-abilene link-chic-newy routing-weight 392
p4p# pid link newy-us-abilene peering-to-nox link-participant-newy-nox
```

In the above example, you may notice that each intradomain PID link (between internal PIDs) has a *routing-weight* attribute. The routing-weight is used by the Portal Server to determine PID-level shortest path routing:

```
p4p# pid routing weights
```

3.3.3 pDistances

pDistances can be configured for each PID-level link in the topology. Recall that only the PIDs that have prefixes assigned to them are visible to users. The Portal Server will only return the total pDistances amongst PIDs visible to users. The total pDistance from one PID to another is computed by the Portal Server by summing the pDistance of each PID-level link along the route between the two PIDs.

Each pDistance is a floating-point value between 0.0 and 100.0.

In many scenarios, PIDs that are “closer” together should have smaller pDistances between them. In particular:

- Intra-PID pDistance is a pDistance from a PID to itself. It conveys the preference for traffic remaining within the same PID. In many scenarios, intra-PID pDistances will be the smallest pDistances in the configuration.
- Inter-PID pDistance is the total pDistance from one internal PID to another internal PID. Inter-PID pDistances convey preferences for traffic traveling from one PID in the ISP to another PID within the same ISP. These will typically be larger than intra-PID pDistances, meaning traffic between different PIDs is preferred less than traffic within the same PID.
- Interdomain pDistance is a pDistance from one internal PID to an external PID. Interdomain pDistances convey preferences for traffic traveling to external locations (*e.g.*, via an ISP’s peering

points). These will typically be larger than both intra-PID and inter-PID pDistances, meaning traffic crossing these peering points is preferred less than traffic within the ISP.

There are several configuration options available for configuring pDistances for individual PID-links:

- Static PID-link pDistance: Explicitly configure the pDistance for a single PID-level link in the topology. This may configure the pDistance for traffic within the same PID (intra-PID), between two internal PIDs (intradomain), or between an internal PID and an external PID (interdomain).

```
p4p# pdistance link link-newy-chic static 9
p4p# pdistance link link-chic-newy static 8
p4p# pdistance link link-participant-newy-nox static 12
```

- Default Intra-PID pDistance: Default pDistance from a PID to itself if not explicitly configured.

```
p4p# pdistance intra-pid default 1
```

- Default PID-link pDistance: Default pDistance for a PID-level link from one PID to a different PID if not explicitly configured.

```
p4p# pdistance pid-link default 10
```

Finally, there are configuration options controlling the computed total pDistance amongst PIDs:

- Default Inter-PID pDistance: Default pDistance of two internal PIDs if there are no routes between them (*i.e.*, no set of PID-level links in the topology connect them).

```
p4p# pdistance inter-pid default 10
```

- Default Interdomain pDistance: pDistance from an internal PID to an external PID if they are connected by a PID-link but no link pDistance is specified:

```
p4p# pdistance interdomain default 65
```

- Interdomain pDistance Policy: Controls how pDistances to external PIDs are computed. By default, the pDistance from an internal PID to an external PID is the sum of the pDistance from the source PID to the “egress” internal PID and the pDistance of the link between “egress” PID and the external destination PID. You may optionally exclude the “intradomain” part of interdomain pDistance with the following command:

```
p4p# pdistance interdomain exclude intradomain
```

After you load or manually modify the configuration, you can force the Portal Server to (re)compute the end-to-end pDistances immediately:

```
p4p# pdistance update
```

Otherwise, the pDistances will be automatically updated periodically.

3.4 Loading a Configuration File

Once a Portal service configuration file is created, it can be loaded into a running Portal Server via the shell:

```
$ p4p_portal_shell p4p.Internet2.edu:6672
p4p# load abilene.conf
```

It is also possible to load a configuration file automatically when initially starting the shell:

```
$ p4p_portal_shell p4p.Internet2.edu:6672 abilene.conf
```

3.5 Configuration Verification and Validation

Once you have generated and installed your configuration, the installed configuration can be displayed and validated.

3.5.1 Viewing Details

The `show` commands can display the running configuration of your Portal service allowing you to verify the installed configuration.

To show the internal view of network topology information:

```
p4p# show topology nodes
p4p# show topology links
```

To show the PID configurations:

```
p4p# show pid nodes
p4p# show pid links
```

To view the PID Map visible to applications (via the `GetPIDMap` interface):

```
p4p# show pidmap
```

To view the PID returned for individual IP addresses to applications (via the `GetPID` interface):

```
p4p# show getpid
p4p# show getpid 128.36.0.1 10.1.254.254 10.254.0.1
```

To view the pDistances returned to applications (via the `GetpDistance` interface):

```
p4p# show pdistance
p4p# show pdistance newy-us-abilene.internet2.edu chic-us-abilene.internet2.edu
p4p# show pdistance chic-us-abilene.internet2.edu peering-to-nox.internet2.edu
```

3.5.2 Configuration Analysis

Other analysis can be performed on the running configurations. In particular, a tool (written in Bash and Perl) has been provided in the `p4p-portal-utils` package that analyzes the relationship between geographical distance and `pDistances` amongst network locations.

NOTE: To use this tool, the `Geo::IP` and `NetAddr::IP` Perl modules and `gnuplot` utility must be installed on your system.

The tool requires the GeoIP City Database. If you do not already have it, the latest database can be retrieved and uncompressed:

```
$ wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
$ gunzip GeoLiteCity.dat.gz
```

The tool can then be run as:

```
$ export GEOIP_DB_CITY=GeoLiteCity.dat
$ p4p_portal_config_analysis.sh p4p.Internet2.edu:6672 abilene
```

where the `GEOIP_DB_CITY` environment variable indicates the path to the uncompressed GeoIP City Database, and the Portal Server's address and administration port number are given. The final argument (`abilene` in this case) indicates an identification string used in the generated output files.

To see what the output files this tool generates:

```
$ p4p_portal_config_analysis.sh
```

You can also view how this analysis tool works by looking into the scripts. Using this as a template, you can easily generate your own customized tools for debugging and testing.

3.5.3 Topology Visualization

Sometimes it is useful to visualize the configured network topology. A tool named `p4p_portal_topoviz` is also provided in the `p4p-portal-utils` package that connects directly to a running Portal Server's administration port, extracts the topology, and outputs it in `graphviz` format ready to be converted to a graphical image.

NOTE: To display the generated graphs, the `graphviz` package must be installed on your system.

`p4p_portal_topoviz` can be used as follows to generate a PNG image of the installed network topology:

```
$ p4p_portal_topoviz --server=p4p.Internet2.edu --port=6672 > abilene_topo.dot
$ neato -Tpng < abilene_topo.dot > abilene_topo.png
```

3.5.4 Other Utilities

Note that the Portal Server administration API is available for use in your own programs. The API is written in C++, but the protocol itself is also quite simple so porting to other languages (*e.g.*, Perl or Python) should be a simple task. For usage of the C++ API, the source code for the Portal Shell itself is a good example.

If you are interested in directly using the administration API, please contact the Yale LANS team.

3.6 Converting Old Configurations

As P4P has evolved, the configuration syntax used by the Portal Shell has evolved as well. The following configuration file versions have been used:

- 1.0: Used in February 2008 field tests
- 2.0: Used in July 2008 field tests
- 2.1: Current version supported by Portal Shell

If you have configuration files older than the currently-supported version, they will need to be converted before loading. Perl scripts have been provided (as part of the `p4p-portal-shell` package) to convert configuration files to the next-highest version. Using this method, old configuration files can be incrementally converted to the most recent version (*e.g.*, convert version 1.0 to 2.0, then version 2.0 to 2.1).

NOTE: To use the conversion scripts, you will need to have the `NetAddr:IP` Perl module installed on your system.

3.6.1 Convert 1.0 to 2.0

Converting configuration files from version 1.0 to 2.0 requires two steps. First, the version 1.0 configuration file must be preprocessed:

```
$ portal-config-v1.0-preprocess.pl my-config-v1.0.conf > my-config-v1.0-preprocessed.conf
```

Next, the preprocessed configuration file can be converted to version 2.0:

```
$ portal-config-v1.0-to-v2.0.pl my-config-v1.0-preprocessed.conf > my-config-v2.0.conf
```

3.6.2 Convert 2.0 to 2.1

Converting configuration files from version 2.0 to version 2.1 is done with the following command:

```
$ portal-config-v2.0-to-v2.1.pl my-config-v2.0.conf > my-config-v2.1.conf
```

IMPORTANT: The P4P Framework now uses a more unified architecture to support interdomain configurations than previous versions. You may notice from the tutorial above that PIDs can be specified to group interdomain network locations. Due to this change, and that the current Portal server supports only IP Prefixes, interdomain configurations for previous versions are not converted.