Packet Handling Service (PHS)

|  |  |
| --- | --- |
| Version | 0.1 |
| Author | Tom Sumardi |
| TODO |  |

Table of Contents

Introduction 1

Abbreviations 1

Conventions/Requirements 1

Summary 1

Components 2

Architecture 2

Component descriptions: 3

# Introduction

This documentation intended to serve the purpose of giving detail level architecture description.

# Abbreviations

TBD

# Conventions/Requirements

* Performance bound
* Bidirectional communication interface from MS to PHS will be through AMQP
* Unidirectional communication interface from PPP to PHS will be through ZeroMQ
* PHS is a service that plugs into PPP component
* Packet outgoing interface will go through different interface than packet incoming interface
* Default PHS plugin will be ‘Packet Injection’

# Summary

PHS will be launched as daemon that can be null operation if nothing to be done on the packet in the case “passive” monitoring mode. In case of “active” monitoring mode, it will inject and route packets as specified. The mode of operations is fictitious to illustrate that the PHS can serve as packet router with storage backend consuming the packets instead of transparent caching solution. PHS by default set to inject packet. In this case, if interested packets are received from the PPP, packet injection will inform the video server to route the traffic to caching server while at the same time stop the current traffic flow. Example of PHS acting as packet injection plugin operation:

* TCP RST packet generation
* Redirection packet generation
* Raw packet vector injection

# Components

Config/Register

null

Pinj

storage

Other…

ZeroMQ

PPP

AMQP

plugin

Net Intf

Thread mgmt

Logging

Figure 1 (component stack)

* Background configuration/registration thread
* Packet Handling Plugin
* ZeroMQ Communication
* Logging

# Architecture

1. Background configuration/registration thread

Background thread is responsible for configuring and registering PHS component, which uses AMQP as communication channel to the MS.

Responsibilities:

* Registers plugin information
* Configure/load plugin (default: Pkt-Inj plugin)

This thread will be idle once configuration and registration have been successfully applied. Configuration can only be done once PHS component has been successfully registered.

state machine:

|----------------| |-----------|

down -> up -> registering -> registered -> loading -> loaded

| |

error

Below is the bi-directional communication between MS and PPP:

PHS -> MS registration:

* PPP uuid
* status (state-machine): registering, registered, error

MS -> PHS registration:

* status (state-machine): registering, registered, error

MS -> PHS configuration:

* packet handling mode: active or passive
* plugin:
  + Pkt-Inj plugin
  + Name
  + ZeroMQ metadata
  + Version
  + status (state-machine): loading, loaded, error

PHS -> MS configuration:

• status (state-machine): loading, loaded, error

1. Packet Handling Plugin

The packet handling plugin will be the last stage of pipeline, which continuously polling for messages coming from PPP. The message coming from PPP will contains header and body information of what to do next. The body can be the entire L4 payload. After processing the message, it is the responsibility of PHS to send the processed message to the final endpoint node. The endpoint can be caching server, storage backend, etc.

PPP -> PHS packet:

* message header
* message body

1. ZeroMQ communication

The uni-direction communication channel between PPP and PHS will be done through zeroMQ. The relationships between PPP with PHS service will be one-to-one relationships with one IPPS thread owns one zeroMQ communication channel. The channel relationships is build based on zeroMQ metadata being supplied from MS.

1. Logging

TBD

## Component descriptions:

??