Waggle communication document 0.1

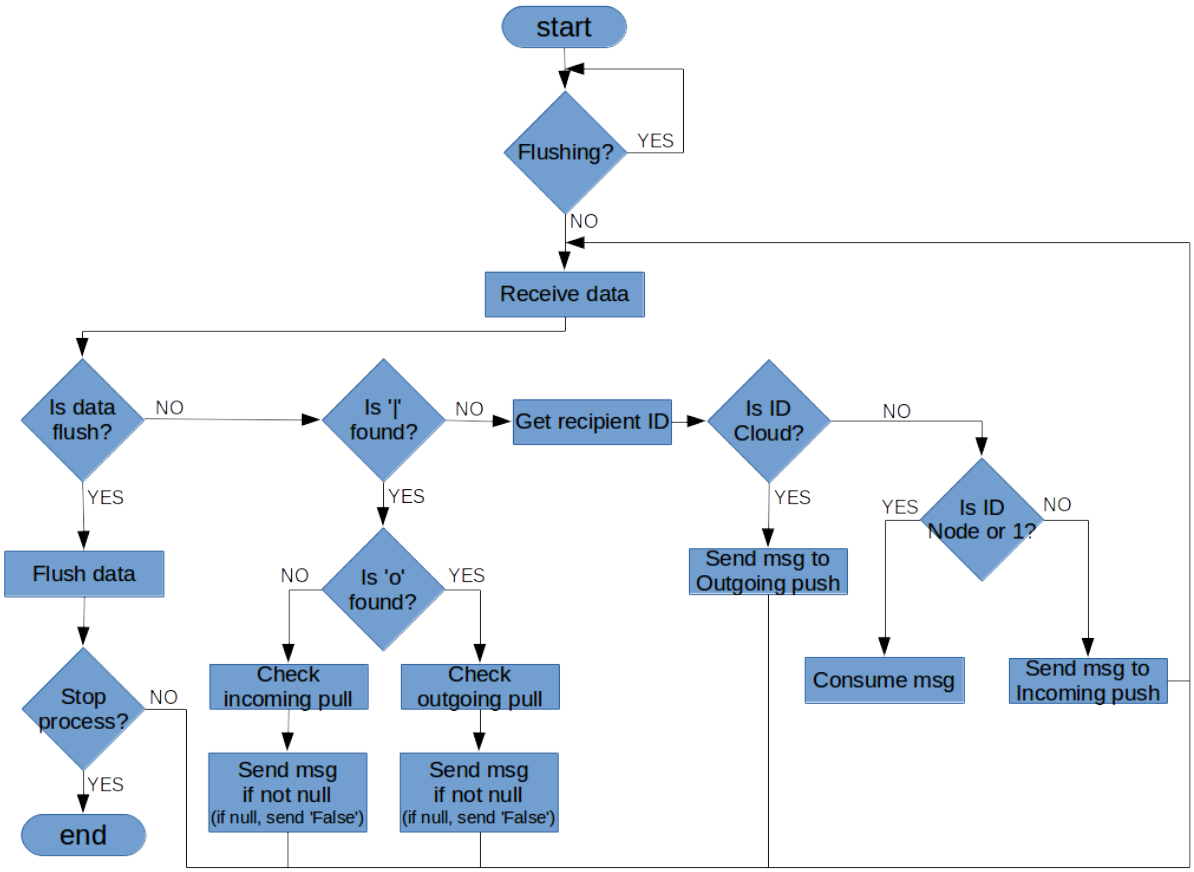
The Waggle Team

2016

This document describes all the design requirements of communication in the Waggle project, including component setup, data flow, message specifications. Messaging protocol version 0.4 is considered.

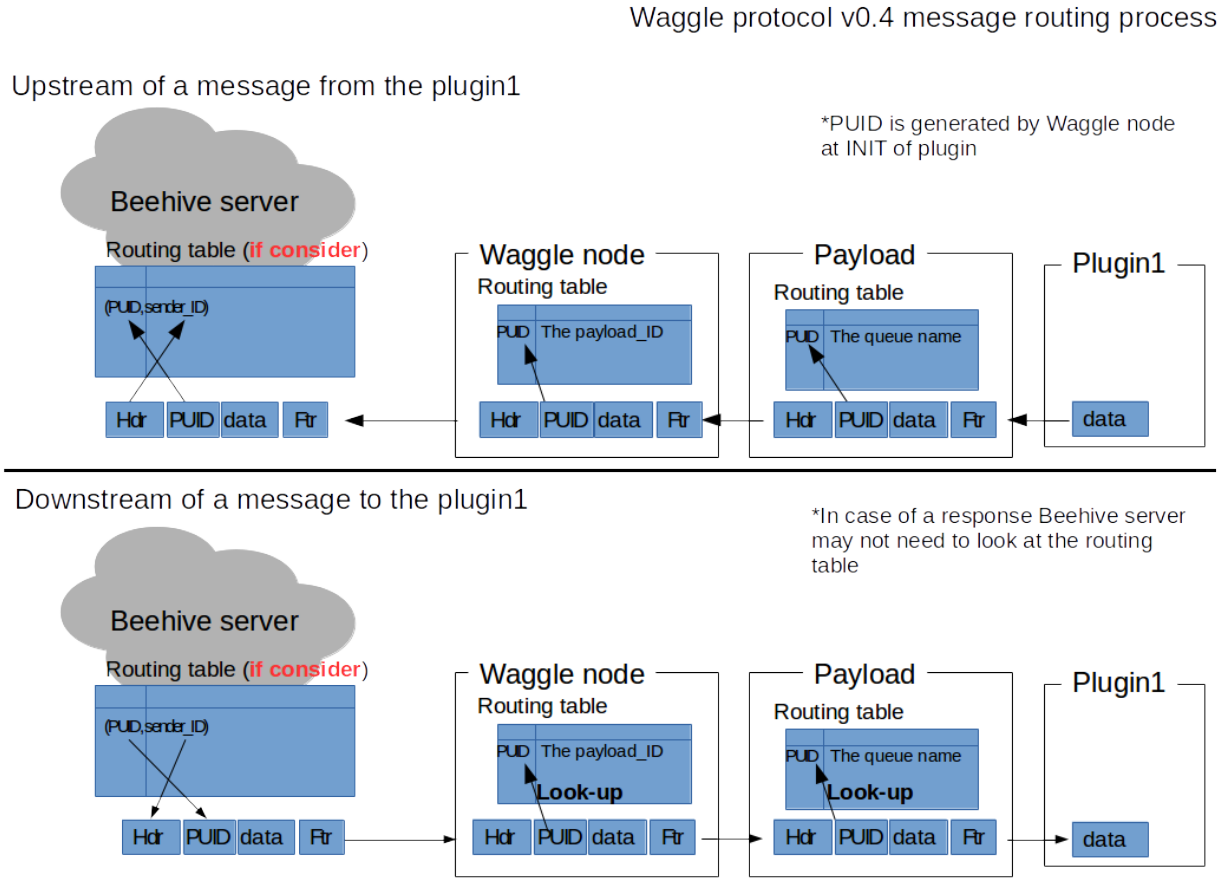
1 Nodecontroller

1.1 Data cache



TODO: data cache is consuming Waggle messages sent to the nodecontroller. We need to take the consumption part out from the data cache and put the part somewhere in internal communication.

1.2 Routing



Routing table is used to distribute Waggle messages to all connected components (i.e., payloads, plugins). All incoming Waggle messages that have $PUID$ in Ext_header set are consiered to be routed (Waggle messages that do not set $PUID$ are excluded for this route and will be consumed by the node). In the table, primary key is a unique number generated by nodecontroller and is mapped with routing information. An example of the routing table would look like,

ID	ROUTING	Meta
0	NODE_ID	"{'name':'example_plugin', 'ver':'0.3', ... }"
1	PAYLOAD1_ID	"{'name':'airsensor', 'ver':'0.4', ... }"
2	PAYLOAD2_ID	"{'name':'example_plugin', 'ver':'0.3', ... }"

In this example ID 0 seems that it is a plugin attached to the nodecontroller. When a Waggle message with the ID 0 comes to the nodecontroller it will go to the 'system_receive' plugin running on the same node. ID 1 is also a plugin attached to the payload1 so the nodecontroller needs to send the message to the payload1 first and the message will be routed again in the payload1.

1.2.1 Registration for routing

The registration request (Msg_Mj_Type 'r' and Msg_Mj_Type 'r') will be sent to the parent node and will also be consumed at the node that has routing table. If the requestor is already in the

routing table, a response message with the corresponding ID will be sent. When a registered plugin lost its ID or rebooted with a different instance a new ID will be assigned to the plugin. **TODO: we will need to clean up the routing table for some IDs that seem never used**

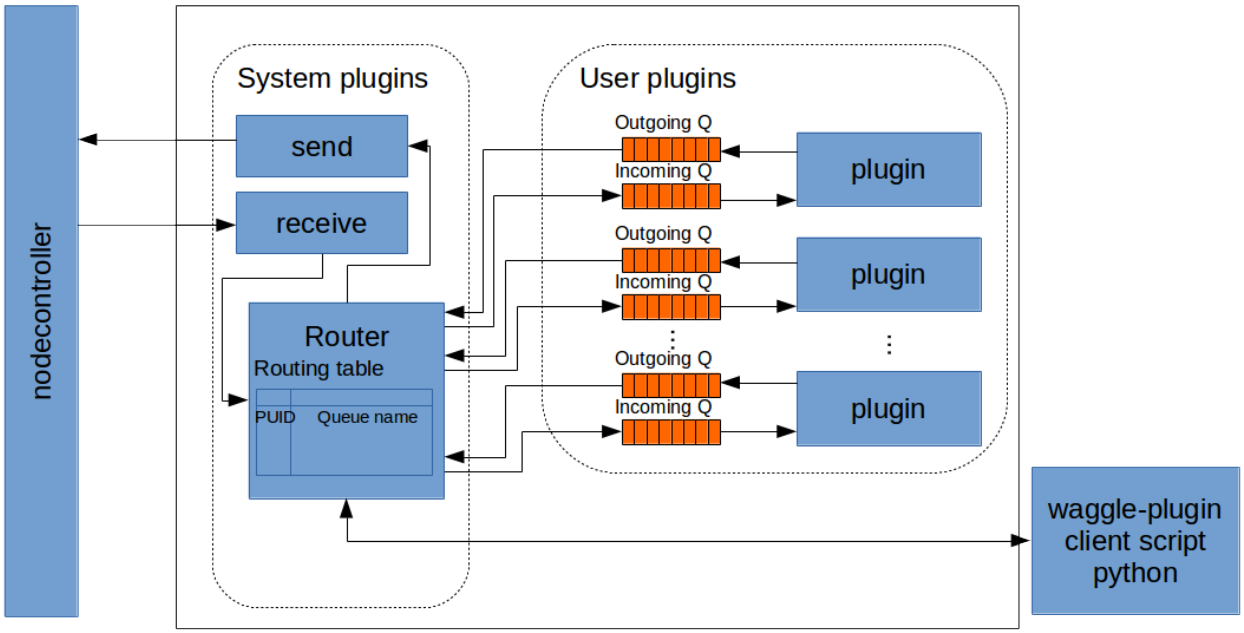
1.2.2 Flushing the routing table to the server

When the nodecontroller boots up it may wait until it gets a number of registration requests from attached plugins or payload nodes along with their meta data (e.g., name, version, instance, etc.). At a certain point of time, the nodecontroller needs to flush the routing table with all the meta data to the Beehive server so that the server could respond requests and sometimes send a message to the plugin registered in the routing table.

1.2.3 Communication with Beehive server

When connection is available to the server, nodecontroller pushes Waggle messages piled in the Data_cache. All the outgoing Waggle messages will get the same S_Uniq_ID set by the ID of the nodecontroller so that the server only takes care of nodes, not plugins nor payloads.

2 Payload



2.1 Plugins

2.1.1 System plugins

- System_send: packetizes JSON data into Waggle message. Waggle messages are sent to attached parent node via TCP/IP.
- System_receive: receives Waggle messages, check CRC, de-serialize the message using JSON, and send it to system_router plugin.
- System_router: routes JSON data. This plugin holds routing table to distribute messages properly. For the details of the routing table, refer to 1.2.

2.1.2 User plugins

The data types used in JSON are listed below.

KEYWORD	MEANING	NOTE
mj_op	Major operation	One char
mi_op	Minor operation	One char
	Plugin/Payload unique identifier	Sender's PUID
	Plugin/Payload unique identifier	Recipient's PUID
meta	Meta data	Dictionary-type information (plugin name, version, instance, etc.)
data	Data	Dictionary-type data
rec	Recipient	Specify recipient (Default is Beehive)
snd_s	Sender's sequence number	Sequence number of the sender (optional)
resp_s	Responder's sequence number	Sequence number of the responder (optional)
snd_ss	Sender's session number	Session number of the sender (optional)
resp_ss	Responder's session number	Session number of the responder (optional)
error	Error message	Error message when occur (optional)

When JSON contains 'error' plugins need to react to it appropriately.

2.1.3 Communication

All communications between system_router and plugins are through JavaScript Object Notation (JSON).

2.1.4 Registration

A plugin should maintain a plugin unique identifier (PUID) generated by the nodecontroller the plugin attached to. If the plugin cannot find the PUID (either the first time or missing), the plugin can request a PUID by sending a registration request to the nodecontroller. As an example would be,

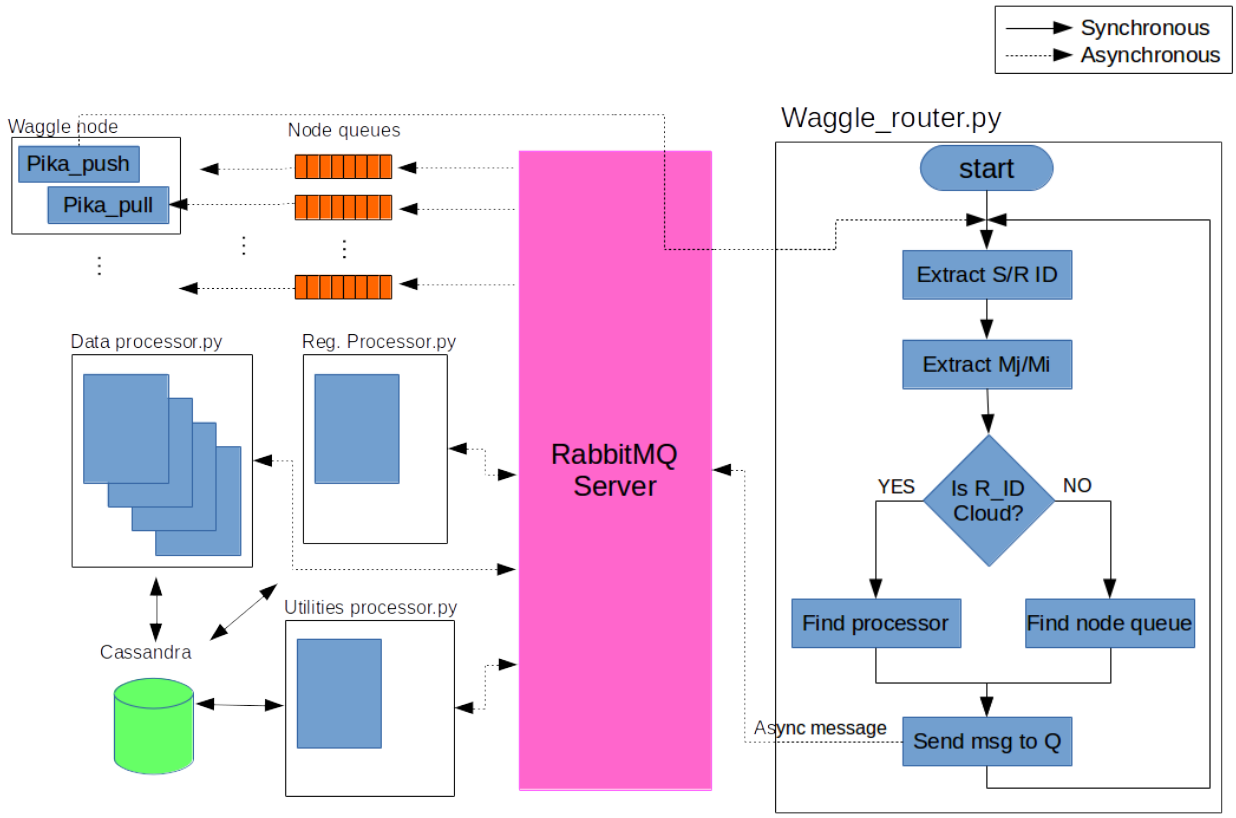
{“mj_op”:“r”, “mi_op”:“r”, “puid”:uuid.uuid4(), “meta”:“... plugin name, version, instance...”}

For “tmpID” use randomly generated ID using uuid package in Python. This “tmpID” will only be used until the plugin gets PUID from the nodecontroller. The PUID should be properly maintained by the plugin (e.g., in a file) and should not be altered.

2.1.5 Listener

If a plugin needs to get data from outside (e.g., the Beehive server or nodes), the plugin should assign an asynchronous listener and make the listener attached to the incoming queue. For the format of incoming message, refer to 2.1.2.

3 Beehive server



3.1 Store data

After unpacking Waggle message, the actual payload message is JSON type. Server process should handle JSON data to pull values and store them in the database. JSON data consist of the following format. for now the format is based on ‘sensor_table’ scheme.

node_id, date, plugin_id, plugin_version, plugin_instance, timestamp, sensor, sensor_meta, data

TODO: Do we need to look up the routing table in the server everytime when a Waggle message comes to the server?