

Introduction

Vert.x is a toolkit for the Java Virtual Machine enabling the implementation of reactive, highly concurrent, polyglot applications. Thanks to its eventbus, any application can interact with Vert.x applications.

The TCP EventBus bridge is a bridge built on top of TCP and it provides facility to interact with any other application which is written in different programming languages. The TCP EventBus bridge can be called as a **server interface** also.

Download and install

Download Vertx from [here](#)

This provides a quick guide to install vert.x [here](#)

Implementation

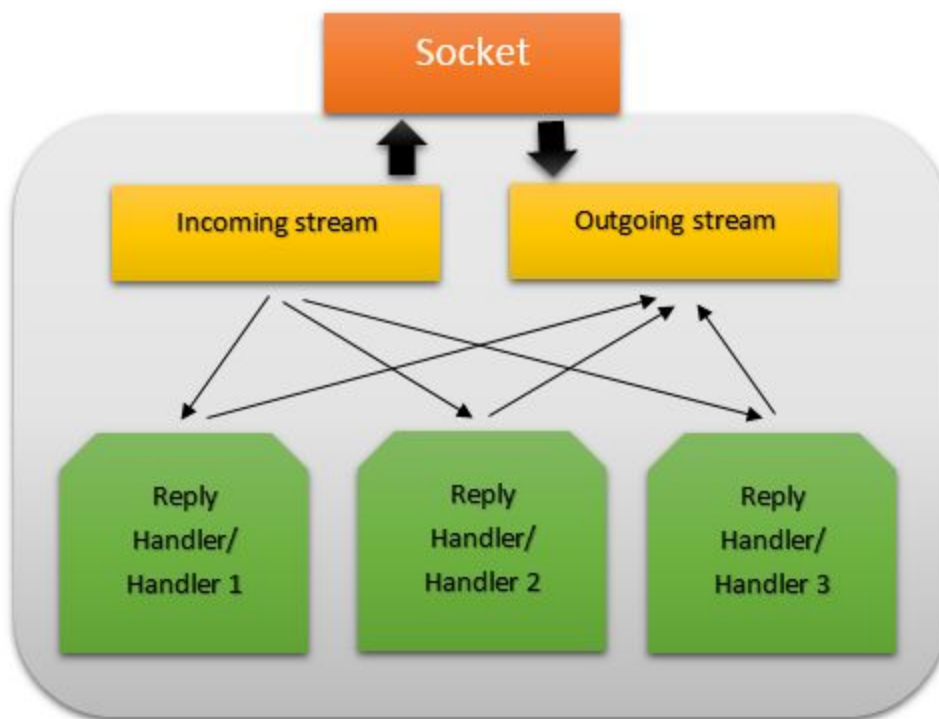


Figure 1- Basic structure of TCP eventbus Bridge

Addressing

Messages are sent on the event bus to an address. It can be any string.

Handlers

A handler is one that receives messages from the bus. A handler can be registered at an address. Many different handlers from the same or different verticles can be registered at the same address. A single handler can be registered by the verticle at many different addresses.

- When messages received, those messages will be **immediately** handled to the registered handlers. Further details will be discussed with the example.

Protocol

Frame is a LV JSON message.

<Frame length: 4 bytes (big endian encoding) >{JSON Object : utf-8}

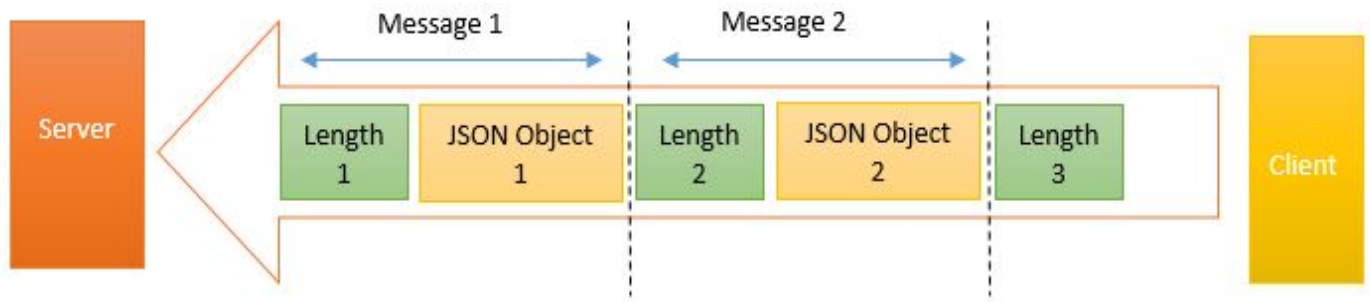


Figure 2- Message stream from a client to the TCP eventbus bridge

Structure of a frame

```
<Length: 4 bytes(Big Endian)><{  
  type: String,  
  address: String,  
  (reply Address : String)(Optional),  
  headers: JsonObject,  
  body: JsonObject  
}: JsonObject>
```

Type

These are the valid types for incoming messages

- 1) "send" - The message will be routed to just one of the handlers registered at that address. If there is more than one handler registered at the address, one will be chosen using a non-strict round-robin algorithm.
- 2) "publish" - The message will be routed to the all registered handlers.
- 3) "register" - register handler for the address
- 4) "unregister" - unregister handler for the address

These are the types that send to the clients

- 1) "message" - reply messages or acknowledgement messages
- 2) "err" - error messages