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**.

Download and install

Download Vertx from here

This provides a quick guide to install vert.x here

Implementation

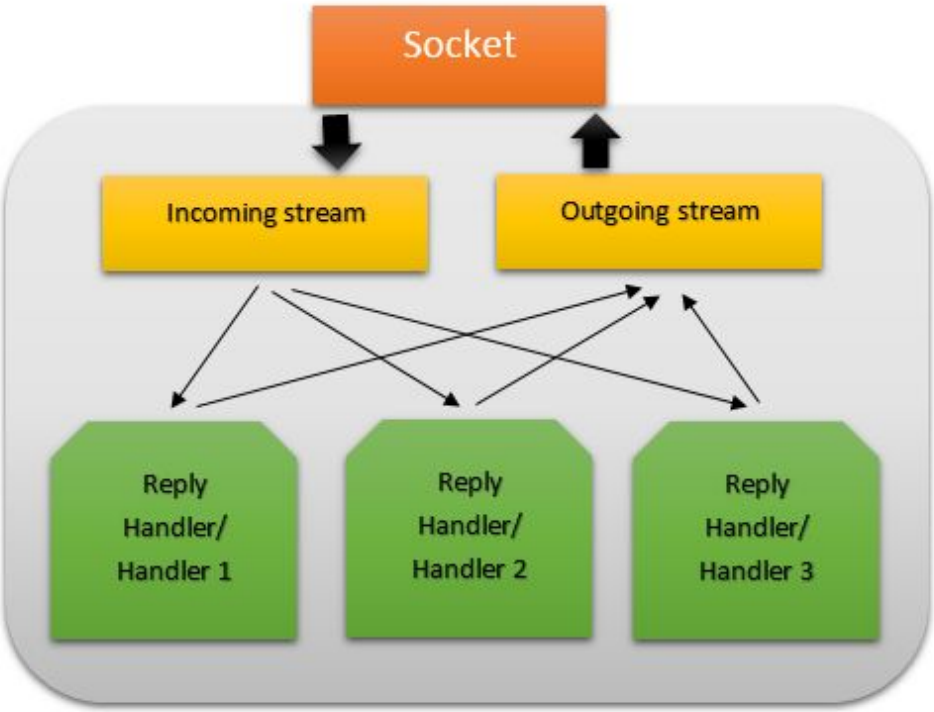


Figure 1- Basic structure of TCP eventbus Bridge

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 later.

Protocol

Frame is a LV JSON message.

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

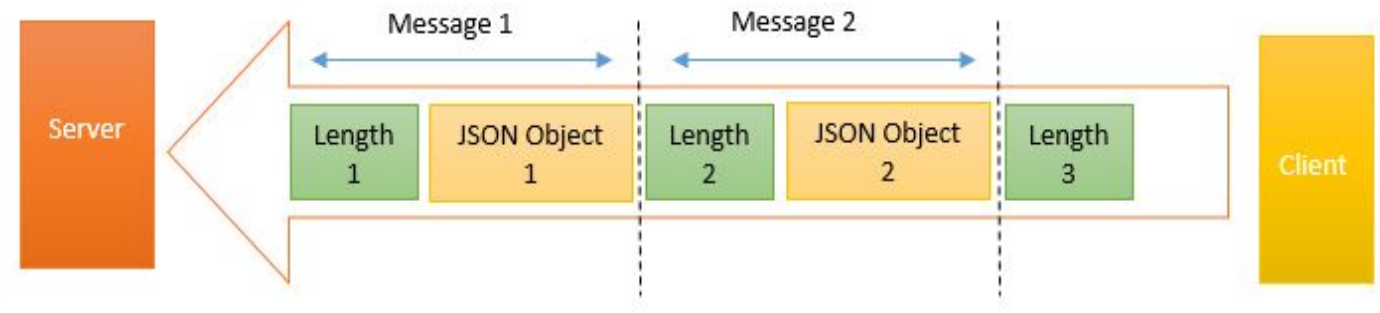


Figure 2- Message stream from a client to the TCP eventbus bridge

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

Example

This code can be download from GitHub. here

```
public class Server extends AbstractVerticle{

  public void start(Future<Void> fut){
    TcpEventBusBridge bridge = TcpEventBusBridge.create(
    vertx,
    new BridgeOptions()
        .addInboundPermitted(new PermittedOptions().setAddress("welcome"))
        .addOutboundPermitted(new PermittedOptions().setAddress("welcome")));

    bridge.listen(7000, res -> {
        if (res.succeeded()) {
        System.out.println("Started");
        } else {
        System.out.println("failed");
        }
    });
    EventBus eb = vertx.eventBus();

    MessageConsumer< JsonObject > consumer=eb.consumer("welcome", message -> {
        System.out.println("Message body: " + message.body());
        System.out.println("Headers:"+message.headers());
        String jsonString = "{\"msg\":\"welcome\"}";
        JsonObject object = new JsonObject(jsonString);
        message.reply(object);
    });
  }
}
```

BridgeOption - addInboundPermitted and addOutboundPermitted

Incoming and outgoing messages can be limited by the address of the messages.

listen(port,handler)

This method is used to start the bridge. More information can be found here.

MessageConsumer< JsonObject > consumer=eb.consumer(Address,Handler)

here address is a string.

Whenever a message comes to the address it will be routed to the handler. This process is called **handler registration.**

message.body()

This returns the body of the message as a json object.

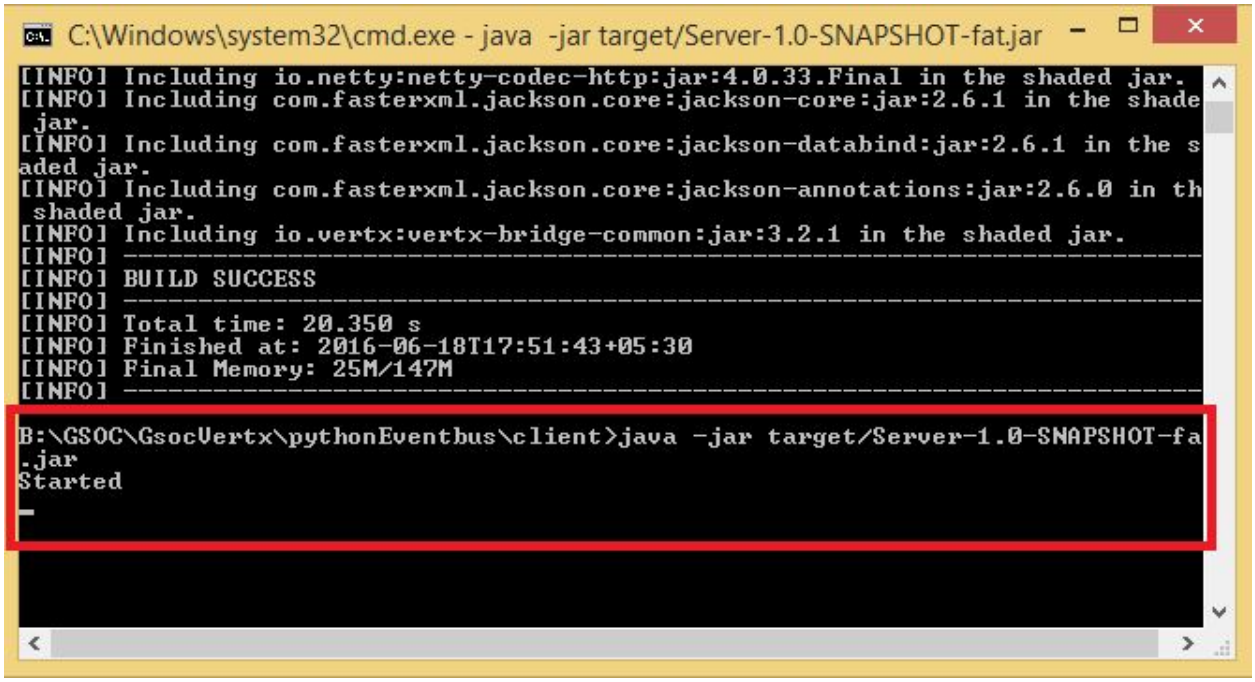
message.reply(jsonObject)

If the received message has a reply address then handler will reply a frame which contains 'jsonObject' in the body, to reply address.

Let's run

download the example from Github. here

Then package it using maven and run. You can find more information about packaging and running of vertx applications here .


C:\Windows\system32\cmd.exe - java -jar target/Server-1.0-SNAPSHOT-fat.jar
[INFO] Including io.netty:netty-codec-http:jar:4.0.33.Final in the shaded jar.
[INFO] Including com.fasterxml.jackson.core:jackson-core:jar:2.6.1 in the shade
jar.
[INFO] Including com.fasterxml.jackson.core:jackson-databind:jar:2.6.1 in the s
aded jar.
[INFO] Including com.fasterxml.jackson.core:jackson-annotations:jar:2.6.0 in th
shaded jar.
[INFO] Including io.vertx:vertx-bridge-common:jar:3.2.1 in the shaded jar.
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 20.350 s
[INFO] Finished at: 2016-06-18T17:51:43+05:30
[INFO] Final Memory: 25M/147M
[INFO] --
B:\GSOC\GsocVertx\pythonEventbus\client>java -jar target/Server-1.0-SNAPSHOT-fa
.jar
Started