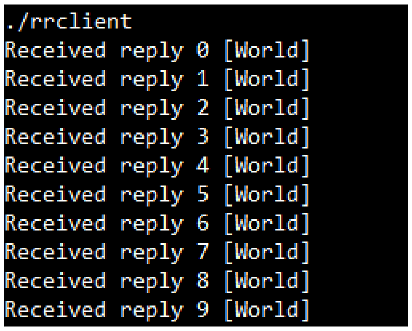
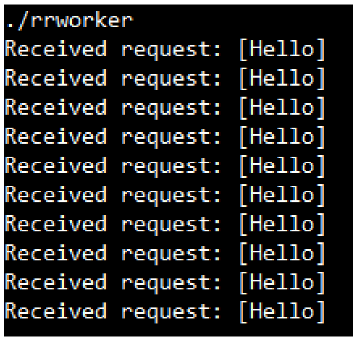
Below you can see snippets with the console output of the components.

rrclient output:



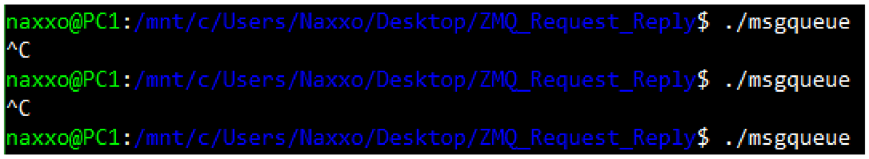
```
./rrclient
Received reply 0 [World]
Received reply 1 [World]
Received reply 2 [World]
Received reply 3 [World]
Received reply 4 [World]
Received reply 5 [World]
Received reply 6 [World]
Received reply 7 [World]
Received reply 8 [World]
Received reply 9 [World]
```

rrworker output:



```
./rrworker
Received request: [Hello]
Received request: [Hello]
Received request: [Hello]
Received request: [Hello]
Received request: [Hello]
Received request: [Hello]
Received request: [Hello]
Received request: [Hello]
Received request: [Hello]
Received request: [Hello]
```

msgqueue output:



```
naxxo@PC1:/mnt/c/Users/Naxxo/Desktop/ZMQ_Request_Reply$ ./msgqueue
^C
naxxo@PC1:/mnt/c/Users/Naxxo/Desktop/ZMQ_Request_Reply$ ./msgqueue
^C
naxxo@PC1:/mnt/c/Users/Naxxo/Desktop/ZMQ_Request_Reply$ ./msgqueue
```

Upon stopping the msgqueue, the exchange of requests/responses is paused. Upon restarting the queue, the exchange continues from where it was put to a halt. We can see that the communication between the client and the worker seems uninterrupted even after 2 restarts of msgqueue.

We can see how this is achieved by checking the API documentation:

"When the frontend is a ZMQ_ROUTER socket, and the backend is a ZMQ_DEALER socket, the proxy shall act as a shared queue that collects requests from a set of clients and distributes these fairly among a set of services. Requests shall be fair queued from frontend connections and distributed evenly across backend connections. Replies shall automatically return to the client that made the original request."