SOFE 4790U

Distributed Systems

CRN 43525

Lab 3

October 24, 2024

Jeremy Mark Tubongbanua	100849092

Table of Contents

Task 1

Task #1: Establish the connection (20 marks)
Use the provided *ZServer.java & ZClient.java* program (available on Canvas). Modify the code in both client and server so that when the client connects to the server, both print a successful message upon start.

See sample code here:

- **https://zeromq.org/languages/java/**
- **https://zeromq.org/get-started/?language=java&library=jeromq#**

In Task1, the client will connect to the running server. When the server receives a connection, the server will receive a message, then send a message back. The client will also be expecting a message back after it has initially sent a message.

This is a simple ping program.

Server Output

```
➜  sofe4790u-lab3 javac -cp ./jeromq.jar task1/ZServer.java && java -cp .
:./jeromq.jar task1.ZServer
Creating ZMQ.Socket...
Binding to tcp://*:5555...
Received request: "Hello"
Sent reply: "Hello from server"
```

Client Output

```
➜  sofe4790u-lab3 javac -cp ./jeromq.jar task1/ZClient.java && java -cp .
:./jeromq.jar task1.ZClient
Connecting to hello world server...
Sending Hello...
Received reply: "Hello from server"
```

ZClient.java

```java
package task1;
import org.zeromq.SocketType;
import org.zeromq.ZContext;
import org.zeromq.ZMQ;

public class ZClient {

    public static void main(String[] args) {

        try (ZContext context = new ZContext()) {
            System.out.println("Connecting to hello world server...");

            ZMQ.Socket socket = context.createSocket(SocketType.REQ);
            socket.connect("tcp://localhost:5555");

            String message = "Hello";
            System.out.println("Sending Hello...");
            socket.send(message);

            String reply = socket.recvStr();
            System.out.println("Received reply: \"" + reply + "\"");
```

```
            socket.close();
            context.close();
        }
    }
}
```

ZServer.java

```
package task1;

import org.zeromq.SocketType;
import org.zeromq.ZContext;
import org.zeromq.ZMQ;

public class ZServer {

    public static void main(String[] args) throws Exception {
        try (ZContext context = new ZContext()) {
            System.out.println("Creating ZMQ.Socket...");
            ZMQ.Socket socket = context.createSocket(SocketType.REP);
            System.out.println("Binding to tcp://*:5555...");
            socket.bind("tcp://*:5555");

            int i = 0;

            while (i < 10) {
                String request = socket.recvStr();
                System.out.println("Received request: \"" + request +
"\"");

                Thread.sleep(100);
                String reply = "Hello from server";
                socket.send(reply);
                System.out.println("Sent reply: \"" + reply + "\"");
                i++;
            }

            socket.close();
            context.close();
        }
    }
}
```

Task 2

Task #2: Prime numbers (20 marks)

In this task, you will get a better understanding of how a client can send a message and get reply from the server. Remember ZeroMQ doesn't know anything about the data you send except its size in bytes. That means you are responsible for formatting it safely so that applications can read it back. Modify the code from task#1 or the *hwserver.java & hwclient.java* (available on Canvas) so that client can send an integer number to the server. Server will receive the number and send all the prime numbers up to the provided number. Client receives the message and prints it. For example: If client sends "10", it will receive "2,3,5,7" from the server.

In Task 2, I wrote isPrime and getPrimesBefore static java methods. isPrime returns true or false given an integer input if the number is prime. getPrimesBefore method returns a List<Integer> which returns all the prime numbers given an input integer argument. For example, inputting 10 into this method will return {2, 3, 5, 7}. The way this method works is through a for loop starting from 2 to the integer argument (non-inclusive) and uses the previous isPrime method that I wrote as a helper function.

In the client, we receive input using Scanner and System.in. We read what integer they inputted, then simply forward that to the server.

Then we simply build a message string to give back to the user to view, which is a comma-separated string with the prime numbers before the initial integer input..

Server Output

```
➜  sofe4790u-lab3 javac -cp ./jeromq.jar task2/hwserver.java && java -cp
.:./jeromq.jar task2.hwserver
Server started...
Received : [10]
Sending : [2, 3, 5, 7]...
Sent.
Received : [190]
Sending : [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59
, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137,
 139, 149, 151, 157, 163, 167, 173, 179, 181]...
Sent.
```

Client Input

```
→  sofe4790u-lab3 javac -cp ./jeromq.jar task2/hwclient.java && java -cp
.:./jeromq.jar task2.hwclient
Connecting to hello world server
Connected to server...
Enter a number:
10
Received : [2, 3, 5, 7]
Enter a number:
190
Received : [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 5
9, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137
, 139, 149, 151, 157, 163, 167, 173, 179, 181]
Enter a number:
```

hwserver.java

```java
package task2;

import java.util.ArrayList;
import java.util.List;
import org.zeromq.SocketType;
import org.zeromq.ZMQ;
import org.zeromq.ZContext;

public class hwserver {
      public static void main(String[] args) throws Exception {
            try (ZContext context = new ZContext()) {
                  // Socket to talk to clients
                  ZMQ.Socket socket = context.createSocket(SocketType.REP);
                  socket.bind("tcp://*:5554");
                  System.out.println("Server started...");

                  while (!Thread.currentThread().isInterrupted()) {
                        byte[] reply = socket.recv(0);
                        String replyStr = new String(reply, ZMQ.CHARSET);
                        System.out.println("Received " + ": [" + replyStr +
"]");

                        Thread.sleep(1000); // Do some 'work'

                        int input = Integer.parseInt(replyStr);

                        List<Integer> primes = getPrimesBefore(input);
```

```
                        String message = "";
                        for (int i = 0; i < primes.size(); i++) {
                              if(i > 0) {
                                    message += ", ";
                              }
                              message += primes.get(i);
                        }

                        System.out.println("Sending " + ": [" + message +
"]...");
                        socket.send(message.getBytes(ZMQ.CHARSET), 0);
                        System.out.println("Sent.");
                  }
            }
      }

      private static boolean isPrime(int input0) {
            if (input0 < 2) {
                  return false;
            }
            for (int i = 2; i < input0; i++) {
                  if (input0 % i == 0) {
                        return false;
                  }
            }
            return true;
      }

      private static List<Integer> getPrimesBefore(int input) {
            List<Integer> primes = new ArrayList<Integer>();
            for (int i = 2; i < input; i++) {
                  if (isPrime(i)) {
                        primes.add(i);
                  }
            }
            return primes;
      }
}
```

hwclient.java

```
package task2;
import java.util.Scanner;
```

```java
import org.zeromq.SocketType;
import org.zeromq.ZMQ;
import org.zeromq.ZContext;

public class hwclient {

	public static void main(String[] args) {
		try (ZContext context = new ZContext()) {
			// Socket to talk to server
			System.out.println("Connecting to hello world server");

			ZMQ.Socket socket = context.createSocket(SocketType.REQ);
			socket.connect("tcp://localhost:5554");
			System.out.println("Connected to server...");

			Scanner scanner = new Scanner(System.in);
			while (true) {
				System.out.println("Enter a number: ");
				String input = scanner.nextLine();
				socket.send(input.getBytes(ZMQ.CHARSET), 0);

				byte[] reply = socket.recv(0);
				String replyStr = new String(reply, ZMQ.CHARSET);
				System.out.println("Received " + ": [" + replyStr +
"]");
			}
		}
	}
}
```

Task 3

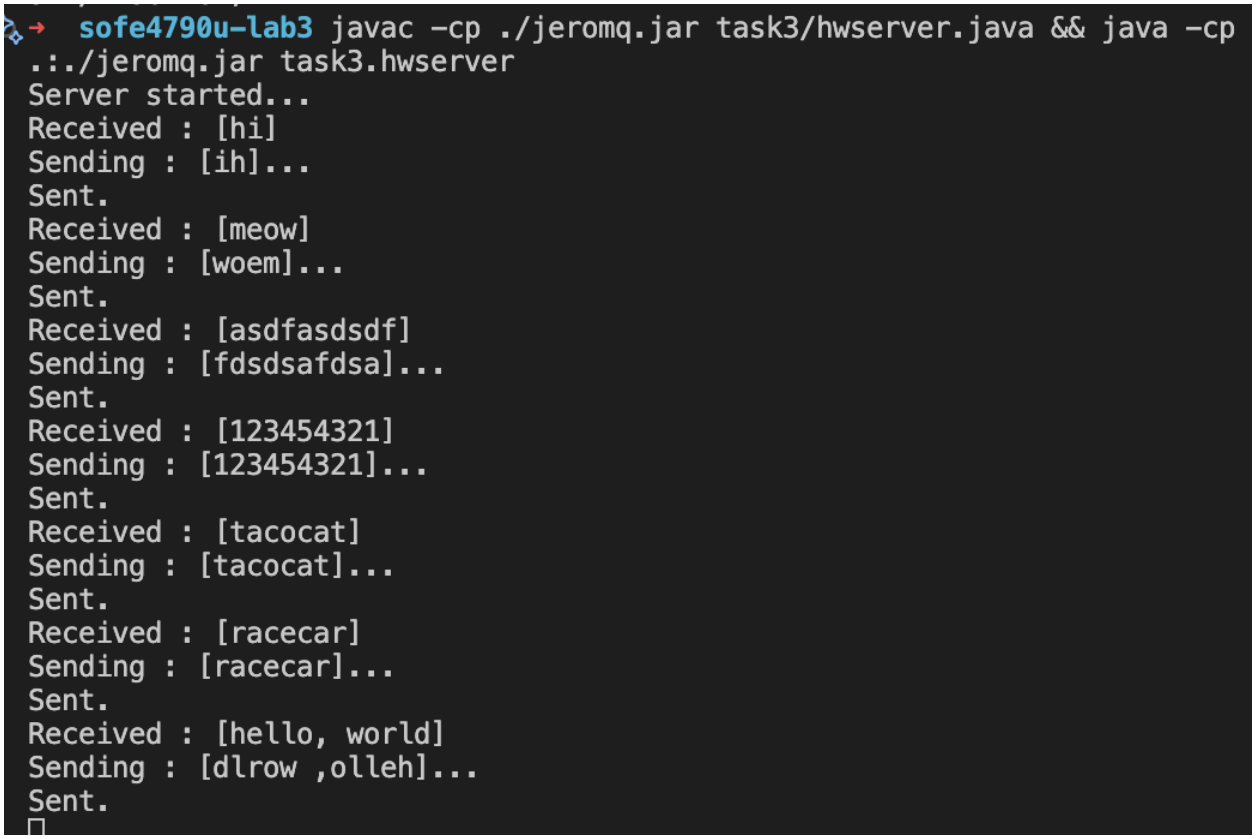
Task #3: Repeated inputs from client (20 marks)

In this task, the client will send some messages repeatedly to the server and server will reply accordingly. Modify the code so that client can take a string as input from the keyboard and send

it to server. Server will print the string, reverse it and send back to client. The client will receive and print the message received (reversed string), and send another message to the server (another string to be reversed) and the process is repeated until the client sends the “close” message to the server which will stop the connection between server and client will terminate.

In task 3, we receive an input string from the client via a ZMQ.Socket. If they send a string like “meow”, then the server will send a message reply back to the client with the reversed string like “woem” I wrote a reverseString() static java method to assist with this.

Server Output



Client Output

```
➜  sofe4790u-lab3 javac -cp ./jeromq.jar task3/hwclient.java && java -cp
.:./jeromq.jar task3.hwclient
Connecting to hello world server
Connected to server...
Enter text:
hi
meow
Received : [ih]
Enter text:
Received : [woem]
Enter text:
asdfasdsdf
Received : [fdsdsafdsa]
Enter text:
123454321
Received : [123454321]
Enter text:
tacocat
Received : [tacocat]
Enter text:
racecar
Received : [racecar]
Enter text:
hello, world
Received : [dlrow ,olleh]
Enter text:
```

hwserver.java

```java
package task3;

import java.util.ArrayList;
import java.util.List;
import org.zeromq.SocketType;
import org.zeromq.ZMQ;
import org.zeromq.ZContext;

public class hwserver {
      public static void main(String[] args) throws Exception {
            try (ZContext context = new ZContext()) {
                  // Socket to talk to clients
                  ZMQ.Socket socket = context.createSocket(SocketType.REP);
```

```
                socket.bind("tcp://*:5554");
                System.out.println("Server started...");

                while (!Thread.currentThread().isInterrupted()) {
                      byte[] reply = socket.recv(0);
                      String replyStr = new String(reply, ZMQ.CHARSET);
                      System.out.println("Received " + ": [" + replyStr +
"]");

                      Thread.sleep(1000); // Do some 'work'

                      String message = reverse(replyStr);

                      System.out.println("Sending " + ": [" + message +
"]...");
                      socket.send(message.getBytes(ZMQ.CHARSET), 0);
                      System.out.println("Sent.");
                }
          }
     }

     private static String reverse(String input) {
          return new StringBuilder(input).reverse().toString();
     }
}
```

hwclient.java

```
package task3;
import java.util.Scanner;
import org.zeromq.SocketType;
import org.zeromq.ZMQ;
import org.zeromq.ZContext;

public class hwclient {

     public static void main(String[] args) {
          try (ZContext context = new ZContext()) {
                // Socket to talk to server
                System.out.println("Connecting to hello world server");
```

```
                ZMQ.Socket socket = context.createSocket(SocketType.REQ);
                socket.connect("tcp://localhost:5554");
                System.out.println("Connected to server...");

                Scanner scanner = new Scanner(System.in);
                while (true) {
                        System.out.println("Enter text: ");
                        String input = scanner.nextLine();
                        socket.send(input.getBytes(ZMQ.CHARSET), 0);

                        byte[] reply = socket.recv(0);
                        String replyStr = new String(reply, ZMQ.CHARSET);
                        System.out.println("Received " + ": [" + replyStr +
"]");
                }
        }
    }
}
```

Task 4

Task#4: Publish-subscribe application (40 marks)

Review, compile and run the sample code: *wuserver* and *wuclient* (available on Canvas). Your task is to adapt this code to accomplish the following task: develop a publish-subscribe application in ZeroMQ for population updates according to postal codes. A set of clients will subscribe to the server with different postal codes taken as input from the keyboard. The server will continuously broadcast updates of population of different areas. When the client listens to the stream of updates of the postal code it has specified, it will print the information received.

How does the server determine the postal code and the population of that area?

Tip: Generate at least 10 random postal codes and population numbers on the server (for simplicity declare the postal code as a 4-digit integer number. For example: 1001, 4349, etc.). Let the server prints the postal codes generated, and use them to subscribe at least 3 clients.

In this task, I commenced 3 clients (Client 1, 2, and 3, each with zip codes 1111, 1301, and 1000) that subscribe to the server. The server publishes random populations of zipcodes 1000-1999 with a random population ranging from 0-999.

Then each client will subscribe to the server, listen for 100 iterations of their particular zipcodes that are published, then calculate the average, then output that as the population.

Server Output

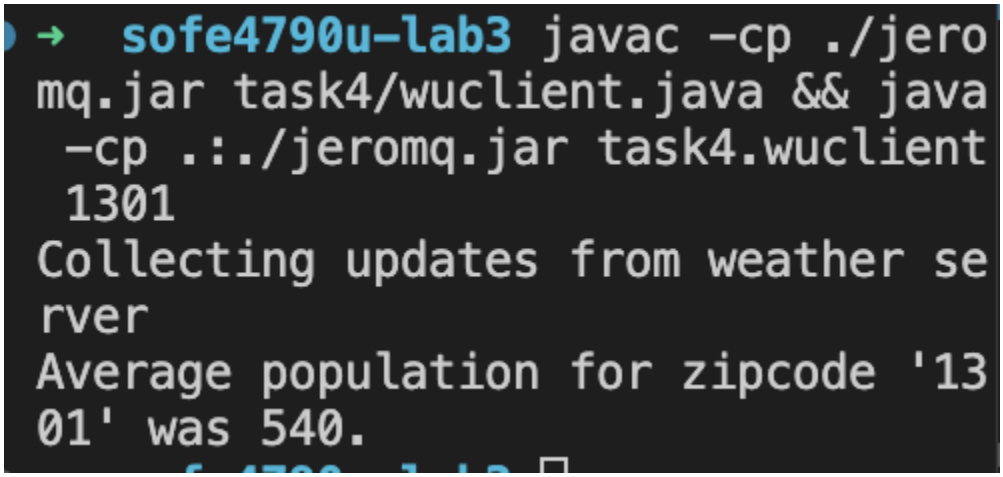
```
→  sofe4790u-lab3 javac -cp ./jeromq.jar task4/wuserver.java && java -cp
  .:./jeromq.jar task4.wuserver

1867 0075
1802 0390
1432 0516
1183 0934
1590 0517
1266 0454
1785 0676
1173 0086
1220 0243
1161 0164
1443 0306
1936 0436
```

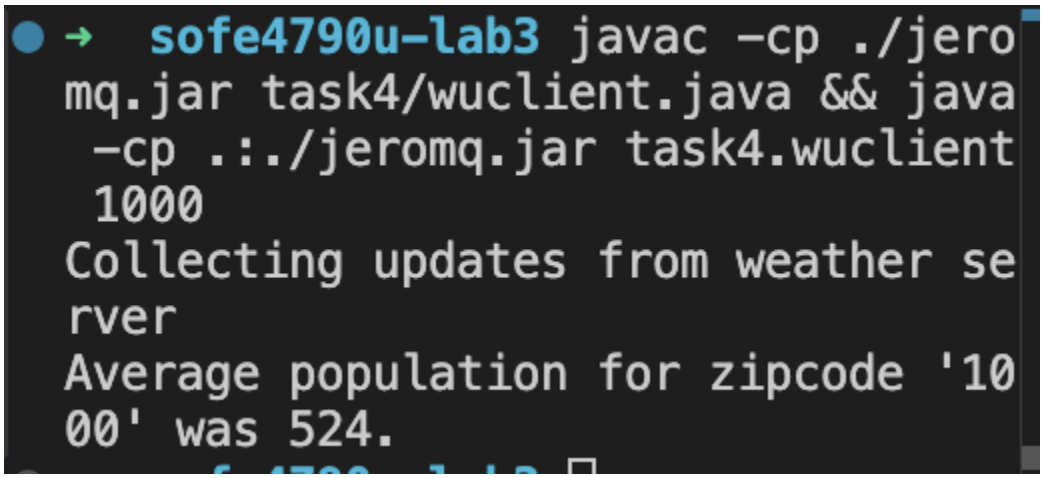
Client 1 Output (Zipcode 1111)

```
→  sofe4790u-lab3 javac -cp ./jero
mq.jar task4/wuclient.java && java
 -cp .:./jeromq.jar task4.wuclient
 1
111
Collecting updates from weather se
rver
Average population for zipcode '11
11' was 541.
```

Client 2 Output (Zipcode 1301)



Client 3 Output (Zipcode 1000)



wuserver.java

```
package task4;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;

import org.zeromq.SocketType;
import org.zeromq.ZMQ;
import org.zeromq.ZContext;
```

```
public class wuserver {

	public static void main(String[] args) throws Exception {
		try (ZContext context = new ZContext()) {
			ZMQ.Socket publisher =
context.createSocket(SocketType.PUB);
			publisher.bind("tcp://*:5556");
			publisher.bind("ipc://weather");

			Random srandom = new Random(System.currentTimeMillis());
			while (!Thread.currentThread().isInterrupted()) {
				int zipcode, population;
				zipcode = 1000 + srandom.nextInt(1000); //
1000-1999 zipcodes
				population = srandom.nextInt(1000); // 0-99 people

				String update = String.format("%04d %04d", zipcode,
population);
				System.out.println(update);
				publisher.send(update, 0);
			}
		}
	}
}
```

wuclient.java

```
package task4;
import java.util.StringTokenizer;

import org.zeromq.SocketType;
import org.zeromq.ZMQ;
import org.zeromq.ZContext;

public class wuclient {
	@SuppressWarnings("unused")
	public static void main(String[] args) {
		try (ZContext context = new ZContext()) {
			System.out.println("Collecting updates from weather
```

```
server");
                ZMQ.Socket subscriber =
context.createSocket(SocketType.SUB);
                subscriber.connect("tcp://localhost:5556");

                String filter = (args.length > 0) ? args[0] : "1001 ";
                subscriber.subscribe(filter.getBytes(ZMQ.CHARSET));

                int update_nbr;
                long total_population = 0;
                for (update_nbr = 0; update_nbr < 100; update_nbr++) {
                      String string = subscriber.recvStr(0).trim();

                      StringTokenizer sscanf = new
StringTokenizer(string, " ");
                      int zipcode = Integer.valueOf(sscanf.nextToken());
                      int population =
Integer.valueOf(sscanf.nextToken());

                      total_population += population;
                }

                System.out.println(String.format("Average population for
zipcode '%s' was %d.", filter,
                            (int) (total_population / update_nbr)));
            }
      }
}
```