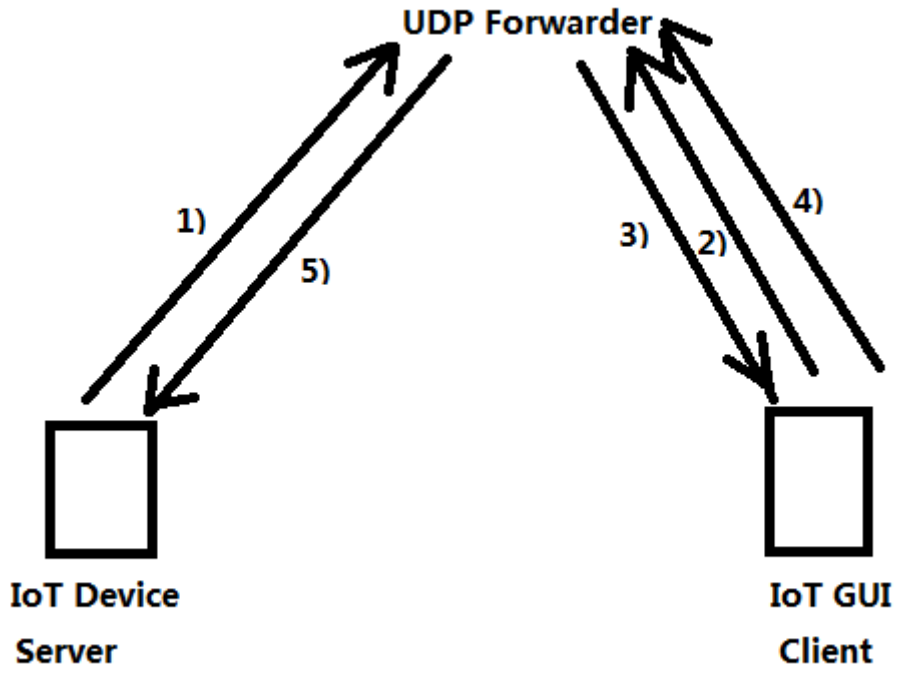
UDPForwarder Design



1) Der Server sendet Request an UDP Formwarder
2) Der Client fragt immer, ob eine Request von Server kommt.
3) UDP Forwarder sendet dem Request aus Server an Client.
4) Client bekommt den Request, dann sendet eine Response zurück
5) UDP Forwarder bekommt die Response aus Client, dann sendet eine Response an Server

1. UDPSocket ist ohne verbunden, schneller aber nicht zuverlässig. Wenn die Packet verloren ist, muss die andere immer warten. Also „long polling“

2. 1x pro Sekunde muss Client UDP Forwarder Kontrollen: Sind Daten für mich da? Wenn Client innerhalb 1000ms kein Request von UDP Forwarder bekommen hat, dann throw exception.

```
_socket.setSoTimeout(1000);
```

3. Pro IoT Device und IoT GUI hat eine eigenes IDs. Server und Client haben eigene IPs, Portnummern.

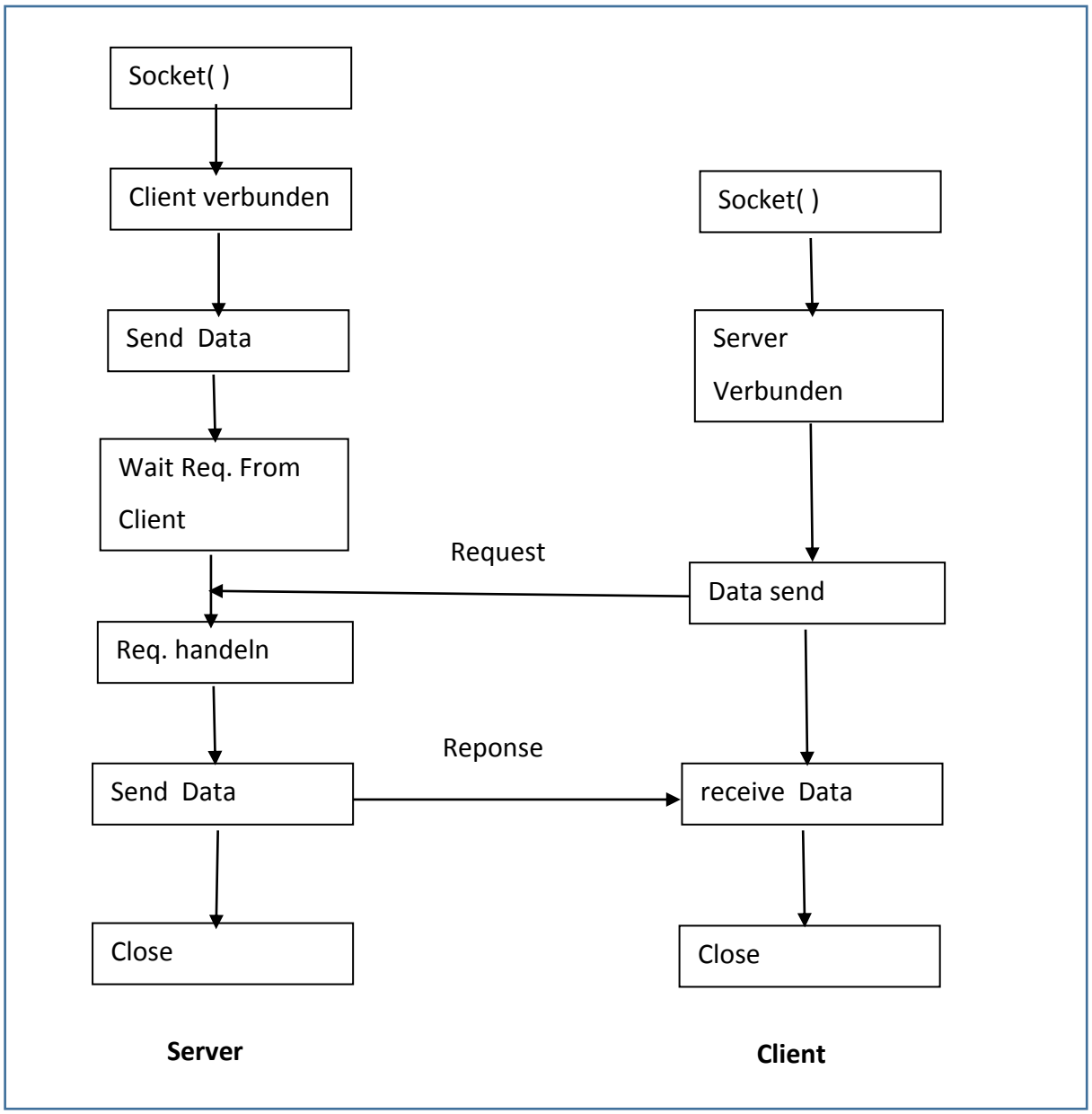
Pro Server:

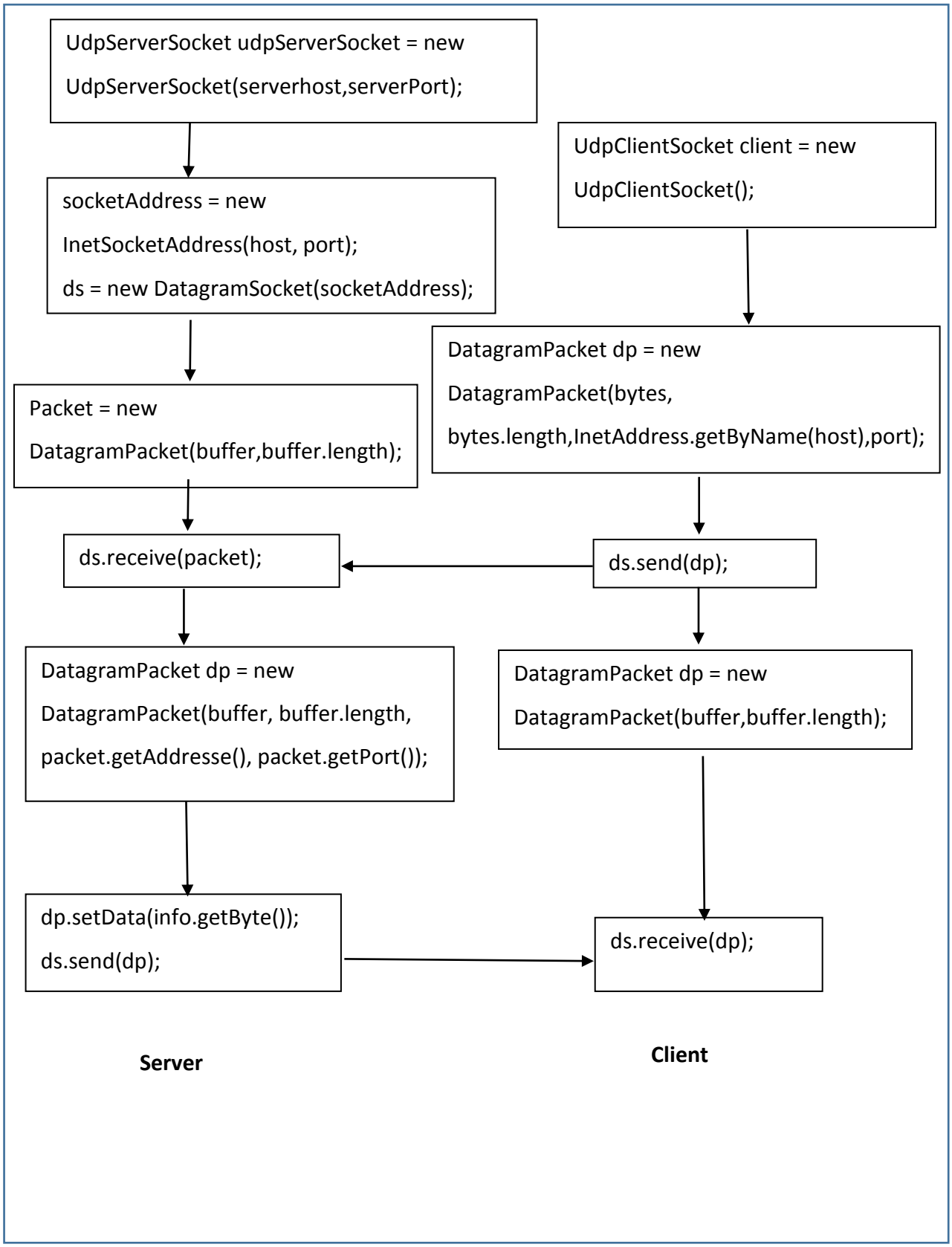
```
long id, String localIP, String localPort, int datagramSize
```

Pro Client:

```
long id, String targetAddress, String targetPort
```

4. Die DatagrammPacket wird in UDP Forwarder gespeichert, und wird nach der Reihe(FIFO) an Client schicken. Wenn UDP Forwarder eine Datagramm an Client geschickt ist, wird diese Datagramm in FIFO Queue gelöscht.




UdpServerSocket udpServerSocket = new
UdpServerSocket(serverhost,serverPort);
UdpClientSocket client = new
UdpClientSocket();
socketAddress = new
InetSocketAddress(host, port);
ds = new DatagramSocket(socketAddress);
DatagramPacket dp = new
DatagramPacket(bytes,
bytes.length,InetAddress.getByName(host),port);
Packet = new
DatagramPacket(buffer,buffer.length);
ds.receive(packet);
ds.send(dp);
DatagramPacket dp = new
DatagramPacket(buffer, buffer.length,
packet.getAddresse(), packet.getPort());
DatagramPacket dp = new
DatagramPacket(buffer,buffer.length);
dp.setData(info.getByte());
ds.send(dp);
ds.receive(dp);
Server
Client