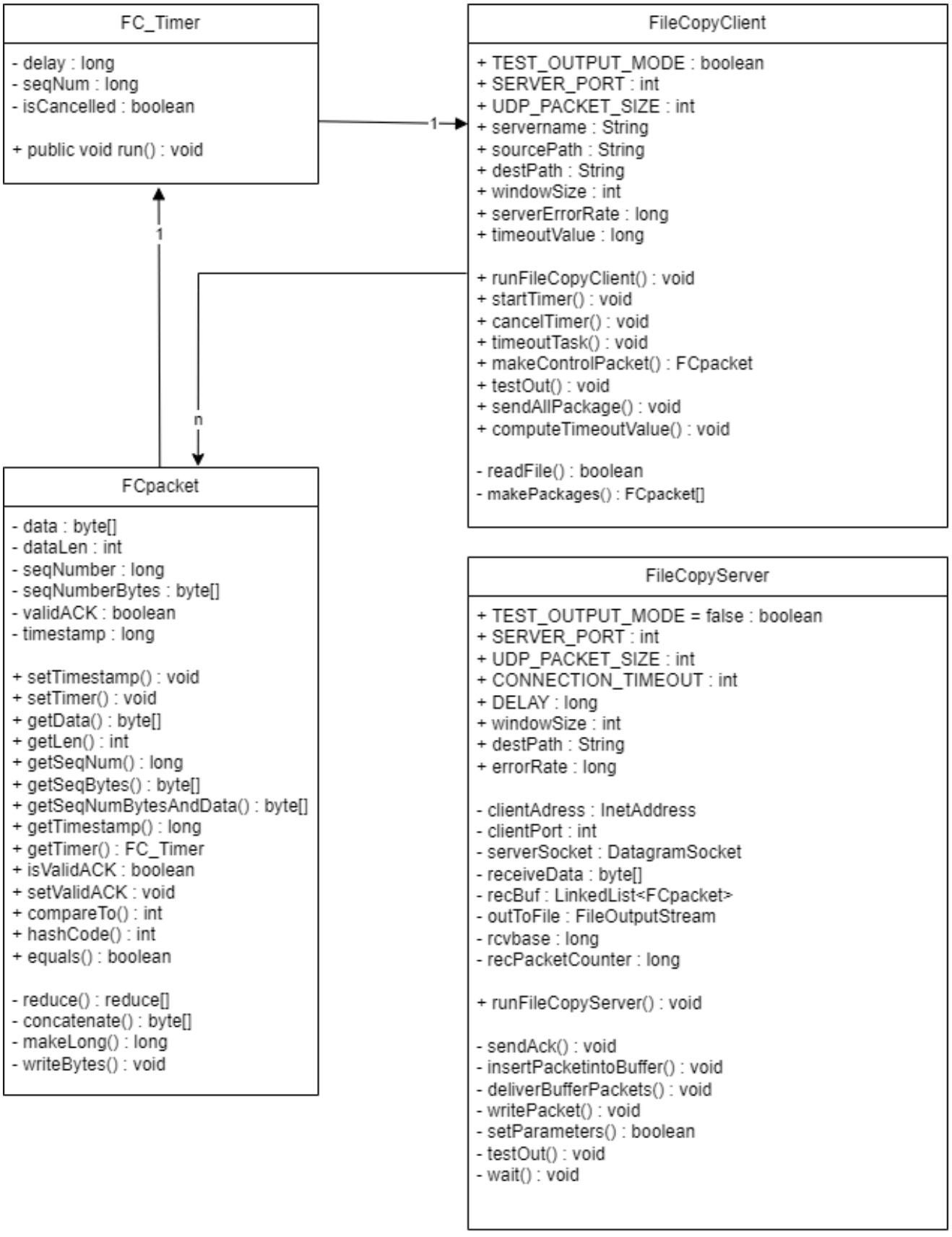
ToDos:

1. RTT-Berechnung
2. RTT-Messung (computeTimeoutValue(long sampleRTT))
3. Methode: - readFile() : boolean
4. Methode: makePackages() : FCpacket[]
5. timeoutTask(long seqNum)
6. Methode: sendAllPackage() : void
7. Methode: computeTimeoutValue() : void
8. runFileCopyClient()

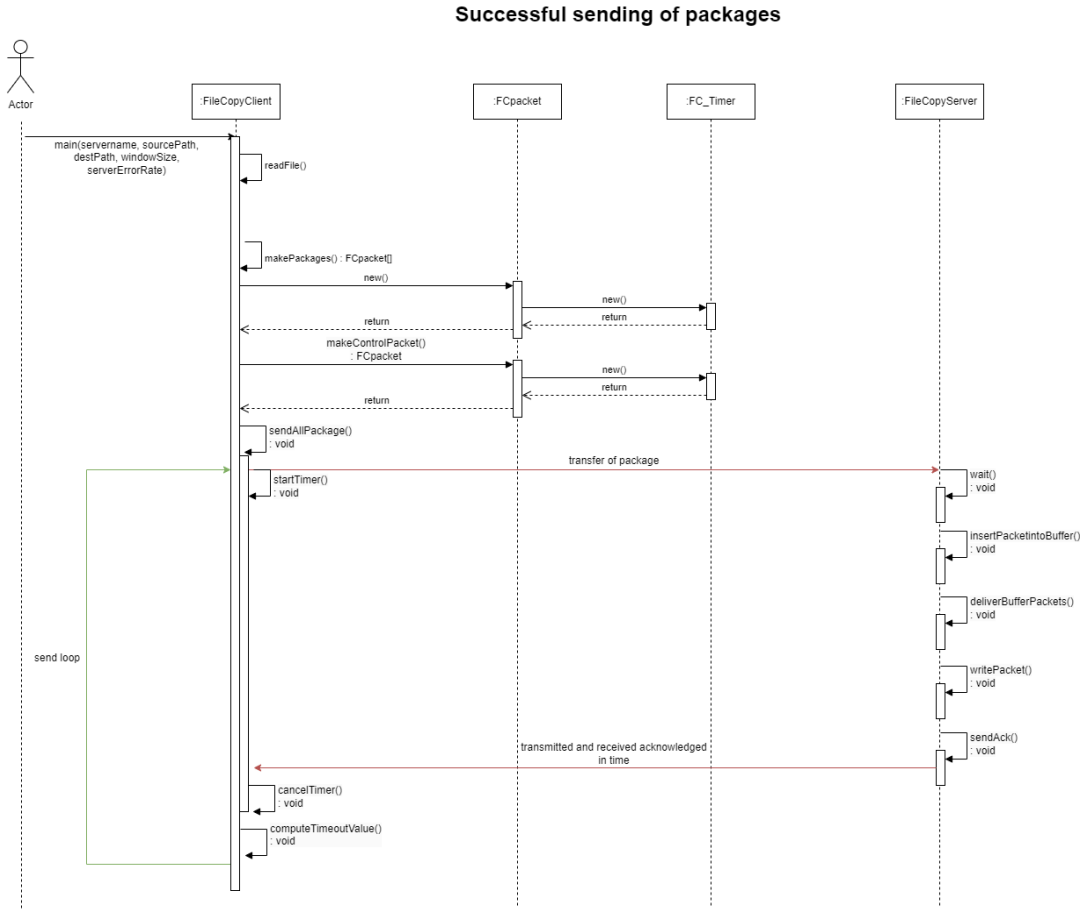
Ablaufpläne

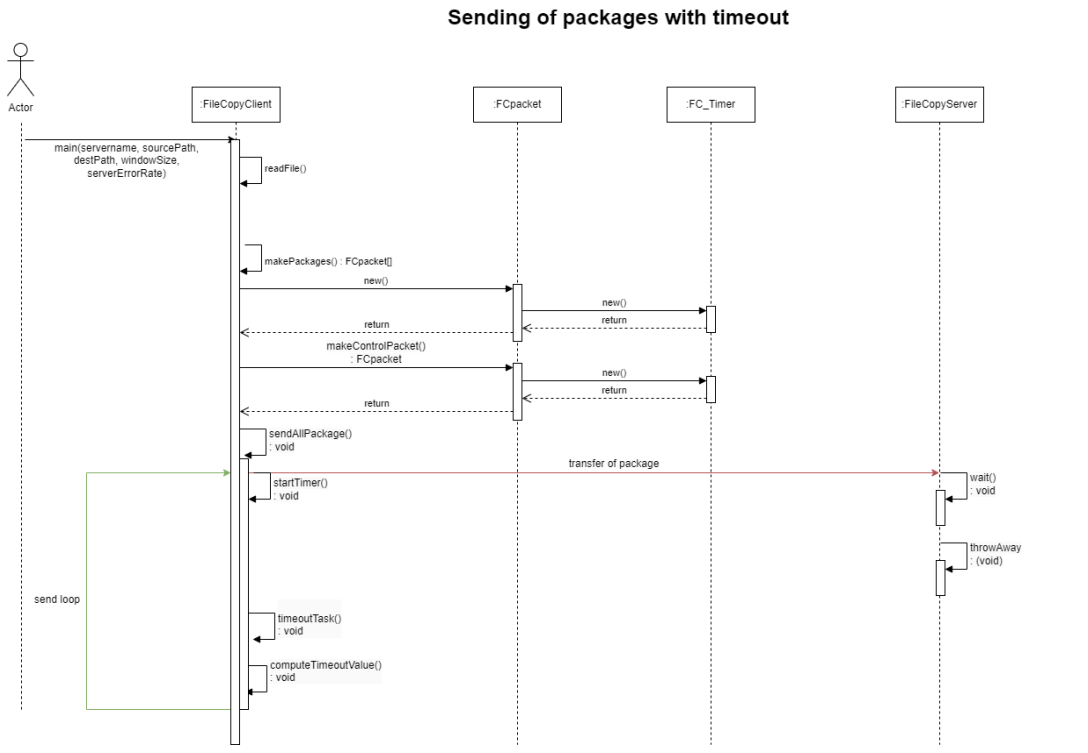
```
public void runFileCopyClient() {

    // ToDo!!
    // 1. Datei einlesen
      // 1.1 check file exists


    // 2. Make Packages from File
    // Make List Of All Packages

    // 3. Fill buffer

    //----> sendAllPackages()

}


public void SendAllPackages(){

  // 1. Send Control Package (explicitly or implicitly)
  // 2. Wait for Ack
  // 3. Loop : for all Packages
      // 3.1 SendPackage(pkg || sendBuffer[0] )
      // 3.2 Start Timer
      // 3.4 Wait for ACK ((OR Timeout))
        // 3.4.1 Remove Pkg from SendBuffer <-- new Method?
        // 3.4.2 No-ACK : TimeOut: resend (-> Loop Continue???))
          // 3.4.2.1 Increase TimeOutCounter
      // 3.5 : computeTimeoutValue
      // 3.6 CancelTimer()
      // 3.7 Put package into buffer and restart loop


  // Schreibe Statistik ....

}
```

UML


FC_Timer
- delay : long
- seqNum : long
- isCancelled : boolean
+ public void run() : void
1
1
FileCopyClient
+ TEST_OUTPUT_MODE : boolean
+ SERVER_PORT : int
+ UDP_PACKET_SIZE : int
+ servername : String
+ sourcePath : String
+ destPath : String
+ windowSize : int
+ serverErrorRate : long
+ timeoutValue : long
+ runFileCopyClient() : void
+ startTimer() : void
+ cancelTimer() : void
+ timeoutTask() : void
+ makeControlPacket() : FCpacket
+ testOut() : void
+ sendAllPackage() : void
+ computeTimeoutValue() : void
- readFile() : boolean
- makePackages() : FCpacket[]
n
FCpacket
- data : byte[]
- dataLen : int
- seqNumber : long
- seqNumberBytes : byte[]
- validACK : boolean
- timestamp : long
+ setTimestamp() : void
+ setTimer() : void
+ getData() : byte[]
+ getLen() : int
+ getSeqNum() : long
+ getSeqBytes() : byte[]
+ getSeqNumBytesAndData() : byte[]
+ getTimestamp() : long
+ getTimer() : FC_Timer
+ isValidACK : boolean
+ setValidACK : void
+ compareTo() : int
+ hashCode() : int
+ equals() : boolean
- reduce() : reduce[]
- concatenate() : byte[]
- makeLong() : long
- writeBytes() : void
FileCopyServer
+ TEST_OUTPUT_MODE = false : boolean
+ SERVER_PORT : int
+ UDP_PACKET_SIZE : int
+ CONNECTION_TIMEOUT : int
+ DELAY : long
+ windowSize : int
+ destPath : String
+ errorRate : long
- clientAdress : InetAddress
- clientPort : int
- serverSocket : DatagramSocket
- receiveData : byte[]
- recBuf : LinkedList<FCpacket>
- outToFile : FileOutputStream
- rcvbase : long
- recPacketCounter : long
+ runFileCopyServer() : void
- sendAck() : void
- insertPacketintoBuffer() : void
- deliverBufferPackets() : void
- writePacket() : void
- setParameters() : boolean
- testOut() : void
- wait() : void

FlowChart


Successful sending of packages
Actor
:FileCopyClient
:FCpacket
:FC_Timer
:FileCopyServer
main(servername, sourcePath, destPath, windowSize, serverErrorRate)
readFile()
makePackages() : FCpacket[]
new()
new()
return
return
makeControlPacket() : FCpacket
new()
return
return
sendAllPackage() : void
transfer of package
startTimer() : void
wait() : void
insertPacketintoBuffer() : void
deliverBufferPackets() : void
writePacket() : void
send loop
transmitted and received acknowledged in time
sendAck() : void
cancelTimer() : void
computeTimeoutValue() : void


Sending of packages with timeout
Actor
:FileCopyClient
:FCpacket
:FC_Timer
:FileCopyServer
main(servername, sourcePath, destPath, windowSize, serverErrorRate)
readFile()
makePackages() : FCpacket[]
new()
new()
return
return
makeControlPacket() : FCpacket
new()
return
return
sendAllPackage() : void
transfer of package
startTimer() : void
wait() : void
throwAway : (void)
send loop
timeoutTask() : void
computeTimeoutValue() : void