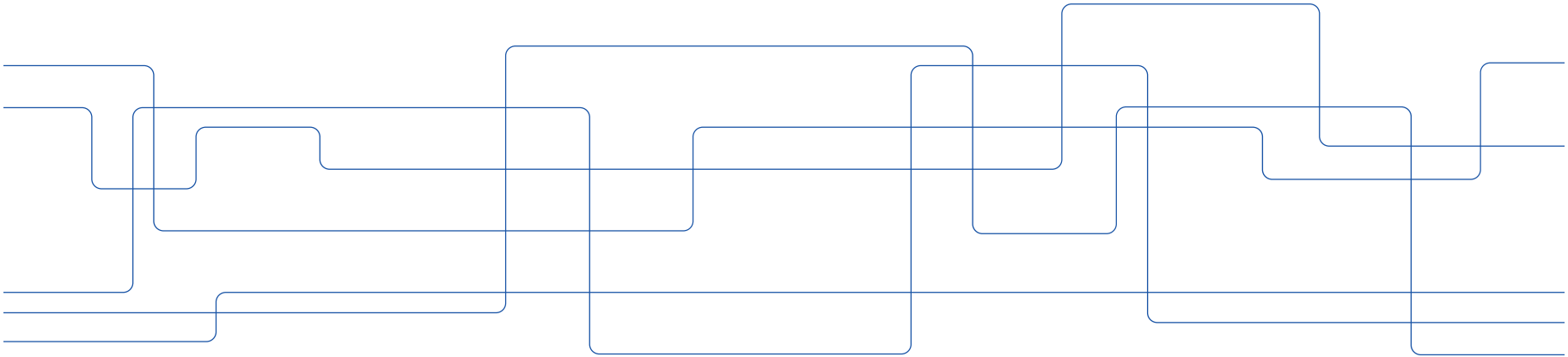
KTH ROYAL INSTITUTE
OF TECHNOLOGY

IK2215 Programming Assignment Introduction

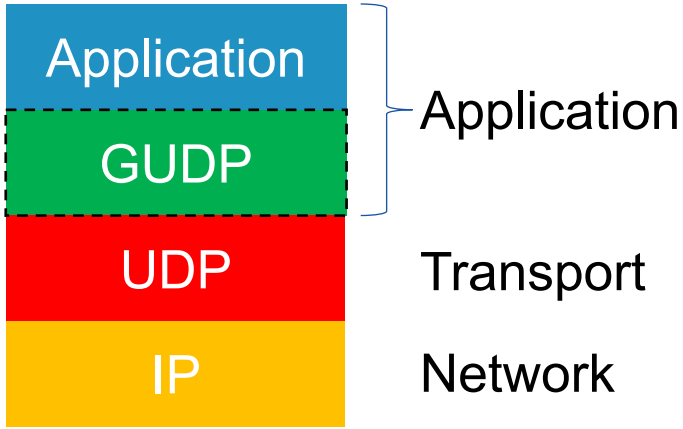
Voravit Tanyingyong



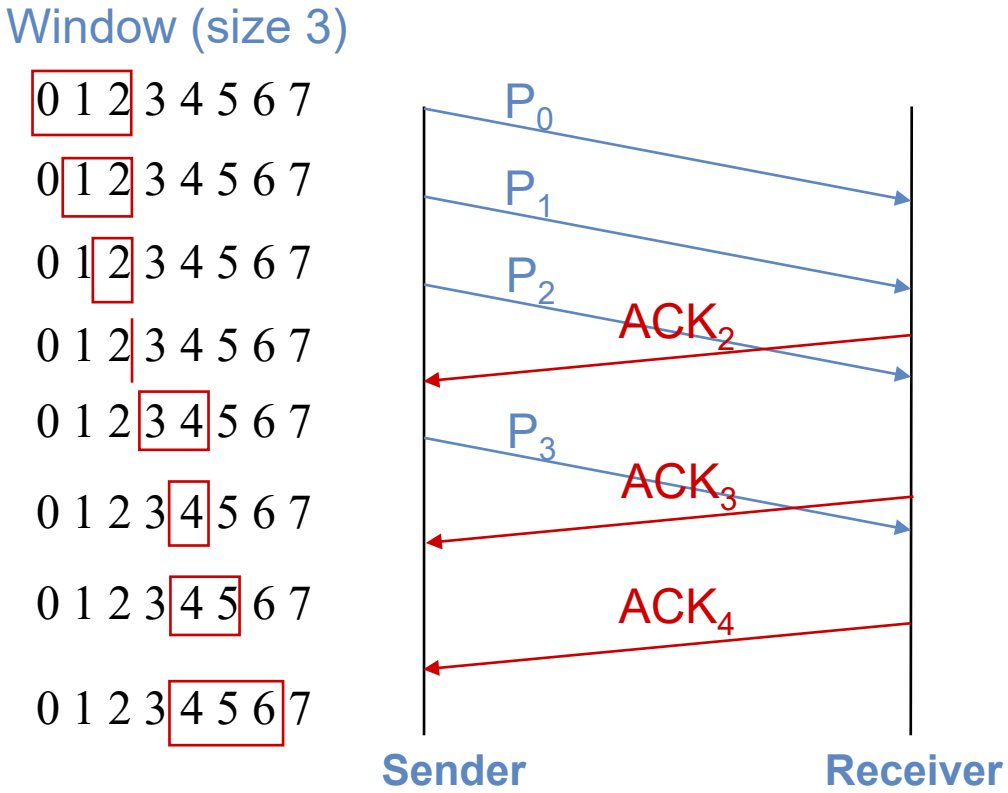
Programming assignment overview

- Design and implement a reliable protocol for sending/receiving datagrams

- Guaranteed UDP (GUDP)
  - Enabling reliable transport over UDP
  - Automatic repeat request (ARQ)
    - > uses acknowledgements and timeouts for reliable transmission
  - Sliding window flow control
    - > multiple packets in flight
  - Asynchronous communication
    - > Unlike TCP, GUDP is not connection-oriented (no connection establishment)



Sliding window flow control



Go-Back-N Sliding Window Protocol

- Understand the details of a basic, sliding window protocol
- An ACK is an ACK (and not a NACK)
  - The receiver sends an ACK only if it receives the next packet in sequence*
  - You cannot use an ACK to tell the sender that a packet has been lost
  - No duplicate ACK detection
- The sender increases the window in accordance with the ACK
- Retransmissions are triggered by timeouts (and nothing else)
  - Receiving an ACK with unexpected sequence number does not trigger a retransmission

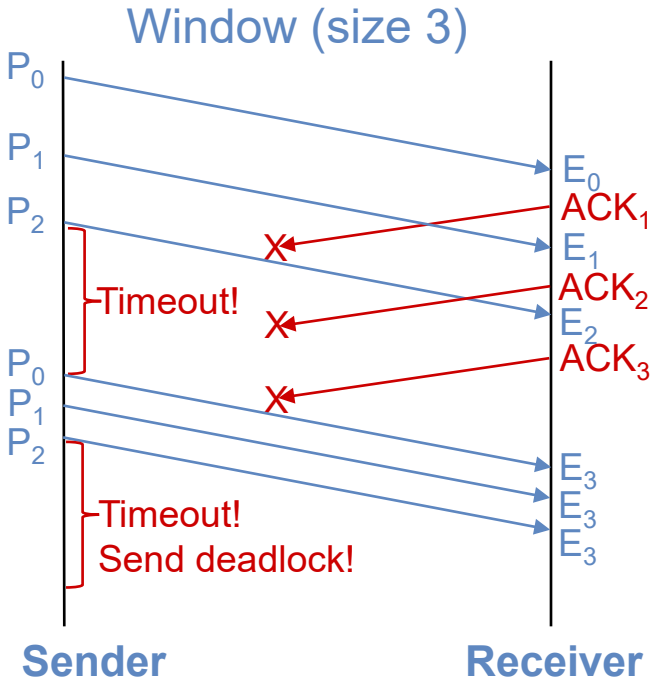
* There is an exception case, which we will discuss in the next slide

Exception case

- The receiver sends an ACK only if it receives the next packet in sequence
- A deadlock occurs when all ACKs (= number of window size) were lost

To resolve this problem:

- Receiver must send ACK with the expected sequence number when it receives a packet with a lower sequence number than the expected sequence number
- Sender upon receiving an ACK can assume all packets with (ACK sequence number - 1) were received successfully



GUDP implementation in java

- GUDP runs in user space, in the same process as the application

We provide:

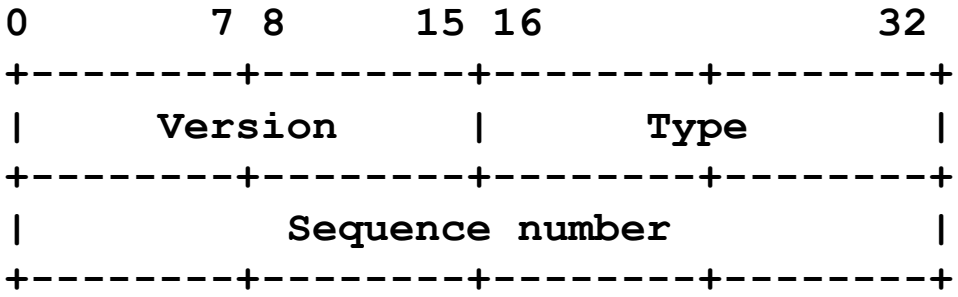
You are not allowed to modified these files!

- **GUDPPacket.java:** A class for GUDP protocol declarations with associated methods to access the GUDP packet header and payload
- **GUDPSocketAPI.java:** Well-defined API (Application Programming Interface) that you must use for your implementation
- **GUDPEndPoint.java (optional):** A class for keeping track of an end point

Your main task is to implement GUDP as a java class: **GUDPSocket.java**

- Sliding window flow control
- ARQ

GUDP header



- Version: version of the RUDP protocol
  - We use version 1!
- Type: packet type
  - BSN, DATA, ACK, and FIN
- How to use sequence numbers:
  - BSN packets: random
  - DATA packets: increases by one for each packet sent
  - ACK packets: sequence number of next expected DATA packet
  - FIN packets: sequence number of last DATA packet plus one

GUDPSocketAPI.java – API you must use

```
import java.net.DatagramPacket;
import java.io.IOException;

public interface GUDPSocketAPI {

    public void send(DatagramPacket packet) throws IOException;
    public void receive(DatagramPacket packet) throws IOException;
    public void finish() throws IOException;
    public void close() throws IOException;
}
```

- Your code must conform to this API
- Class/method declarations defined for the assignment
- You will write the GUDPSocket class that implements this API
  - You may add variables, methods, and inner classes in GUDPSocket.java

GUDPSocket.java – skeleton code for you

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.io.IOException;
public class GUDPSocket implements GUDPSocketAPI {
    DatagramSocket datagramSocket;
    public GUDPSocket(DatagramSocket socket) {
        datagramSocket = socket;
    }
    public void send(DatagramPacket packet) throws IOException {
    }
    public void receive(DatagramPacket packet) throws IOException {
    }
    public void finish() throws IOException {
    }
    public void close() throws IOException {
    }
}
```

send()

```
public void send(DatagramPacket packet) throws IOException;
```

- Send a packet
- The application put packet in the DatagramPacket format
- The destination address/port included in the packet
- Non-blocking – returns immediately
  - GDUP queue packet for future delivery

receive()

```
public void receive(DatagramPacket packet) throws IOException;
```

- Receive a packet
- The application fetch a packet from GUDP if there is one, otherwise wait until a packet arrives
- The application handles packets from different senders (which can be differentiated based on the information in the packet)

finish()

```
public void finish() throws IOException;
```

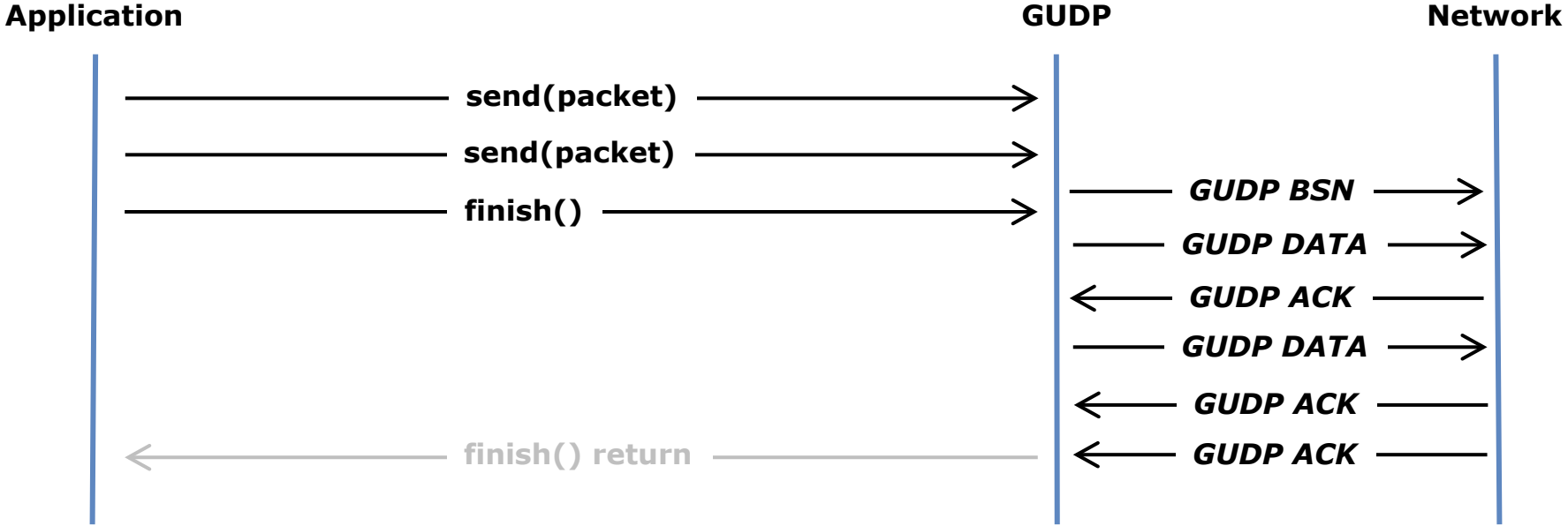
- Finish sending
- The application calls this method to inform GUDP that it's done sending
- GUDP completes the actual sending and return when it is done, otherwise report error/timeout by throwing the IOException
  - Retransmission may occur due to packet lost or arriving out-of-order
  - You may clean up data structure that you use to track destination end points

close()

```
public void close() throws IOException;
```

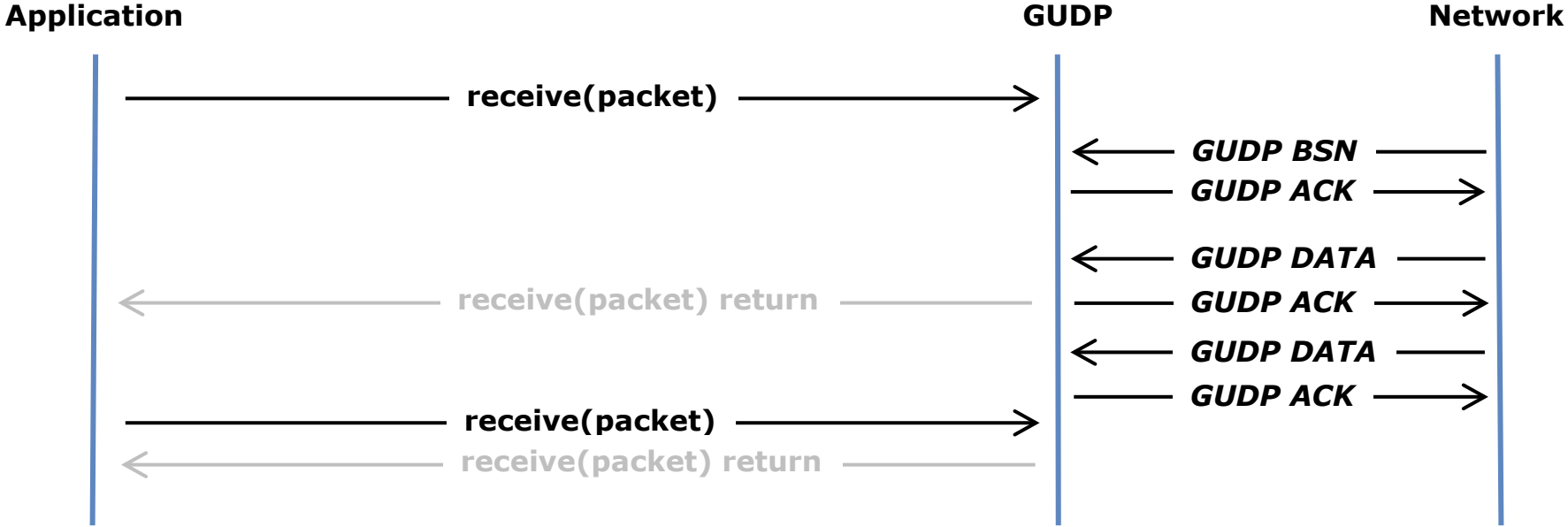
- Close the GUDP socket
- The application calls this method to terminate the GUDP socket
- GUDP cleans up, closes the socket, and return.

GUDP sender side



- Data transfer may happen after the application passed all packets to GUDP
- GUDP can send multiple packets (<= window size) before it receives any ACK

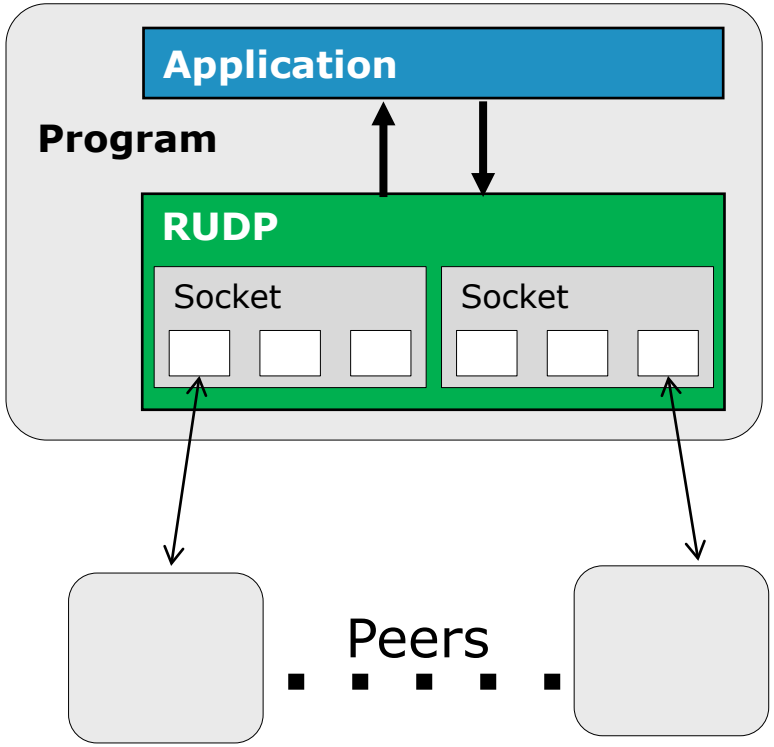
GUDP receiver side



- Receive returns only after GUDP has DATA
- Receiver may keep socket open to receive more DATA

Protocol control block

- An application can open multiple GUDP sockets
- Each GUDP socket can be used for communication with multiple peers
- Two levels
  - Multiple GUDP sockets
  - Multiple peers per socket
- Need to
  - Maintain state for per-socket “peers”
  - Have a way to look up peer state
  - Maintain queues with outbound packets



Grading overview

The application should be able to:

- Send one or more files to one or more destinations
- Receive multiple files from one or more sources
- Handle unexpected situations gracefully
- Work with other implementations

To pass, you must meet two criteria below:

1. Application must be able to **send and receive one file on one destination**
   - GUDP must be used in data transmission (show on the wire correctly)
   - Sliding window flow control is working correctly (multiple packets in-flight)
   - ARQ mechanism is working correctly (handle packet loss correctly)
2. And score **at least 3 out of 6 points**

- Deadline: **Tue 3 Oct** at 17:00
- Make-up deadline: **Wed 11 Oct** at 17:00

Plagiarism*

Plagiarism in practical work and computing code

“It is important that students ‘do their own work’ when they write computer code, when document an experiment, create a design or answer a mathematical problem. If they do not do these activities themselves, yet claim the results as their own, this is plagiarism.”

- Students who, with unauthorized aids or otherwise attempt to mislead the exam or when a student's performance is otherwise to be assessed, may lead to disciplinary action.

* More information on KTH webpage about Cheating and plagiarism

Grading test cases

>=2.5 points

1. **Multiple packets in-flight** **(0.5p)**
2. **Send and receive files with your code without loss** **(0.5p)**
3. **Send one file to other receiver without loss** **(0.5p)**
4. **Send one file to other receiver with loss** **(0.5p)**
5. **Receive one file from other sender without loss** **(0.5p)**
6. **Receive one file from other sender with loss** **(0.5p)**
7. Send one file to multiple receivers without loss (0.5p)
8. Send one file to multiple receivers with loss (0.5p)
9. Send multiple files to other receiver without loss (0.5p)
10. Send multiple files to other receiver with loss (0.5p)
11. Receive multiple files from other sender without loss (0.5p)
12. Receive multiple files from other sender with loss (0.5p)

>=3 points

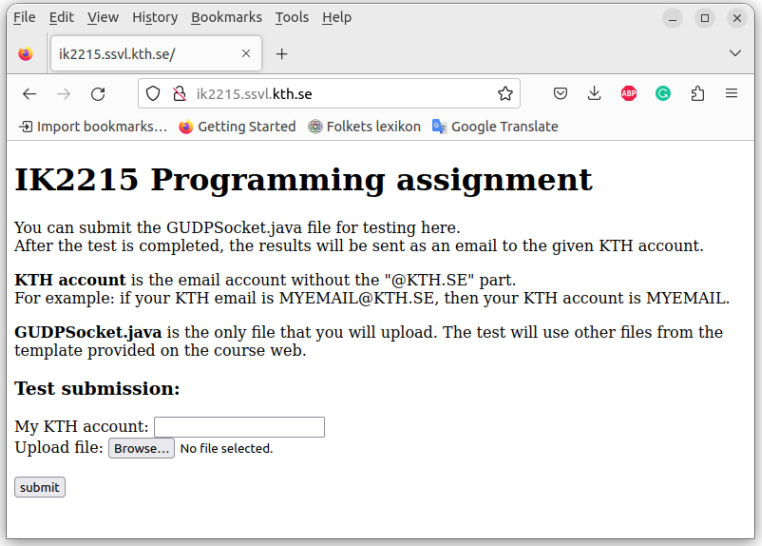
Testing

- We provide sample applications that you can use to test your GUDP code
  - VSFtp.java: A class for a simple file transfer protocol
  - VSSend.java: An application for sending files over VSFtp
  - VSRecv.java: An application for receiving files over VSFtp
- You are responsible for identifying relevant test cases and performing tests
- Think through the protocol carefully and know how it should work exactly
- Think through the dynamic behaviour of the GUDP library
  - What happens, and when?
- Define the protocol states and transitions
  - <current state, event, action, new state>
- If you have question:
  - Discussion forum: Q&A for lab activities
  - Q&A sessions for verbal discussion or additional support

Test service – http://ik2215.ssvl.kth.se

- You must provide:
  - Your KTH account i.e., KTH email without the “KTH@SE” part
  - Your GUDPSocket.java file

- The test runs at 00:00 everyday
  - Slow: > 5 minutes per submission
- Results send to provided KTH account

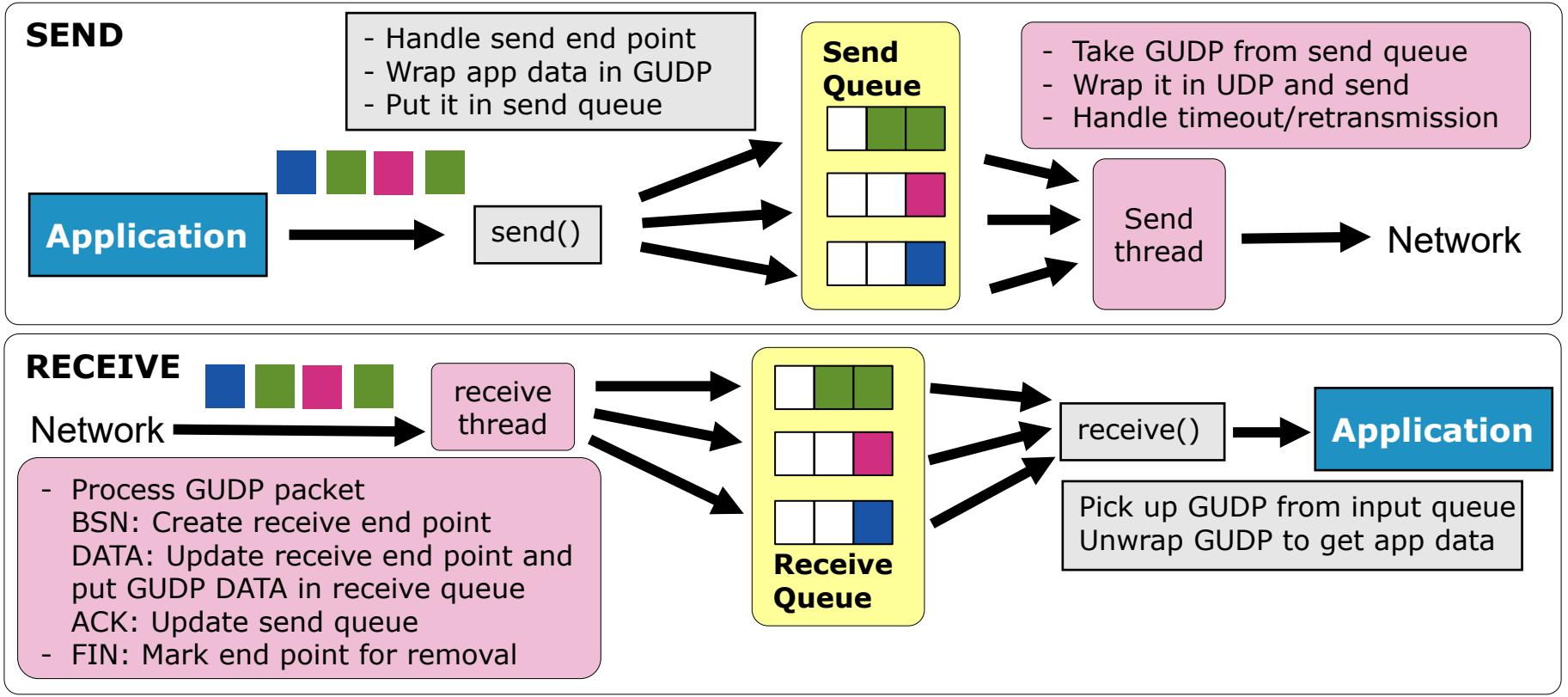


Example test output

```
OK: Code compiles without error.
### TEST1: Check sender packet content (0.5p)
OK: GUDP version must be 1
OK: First packet is GUDP BSN (type 2)
OK: Sequence number is random and not zero or one
OK: BSN packet contains only GUDP header
OK: GUDP version must be 1
OK: Second packet is GUDP DATA (type 1)
OK: Sequence number should be random and not zero
OK: Second packet has an increment sequence number
OK: data packet seems to contain GUDP header + payload
TEST1: OK      0.5p
### TEST2: send and receive files with your code without loss (0.5p)
OK: Your code can send and receive one file
OK: Your code can send and receive multiple files
TEST2: OK      0.5p
### TEST3: send one file to other receiver without loss (0.5p)
OK: Your code can send one file to other receiver
TEST3: OK      0.5p
### TEST4: send one file to other receiver with loss (0.5p)
OK: Your code can send one file when first BSN is lost
OK: Your code can send one file when first DATA is lost
OK: Your code can send one file when first FIN is lost
OK: Your code can send one file when first ACK is lost
OK: Your code can send one file with random loss
TEST4: OK      0.5p
### TEST5: receive one file from other sender without loss (0.5p)
OK: Your code can receive one file from other sender
TEST5: OK      0.5p
```

```
### TEST6: receive one file from other sender with loss (0.5p)
OK: Your code can receive one file when first BSN is lost
OK: Your code can receive one file when first DATA is lost
OK: Your code can receive one file when first FIN is lost
OK: Your code can receive one file when first ACK is lost
OK: Your code can receive one file with random loss
TEST6: OK      0.5p
### TEST7: send one file to multiple receivers without loss (0.5p)
OK: Your code can send one file to multiple receivers
TEST7: OK      0.5p
### TEST8: send one file to multiple receivers with loss (0.5p)
OK: Your code can send one file to multiple receivers
TEST8: OK      0.5p
### TEST9: send multiple files to other receiver without loss (0.5p)
OK: Your code can send multiple files to other receiver
TEST9: OK      0.5p
### TEST10: send multiple files to other receiver with loss (0.5p)
OK: Your code can send multiple files to other receiver
TEST10: OK    0.5p
### TEST11: receive multiple files from other sender without loss (0.5p)
OK: Your code can receive one file from other sender
TEST11: OK    0.5p
### TEST12: receive multiple files from other sender with loss (0.5p)
OK: Your code can receive one file from other sender
TEST12: OK    0.5p
##########
IMPORTANT: You pass only if scores of TEST1-6 >=2.5 points and TEST1-12 >=3.0 points.
You get the scores only when you pass. Otherwise, you get 0 points
RESULTS: PASS
SCORE: 6.0
##########
```

Example of send and receive implementation



Useful resources

- Course book: 8th and 7th edition
  - Read Chapter 3.4 through Chapter 3.4.3 Go-Back-N (GBN)
- TCP Operational Overview and the TCP Finite State Machine (FSM)
- Producer-consumer in Java: Baeldung, geeksforgeeks
- Java queue implementations: Oracle, Baeldung, geeksforgeeks,
- Java documentation for different classes:
  - DatagramSocket, DatagramPacket,
  - LinkedList, ArrayDeque
- Java wait() and notify() methods