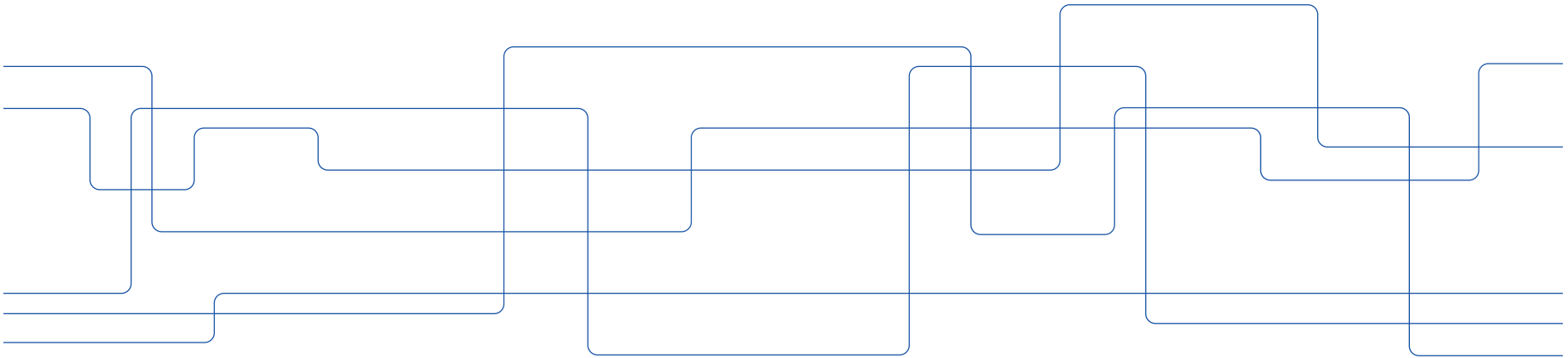
KTH ROYAL INSTITUTE OF TECHNOLOGY

IK2215 Programming Assignment Introduction

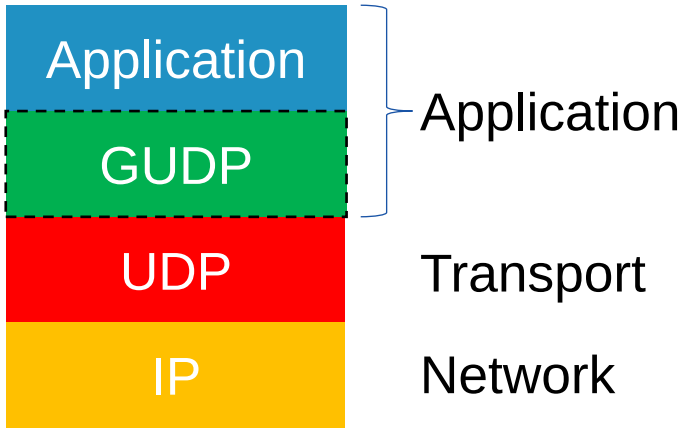
Voravit Tanyingyong



Programming assignment overview

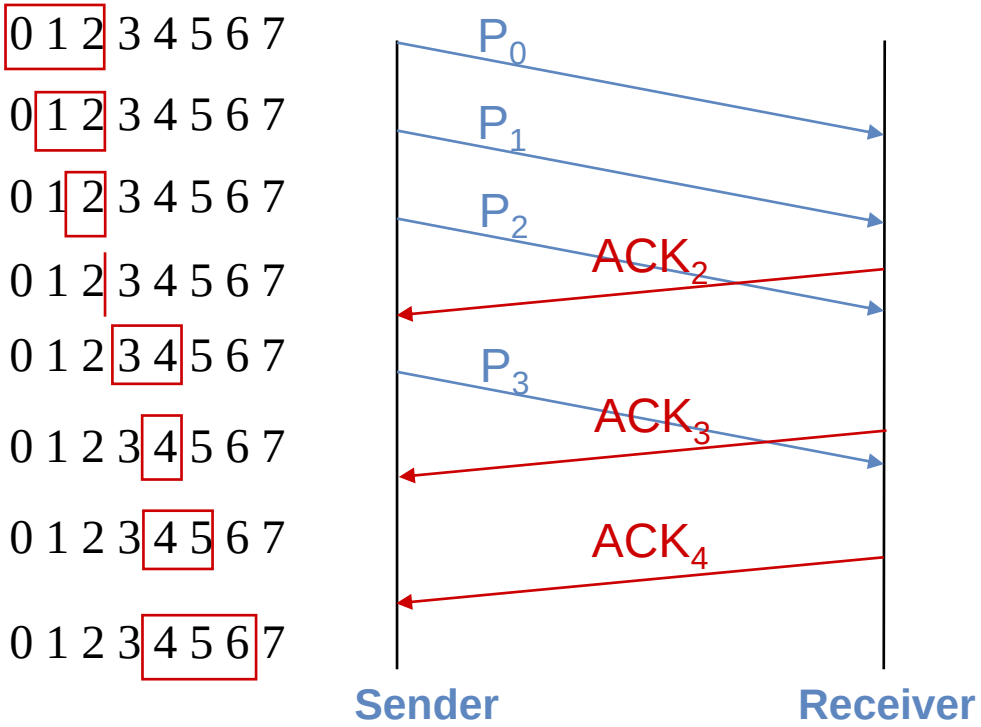
- Design and implement a reliable protocol for sending/receiving datagrams

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



Sliding window flow control

Window (size 3)



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

GUDP implementation in java

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

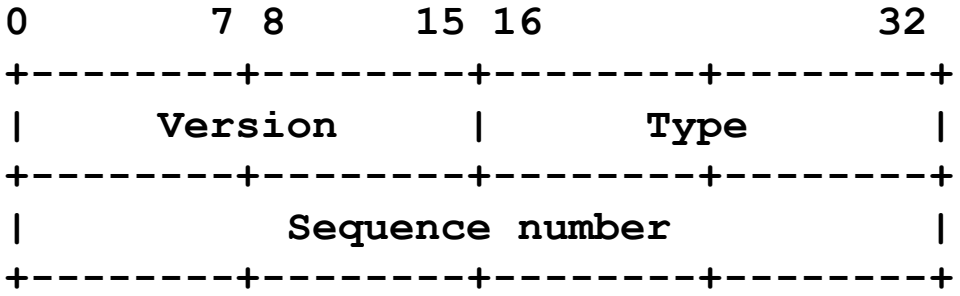
We provide:

- A class for GUDP protocol declarations with associated methods to access the GUDP packet header and payload (GUDPPacket.java)
- Well-defined API (Application Programming Interface) that you must use for your implementation (GUDPSocketAPI.java)

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, and ACK
- 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

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 provided by us
- You will write the GUDPSocket class that implements this API
  - You may add methods and inner classes

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 sender address/port included in the provided DatagramPacket packet
- The application fetch packet from GUDP if there is one, otherwise wait until a packet arrives

finish()

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

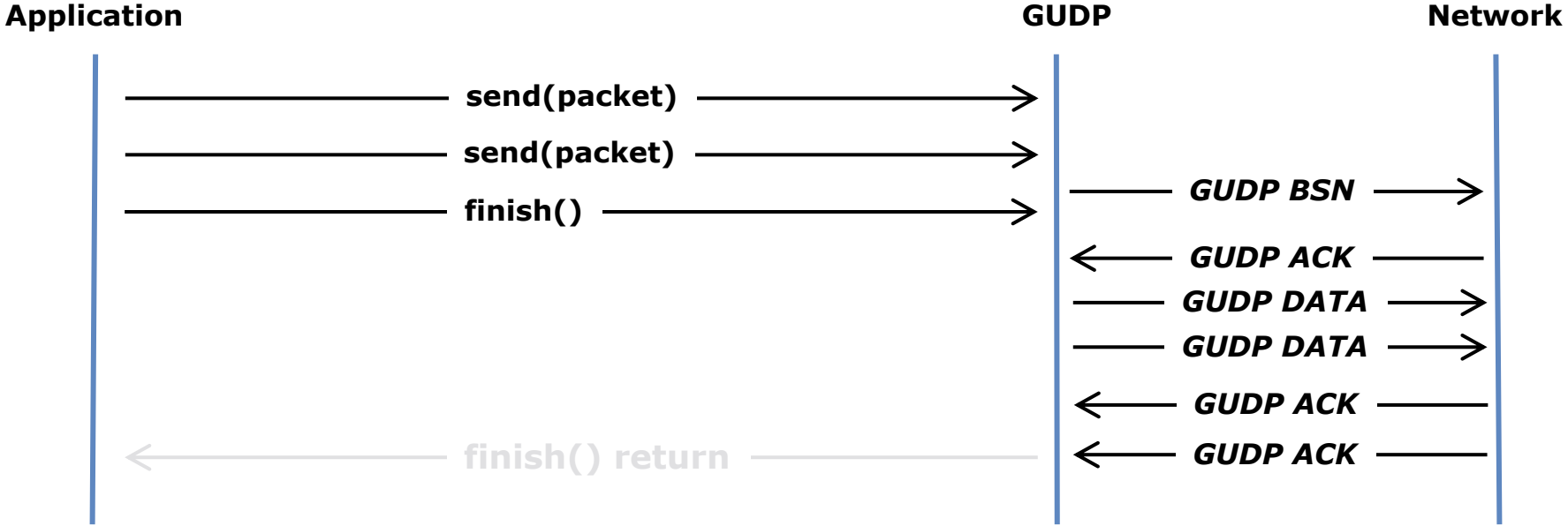
- Finish sending
- The application calls this method to inform GUDP that it’s done sending
- GUDP completes the actual sending and return when it is done, otherwise report error/timeout by throwing the IOException
  - Retransmission may occur due to packet lost or arriving out-of-order

close()

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

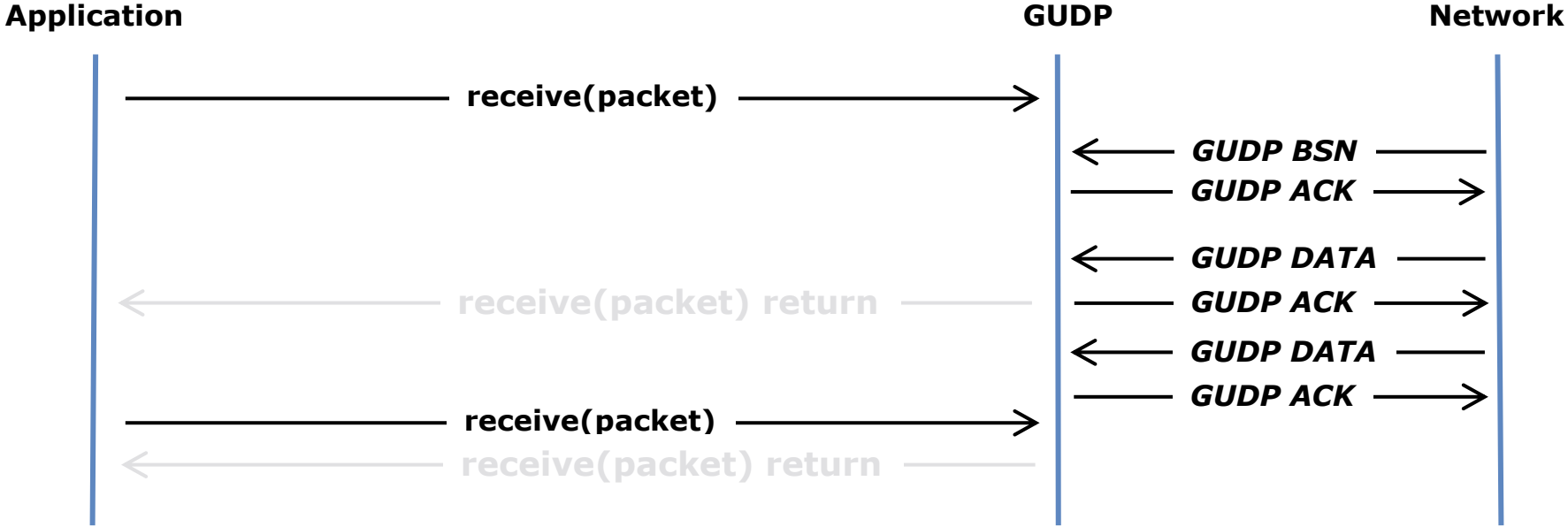
- Close the GUDP socket
- The application calls this method to terminate the GUDP socket
- GUDP cleans up, closes the socket, and return.

GUDP sender side



- Note that in this example, the entire data transfer takes place after the application passed all packets to GUDP
  - It could happen for you as well

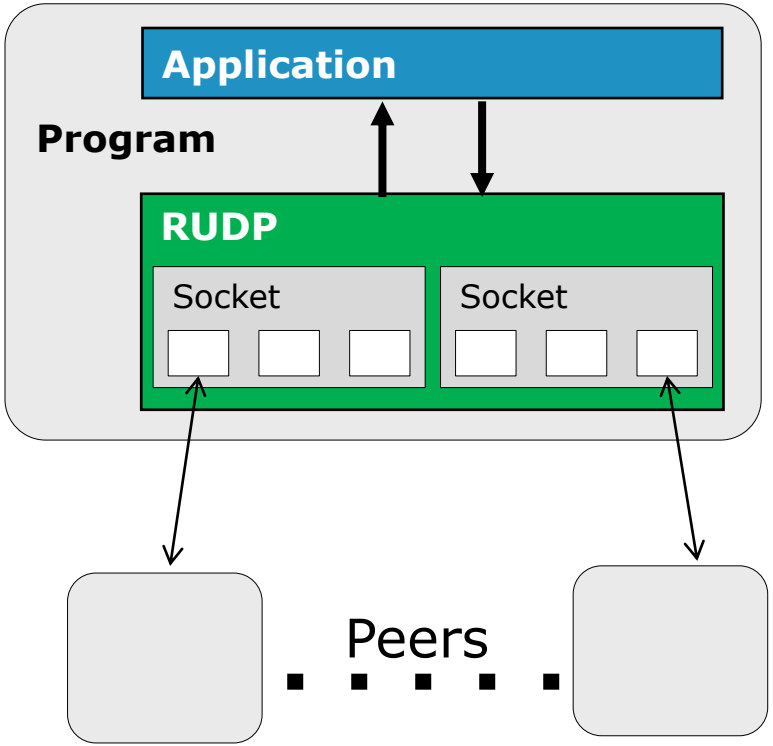
GUDP receiver side



- 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

- Score at least 3 out of 6 points
- To get a pass (3 points), an application must be able to **send one file to one destination** and your implementation must show the following:
  - GUDP must be used in data transmission (show correctly on the wire)
  - Sliding window flow control is working correctly
  - ARQ mechanism is working correctly
- 1 extra point if an application can **send multiple files to one destination**
- 1 extra point if an application can **send multiple files to multiple destinations**
- 1 extra point if your implementation can **handle unexpected situations gracefully**

- Deadline: **Mon 3 Oct** at 17:00
- Make-up deadline: **Mon 17 Oct** at 17:00

Testing

- You submit your code once you are done before the deadline
- We will test it and check that it works after the deadline
- You are responsible for identifying relevant test cases and performing tests
- Discuss on the lab forum

- We will try to provide you with sample applications

What You Need to think about

- One of the main challenges is understanding exactly how the code should work
- Think through the protocol carefully
- 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 or recitation sessions for verbal discussion or additional support

Example of send and receive implementation