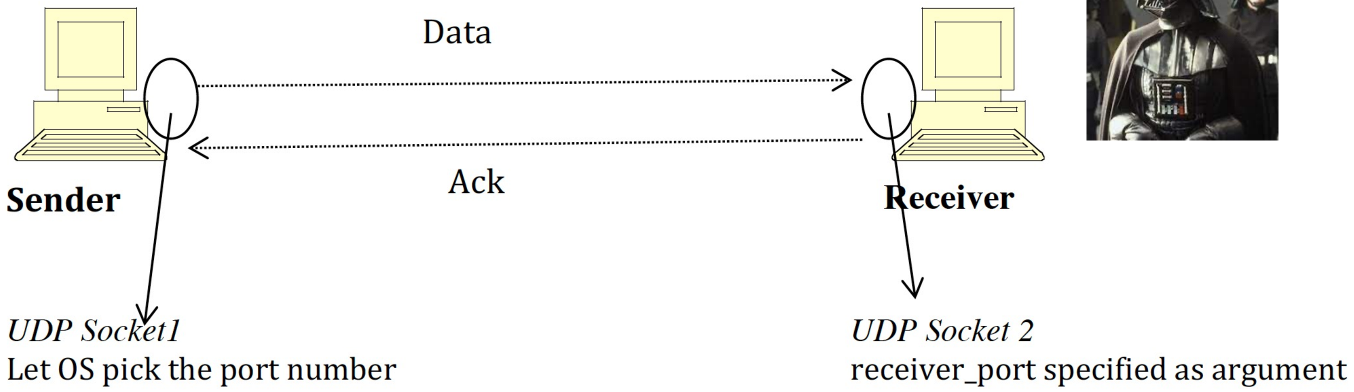
COMP 3331/9331 Assignment T2 2021

All details are in the specification

- **READ THE SPECIFICATION**
- **READ THE SPECIFICATION (AGAIN)**
- Information about deadlines, file names, submission instructions, marking guidelines, example interactions and various other specifics are in the specification
- Choice of programming languages: C, Java, Python
- This talk provides a high-level overview

Padawan Transport Protocol


Data
Ack
Sender
Receiver
UDP Socket1
Let OS pick the port number
UDP Socket 2
receiver_port specified as argument

Goal: Implement a stripped-down version of TCP for reliable uni-directional transfer of data from Sender to Receiver

Must use UDP

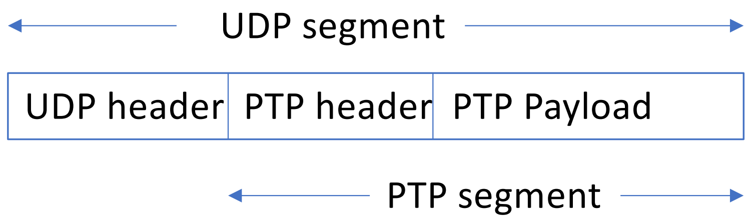
PTP - Inclusions

- 3-way connection setup (SYN, SYN+ACK, ACK)
- 4-way connection termination (FIN, ACK, FIN, ACK)
  - Possible to combine the ACK and FIN from the Receiver in one message (effectively making it a 3-way process)
- Sender must maintain a single timer and transmit the oldest unacknowledged segment
- Receiver should buffer out of order segments
- Fast retransmit (i.e., retransmission on reception of triple duplicate ACKs)
- Include sequence and ack numbers like TCP
- Use MWS (command-line argument) as window size
- Use MSS (command-line argument) as the size of the data payload in PTP segment
  - MWS >= MSS and exactly divisible by MSS

PTP - Exclusions

- No need to randomize initial sequence number
- No need to implement timeout estimation (timeout value provided as command line argument)
- No need to double timeout interval
- No need to implement delayed ACKs
- No need to implement any flow control or congestion control
- No need to deal with corrupted packets
- No need to deal with abnormal behaviour (e.g., RST)

PTP Segment Format



- You will need to decide on the format of the PTP headers. You can draw inspiration from TCP.
- PTP Segments will be encapsulated within UDP segments. No need to include port #'s in the PTP header. You will have to fill the port numbers in the UDP headers.
- Same format for data and ACK segments
  - ACK segment contains no data

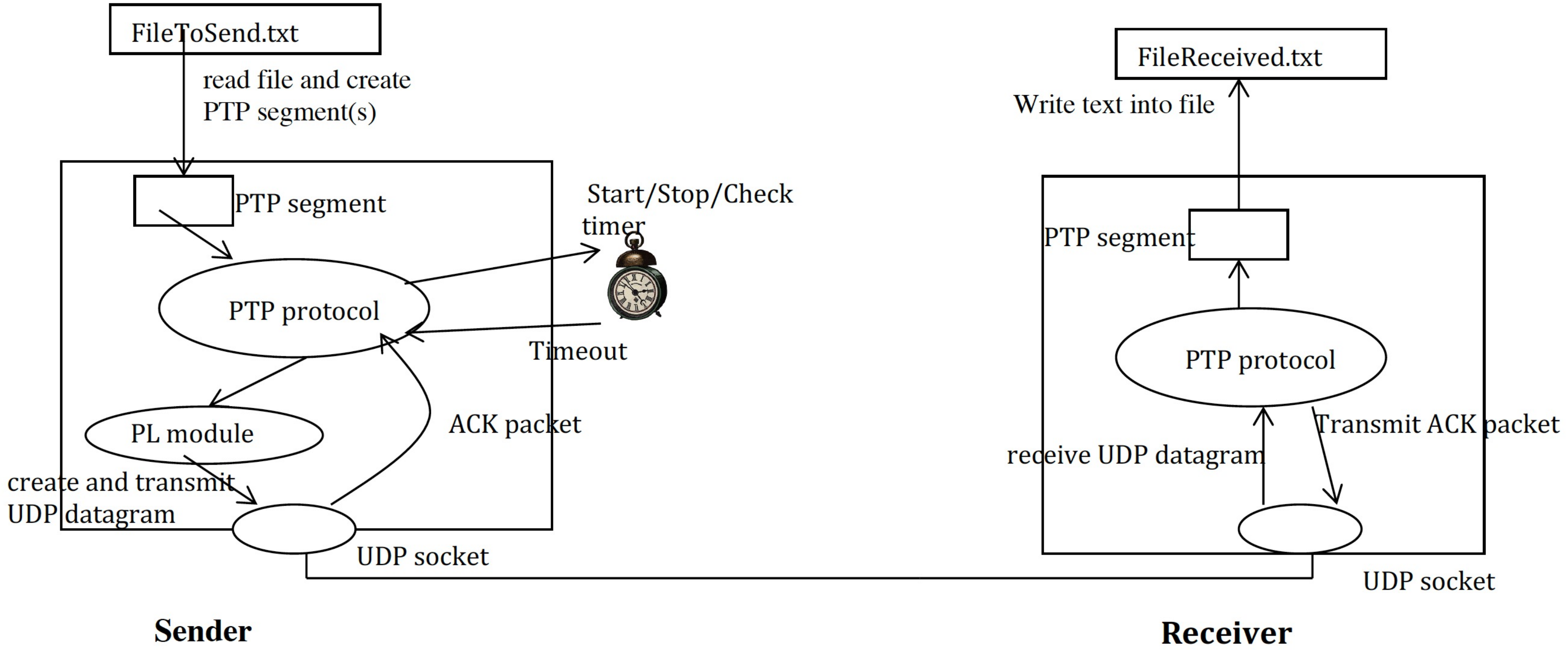
PL Module

- Purpose: simulate packet loss
- Implemented at the Sender
- You can assume that packets will never be delayed or corrupted
- Exclude PTP segments for connection establishment or teardown
- Exclude acknowledgments
- PTP data segments are dropped with a probability pdrop
- Code fragments provided for pseudo-random generation
- You are required to use a fixed seed, provided as command line argument

Execution

- Receiver
  - Command line arguments:
    - receiver_port (use a value greater than 1023 and less than 65536)
    - FileReceived.txt (to be created by your program into which received data is written)
  - Executed first – waits for Sender to connect
- Sender
  - Command line arguments:
    - receiver_host_ip (use "127.0.0.1" as Sender and Receiver will be executed on same machine)
    - Receiver_port (should match the first argument for the Receiver)
    - FileToSend.txt (text file to be sent to receiver, file exists in current working directory)
    - MWS: maximum window size in bytes
    - MSS: maximum segment size in bytes
    - Timeout: value of timeout in milliseconds
    - Pdrop: packet drop probability, between 0 and 1
    - Seed: random number generation seed (an integer)
  - Let the OS pick an unused port
  - Sender should send UDP segments to Receiver ("127.0.0.1", receiver_port)
- You may assume that the correct number of command line arguments in the expected format will be always provided

Execution


FileToSend.txt
read file and create PTP segment(s)
PTP segment
PTP protocol
Start/Stop/Check timer
Timeout
PL module
ACK packet
create and transmit UDP datagram
UDP socket
Sender
FileReceived.txt
Write text into file
PTP segment
PTP protocol
Transmit ACK packet
receive UDP datagram
UDP socket
Receiver

Receiver Design

1. Connection setup
2. Data Transmission (repeat until end of file)
   a) Receive PTP segment
   b) Send ACK segment
   c) Buffer data or write data into file
3. Connection teardown

Sender Design

1. Connection setup
2. Data Transmission (repeat until end of file)
    a. Read file
    b. Create PTP segment
    c. Start Timer if required (retransmit oldest unacknowledged segment on expiry)
    d. Send PTP segment to PL module
    e. If PTP segment is not dropped, transmit to Receiver
    f. Process ACK if received
3. Connection teardown

Note: Sender needs to deal with multiple events, so you wish to explore the use of – (i) multiple threads (ii) non-blocking or asynchronous IO using polling, i.e., select()

Logs – Very Important

- Sender_log.txt
  - <snd/rcv/drop> <time> <type of packet> <seq-number> <number-of-bytes> <ack-number>
  - Statistics at the end of the file transfer
- Receiver_log.txt
  - Similar format
  - Statistics at the end of the file transfer
- Samples provided in the spec
- Fields should be tab-separated
- IMPORTANT: If logs are missing, then your submission will only be marked out of 25% of the marks

Marking Criteria

- Simple stop-and-wait (5 marks)
  - MWS = MSS
    - Pdrop = 0
    - Pdrop = different values
- Pipelining (12 marks)
  - MWS > MSS
    - Pdrop = 0
    - Pdrop = different values
  - Varying MWS, pdrop, timeout
- Report (3 marks)
- Non-CSE Students may opt for a reduced-functionality spec
    - **MUST request approval** – check spec for details

How to start and progress

- Start with a stop-and-wait protocol – one packet at a time (similar to RDT 3.0) without the PL module
  - Make sure you can transfer a file correctly from Sender to Receiver
- Next introduce the PL module
  - Make sure you can transfer a file correctly from Sender to Receiver
- Extend to a window-based protocol (i.e., sending MWS bytes at a time)
  - First disable PL module and ensure that a file can be transferred correctly
  - Next enable PL module and ensure that a file can be transferred correctly
- **STRONGLY SUGGEST TO DEVELOP YOUR IMPLEMENTATION IN VLAB ENVIRONMENT (NOT ON YOUR NATIVE MACHINE)**
  - Added benefit – CSE accounts are backed up
- **IF you develop on your machine, make sure you TEST EXTENSIVELY IN VLAB**

Resources

- Many program snippets are on the web page
  - Including multi-threading code snippets
- Your socket programming experience in Labs 2 and 3 will be useful
- Repository of resources is here

Testing

- Test, Test, Test
- Server and client(s) executing on **same machine**
- Emphasis on correct behaviour
- **MUST Test In VLAB environment**
- If we cannot run your code, then we cannot award you any marks
- Your assignment will be MANUALLY marked by your tutors

Plagiarism

- **DO NOT DO IT**
- If caught
  - You will receive zero marks (and there may be further repercussions if this is not your first offence)
  - Your name will be added to the school plagiarism register

Seeking Help

- Assignment specific consults (for all 3 programming languages) from Weeks 7-10
  - Schedule is available on assignment page
- Course message forum
  - Read posts from other students before posting your question
- Read the spec – very often your answer will be in there