Project 2: Reliable data transfer over UDP

Fall 2020
COMP4621

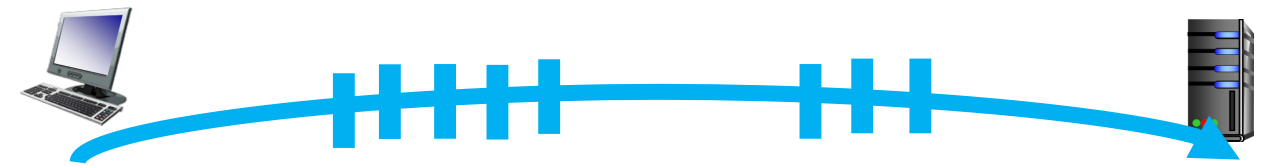
TA: XIE Wentao
wxieaj@cse.ust.hk

Project 2 Materials

Project overview

- In this project, you are asked to implement an application layer-level reliable data transfer protocol that enables reliable file transfer from a sender to a receiver.
- All implementation should be written in Python3. No high-level networking-related abstractions are allowed in this project.
- Project release: Nov. 6 (Friday).
- Project deadline: Dec. 5 (Saturday) 23:59 HKT
- Phase 1 (Nov. 6 ~ Nov. 20): only interfaces will be released.
- Phase 2 (Nov. 21 ~ Dec. 5): skeleton code will be released.
- Submissions within phase 1 will receive 10 (/100) bonus points.

Task description



1. You need to design your RDT protocol that runs on the sender.
2. Protocol needs to **support pipelining**.
3. Protocol needs to handle **corrupted packets**.
4. Protocol needs to handle **packet loss**.
5. Protocol needs to handle **out-of-order packets**.

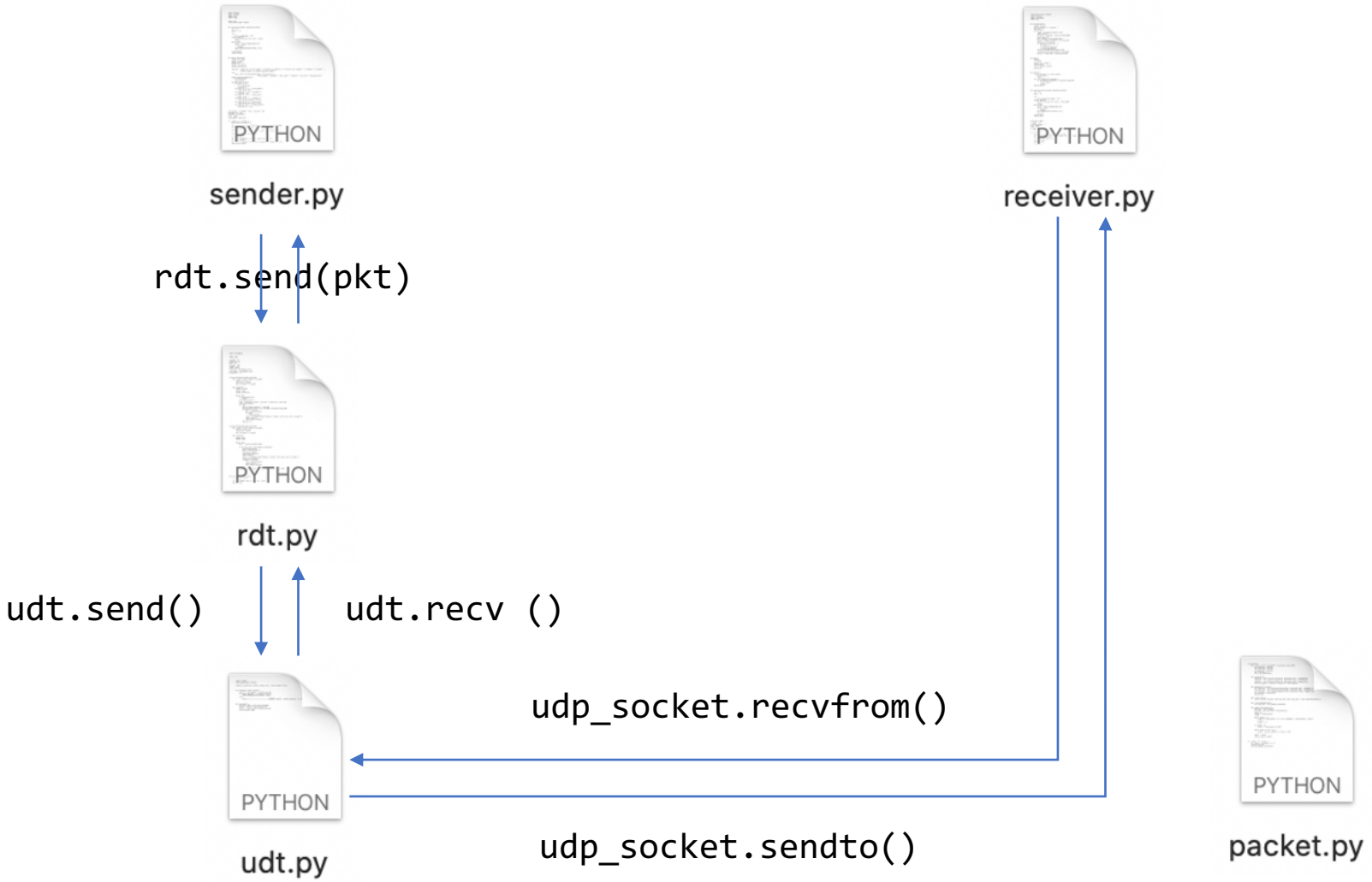
1. The implementation of the receiver is provided.
2. Receiver sends ACKs only when the received packet is with the expected sequence number and the checksum is correct (like in RDT3.0).
3. Receiver sends cumulative ACKs to the sender (like in Go-Back-N).
4. ACK number is the latest sequence number in the buffer.

Task description

Your design

Provided

Task description – project logic



Task description – `udt.py`

Functions:

- Send packets to a UDP socket.
- Receive packets from a UDP socket.

```python
from packet import Packet


def send(sock, addr, packet):
    print('<- Send packet', packet.seq_num)
    sock.sendto(packet.encode(), addr)


def recv(sock):
    pkt_byte, addr = sock.recvfrom(1024)
    packet = Packet().decode(pkt_byte)
    print('-> Receive ACK', packet.ack_num)
    return packet, addr
```

Task description – `receiver.py`

```python
def recv(sock):
    global rcv_pkt_buffer
    print('Receiver is running.’)
    ack_num = -1
    while True:
        pkt_byte, snd_addr = sock.recvfrom(1024)

        pkt = Packet().decode(pkt_byte)
        print('-> Receive packet', pkt.seq_num)

        if (pkt.seq_num == ack_num + 1) & (pkt.chk_sum == pkt.compute_checksum()):
            ack_num += 1
            f.write(pkt.payload)
            rcv_pkt_buffer.append(pkt)

        ack_pkt = Packet(ack_num=ack_num)
        sock.sendto(ack_pkt.encode(), snd_addr)
        print('<- Send ACK', ack_pkt.ack_num)
```

Functions

1. Reply an ACK when the received packet is not corrupted and has the right sequence number.
2. Check whether the file is correctly received. (For TAs to evaluate your implementation)

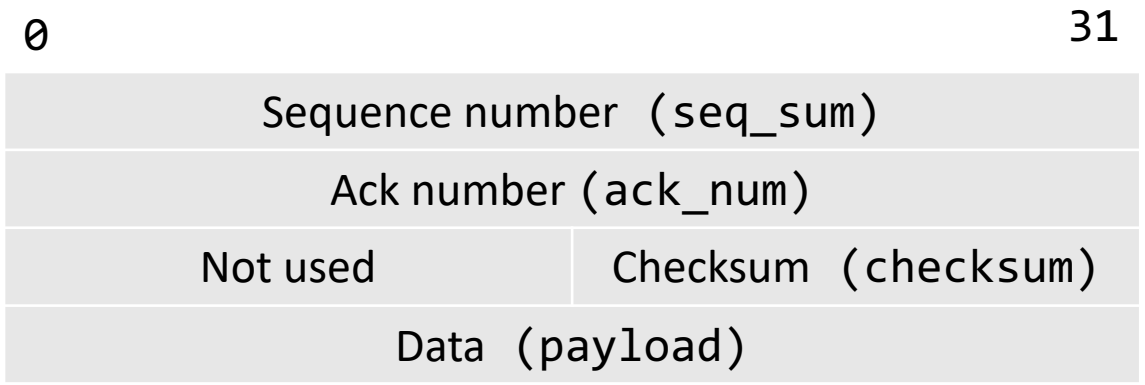
```python
def check():
    true_pkt_buffer = _collect_pkt(sent_file_name)
    if len(true_pkt_buffer) != len(rcv_pkt_buffer):
        print('Fail')
        return
    for i in range(len(true_pkt_buffer)):
        if true_pkt_buffer[i].payload != rcv_pkt_buffer[i].payload:
            print('Fail')
            return
    print('Pass')


def _collect_pkt(file_name, payload_len=512):
    ...

file_name = 'recv.txt'
sent_file_name = 'doc1.txt'
ip = 'localhost'
port = 8080
payload_len = 512
rcv_pkt_buffer = []

if __name__ == '__main__’:
    parse_args(sys.argv[1:])
    f = open(file_name, 'wb+')
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((ip, port))
    try:
        with f, sock:
            recv(sock)
    except KeyboardInterrupt:
        check()
        exit(0)
```

Task description – `packet.py`




```python
class Packet:
    def __init__(self, payload=b"", seq_num=0, ack_num=0):
        self.seq_num = seq_num
        self.ack_num = ack_num
        self.payload = payload
        self.chk_sum = 0

    def encode(self):
        return b"".

    def decode(self, packet):
        return self

    def compute_checksum(self):
        return 0

    def __str__(self):
        return f"{self.seq_num} {self.ack_num} {self.chk_sum} \n
{self.payload.decode()}"
```

Requirements:

- `encode()`: Encode the packet into bytes following the above format.
- `decode()`: Decode the packet from bytes.
- `compute_checksum()`: Compute the checksum of this packet.

Task description – `sender.py`

Requirement:

- Read data from the file-to-send.
- Segment the data into packets and store the packets into a packet buffer.
- Use `rdt.send()` to send all the packets.

```
import rdt

def parse_args(argv):
    ...

payload_len = 512
ip_addr = 'localhost'
port = 8080
file_name = 'doc2.txt'

if __name__ == '__main__':
    parse_args(sys.argv[1:])
    snd_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    snd_socket.bind(('localhost', 0))
    rdt.pkt_buffer = []
    rdt.send(snd_socket, (ip_addr, port))
    snd_socket.close()
```

Task description – `rdt.py`

Requirements:

- `send()` implements the RDT protocol.

```python
import udt

pkt_buffer = []


def send(sock, rcv_addr):
    global pkt_buffer

    for pkt in pkt_buffer:
        udt.send(sock, rcv_addr, pkt)
```

How to run the program

- Start the receiver:
  - `python receiver.py -f <file-to-wirte name> -s <file-sent name> -p <payload_len>`
- Start the sender:
  - `python sender.py -f <file-to-sent name> -i <recv_ip> -p <recv_port> -p <payload_len>`
- Stop the receiver:
  - `Ctrl+C` and the program will print “Pass” or “Fail” indicating whether the file is correctly received

Notes

- You are free to ignore the provided code and build the project from scratch, or factorize the given code (except for `udt.py`).
- In that case, you need to submit a README to tell the TAs how to test your code
- However, your design MUST use `udt.py` to send/receive data and you MUST NOT modify `udt.py` because we will integrate our test cases in `udt.py` to test your code.

Grading scheme

1. Runnable: 20%
2. Pipelining: 20%
3. Test cases: 60%
   - Handling corrupted packet: 20%
   - Handling packet loss: 20%
   - Handling out-of-order packet: 20%
4. Design document (optional): If some test cases are failed, TAs will check this document and your code to see whether your design logic is correct.
5. README (optional): If you refactorize the given code or you build your project from scratch, you need to tell the TAs how to test your code.

udt.py

```python
from packet import Packet


def send(sock, addr, packet):
    print('<- Send packet', packet.seq_num)
    sock.sendto(packet.encode(), addr)


def recv(sock):
    pkt_byte, addr = sock.recvfrom(1024)
    packet = Packet().decode(pkt_byte)
    print('-> Receive ACK', packet.ack_num)
    return packet, addr
```

Where/what/how to turn in

- Where
  - Submissions should be made through Canvas under Assignments-Project 2

- What
  - If you don't change anything in `receiver.py`, and you follow the provided interfaces
    - ✓ Submit `rdt.py`, `sender.py` and `packet.py`
  - If you modify `receiver.py`
    - ✓ Submit `rdt.py`, `sender.py`, `packet.py`, `receiver.py` and a README
  - If you don't use the provided interfaces
    - ✓ Submit `rdt.py`, `sender.py`, `packet.py` and a README

- How
  - Pack your code and documents into one zip file, name it as *<project2-your_name-your_ust_id>.zip* and upload it to Canvas.