Project #2

A Reliable File Transfer Service over TCP and UDP

2023.11.01

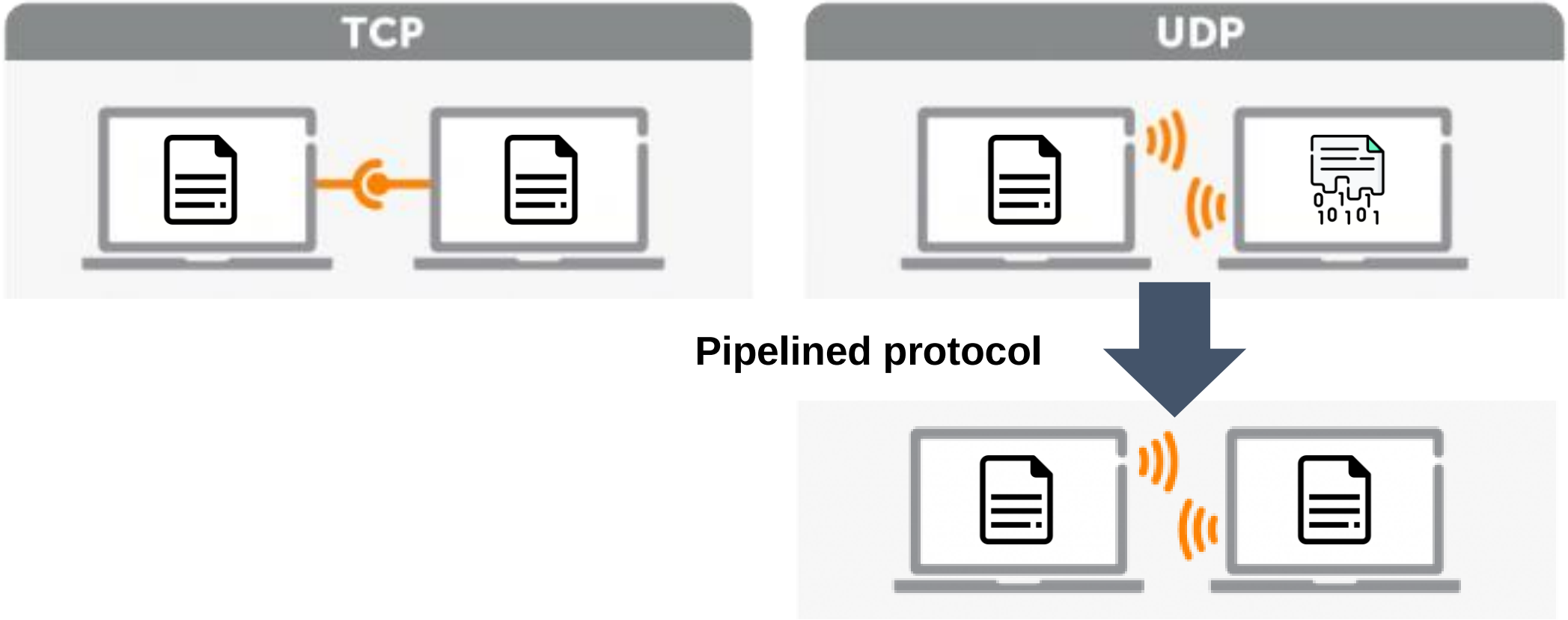
Review of TCP and UDP

- TCP (Transmission Control Protocol)
  - Connection with 3-way handshake
  - Provides reliable data transfer
  - Slower than UDP

- UDP (User Datagram Protocol)
  - Sends data without connection
  - Provides unreliable data transfer
  - Faster than TCP

Project #2

- A reliable File transfer service over TCP and UDP



Project #2

- tcp_file_send()
- tcp_file_receive()

```
def tcp_file_send(self, filename: str, tcp_send_func: Callable)-> None:
    basename = os.path.basename(filename)
    self.file_pointer = open(filename, "rb")

    # packet의 파일 이름(basename)을 전송한다.
    #
    # todo
    #
    # 이름 전송 종료

    # 파일을 구성하는 data를 전송한다.
    # tcp_file_data_packet이 생성하는 packet을 tcp를 이용해 전부 전송한다.
    #
    # todo
    #
    # 파일 data 전송 종료

    # TCP_FILE_TRANSFER_END을 전송하여
    # 파일의 전송이 끝냈음을 알린다.
    #
    # todo
    #
    # TCP_FILE_TRANSFER_END을 전송 종료


    # 파일 닫기
    self.file_pointer.close()
    self.file_pointer = None
```

```
def tcp_file_receive(self, packet) -> int:
    packet_type, data = self.tcp_packet_unpack(packet)


    if packet_type == PACKET_TYPE_FILE_START:
        basename = data.decode(ENCODING)
        self.file_name = basename
        file_path = './downloads/(tcp) '+basename
        # 파일의 이름을 받아 file_path 위치에 self.file_pointer를 생성한다.
        #
        # todo
        #
        return 0

    elif packet_type == PACKET_TYPE_FILE_DATA:
        # self.file_pointer에 전송 받은 data를 저장한다.
        # todo
        #
        return 1

    elif packet_type == PACKET_TYPE_FILE_END:
        # 파일 전송이 끝난 것을 확인하고 file_pointer를 종료한다.
        #
        # todo
        #
        return 2
```

Project #2

- udp_send_with_record()
- udp_file_send()
- udp_file_receive()
- udp_pipeline()

```python
def udp_send_with_record(self, packet_type: bytes, data: bytes, udp_send_func: Callable) -> None:
    packet = self.udp_packet_pack(packet_type, self.udp_last_ack_num, data)
    udp_send_func(packet)
    # GBN, SR을 통
    # 또한 self.u
    #
    # todo
    #
```

```python
def udp_file_send(self, filename: str, udp_send_func: Callable) -> None:
    basename = os.path.basename(filename)
    self.file_pointer = open(filename, "rb")
    # udp를 통해 파일의 basename을 전송하고 ack를 기다린다.
    # hint : self.udp_file_name_transfer 함수를 활용할 것
    #
    # todo
    #

    data_ready, data = self.udp_file_data()
    while data_ready:
        if len(self.udp_send_packet) < UDP_WINDOW_SIZE: #window의 크기보다 전송한 패킷의 양의 적은 경우
            #
            # todo
            #

            data_ready, data = self.udp_file_data() # 다음 전송할 data를 준비한다.

        else:
            # PIPELINE을 위한 window를 전체를 사용하여 ack를 기다리며 timeout에 대처한다.
            #
            # todo
            #
            pass
    # 모든 파일 data의 ack를 기다리고 timeout에 대처한다.
    #
    # todo
    #

    # 파일 전송이 완료되었음을 알리고 ack에 대비한다.
    #
    # todo
    #

    # 파일 포인터를 제거한다.
    self.file_pointer.close()
    self.file_pointer = None
```

```python
def udp_file_receive(self, packet: bytes, udp_send_func: Callable) -> int:
    ack_bytes = self.udp_ack_bytes(packet)
    packet_type, ack_num, data = self.udp_packet_unpack(packet)

    if packet_type != PACKET_TYPE_FILE_ACK:
        # 받은 packet에 대한 ack를 전송한다.
        #
        # todo
        #
        pass

    if packet_type == PACKET_TYPE_FILE_START:  # file transfer start
        if self.file_pointer is not None:
            self.file_pointer.close()
        basename = data.decode(ENCODING)
        self.file_name = basename
        file_path = './downloads/(udp) '+basename
        # 파일의 이름을 받아 file_path 위치에 self.file_pointer를 생성하고,
        # 그다음 받을 파일의 data의 시작 packet의 ack_num를 self.file_packet_start에 저장하여
        # 연속된 packet을 받을 수 있게 준비한다.
        #
        # todo
        #
        return 0

    elif packet_type == PACKET_TYPE_FILE_DATA:  # file transfer
        if not self.udp_recv_flag[ack_num]:
            # 처음 받은 packet인지 확인하고
            # 처음 받는 packet이라면 self.udp_recv_packet[ack_num]에 저장하고
            # self.udp_recv_flag[ack_num]에서 확인할 수 있게 표시한다.
            #
            # todo
            #
            pass

        # self.udp_recv_packet에 self.file_packet_start에서 부터 연속된
        # 패킷이 저장되어 있다면 이를 self.file_pointer를 이용해 파일로 저장하고
        # self.udp_recv_flag를 update한다.
        # 또한 self.file_packet_start 역시 update한다.
        #
        # todo
        #
        return 1


    elif packet_type == PACKET_TYPE_FILE_END:  # file transfer end
        # 파일 전송이 끝난 것을 확인하고 파일을 종료한다.
        if self.file_pointer is not None:
            self.file_pointer.close()
            self.file_pointer = None
        return 2

    elif packet_type == PACKET_TYPE_FILE_ACK:  # ack
        # GBN, SR을 위해 self.udp_a
        # hint: self.udp_ack_num으로
        # window를 옮겨준다 (self.u
        #
        # todo
        #
        return 1
    return 1
```

```python
def udp_pipeline(self, udp_send_func: Callable) -> None:
    # GBN, SR 중 하나의 알고리즘을 선택하여 ACK를 관리한다.
    # hint: self.udp_send_packet[ack_num]에 저장시
    # (send time, packet)형태로 저장할 것
    #
    #
    pass
```

Project #2 유의사항

- 동일 폴더 내 pj_1.py 위치해야 함

```
def tcp_file_send(self, filename: str, tcp_send_func: Callable) -> None:
    basename = os.path.basename(filename)
    self.file_pointer = open(filename, "rb")
```

```
def udp_file_send(self, filename: str, udp_send_func: Callable) -> None:
    basename = os.path.basename(filename)
```

- pj_1.py에서 구현한 tcp와 udp의 recv/recvfrom 함수의 parameter는 다음으로 고정
  - APP_HEADER_LEN+PACKET_SIZE
  - config.py에 값 명시되어 있음

```
def tcp_recv(self) -> bytes:
    # use self.tcp_socket
    data = self.tcp_socket.recv(APP_HEADER_LEN+PACKET_SIZE)
    return data

def udp_recv(self) -> bytes:
    # use self.udp_resv_socket
    data, _ = self.udp_socket.recvfrom(APP_HEADER_LEN+PACKET_SIZE)
    return data
```

제출방법

- 팀원 중 한 명만 제출
- 제출 파일
  - Project 압축 파일 (파일명: 팀이름_zip)
  - 보고서 (파일명: 팀이름_pdf)
    - 구현 환경: OS 정보, Python version
    - 구현 코드: 사용한 pipelined protocol, 스크린샷/텍스트 첨부 가능, 구현 코드 설명
    - 정상 동작 스크린샷(server/client 실행 창), UDP 통신으로 file 전송 시 data loss 일어나지 않는 것 보여주어야 함
- 제출 마감
  - 11월 21일(화) 23시 59분
  - 단, 질문은 20일(월) 18시까지만 가능