z5361987
Anson Lai

COMP3331 Assignment Report

This report describes the implementation of the STP protocol using Python 3. The implementation consists of three Python files: sender.py, receiver.py, and segment.py.

The sender will send segments to the receiver and wait for an acknowledgement segment from the receiver, similar to the TCP protocol. In the sender, a sliding window is implemented so that it will send files in a pipelined manner. The receiver will simulate packet loss and ACK loss by using the random built-in library. The segment.py file is created to handle the segment creation according to the specified segment format. It will handle the pack and unpack of segments from the type and data header.

The listen() function is multithreading in the sender, which is looping and trying to receive incoming ACK segments from the receiver.

If the sender sends a segment but does not get the ACK segment before a timeout, it should retransmit the segment again until the corresponding ACK segment is received. Therefore, a buffer is maintained to store all the segments. In the sender class, a data structure is implemented as follows to store each segment sent. Whenever the corresponding ACK segment of the data segment is received, the segment ack_received boolean is set to true:

```
@dataclass
class Segment:
        type: int
        seqno: int
        data: bytes
        exp_ack: int
        packet: bytes
        ack_received: bool = False
```

When the ACK is received by the sender, it will update the corresponding segment in the buffer to received, and the sliding window is moved forward to create room for sending new segment. When the ACK timeout, it will check the buffer of segments to look for the oldest unacknowledged segment and retransmit it to the receiver.

The receiver has a data structure of struct implemented to represent the received segment stored in a buffer. It is used to store the received segment from the sender. Only the original segments will be stored in the buffer, and the duplicate ones are ignored. Also, it will insert the segments in the correct order using the segment's sequence number. The struct is as follows:

```
@dataclass
class Buffer_obj:
        seqno: int
        exp_next_seqno: int
        data: bytes
```