Simple Transport Protocol Report

- Author: Jack Jiang
- Date: Oct, 2018
- For: 2018S2 COMP9331 Assignment

List of File and Classes

report.pdf

- this file

sender.py

- RTTEstimator: a class for timeout estimation
- Sender: sender finite state machine

receiver.py

- Receiver: sender finite state machine

scp.py

- ScpMath: checksum calculation and other helper functions
- ScpPackage: SCP package abstraction
- ScpLogger: SCP logger class

Acknowlegement

Function bytes_to_int

- Location: scp.py -> Class ScpMath -> bytes_to_int
- From: https://coderwall.com/p/x6xtxq/convert-bytes-to-int-or-int-to-bytes-in-python
- Origin Author: Luã de Souza
- Modification: No

Function int_to_bytes

- Location: scp.py -> Class ScpMath -> int_to_bytes
- From: https://coderwall.com/p/x6xtxq/convert-bytes-to-int-or-int-to-bytes-in-python
- Origin Author: Luã de Souza
- Modification: by Jack Jiang

List of Feature

1. A three-way handshake
2. a four-segment connection termination
3. sequence number

4. acknowledgement
5. checksum
6. pipline sending with a sender buffer
7. timer with a rount-trip-time estimation
8. fast retransmit
9. PLD module with the ability to drop, dulplicate, corrupt, reorder or delay packages

STP Header Defination

No	**segment**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	sequence								
1	sequence								
2	sequence								
3	sequence								
4	acknowledge								
5	acknowledge								
6	acknowledge								
7	acknowledge								
8	flag	000	000	000	ACK	000	000	SYN	FIN
9	flag	000	000	000	000	000	000	000	000
10	window	000	000	000	000	000	000	000	000
11	window	000	000	000	000	000	000	000	000
12	checksum								
13	checksum								

P.S. All 000 field means reserved for further usage

Trade-off and improvements

Fixed Receiver Buffer

The receiver buffer is fixed to 4096, which means we can not send packages greater than that. The improvement would be make use of the reserved window field in header to determine the buffer size of the receiver.

No reordered package buffer

Currently we only allowed one reordered package at the same time. If there is a reordered package which has not been sent, the reorder function in PLD module will be temporarily disabled. I think this could not reflect the real scenario. The improvement would be use a queue to buffer the reordered packages.

No delayed package buffer

similar to above

Questions

Question a

- receiver sequency number (pDrop = 0.1)

`python3 sender.py localhost 1234 test0.pdf 500 100 4 0.1 0 0 0 0 0 0 100`

```
0    0    0  100  300  400  500  600  200  300  400  500  600  700  800
900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900 2000 2100 2200 2300
2400 2500 2600 2800 2900 3000 2600 2700 2800 2900 3000 2700 2800 2900 3000
2700 2800 2900 3000 2700 2800 2900 3000 2700 2800 2900 3000 2700 2800 2900
3000 2700 2800 2900 3000 2800 2900 3000 2900 3000 3000 3000
```

- receiver sequency number (pDrop = 0.3)

`python3 sender.py localhost 1234 test0.pdf 500 100 4 0.3 0 0 0 0 0 0 100`

```
0     0     0    100   300   600   200   300   400   500   600   700   800
1000  1200  1300   900  1000  1100  1200  1300  1400  1600  1700  1800
1900  1500  1600  1700  1800  1900  2000  2100  2400  2600  2200  2300
2700  2400  2500  2600  2800  2900  2600  2700  2800  2900  3000  2700
2800  2900  3000  2700  2800  2900  3000  2700  2800  2900  3000  2700
2800  2900  3000  2700  2800  2900  3000  2800  2900  3000  2900  3000
3000  3000
```

- Answer a

the package drop occurs when the sequency number not grow by MSS, for example: the package number should be 100, 200, 300; but the real case are 100, 300, 400; therefore, the package with sequnecy number 200 is dropped.

Question b c

it just takes too long.

Testing

Receiver

`python3 receiver.py 1234 received_file.pdf`

Sender

`ip port file MWS MSS gamma pDrop pDuplicate pCorrupt pOrder maxOrder pDelay maxDelay seed`

1. stop and wait without PLD

`python3 sender.py localhost 1234 test0.pdf 512 512 4 0 0 0 0 0 0 0 0`

2. stop and wait with pDrop

`python3 sender.py localhost 1234 test0.pdf 512 512 4 0.2 0 0 0 0 0 0 0`

3. pipline without PLD

`python3 sender.py localhost 1234 test0.pdf 2048 512 4 0 0 0 0 0 0 0 0`

4. pipline with pDrop

`python3 sender.py localhost 1234 test0.pdf 2048 512 4 0.2 0 0 0 0 0 0 0`

5. pipline with pDuplicate

`python3 sender.py localhost 1234 test0.pdf 2048 512 4 0 0.2 0 0 0 0 0 0`

6. pipline with pCorrupt

`python3 sender.py localhost 1234 test0.pdf 2048 512 4 0 0 0.2 0 0 0 0 0`

7. pipline with pOrder

`python3 sender.py localhost 1234 test0.pdf 2048 512 4 0 0 0 0.4 2 0 0 0`

8. pipline with pDelay

`python3 sender.py localhost 1234 test0.pdf 2048 512 4 0 0 0 0 0 0.3 1 0`