Reliable Transfer over Unreliable Channel

凌康伟 5140219295

April 24, 2017

1 Usage

Platform: Ubuntu 14.04, python 3.6

Start a receiver firstly, and then start the sender.

1. Receiver

```
$ python3 receiver.py <protocol> <port> [prob_loss]
```

2. Sender

```
$ python3 sender.py <protocol> <remotehost> <port> <file_to_transfer>
```

`protocol` can be one of the followings.

`udp` transfer using basic UDP socket

`rdt` rdt3.0

`GBN` Go-Back-N (window size: 20)

`tcp` TCP

`prob_loss` is the probability of dropping packet manually (for simulation), defaults to 0.

Below is an example to transfer a file via GBN.

1. start the receiver.

```
$ python3 receiver.py GBN localhost 12345
```

2. start the sender.

```
$ python3 sender.py GBN localhost 12345 files/lecture4.pdf
```

3. view results.

```
# at sender side
$ python3 ...
start to send file: files/lecture4.pdf

Took 0.15s to transfer.
```

```
# at receiver side
$ python3 ...
Successfully setup socket at ('59.78.15.253', 12345)
waiting for file
acks: 257, NAK: 0.
last packet received.
Received: lecture4.pdf.
```

2 High-level structure

`sender.py`, `receiver.py` entrance of send and receive process.

`transfer.py` definitions of different protocols used by sender and receiver.

`packet.py` functions of making packet and parsing packet from raw data.

`utils.py` definitions of utility functions (checksum).

`files/` some files I used to test the transfer protocols

`saved/` place to store saved files

2.1 protocols

Each protocol inherits from the `BaseTransfer` class, which implements the common procedure `sendFile` and `recvFile` (timing, filename exchange). Each protocol only needs to implement the `send` and `recv` method. I utilized python genertor to simulate a stream sender/receiver.

3 Test

For testing, a 12MB pdf file is used as the test file for transfer. I found out that when packet size is small (< 8192), tcp is much faster than udp or other protocols I've implemented. This is because TCP has its own mechanism to pack up small packets, resulting less overhead. Especially when running on OSX system, due to some OS regulations, the speed of udp (and rdt, GBN implemented over udp) is greatly cut down.