CN HW2

b04705006 王馨儀

Environment

ubuntu 16.04

Language

C, with basic library: unistd, sys, signal...

Usage

command

`make:` build agent, sender, receiver, sender(bonus), receiver(bonus)

`make run:` run agent(default port 3001), sender(3000), receiver(3002) at the same time

**modify the source and destination file name here

`make run_b:` run 3 agent with port 3001, 3002, 3003, sender(port 3000), receiver(3004)

**modify the source and destination file name here

`make clean:` clean all binaries

log of agent, sender, receiver will be written to according .log files

Argument

```
./sender <sender port> <agent port> <source file>
./agent <agent port> <sender port> <receiver port> <loss rate(0-10)>
./receiver <receiver port> <agent port> <target file>
(change arguments directly in Makefile)
```

Implement detail

Sender

0. Ack data structure: id(#number), type(ack or finack)

0. Data data structure: id(#number), type(data or fin), payload(char*), payload_size(int)

1. Initialize

set up fd for udp, file pointer for source....

sender socket and agent socket with given port and default IP(127.0.0.1), using structure `sockaddr_in`

set up some variables, e.g. threshold, window size…

set up alarm signal for detecting timeout(1s).

2. While loop, use `recv()` to listen from agent:

If Data timeout, SIGALRM would interrupt recv with errno = EINTR:

- calculate first unacked data.
- Update new window size and new threshold.
- mark datas in new window as unacked.
- resend datas.

If Finack timeout:

- resend Fin.

If receive an Ack:

- mark the data with same id as acked.
- check* if datas in current windows is all acked. If yes:
  - stop alarm.
  - If this is the last window(all datas are sent and receive Acks):
    - send fin.
  - Else:
    - update new window size, and new threshold.
    - mark datas in new window as unacked.
    - send next datas.
- *keep a variable for # of acked data(add up only when acked for first time). If this is equal to window size then we know it’s all acked.

If receive an Finack:

- break loop, end program.

3. (re)Send data:

- send datas according to current window size, through `sendto()`.
- set `alarm(1)` after sending all datas.
- read 1KB from source file. If length is less then 1KB, we know sending data is finished. Ready to send Fin.

4. Send fin:
    If receive Ack from last data
        send Fin.

Agent

0. Ack, Data data structure is same as Sender
1. Initialize
set up fd for udp
set up sender, agent, receiver socket using `sockaddr_in`
2. While loop, use `recv()` to listen from sender or receiver:
If from sender(check with port number):
    random a number. If it's lower than the given loss rate: drop data. Else: send to receiver.
If from receiver:
    send to sender.
    If the ack type is Finack:
        break loop, end program.

Receiver

0. Ack, Data data structure is same as Sender
1. Initialize
much like what sender does
2. While loop, use `recv()` to listen from agent:
If receive Data within current buffer range:
    If first receive: write this data to array. Else: ignore.
    Send Ack.
If receive Data prior to current buffer range:
    Send Ack.
If receive Data after than current buffer range:
    Check if current buffer is full. If yes:
        flush to target file.
        Update buffer range.
        write this data to array.
    Else:
        Drop.
If receive Fin:
    Send Finack.
    break loop, end program.

Bonus-Sender

from 1 agent to 3 agent.
Send by Module 3: send to agent[data.id%3].

Bonus-Receiver

from 1 agent to 3 agent.
Send by Module 3: send to agent[ack.id%3].

Difficulties

1. At first didn't realize it's possible that receiver can receive data with id prior to buffer range. This would cause core dump in receiver(index out of range).
Case:
Sender->[...30,31,32]->Agent(drop 30,31)->Receiver: ...29,X,X,32 [sender timeout because of 30,31]
Sender->[30,31,32,33]->Agent(drop 32)->Receiver: (write 1~32) 33,X... [sender timeout because of 32]
Sender->[32,33]->Agent->Receiver: [receive 32, prior than current buffer range 33~64]

2. Many conditions in receiver: when to send Fin, what to do when timeout on waiting for Finack, mark send or resend when logging.