Robust Protocol Challenge memo

What the example code is doing

common

- creates a single socket and uses it.
- communicating with UDP

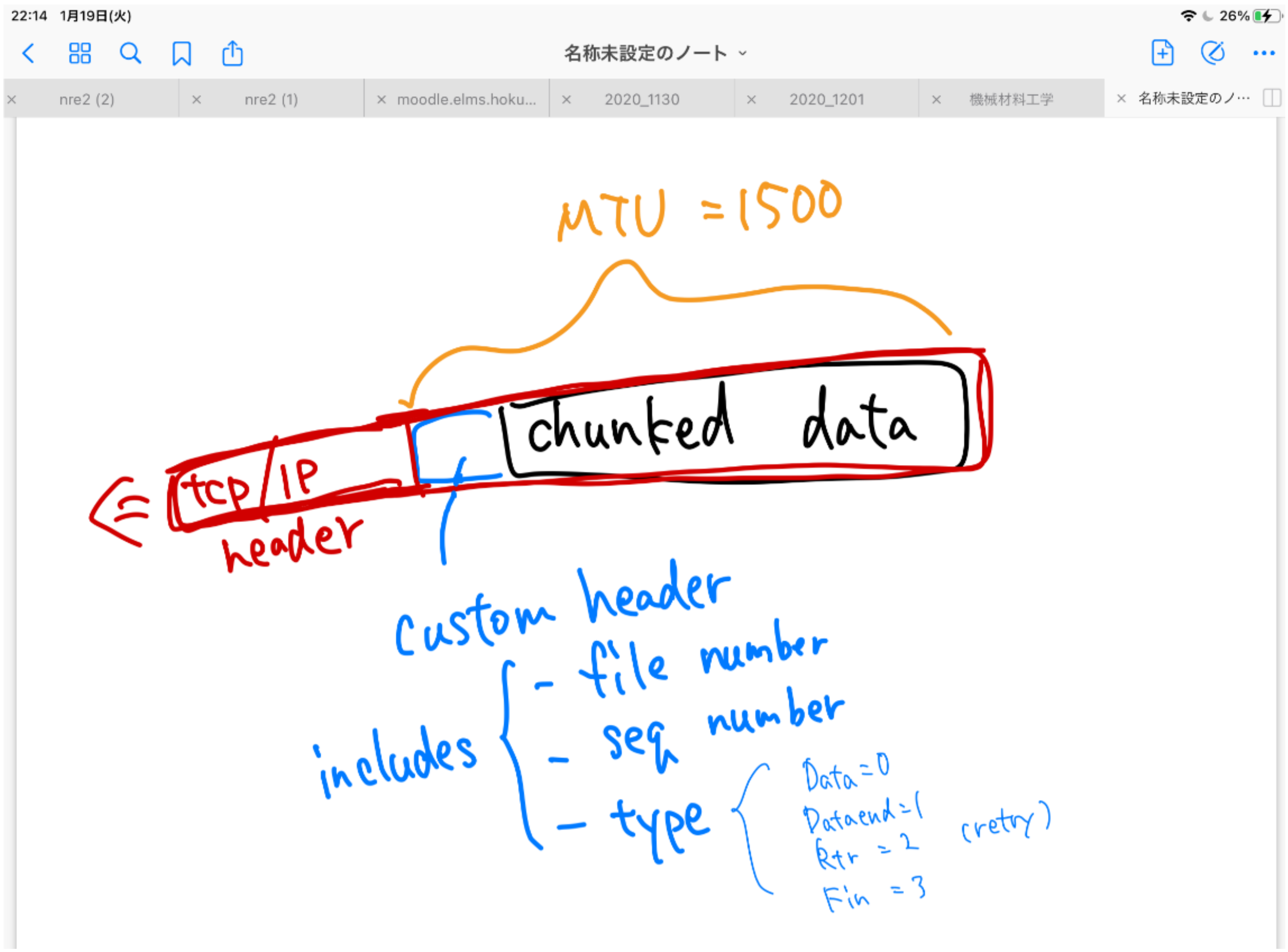
sender

- At first splits the file into MTU-sized pieces and adds custom header to each piece.
    - The custom header consists of data type, file number, and sequence number.
- There is a queue where [retransmission requests or file reception completion notification] arrive.
- While sending prepared pieces, if a retransmission request or reception completion comes to the queue, it will process that.

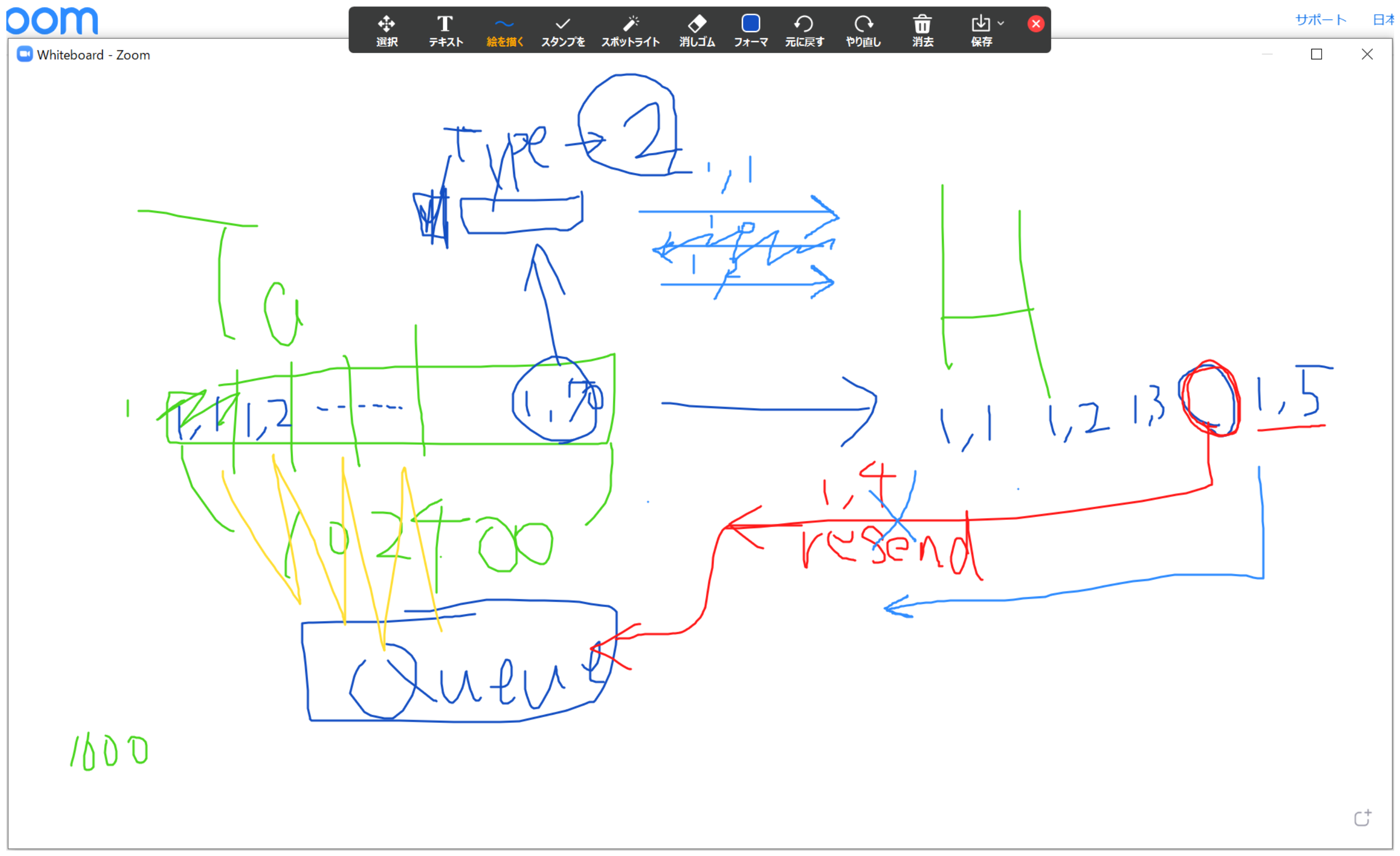
- loop the above until the time comes.

receiver

- manages data with three dicts
    - `recieved_files_flag = {0: False, 1: False, ...}` manages whether files have been received completely
    - `received_files_length = {0: Null, 1: 60, 2: 61, ...}` figures out the number of the seq and keep it.
    - `recieved_files_data = {0: [b'<byte chunk>', b'...', ...(×100)], 1: [...], ...}` stores data which received
- When there is no seq number less than the received seq number, request retransmission of it.
- Once all the seqs are in place, remove the custom headers and assembles the file



- we talked about what the example code is doing(1/19 night)



Idea or Problems or Questions? (just write it down)

- logging out how many resend requests or any other is happening
- Sometimes a jammer will lose the entire packet. What happens if some of them are lost?
  - how long does `echo 1` or `echo 0` take ??

```
root@Taro:/home/pi/team05/githubsample/robust# time echo 1 > /sys/class/gpio/gpio1

real    0m0.000s
user    0m0.000s
sys     0m0.000s
```

```
pi@Taro:~/team05/githubsample/robust $ time sudo echo 1 > /sys/class/gpio/gpio17/v

real    0m0.062s
user    0m0.020s
sys     0m0.044s
```

```
pi@Taro:~/team05/githubsample/robust $ time echo 1 > /sys/class/gpio/gpio17/value

real    0m0.001s
user    0m0.000s
sys     0m0.000s
```

> I think we can reference https://github.com/pratiklotia/Client-Server-Fast-File-Transfer-using-UDP-on-Python (https://github.com/pratiklotia/Client-Server-Fast-File-Transfer-using-UDP-on-Python)
> — ching

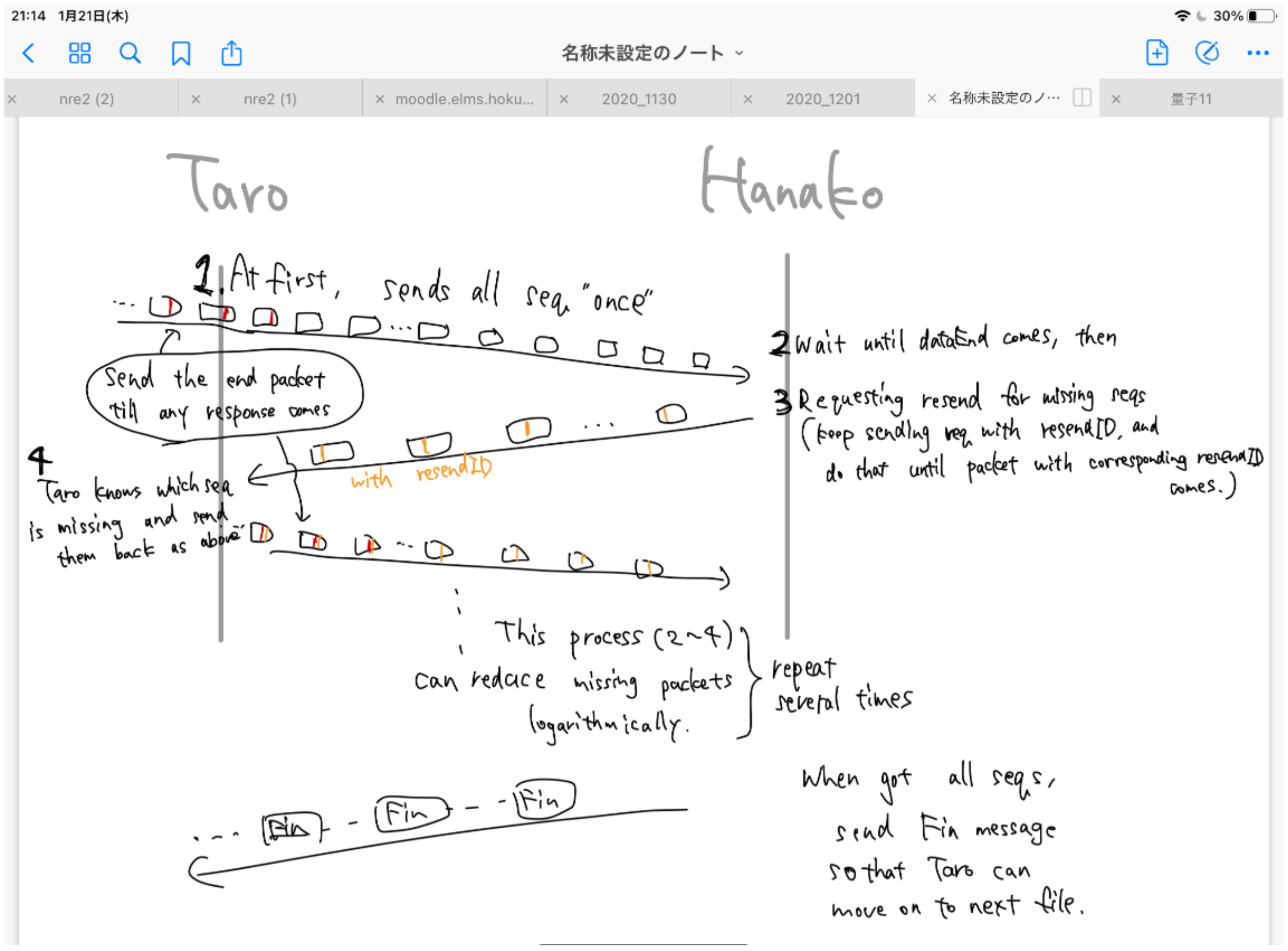
Related articles (I hope so)

- Python Socket Receive Large Amount of Data https://stackoverflow.com/questions/17667903/python-socket-receive-large-amount-of-data (https://stackoverflow.com/questions/17667903/python-socket-receive-large-amount-of-data)
- Python socket sends faster than receiver https://stackoverflow.com/questions/44945324/python-socket-sends-faster-than-receiver (https://stackoverflow.com/questions/44945324/python-socket-sends-faster-than-receiver)
- Fastest way to process and save UDP flow in python https://stackoverflow.com/questions/23660631/fastest-way-to-process-and-save-udp-flow-in-python (https://stackoverflow.com/questions/23660631/fastest-way-to-process-and-save-udp-flow-in-python)

1/20

- Perform single-threaded and multi-threaded measurements
  - there was almost no difference
- Count how many requests for retransmissions were made, and how many completion reports were made.
  - we gathered some logs and found out that there were so many meaningless resend request

- one idea from Hiro


21:14 1月21日(木)
30%
名称未設定のノート
nre2 (2)
nre2 (1)
moodle.elms.hoku...
2020_1130
2020_1201
名称未設定のノ…
量子11
Taro
Hanako
1. At first, sends all seq "once"
Send the end packet till any response comes
2 Wait until dataEnd comes, then
3 Requesting resend for missing seqs (keep sending req with resendID, and do that until packet with corresponding resendID comes.)
with resendID
4 Taro knows which seq is missing and send them back as above
This process (2~4) can reduce missing packets logarithmically.
repeat several times
Fin
When got all seqs, send Fin message so that Taro can move on to next file.

- 1/21 we discussed about the idea
    - the one above
    - BCH error correction
        - If we assume that all missing packets' bits are 0, then the bits that are erroneous in that packet are half of the packet. In total, 50% of the packets received plus 50% of missing packets are correct, so we only need to correct 25% of the errors "on average".
    - think_outside_the_box
- Hiro is now making progress implementing my own idea and maybe we can try it in evening meeting