Development notes

Transport

TODO:

- implement threaded sending and receiving
- research weather
- implement packet class
- figure out packet timers

Flow control

OBS: modulo af SEQ OBS: add new packets to window on ACK OBS: when adding to buffer, check if within window OBS: handle holes in buffer when passing buffer to oSHIT OBS: better timers in python - callback methods? timers? OBS: enlargen timeouts when in recovery mode

Receiving

- Packet arrives
- (optionally) Decrypt payload ? verify checksum
- Check flags
- `if ACK:`
  - [DONE] Everything in `txwindow` up to packet with `SEQ` is good and can be removed
  - [DONE] Delete every packet with no. less than `SEQ` from `txwindow`
  - Ask for new `Packet` from Logic
- `if NACK:`
  - Re-transmit packet with SEQ
  - Prepend packet with SEQ to `txqueue`
- `if !(ACK || NACK):`
  - Send ACK
  - Pass payload on to business logic
  - `if SEQ > lastreceived + 1:`
    - `if len(lastreceived) == 0:`
      - send `NACK`
    - insert in `rxbuffer`
  - `else:`
    - `lastreceived = SEQ`

  - if packet with no. `lastreceived + 1` in `rxbuffer`:
    * pass rxbuffer to businesslogic

Sending

Threaded loop: - pop Packet from `txqueue` - add timeout time to Packet - transmit packet - `for packet in window:` - `if currenttime > packet.timeout:` - append packet to `txqueue` - `if len(window) < WSIZE:` - construct new packet - append to window - append to `txqueue`

Threading

Only the order they're passed to Transport needs to be sequential Transport will always take the first element from a list So the list in Incoming needs to be appended sequentially

Example: - rxthread receives packet - rxthread appends to rxqueue - rxthread starts packethandler thread - packethandler acquires rxqueue lock - packethandler does its shit - maybe passes the packet on to application layer - packethandler releases the rxqueue lock

Thread Locks and Condition objects

- `rxlock`
  - Incoming notify Transport
  - used by `Incoming` to notify `Transport` of new `Packet`s received
- `txlock`
  - Transport notify Outgoing
  - used by `Transport` to notify `Outgoing` of new `Packet`s to send
-

Communicating with Logic

A way to communicate threaded between Transport and Logic

- Tx:
  - Logic programmer should call `self.send()` with new `Packet`s
  - New items should be added to `txwindow` and `txqueue` when `len(txwindow) < self.WSIZE`
  - `Packet`s should not be produced all at once, for memory
  - *Solutions:*
    * Have threaded loop in `Logic` parent class to run a `self.get_next_packet()` when theres room in `txwindow`

            - Pro: could also have interaction with `Transport` wrapped in `Logic` parent
            - Con: one more class, one more thread
        * `in_ack()` calls `Logic.make_packet()`
            - when removing packet from `txwindow`,
            - `get_new_packet()` does `notify()` to threaded loop
            - `Logic`s thread loop makes `Packet`s until `len(txwindow) == self.WSIZE`
- Rx:
    - `Packet`s should be read sequentially from `rxqueue`
    - *Solutions:*
        * threaded loop in `Logic` parent class, to wait for .notify() from `in_payload()`
            - CON: need to share `Condition` object
        * `in_payload()` calls `Logic.new_packet()` which adds it to a list, and does `notify()` to a threaded loop
            - PRO: only one method call interaction

Logic Solution plan

Two methods in `Logic`:

```
- ‘tr_new_incoming(packet)‘
    - Adds packet to ‘Logic._in‘
    - Call ‘.notify()‘ to alert incoming thread loop
- ‘tr_new_outgoing()‘
    - Call ‘.notify()‘ to alert incoming thread loop
    - Thread loop keeps running until ‘len(txwindow) == Transport.WSIZE‘
```

To threaded loops in `Logic`:

- `_inloop()`
    - sleeps when `self._inbuffer` is empty
    - wakes when notified by `tr_new_incoming()`
    - calls method in subclass until `_inbuffer` is empty again
- `_outloop()`
    - sleeps when `Transport.txwindow < Transport.WSIZE` – rel. TODO: make txgraveyard
    - wakes when notified by `tr_new_outgoing()`
    - calls method in sublcass until `Transport.txwindow == Transport.WSIZE`

Transport architecture

Transport: - works only with complete Packet objects - sending window - receiving window - window bounds tracking - on ACK (`max+=1`) - on transmit (`min+=1`) - track latest received SEQ In: - reads incoming data - makes packet object - decryption - read seq number - read flags - read payload ? passes it up to

Transport SOMEHOW Out: - is notified about new packets in `txqueue` - sends bytes from packet object

`InPacket(data)` - lots more stuff - `if config['crypto']:` self.crypto.decrypt()

OutPacket(data) - created in App layer

How to treat SEQ

`SEQ` in `Packet`: 0-255 N_{max} (Max window size): 0-255