

CSE123: Computer Networks

Spring 2021, Discussion - 3

Project 1b TA - Zesen “Jason” Zhang

Note for Project 1

1. We will grade the project this week and hopefully you will get your grades next week.
2. If you do poorly in project 1, we do not want to make you in double jeopardy. We will ask the students with highest score to share their repo.
3. Project 2 is kind of based on Project 1, if you put enough time on Project 1 including understand the skeleton code as well as implement frames and error checking function properly, in Project 2 you only need to implement the SWP.

Overview

- GOAL: Simulate the Transport Layer network protocol
- 1. Implement communication between two or more hosts
- 2. Hosts implemented as Threads
- 3. 2 types of Hosts: Senders and Receivers
- 4. Practice implementing frame and error handling. In this project, you are going to cut up the message into multiple frames, if the message length exceed the payload length. (MAX_FRAME=64)
- 5. Messages can be corrupted and dropped.
- 6. Implement reliable transmission, Slide-Window Protocol (SWP) at this time. (Lecture 6)

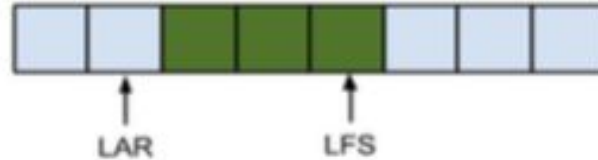
Sliding Window Protocol

- **struct Sender_t**
 - **SWS** – Sliding window size
 - **LAR** (Last Acknowledgement Received) - Sequence number of last acknowledgement received, defines lower bound of the sender window
 - **LFS** (Last Frame Sent)- Sequence number of the last frame sent, defines upper bound of the window
 - **Window is from $[LAR+1, LFS]$** , that is all frames that have been sent but not yet Acked.

Frame Sequence Number in Sender

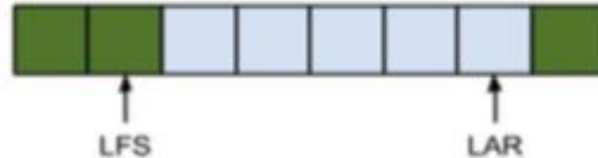
CASE 1: Usual Case
 $LAR \leq LFS$

$LAR \leq LFS \ \&\& \ seqNo > LAR \ \&\& \ seqNo \leq LFS$



CASE 2: Sequence Number Wrap Around
 $LAR > LFS$

$LAR > LFS \ \&\& \ (seqNo > LAR \ || \ seqNo \leq LFS)$



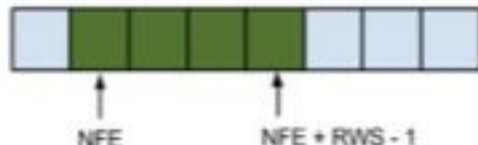
Sliding Window Protocol

- struct Receiver_t
 - **RWS** - Max receiver window size
 - **NFE** - Next Frame Expected
 - **LFR** - Sequence number of largest consecutive frame received
 - **LAF** - Sequence number of largest acceptable frame
 - **LFR** = $NFE - 1$
 - **LAF** = $NFE + RWS - 1$

Frame Sequence Number in Receiver

CASE 1: Usual Case
 $NFE + RWS - 1 \geq NFE$

$NFE + RWS - 1 \geq NFE \ \&\& \ seqNo \geq NFE \ \&\& \ seqNo \leq NFE + RWS - 1$

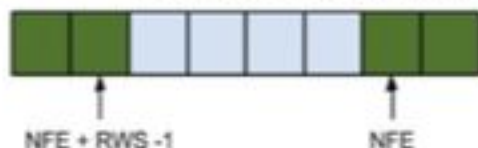


Remember NFE is just $LFR + 1$ and LAF is just $NFE + RWS - 1$.

Green sequence numbers are in window and grey are outside.

CASE 2: Sequence Number Wrap Around
 $NFE + RWS - 1 < NFE$

$NFE + RWS - 1 < NFE \ \&\& \ (seqNo \geq NFE \ || \ seqNo \leq NFE + RWS - 1)$



Receiver with $RWS = 4$, sequence number in $[0, 7]$

Sender Buffer/Window

- Sender need to maintain window(buffer) while sending packets out
- The window is like this:
 - ```
struct sendQ_slot {
 struct timeval* timeout; // event associate with send timeout
 Frame frame;
} sendQ[SWS];
```
- Timeout is of type struct timeval (declared in sys/time.h)
- Index in to the sender buffer using (sequence number % SWS)

# Receiver Buffer/Window

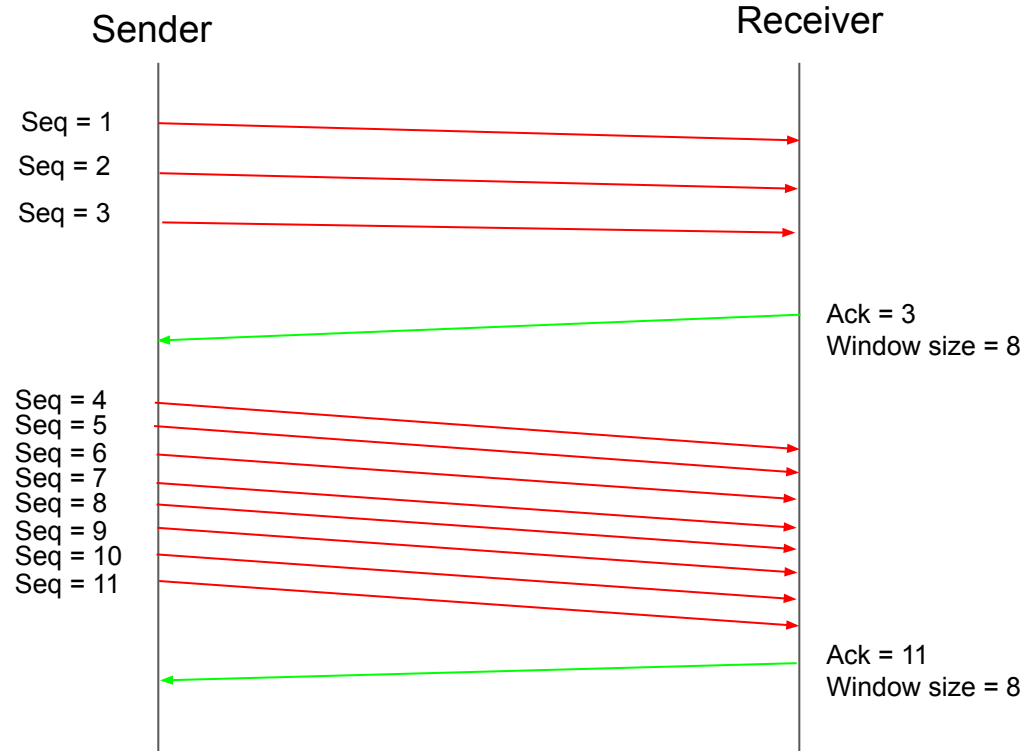
- Similarly, it is better for receiver to maintain a window too.

- Example:

```
struct recvQ_slot {
 struct Frame_t* frame
} recvQ[RWS]
```

- Why don't we need a timeout here ?
- Index in to the receiver buffer using (sequence number % RWS)

# Communications between



## **SWP: Sequence Number Wrap Around**

- You should NOT use more than 8 bits (unsigned char) for seq/ack numbers.
- You need to handle sequence number wrap around once the value reaches 255. Your seq/ack number should wrap back to 0.
- How to do this?
- Answer: % modulus

# Multiple Senders/Receivers

1. Window size could be shortened as it received packet from different senders.
2. The frame in sender buffer could be sent out to different receivers.
- 3.