Announcements

- ## Reading
  - Today: 3.5-3.6
  - Thursday: 4.1 & 4.2

- ## Second Midterm:
  - Tuesday April 15
  - covers material from chapters: 1-3, 5-6
    - emphasis on material since last midterm

- ## Suggested Problems:
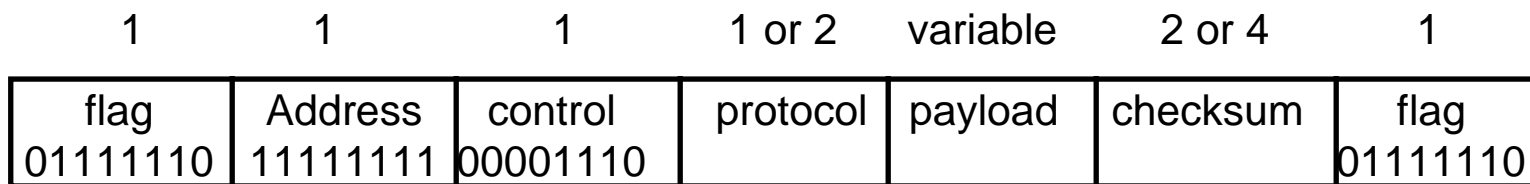 - chapter 3: 1,3, 5, 8, 10, 29
  - chapter 5: 1,3,5,6,7,8,11,17,34,40

PPP Protocol

- **Link Protocol for Serial Lines**
  - Supports multiple network protocols: IP, IPX, CLNP, …
  - designed for dialup or leased lines
- **Link Establishment**
  - configure-request: list of proposed options and values
  - configure-{ack/nack}: will (won't) use the requested option
  - NCP protocol
    - per network level protocol
    - used to establish network attributes (e.g. addresses)

1	1	1	1 or 2	variable	2 or 4	1
flag 01111110	Address 11111111	control 00001110	protocol	payload	checksum	flag 01111110

ATM Datalink Protocol

- **Header**
  - use CRC over the 32 bits of the header
- **How to find cell boundary?**
  - use shifty register to check for valid checksum
    - 1/256 chance of a random match
  - use HUNT mode to increase chances
    - after a good cell, skip to the next cell boundary
    - must receive δ cells with checksum matches
- **Detecting loss of synchronization**
  - one bad cell is probably an error
  - many bad cells is likely a slip (loss of sync)
  - if α bad cells are seen in a row, switch to hunt mode

Project Startup

- **first argument to the application is the node number**
  - a small integer
  - this is passed into AAL_init via argc/argv
- **AAL7_Init**
  - gets id of self
  - calls config_get_info
  - starts at least one network thread
  - returns to main thread
- **UDP setup**
  - similar to project #1: lookup hostname, bind port, …
  - use garbler routines for **send**:
    - garb_sendto_normal
    - garb_sendto_routing

copyright 1997 Jeffrey K. Hollingsworth

Project Synchronization

- ● **Need to coordinate access to shared resources**
    - – use mutex to guard access to a shared data structure
- ● **Queue abstraction is very useful**
    - – enqueue: add item to queue
    - – dequeue: remove item, **block** if not ready
    - – head: return head of queue without dequeue
    - – probe: test if the queue is empty
    - – must use a mutex to protect access to queue
    - – build a producer/consumer test program
- ● **Multiple application threads**
    - – our test application is multi-threaded
    - – must be able to support multiple pending calls into session and network layers

Project Suggestions

- ## KISS
  - get something end-to-end working, wait on optimizations

- ## Use test programs
  - build queue, then test it
  - build timers, then test them

- ## Compiler Issues
  - need to link application -lpthread
  - need to use g++ or cc, not gcc when including pthread.h