Today's Objectives

- TCP Wrap Up
- Services: Buzzword Bingo
- Threads and Synchronization

Review

- What layer is TCP?
  - What does it add that the lower level does not have?
  - How does it add that functionality?

TCP Properties

- Sequence Numbers, Acknowledgements Numbers
  - For ordering data
  - For identifying missing data, duplicate data
- Timeout
  - For lost data
- Sliding Window
  - For flow control, congestion control
  - (not just 8 packets)

TCP Flow Control

- TCP is a sliding window protocol based on byte streams, not packets
  - For window size *n*, can send up to *n* bytes without receiving an acknowledgement
  - When the data is acknowledged, window slides forward
- Each packet advertises a window size inside TCP header field
  - Number indicates number of bytes the receiver is willing to buffer

SERVICES: BUZZWORD BINGO

Perusall

- What works, what doesn't?

Buzzword Bingo

- Document on Box

PROCESSES AND THREADS

What is a Process?

- Process – a sequential program execution
- Ideally, we would like our OS to be capable of running multiple processes/jobs at once (i.e., *multiprogramming*)
- **Challenge**: how to implement & ensure efficient use of system resources?

Difference between a process and a program

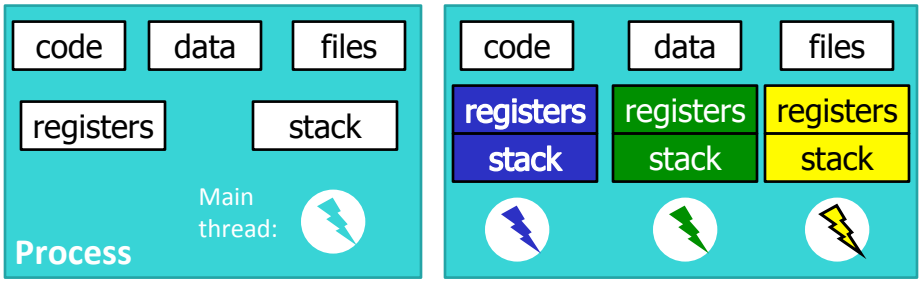
- Baking analogy:
  - Recipe = Program
  - Baker = Processor
  - Ingredients = data
  - Baking the cake = Process

Creating the Illusion of Concurrency

- ***Interleave*** the execution of existing processes to maximize processor utilization
  - Idea: while one process is blocked on (slow) I/O operations, allow another process to have the CPU to make progress
- Provides the illusion of concurrency on a one-processor system
- Provide reasonable response times to processes

Threads and Processes: Execution Environments

- Execution environments
  - address space (memory)
  - thread synchronization and communication resources (sockets)
  - higher-level resources likes open files
- Each process has its own separate execution environment
- Threads ***share*** a single execution environment



Threads vs Processes

If this were an OS class, would discuss threads and processes for weeks

- Your book discusses them briefly

Process

- Single activity that processor can execute
- "Heavyweight"
- Independent tasks
- Have a private address space
- Only interact with other processes through inter-process communication

Thread

- "thread of execution"
- "Lightweight"
- Shares state information with other threads within a single process
  - Every process has a thread
- Easily interact with other threads b/c memory is shared

Threads vs. Processes

- Benefits of using threads instead of processes
  - Low communication/context switching overhead since execution environment/address space is shared
  - Easy to take advantage of parellelism multi-processor computers
- Disadvantages of threads
  - Shared resources can lead to synchronization problems
  - Need to be careful to avoid deadlocks
  - Need to provide atomic operations

Synchronizing Threads

- Have shared data
  - What helps threads be lightweight
- Threads can be interrupted by the OS scheduler at any time

What challenges does this present?
How can we address these challenges?

Exploring Issues with Shared Data

- Shared Data: an array list (representation: Deck of Cards)
- Discussion
  - What are the operations we can do to the array list?
  - How might we have multiple threads interacting with the list?
    - What would the code look like?
  - Assign people to these roles, acting as concurrent threads
    - Play out different scenarios: how could the threads interleave?
      - Consider what the code would look like

Discussion

- What are the operations we can do to the array list?
  - Top card?, Draw card(s), Shuffle, Add cards, Cut deck

```
def addCard(self, card):
      self.cards.append(card)

def removeCard(self, card):
      self.cards.remove(card)

def topCard(self):
      return self.cards.get(0)

def shuffle(self):
      random.shuffle(self.cards)
…
```


Discussion

- What are the operations we can do to the array list?
  - Top card?, Draw card(s), Shuffle, Add cards, Cut deck

Thread1:
- `deck.pop(0)`
- `deck.pop(0)`

Thread2:
- `deck.get(0)`
- `deck.pop(0)`

Exploring Issues with Shared Data

- Shared Data: an array list (representation: Deck of Cards)
- Discussion
  - What are the problematic situations?
  - What causes them?
  - How can you prevent problematic situations?
    - Motivates synchronization mechanisms

Critical Section

- Sections of code that have to happen uninterrupted or ***atomically***
  - Only one thread can execute at a time

Looking Ahead

- Web Server due next Friday