TIMEOUTS AND CONCURRENCY
DISCUSSION AND SLI.DO ACTIVITY

George Porter
Apr 20, 2018

ATTRIBUTION

• These slides are released under an Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0) Creative Commons license
124 ANNOUNCEMENTS

Today:
- Finishing up timeouts
- Concurrency
- Sli.do activity

HW1 due today

Outline

1. Signals and timeouts
2. Concurrency
3. Sli.do activity
PER-SOCKET TIMEOUTS

struct timeval timeout;
timeout.tv_sec = 10;
timeout.tv_usec = 0;

if (setsockopt (sockfd, SOL_SOCKET, SO_RCVTIMEO, (char *)&timeout,
sizeof(timeout)) < 0)
    error("setsockopt failed\n");

if (setsockopt (sockfd, SOL_SOCKET, SO_SNDTIMEO, (char *)&timeout,
sizeof(timeout)) < 0)
    error("setsockopt failed\n");

Outline

1. Signals and timeouts
2. Concurrency
3. Sli.do activity
CONCURRENCY VS PARALLELISM

• Both deal with doing a lot at once, but aren’t the same thing
  • Given set of tasks \{T_1,T_2,\ldots,T_n\}

• Concurrency:
  • Progress of multiple elements of the set overlap in time

• Parallelism:
  • Progress on elements of the set occur at the same time

CONCURRENY

• Might be parallel, might not be parallel

• A single thread of execution can time slice a set of tasks to make partial progress over time
  • Time 0: Work on first 25% of Task 0
  • Time 1: Work on first 25% of Task 1
  • Time 2: Work on first 25% of Task 2
  • Time 3: Work on first 25% of Task 3
  • Time 4: Work on second 25% of Task 0
  • Time 5: Work on second 25% of Task 1
  • ...

PARALLELISM

Multiple execution units enable progress to be made simultaneously

**Processor 1**
- Time 0: 1st 25% of Task1
- Time 1: 2nd 25% of Task1
- Time 2: 3rd 25% of Task1
- Time 3: 4th 25% of Task1
- Time 4: 1st 25% of Task3

**Processor 2**
- Time 0: 1st 25% of Task2
- Time 1: 2nd 25% of Task2
- Time 2: 3rd 25% of Task2
- Time 3: 4th 25% of Task2
- Time 4: 1st 25% of Task4

FLASH TRAFFIC

- USGS Pasadena, CA office Earthquake site
- Oct 16, 1999 earthquake
THREADING AND PERFORMANCE

• Too much parallelism causes thrashing, excessive switching, lower performance

Outline

1. Signals and timeouts
2. Concurrency
3. Sli.do activity
• What is wrong with this server code?

```c
uint64_t x;
int ret = recv(sock, &x, sizeof(uint64_t), 0);
if (ret < 0) { ...handle error...}
player.updatePlayerPosition(x);
```