EECS 489 - Winter 2024

Discussion 3

Assignment 2 is out!

- Link to Assignment 2
- Due Date: **Friday, February 23rd @ 11:59 pm EDT**
- Repos will be hosted under https://github.com/eecs489
  - Invites will be sent out in the next few days
- If you have not done so yet, fill out the group formation form ASAP
  - https://docs.google.com/forms/d/e/1FAIpQLSfnVI9BlnTZL07BtIb4BGhQm5V4rvQbgXAsGgsYH2TfJroHNg/viewform?usp=sf_link

Assignment 2 is out!

- If you are looking for additional group members, fill out this form:
  - https://docs.google.com/forms/d/e/1FAIpQLSdrmqSxOshX77EpNZgNiXphg2_f7IFz8GoITDlDaWzjzLyO1A/viewform?usp=sf_link
  - This will help match people looking for different groups
    - **Do not fill out if you have a group set**

Assignment 2 is out!

START EARLY

- This is considered the hardest project in the class
- 2 large components that can be done in parallel

Today

- Assignment 1 Recap
- DNS Lecture-style Questions
- Assignment 2 Prep

Assignment 1 Recap: recv()

```
ssize_t recv(int sockfd, const void * buf, size_t len, int flags);

// Example: (bytes_recv <= MSG_SIZE (may not get all of them)

ssize_t bytes_recv = recv(sockfd, buffer, MSG_SIZE, 0);

// Also can do this: (bytes_recv == MSG_SIZE, but we block here)

ssize_t bytes_recv = recv(sockfd, buffer, MSG_SIZE, MSG_WAITALL);
```

Assignment 1 Recap: Is this correct?

```
int client_sd;
char buffer[1000];
ssize_t total = 0;
while (true) {
      ssize_t recv_bytes = recv(sockfd, buffer, MSG_SIZE, 0);
      total += recv_bytes;
      // Expecting some designation that the client is done sending
      if (buffer[0] == ‘F’) {
            break;
      }
}
```

Assignment 1 Recap: Better

```
int client_sd;
char buffer[1000];
ssize_t total = 0;
while (true) {
      ssize_t recv_bytes = recv(sockfd, buffer, MSG_SIZE, MSG_WAITALL);
      total += recv_bytes;
      // Expecting some designation that the client is done sending
      if (buffer[0] == ‘F’) {
            break;
      }
}
```

Q1

- Suppose the EECS department has their own DNS server for all computers in the department
- How could you determine if an external web site was likely to be accessed from another computer in EECS a couple of seconds ago?

Q1

- Suppose the EECS department has their own DNS server for all computers in the department
- How could you determine if an external web site was likely to be accessed from another computer in EECS a couple of seconds ago?
  - Perform two consecutive dig queries and compare the query time

Q2

- Suppose you are trying to access the page web.eecs.umich.edu/course/eecs489. You are connected to your home WiFi with its own local DNS (from your ISP), and are not connected to MWireless/UMich's network.
- Give the order of name servers queried over time and their replies.
- Assume:
  - No prior caching, and **Recursive** name resolution
  - umich.edu and eecs.umich.edu are in separate zones, eecs.umich.edu is authoritative for all hostnames ending in .eecs.umich.edu

Q2

- Suppose you are trying to access the page web.eecs.umich.edu/course/eecs489. You are connected to your home WiFi with its own local DNS (from your ISP), and are not connected to MWireless/UMich's network.
- Give the order of name servers queried over time and their replies.
- Assume:
  - No prior caching, and **Recursive** name resolution
  - umich.edu and eecs.umich.edu are in separate zones, eecs.umich.edu is authoritative for all hostnames ending in .eecs.umich.edu
- Queries: local DNS -> root -> edu -> umich -> eecs
- Replies: (eecs -> umich), (umich -> edu), (edu -> root), (root -> local DNS)
- Reason: Recursive means we have to go through it from the client to local DNS, all the way up and down to eecs; replies follow in the reverse

Q3

- Suppose you are trying to access the page web.eecs.umich.edu/course/eecs489. You are connected to your home WiFi with its own local DNS (from your ISP), and are not connected to MWireless/UMich's network.
- Give the order of name servers queried over time and their replies.
- Assume:
  - No prior caching, and **Iterative** name resolution
  - umich.edu and eecs.umich.edu are in separate zones, eecs.umich.edu is authoritative for all hostnames ending in .eecs.umich.edu

Q3

- Suppose you are trying to access the page web.eecs.umich.edu/course/eecs489. You are connected to your home WiFi with its own local DNS (from your ISP), and are not connected to MWireless/UMich's network.
- Give the order of name servers queried over time and their replies.
- Assume:
  - No prior caching, and **Iterative** name resolution
  - umich.edu and eecs.umich.edu are in separate zones, eecs.umich.edu is authoritative for all hostnames ending in .eecs.umich.edu
- Queries: (local DNS -> root), (local DNS -> edu), (local DNS -> umich), (local DNS -> eecs)
- Replies: (root -> local), (edu -> local), (umich -> local), (eecs -> local)
- Reason: Iterative has queries from the top to bottom, same for replies

Assignment 2: A1 Recap

- In A1, we didn’t consider trying to handle multiple clients at one time
- Assume we needed to handle multiple connections, how would we do this?
  - One thought: Multithreading (thread for each connection)
  - What else?

Assignment 2: select()

- select(): I/O Multiplexing
- Allows a program to monitor multiple file descriptors (connections)
- Waits until one or more of the file descriptors are ready/active for some I/O operation

Assignment 2: select()

```
#include <sys/select.h>

int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
struct timeval *timeout);

// For A2, we only care about readfds, also no timeout

// so typically we invoke with

select(FD_SETSIZE, &readfds, NULL, NULL, NULL);
```

Assignment 2: macros

Some useful macros:

- `void FD_SET(int fd, fd_set *set); // Add fd to the set`
- `void FD_CLR(int fd, fd_set *set); // Remove fd from the set`
- `int FD_ISSET(int fd, fd_set *set); // Return true if fd is in set, might not be after select()`
- `void FD_ZERO(fd_set *set); // Clear all entries from set`

Assignment 2: Demo Code

- Code is posted on GitHub here

Wrap-Up

- Thanks for coming!
- Make sure to start Assignment 2!
  - Hardest of the projects
- Example code will be on GitHub under the Discussion folder