Project 1 Discussion

Ming Liu
Bug – Part1 Step C2

• If you receive 16 bytes as the payload_len, it's a mistake in our implementation so you can disregard that as it doesn't affect any of the later stages anyway. However don't make the same mistake in your part 2 stage c2.
Key Data Structure ➔ <netinet/in.h>

- **Generic socket address structure**

  ```c
  struct sockaddr {
    unsigned short sa_family;  // address family, AF_XXX
    char sa_data[14];         // 14 bytes of protocol address
  }
  ;
  ```

- **IPv4 socket address structure**

  ```c
  struct in_addr{
    in_addr_t s_addr;       /* 32 bit IPv4 network byte ordered address*/
  }
  ;
  struct sockaddr_in {
    uint8_t sin_len;       /* length of structure (16)*/
    sa_family_t sin_family; /* AF_INET*/
    in_port_t sin_port;    /* 16 bit TCP or UDP port number */
    struct in_addr sin_addr; /* 32 bit IPv4 address*/
    char sin_zero[8];      /* not used but always set to zero */
  }
  ;
  ```
Host Byte Order and Network Byte Order

• Big endian and little endian
  ✓ Intel, PowerPC
• htons, htonl, ntohs, ntohl
TCP ➔ Connection Oriented Service

TCP Client:
- `socket()`
- `connect()`
- `write()`
- `read()`
- `close()`

TCP Server:
- `socket()`
- `bind()`
- `listen()`
- `accept()` blocks until connection from client
- `read()`
- `do something`
- `write()`
- `read()`
- `close()`

TCP connection establishment
UDP → Connection Less Service

UDP Client
- socket()
- sendto()
- recvfrom()
- close()

UDP Server
- socket()
- bind()
- recvfrom()
- do something
- sendto()
Handle Multiple Clients 1 ➔ Multi-processes

```c
pid_t pid;
int listenfd, connfd;
listenfd = socket(...);
bind(listenfd, ...);
listen(listenfd, ...);
for (; ; ) {
    connfd = accept(listenfd, ...);  /* blocking call */
    if ( (pid = fork()) == 0 ) {
        close(listenfd);  /* child closes listening socket */
        /***process the request doing something using connfd ***/
        close(connfd);
        exit(0);  /* child terminates */
    }
    close(listenfd);  /*parent closes connected socket*/
}
```
Handle Multiple Clients 2 ➔ I/O multiplexing

- Select
- poll / epoll
I/O blocking
I/O non-blocking

- `fcntl` API
Multithread

- Kernel thread supported
- Pthread --> Threadpool
- Select, Poll, Epoll
- USENIX ATC 1999, Flash: An Efficient and Portable Web Server
- Unix Network Programming Volume 1