CS 471/571 Spring 2018

Lecture 1
Outline

• Example Web Request
• Performance Measures
• UDP Example Network Programming in C
• TCP Example Network Programming in C
Simple Example

• Follow a Web page request through the network
Uniform resource locator (url)

```
GET http://www.someplace.faraway.com/index.html HTTP/1.1
```

HTTP/1.1 202 Accept
Domain Name Server

• Where is someplace.faraway.com?
• Addresses
  – Consider a cell phone analogy
  – Call by choosing a person’s name?
• IP address
  – For example 138.49.196.100
• Port number (80 for http)
<table>
<thead>
<tr>
<th>Frame Header</th>
<th>IP Header</th>
<th>TCP Header</th>
<th>HTTP Header</th>
<th>HTTP Data</th>
<th>Frame Trailer</th>
</tr>
</thead>
</table>
Packet Switching

- Packet
- Store and forward
- Forwarding table
Performance Metrics

- Bandwidth (Data Rate)
- Transmission delay
- Propagation delay
Packet Delay

- Processing delay
- Queuing delay
- Transmission delay
  - Bits sent/Data rate
- Propagation delay
  - Depends on transmission medium and distance between the sender and the receiver
  - Distance/Propagation speed
  - Propagation speed
    - Between $2 \times 10^8$ meters/second and $3 \times 10^8$ meters/second
Packet Delay

• Total packet delay in a node
  – Processing delay \( (\text{procD}_i) \) + queuing delay \( (\text{queD}_i) \) + Transmission delay \( (\text{tranD}_i) \) + propagation delay \( (\text{propD}_i) \)

• End-to-end delay
  – Suppose the packet traverses \( N \) nodes for the source to the destination
  – One way end-to-end delay
    • \( \sum_{i=1}^{N} (\text{procD}_i + \text{queD}_i + \text{tranD}_i + \text{propD}_i) \)
Queuing Delay and Packet Loss

• Arrival rate (bits/second)
  – Modeled with statistical measures
  – The book uses the formula $L/a$ where $L$ is the size of the packets (in bits), $a$ is the arrival rate of packets

• Traffic intensity
  – Arrival rate/Data rate
  – When traffic intensity $> 1$ queuing delay occurs
  – Traffic intensity varies over time

• Packet loss
Average Queuing Delay

![Graph showing the relationship between average queueing delay and \( \frac{L_a}{R} \).]
Throughput

• Total bits sent/time measured
  – Instantaneous
  – Average

• Bottleneck links
  – The throughput cannot be greater than lowest throughput link
Throughput

server sends bits (fluid) into pipe

pipe that can carry fluid at rate $R_S$ bits/sec

pipe that can carry fluid at rate $R_C$ bits/sec
Throughput

Maximum Throughput

Throughput

Offered Load
Other Performance Metrics

• Round trip delay
  – Packet sent
  – Acknowledgement (ACK) received

• \((\text{Round trip delay}) \times (\text{data rate})\)
  – This formula is useful because it indicates the number of bits transmitted before an ACK can be received
Practice Problems

• Suppose you want to move a 200MB file from machine A to machine B. Machine A is connected to machine B by a fiber link that is 10 km long. The data rate is 100 Mbps and the propagation speed is $2 \times 10^8$ m/s.

• (a) If you can transmit continuously, how long (measured from the time the first bit is sent) will it take for the last bit to arrive at B?
Practice Problems

• Now suppose the file is broken into packets of 4096 bytes (ignore packet overhead).

• (b) If the protocol used requires each packet to be acknowledged (ACK) before the next packet is sent and the processing time to generate an ACK and to transmit the ACK is negligible, how long will it take to send the file measured from the time the first bit is sent to the time the last ACK arrives.

• (c) Repeat the above problem except assume an ACK must only be sent once for every 20 packets.
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <ctype.h>
int main(int argc, char *argv[]) {
    int sockfd;
    struct sockaddr_in server;
    struct sockaddr_in client;
    char msg[100];
    unsigned int clen;
    int len;
    int i;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&server, 0, sizeof(struct sockaddr_in));
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    server.sin_port = htons(12000);
UDP Echo Server

bind(sockfd, (struct sockaddr *) &server, sizeof(struct sockaddr_in));

clen = sizeof(client);
len = recvfrom(sockfd, msg, 100, 0, (struct sockaddr *) &client, &clen);

for (i = 0; i < len; i++){
    msg[i] = toupper(msg[i]);
}
sendto(sockfd, msg, len, 0, (struct sockaddr *) &client, clen);

close(sockfd);
UDP Echo Client

//includes not shown
int main(int argc, char *argv[]) {
    int sockfd;
    struct sockaddr_in server;
        unsigned int slen;
    int len;
    char msg[100];

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    memset(&server, 0, sizeof(struct sockaddr_in));
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    server.sin_port = htons(12000);
UDP Echo Client

```c
slen = sizeof(server);
len = strlen(argv[1]);
sendto(sockfd, argv[1], len, 0, (struct sockaddr *) &server, slen);
len = recvfrom(sockfd, msg, 100, 0, NULL, NULL);
msg[len] = '\0';
printf("%s\n", msg);
close(sockfd);
```
int main(int argc, char *argv[]) {
    int sockfd1, sockfd2;
    struct sockaddr_in server;
    struct sockaddr_in client;
    char msg[100];
    unsigned int clen;
    int len;
    int i;

    sockfd1 = socket(AF_INET, SOCK_STREAM, 0);

    memset(&server, 0, sizeof(struct sockaddr_in));
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    server.sin_port = htons(12000);
UDP TCP Server

bind(sockfd1, (struct sockaddr *) &server, sizeof(struct sockaddr_in));

listen(sockfd1, 5);

sockfd2 = accept(sockfd1, (struct sockaddr *) &client, &clen);

len = recv(sockfd2, msg, 100, 0);
for (i = 0; i < len; i++)
    msg[i] = toupper(msg[i]);

send(sockfd2, msg, len, 0);

close(sockfd2);
close(sockfd1);
}
UDP TCP Client

```c
int main(int argc, char *argv[]) {
    int sockfd;
    struct sockaddr_in server;
    char msg[100];

    sockfd = socket(AF_INET, SOCK_STREAM, 0);

    memset(&server, 0, sizeof(struct sockaddr_in));
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    server.sin_port = htons(12000);
```
UDP TCP Client

connect(sockfd, (struct sockaddr *) &server, sizeof(struct sockaddr_in));
send(sockfd, argv[1], strlen(argv[1]), 0);
int len = recv(sockfd, msg, 100, 0);
msg[len] = '\0';
printf("%s\n", msg);
close(sockfd);