

Network Programming - Spring 2004  
Midterm Exam

My Name is: \_\_\_\_\_ Answer Key \_\_\_\_\_

This test contains 6 pages. Please make sure your test packet is complete.

---

**Question 1 (10 pts): TCP/IP**

**1A (5 pts):** Describe the network traffic that results from a call to connect() on a TCP socket. You should assume that the destination address given to connect() is valid (there is a TCP server waiting at the specified address).

3 way handshake:  
Client sends SYN segment with ISN  
Server sends back SYN/ACK with it's ISN  
Client sends back ACK

**1B (5 pts):** Describe how you would go about providing a reliable message-oriented service using UDP.

Receiver of each datagram sends back an acknowledgment message. Sender does timeout and retransmission (if ack not received...).

Each message must have some kind of identifier so the sender can match acks to messages.

**Question 2 (20 pts): The OSI Reference Model.** This question is **not** about TCP/IP!

**3A (5 pts):** What is the OSI reference model?

Standard created by the ISO describing network communication using a layered model. There are 7 layers in the model:

- application
- presentation
- session
- transport
- network
- datalink
- physical

**3B (5 pts):** Tell me about the Data-Link layer.

Provides communication between 2 machines (end-systems) on a network (*on a wire*). For multiaccess networks this includes addressing issues.

Framing of individual messages is also handled by the datalink layer – it uses the physical layer to send/receive individual bits, the datalink layer provides the notion of an “ordered sequence” of bits as a chunk.

**3C (5 pts):** Tell me about the Network layer.

Network layer provides communication between machined (end-systems) that may be on possible different networks. The network layer uses the datalink layer to send/receive messages, and can forward individual messages to different network (provides routing).

IP is an example of a network layer.

**3D (5 pts):** Tell me about the Transport layer.

Transport uses the network layer to send/receive messages to machines on possible different networks, and then is responsible for delivering each message to the right process (provides process-to\_process communication).

TCP and UDP are examples of transport layers.

**Question 3 (20 pts):** Matching - choose (from among the choices listed below) the word/phrase that best matches each statement.

CHOICES:				
Physical Layer	TCP/IP	Sockets	UDP	ARP
Datalink Layer	RARP	FIN	LAN	WAN
Network Layer	End-to-End	Checksum	ACK	SYN
Transport Layer	ICMP	DNS	Flow Control	Hop-by-Hop
Port number	IPv4 Address	OSI	ISO	CGI
Bridge	Repeater	Router	Hub	Gateway
HTTP	TFTP	TELNET	DNS	SSI

TCP Segment type used to communicate an Initial Sequence Number.

SYN

Network Device that forwards *frames* from one *wire* to another.

bridge

A service provided by TCP that is not provided by UDP.

flow control

IP corresponds to this layer in the OSI Reference model.

Network

Used to convert hostnames to IP addresses

DNS

Used to convert IP addresses to MAC (datalink) addresses

ARP

Identifier used by UDP to identify destination endpoint once message is delivered to the destination machine.

Port number

Network API that can be used to write web servers and clients.

Sockets

A network device that translates between protocols (or encapsulates one protocol in another).

Gateway

Layer in the OSI Reference Model that provides communication between processes.

Transport

#### Question 4 (10 pts): Programming

Identify all the problems associated with the following code, and fix or rewrite the code so that it is safe and correct. The function is supposed to create a copy of a string and return a pointer to the new copy.

```
char *string_duplicate(char *s) {
    char *cpy;

    cpy = malloc(sizeof(s));

    strncpy(cpy,s,strlen(s));

    return(cpy);
}
```

The big problem is that `sizeof(s)` will always return the size of a pointer to `char` (typically a pointer is 4 bytes), so the string being copied might be more than 4 bytes and there will be a buffer overflow...

The `strncpy` is also a problem as we need to make sure the new copy (in `cpy`) has a null at the end of it.

New code:

```
char *string_duplicate(char *s) {
    char *cpy;

    cpy = malloc(strlen(s)+1);
    strncpy(cpy,s,strlen(s)+1);

    return(cpy);
}
```

**Question 5 (10 pts): Case Studies:**

**5A (5 pts):** Describe (in detail) how the TELNET protocol provides both control information and data over a single TCP connection.

There is a special byte value (named IAC, value is 255) that is used as a prefix for all control information. Each time an IAC is received the receiver knows the next byte is some control information.

To include the byte value 255 in the stream, the sender sends two bytes, the first is interpreted as a control prefix, the second tells the receiver "no control info is coming, instead please insert a 255 in the data stream).

**5B (5 pts):** Given an HTML form with action "<http://foo.com/blah.cgi>", the method set to "GET", form fields with names "id" and "nickname", and the user types in the string "jones" in the id textbox, and the string "1337 dude" as the nickname. Show a valid HTTP 1.1 request that could be sent by the browser when the user submits the form. You must show a complete request, not just the request line!

```
GET /blah.cgi?is=jones&nickname=1337+dude HTTP/1.1
Host: foo.com
```

**Question 6 (30 pts): Programming**

We want a process that can operate as both a TCP echo server and a UDP echo server. The process can provide service to many clients at the same time, but involves a single process that does not start up any other threads. Provide a high-level description of this program (a few lines describing how the program works), and then provide **as much of the code as possible**. Pseudo code is fine (we won't be compiling your code!), but make sure you mention the name of any system calls you are using.

Sample code on next page

**Extra Credit (0 pts):** A network programmer and Bill Gates walk into a bar. Bill approaches the bar tender and says "I'll have a club soda". The bartender looks at the network programmer and says "and what do you want?". What does the network programmer say?

```

int main(int argc, char **argv) {
    int ld, sd, udp;
    struct sockaddr_in skaddr;
    int length;
    fd_set fd;
    int max;
    int n;
    char buff[1000];

    if ((ld = socket( PF_INET, SOCK_STREAM, 0 )) < 0) {
        perror("Problem creating socket\n");
        exit(1);
    }

    skaddr.sin_family = AF_INET;
    skaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    skaddr.sin_port = htons(atoi(argv[1]));

    if (bind(ld, (struct sockaddr *) &skaddr, sizeof(skaddr))<0) {
        perror("Problem binding\n");
        exit(0);
    }

    if (listen(ld,5) < 0 ) {
        perror("Error calling listen\n");
        exit(1);
    }

    /* create UDP socket and bind it */

    if ((udp = socket( PF_INET, SOCK_DGRAM, 0 )) < 0) {
        printf("Problem creating socket\n"); exit(1);
    }

    if (bind(udp, (struct sockaddr *) &skaddr, sizeof(skaddr))<0) {
        printf("Problem binding\n");
        exit(0);
    }

    /* now process incoming connections forever ... */
    max = (ld > udp ? ld : udp);
    while (1) {
        /* Inialize the fd_set */
        FD_ZERO(&fd);
        FD_SET(ld, &fd);          /* add passive tcp socket */
        FD_SET(udp, &fd);        /* add udp socket */

        /* call select */
        if (select(max+1, &fd, NULL, NULL, NULL) < 0) {
            perror("select problem");
            exit(1);
        }

        if (FD_ISSET(ld, &fd) ) {
            if ((sd = accept(ld, (struct sockaddr *) &skaddr, &length))<0) {
                perror("accept problem");
                exit(1);
            }
            while (n = read(sd, buff, 1000)) {
                if (write(sd, buff, n) < 0)
                    break;
            }
            close(sd);
        }
        if (FD_ISSET(udp, &fd)) {
            /* read a datagram from the socket (put result in bufin) */
            n = recvfrom(udp, buff, 1000, 0, (struct sockaddr *) &skaddr, &length);
            if (n < 0) {
                perror("Error receiving data");
            } else {
                /* Got something, just send it back */
                sendto(udp, buff, n, 0, (struct sockaddr *) &skaddr, length);
            }
        }
    }
}

```