File: Exam 2 QuestionBank.docx Questions for CPSC3600 Exam 2
Last updated: 3/8/2020

Disclaimer: I reserve the right to make minor modifications to any question that appears on an exam. The modification might simply change a number, adding or removing multiple choice options, or it might involve editing the question so it reads more clearly. However, the intent is that the question banks give you the questions that will appear on exams in advance. I purposely am not providing the solutions so that you make the attempt to answer the questions on your own thereby helping you learn the material.

To prepare for the exams, my suggestion is to start at least 1 week prior to the exam, review all assigned reading, go through in detail the solutions to the exercises and the homework. And then, work through as many of the questions in the Question banks that you can. If you are unsure of an answer, you can ask me or the TAs and we will ask you your thinking and then help you get to the correct answer.

Exam 2 will be taken in class, no notes, cheat sheets, books, or computers. Exam2 will look similar to Exam 1 except it will include a 4th section of questions (see below).

- Section 1: Fill in the blank, matching, true/false
- Section 2: Problems involving numeric computation or short answers
- Section 3: Questions based on code snippets from UDPEchoV3 (located in git ./code/UDPEchoV3)
- Section 4: Note this might go away. But I'm thinking of adding a couple extra credit questions. These will be questions based loosely on questions from the exam question bank but modified such that if you get them correct you are showing me that you really do know the material.

STATUS: As of class time 3/9/2020, I still have to add questions in section 3 related to UDPEchoV3. I might edit questions from Section1,2 – possibly adding a 1-2 more. Let's say that the question bank will be frozen by tonight's review session at 7:00pm.

Section 1 will contain a number of questions that are True/False, multiple choice, or fill in the blank

---

Q1.1 True/False - The sockets library call getaddrinfo() is the DNS resolver that we mentioned in class that runs on all hosts.

Solution: F getaddrinfo() might invoke the resolver but there is a different library interface to the resolver

Q1.2 A client network program that must interact with a server Host issues a DNS _____ in order to learn the _____ of the server host.

Solution: Query IP address

Q1.3 True/False : Using our UDPEchoV3 client, if we specify a server IP address of 130.127.201.28 (assume a valid address), the clemson name server will provide an authoritative answer to the client DNS query.

Solution: F If a server name is given in dotted decimal format, the getaddrinfo is smart enough to know how to convert that string into a valid uint32_t IP address.

Q1.4 The UDPEcho server program is likely to perform at least the following sockets library calls. Identify which of the following are true or valid in the sense that they seem correct,necessary, and in order relative to its place in the list of items:

1. initializes all variables
2. mallocs a network buffer large enough to hold at least100,000 bytes
3. ~~sets up the struct addrinfo with the client address information~~
4. Issues a socket() call to create the socket
5. ~~Issues a DHCP request to obtain an IP address~~
6. ~~Issues a bind specifying the port number it wishes to use as a part of its socket endpoint.~~
7. Issues a sendto() to inform the client it is ready
8. Issues a recvfrom() where it blocks until client data arrives
9. ~~Once the server is ready to terminate, it issues a DHCP REL to release the IP address~~
10. And then it issues a close passing the socket descriptor.

Solution: items 1,2,4,7,8,10 are true. The other items are NOT true.

Q1.5 A 'man recvfrom' shows:

ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags,
struct sockaddr *src_addr, socklen_t *addrlen);

And our UDPEchoV3 client invocation:

*rc = recvfrom(sock, RxBuffer, messageSize, 0, (struct sockaddr *) &fromAddr, &fromAddrLen);*

Select all of the statements that are true:

- The returned value that gets assigned to the rc variable represents either an error indicator or the number of bytes that were not sent successfully
- To receive up to 4096 bytes, the following code would work:
    - *Uint16_t *bufPtr*

- *bufPtr = (uint16_t *) malloc (4096 * (sizeof (uint16_t));*
- *rc = recvfrom(sock, bufPtr, 4096, 0, (struct sockaddr *) &fromAddr, &fromAddrLen);*

- If the host that sent the data that arrives on the recvfrom uses an IP V6 address, we would have to modify the fromAddr before using it as the destination address on the sendto().

Solution:

- The returned value that gets assigned to the rc variable represents either an error indicator or the number of bytes that were not sent successfully
  - FALSE: it's either an error indicator or the number of bytes successfully received
- To receive up to 4096 bytes, the following code would work:

  - *Uint16_t *bufPtr*
  - *bufPtr = (uint16_t *) malloc (4096 * (sizeof (uint16_t));*
  - *rc = recvfrom(sock, bufPtr, 4096, 0, (struct sockaddr *) &fromAddr, &fromAddrLen);*

  False…this would not be correct. The second line would be:
  - *bufPtr = (uint16_t *) malloc (4096 / (sizeof (uint16_t));*

- If the host that sent the data that arrives on the recvfrom uses an IP V6 address, we can always reuse the fromAddr for any subsequent sendto()'s back to the host as long as fromAddr is defined as the generic sockaddr struct.

FALSE: fromAddr MUST be a struct sockaddr_storage.

1.6 ARP requires a specific ARP server directly connected on the network

Solution: False. ARP is a distributed application – there is no centralized server

1.7 True/False ARP REQ messages can not be forwarded by a router

Solution: True - LIMITED broadcasts are NEVER allowed to be routed

1.8 Entries in a local ARP cache _____ so that the protocol can dynamically correct for the case when any Host on the network experiences a change of its _____

Solution: timeout IP address

1.9 DNS involves at least the following components: _____, _____, and at least the _____.

Solution DNS involves at least the following components: __local resolver_____, __local domain name server, and root name server.

1.10 An recursive DNS Query requires the _____ to perform all of the _____.

Solution: A recursive DNS Query requires the ___local name server_____ to perform all of the _____work_____.

1.11 If a student at Clemson issues 'ping www.clemsonBadAddress.edu' (a bad domain name), the node that most likely identifies this error is the _____.

Solution If a student at Clemson issues 'ping www.clemsonBadAddress.edu' (a bad domain name), the node that most likely identifies this error is the ___the .edu TLD name server_____.

1.12 The DHCP protocol can provide a host at least the following config information : _____, _____, _____, _____.

Solution: an IP address, the subnet mask, the first hop router and the DNS name server.

1.13 DHCP 'leases' a host it's IP information. The only way the address is released and placed by in the DHCP server's pool is when the lease period times out.
Solution: False. The Host can specially issue a DHCP release command, avoiding potentially wasting the address for the entire lease period.

Section 2 will be problems/short answer questions

Q2.1 If we were to issue a 'ping -c 4 www.sears.com', we are likely to see the first ping RTT to be larger than subsequent RTTs. Why ?

Solution: The first ping requires a DNS lookup which might take hundreds of ms's. But the next iteration, the program has the IP address and so the RTT would not include the time required for DNS.

Q2.2 . How big is the MAC address space? The IPv4 address space? The IPv6 address space?

Solution: 2^{48} MAC addresses; 2^{32} IPv4 addresses; 2^{128} IPv6 addresses.

Q 2.3 Kurose text review problems from Chapter 6 (on ARP) R10

Suppose nodes A,B, and C each attach to the same broadcast LAN. If A sends thousands of IP datagrams to B with each encapsulating frame addressed to the MAC address of B, will C's adapter process those frames? If so will C's adapter pass the IP datagrams in these frames to the network layer of C? How would your answer change if A sends these frames with the MAC broadcast address?

Solution: C's adapter (and as well as all other Host adapters directly connected to the broadcast LAN) will receive the frame. C's PHY layer (a combination of the hardware NIC and the system device driver) will only pass up to the IP layer frames whose destination MAC address either matches the MAC address of C's interface OR if the destination MAC address is a broadcast address (6 octets of all 1's)

If A sends the thousands of IP datagrams using a destination IP address of all 1's (the limited IP broadcast address), all frames will encapsulate the IP datagram and have a destination MAC address of all 1's (i.e., a broadcast MAC address). All hosts will receive the frame and pass the frame to the IP layer. If the destination IP address in the IP datagram does not have a broadcast address or is not the received host's IP address, the datagram is simply discarded. If the IP address is valid, the IP network layer next looks at the protocol ID field in the IP header to see if it should forward the datagram to either the TCP, UDP, or ICMP transport protocol component. Assuming the protocol field matches one of these, the datagram is passed to the correct transport layer. For TCP and UDP, these layers look at the destination port number to find the corresponding socket endpoint instance to complete the delivery of data to the application. For ICMP, the information in the ICMP header is used (ICMP ID, type,class) to transfer the datagram to the right location in the ICMP layer for processing.

Q 2.4 Kurose text review problems from Chapter 6 (on ARP) R11

Why is an ARP query sent within a broadcast frame? Why is an ARP response sent within a unicast frame with the destination MAC address set to that of the Host that sent the query?

Solution: Let's assume Host A wants to Ping Host B. Host A knows the IP address of Host B,but it needs to learn Host B's MAC address. So Host A issues an ARP Query. An ARP query is sent in a broadcast frame because the querying host does not know the destination's IP address and therefore can ONLY use a broadcast. All Hosts that

are directly connected to the network receive the broadcast. Only the Host whose IP address matches the IP address in the ARP query replies. Since the ARP Query contains the IP address and MAC address of the client (Host A), Host B issues a Reply to Host A using unicast rather than a broadcast. Note: Unicast implies the IP datagram destination IP address is Host A (the source IP is Host B). The destination MAC address of he Reply is that of Host A and the destination IP address is that of Host A.

Q2.5 Outline a very simple UDP server whose job is to echo anything it receives. You just need to identify the socket calls. You do need to declare the variables used by the socket calls - but we don't expect you to write down correct "C" code. You do NOT need to show the code that would be required before or after the socket calls. Your answer should be sufficient to convince us that:

1. you understand the socket calls that are required
2. you have a basic understanding of the parameters and returned data from these library routines.Write the server pseudo-code in your own words.

Server:

- inits variables, possibly malloc's data buffers
- issues a socket() to create a UDP socket on the local host
- sock=socket
- rc = bind()
- while (true)
- rc = recvfrom()
- Rc = sendto()
-

Q 2.6 Same as the previous question, but this time outline the client code. Assume the client is our UDPEchoV3 program.

Client:

- inits variables, possibly malloc's data buffers
- char *bufPtr = malloc(sizeof(char)*MAX_BUFFER_SIZE);
- sock=socket
- while (true)
- rc = sendto(bufPtr);
- Rc = recvfrom(bufPtr);
-

Q2.7 ARP Protocol: Case 1: Host A sends to Host D. Issues:

•Why might there not be a response from D ?
•What happens if D does not respond?
•ARP cache entries timeout after 20 minutes. Why?
•Why is the entry removed regardless of if the host uses the information during the 20 minutes?

Solution:

- D might not exit, or the reply might be dropped
- If D does not respond, ARP itself does nothing. It will wait for an amount of time and then remove the state surrounding the outstanding REQ. When it times out, it will drop the packet (or packets) waiting for the ARP resolution. It's up to the application to retry.
- ARP cache entries timeout after 20 minutes to allow for the following: suppose a Host on the network gets a new IP address (e.g., its DHCP lease period expires and it is assigned a new address). ARP is meant to dynamically support MAC – IP address bindings.
- The cache entry is removed after the TTL period even if the application has been 'hitting' the cache entry even moments before the TTL expires. This is because, ARP has no way of knowing if the application is simple retransmitting. In which case, the entry needs to be removed causing a new ARP 'whois and I am" exchange so the Host learns the updated IP address.

Q2.8 For a communication session between a pair of processes, which process is the client and which is the server?

Solution: typically, the program that starts and blocks on a recv waiting for data is the "server". The program that immediately sends to a destination host is the 'client. In TCP, these two different types of starting a new connection is referred to as a passive and active open respectively.

Q2.9 What does zone of authority mean?

Solution: Zone of authority means domain names that are under the admin control of a particular name server. The NS would always provide authoritative answers to client queries.

Q2.10 What is the difference between an authoritative and non-authoritative DNS ANSWER?

Solution: Authoritative means the answer comes from a NS in the zone of authority. Non-authoritative answers typically means they are answers from a NS cache (that is not in the zone of authority).

Q2.11 If you were to install ubuntu 18.04 server (rather than the desktop version we use) what is the name of the DNS software it would use if we configured it to operate as a NS ?

Solution: Most likely, the bind DNS implementation

Q2.12 On ubuntu, how can we effectively add a local cache DNS binding?

Solution: We can edit the file /etc/hosts and add any number of entries (i.e., domain name and IP address). I would not recommend doing this however especially for hosts not under your control.

Section 3 will involve answering questions related to code snippets. Refer to the source code identified in the question. Note that questions related to UDPEchoV3 will reference the pdf posted on our course web page UDPEchoV3 For Printing. This lists client.c, server.c, session.h, and session.c in a single file using global line numbers.

Questions Q3-1 - Q3-4 refers to the file: LectureNotes-3/4/2020.pdf posted on the course web page. Note that they also apply to the latest UDPEchoV3 code. The solution to Q3.1 was added to client.c in UDPEchoV2.

Recommendation: Spend your time studying ONLY the UDPEchoV3 code. All exam questions will come from that code (rather than all the code that led up to UDPEchoV3).

Q3.1 The following set of questions refer to Lecture notes 3/4/2020 :

- How many octets are used by a generic socket address structure ? (i.e., struct sockaddr)
- How many octets are used by an IPV4 address structure ? (i.e., struct sockaddr_in)
- How many octets are used by an IPV6 address structure ? (i.e., struct sockaddr_in6)
- How many octets are used by a struct sockaddr_storage ? (Hint: for this I suggest adding a printf in the UDPEchoV2 client code to see the results of sizeof(struct sockaddr_storage)

Refer to UDPEchoV2 client.c code at line 259 (search for LECTURE) – this was recently added with the following code:

```
//Code to confirm the sizes of various structs and data types
259 //LECTURE
260   uint32_t tmpW, tmpX, tmpY, tmpZ;
261   tmpW = sizeof(struct sockaddr);
262   tmpX = sizeof(struct sockaddr_in);
263   tmpY = sizeof(struct sockaddr_in6);
264   tmpZ = sizeof(struct sockaddr_storage);
265   printf("client: size sockadd:%d, sockaddr_in:%d, sockaddr_in6:%d, sockaddr_storage:%d \n",
266         tmpW, tmpX, tmpY, tmpZ);
```

The size of the 4 structs:
client: size sockadd:16, sockaddr_in:16, sockaddr_in6:28, sockaddr_storage:128
Notice that the IPV6 address struct is larger than the generic socket struct sockaddr. This is why a struct sockaddr_storage is allocated and then ptrs to sockaddr, sockadd_in, sockaddr_in6 are all cast to point to this memory – it's guaranteed to be large enough for either type of address.

Q3.2 In the client code that calls the findAF helper function, we see myIPV6Addr will be set to point to an IPV6 address. We know that a struct sockaddr_ip6 is larger (in octets) than a struct sockaddr. Might this cause problems?

This question continues with the previous question. Yes…this can cause a problem. The following will cause a seg fault if the returned address is an IPV6 address:

```
struct sockaddr myaddr;
struct sockaddr *myAddrPtr = &myaddr;
....
 rc =  recvfrom(sock, RxBuffer, messageSize, 0, myAddPtr, &fromAddrLen);
```

However, the following would fix this:

```
struct sockaddr_storage  myaddr;
struct sockaddr *myAddrPtr = (struct sockaddr *) &myaddr;
....
 rc =  recvfrom(sock, RxBuffer, messageSize, 0, myAddPtr, &fromAddrLen
```

Q3.3 True/false: Line 11 of the code in Figure 4 can be set just once outside of the loop rather than be set each iteration (as currently done in the code).

Solution: No….

```
TxIntPtr  = (uint32_t *) TxBuffer;
```

This line needs to be set each loop iteration because the pointer is incremented – so it needs to be reset to point to the first octet of the TxBuffer before it's used in the loop.

Q 3.4 How does the client code (don't worry about the server code) change if we were to assume the sequenceNumber that is sent with each message must be 8 octets rather than 4 octets?

Summary of the changes:

- The client's declaration of the variable must be changed to : uint64_t sequenceNumber
- The struct messageHeaderDefault sequenceNum must also change from a uint32_t to a uint64_t

```
typedef struct {
  uint64_t sequenceNum;
  uint32_t timeSentSeconds;
  uint32_t timeSentNanoSeconds;
} messageHeaderDefault;
```

- When the client packs the network buffer, the routine htonll () must be used instead of htonl() - note this is not a standard sockets function but you can see an implementation in the utils.c file.
- When packing, care must be taken to ensure a uint64_t is copied from the message header sequenceNum struct member to the network buffer. One way is to use a new pointer as shown below

```
 //pack the header into the network buffer
   TxInt64Ptr  = (uint64_t *) TxBuffer;
   *TxInt64Ptr++  = htonll(TxHeaderPtr->sequenceNum);
  //The TxInt64Ptr now points to the correct octet in the network buffer
  TxIntPtr  = (uint32_t *) TxInt64Ptr;
   *TxIntPtr++  = htonl(TxHeaderPtr->timeSentSeconds);
   *TxIntPtr++  = htonl(TxHeaderPtr->timeSentNanoSeconds);
```

Q3.5 In Figure 5, line 6 issues a recvfrom to get the reply from the server

```
rc =  recvfrom(sock, RxBuffer, messageSize, 0, (struct sockaddr *) &fromAddr, &fromAddrLen);
```

Select all of the statements below which are true:

- After a return, the child process that gets created runs until the parent process terminates
- An rc that is returned with a value of 0 implies an error
- The source IP address from the server is IPV6 if the following condition is true:

```
If (fromAddr->sa_family == AF_UNSPEC)
```

- The source IP address from the server is IPV4 if the following condition is true:

  If (fromAddrLen == 4)
- After the call, exactly messageSize number of octets will have been placed in the RxBuffer

All 5 statements are false EXCEPT for the second.

- After a return, the child process that gets created runs until the parent process terminates - this makes no sense. The code does not involve a fork.
- An rc that is returned with a value of 0 implies an error - this is an ambiguous question – sorry! A 'man recvfrom' indicates recvfrom can return a 0 and it might mean a UDP datagram arrived with no data. But in our program, we would interpret this as an error. So the answer to this question is true.
- The source IP address from the server is IPV6 if the following condition is true:

  If (fromAddr->sa_family == AF_UNSPEC)

  NO! sa_family of AF_UNSPEC on the address filled in by the system processing a recvfrom would be very much an error. It should NEVER happen.
- The source IP address from the server is IPV4 if the following condition is true:

  If (fromAddrLen == 4)

  NO! The fromAddrLen is telling us the size of the struct sockaddr NOT the actual size of the IP address.

- After the call, exactly messageSize number of octets will have been placed in the RxBuffer

  NO! The messageSize parameter is telling the system to transfer at most messageSize bytes into the buffer. The actual number of bytes transferred is indicated by the value returned by recvfrom().

Section - example extra credit exam questions