ECE 315 Assignment #4 Solutions

ECE 315 - Computer Interfacing

Assignment #4 Solutions

Due: In the ECE 315 assignment box at 15:45 on Wednesday, Apr. 8, 2020

---

1. Both the Boostrap Protocol (BOOTP) and the Dynamic Host Configuration Protocol (DHCP) use UDP, which has no guarantee of delivery, instead of TCP, which does guarantee that the data payload will be delivered to the destination node. BOOTP and DHCP have the important role of assigning IP addresses to nodes on a subnetwork. So why would one use UDP over TCP at the transport layer for such an important function? (Recall that TCP sets up point-to-point connections between nodes that have IP addresses.) How do BOOTP and DHCP work when the client does not know what its IP address is?

   [15 marks]
   BOOTP uses a two-step protocol where the newly active client first broadcasts out onto the subnet using broadcast mode to make contact with the BOOTP server. The BOOTP server sends to the client's physical layer address a frame that contains an IP datagram. This datagram includes the client's assigned IP address, and the IP address of the server that the client must contact to download its operating system software (that is, the client's *boot image*). Once the client receives the datagram from the BOOTP server, it proceeds to contact the boot server to download its boot image using UDP/IP. *UDP is used instead of TCP to minimize the amount of protocol overhead in the datagrams. In addition, UDP is simpler to implement in the initial ROM software that runs on the client before it has installed its binary image.* The correctness of the boot image can be verified using a checksum. Detection of a bad checksum would trigger a new attempt to download the client's image.

   DHCP follows a more complicated four-step protocol that allows a newly active networked client device to request an IP address. Until the IP requesting protocol has fully completed, both the client and server(s) uses the broadcast addressing mode that is supported by both the physical layer network (e.g., Ethernet) and by IP.

   1. *Discovery*: The client broadcasts out an IP datagram to seek out the server that will provide it with its IP address. The payload includes flags that allow the client to also request the subnet mask, the gateway address, the domain name, and a list of domain name servers (DNSs). The broadcast frame uses its fixed hardware address (e.g., a 6-byte Ethernet MAC address) as the source address and a special broadcast mode address (e.g., 0xFFFFFFFFFFFF in Ethernet) as the destination address. The enclosed IP datagram uses the dummy IP source address 0.0.0.0 and the special broadcast mode destination address 255.255.255.255. The source (client) port number is 68 while the destination (server) port number is 67. TCP cannot yet be used here since the client does not yet have an IP address nor an initial sequence number. In addition, a simple time-out mechanism can be used to recover from a lost server discovery request, and so *UDP is sufficient for the job and is simpler than TCP*.

   2. *Offer*: The server broadcasts out an IP datagram that contains in its payload an offered IP address for the client; the payload also contains various requested options such as the subnet mask, the gateway's IP address, server's IP address, and the IP addresses of the DNSs. The DHCP server first attempts to find a client IP address in a list of client MAC addresses and their corresponding *static IP addresses*. If this search fails, the DHCP then assigns to the client an IP address from a pool of available *dynamic IP addresses*. A lease time (in seconds) is included in the IP datagram if the offered client IP address is from the pool of dynamic IP addresses. The source IP address is the server's IP address, and the destination address is still the broadcast IP destination 255.255.255.255. The source (server) port number is 67 while the destination (client) port number is 68. In the case of an Ethernet frame, the source address is the server's MAC address, and the destination address is the client's MAC address.

   3. *Request*: The client responds by broadcasting out an IP datagram whose payload confirms the IP address that is being offered to it as as well as the server that made the offer. Note that it is possible to have multiple DHCP servers on a subnet, and the ambiguity over multiple offers of IP addresses must be resolved. The source IP address is the client's dummy IP address 0.0.0.0, and the destination address is still the broadcast IP destination 255.255.255.255. The source (client) port number is 68 while the destination

(server) port number is 67. In the case of an Ethernet frame, the source address is the client's MAC address, and the destination address is still the broadcast destinationa address 0xFFFFFFFFFFFF.

4. *Acknowledge*: The server broadcasts an IP datagram that is nearly identical to the offer datagram. One field in the payload either acknowledges the final step in the protocol, or cancels the process (a negative acknowledgement).

Once the client has accepted its IP address and has all of the additional information that it requires (e.g., subnet mask, IP address of the subnet gateway, the IP address of at least one DNS) it can begin using TCP.

2. Briefly explain the similarities and differences between server-side scripting and client-side scripting in the design of dynamic webpages. What do you think are major advantages and disadvantages, with respect to each other, of these two types of scripting when used to create dynamic webpages?

[10 marks]
A *dynamic* webpage is a webpage that changes its content and/or appearance to reflect elapsed time (e.g., a clock display or an animation), changes in data (e.g., sensor readings, system status), and/or user inputs. Server-side and client-side scripting are similar in that both are software techniques that can be used to produce dynamic webpages. However, server-side scripting is required to cause the appearance of a webpage to change depending on data and other information that is stored on the server. Client-side scripting is required to produce changes to a displayed webpage in response to user inputs. Client-side scripting is also generally required to produce rapidly changing dynamic effects on the displayed webpage at the client, such as sounds, animations, and videos.

In *server-side scripting*, the content and/or the appearance of a webpage that is produced by the server varies depending on status and/or data information in the server. For example, it is common for a server to host a database, and the content of webpages will be specified using a programming language that modifies the content depending on data that is retrieved from the database. Various server-side scripting languages have been developed to simply the creation of dynamic webpages on servers. For example, PHP is a widely used interpreted scripting language that can be used to generate HTML pages as part of HTTP responses. The content of those pages can vary depending on data that is retrieved from the server host, such as a database or the content of hardware registers.

In *client-side scripting*, the appearance of a webpage that is being held and viewed by a browser application is updated under the control of programs that are interpretted by the browser. For example, the Javascript language can be used to create interactive webpages that respond to user inputs at the client.

3. Briefly explain how the design choices that were made in the architecture and implementation of the lightweight IP (lwIP) stack have enhanced its portability and efficiency across different microcomputers and operating system environments.

[10 marks]
The lwIP uses several different design choices to enhance its portability across different microcomputers and operating systems:

- The lwIP is implemented using the standard ANSI C programming language, which is supported by compilers for all the major microcomputer hardware platforms. Much of the work involved in porting lwIP to a new hardware platform is thus done with little designer effort by the compiler.
- The lwIP makes minimal assumptions about the host operating system (OS). The lwIP uses its own buffer management system, independent of the message system of the OS. The lwIP requires the presence of a hardware timer (to allow it to implement TCP timeouts). It requires a blocking message receive function in the OS so that tasks can be blocked when they wait for arriving IP datagrams. It requires a nonblocking message post function (for sending IP datagrams), and it needs a semaphore mechanism (for synchronizing the lwIP task and interrupt service routine calls triggered by the communications hardware). These functions are available in most standard operatig systems.
- The software architecture of lwIP, like that of MicroC/OS, explicitly partitions the software into a generic OS-independent software core, the OS-dependent routines(the OS Emulation Layer), the communications hardware-dependent routines (the hardware drivers), and the application programming interface (API). This architecture makes it easier to port the lwIP since it is easier to identify the relatively few places where software might need to be changed.

- The lwIP provides the user application with a simple API is similar to the socket mechanism that is provided in the Unix and Linux operating systems. The simple API decouples the application code from the implementation details of lwIP. The application code will likely not need to be changed because of the lwIP interface if the underlying hardware platform is changed.

4. It is standard recommended design practice to use data buffers and flow control mechanisms to decouple the operation of digital systems that are processing a stream of data. Briefly explain the different and complementary roles played by FIFO buffers and flow control mechanisms in ensuring that those digital systems are well decoupled. What problems would occur if only data buffers, or only flow control mechanisms, were to be used?

   [10 marks]
   The *data buffers accommodate short-term fluctuations* in data production and data consumption rates while a *flow control mechansim accommodates long-term average mismatches* in data production and data consumption rates. The presence of flow control allows the full capacity of the data buffers to be more effectively used so that the operations of the data producer and data consumer are well decoupled, with minimal risk that the data buffer will get completely empty or completely full. Also, the presence of the data buffers prevents the flow control mechanism from unduely interfering with the operation of the data producer and data consumer. More generally, using the two mechansims in combination safely decouples the operation of the data producer from the operation of the data consumer.

   If only data buffers are present without flow control, then any mismatch in the average data flow rate between a data producer and a data consumer would cause problems. If the data production rate is higher than the data consumption rate, then the data buffer in the data consumer will fill up to its capacity and then more arriving data would get lost. If the data production rate is lower than the data consumption rate, then the data buffer in the data consumer would go empty and the data consumer would have to stop until new data arrives from the data producer.

   If only flow control is present without data buffers, then the data flow would be slowed down because the data transfers would be held back by the delays in the flow control mechanism. In the worst case the flow control mechansim would force the data producer and data consumer to operate in "lock step" since there would be no temporary storage place that could allow the data producer to ever get ahead (or behind) the data consumer.

5. Lecture slide 9-26 illustrates the flow of datagrams coming into and flowing from the lwIP. Briefly explain how the sequence numbers and acknowledgement numbers are processed in the both the receive and transmit directions.

   [25 marks]
   In the transmit direction:

   The lwIP receives an ordered stream of bytes to transmit into a socket, which is attached to a particular application. This application is associated with the transmit port number that is loaded into the transmitted TCP segments. *The 32-bit sequence of the first byte in the transmit direction is determined (possibly randomly) when the TCP connection is set up with the first transmitted SYN segment. Subsquent bytes to transmit are assigned ascending sequence numbers, modulo 2^(32). Bytes to transmit are written into the input port of a transmit FIFO payload buffer, which provides elastic storage.*

   A new TCP segment will be formed once a sufficient number of bytes have accummulated in the transmit FIFO payload buffer, and if there is sufficient room in the transmit window to allow more unacknowledged bytes to be transmitted. (The TCP layer keeps track of the sequence number of the last transmitted byte to be acknowledged by the other node, and thus it can keep track of how many more unacknowledged bytes can still be transmitted without exceeding the transmit window size.) If no more unacknowledged bytes can be transmitted, then further TCP transmission is disabled until more previously transmitted bytes are acknowledged.

   If more unacknowledged bytes can indeed be transmitted, then the payload field of a new TCP segment is formed from bytes that are from the output port of the transmit FIFO buffer. The size of the new TCP payload is limited by the maximum transmit segment size, which was negotiated earlier when the TCP connection was first set up. (It is not clear from the documentation, but lwIP could possibly choose to form a TCP segment that is smaller than the maximum size if the maximum number of unacknowledged transmittable bytes is reached while loading the transmit payload.) *A TCP header is created for the TCP transmit payload. That header uses as the 32-bit sequence number the sequence number of the first payload byte. The acknowledgement number field in the TCP header is filled with the sequence number of the next in-order byte that is expected to be received in the opposite direction (from the other node). This acknowledgement number is increased according to updates from the receive side of the TCP layer.*

TCP calls the IP layer function ip_route() to obtain the next node on the datagram's route to the destination node as well as the local hardware output port through which the datagram should sent. Then TCP passes the transmit segment down to the IP layer, where an IP header is prepended to the segment to form an IP datagram. At the same time TCP starts a hardware timer so that it can retransmit the TCP segment later if all of the payload bytes are not acknowledged before the set time-out limit. The IP layer then calls the appropriate network interface driver, which creates the physical layer frame around the IP datagram and then transmits that frame using the network interface hardware.

In the receive direction:

The network interface hardware receives frames containing arriving data. An interrupt service routine transfers the arriving data from the frame into a physical layer buffer. The frame overhead fields are removed and the resulting received IP datagram is passed up to the IP layer. The IP layer recomputes the IP header checksum and compares it with the value in the IP header checksum field. If the checksums do not match then the IP datagram is discarded. If the IP checksums are identical then the source and destination addresses are checked to determine whether or not the IP datagram should be accepted and passed up to the TCP layer for further processing. If the IP datagram is acceptable, then the IP header is removed and the TCP function tcp_input() is called with the TCP segment; otherwise the IP datagram is discarded.

The TCP function tcp_input() recomputes the TCP checksum over the entire segment. If the recomputed TCP checksum does not match the checksum in the TCP segment header then the segment is discarded. If the two TCP checksums match then the function tcp_process() is called to update the TCP state machine for the socket.

The acknowledgement number is extracted from the received TCP header and passed over to the transmit side of the TCP layer so that the sequence number of the last in-order acknowledged transmitted byte can be updated. If this new last acknowledged transmitted byte implies that all of the payload bytes in a retained transmitted TCP segment have now been acknowledged, then that retained transmit TCP segment buffer and the associated timer data structure can be recycled.

If the application corresponding to the destination port number is able to receive data then the TCP layer extracts the received TCP payload bytes and writes them into the input port of a receive FIFO buffer. *The acknowledgement number in the transmit direction is updated up to the sequence number of the last data byte that was received in order.* A buffer containing any newly received in-order received bytes (if any are available) is read from the output port of the receive FIFO, and that buffer is passed on to the application task. If the application task had been blocked waiting for received data, the operating system takes care of moving the task to the ready-to-run queue.

6. Briefly explain what a ping-pong buffer is. Also briefly explain the possible advantages that it brings in some applications. Give at least two different illustrative examples of data processing applications where one can benefit from the use of a ping-pong buffer.

[10 marks]
A ping-pong buffer consists of two equal-sized data buffers that take turns playing the role of write buffer and read buffer. A control signal is used to swap the roles of the two buffers. (The swapping operation is most simply achieved by interchanging base pointers for the two buffers.) The write buffer is loaded with data through a random-access write port, where any address order can be used. At the same time, data that is stored in the read buffer is read out using a separate random-access read port.

Like a first-in first-out (FIFO) buffer, a ping-pong buffer decouples the write-side data rate from the read-side data rate. The main advantage of the ping-pong buffer is that it decouples the address order that is used from writing into the buffer from the address order that is used for reading. This allows different address orders to be chosen to suit the data producer process and the data consumer process.

One example application for a ping-pong buffer arises in computer graphics, where the write side of the buffer is loaded depending on a object drawing process. Meanwhile, the read side of the buffer can be scanned out line-by-line, to suit the address order that is used by the hardware that controls the display pixels.

A second example application is encryption, where the byte order in either the write side or the read side is scrambled by modifying the order of the addresses. Data that is written to the write side using ascending addresses is read out from the read side using scrambled addresses. The data scrambling could be undone by passing the data through a second ping-pong buffer.

7. In your own words, briefly explain how TCP/IP detects and recovers from the situation where IP datagrams that are received out of order because of the different routes that were taken by different datagrams travelling through the network from the source node to the destination node.

   [10 marks]
   A circular data buffer is maintained by the TCP layer in the local node for the receive direction of the TCP connection. This buffer is used to temporarily store the arriving payload bytes in the order that is defined by the sequence numbers that were assigned to the bytes by the transmitting node at the other end of the TCP connection. The details of the routes that the IP datagrams took when travelling from the source node to the destination node are irrelevant to the process of reconstructing the ordered sequence of data bytes.

   Thus if an IP datagram gets delayed or lost, a gap of unwritten data locations will appear in the circular buffer. Later on that gap may get filled in if the corresponding TCP segment happens to arrive late and out-of-order compared to the TCP segments that were actually sent just before and just after the delayed TCP segment. The gap will also get filled in if the late TCP segment is declared lost and happens to be retransmitted.

8. Briefly explain how the four direct memory access controllers (DMACs) in the MCF5234 microcontroller unit support circular buffers in either the source or destination buffers. What constraints are imposed by this feature on the size and base addresses of those buffers when they are used as circular buffers?

   [10 marks]
   The four DMACs each have a 32-bit DMA Channel Control Register, DCRn, where n = 0, 1, 2 or 3. Bits 15..12 of the DCRn control the configuration of the source data buffer. If this field value is 0000 then the circular source data buffer function is disabled. If the field is 0001 then there is a circular source data buffer with 16-byte capacity that starts an a 16-byte address boundary (the lower four address bits of the first address must be 0). If the field is 0010 then the size of the circular buffer is 32 bytes. The data buffer size increases by factors of 2 up until the maximum size of 256 kbytes, which is specified using the field value 1111. In each case, the data buffer must start on an address that is a whole multiple of the data buffer size.

   A circular buffer for the destination can be enabled in the same way using bits 11..8 of the DCRn, where n = 0, 2, 3 or 3. The same restrictions apply on the starting address of the circular buffers for both the source data buffer and the destination data buffer.