ECE 315 Assignment #3

ECE 315 - Computer Interfacing

Assignment #3 Solutions

Due: In the ECE 315 assignment box at 15:45 on Friday, Mar. 20, 2020

---

1. The *Media Access Control* (MAC) sublayer in communication interfaces implements the *data link protocol*, using the terminology of the 7-layer ISO-OSI reference model for computer networks. After consulting reputable resources, use your own words to briefly explain the major functions that are provided by the MAC layer. Briefly contrast those functions with those that are provided by the *physical layer transceiver* (PHY).

   [20 marks]
   In general the MAC layer is responsible for communicating correctly formed frames of bits from one node to and from another node over direct links through a physical communications medium (e.g., cable, wireless, optical). The MAC layer is the main part of the data link layer (other optional sublayers may also be present in the data link layer). The MAC layer provides a control, status and data interface for the overlying network layer. The MAC layer also interacts through a control, status and data interface to the physical layer hardware (the PHY) that is connected to the communications medium.

   Typical MAC functions in the *transmit direction* include:

   - The MAC layer uses the MAC protocol to obtain the right to transmit onto the communications medium via the PHY layer. The communication medium may be shared with other possible transmitters. The MAC layer recovers from collisions, which is the situation when two or more transmitting nodes attempt to communicate over the same medium at the same time.
   - The MAC negotiates with its peers, on the same communications medium, any choices in communication modes such has full duplex versus half duplex communication, the frequency bands (if there is a choice) and the bitrates (if there is a choice).
   - The MAC layer receives data buffers from the higher-level network layer to transmit onto the communications medium via the PHY. Those data buffers may be split up into smaller buffers that suit the required data frame structure.
   - The MAC must construct one or more bit frames of the correct length and structure to carry the data that it has been asked to send. The MAC forms any necessary header and trailers fields that are required by the frame structure. Typical fields include a synchronization field at the start of the frame, address fields, padding fields, and error detection and correction fields.

   Typical MAC functions in the *receive direction* include:

   - The MAC must use the MAC protocol to negotiate communication modes with its peers on the communications medium, and to become ready to accept incoming data frames.
   - The MAC receives data frames, verifies the frame structure, and extracts the various fields. Information in the header and trailer is processed. A checksum may be computed over the header, trailer and/or data payload fields. The MAC may decide to reject the data frame if the frame structure is found to be incorrect and/or the header or trailer information precludes further processing of the data (e.g., the receiver address is incorrect, or the transmitter address is not authorized).
   - The header and trailer fields in the frame are removed and the data payload is passed up to the network layer.

2. The *Fast Ethernet Controller* (FEC) block in the MCF5234 microcontroller provides hardware functions that implement many of the mechanisms required by the MAC layer for an Ethernet interface. For example, the FEC provides a *First-In First-Out* (FIFO) buffer and a *Direct Memory Access* (DMA) controller. Briefly explain the purpose of the FIFO and the DMA controller in the FEC. The FIFO has a fixed capacity that must be partitioned between the transmit and receive directions. How would a system designer decide how much FIFO capacity to allocate for each direction? How would that decision be implemented using a value loaded into an FEC register?

   [10 marks]

The FIFO in the FEC provides *elastic temporary data storage for both the transmit and receive directions* so that the data flow down to (up from) the PHY does not have to be exactly synchronous with the data flow into (out from) the FEC. The FIFO thus supports the MAC function (of the FEC) of exchanging transmit and receive data with the network layer. The total capacity of the FIFO is split into a transmit partition and a receive partition. The relative size of the partitions can be adjusted to accommodate greater anticipated data flow burstiness in either of the two data directions. In the transmit direction, the transmit partition of the FIFO allows the FEC to get a head start and to temporarily store data prior to the time when PHY is told to start transmitting a new data frame. In the receive direction, the receive partition allows received data to pile up a bit within the FEC at times when the CPU might be busy.

The Direct Memory Access (DMA) controller offloads the CPU from of the workload of transferring transmitting data from data buffers in main memory down to the FEC, and of transferring received data from the FEC up to data buffers in main memory. The DMA is more efficient than the CPU at producing the large data movements since DMA uses what is in effect a block move operation, which replaces a large number of CPU MOVE instructions. Using DMA to implement the data movements to and from the FEC frees up the CPU to do other things.

3. Consider the receive buffers that are used in the Ethernet interface of the MCF5234 microcontroller. If the L flag in a *receive buffer descriptor* (RxBD) is written by the FEC hardware to 0, then this means that the corresponding Rx buffer is not the last in the received frame; it also means that the data length in the RxBD will be the same value that was written into the *Ethernet Maximum Receiver Buffer Size Register* (EMRBR). If, however, the L flag in the RxBD is written by the FEC to 1, then this means that the corresponding Rx buffer is the last one in the received frame; it also means that the data length in the RxDB will be equal to the total length of the received frame. What would be the advantage of loading a value in the EMRBR that is at least as big as the length of the longest expected Ethernet frame? What would be a possible disadvantage of loading such a value in the EMRMR instead of a value that is quite a bit smaller than the length of the longest Ethernet frame?

   [5 marks]
   The advantage of loading the EMRBR with the value that is at least as big as the length of the longest expected Ethernet frame would mean that a received data frame would always bit inside only one receive buffer.

   The possible disadvantage of loading such a large value in the EMRBR is that the arriving data frames may actually be smaller than expected. Storing small data frames into the receive buffers would be wasteful since that situation would cause many receive buffers to be only partly full.

4. On lecture slide 10-10 a formula is given for the angle of rotation corresponding to one full step of a stepper motor. If N_s denotes the number of stator teeth and N_r denotes the number of rotor teeth, the step angle is given by (360 deg/N_s) - (360 deg/Nr). Provide a justification for this formula.

   [10 marks]
   Consider the situation at the left side of slide 10-10. Because of the symmetry, we need only to consider a rotation of the rotor by one step going clockwise. (A counter-clockwise step is handled similarly, and so this case does not need to be considered.) For simplicity, we will consider the wave drive waveform, where only one of the stator poles '1' or '2' is energized at any one time. At the illustrated step, the fixed magnetic polarity of the "a" and "f" rotor teeth is the opposite of the polarity of the energized "1" pole of the stator electromagnet. (Opposite magetic poles attract one another.) When pole "1" is de-energized, and pole "2" is energized, then the rotor will rotate clockwise to bring rotor teeth "g" and "b" into alignment with "2". The angle difference from "1" to "2" on the stator is (360 deg./N_s), where the number N_s of stator teeth is 8 in this example. Similarly, the angle difference from "f" to "g" on the rotor is (360 deg./N_r), where the number N_r of rotor teeth is 10 in this example. The step angle is simply the angle that the rotor must turn through, which is (360 deg./N_s) - (360 deg./N_r).

   Extending the argument from the wave drive waveform to the full step waveform boils down to realizing that the effect of having the "2" pole energized while the polarity of the "1" pole is reversed will only slightly rotate the positions of the stable step positions of the rotor. For example, looking at the figure on slide 10-10, a new stable rotor position will have the angle "m" that is midway between "a" and "b" rotated clockwise slightly so that "m" is aligned to the angle between "1" and "2".

5. The stepper motor control algorithm that is described in lecture slides 10-30 to 10-35 assumes that there is one fixed (maximum) slew rate, which will correspond to a particular stepper motor (SM) rotor rotation rate. If the SM drives a train, it is possible that different stretches of track will have different maximum speeds. In that case, the slew rate would have change to conform to the local speed limit. Briefly outline how the SM control algorithm should be modified so that the SLEW period could be made a variable that changes depending on the local speed limit. How, if required, would the code need to be modified in each of the four control algorithm states?

[15 marks]
Each of the four states in the control algorithm is considered below:

- *Stationary state*: No changes would be required to this state since the motor is not moving. The value of SLEW_PERIOD could be freely changed.
- *Accelerating state* (function do_accelerating): If the SLEW_PERIOD is reduced, then the acceleration will likely need to continue longer since the new slew rate is faster that the previous slew rate. This possibility is already handled correctly by the existing code. If, on the other hand, the SLEW_PERIOD is increased, then the stepper motor might have to stop accelerating immediately and either enter the decelerating state (if the current step_period is smaller than the new SLEW_PERIOD) or the slewing state (if the current step_period is the same or sufficiently close to the new SLEW_PERIOD). Thus the condition in the code for entering the decelerating state should be modified to also include the possibility that the current step period is smaller than the new step period. The condition for entering the slewing state should be modified so that the slewing state is entered if the present_period is within some small margin above or below the new SLEW_PERIOD.
- *Slewing state* (function do_slewing): If the SLEW_PERIOD is decreased, then the motor needs to be accelerated so that it increases up to the new slew rate; however, if the SLEW_PERIOD is increased, then the motor needs to be decelerated so that it slows down to the new slew rate. The condition for entering the decelerating state does not need to be changed since the minimum stopping distance will be recomputed using the new SLEW_PERIOD. However, a new conditional branch to the accelerating state is required before the default else branch. The new branch to the accelerating routine would look for the condition that the current step_period is greater than the new SLEW_PERIOD.
- *Decelerating state* (function do_decelerating): If the SLEW_PERIOD is increased, then the deceleration will likely need to continue longer since the new slew rate is slower that the previous slew rate. This possibility is already handled correctly by the existing code. If, on the other hand, the SLEW_PERIOD is decreased, then the stepper motor might have to stop decelerating immediately and either enter the accelerating state (if the current step_period is greater than the new SLEW_PERIOD) or the slewing state (if the current step_period is the same or sufficiently close to the new SLEW_PERIOD). A new conditional branch to the accelerating state is required before the default else branch to remain in the decelerating state. The condition for this branch is that current step period is greater than the new STEP_PERIOD. The condition for entering the slewing state should be modified so that the slewing state is entered if the present_period is within some small margin above or below the new SLEW_PERIOD.

6. The *Internetworking Protocol* (IP) is concerned primarily with providing a simple and convenient way of getting a variety of different packet switched networks to work together as a single network. Briefly discuss how this practical priority explains why IP provides only an unreliable and connectionless service using its own node addressing scheme, even if at least some of the underlying networks might already provide reliable communication; also, some of the underlying networks might already provide a very large number of unique addresses for every node (e.g., the six-byte Ethernet physical addresses).

   [15 marks]
   The basic strategy that is used in IP to meet its design requirements is to find a minimum consensus of capabilities that can be met by the vast majority of possible underlying physical networks.

   - By only providing "best effort" delivery, with no guarantee of reliable delivery, IP is building upon the service qualilty that is provided by both reliable and unreliable physical networks.
   - By reserving for itself the right to discard IP datagrams, IP has a simple way of dealing with traffic congestion and equipment failure.
   - By providing its own new global addresses, IP overcomes the limitations of the address schemes that are used in the underlying physical networks.
   - By reserving the right to split up datagrams into smaller datagrams, IP gives itself a simple way of shrinking the size of its datagrams to fit into the smaller available datagrams of an underlying network.
   - By not guaranteeing delivery along fixed connection paths, IP ensures that it has the flexibility to route each IP datagram separately using the best available path to the destination node that is known at the time that the datagram arrives at any intervening node. The flexibility to reroute datagrams makes it easy to reroute a flow of datagrams to avoid failed equipment or parts of the network that are congested with too much traffic.

7. In your own words, and with reference to the 11-state TCP finite state machine, explain what happens to the TCP enitity in a server when the TCP state advances from the CLOSED state to the ESTABLISHED state. Be sure to explain what causes the state-to-state transitions, and specify all of the states that are entered by the server. Then explain what happens when the client, with whom the server is exchanging data messages, decides to terminate the connection. As before, explain the state-to-state transitions and the states through which the TCP entity passes.

   [15 marks]
   The server node's TCP entity starts off in the CLOSED state and then, as part of its initialization, it is associated with a port number on the server before it enters the LISTEN state. After the TCP entity enters the LISTEN state, the server blocks waiting for a SYN segment from a client. A SYN segment is a TCP segment that has the SYN flag set and the sequence number field loaded with the first sequence number from a client node. When such a SYN segment is received, three things happen: (1) a new instance of the server and TCP entity is created; (2) that new instance sends a SYN segment of its own back to the client, a segment that has both the SYN and ACK flags set, the acknowledgement number field loaded with the client's first sequence number, and the sequence number field loaded with the server's first sequence number; and (3) the new instance of the server and TCP entity enters the SYN_RCVD state. (Note: There may be a limit on the number of possible server instances. If that limit has been reached, them any further SYN segments from requesting clients would either be queued up or simply discarded.) The server instance then blocks while it waits for an ACK segment from the client, that is, a segment has the ACK bit set and the acknowledgement number field loaded with the first sequence number to be used by the server. When such an ACK segment arrives from the client, the server instance moves to the ESTABLISHED state. In the ESTABLISHED stater the server instance and the client exchange TCP segments that contain application-specific data.

   If the client decides to terminate the connection, then the client sends a segment to the server instance that has the FIN flag set. The server instance responds by sending a segment back to the client with the ACK bit set, and then the server enters the CLOSE_WAIT state. The server stays in the CLOSE_WAIT state as it closes down the server application that is handling the TCP connection. When the server's application has stopped, the server instance sends its own FIN segment back to the client, then advances to the LAST_ACK state, and then blocks waiting for an ACK segment from the client. Once the client receives the FIN segment from the server, it can safely proceed to the last step of ending the TCP connection, which is to send an ACK segment back to the server. When the server receives this last ACK segment from the client, the server enters the CLOSED state as the TCP connection becomes fully terminated at both ends. (This last state transition is equivalent to recycling the instance of the server and its associated TCP entity.)

8. When a TCP connection is established, the window size and maximum segment size parameters must be agreed upon by the two end nodes for both directions of the connection. First, briefly explain what factors should be taken into account when these two values are selected for both directions. Then explain how the values are communicated between the two end nodes of the connection.

   [10 marks]
   The *window size* should be chosen to reflect the expected conditions on the connections between the two communicating nodes. If the reliability of the connection is relatively high, but the burstiness of the traffic can sometimes be high, then using a larger window makes sense since any traffic burstiness till tend to be accommodated without impeding the average data traffic rate. However, if the reliability of the connection is relatively low then choosing a larger window size would be inefficient since it will more often happen that the transmitter will tend to be held back waiting for a relatively large number of TCP timeout intervals at a time instead of a small number of timeout intervals. It is thus more efficient to use a smaller window size for less reliable connections. Also, if the receiving node has relatively limited available memory capacity for data buffers, then the window size must not be set so high that the receiver can be swamped with a burst of traffic that overwhelms the receiver's storage capacity.

   The *maximum segment size* is more efficient if underlying physical network(s) can handle large data packets because that way the fraction of transmitted bits that carry useful application data will be higher. However, that efficiency will not be achieved if the data packets must travel over physical layer networks, like Ethernet, where the standard sets a limit on the maximum data frame size (about 1500 bytes in Ethernet). The maximum segment size should also take into account the available memory size in the receiving nodes.