ECE 315 Assignment 1

Arun Woosaree
1514457

March 22, 2020

1

1. MAC determines which device can access the medium at one moment
2. provides an abstraction for the physical layer to the other higher higher-level layers in the OSI model.
3. collision detection, resolution, and avoidance (like CSMA/CD and CSMA/CA for example). Also handles retransmission
4. offers some protection against transmission errors, usually through frame check sequences
5. resolves addressing of destination stations
6. controls access to the PHY
7. translates data packets to digital bits for the PHY

Compared to the PHY, the MAC layer operates at one layer above the PHY in the OSI model. Additionally, the physical layer does not handle error detection and correction. It does not determine which device can access the medium either. The physical layer transmits raw bits, instead of data frames. It is an interface to the transmission medium, while the MAC layer is an interface to the physical layer. The MAC converts data packets to digital bits, while the PHY converts between digital and analog signals.

2

The FIFO buffer in the FEC helps with handling short term burstiness of the data that is both recieved by the FEC and transmitted from the FEC from the CPU. If the CPU cannot keep up for a moment with sending bits or receiving bits fast enough, the buffer ensures that data is available to be processed when the CPU is ready without having to abandon or truncate packets.

The DMA allows for data to be transferred more efficiently between the system's memory, and the FIFO buffer in the FEC. The DMA controller allows for the movement of blocks to take up less CPU cycles, since the direct memory access is done in place of a bunch of CPU move instructions.

If the system designer is expecting more bits to be sent than recieved, they may choose to partition the FIFO buffer such that more space is allocated for sending bits than receiving. Similarly, if more bits are expected to be received than sent, then the FIFO buffer may be allocated such that the buffer for receiving is larger.

This decision would be implemented modifying the value in the FEC register. The buffer descriptors in the transmit and receive direction would be modified as needed to implement the design decision made by the system designer.

3

Advantages

- A receive interrupt will always happen when a packet is received, meaning no ethernet packets would be ignored by the system
- If the EMRBR value was smaller, ethernet frames that are longer than expected will be dropped. With a larger EMRBR value, this is not a problem.

Disadvantages

- The FEC would accept frames that are larger than the ethernet standard, if the value in the EMRBR register is too big which could result in garbage data being read

4

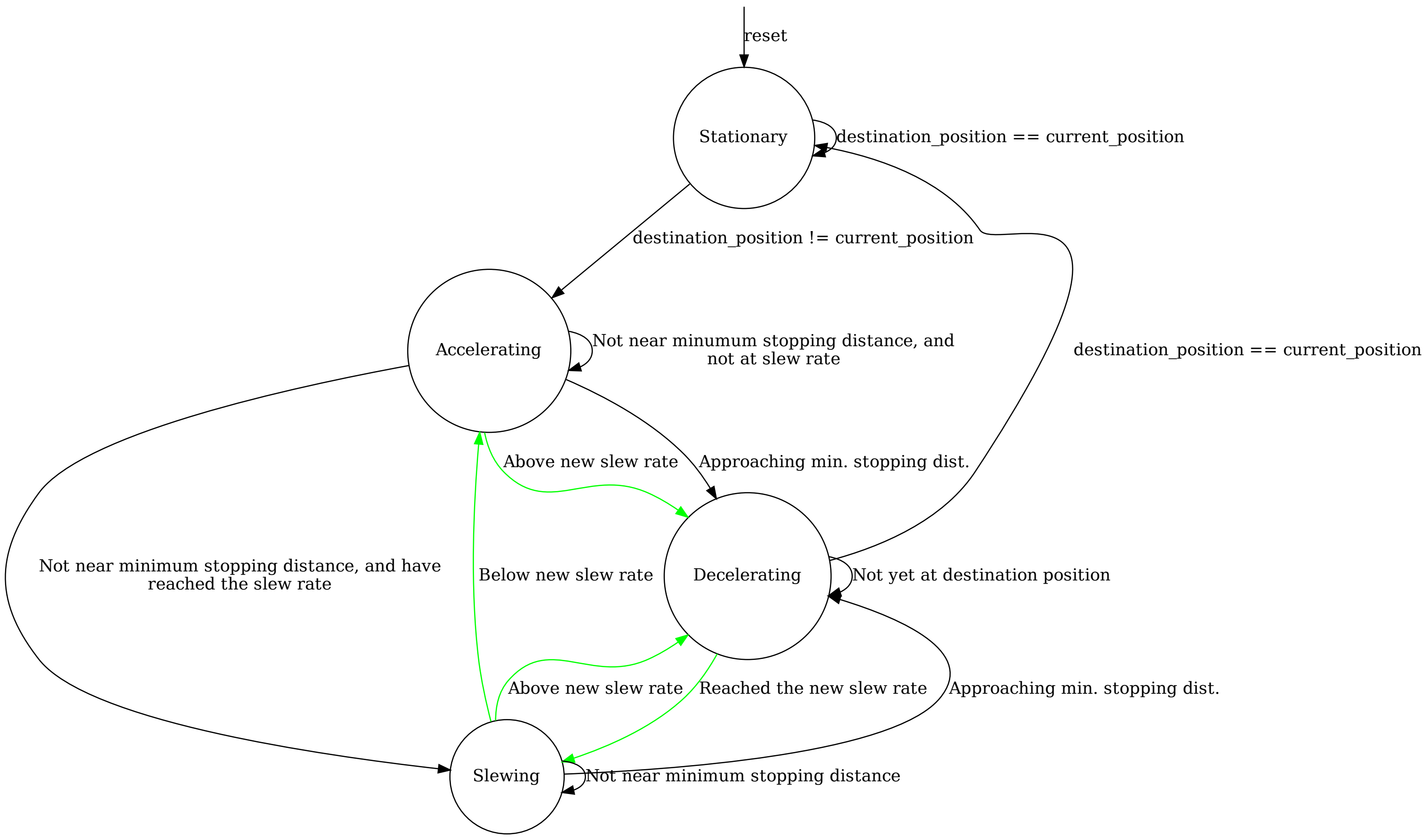
The number of teeth a gear has determines how many steps there are in a full rotation. For example, a gear with four teeth will have 4 steps in a 360°rotation. Therefore, the step angle for the one gear would be $360°/4 = 25°$. The formula is 360°divided by the number of teeth. Now, in the stepper motor, there is a rotor and stator, and the rotor has more teeth than the stator. Because the rotor has more teeth than the stator, we can see both visually and mathematically that the individual step angle for the gear would be smaller than the stator. Because these gears rotate in opposite directions, the overall rotation would be the larger step angle of the stator, subtracted by the smaller step angle of the rotor. We subtract the step angle of the rotor, because it goes in the opposite direction of the stator. So, overall, we get

$$\frac{360°}{n_s} - \frac{360°}{n_r}$$

where n_s is the number of teeth of the stator, and n_r is the number of teeth of the rotor. This matches with the formula described in slide 10-10.

5

The stepper motor control algorithm could be modified as follows


reset
Stationary
destination_position == current_position
destination_position != current_position
Accelerating
Not near minumum stopping distance, and not at slew rate
destination_position == current_position
Above new slew rate
Approaching min. stopping dist.
Not near minimum stopping distance, and have reached the slew rate
Below new slew rate
Decelerating
Not yet at destination position
Above new slew rate
Reached the new slew rate
Approaching min. stopping dist.
Slewing
Not near minimum stopping distance

Basically, in the four code snippets, a new global variable would be added next to the other three (`present_state`, `present_position`, `step_period`) called `slew_rate`.

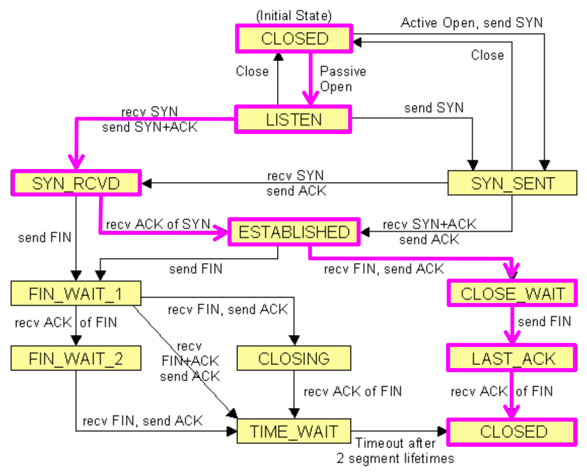
In the acceleration and slewing code, there would be a check to see if the slewing rate changed. If the current slewing rate is faster than what the new slewing rate is supposed to be, then the state should switch to the decelerating state. Similarly, in the decelerating and slewing code, if the new slewing rate is faster than the current slewing rate, it should transition to the accelerating state.

6

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 or a large number of unique addresses for every node because the priority is compatibility between the existing networks. Not all networks connected to the internet are reliable. Not all of them have a very large number of unique addresses for every node. If a connection is made between an unreliable network, and a reliable one, overall, the transmission is still unreliable. Providing an only unreliable allows for greater compatibility between the networks. It also allows for separation of concerns. If reliability is required, or state information about the connection is required, that can be provided by the higher level layers. For example, the TCP protocol can be used to have the illusion of a reliable connection. Similarly, IP using its own addressing scheme allows for the upper layers to use their own addressing schemes which allow for more options and compatibility for connections and protocols.

7

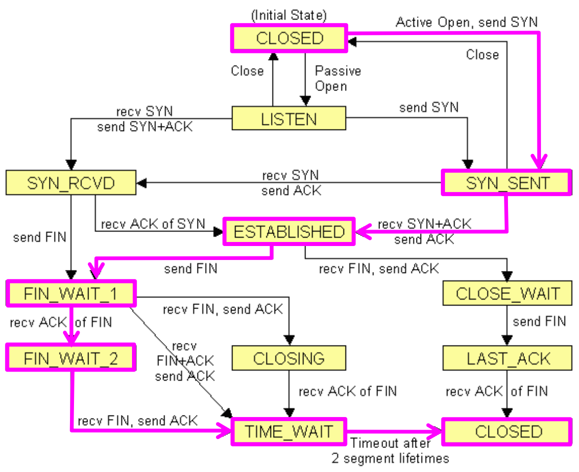
Normal TCP State Sequence in Servers



8-40

On the server side, it starts in the CLOSED state, which means there is no connection. When the server performs a passive open, it creates a transmission control block and readies itself to recieve a SYN segment. This is known as the LISTEN state. An active opener sends the server a SYN segment. The server responds with its own SYN segment. The server also acknowledges the client's SYN by sending a ACK. This is known as the SYN+RCVD state. When the server receives the ACK from the client in response to the SYN it sent, the connection is established. (The ESTABLISHED state)

Normal TCP State Sequence in Clients



8-41

On the client side, when the server decides to terminate the connection, it receives a FIN. The client's application also sends a FIN segment. Once the client has sent its FIN, it transitions to the FIN_WAIT_1 state. In the normal case, it receives an ACK from the server for the FIN which was sent, and goes to the FIN_WAIT_2 state where it continues to wait for the FIN segment from the server. When it does, it sends an ACK for the server and transitions to the TIME_WAIT state. From there, the client waits for double the maximum segment lifetimes to make sure the ACK sent to the server was received. When the timer expires, the connection is terminated and the state CLOSED is reached.

What can also happen is that in the FIN_WAIT_1 state, the client receives the server's FIN segment before it acknowledges the client's FIN segment. The client sends an ACK to let the server it received the FIN segment, and the transition is made to the CLOSING state. From there, the client waits for double the maximum segment lifetimes to make sure the ACK sent to the server was received. When the timer expires, the connection is terminated and the state CLOSED is reached.

8

The window size specifies how many bytes can be transferred without acknowledgement from the other device. When selecting the window size, it is important to consider that a larger window size allows for greater variability in timing, but also requires the reveiver to be prepared to receive larger bursts of data bits. A larger window size is more desireable for high speed and reliable connections, like optical fibre. However, selecting too large of a window size means that the receiver may not have enough space in its receive buffer which would lead to loss in data.

However, if the connection is not as reliable, a smaller window size may be more desireable, since it requires acknowledgement that the data was recieved more often. This can avoid repeated transmissions of data. However, selecting too small of a window size can lead to inefficient utilzation of the connection.

When a window size of 0 is received, the sender stops sending data.

The window size is advertised by the receiver, so that the sender knows how many bytes it can send before waiting for an acknowledgement.

When selecting the maximum size segment size, it is important to consider that it specifies how much data the receiver can receive in a single TCP segment. Having a smaller maximum segment size will reduce IP fragmentation, however, this comes with the cost of higher overhead. It is typically determined by the operating system of the communicating devices, and each device can use a different maximum segment size. It is announced during the TCP handshake in the SYN, and is not a negotiated parameter.