Fall 2021 CS120 Project 2. Multiple Access

Due Date: Nov. 14, 2021

(16 points + 4 points)

Suggested workload: 6 FULL days

Please read the following instructions carefully:

- This project is to be completed by each group **individually**.
- Submit your code through Blackboard. The submission is performed by one of the group members.
- Each group needs to submit the code **once and only once**. Immediately after TAs' checking.

Overview

This project will augment the physical data communication link from Project 1 to support mutual communication among multiple nodes. The design goal of this project is very similar to Ethernet. We call it *Athernet*. As Figure 1 shows, a core component of *Athernet* is the mechanism to manage the transmission of each transceiver so that the communication medium can be shared efficiently, i.e., the Medium Access Control (MAC) protocol.

In order to finish this project and remaining projects, the following accessories are provided. Each group is eligible to borrow one set that includes:
1. Tee*3 https://item.jd.com/5005384.html
2. USB Sound Card*3 https://item.jd.com/1804882.html
3. Line*4 https://item.jd.com/1192674.html

Contact TAs to borrow the accessory set.

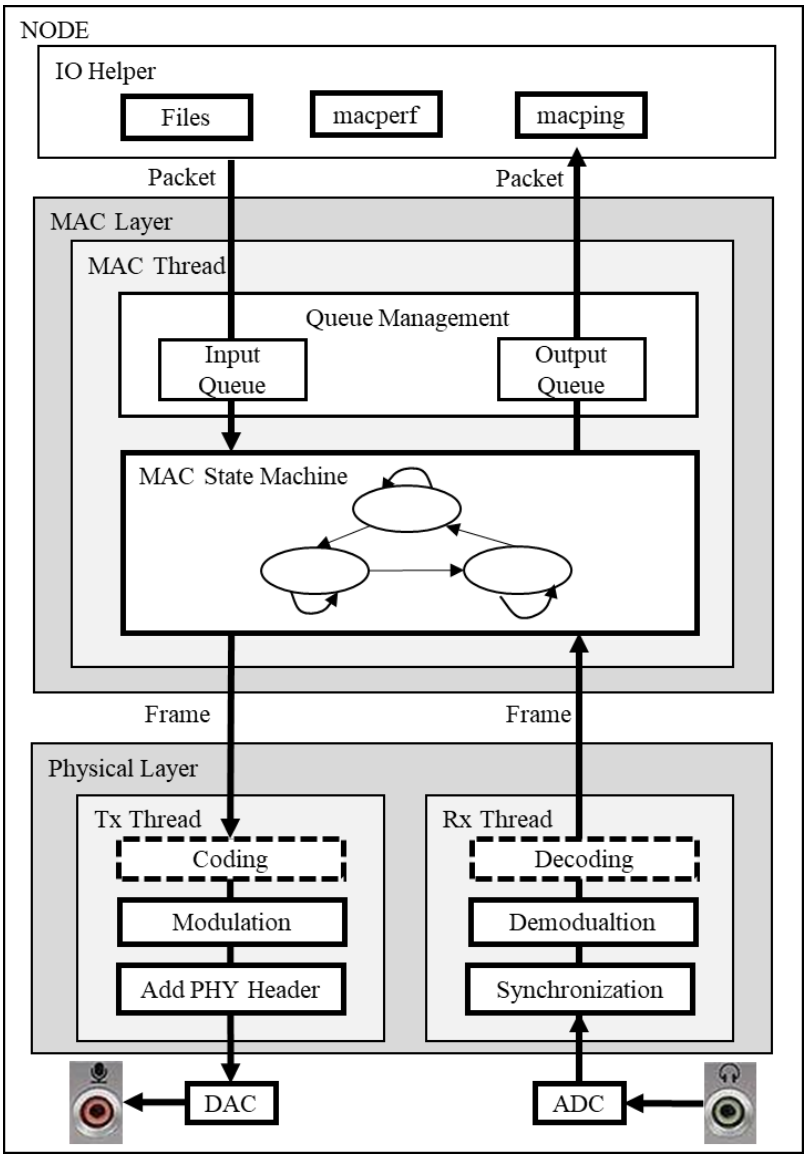

Figure 1 Project2 Overview

(Hierarchy of Tasks. Tasks are graded according to their hierarchy. A full score of one part automatically guarantees the full score of its subparts. In this project, Part 1 is a subpart of Part 2 and Part 3. Part 2 is a subpart of Part 3. The hierarchy is denoted as Part1<Part2<Part3.)

Part 1. (4 points) From Wireless to Wire

The main purpose of moving from wireless link to wired link is to reduce the heterogeneity and complexity in acoustic hardware, e.g., ringing effect from mechanical vibration, distortions from nonlinear amplifiers, uneven response, echos, etc. The method to establish a wired link between two nodes is connecting their 3.5mm analog audio interfaces, e.g., connecting the MIC port to the Speaker port (See the red line in Figure 2). Note that the signal flowing between the two ports is no longer an acoustic signal. It is an electric wave generated by the DAC.

The electric signal output from the Speaker port is normally used to stimulate the mechanical hardware, e.g., the speaker, to generate sound. On the other hand, the ADC in the MIC port will transform any the input electric signal into digital samples, no matter the signal is generated by a DAC from other device (red) or a microphone (black). Generally speaking, through the wired connection between DAC and ADC, the received signal contains much fewer distortions than the signal in Project1. Your code of Project1 should work seamlessly and very likely much better in the wired situation.

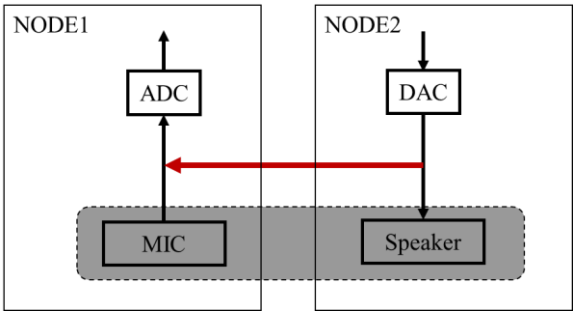

Figure 2 Wire Connection

In this part, your task is to set up the wired link between two nodes. This task is almost the same as Project1.Part3, but we expect higher performance (>5kbps).

Checkpoints:

The group provides two devices: NODE1 and NODE2, connected by audio cables.

CK(4 points). TAs provide a binary file “INPUT.bin” which contains 6250 Bytes. NODE1 sends bits according to this file. NODE2 stores the received bytes into a binary file “OUTPUT.bin”.

The transmission must be finished within 20 seconds.

Transmission Time	Points
[0, 10s]	100%
(10s, 15s]	80%
(15s, 20s]	50%
>20s	0%

TAs compare the difference between INPUT.bin and OUTPUT.bin:

Accuracy	Points
<80%	*0%
80% to 99%	*80%
>99%	*100%

Tips:

a. You are recommended to unplug the chargers/power adapters of your devices when doing wired experiments. The reason is that different AC-to-DC adapters (especially the ones without ground plugin) may have different voltages of their “ground”, so do the “ground” of the charging devices. When interconnecting two devices with wires, they may have different “ground”. In some cases, the problem just results in biased DC levels in the received signal from connected devices. Sometimes it may bring harmful damages. If you have to connect two devices with different “ground” (e.g., a charging laptop and a desktop), you should be very careful.

b. The throughput can be increased by modifying your physical layer (PHY) of project 1, e.g., decrease the time for each symbol. You can also develop a new PHY suitable for the wired situation.

c. You can design shorter preamble to reduce the header overhead.

d. You can use the audio card of your own devices. Sometime they have better performance (in terms of the quality of the generated/received signal) than the offered ones.

e. In some laptops and smartphones, the MIC in and speaker ports are combined into a single port, you may need an adapter to split them.

f. In project 1, we use carrier wave to convey information over the air wirelessly. In project 2, it is not necessary to so. You can directly use line code and transmit the baseband signal in the wire, which might be more efficient.

Part 2. (6 points) A Simple Reliable Link

To build a CSMA MAC from ground up, a good warm up is a simple send-and-pray protocol. A state machine is helpful in describing a protocol. The state machine of the send-and-pray protocol is shown in Figure 3. The node can only be in one of the states at any moment. The transition of the state is driven by some events. The arrows connecting states identify the event and the transition direction.

From the state machine in Figure, it is clear that the node switches between Tx and Rx states, meaning that the node cannot transmit and receive at the same time (i.e., half duplex). Supporting full duplex may require more complex designs. Half duplex is the assumption of *Athernet* nodes.

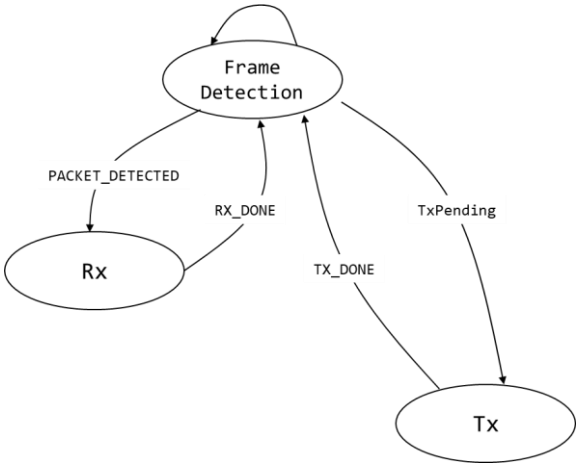

Figure 3 Send-and-pray State Machine

The goal of this task is to help you polish your physical layer code. Making a clear abstraction of the functionalities of the physical layer from project 1 helps a lot in implementing MAC protocols. For example, the following primitives are necessary ones that the physical layer interface must support for implementing the above simple protocol:

- Function: `PHYSend(MACFrame)`, is provided by PHY.Tx thread. MAC thread can call this function to send one MAC frame. The finishing of the transmission is notified by `TX_DONE` event.
- Event: `TX_DONE`, is issued by PHY.Tx thread, MAC thread can capture this event and process some necessary jobs (e.g., dequeuer of the FIFO). Then the state proceeds from Tx to FrameDetection.
- Event: `PACKET_DETECTED`. If a packet header is detected in PHY.Rx thread, the thread will issue this event. Once the MAC thread receives this event, it will switch to the Rx state.
- Buffer: `PHYRxFrame`, is shared between PHY.Rx thread and MAC thread. Once a frame is correctly received by the physical layer, PHY.Rx thread will put the frame into this buffer.
- Event: `RX_DONE`, is issued by PHY.Rx thread, it notifies MAC thread to take the frame in `PHYRxFrame` and proceed to FrameDetection state.

It is easy to extend the send-and-pray protocol to a more effective one: stop-and-wait ACK protocol. “Upgrading” the physical link in Project1 with ACK ability will provide reliability in data communication. A reference stop-and-wait protocol is shown in Figure 4.

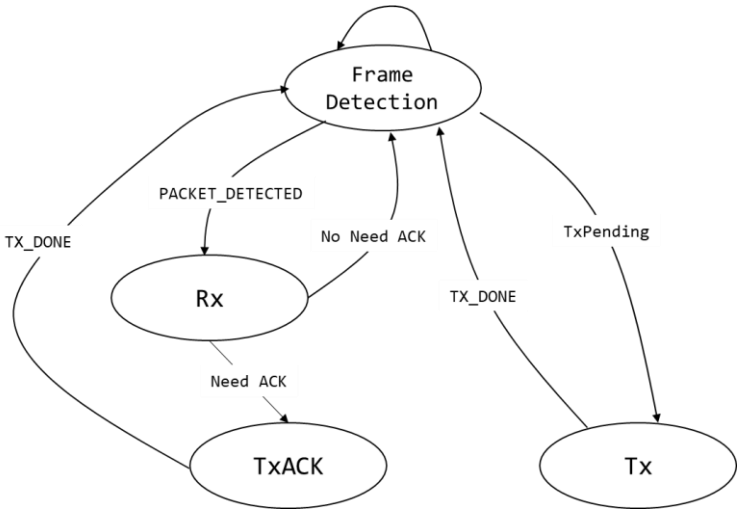

Figure 4 Stop-and-wait State Machine

The state machine contains more states and events than send-and-pray protocol. One obvious difference is that, after correctly receiving a frame, the node needs to reply an ACK immediately. Note that the state machine does not contain full details of this protocol. For example, when handling `Tx_DONE` event, MAC thread should set timeout counter for waiting ACK, and if the ACK is not received, the packet should be retransmitted.

Checkpoints:

The group provides two devices: NODE1 and NODE2, connected by two cables.

CK1(3 points). Similar to Part 1. TAs provide a binary file "INPUT.bin" which contains 6250 bytes. NODE1 sends bits according to this file. NODE2 stores the received bytes into a binary file "OUTPUT.bin".

The transmission must be finished within 20 seconds.

Transmission Time	Points
<10s	100%
[0, 10s]	100%
(10s, 15s]	80%
(15s, 20s]	50%
>20s	0%

TAs compare the difference between INPUT.bin and OUTPUT.bin:

<100%	*0%
100%	*100%

CK2(3 points). Redo CK2. TAs can unplug one of the wires, and the transmitter should be able to identify the event and display "link error" (according to retransmission times).

Tips:

a. You may want to use thread safe data structures in delivering data between threads.
b. The inter-frame time and time-out time should be carefully designed (refer to Lec5 and Lec6).
c. You may want to use a Tee to connect the signal

Part 3. (2 points) CSMA

In order to efficiently support multiple nodes in the shared communication medium, their transmissions should be carefully coordinated. There are many different MAC protocols. The one we choose in *Athernet* is CSMA, which is widely used in Ethernet and Wi-Fi.

CSMA Protocol. The intuition of CSMA protocol can be simply put as "listen before talk". The lecture and the textbook have covered it quite well. The state machine of CSMA protocol is shown in Figure 5. Check more information from references [1-5].

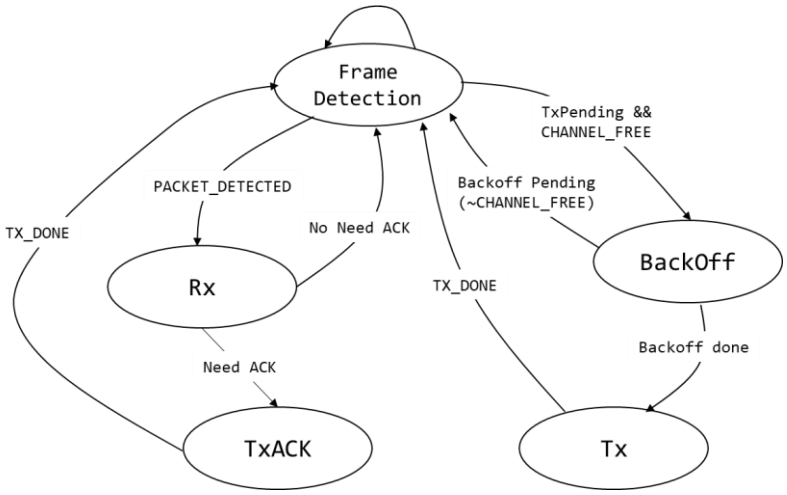

Figure 5 CSMA+ACK State Machine [1]

Frame Format. In order to support multiple nodes, each node must have a unique destination identifier, which contributes the destination and source address in MAC frames. Using 4 bits for each address field is suggested. A seq field is used to differentiate the frames. A type field is very useful for differentiating different kinds of frames, e.g., ACKs and data frames. Using 4 bits for type field is suggested. Finally, the maximum MAC payload must be limited, otherwise, the network may have large delay (see Part5).

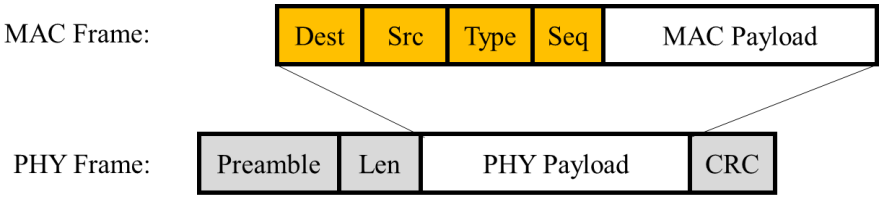

Figure 6 Frame Format

Network with Jamming. Using your accessories to connect 2 nodes like Part1, so the nodes are able to hear each other. Additionally, as shown in Figure 7, we introduce a jamming source to test the CSMA mechanism. Each node should be able to hear the output from the Jamming source, which intermittently plays high-power noise signals. The detailed parameters of the jamming source are coded in the `JammingWav.m`. Check them carefully to tune the parameters of the CSMA design. The key point is that, as CSMA is enabled in the two nodes, the communication should be able to work under the silent intervals of the Jamming source.

We go a little bit further about the Figure 7. First, the jamming signal is generated by a sound card. The signal is first split by a Tee and then merged with Rx signals of the nodes. The term "merge" means a mix of two electronic waves. In the ideal case when the DC levels of these nodes are the same, "merge" simply means sum or superposition. Second, another important implication of Figure 7 is that node1/2 can hear the signal generated by themselves. This is because the Tee is an undirected device. When the Tx signal by node 1 arrives at the Tee of node 2, it also is split and fed to the direction for the Jamming src. At the Tee of the Jamming src, that signal again split and final loop back to the Rx interface of node1. The same situation happens for the Tx signal of noted. I want to emphasize is that this is a feature rather than a bug of the topology. As when nodes can hear the transmission of themselves, there are no possibilities for full duplex implementations, which is why we use CSMA in early Ethernet and Wi-Fi.

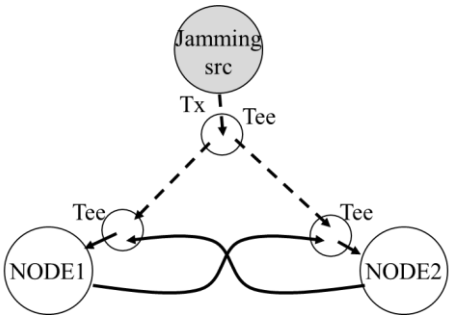

Figure 7 Network Topology

Checkpoints:
The group provides two devices: NODE1, NODE2 connected by wires.
TAs provide one Jamming source. The jamming source is generated by `JammingWav.m`.
The group provides the interface to connect the jamming source.
The topology is shown in Figure 7.

CK1(2 points). "INPUT1to2.bin" of 6250 Bytes and "INPUT2to1.bin" of 5000 Bytes are provided.
NODE1: transmits INPUT1to2.bin to NODE2, store the received file to OUTPUT2to1.bin
NODE2: transmits INPUT2to1.bin to NODE1, store the received file to OUTPUT1to2.bin
Jamming Src: play "Jamming.wav"

Jamming Src starts first. The transmission of NODE2 must be started immediately after the start of NODE1. The transmission of NODE2 must be finished before NODE1. The received files must be error free. The rank of Part7 is determined by the transmission time of NODE1.

The transmission of NODE1 must be finished within 120 seconds.

Transmission Time	Points
[0, 60s]	100%
(60s, 90s]	80%
(90s, 120s]	50%
>120s	0%

TAs compare the difference between INPUT*.bin and OUTPUT*.bin:

100%	*100%
<100%	*0%

CK2(2 points). Redo CK1. TAs can unplug one of the wires, and transmitter should be able to identify the event and display “link error”.

Part 4. (2 points) macperf Utility

`macperf` utility is used to measure the throughput between two nodes in *Athernet*. The utility is similar to the `iperf` utility, but it is specialized for *Athernet* and works in a different layer.
The working flow of `macperf` utility:

- `macperf` is invoked by executing `macperf dest_address`
- By default, `macperf` frame is generated with random MAC payload (choose your own payload size) and the type field of the MAC frame is set to type: `DATA`.
- The sender tries its best to send out `macperf` packets
- The sender counts and prints the throughput on the screen every one second.

Checkpoints:
The group provides two devices: NODE1, NODE2. The topology is shown in Figure 7 without the jamming source (the jamming source is disabled, but the tee of the jamming source still connects NODE1 and NODE2).

CK(2 points). TAs will check the functionality of `macping` utility in the following case:
NODE1: `macperf NODE2`
NODE2: `macperf NODE1`

TAs record the throughput of NODE1 and NODE2.

If NODE1 TH >1.0kbps && NODE2 TH>1.0kbps	100%
otherwise	0%

Part 5. (2 points) macping Utility

`macping` utility is used to measure the round trip delay between two nodes in *Athernet*. The utility is similar to `ping` utility in current operating systems, but it is specialized for *Athernet* and works in a different layer.

The working flow of `macping` utility:

- `macping` is invoked by executing `macping dest_address`
- A `macping` frame is generated with zero MAC payload and the type field of the MAC frame is set to type: `MACPING_REQ`.
- The `macping` frame is timestamped when sending into the PHY layer.
- The node with `dest_address` is responsible for automatically replying a frame targeting the sender and having its type field set to type: `MACPING_REPLY`.
- If the sender receives `MACPING_REPLY`, it calculates and prints the round trip delay on the screen.
- If the sender does not receive `MACPING_REPLY` after 2 seconds, the sender prints TIMEOUT on screen.

Checkpoints:

The group provides two devices: NODE1, NODE2.

The topology is shown in Figure 7 without the jamming source (the jamming source is disabled, but the tee of the jamming source still connects NODE1 and NODE2).

CK(2 points). TAs will check the functionality of `macping` utility in the following case:

NODE1: `macping NODE2`

NODE2: `macperf NODE1`

TAs record the RTT of NODE1 to NODE2 and the throughput of NODE2.

NODE1 RTT < 150 ms && NODE2 TH > 2.0kbps	100%
otherwise	0%

(Tasks with “Optional” tag are optional tasks. The instructor is responsible for checking and grading the optional tasks. Contact the instructor to check if you have finished one or more of them)

Part 6. (Optional + 2 points) Collision Detection

Network performance is determined by many factors. High throughput physical layer does not necessarily indicate high network performance. To further increase the efficiency of CSMA protocol, PHY.Tx thread can be aborted if PHY.Rx thread indicates high power, i.e., CSMA/CD.

Checkpoints:
Check settings are the same as Part3.
The throughput of NODE1 should be higher when collision detection is enabled.

Part 7. (Optional + 2 points) Performance Rank

Network systems always demand high performance, but high performance is hard to achieve. This task is to reward groups who will have achieved high throughput performance. In the first round, top 5 groups are selected according to their throughput performance in Part3. The second round will be held on the lecture, the final rank is determined by the performance in the second round.

Checkpoints:
Check settings are alomost the same as Part3.
TAs provide one Jamming source.
The group provides the interface to connect the jamming source.
The topology is shown in Figure 7.

CK(4 points). “INPUT1to2.bin” of 2MB and “INPUT2to1.bin” of 1MB are provided.
NODE1: transmits INPUT1to2.bin to NODE2, store the received file to OUTPUT2to1.bin
NODE2: transmits INPUT2to1.bin to NODE1, store the received file to OUTPUT1to2.bin
Jamming Src: play “Jamming.wav”

The transmission of NODE2 must be started immediately after the start of NODE1. The transmission of NODE2 must be finished before NODE1. The received files must be error free. The rank is determined by the transmission time of NODE1.

Rank	Points
1	100%
2	50%
3	25%
4	12.5%
5	0%

Reference and Useful Links

[1] Sora: High Performance Software Radio Using General Purpose Multi-core Processors https://www.usenix.org/legacy/events/nsdi09/tech/full_papers/tan/tan.pdf
[2] WARP CSMAMAC https://warpproject.org/trac/wiki/CSMAMAC
[3] WARP CSMAMAC Src
https://warpproject.org/trac/browser/ResearchApps/MAC/CSMAMAC/csmaMac.c
[4] NS3 MAC Low https://www.nsnam.org/doxygen/mac-low_8cc_source.html
[5] GNURadio MAC Report http://gnumac.wikispaces.com/file/view/final-project.pdf