WARP WIRELESS OPEN-ACCESS RESEARCH PLATFORM

Lab 5: uniMac

Chris Hunter & Patrick Murphy

Rice University

WARP Project

Document Revision 7

November 22, 2007

1 Introduction

In the previous lab, WARP nodes had no concept of addresses. The "MAC" was little more than software that directly transferred packets from Ethernet to the wireless PHY and vice versa. In this lab, we will implement a "true" medium-access layer to control nodes in the topology shown in Figure 1.

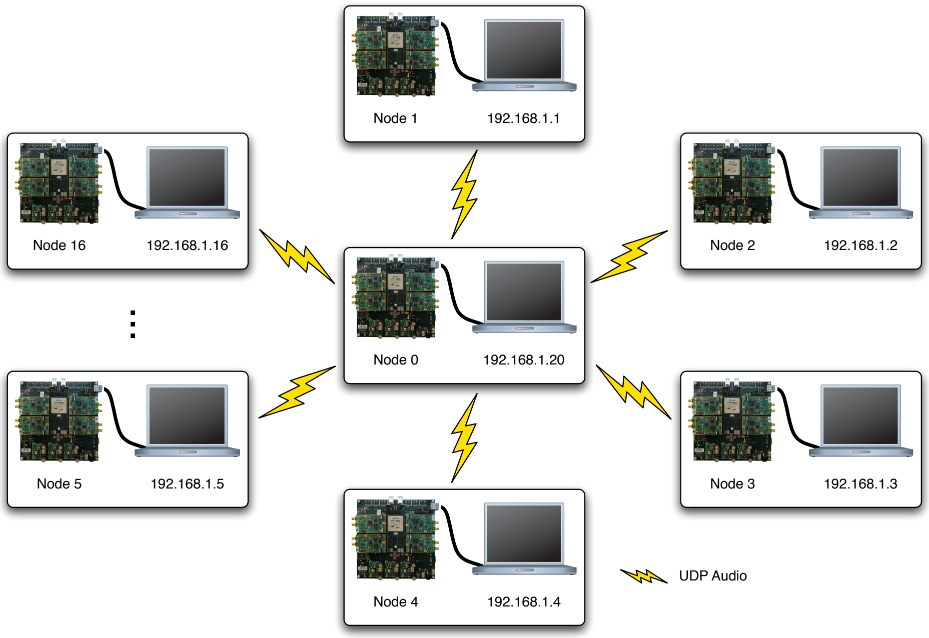


Figure 1: *Lab Topology*

In this experiment, a central transmitter will stream UDP audio to each of the participants' PCs. Each wireless node has a unique MAC address, allowing the node to process only its own packets. For the purpose of this lab, the local MAC addresses are set using the DIP switch on each WARP FPGA board; each node is already configured with a different DIP switch value.

The central node implements a routing table, mapping WARP MAC addresses to the attached PC's IP address. This node also implements a modified version of a CSMA MAC. The central node's MAC expects to receive an ACK for each packet it sends. It will re-transmit unacknowledged packets 8 times before dropping the packet entirely.

Note: All files are stored in `C:\workshop\userN\` where `userN` is your user login location. This location will be referred to as `.\` for the rest of the lab.

In this lab, you need to implement code which realizes the following behavior:

- Check the destination address of each received wireless packet. If it is addressed to your node, send an ACK to node 0 and send the packet via Ethernet.
- If a packet is received via Ethernet, the MAC should send it wirelessly to node 0. You do not need to implement the backoff/retransmit state machine.

This behavior is illustrated in Figure 2.

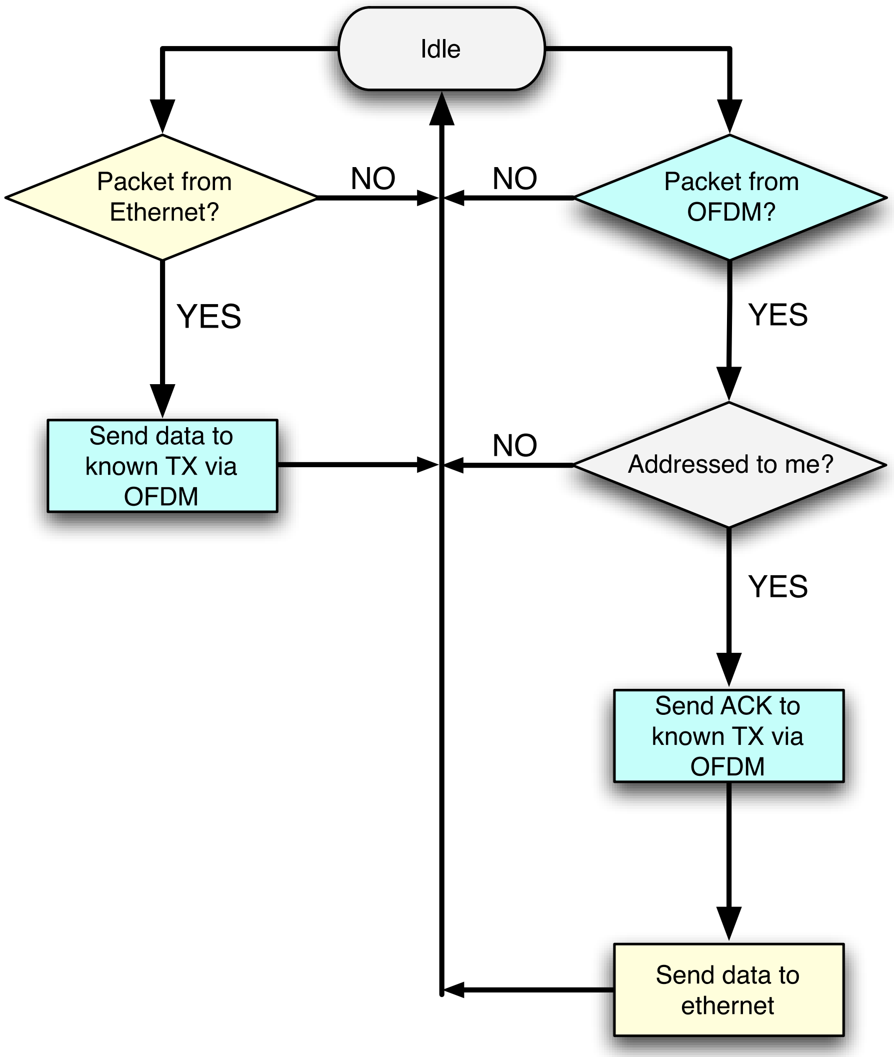


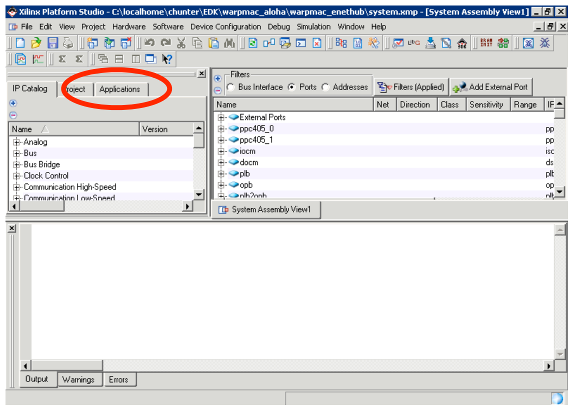
Figure 2: *uniMac State Diagram*

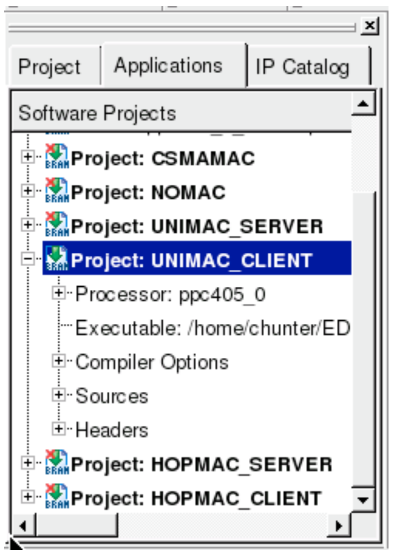
It is important to note that, while the transmitter has a MAC capable of retransmissions, your MAC implementation will not. Because of the unidirectional nature of UDP, the only Ethernet traffic which will be forwarded to node 0 is an ARP reply to establish the Ethernet MAC address lookup tables on both PCs.

The WARPMAC_API will be required throughout this lab exercise. Skeleton code is provided that should compile without user modification. By default, the project will blink user LEDs on the board upon the reception of good wireless packets. There are comments in the provided skeleton code which explain the code you must write to complete the MAC.

2 Instructions

1. Open the `.\Labs4_6_MAC\system.xmp` associated with this lab
2. Because we will only be dealing with the software project in this lab, we can ignore everything in the "System Assembly View." Click on the "Applications" tab as shown in the first plot of Figure 3
3. There are two software projects in this tab, as shown in second plot of Figure 3. The "UNIMAC_SERVER" project is provided for reference only; this is the code which is running on the central wireless node. You are responsible for modifying the "UNIMAC_CLIENT" project. By right-clicking on the projects, you can select which is selected for initialization by checking "Mark to Initialize BRAMs." In the interest of maintaining the custom network in this lab, please do not download the server project to the board.


Xilinx Platform Studio - C:\localhome\chunter\EDK\warpmac_aloha\warpmac_enethub\system.xmp - [System Assembly View1]
IP Catalog
Project
Applications
System Assembly View1
Output
Warnings
Errors



(a) **(b)**

Figure 3: XPS Window

4. Within the "Sources" and "Headers" hierarchies, you will be greeted by a number of files required for the project to build:
   - *uniMac.c* - This file is the where all of the user's modifications will take place. It is the top-level code where the MAC algorithm resides.
   - *warpmac.c* and *warpphy.c* - These file contains all of the MAC development and PHY interface framework. These frameworks provide the user high-level functions for abstracting interactions with the wireless and wired network interfaces.
5. Open the *uniMac.c* file and modify the skeleton code to implement the functionality specified in the comments.

3 Testing your MAC

Launch VLC Media Player, open the network stream as shown in Figure 4, click "OK," and finally un-mute the computer.

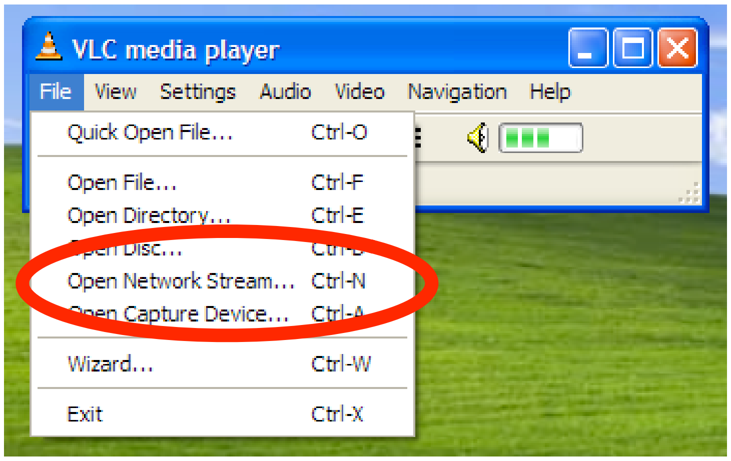


Figure 4: *Open the network stream and listen to the audio*

4 Advanced Extension: Digital Radio Receiver

Update your code to "steal" the audio streams of the other groups without changing the IP address of your computer. This extension is considerably more difficult to accomplish than the original, so it is much more freeform and "at-your-own-pace." In general, the idea is to filter through an audio stream other than the one meant for you at the MAC-address level. You would then need to overwrite the destination IP and Ethernet MAC addresses in the payload of the received packet with those of your computer and send it over Ethernet. This would effectively trick you computer into thinking that UDP stream was meant of it. However, there are many pitfalls to this extension:

- By overwriting payload elements, you have corrupted the UDP and IP checksums present. These also must be modified accordingly.
- The user should not acknowledge packets meant for someone else. ACKs would almost certainly collide if another user was listening to that stream

If the extension works, generalize it to use the pushbuttons to switch between "radio stations." This extension will require modification to *warpmac.c* and *warpphy.c*, so feel free to peruse and modify that code.