WIRELESS OPEN-ACCESS
RESEARCH PLATFORM

Lab 5: uniMac

Chris Hunter

Rice University
CMC Lab

Document Revision 1.1

March 22, 2007

1 Introduction

The WARP networking lab is intended to introduce the tools required for MAC development on WARP. In previous labs, code on the PowerPC was used to drive cores running in the FPGA fabric. We will make extensive use of the WARPMAC framework (http://warp.rice.edu/trac/wiki/WARPMAC). The framework will handle the hardware interaction, and we will program a basic MAC.

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.

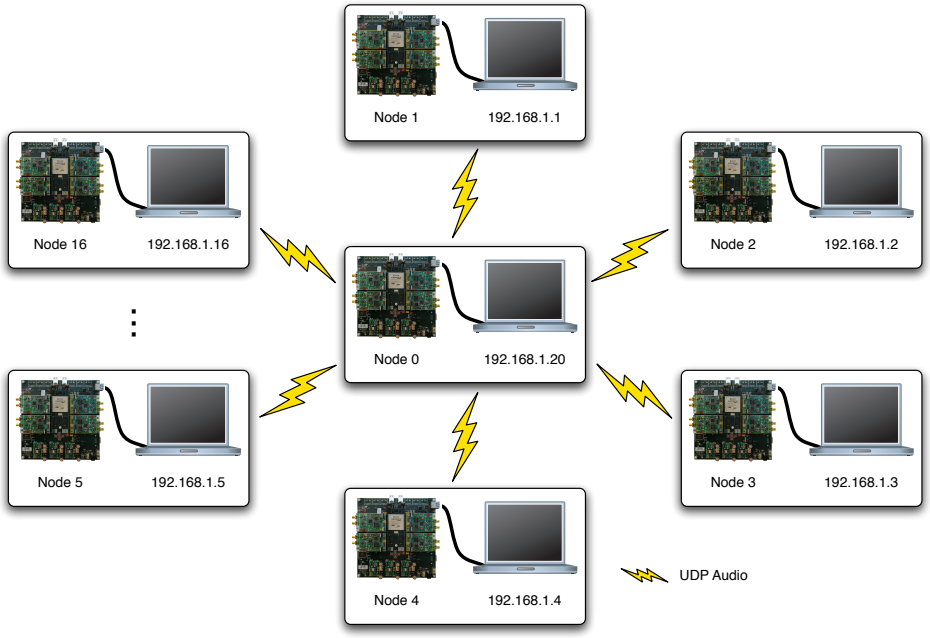

Figure 1: *Lab Topology*

In this experiment, a central transmitter will stream UDP audio to each of the participants' PCs. True addressing will take place, so a destination PC should only be able to access packets meant for that particular PC. To accomplish this, we will address nodes using the dip switches located on the FPGA boards. The value read from the switches will serve as an index into a default routing table that is known by all nodes. The value read from the switch is the decimal representation of the binary pattern. An example is shown in Figure 2.

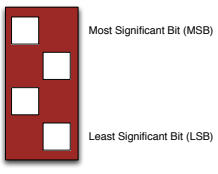

Figure 2: *The value '5' is read by the PowerPC for this configuration*

The central transmitter, Node 0, will be provided for the exercise. It implements a slightly modified version of the CSMA MAC implementation. To increase the reliability of each link, the transmit node will retransmit packets until the participants' PCs send acknowledgments back or until a maximum number of retransmits is achieved.

The user's code is responsible for the following behavior:

- If a received packet was addressed to the user's node, the MAC should first generate an ACK packet and and transmit it back to Node 0. Second, the MAC should send the packet over ethernet like the previous lab.
- If there is a packet to transmit from Ethernet, the MAC should send it to Node 0.

These points are summarized in the flowchart in Figure 3.

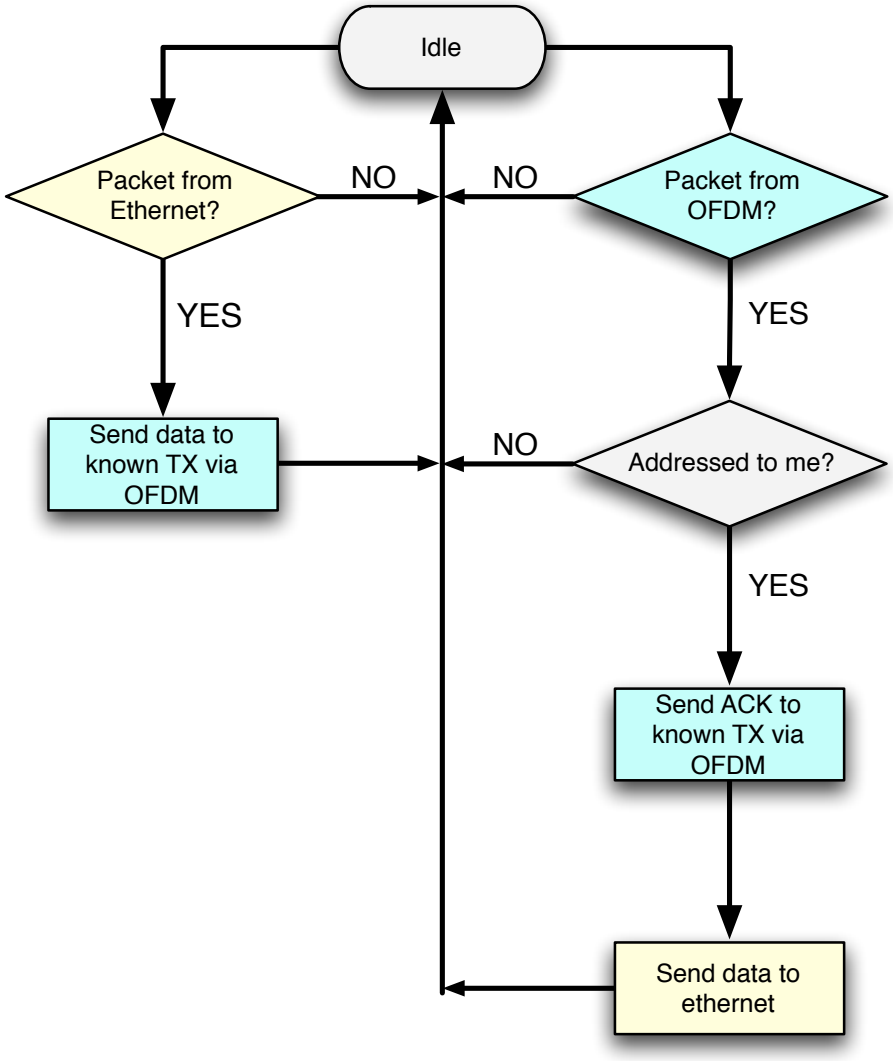

Figure 3: *uniMac State Diagram*

It is important to note that, while the transmitter has a MAC capable of retransmissions, the user's MAC will not. Because of the unidirectional nature of UDP, the only traffic that needs to be send to Node 0 is an ARP reply to establish the ARP table on both computers.

The WARPMAC_API will be required throughout this lab exercise. Skeleton code will be given that should compile without user modification. By default, the project will blink user LEDs on the board upon the reception of packets that pass the checksum. Within the given code will be comments describing the code that needs to be written.

2 Instructions

1. Open the `C:\workshop\lab5\warpmac_uniMac\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 4
3. There are two software projects in this tab, as shown in second plot of Figure 4. The "server" project is provided for reference only. It is the source behind used in the transmit project. You are responsible for modifying the "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.

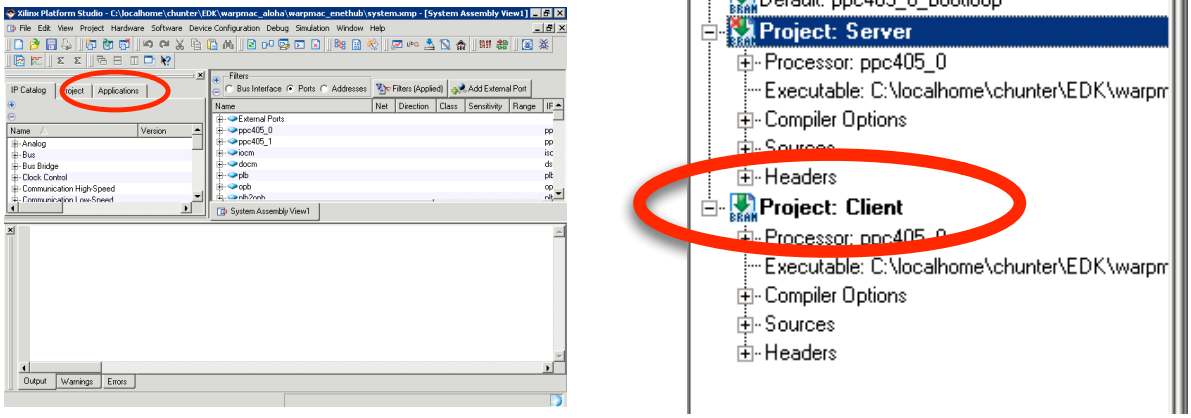

Figure 4: XPS Window

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 *warpmac.h* - These file contains all of the MAC development framework. They give the user high-level functions for abstracting hardware interactions.
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 5, click "OK," and finally un-mute the computer.

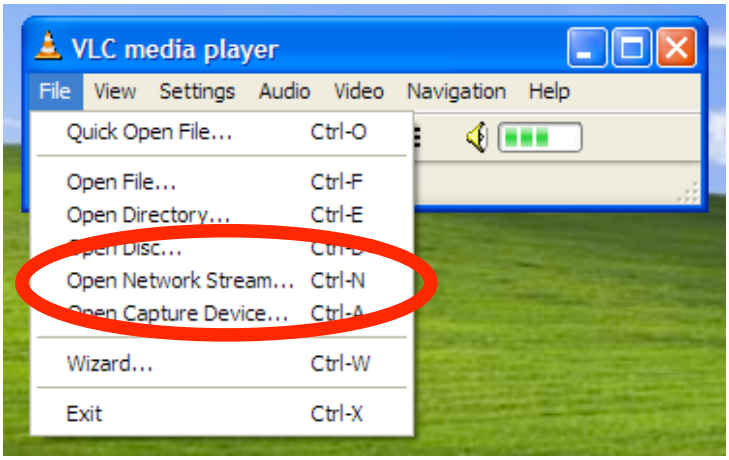

Figure 5: *Open the network stream and listen to the audio*

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."