WIRELESS OPEN-ACCESS
RESEARCH PLATFORM

Lab 6: hopMac

Chris Hunter

Rice University

WARP Project

Document Revision 6

November 3, 2007

1 Introduction

In the previous lab, a central transmitter streamed UDP audio to each of the participants' PCs in a topology shown by Figure 1.

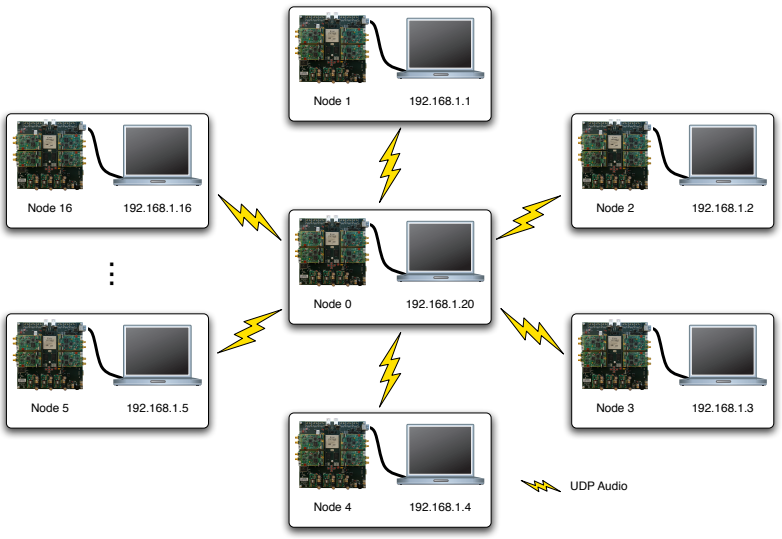

Figure 1: *Lab Topology*

This lab will have an identical configuration. The twist here, however, is that the transmitter will periodically hop to a different center frequency (numbered 1 through 11). Before doing so, it will broadcast a special "HOPPACKET" packet to notify users that the transmitter is about to hop and to what channel. If your MAC algorithm is coded properly, your node should hop frequencies to follow the central transmitter in its journey through the spectrum. At the application level, the audio stream should be maintained, and the frequency hopping should be indiscernible.

The user's code is responsible for all of the behavior of the previous lab. You can use your completed code, or you can use the completed code provided. Additionally, you are responsible for the following behavior:

- If the packet was sent to the wireless broadcast MAC address (255:255:255:255:255:255), and it was a "HOPPACKET" type, switch to the channel specified in the packet header. We use the Macframe.header field 'reserved1' to hold this value. The critical elements of the HOPPACKET Macframe are shown in Figure 2. Refer to the WARP API documentation (http://warp.rice.edu/WARP_API) for details on the full MacFrame structure definition.

```
Macframe FollowMe;

FollowMe.header.pktType = HOPPACKET;
FollowMe.header.fullRate = 2;
FollowMe.header.reserved1 = NewWirelessChannel; //This is your target channel
FollowMe.header.length = 0; //Payload length, not including header
FollowMe.header.destAddr = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
 ...
```

Figure 2: *The HOPPACKET Macframe structure.*

The full software state of the client node is shown in Figure 3.

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 which compiles without user modification and implements the behavior of the previous lab. By default, the project will blink user LEDs on the board upon the reception of packets that pass the checksum. The skeleton code contains helpful comments to guide your implementation.

2 Extensions

- Gather statistics on a per-channel basis. Can you infer channel conditions using aggregate statistics at the MAC layer?
- If your node misses the HOPPACKET packet, it will wait in its channel until the transmitter happens to jump back in to that channel. Try to infer when this occurs by listening for packets. If no packets have been received for some time, assume you lost sync. Can you sweep through the spectrum and try to lock back on? The PowerPC natively supports some timing functions, but the resulting code will be a lot easier if you use the *ofdm_timer* custom peripheral in the model. See how *csmaMac.c* and *warpmac.c* handle the use of that core.

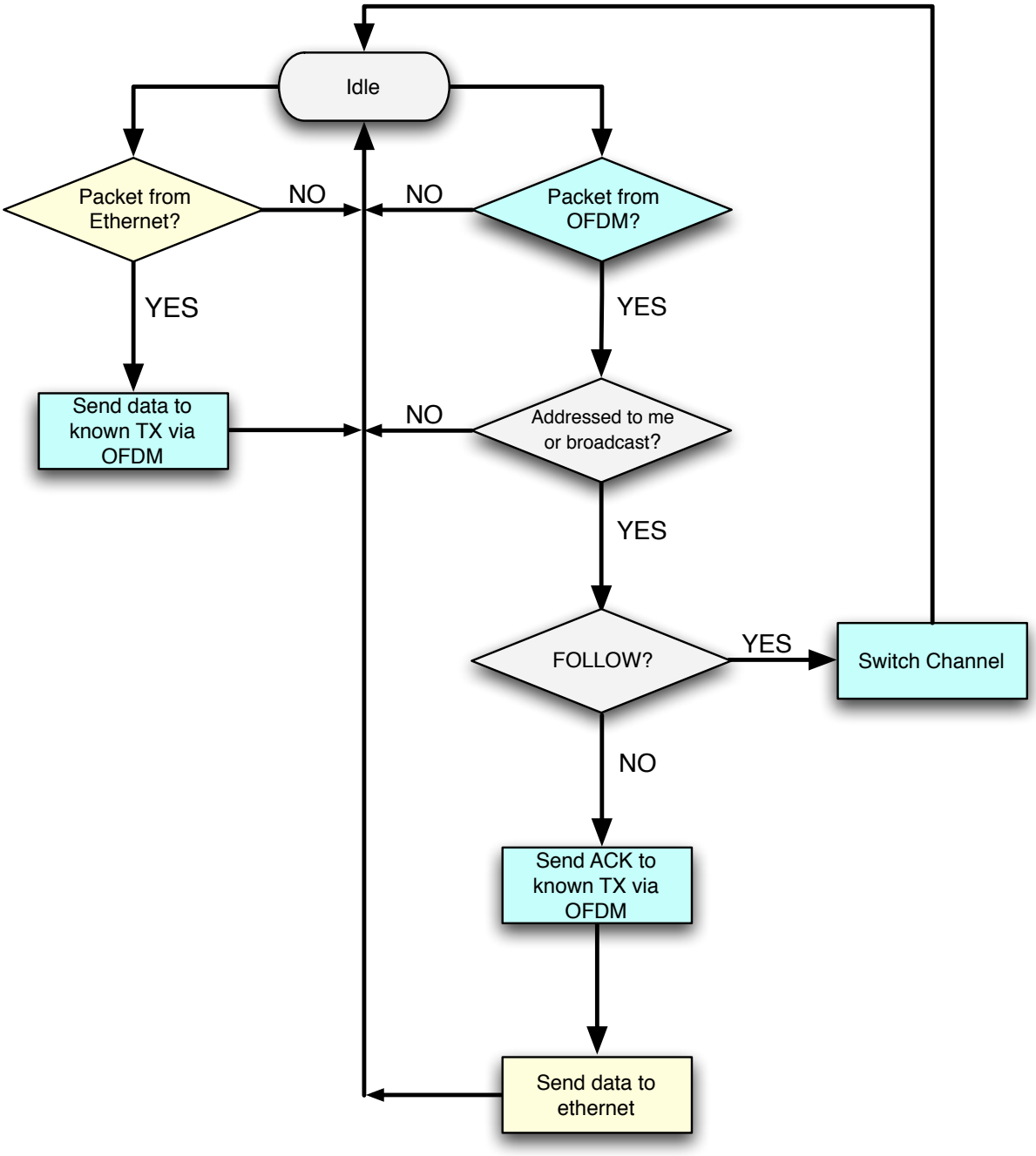

Figure 3: *State diagram for frequency-hopping MAC*