Homework 3: ping in XINU

Due: November 4th, 23:59 Central time

To be completed in teams of two

Please submit a compressed file with your modified xinunets directory at the corresponding D2L dropbox.

INTRODUCTION

In this assignment, you will be implementing the `ping` utility for the XINU operating system. XINU is a small OS that has been successfully used for educational purposes and for some commercial embedded systems. If interested in the history of it, refer to Dr. Comer's XINU page. For this assignment, we will be building upon a XINU kernel version that Dr. Brylow and his team ported to the Linksys routers. These routers are located in the Systems lab. If interested in the progress of the XINU project at Marquette, check out Dr. Brylow's embedded XINU page.

The resources you will need for this assignment are:

1. The OS implementation files that you will be expanding: xinunets.tar.gz (located under Files of this channel)
2. Documentation for the corresponding Embedded XINU version

BUILDING AND RUNNING XINU

You need to transfer the providednxinunets files over to morbius and work from there.

To build the XINU operating system, perform the following steps:

1. Change directory into the subdirectory `compile`. This directory contains the XINU project `Makefile`, and is where all of the compiled ".o" files will go.
2. Execute the command:

   ```
   make clean
   ```

   By standard convention, almost all Makefiles include a target called `clean` that removes everything except the source code. The tar-ball you unpacked already should be clean, but it never hurts to make sure that you are starting from a clean state. You may find yourself using this command often.
3. Execute the following command:

   ```
   make
   ```

This should produce several lines of output as each source file is compiled, and the resulting object files are linked together to form the operating system, a simple set of library functions, and the boot loader. If all goes as it should, you should find the directory full of `.o` files from all of the source code in the other subdirectories, and most importantly, a newly compiled operating system image called `xinu.boot`.

Your XINU image is now ready to be run on a backend machine (i.e. a Linksys router). To transfer it there, we have a special utility called **`mips-console`**. Execute `mips-console` in the compile directory where your `xinu.boot` file resides. `Mips-console` will connect your terminal to the first available backend machine, and you should see a message like:

connection 'voc', class 'mips', host 'morbius.mscs.mu.edu'

depending on which backend you get. This will be immediately followed by a stream of automated commands as the embedded target system boots, configures its network settings, and uploads your `xinu.boot` kernel.

The mips-console is *modal* (like the vi editor). You start out in direct connection mode, in which your terminal connects directly through special hardware to the serial console on your backend machine. To get out of mips-console, hit Control-Space, followed by the 'q' key.

The backend machines in the lab have private IP addresses. You can use the following IP addresses to test your code:

192.168.6.10

192.168.6.20

192.168.6.21

ASSIGNMENT OBJECTIVES

Building upon the provided Embedded Xinu kernel, implement a basic ICMP component (with IPv4 headers) for the system that supports echo request (ping) and echo reply. Your component should feature the following:

- A shell command "ping" that initiates a sequence of echo requests and displays corresponding echo replies.
- A mechanism for answering incoming ICMP echo requests with echo replies.

Your `ping` command will need to resolve the MAC address of the destination IP address using the `arpResolve` function provided (see `network/arp/arpResolve.c`). Also, note that a checksum function is already implemented (see `include/ether.h`)

Happy Coding!