CS444/544 Tutorial
Programming Project 1 - vdump
Due: Monday, January 23 (midnight)

For your first network programming project, you will write a packet printing program called *vdump* that reads and prints raw ethernet packets. Your program must be written in "C/C++" under Unix.

Because we don't have ready access to physical networks, we're going to be using a virtual network library called `vrouter`. It will act very much like a real network with real network interfaces.

Access to this library is through the following routines:

1.
```
typedef struct _lan {
        char lname[6];      // lan name : "eth0" - "eth15"
        char hwaddr[18];    // of the form : "aa:bb:cc:dd:ee:ff"
        struct _lan *next; // if we want to be in a linked-list
} lan;
typedef struct _lanlist {
        int cnt;           // number of elements in this list
        struct _lan *lan; // pointer to the first guy
} lanlist;
lanlist* whichlans(void);
```

   This routine returns a list of the network interfaces on the machine, including the name of each interface and its hardware address. It returns NULL if it fails.

2.
```
typedef struct _INTERFACE INTERFACE;
INTERFACE* openinterface(char *lname, int group);
```

   This routine opens a network interface and returns a handle to it. The `lname` is the name of the interface, as returned by `whichlans()`. The group number is used to separate networks between students. Each group number has its own packets. It returns NULL if it fails.

3. `int readpacket(INTERFACE *iface, char *buf, int len);`

   This routine reads one packet from the interface specified into the buffer `buf` (of maximum size `len`). It returns the number of bytes in the returned packet or -1 if it fails.

4. `int closeinterface(INTERFACE *iface);`

   Call this routine when you're finished using an interface.

5. `export VROUTER_DEBUG=9`

   Setting this environment variable to a value higher than 0 will enable debugging information from the vrouter library.

Your program must support the following options:

-d
To print debugging information (more detailed output)

-l
List all interfaces (names and hardware addresses) and exit

-g NUM
Use group NUM. By default, you should use group 0.

-i IFACE
Use interface IFACE. By default, you should use the first interface returned by `whichlans()`.

–hwsource SADDR
Only show packets from source HW address SADDR. See the "Filtering" section below for more information. By default, your program should assume `"*:*:*:*:*:*"` (match all packets).

–hwdest DADDR
Only show packets going to HW address DADDR. See the "Filtering" section below for more information. By default, your program should assume `"*:*:*:*:*:*"` (match all packets).

-h Print a quick summary of the command line arguments and exit. This information should also be printed if you don't understand one of the command line arguments.

The output from your program should look like this:

```
BSH:picard> ./vdump -g 0
Using group 0
Using interface 'eth1'
Using source filter '*:*:*:*:*:*'
Using destination filter '*:*:*:*:*:*'

packet 1 - length 68 bytes:
  Eth Dest: 00:0d:93:af:d3:dc
  Eth Src:  08:00:20:d9:01:87
  Eth Type: 0x800
  Eth Data:
       45 00 00 3c 00 5c 40 00 40 06 29 63 84 eb 03 9a 84 eb 03 8d
       13 89 de 71 fb 8b 2c 8b b2 2d d5 88 a0 12 c0 50 a5 a1 00 00

packet 2 - length 66 bytes:
  Eth Dest: 08:00:20:d9:01:87
  Eth Src:  00:0d:93:af:d3:dc
  Eth Type: 0x800
  Eth Data:
       45 00 00 34 02 bf 40 00 40 06 27 08 84 eb 03 8d 84 eb 03 9a
       de 71 13 89 b2 2d d5 88 fb 8b 2c 8c 80 10 ff ff 11 24 00 00
```

For each packet, you should print the ethernet source and destination in standard form, the ethernet type field in hexadecimal, and also print the first 40 bytes of the ethernet data (in hex, as above).
At a minimum, your program should contain:

Filtering

Your vdump program should be able to filter the packets it captures based on source and destination hardware address. To support this, the vdump program needs the "–hwsource" and "–hwdest" options, which tell vdump to show packets coming from and going to a certain hardware address.

In order for a packet to be displayed, both the source **and** destination filters must match the Ethernet source and destination fields of the packet, respectively.

For example, "`--hwsource 00:12:34:56:78:9A`" would tell vdump to show packets with a **source** hardware address of `00:12:34:56:78:9A`. On the other hand, "`--hwdest 00:12:34:56:78:9A`" would tell vdump to only show packets with a **destination** hardware address of `00:12:34:56:78:9A`.

In addition to supporting filtering by a single hardware address, your vdump program must support the use of wildcards in the hardware address being captured. For example, "`--hwsource 00:*:34:56:78:9A`" would mean that packets with source hardware addresses of "`00:12:34:56:78:9A`", "`00:34:34:56:78:9A`", and "`00:4C:34:56:78:9A`" would all be displayed.

Note that a filter of "`*:*:*:*:*:*`" matches all addresses.

A few more examples:

Filter "`12:34:56:78:9A:BC`" does not match "`78:64:92:78:9A:BC`"

Filter "`12:34:56:78:9A:BC`" matches "`12:34:56:78:9A:BC`"

Filter "`1:2:3:a:b:c`" matches "`01:02:03:0A:0B:0C`" (watch leading zeroes and case!)

Filter "`*:34:78:*:9A:BC`" does not match "`12:35:3C:9A:9A:BC`"

Filter "`*:34:78:*:9A:BC`" matches "`12:34:78:9A:9A:BC`"

Modularity

You should have separate procedures to print different information, read the packets, etc. You'll be using the routines from this program in later projects!

Structures

You should define your own structure for the Ethernet header. You may copy the standard header definitions if you wish, but it **must** be defined in the files that you turn in.

Error Checking

You must error check all arguments, system calls, vrouter library calls, and etc. Be **very** sure to sanity check any arguments provided along with the –hwsource and –hwdest options.

Hints:

support files

All of the files that you need for this assignment are on the web here:

http://oucsace.cs.ohiou.edu/ osterman/protected/archives/cs444.archive/assignments/vdump/

or on prime here:

`/home/osterman/archives/cs544.archive/assignments/vdump`

start with the files (`Makefile`, `vdump.cc`) in the `headstart` directory there.

packets

Your virtual network won't have any packets on it unless you put them there. You should use the program `vr_pickup` (in the class archive for this assignment) to put packets onto your network. You can run it from any machine in the lab and it takes the same "-g" option as your program to specify the group. With no other arguments, it grabs "appropriate" packets from the real network and puts them onto your virtual network. With the "`-r packets.dmp`" argument, it will read packets from the file and put them onto your virtual network.

Work in Stages

Start by getting the argument parsing and debugging code to work (to print the arguments). Then add the "-l" option. Then get `openinterface()` working and say "I got one" every time `readpacket()` returns something. Next, write a subroutine to print the raw ethernet packets. Finally, add filtering code into the project and make sure that works.

Your executable program should be called *vdump*. You may add more command line options if you prefer, and make the program more powerful if you want. However, for grading, the command "*./vdump -g GROUP -i INTERFACE*" must work. I will give you the "official grading input files" later this week. Your submitted program must include a `Makefile`, and the command `make` must generate the program `vdump`.