Appendix K

Sample program using libpcap

```
#include <stdio.h>
#include <stdlib.h>
#include <pcap.h>            /* if this gives you an error try pcap/pcap.h */
#include <errno.h>
#include <time.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>

/*
   Compile with: gcc pcaprawpkt.c -o pcaprawpkt -lpcap

   struct pcap_pkthdr {
        struct timeval ts;       time stamp
        bpf_u_int32 caplen;      length of portion present
        bpf_u_int32 len;         length this packet (off wire)
};
*/

#define BUFFSIZE   1500

void print_timestamp(const struct pcap_pkthdr *hdr)
{
  struct tm *timeptr;

  timeptr = localtime( (const time_t*) &hdr->ts.tv_sec);

  printf("%d:%d:%d.%d ", timeptr->tm_hour, timeptr->tm_min, timeptr->tm_sec, hdr->ts.tv_usec);
}
```

```
void print_MAC_address(u_char *ptr)
{
  int i;

  /* MAC address is six octets or six bytes, so i should begin from 1 in
     order to print the first five octets with ":" as separator. */
  for (i=1; i < ETHER_ADDR_LEN; i++) {

    printf("%.2x:", *ptr++);

  }

   printf("%.2x\n", *ptr);
}

void print_type(int type)
{

  switch(type){
  case ETHERTYPE_IP:
    printf("Ethernet type hex:0x%.4x is an IP packet\n", type);
    break;
  case ETHERTYPE_ARP:
    printf("Ethernet type hex:0x%.4x is an ARP packet\n", type);
    break;
  default:
    printf("Ethernet type hex:0x%.4x not IP", type);
  }

}

static void print_packet(u_char *user, const struct pcap_pkthdr *hdr, const u_char *packet)
{
  struct ether_header *eptr;  /* net/ethernet.h */

  print_timestamp(hdr);


  printf("Received packet of length %d and capture length %d on device %s\n", hdr->len, hdr->caplen, user);

  eptr = (struct ether_header *) packet;

  printf("Destination Address: ");
  print_MAC_address(eptr->ether_dhost);

  printf("Source address: ");
  print_MAC_address(eptr->ether_shost);

  print_type(ntohs(eptr->ether_type));
  printf("\n\n");

}
```

```
int main(int argc, char **argv)
{
  char *device;
  char errbuf[PCAP_ERRBUF_SIZE];
  int promiscuous = 1;            /* 1 = promiscuous; 0 = is not */
  pcap_t* descr;

  device = pcap_lookupdev(errbuf);

  if(device == NULL) {
    printf("%s\n", errbuf);
    exit(1);
  }

  printf("Device: %s\n", device);

  descr = pcap_open_live(device, BUFFSIZE, promiscuous, 0, errbuf);

  if (descr == NULL) {
    printf("pcap_open_live(): %s\n",errbuf);
    exit(1);
  }

  pcap_loop(descr, -1,  print_packet, device);

  pcap_close(descr);

  return 0;
}
```