1. Introduction

1.1. What is ICMP Ping Replies?

Definition of Internet Control Message Protocol (ICMP) - ICMP is the required part of any IP implementation process. It is also a diagnostic utility and reports errors. It is a required part of any IP implementation process. It is essential to understand how ICMP works and what it can possibly generate in a specific type of ICMP.

ICMP’s are used by routers, intermediary devices, or hosts to communicate updates or error information to other routers, intermediary devices, or hosts.
**Definition of Ping** - Ping is a utility tool used to test network connectivity. Ping uses the Internet Control Message Protocol (ICMP) to send out packets to the target host. The ICMP packets sent to the host are called echo_requests. The host who receives that packet could then send the packet back called echo_response.

Image 1 shows a structure of a packet. Occupying 8 bytes, the first section signify if the package is an echo_request (8) or an echo_response (0). The second section also contains 8 bytes and that it always contains the code zero (0). The next section contains 16 bytes that checks the header's sum. The next section, also containing 16 bytes and checks if the second section equals to 0. This is then used as an identifier to assist in matching the echo_request with the echo_response. Similar to the previous section of the packet, the next packet also contains 16 bytes and assist in matching the echos and replies. The last section contains a variable size of data that is used to ensure the correct size of the packet.

![Image 1: Echo Reply Header](http://www.firewall.cx/networking-topics/protocols/icmp-protocol/152-icmp-echo-ping.html)

2. How to implement ping replies?

2.1. Step-by-Step Overview

**Setup step:**

Include proper headers built in linux:

```c
#include <linux/ip.h>
#include <linux/icmp.h>
#include <linux/tcp.h>

// Helpful for differentiating the ping section
#define IMPLEMENT_PING
```

**OPERATING SYSTEMS AND SYSTEMS PROGRAMMING** | Professor Franco 2
1. **Inside of the ip header operations detect ICMP protocol for socket buffer:**
   
   *At the top of the function, declare the following variables:

   ```
   #ifdef IMPLEMENT_PING
   struct iphdr *ih;
   struct icmphdr *icmph;
   #endif
   ```

   *Right after you declare all of the variables, include the following code to create a special ICMP header:

   ```
   #ifdef IMPLEMENT_PING
   ih = (struct iphdr *)(skb->data + sizeof(struct ethhdr));
   if(ih->protocol == IPPROTO_ICMP){
       printk(KERN_INFO "Recieved ICMP ip header\n");
       // Get the ICMP header from the socket buffer
       icmph = icmp_hdr(skb);
       if(icmph->type == ICMP_ECHO){
           memcpy(eth->h_source, dev->dev_addr, dev->addr_len);
           memcpy(eth->h_dest, eth->h_source, dev->addr_len);
           return dev->hard_header_len;
       }
   }
   #endif
   ```

2. **Inside of the start transmit function, modify the OS packet slightly to generate a the correct contents of the ICMP ping reply:**

   *At the top of the function, declare the following variable:

   ```
   #ifdef IMPLEMENT_PING
   struct icmphdr *icmph;
   #endif
   ```

   *Right after the declaration, implement the following lines of code. It takes care of sending the skb packet containing the ICMP reply:

   ```
   #ifdef IMPLEMENT_PING
   if(ih->protocol == IPPROTO_ICMP){
       printk(KERN_INFO "Recieved ICMP ip header\n");
   }
   #endif
// Get the ICMP header from the socket buffer
icmph = icmp_hdr(skb);

if(icmph->type == ICMP_ECHO){
    printk(KERN_INFO "ICMP is an Echo\n");

    // Switch the type to ECHOREPLY
    icmph->type = ICMP_ECHOREPLY;

    // Switch fourth octet of ip header
    printk(KERN_INFO "start_transmit - switching fourth octet\n");
    ((u8*)saddr)[3] = 2;
    ((u8*)daddr)[3] = 1;

    // Add the eth protocol and dest. addr to the packet
    printk(KERN_INFO "start_transmit - Setting ethernet destination\n");
    memcpy(eth->h_dest, dev->dev_addr, dev->addr_len);
    skb->dev = dev;
    skb->protocol = eth_type_trans(skb, dev);

    // Recompute the checksum
    printk(KERN_INFO "start_transmit - Calling ip_fast_csum\n");
    ih->check = 0;
    ih->check = ip_fast_csum((unsigned char *)ih, ih->ihl);

    printk(KERN_INFO "start_transmit - Sending SKB containing ARP reply\n");
    netif_rx(skb);
    return NETDEV_TX_OK;
}
#endif

3. Application to the Course

1. ICMP Ping Replies applies to Lab 7: Net Device Driver
2. The implementation was completed by Jacob Chesley, a 4th year student in the course during the Fall of 2014.
3. The source code he implemented is archived by Professor Franco.
3. References

http://www.firewall.cx/networking-topics/protocols/icmp-protocol/152-icmp-echo-ping.html

http://support.microsoft.com/kb/170292