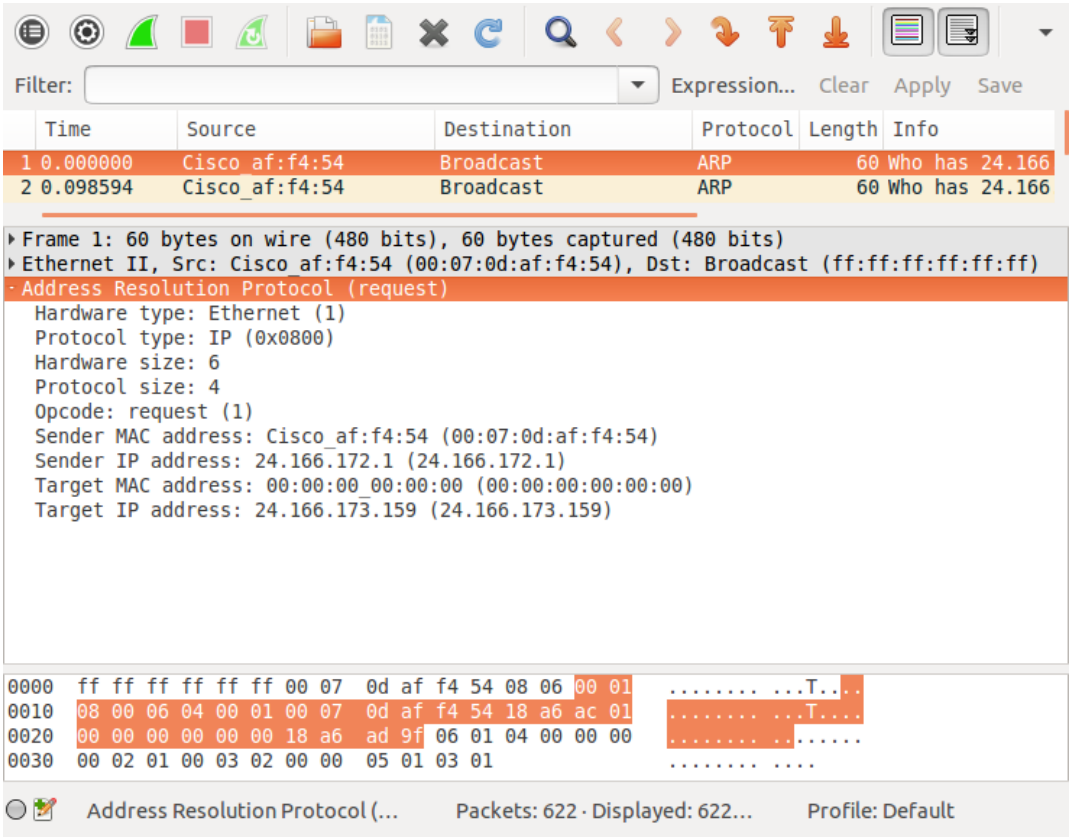
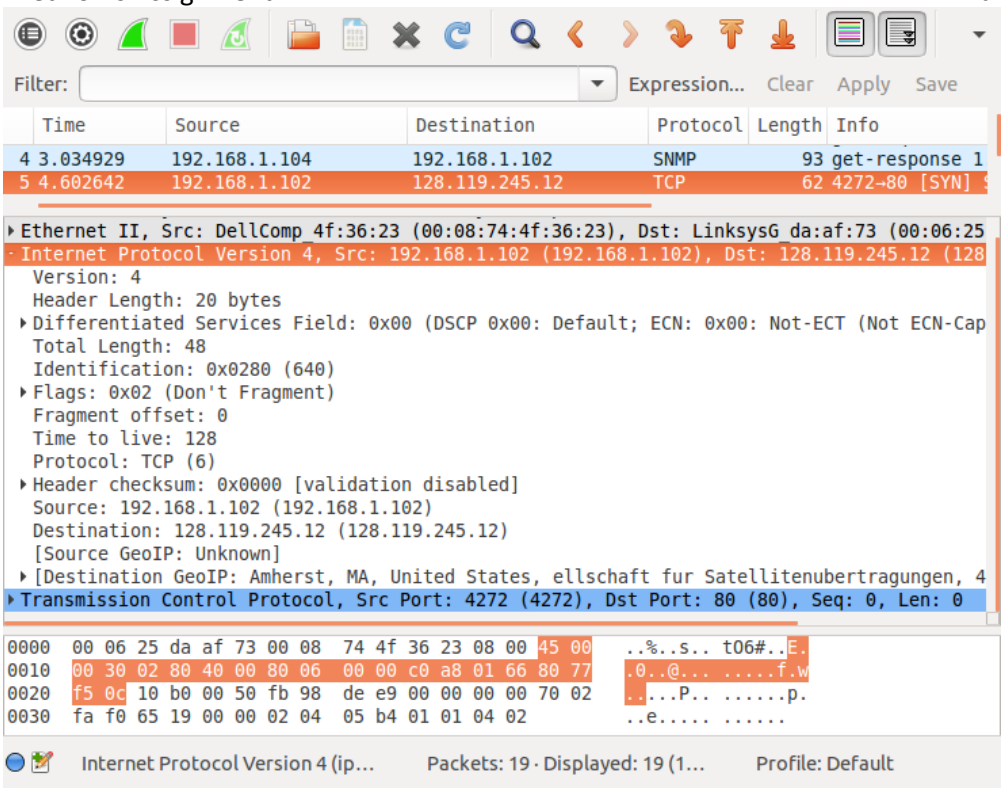
Computer Networks Assignment #4 Zhang Zhexian (0545080) zhangzhexian@outlook.com

1. [Chapter 4 Hands-on 2] Use Wireshark or similar software to capture packets for couple of seconds. Find an ARP packet and an IP packet from the data you have captured. Compare the difference between the MAC header of these two packets. Can you find the protocol ID for ARP and IP? Is the destination address of the ARP packet a broadcast address or a unicast address? Is this ARP packet a request or reply packet? Examine the payload of this ARP packet.

   This is the ARP packet, the MAC header is Cisco_af:f4:54, the protocol ID is 0x0800 (type IP), the destination is a broadcast address, the packet is a request packet, and the payload is short that starts with all 1's.

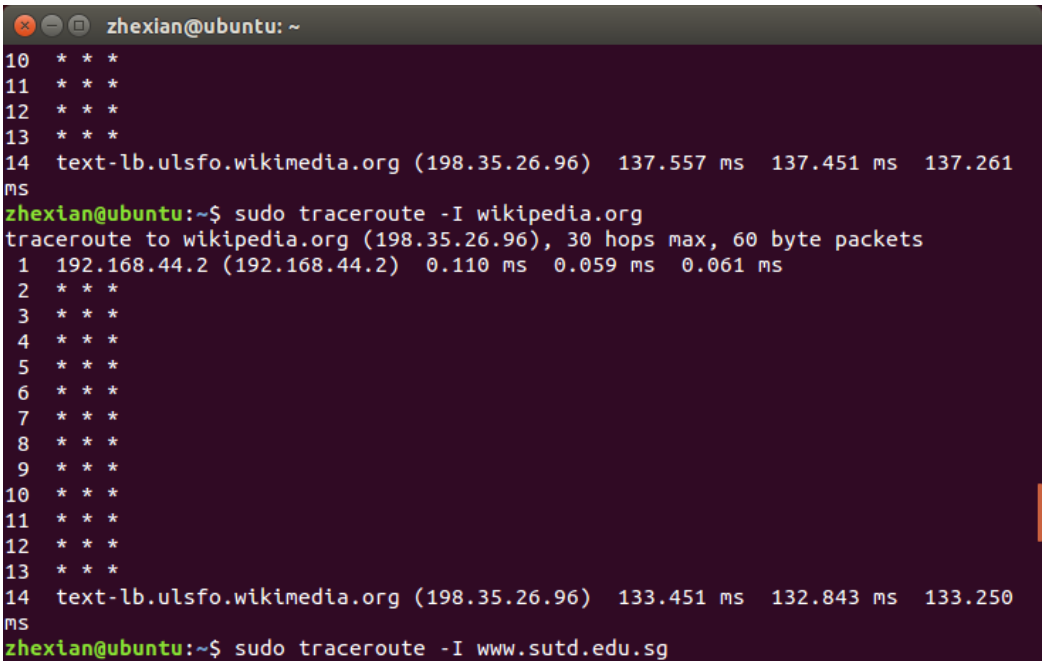


   This is the IP packet, the MAC header is 192.168.1.102 (the difference is due to the fact that this is a difference capture from the previous ARP packet), and the protocol ID is 0x0280 (type TCP).



2. [Chapter 4 Hands-on 6] Use visualroute or traceroute to find out the infrastructure of your domain and the routes to foreign countries. (Hint: traceroute will give you a list of routers; try to identify different types of routers by their subnet addresses and round trip delays.)

I did a traceroute to Wikipedia.org:



It seems that the first router (192.168.44.2) is a local router due to its fast response time (average 0.076ms); the last router (in hop 14) is the destination the route to foreign country. It takes a significantly longer time to respond (average 133.181ms) and it is within the same subnet as the destination.

3. [Chapter 4 Hands-on 10] Trace ip_route_input() and ip_route_output_key() in the Linux source codes. Describe how IP packets are forwarded to the upper layer and the next hop, respectively. (Hint: Both functions can be found in net/ipv4/route.c.)

ip_route_input():

```
/* called in rcu_read_lock() section */
static int ip_route_input_mc(struct sk_buff *skb, __be32 daddr, __be32 saddr,
                             u8 tos, struct net_device *dev, int our)
{
        struct rtable *rth;
        struct in_device *in_dev = __in_dev_get_rcu(dev);
        unsigned int flags = RTCF_MULTICAST;
        u32 itag = 0;
        int err;

        /* Primary sanity checks. */

        if (!in_dev)
                return -EINVAL;

        if (ipv4_is_multicast(saddr) || ipv4_is_lbcast(saddr) ||
            skb->protocol != htons(ETH_P_IP))
                goto e_inval;

        if (ipv4_is_loopback(saddr) && !IN_DEV_ROUTE_LOCALNET(in_dev))
                goto e_inval;

        if (ipv4_is_zeronet(saddr)) {
                if (!ipv4_is_local_multicast(daddr))
                        goto e_inval;
        } else {
                err = fib_validate_source(skb, saddr, 0, tos, 0, dev,
                                          in_dev, &itag);
                if (err < 0)
                        goto e_err;
        }
        if (our)
                flags |= RTCF_LOCAL;

        rth = rt_dst_alloc(dev_net(dev)->loopback_dev, flags, RTN_MULTICAST,
                           IN_DEV_CONF_GET(in_dev, NOPOLICY), false, false);
```

```
        if (!rth)
                goto e_nobufs;

#ifdef CONFIG_IP_ROUTE_CLASSID
        rth->dst.tclassid = itag;
#endif
        rth->dst.output = ip_rt_bug;
        rth->rt_is_input= 1;

#ifdef CONFIG_IP_MROUTE
        if (!ipv4_is_local_multicast(daddr) && IN_DEV_MFORWARD(in_dev))
                rth->dst.input = ip_mr_input;
#endif
        RT_CACHE_STAT_INC(in_slow_mc);

        skb_dst_set(skb, &rth->dst);
        return 0;

e_nobufs:
        return -ENOBUFS;
e_inval:
        return -EINVAL;
e_err:
        return err;
}
```

how IP packets are forwarded to the upper layer and the next hop
The IP packet forwarding destination is set by the function rt_dst_alloc() and skb_dst_set(skb, &rth->dst) to the upper layer and next hop respectively

ip_route_output_key():

```
/*
 * Major route resolver routine.
 */

struct rtable *__ip_route_output_key_hash(struct net *net, struct flowi4 *fl4,
                                          int mp_hash)
{
        struct net_device *dev_out = NULL;
        __u8 tos = RT_FL_TOS(fl4);
        unsigned int flags = 0;
        struct fib_result res;
        struct rtable *rth;
        int orig_oif;
        int err = -ENETUNREACH;

        res.tclassid    = 0;
        res.fi          = NULL;
        res.table       = NULL;

        orig_oif = fl4->flowi4_oif;

        fl4->flowi4_iif = LOOPBACK_IFINDEX;
        fl4->flowi4_tos = tos & IPTOS_RT_MASK;
        fl4->flowi4_scope = ((tos & RTO_ONLINK) ?
                        RT_SCOPE_LINK : RT_SCOPE_UNIVERSE);

        rcu_read_lock();
        if (fl4->saddr) {
                rth = ERR_PTR(-EINVAL);
                if (ipv4_is_multicast(fl4->saddr) ||
                    ipv4_is_lbcast(fl4->saddr) ||
                    ipv4_is_zeronet(fl4->saddr))
                        goto out;
```

```
        /* I removed check for oif == dev_out->oif here.
           It was wrong for two reasons:
           1. ip_dev_find(net, saddr) can return wrong iface, if saddr
              is assigned to multiple interfaces.
           2. Moreover, we are allowed to send packets with saddr
              of another iface. --ANK
         */

        if (fl4->flowi4_oif == 0 &&
            (ipv4_is_multicast(fl4->daddr) ||
             ipv4_is_lbcast(fl4->daddr))) {
                /* It is equivalent to inet_addr_type(saddr) == RTN_LOCAL */
                dev_out = __ip_dev_find(net, fl4->saddr, false);
                if (!dev_out)
                        goto out;

                /* Special hack: user can direct multicasts
                   and limited broadcast via necessary interface
                   without fiddling with IP_MULTICAST_IF or IP_PKTINFO.
                   This hack is not just for fun, it allows
                   vic,vat and friends to work.
                   They bind socket to loopback, set ttl to zero
                   and expect that it will work.
                   From the viewpoint of routing cache they are broken,
                   because we are not allowed to build multicast path
                   with loopback source addr (look, routing cache
                   cannot know, that ttl is zero, so that packet
                   will not leave this host and route is valid).
                   Luckily, this hack is good workaround.
                 */

                fl4->flowi4_oif = dev_out->ifindex;
                goto make_route;
        }

        if (!(fl4->flowi4_flags & FLOWI_FLAG_ANYSRC)) {
                /* It is equivalent to inet_addr_type(saddr) == RTN_LOCAL */
                if (!__ip_dev_find(net, fl4->saddr, false))
                        goto out;
        }
}
```

etc.

The ip_route_output_key() function maintains a routing hash table. If checking is passed, “make_route” will be called, and rth = __mkroute_output(&res, fl4, orig_oif, dev_out, flags) will be set.

4. [Chapter 4 Written 1] What would be the problems when two hosts use the same IP address and ignore each other’s existence?

   The IP address is supposed to be unique for each host, so that the host may be globally identifiable and locatable. When two hosts use the same IP address and ignore each other’s existence, the router will have problem identifying the network interface for sending and receiving IP packets, and the IP packets may not be sent to the correct host.

5. [Chapter 4 Written 7] Consider an IP packet traversing a router:
   Which fields in the IP header must be changed by a router when an IP packet traverses the router?

   - Time-to-Live (TTL), because each router will decrease TTL by one before forwarding it to the next hop router.
   - Header Checksum, because packet header (at least the TTL field) is changed, and the checksum is calculated against the entire packet header, the header checksum must be changed.

   Which fields in the IP header may be changed by a router?

   - Identifier: if packet fragmentation is performed, the identifier field will be used to indicate fragments of the same packet
   - Flags: if packet fragmentation is performed, the "more fragment" bit in the flags field will be set to indicate whether this fragment is the last fragment
   - Fragmentation offset: if packet fragmentation is performed, the offset will be recorded in the fragmentation offset field of the IP header to determine the position of each fragment in the original packet

   Design an efficient algorithm for recalculating the checksum field. (Hint: think about how these fields are changed.)

   The checksum field is computed and filled by treating the whole IP header as a sequence of 16-bit words, summing these words using 1's complement arithmetic, and then complementing the result.

   We may recalculate the checksum after the field change by first temporarily store the changes in field length (how much bit increase or decrease compare to original field value), sum all the changes across all fields, and then apply the changes directly to the original checksum.

6. [Chapter 4 Written 8] Consider a company assigned an IP prefix of 163.168.80.0/22. This company owns three branches; these have 440, 70, and 25 computers, respectively. A router with two WAN interfaces is allocated at each branch to provide internetworking such that three routers are fully connected. If you are asked to plan subnet addresses for these three branches as well as addresses for router interfaces, what would you do? (Hint: a subnet is also required for each link between two routers.)

   This 22-bit IP prefix allows for 10-bit host ID.

   The 25 computers may be coded by 4-bit of address (which accommodates 30 units of information), so it can be assigned IP subnet 163.168.80.0/28

   The 70 computers may be coded by 6-bit of address (which accommodates 126 units of information), so it can be assigned IP subnet 163.168.80.0/26

   The 440 computers may be coded by 8-bit of address (which accommodates 510 units of information), so it can be assigned IP subnet 163.168.80.0/24

   All three routers will be assigned IP subnet 163.168.80.0/24 as well.

7. [Chapter 4 Written 18] Compare the differences between ICMPv4 and ICMPv6. Do we still need DHCP, ARP, and IGMP in IPv6?

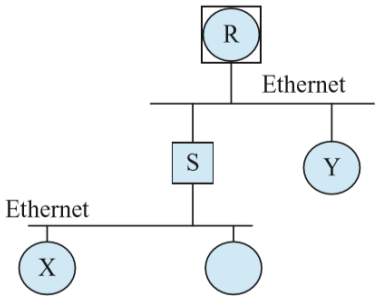
   The values of the type field in ICMPv4 is the same for both error and informational messages, while in ICMPv6, types 0 through 127 are all errors, and types 128 through 255 are all informational.

   In ICMPv6, several new message types appeared, such as communication with destination administratively prohibited, packet too big, and router solicitation.

   Also, ICMPv6 is used for several purposes beyond simple error reporting and signalling in ICMPv4. It is used for:

   - Neighbor Discovery (ND), which plays the same role as ARP does for IPv4.
   - Router Discovery function used for configuring hosts and multicast address management, so no DHCP and IGMP is needed for ICMPv6 anymore.

8. [Chapter 4 Written 29] Consider the following LAN with one Ether switch S, one intra-domain router R, and two hosts X and Y. Assume switch S has been just powered on.



   a. Describe the routing and address resolution steps performed at X, Y, and S when X sends an IP packet to Y.

      Since X and Y are not in the same IP subnet, the packet is sent to router R for intra-domain routing.
      The Address Resolution Protocol (ARP) is used to determine the MAC address of Y given its IP address. Host X broadcast an ARP request message to all hosts within the subnet.
      When host Y recognized that the destination IP address matches its own IP address, it send the ARP reply using unicast to host X.
      After the successful round of ARP, both host X and Y will update their cache table of (IP address, MAC address) pairs.

   b. Describe the routing and address resolution steps performed at X, Y, and S when Y replies an IP packet to X.

      When Y replies an IP packet, the destination IP address is checked against its cached table, and the MAC address correspond to host X's IP address is found. Thus, the IP packet is sent directly to Ether switch S to be forwarded to X.

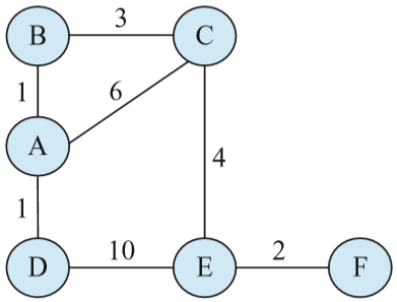
c. Describe the routing and address resolution steps performed at X, S, and R when X sends an IP packet to a host that is outside the domain. (Hint: Do not forget to explain how X knows of the router R.)

Normally, there is an entry recording a default router with the destination address '0.0.0.0/0'. If the destination of a packet does not match any entries of the routing table, it will be forwarded to the default router.

When the packet arrives at a the default local router (intra-domain router R) in host X's autonomous system (AS) with destination as an IP which doesn't belong to the local AS, the router R will forward the packet to the border router.

The border router will have physical links connected to border routers of other ASs. It will deliver the packet to the correct destination AS border router, and the destination border router will send the packet to the destination host using local routing algorithm.

9. [Chapter 4 Written 30] Consider the following network topology. Show how node A constructs its routing table using link state routing and distance vector routing, respectively.



Link state routing (Dijkstra's algorithm):

Iteration	Least-Cost Set	C(B), p(B)	C(C), p(C)	C(D), p(D)	C(E), p(E)	C(F), p(F)
1	A	1, A	6, A	1, A	∞	∞
2	AB	0	4, B	1, A	∞	∞
3	ABD	0	4, B	0	11, D	∞
4	ABDC	0	0	0	8, B	∞
5	ABDCE	0	0	0	0	10, E
6	ABDCEF	0	0	0	0	0

Routing table of node A:

Destination	Cost	Next Hop
B	1	B
C	4	B
D	1	D
E	8	B
F	10	B

Distance vector routing (Bellman-Ford algorithm):

Step 0:

Destination	Cost	Next Hop
B	1	B
C	6	C
D	1	D

Step 1:

Destination	Cost	Next Hop
B	1	B
C	4	B
D	1	D
E	10	C

Step 2:

Destination	Cost	Next Hop
B	1	B
C	4	B
D	1	D
E	8	B

Step 3:

Destination	Cost	Next Hop
B	1	B
C	4	B
D	1	D
E	8	B
F	10	B

10. [Chapter 4 Written 34] Distance vector routing algorithm is adopted in intra-domain routing (e.g., RIP) as well as inter-domain routing (e.g., BGP), but it is implemented with different concerns and additional features. Compare the difference between intra-domain routing and inter-domain routing when both use the distance vector algorithm.

    In intra-domain routing, optimization in resource utilization is valued, while in inter-domain routing, reachability, stability and scalability are more important factors.

    In inter-domain routing, the requirement for finding a loop-free path to reach a destination is non-trivial and more strictly followed than in intra-domain routing.

    For distance vector algorithm RIP in intra-domain routing, its messages are sent over UDP using port 52, while for BGP in inter-domain routing, the messages are sent over TCP on port 179 to ensure a stable TCP connection is established.

    In RIP, the routers in path only exchange information with neighbors, whereas in BGP, in order to prevent looping, the complete path information is also advertised when exchanging a route entry.

11. [Chapter 4 Written 50] Show the multicast tree built by DVMRP in the following network topology: