KOÇ UNIVERSITY
COMP 416 - Computer Networks, Fall 2020

Project 3

Berkay Barlas, 0054512

January 11, 2021

Contents

1 Part 1 - ICMP Analysis

```
Last login: Sun Jan 10 17:48:29 on ttys000
➜  ~ ping www.berkeley.edu
PING www-production-1113102805.us-west-2.elb.amazonaws.com (34.212.212.2): 56 data bytes
64 bytes from 34.212.212.2: icmp_seq=0 ttl=221 time=221.316 ms
64 bytes from 34.212.212.2: icmp_seq=1 ttl=221 time=221.635 ms
64 bytes from 34.212.212.2: icmp_seq=2 ttl=221 time=224.213 ms
64 bytes from 34.212.212.2: icmp_seq=3 ttl=221 time=223.373 ms
64 bytes from 34.212.212.2: icmp_seq=4 ttl=221 time=222.074 ms
^C
--- www-production-1113102805.us-west-2.elb.amazonaws.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 221.316/222.522/224.213/1.098 ms
➜  ~
```

Figure 1: Execution of Ping command on www.berkeley.edu url

1.1 1.a: Ping Analysis

1. What are the three layers in the ICMP packet?

   ICMP is a supporting protocol in the Internet protocol that operates in network layer. Therefore, there are 3 layers in a ICMP packet; Network Layer, Data Link Layer, Physical Layer.

```
▶ Ethernet II, Src: Apple_55:31:be (f0:18:98:55:31:be), Dst: ASUSTekC_75:8d:08 (d4:5d:64:75:8d:08)
▶ Internet Protocol Version 4, Src: 192.168.1.8, Dst: 54.187.154.104
▶ Internet Control Message Protocol
```

Figure 2: Wireshark layer analysis of ICMP packet

2. What is TTL and its significance? Which layer does it reside in and is it constant (format and no. of bits wise) across IPV4 and IPV6 ping commands?

   Time to live (TTL) is a mechanism that limits the lifespan (hop limit) of data in a computer network. It resides in the Network Layer. Its a 8-bit field with the maximum value of 255 for both IPV4 and IPV6. However, the location of TTL in IPV4 and IPV6 header are different.

```
 16 2.257119   192.168.1.8   54.187.154.104   ICMP   98
 30 3.262245   192.168.1.8   54.187.154.104   ICMP   98
 40 4.266185   192.168.1.8   54.187.154.104   ICMP   98
 44 5.266499   192.168.1.8   54.187.154.104   ICMP   98
 50 6.269790   192.168.1.8   54.187.154.104   ICMP   98

▶ Frame 16: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en0, id 0
▶ Ethernet II, Src: Apple_55:31:be (f0:18:98:55:31:be), Dst: ASUSTekC_75:8d:08 (d4:5d:64:75:8d:08)
▼ Internet Protocol Version 4, Src: 192.168.1.8, Dst: 54.187.154.104
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x12bc (4796)
  ▶ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 64
    Protocol: ICMP (1)
    Header Checksum: 0xd519 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.8
    Destination Address: 54.187.154.104
▼ Internet Control Message Protocol
```

Figure 3: Wireshark TTL analysis of ICMP packet

3. Why is it that an ICMP packet does not have source and destination port numbers?

   There are no UDP or TCP source and destination port numbers associated with ICMP packets because port numbers are related with transport layer. Since, ICMP is a network layer protocol it doesn't have transport layer data.

4. What is the length of the datafield of the ICMP part Type-8? Elaborate on the structure of the datafield citing any correspondingly common and changing parts across various messages? If part of the data field, what do you think is the reason for that?

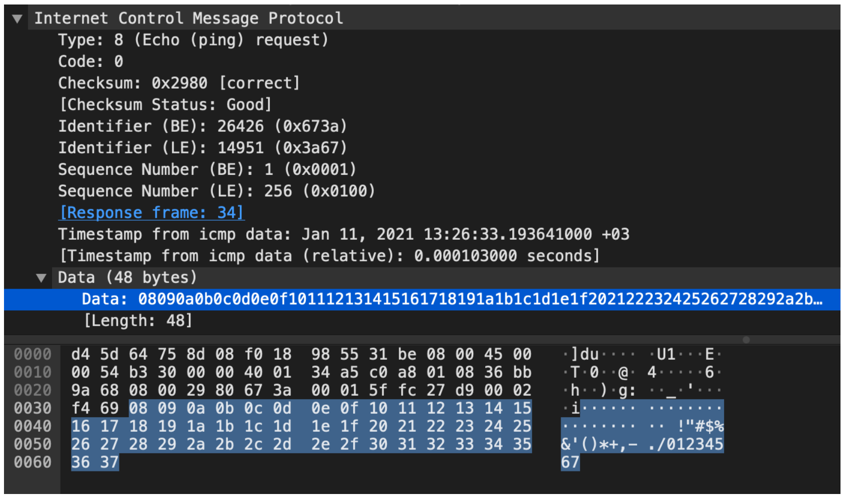


Figure 4: ICMP packet datafield analysis

   The length of the datafield of the ICMP part Type-8 is 48 bytes. Type 8 is used for "Echo Request" and it has same value for all requests.

5. Find the minimum TTL below which the ping messages do not reach your particular URL destination.

   According to my trials the minimum TTL to reach destination url "www.berkeley.edu" is 39.

```
  -m ttl     Set the IP Time To Live for outgoing packets.
           If not specified, the kernel uses the value of the net.inet.ip.ttl MIB variable.
```

Figure 5: Set TTL flag from "man ping"

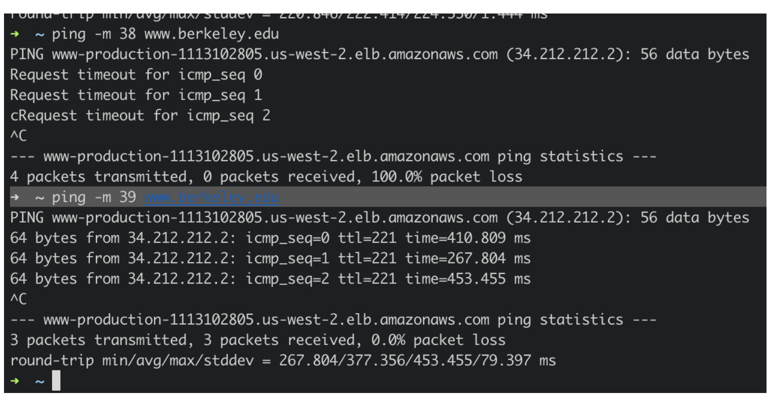


Figure 6: Minimum TTL for ping messages

6. How do the Identifier and Sequence Number compare for successive echo request packets?

   For each successive echo request packets, identifier numbers doesn't change,however, Sequence Numbers increase by 1 for BE and increase by 256 for LE values.

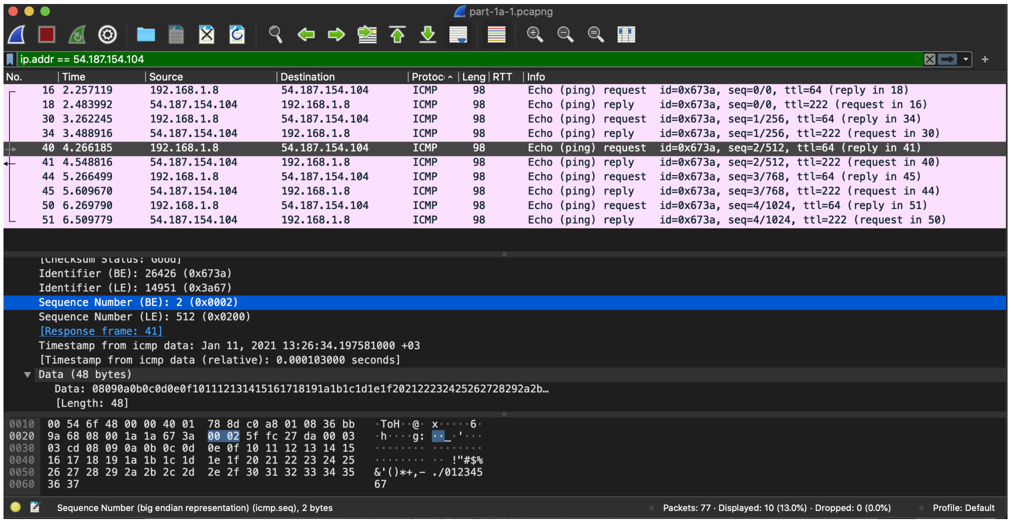


Figure 7: Echo request packets in Wireshark

1.2 1.b : Traceroute Analysis

7. How long is the ICMP header of a TTL Exceeded packet? Select different parts of the header in Wireshark to see how they correspond to the bytes in the packet

   The ICMP header of a TTL Exceeded packet is 8 bytes; 1 byte Type, 1 byte Code, 2 byte Checksum and 4 byte for unused.

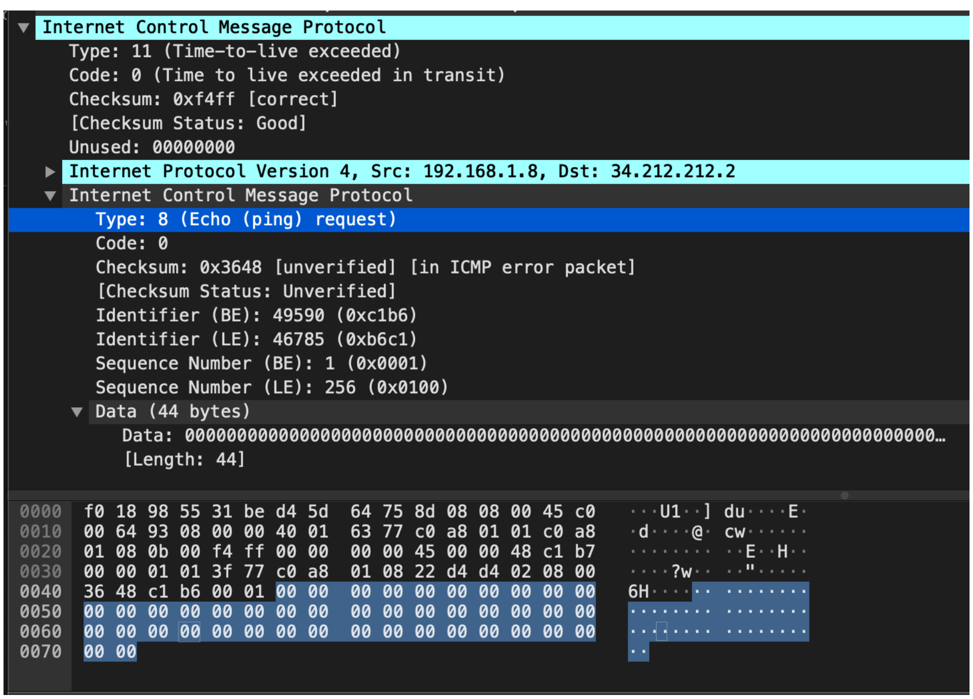
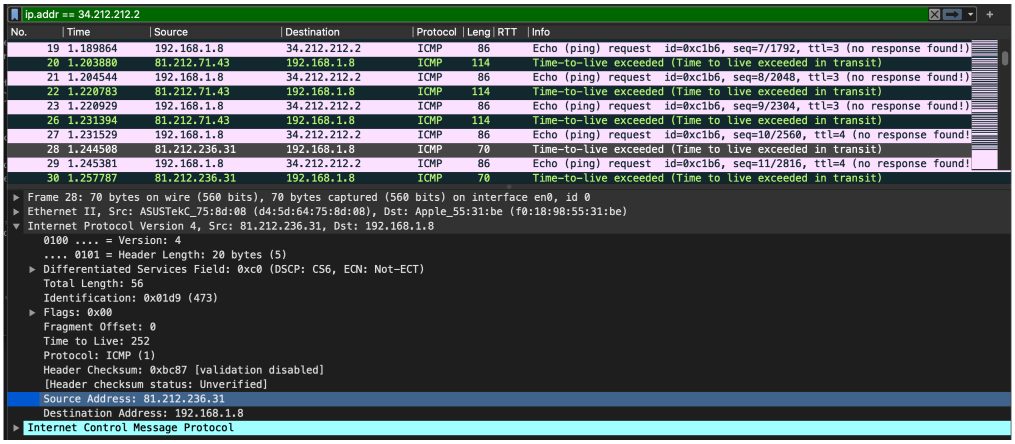


Figure 8: TTL Exceeded packet Wireshark analysis

8. How does your computer (the source) learn the IP address of a router along the path from a TTL exceeded packet?

   Traceroute command sends Echo(ping) Requests starting from TTL=1 and increasing until reaching the final destination. When each packets dies along the path recieved "Time-to-live-exceeded" messages reveals IP addresses in path with Source Address field.



ip.addr == 34.212.212.2

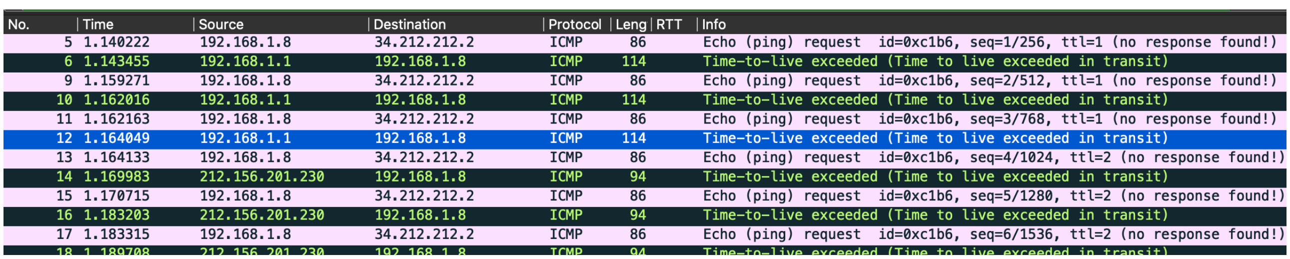
No.	Time	Source	Destination	Protocol	Leng	RTT	Info
19	1.189864	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=7/1792, ttl=3 (no response found!)
20	1.203880	81.212.71.43	192.168.1.8	ICMP	114		Time-to-live exceeded (Time to live exceeded in transit)
21	1.204544	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=8/2048, ttl=3 (no response found!)
22	1.220783	81.212.71.43	192.168.1.8	ICMP	114		Time-to-live exceeded (Time to live exceeded in transit)
23	1.220929	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=9/2304, ttl=3 (no response found!)
26	1.231394	81.212.71.43	192.168.1.8	ICMP	114		Time-to-live exceeded (Time to live exceeded in transit)
27	1.231529	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=10/2560, ttl=4 (no response found!
28	1.244508	81.212.236.31	192.168.1.8	ICMP	70		Time-to-live exceeded (Time to live exceeded in transit)
29	1.245381	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=11/2816, ttl=4 (no response found!
30	1.257787	81.212.236.31	192.168.1.8	ICMP	70		Time-to-live exceeded (Time to live exceeded in transit)

```
▶ Frame 28: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface en0, id 0
▶ Ethernet II, Src: ASUSTekC_75:8d:08 (d4:5d:64:75:8d:08), Dst: Apple_55:31:be (f0:18:98:55:31:be)
▼ Internet Protocol Version 4, Src: 81.212.236.31, Dst: 192.168.1.8
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x01d9 (473)
  ▶ Flags: 0x00
    Fragment Offset: 0
    Time to Live: 252
    Protocol: ICMP (1)
    Header Checksum: 0xbc87 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 81.212.236.31
    Destination Address: 192.168.1.8
▶ Internet Control Message Protocol
```

Figure 9: Traceroute messages in Wireshark

9. How many times is each router along the path probed by traceroute?

   Every route in the path probed 3 times with traceroute command.



No.	Time	Source	Destination	Protocol	Leng	RTT	Info
5	1.140222	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=1/256, ttl=1 (no response found!)
6	1.143455	192.168.1.1	192.168.1.8	ICMP	114		Time-to-live exceeded (Time to live exceeded in transit)
9	1.159271	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=2/512, ttl=1 (no response found!)
10	1.162016	192.168.1.1	192.168.1.8	ICMP	114		Time-to-live exceeded (Time to live exceeded in transit)
11	1.162163	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=3/768, ttl=1 (no response found!)
12	1.164049	192.168.1.1	192.168.1.8	ICMP	114		Time-to-live exceeded (Time to live exceeded in transit)
13	1.164133	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=4/1024, ttl=2 (no response found!)
14	1.169983	212.156.201.230	192.168.1.8	ICMP	94		Time-to-live exceeded (Time to live exceeded in transit)
15	1.170715	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=5/1280, ttl=2 (no response found!)
16	1.183203	212.156.201.230	192.168.1.8	ICMP	94		Time-to-live exceeded (Time to live exceeded in transit)
17	1.183315	192.168.1.8	34.212.212.2	ICMP	86		Echo (ping) request id=0xc1b6, seq=6/1536, ttl=2 (no response found!)
18	1.189708	212.156.201.230	192.168.1.8	ICMP	94		Time-to-live exceeded (Time to live exceeded in transit)

Figure 10: 2 different routes with 6 probes

10. Within the tracert measurements, is there a link whose delay is significantly longer than others? The echo request packets sent by traceroute are probing successively more distant routers along the path. You can look at these packets and see how they differ when they elicit responses from different routers.

    Between 8th and 9th hops there is a huge increase in delay, from 60ms to 200ms. Also from 22th to 37th hop, routers doesn't respond at all to traceroute messages. This might caused by Amazon Web Services internal network behavior. Some of these routes probably don't have direct access to internet.

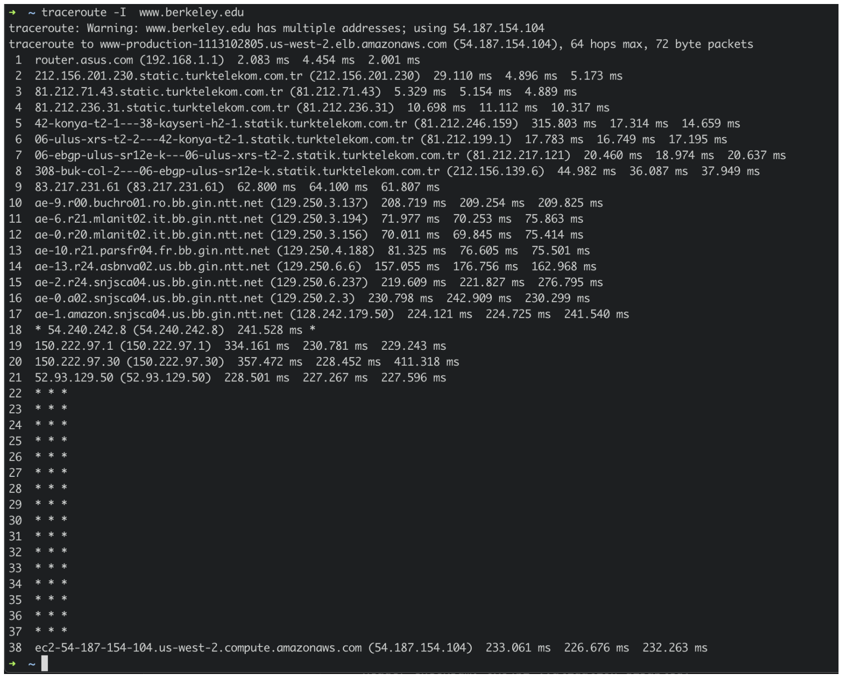


Figure 11: Traceroute command execution on www.berkeley.edu

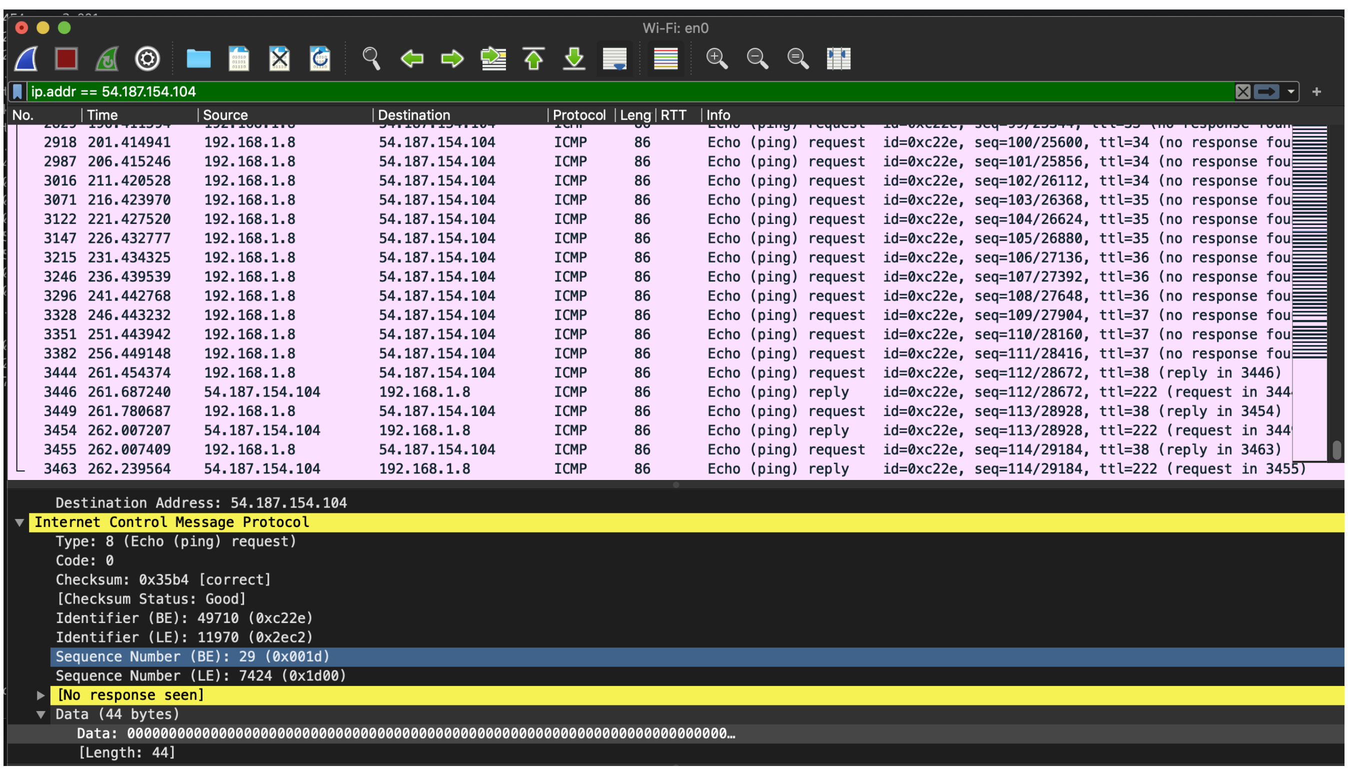


Figure 12: Stop-and-Wait ARQ Protocol Analysis start

2 Part 2 - Routing Implementation

2.1 Naive Flooding Algorithm

11. See that this algorithm succeeds in topology 1. What does the total communication cost represent, and why is it different from the path cost?

    The path cost shows the total cost if links from origin to destination in resulted path. Which 1→3→4 , 2+7=9 in topology 1.

    The total communication cost shows the cost of routing of packets to determine the path. 2+5+7+8=22 in topology 1.

12. See that this algorithm fails in topology 2, and the simulator notes that the protocol does not converge. Why is this the case?

    Naive Flooding Algorithm doesn't converge in topology 2 because packets routes between nodes 1,2,3 in loop and never goes to node 4. This causes multiple hops to same node and missing some node in certain topologies such as topology 2.

2.2 Flooding Algorithm

In addition to implementation of Naive Flooding Algorithm, I used a boolean value (visited) to keep state and only return the neighbors the first time and return empty list otherwise.

```
// Visited flag
boolean visited = false;

@Override
public List<NeighborInfo> selectNeighbors(String origin, String destination, String
    previousHop,
                                        List<NeighborInfo> neighbors) {
    // Check visited flag
    if (visited) {
        // Return empty list after first call
        return new ArrayList();
    }
    // Set visited flag true
    visited = true;

    // Find the list of neighbors, excluding the previous hop.
    List<NeighborInfo> chosen = neighbors.stream()
            // Make sure that we do not route back to the previous hop.
            .filter(n -> !n.address.equals(previousHop))
            .collect(Collectors.toList());

    // Return the chosen nodes.
    return chosen;
}
```

13. Consider the path taken by the packet in topology 2 with this algorithm. Is this what you have expected? Why or why not?

    The resulted path is 1→3→4 with path cost 9 and communication cost 2+1+5+5+8+7=28. This is the expected result because it's the shortest possible path.

2.3 Naive Minimum Cost Algorithm

```
@Override
public List<NeighborInfo> selectNeighbors(String origin, String destination, String
    previousHop, List<NeighborInfo> neighbors) {

   // Find the list of neighbors, excluding the previous hop.
   List<NeighborInfo> chosen = Arrays.asList(neighbors.stream()
         // Make sure that we do not route back to the previous hop.
         .filter(n -> !n.address.equals(previousHop))
         .min((i,j)-> Integer.compare(i.cost, j.cost))
         .orElseThrow(NoSuchElementException::new));
   // Return the chosen nodes.
   return chosen;
}
```

14. If you implemented the algorithm as specified, it should succeed in topology 1 but fail in topology 2. Why does it fail in topology 2?

    I will stay in infinite loop between nodes 1,2, and 3 due to high link costs of 2→4 and 3→4 in topology 2. In topology 1 there isn't any alternative path for nodes 2 and 3 therefore they will choose the path to node 4.

2.4 Minimum Cost Algorithm

```
// IMPORTANT: Use this random number generator.
Random rand = new Random(6391238);

// IMPORTANT: You can maintain a state, e.g., a set of neighbors.
HashSet<String> visited = new HashSet<>();

@Override
public List<NeighborInfo> selectNeighbors(String origin, String destination, String
    previousHop,
                                 List<NeighborInfo> neighbors) {

   if (visited.size() >= neighbors.size()) {
      int selectedIndex = rand.nextInt(neighbors.size());
      // Return the randomly chosen node.
      return Arrays.asList(neighbors.get(selectedIndex));
   }

   visited.add(previousHop);
   // Find the list of neighbors, excluding the previous hop.
   NeighborInfo chosen = neighbors.stream()
         // Make sure that we do not route back to the previous hop.
         .filter(n -> !visited.contains(n.address))
         .min(Comparator.comparingInt(i -> i.cost))
         .orElseThrow(NoSuchElementException::new);

   visited.add(chosen.address);

   // Return the chosen nodes.
   return Arrays.asList(chosen);
}
```

15. List the nodes in the exclusion set of each node at the end of the simulation of topology 2.

    Resulting Path: 1→3→2→1→2→4

    Exclusion sets for each node:

    Node 1: [Node 2, Node 3]

    Node 2: [Node 1, Node 3, Node 4]

    Node 3: [Node 1, Node 2]

    Node 4: [] (Empty)