NSSA-220
Mini Project 2

Packet Capture Analysis Tool

Mini Project 2 Preliminaries

- Done in teams of 3 students
  - You are required to submit peer reviews as part of this project to encourage reasonable contributions from each team member
- **Fair warning**: do NOT wait to start this project. **It will not go well if you do!**

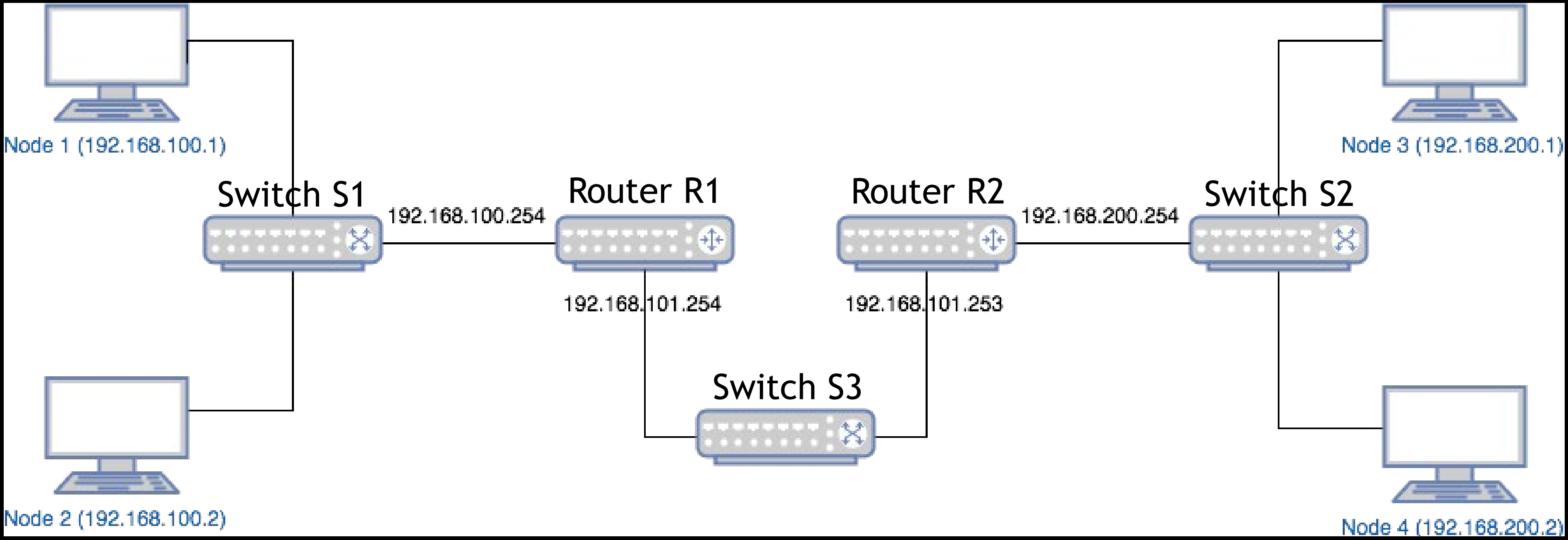
Packet Capture Analysis (PCA)

- Network engineers and security analysts are often interested in analyzing network packet captures to then analyze network activity
- Network activity analysis may result in outcomes such as introducing additional network components for load balancing, new routes/paths through the network, or spinning up further analysis for confirming networking breaches

PCA continued

- Typically, individuals and organizations will collect packet captures, but do nothing substantial with them
- The purpose of this project is to create a Packet Capture Analysis (PCA) tool that computes metrics from these packet captures that could be used in decision making

Network Topology Diagram

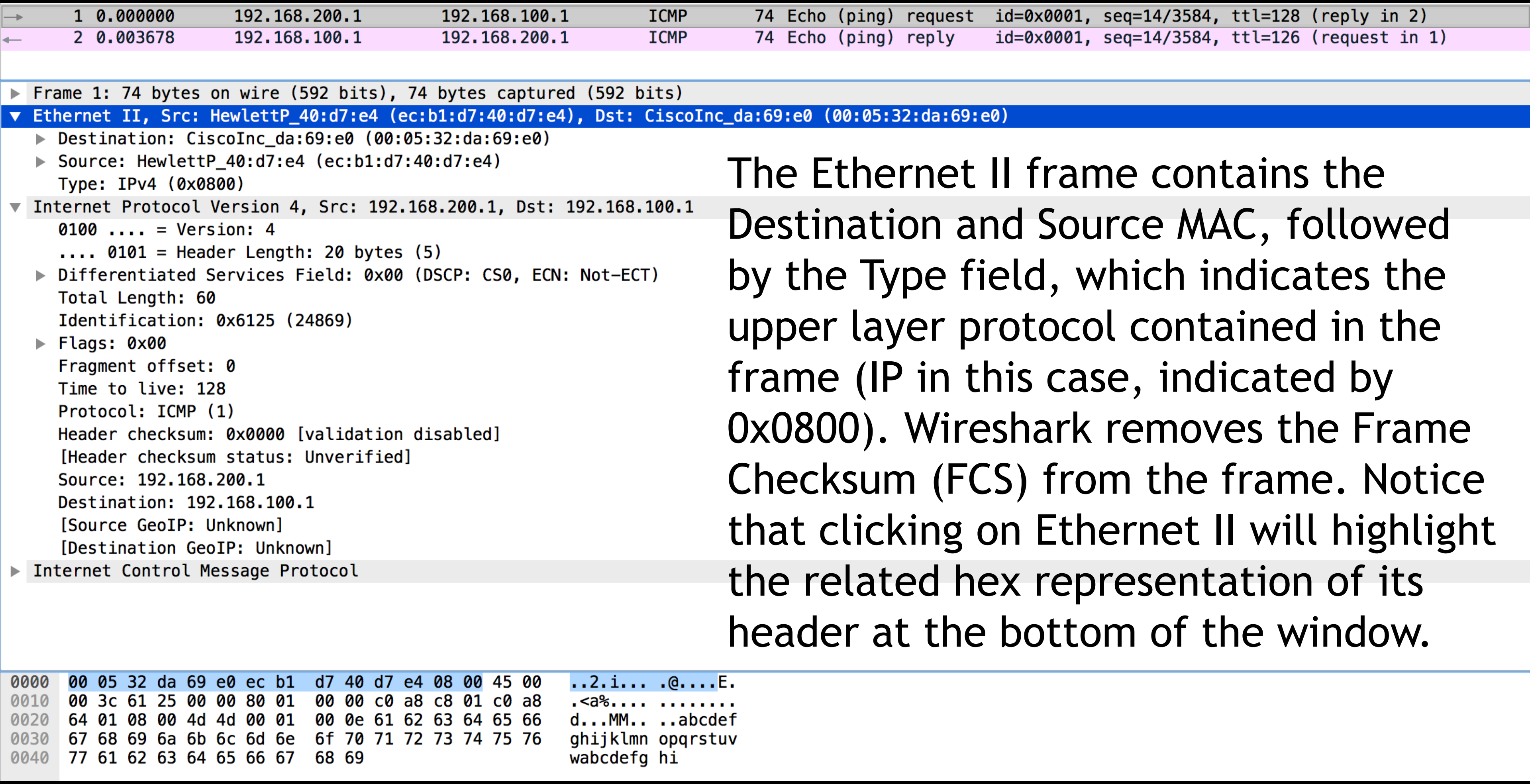


Packets were captured at each of the 4 nodes in the topology. ICMP requests were manually sent between nodes using a simple schedule.

Internet Control Message Protocol (ICMP)

- ICMP is used by the Internet Protocol to send error messages and operational/diagnostic information to devices in a network
- We'll focus on the messages generated by the **ping** program
  - Echo Request (ICMP Type 8 message)
  - Echo Reply (ICMP Type 0 message)
  - Used in tandem to verify connectivity between network devices

Echo Request Example

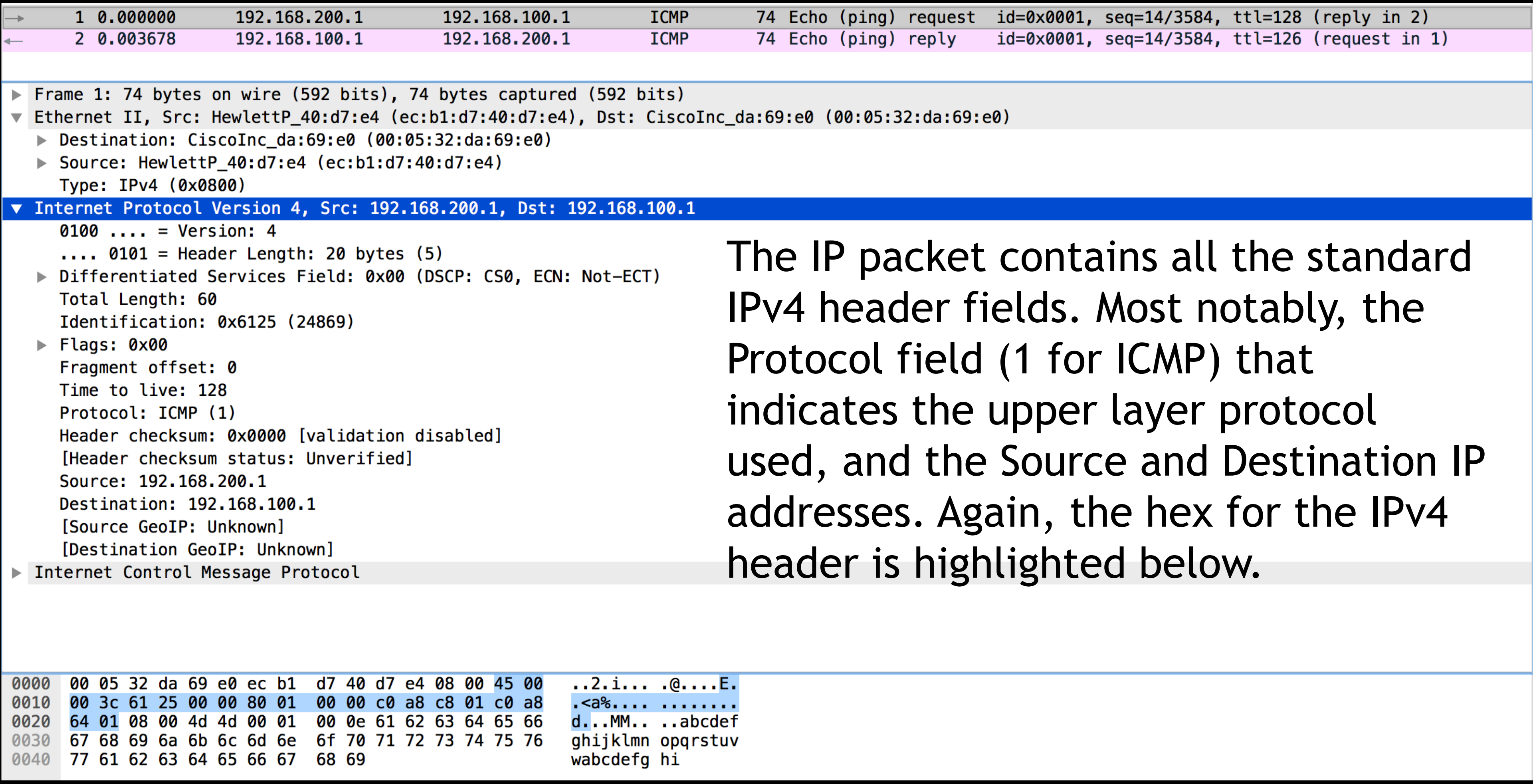


```
→  1 0.000000   192.168.200.1   192.168.100.1   ICMP   74 Echo (ping) request  id=0x0001, seq=14/3584, ttl=128 (reply in 2)
←  2 0.003678   192.168.100.1   192.168.200.1   ICMP   74 Echo (ping) reply    id=0x0001, seq=14/3584, ttl=126 (request in 1)

▶ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▼ Ethernet II, Src: HewlettP_40:d7:e4 (ec:b1:d7:40:d7:e4), Dst: CiscoInc_da:69:e0 (00:05:32:da:69:e0)
  ▶ Destination: CiscoInc_da:69:e0 (00:05:32:da:69:e0)
  ▶ Source: HewlettP_40:d7:e4 (ec:b1:d7:40:d7:e4)
    Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 192.168.200.1, Dst: 192.168.100.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x6125 (24869)
  ▶ Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.200.1
    Destination: 192.168.100.1
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▶ Internet Control Message Protocol

0000  00 05 32 da 69 e0 ec b1  d7 40 d7 e4 08 00 45 00   ..2.i... .@....E.
0010  00 3c 61 25 00 00 80 01  00 00 c0 a8 c8 01 c0 a8   .<a%.... ........
0020  64 01 08 00 4d 4d 00 01  00 0e 61 62 63 64 65 66   d...MM.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

The Ethernet II frame contains the Destination and Source MAC, followed by the Type field, which indicates the upper layer protocol contained in the frame (IP in this case, indicated by 0x0800). Wireshark removes the Frame Checksum (FCS) from the frame. Notice that clicking on Ethernet II will highlight the related hex representation of its header at the bottom of the window.

ICMP operates on top of the Internet Protocol (at Layer 3) and is therefore contained within an Ethernet II frame/IP packet

Echo Request Example (cont.)



```
→   1 0.000000     192.168.200.1      192.168.100.1      ICMP      74  Echo (ping) request  id=0x0001, seq=14/3584, ttl=128 (reply in 2)
←   2 0.003678     192.168.100.1      192.168.200.1      ICMP      74  Echo (ping) reply    id=0x0001, seq=14/3584, ttl=126 (request in 1)
```

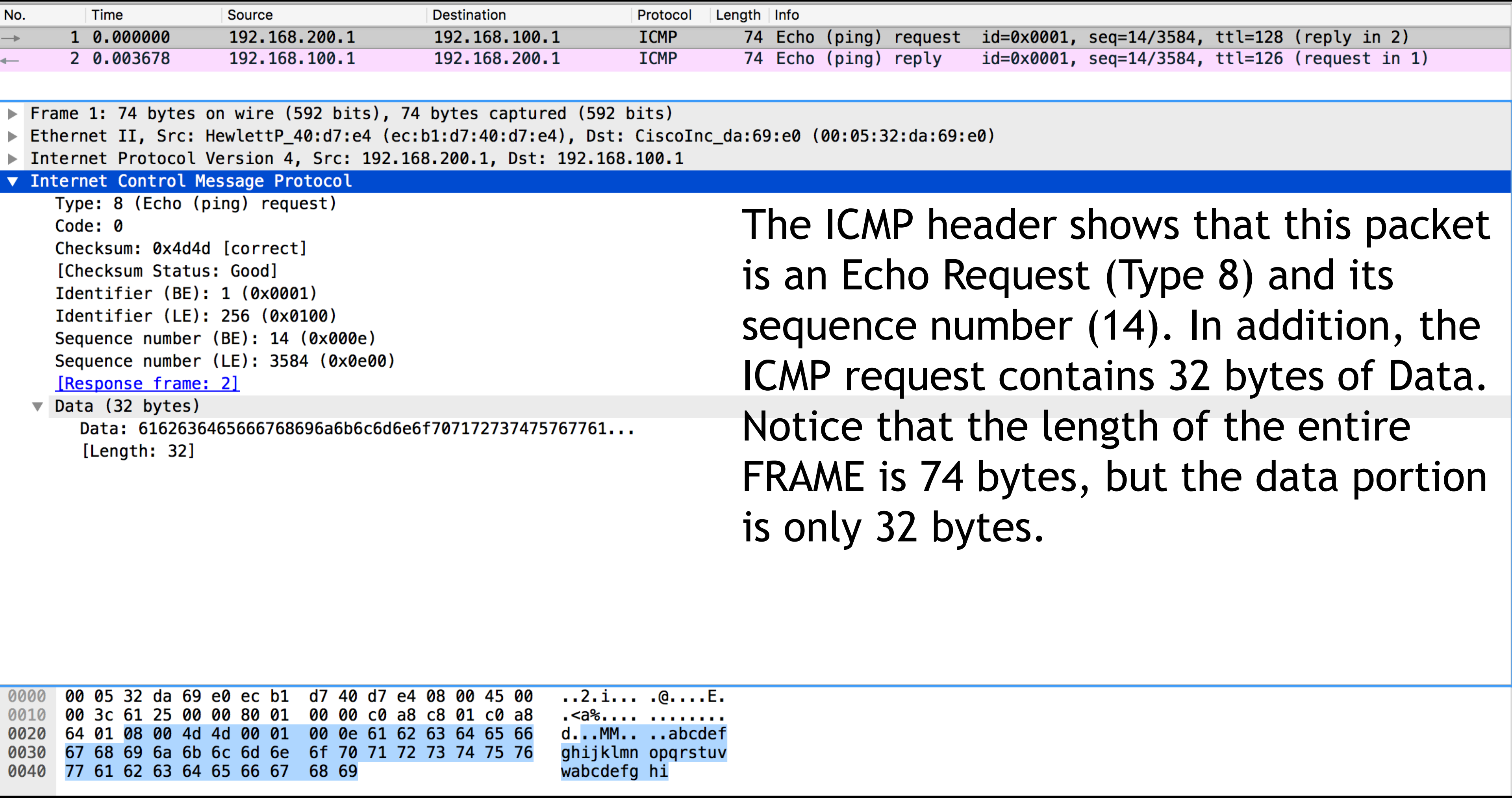
```
▶ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▼ Ethernet II, Src: HewlettP_40:d7:e4 (ec:b1:d7:40:d7:e4), Dst: CiscoInc_da:69:e0 (00:05:32:da:69:e0)
  ▶ Destination: CiscoInc_da:69:e0 (00:05:32:da:69:e0)
  ▶ Source: HewlettP_40:d7:e4 (ec:b1:d7:40:d7:e4)
    Type: IPv4 (0x0800)
▼ Internet Protocol Version 4, Src: 192.168.200.1, Dst: 192.168.100.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x6125 (24869)
  ▶ Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.200.1
    Destination: 192.168.100.1
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▶ Internet Control Message Protocol
```

The IP packet contains all the standard IPv4 header fields. Most notably, the Protocol field (1 for ICMP) that indicates the upper layer protocol used, and the Source and Destination IP addresses. Again, the hex for the IPv4 header is highlighted below.

```
0000  00 05 32 da 69 e0 ec b1  d7 40 d7 e4 08 00 45 00   ..2.i... .@....E.
0010  00 3c 61 25 00 00 80 01  00 00 c0 a8 c8 01 c0 a8   .<a%.... ........
0020  64 01 08 00 4d 4d 00 01  00 0e 61 62 63 64 65 66   d...MM.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

ICMP operates on top of the Internet Protocol (at Layer 3) and is therefore contained within an Ethernet II frame/IP packet

Echo Request Example (cont.)



```
No.  Time      Source          Destination     Protocol Length Info
→  1 0.000000  192.168.200.1   192.168.100.1   ICMP       74 Echo (ping) request  id=0x0001, seq=14/3584, ttl=128 (reply in 2)
←  2 0.003678  192.168.100.1   192.168.200.1   ICMP       74 Echo (ping) reply    id=0x0001, seq=14/3584, ttl=126 (request in 1)
```

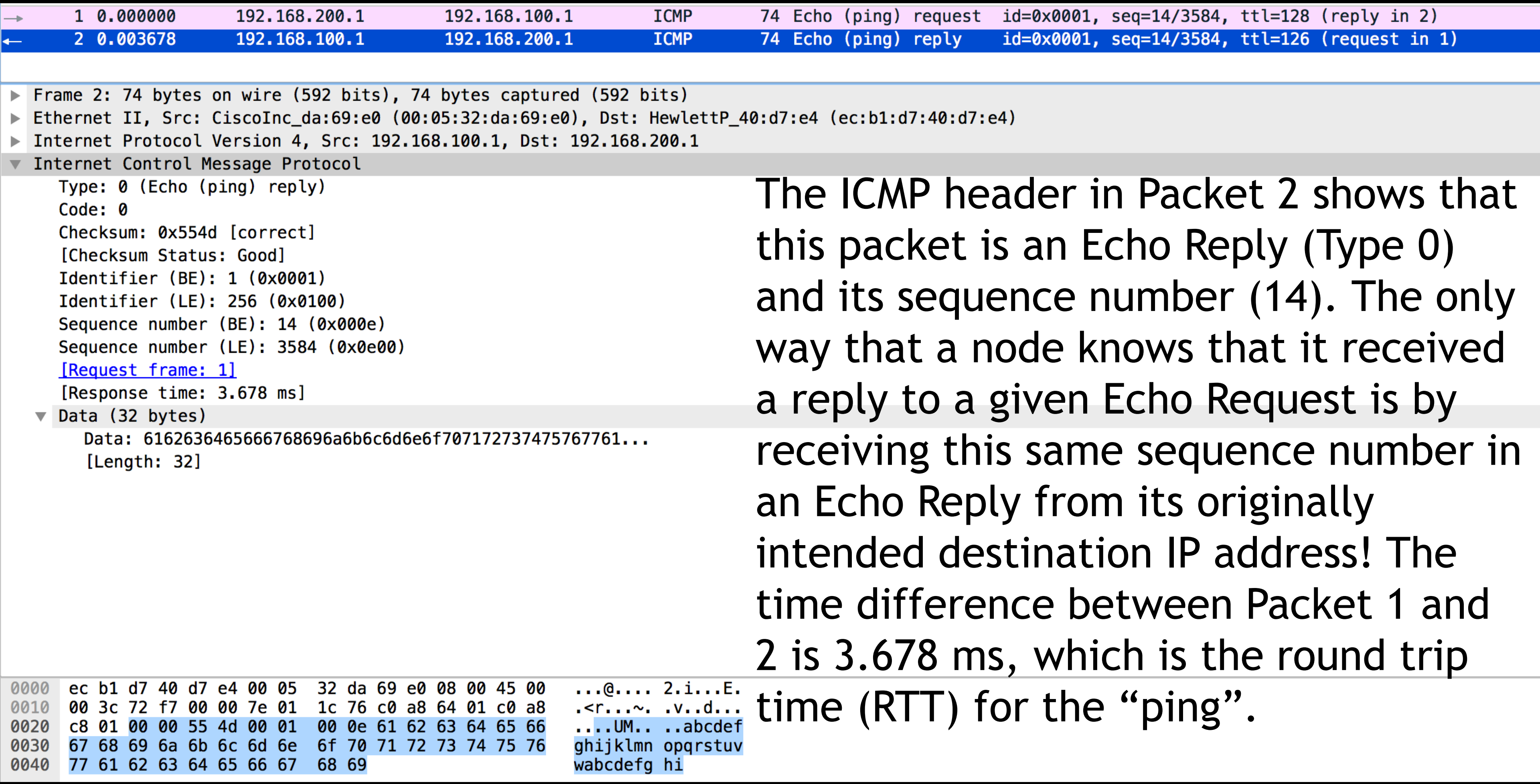
```
▶ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▶ Ethernet II, Src: HewlettP_40:d7:e4 (ec:b1:d7:40:d7:e4), Dst: CiscoInc_da:69:e0 (00:05:32:da:69:e0)
▶ Internet Protocol Version 4, Src: 192.168.200.1, Dst: 192.168.100.1
▼ Internet Control Message Protocol
     Type: 8 (Echo (ping) request)
     Code: 0
     Checksum: 0x4d4d [correct]
     [Checksum Status: Good]
     Identifier (BE): 1 (0x0001)
     Identifier (LE): 256 (0x0100)
     Sequence number (BE): 14 (0x000e)
     Sequence number (LE): 3584 (0x0e00)
     [Response frame: 2]
   ▼ Data (32 bytes)
        Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
        [Length: 32]
```

```
0000  00 05 32 da 69 e0 ec b1  d7 40 d7 e4 08 00 45 00   ..2.i... .@....E.
0010  00 3c 61 25 00 00 80 01  00 00 c0 a8 c8 01 c0 a8   .<a%.... ........
0020  64 01 08 00 4d 4d 00 01  00 0e 61 62 63 64 65 66   d...MM.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

The ICMP header shows that this packet is an Echo Request (Type 8) and its sequence number (14). In addition, the ICMP request contains 32 bytes of Data. Notice that the length of the entire FRAME is 74 bytes, but the data portion is only 32 bytes.

The Echo Request was sent at Time 0.000000. This time indicates the time since the packet capture session was started on the node.

Echo Reply Example



```
→   1 0.000000   192.168.200.1   192.168.100.1   ICMP   74 Echo (ping) request  id=0x0001, seq=14/3584, ttl=128 (reply in 2)
←   2 0.003678   192.168.100.1   192.168.200.1   ICMP   74 Echo (ping) reply    id=0x0001, seq=14/3584, ttl=126 (request in 1)

▶ Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
▶ Ethernet II, Src: CiscoInc_da:69:e0 (00:05:32:da:69:e0), Dst: HewlettP_40:d7:e4 (ec:b1:d7:40:d7:e4)
▶ Internet Protocol Version 4, Src: 192.168.100.1, Dst: 192.168.200.1
▼ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x554d [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 14 (0x000e)
    Sequence number (LE): 3584 (0x0e00)
    [Request frame: 1]
    [Response time: 3.678 ms]
  ▼ Data (32 bytes)
      Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
      [Length: 32]

0000  ec b1 d7 40 d7 e4 00 05  32 da 69 e0 08 00 45 00   ...@.... 2.i...E.
0010  00 3c 72 f7 00 00 7e 01  1c 76 c0 a8 64 01 c0 a8   .<r...~. .v..d...
0020  c8 01 00 00 55 4d 00 01  00 0e 61 62 63 64 65 66   ....UM.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

The ICMP header in Packet 2 shows that this packet is an Echo Reply (Type 0) and its sequence number (14). The only way that a node knows that it received a reply to a given Echo Request is by receiving this same sequence number in an Echo Reply from its originally intended destination IP address! The time difference between Packet 1 and 2 is 3.678 ms, which is the round trip time (RTT) for the “ping”.

The combination of Source/Destination IP and sequence number allows you to associate an Echo Request/Reply pair.

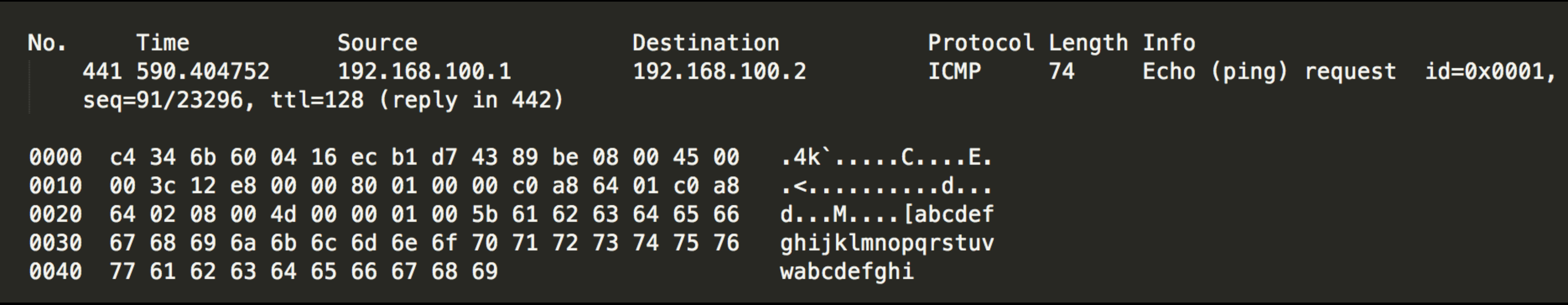
PCA Tool

- The packet capture analysis tool will consist of three main phases
  - **Packet Filtering: keep only the packets we want to analyze**
  - **Packet Parsing: read relevant packet fields into memory for processing**
  - **Compute Metrics: using packet fields to compute metrics**
- Your task is to filter select ICMP packets out of packet captures containing ~8000 packets collected across 4 nodes and compute 13 metrics from them

PCA Phase 1 - Packet Filtering

- You’ll be given one PCAP file per node (see Node*.pcap) and a raw text file derived from the PCAP (see Node*.txt)
- Capture files contain anywhere from 1300-1800 packets
- The packet filtering phase will filter the raw text file so that only ICMP Echo Request and ICMP Echo Reply packets remain and are placed in a new filtered output file (Node*_filtered.txt)

PCA Phase 2 – Packet Parsing



```
No.     Time           Source                Destination           Protocol Length Info
    441 590.404752     192.168.100.1         192.168.100.2         ICMP     74     Echo (ping) request  id=0x0001,
    seq=91/23296, ttl=128 (reply in 442)

0000  c4 34 6b 60 04 16 ec b1 d7 43 89 be 08 00 45 00   .4k`.....C....E.
0010  00 3c 12 e8 00 00 80 01 00 00 c0 a8 64 01 c0 a8   .<..........d...
0020  64 02 08 00 4d 00 00 01 00 5b 61 62 63 64 65 66   d...M....[abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76   ghijklmnopqrstuv
0040  77 61 62 63 64 65 66 67 68 69                     wabcdefghi
```

- Before you can compute metrics, you must parse the filtered raw text files and read packet fields into your tool
- You may choose to parse the summary line text or the hex (**bonus points will be awarded for parsing the hex)**
- The fields you need will be determined by the metrics you need to compute

PCA Phase 3 – Compute Metrics

- All 13 metrics you collect will be on a "***per node***" basis.
- You will be calculating three categories of metrics
  - Data size metrics (8 metrics)
  - Time based metrics (4 metrics)
  - Distance metric (1 metric)

Data Size Metrics

- These metrics indicates how many packets a node sends/receive and the related amount of data/bytes sent/received
- 1. Number of Echo Requests sent
- 2. Number of Echo Requests received
- 3. Number of Echo Replies sent
- 4. Number of Echo Replies received

Data Size Metrics (cont.)

- 5. Total Echo Request bytes sent
  - In bytes, based on the size of the “frame”
- 6. Total Echo Request bytes received
  - In bytes, based on the size of the “frame”
- 7. Total Echo Request data sent
  - In bytes, based on amount of data in the ICMP payload
- 8. Total Echo Request data received
  - In bytes, based on amount of data in the ICMP payload

Time Based Metrics

- These metrics indicate how “quickly” data is getting through the network in terms of time and rate
- 1. Average Ping Round Trip Time (RTT)
  - Ping RTT is defined as the time between sending an Echo Request packet and receiving a corresponding Echo Reply packet from the destination
  - Measured in milliseconds

Time Based Metrics (cont.)

- 2. Echo Request Throughput (in kB/sec)
  - Defined as the sum of the frame sizes of all Echo Request packets sent by the node divided by the sum of all Ping RTTs
- 3. Echo Request Goodput (in kB/sec)
  - Defined as the sum of the ICMP payloads of all Echo Request packets sent by the node divided by the sum of all Ping RTTs

Time Based Metrics (cont.)

- 4. Average Reply Delay (in microseconds)
  - Defined as the time between a node receiving an Echo Request packet and sending an Echo Reply packet back to the source

Distance Metric

- Average number of hops per Echo Request
  - The hop count of an Echo Request is defined as the number of networks that an Echo Request packet must traverse in order to reach its destination
  - Hop count will be 1 if the destination is on a node's network or 3 if it has to go through routers to reach its destination
  - You *cannot* hard code this logic since it's not accurate for any given network, just this topology. (Hint: think about a field in the IP header in the Echo Reply)

General Code Structure

- All of your code should originate in a file called packet_analyzer.py
- Each project phase should be contained in their own .py files
  - **Packet Filtering in filter_packets.py**
  - **Packet Parsing in packet_parser.py**
  - **Compute Metrics in compute_metrics.py**
- See the provided .py files for how to properly import the project phases code into the main code

PCA Tool Grading

- See Mini Project 2 Grading Sheet for details
- You can copy the table from the grading sheet to make your own table to keep track of requirements
- Grades may be adjusted based upon peer reviews
  - Bonus points for heroic effort
  - (Major) point loss for lack of effort

Project Submission

- Submit a single zip file to your group’s project submission dropbox
- The zip file will contain
  - All .py files
  - Four raw input files
  - Four filtered packet capture files
  - Output file containing the metrics computed for each end node (format is provided on myCourses)

Ask for help!

- Don’t suffer in silence. Ask me or your TA for help sooner rather than later!
  - Attend my office hours or the TA’s
  - Make an appointment outside of office hours
  - Send an email

- If you’re not sure if you’ve met a specific requirement, please ask!