1.The point-to-point link delays are 2 ms, but the end-to-end request-response delays are not around 4 ms for the original topology. According to the console output, the end-to-end request-response delay is around 8 ms for the original topology and around 15 ms for the modified topology.

This is because end-to-end delay includes not only point-to-point delay (propagation delay), but also processing, queuing, and transmission delays. The total nodal delay should be calculated by

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

2. Yes, the first end-to-end delay for request-response is significantly longer than the subsequent ones.

In the CSMA channel, the source node needs to know the mac address of the target node, so the first packet transfer needs to use ARP(a network protocol used to find out the hardware (MAC) address of a device from an IP address) and subsequent nodes can directly find the mac address of the target node from the ARP table, which cause the first end-to-end delay is longer than subsequent.

```
zhonghui@lawn-128-61-49-7 ns-3.35 % tcpdump -nn -tt -r Lab1_part2-1-0.pcap
reading from file Lab1_part2-1-0.pcap, link-type PPP (PPP)
2.003686 IP 10.1.1.1.49153 > 10.1.3.2.9: UDP, length 1024
2.027294 IP 10.1.3.2.9 > 10.1.1.1.49153: UDP, length 1024
```

```
zhonghui@lawn-128-61-49-7 ns-3.35 % tcpdump -nn -tt -r Lab1_part2-1-1.pcap
reading from file Lab1_part2-1-1.pcap, link-type EN10MB (Ethernet)
2.011686 ARP, Request who-has 10.1.2.5 (ff:ff:ff:ff:ff:ff) tell 10.1.2.1, length 50
2.011710 ARP, Reply 10.1.2.5 is-at 00:00:00:00:00:09, length 50
2.011710 IP 10.1.1.1.49153 > 10.1.3.2.9: UDP, length 1024
2.027188 ARP, Request who-has 10.1.2.1 (ff:ff:ff:ff:ff:ff) tell 10.1.2.5, length 50
2.027188 ARP, Reply 10.1.2.1 is-at 00:00:00:00:00:05, length 50
2.027294 IP 10.1.3.2.9 > 10.1.1.1.49153: UDP, length 1024
```

Through the pcap file, we can see that the first packet stays in the n1 node for almost 7ms, and the process of ARP execution does not even need 1ms.

Then, through looking at the ArpL3Protocol API documentation.

- **RequestJitter**: The jitter in ms a node is allowed to wait before sending an ARP request. Some jitter aims to prevent collisions. By default, the model will wait for a duration in ms defined by a uniform random-variable between 0 and RequestJitter
  - Set with class: ns3::PointerValue
  - Underlying type: ns3::Ptr< ns3::RandomVariableStream>
  - Initial value: ns3::UniformRandomVariable[Min=0.0|Max=10.0]
  - Flags: construct write read

It can be found that ARP requests needs to wait a random period of time before sending. Hence, Setting the waiting time before ARP sending to 0ms can shorten the first end-to-end delay.

```
//Set 0ms allowed to wait before sending an ARP request.
Config::SetDefault ("ns3::ArpL3Protocol::RequestJitter",
StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=0.0]"));
```