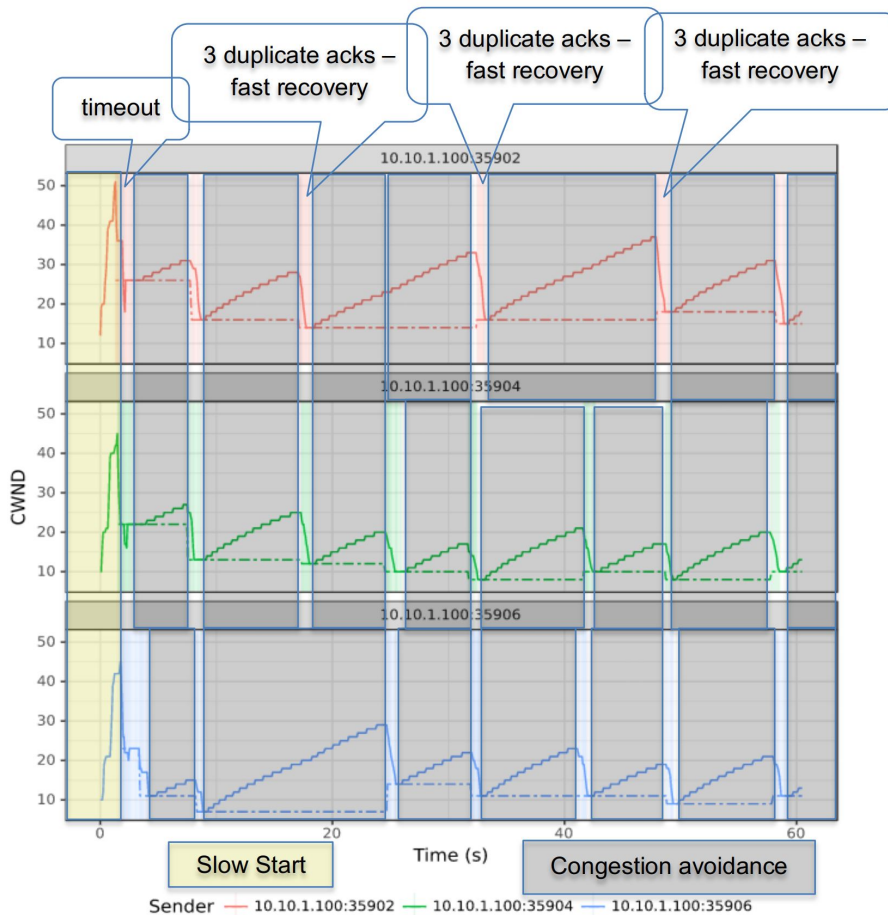


1 TCP Congestion Control

Exercise 1.1

Configure the topology as described in the “Run my experiment” section of the tutorial.

1. Include your annotated plot of the three flows sharing the bottleneck link using TCP Reno.



2. Using your plot and/or experiment data, explain the behaviour of TCP Reno in the slow start and congestion avoidance phases. In addition, explain what happens to both the congestion window and the slow start threshold when multiple duplicate ACKs are received.

- **Slow start** periods, where the congestion window increases rapidly.
- **Congestion avoidance** periods (when the congestion window increases linearly from the slow start threshold)
- **Fast recovery**, TCP Reno, will enter “fast recovery” in response to 3 duplicate ACKs. The slow start threshold is set to half of the CWND at the time of the loss event, the new CWND is set to the slow start threshold, and the flow enters “congestion avoidance” mode. In other words, the receipt of triple duplicate ACKs indicates that some segments are getting through, and so

congestion may be less severe. Thus instead of wasting bandwidth, TCP tries a more aggressive strategy, by starting cwind at the current ssthresh value rather than at 1 MSS. This is $\frac{1}{2}$ the cwind value when the packet was dropped.

- **slow start**, Instances of ACK timeout, if any. This will cause the congestion window to go back to 1 MSS, and the flow enters “slow start” mode.

The behavior of TCP Reno during the slow start and congestion avoidance phases is depicted in the figure 1.

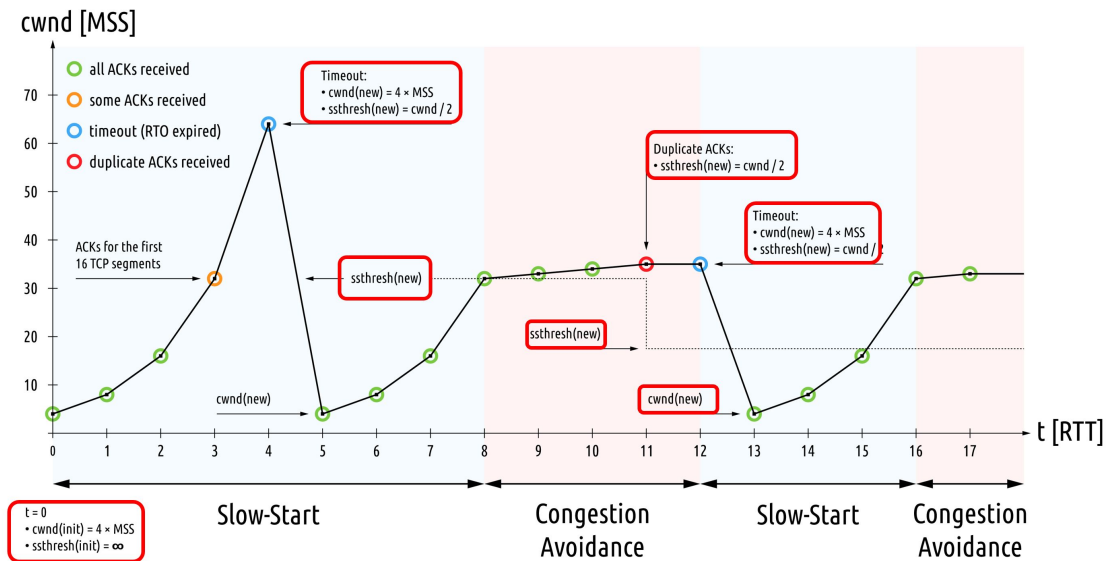


Figure 1: Image is GPLv3, via Wikimedia Commons.

Exercise 1.2

Under the “Optional: Other Congestion Control Algorithms” section of the tutorial, repeat the steps in the “Generating Data” section using the TCP Cubic variant. Download the updated sender-ss.svg image file to your computer. Study the TCP Cubic algorithm and use it to annotate your plot, as you did in Exercise 1.1 for TCP Reno.

1. Include your annotated plot of the three flows sharing the bottleneck link using TCP Cubic.
Figure 2 shows the three flows sharing the bottleneck link. This figure could be different according to the network condition.
2. Using your plot and/or experiment data, explain the behaviour of TCP Cubic.

TCP CUBIC increases the size of cwind on the function of the cube between the current time and the future point of time when cwind will equal the value of cwind before the last loss event when operating in “congestion avoidance.” So, while the current value of cwind is far from the old value of cwind, TCP CUBIC is rapidly increasing. TCP CUBIC gradually increases when the current value of cwind approaches the old value of cwind. After TCP cubic has successfully operated near (either slightly below or slightly above) the old value of cwind for a certain amount of time, it begins to rapidly increase to probe for a new operating point.

3. Why does TCP Cubic reach the available bandwidth faster than TCP Reno?

The TCP Cubic modifies the linear window growth function of existing TCP standards to be a cubic function. It also achieves more equitable bandwidth allocations among flows with different RTTs (round trip times) by making the window growth to be independent of RTT

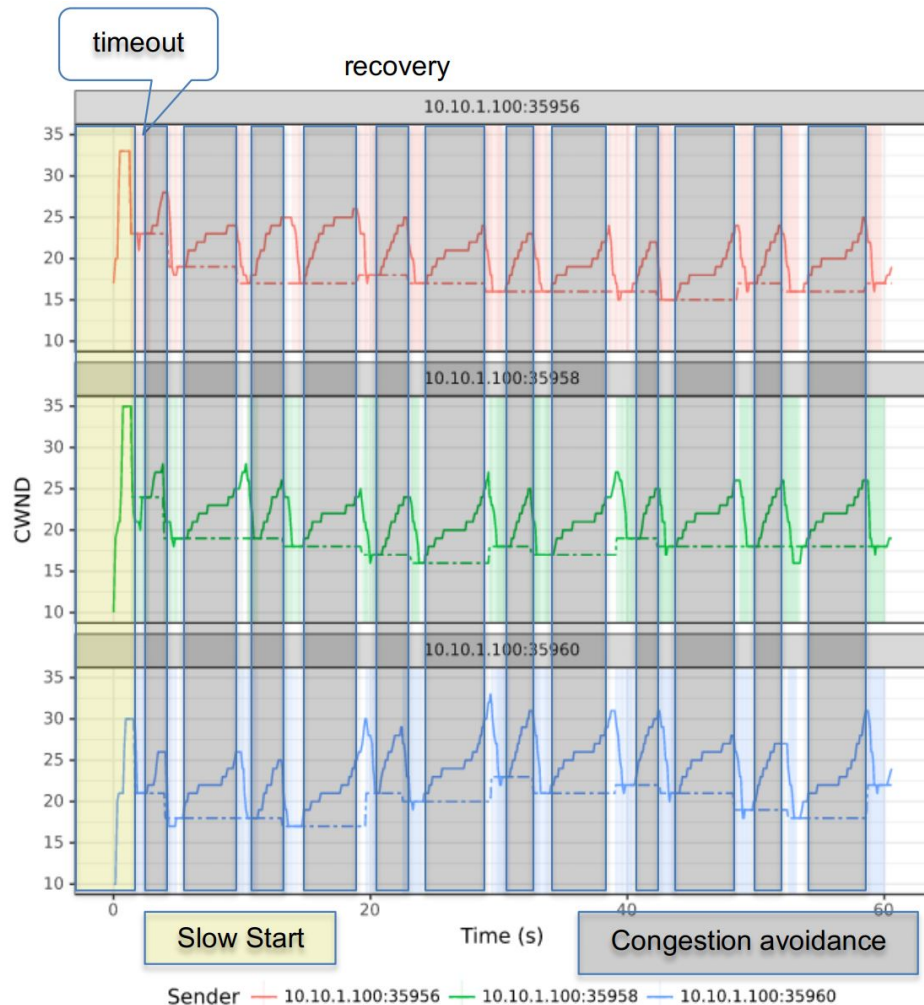


Figure 2: Annotated plot of the three flows sharing the bottleneck link

Exercise 1.3

Under “Optional: Low Delay Congestion Control” section of the tutorial, repeat the steps in the “Generating Data” section using the TCP Vegas variant. Make a note of the iperf3 throughput and the RTT estimated by ping during the TCP Vegas flow. TCP Vegas does not work well when it shares a bottleneck link with a flow using a loss-based algorithm, e.g., TCP Reno. To help understand the problem, you will send one TCP Reno flow and one TCP Vegas flow through the bottleneck router. For your lab report:

1. Make a note of the throughput reported by iperf3 for the TCP Reno flow and the TCP Vegas flow.
Note: the numbers could be different depending on network condition. The bottleneck link is 1 Mbps. The purpose of this question is to demonstrate that TCP Vegas’s performance is extremely poor.
TCP Reno, Throughput=909 Kbps
TCP Vegas, Throughput=60 kbps
2. Using your throughput measurements, and knowledge of the congestion control mechanisms used by each algorithm, can you explain why TCP Vegas does not work well in this scenario?
TCP Reno’s congestion window grows rapidly without considering the delays in the network. TCP Vegas, on the other hand, determines congestion from delays and reduces its send rate accordingly. TCP Vegas is more concerned with avoiding packet losses.

Exercise 1.4

Under “Optional: Explicit Congestion Notification (ECN)” section of the tutorial, use active queue management to configure a queue in both directions on the router, and enable ECN in TCP on each host. Prepare to capture the TCP flow on each host, and then start the experiment. When the experiment finishes, transfer the packet captures (i.e., the .pcap files) to your computer using scp.

1. Compare the delay performance of Reno with ECN to that from the experiment (Exercise 1.3) showing the delay performance without ECN.

When ECN is enabled, the RTT and throughput both decrease. This reduction should be reflected in the answers.

2. Look for the ECN-related fields in the IP header and TCP header, during connection establishment and during data transfer. Annotate your packet capture files by drawing a circle or a box around the fields to show:

- The ECN handshake, i.e., the ECN-setup SYN packet and corresponding ECN-setup SYN-ACK. It can be seen in figure 3.

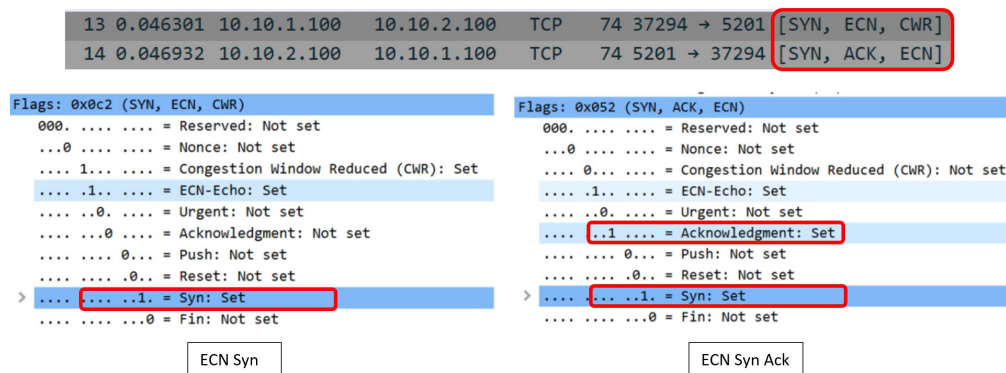


Figure 3: SYN and SYN-ACK packets

- Select a data packet and show the two ECN bits in the IP header set to either 01 or 10. It can be seen in the figure 4.

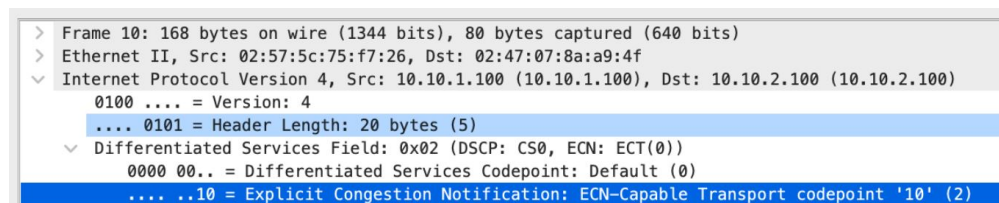


Figure 4: Data packet with the two ECN bits in the IP header set to 10

- Find an instance of a “congestion experienced” signal, i.e., the two ECN bits in the IP header set to 11, and subsequent “congestion window reduced” (CWR) flag in the TCP header of the next packet.

First, the receiver gets the congestion-experienced signal from the router, Figure 5 (ECN = 11).

Second, the receiver replies with an ACK where the ECN-Echo bit is set in the TCP Header.

Third, the sender receives the echo bit and reduces their CWND, figure 6.

```

> Frame 762: 1514 bytes on wire (12112 bits), 80 bytes captured (640 bits)
> Ethernet II, Src: 02:57:5c:75:f7:26, Dst: 02:47:07:8a:a9:4f
> Internet Protocol Version 4, Src: 10.10.1.100 (10.10.1.100), Dst: 10.10.2.100 (10.10.2.100)
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x03 (DSCP: CS0, ECN: CE)
    0000 00.. = Differentiated Services Codepoint: Default (0)
    .... ..11 = Explicit Congestion Notification: Congestion Experienced (3)
  Total Length: 1500

```

Figure 5: Congestion experienced

```

> Flags: 0x090 (ACK, CWR)
  000. .... = Reserved: Not set
  ...0 .... = Nonce: Not set
  .... 1... = Congestion Window Reduced (CWR): Set
  .... .0.. = ECN-Echo: Not set

```

Figure 6: When the sender receives the echo bit, reduce CWND.

2 Basic Home Gateway Services: DHCP, DNS, NAT

Exercise 2.1

In the section “Observe a DHCP request and response:”

1. Take a screen shot showing the DHCP discover message sent by the client to try and find DHCP servers on the LAN. What are the source and destination IP addresses in this request? Why are these addresses used?

```

Select nicolean@gateway: ~
100%[----->] 122 ---K/s in 0s
2020-04-12 15:42:00 (1.48 MB/s) - '/etc/dnsmasq.conf' saved [122/122]
nicolean@gateway:~$ sudo service dnsmasq restart
* Restarting DNS forwarder and DHCP server dnsmasq [ OK ]
nicolean@gateway:~$ sudo tcpdump -i eth1 -n -e -v "udp port 67 or udp port 68"
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
15:42:29.073504 02:18:ef:27:07:7b > ff:ff:ff:ff:ff:ff ethertype IPv4 (0x0800), length 342: (tos 0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17),
length 328)
0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 02:18:ef:27:07:7b, length 300, xid 0x43e45e0e, Flags [none]
Client-Ethernet-Address 02:18:ef:27:07:7b
Vendor-rfc1048 Extensions
Magic Cookie 0x63825363
DHCP-Message Option 53, length 1: Discover
Hostname Option 12, length 10: "<hostname>"
Parameter-Request Option 55, length 13:
Subnet-Mask, BR, Time-Zone, Default-Gateway
Domain-Name, Domain-Name-Server, Option 119, Hostname
Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
NTP

Select nicolean@client-1: ~
nicolean@client-1:~$ sudo dhclient -d eth1
Internet Systems Consortium DHCP Client 4.2.4
Copyright 2004-2012 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth1/02:18:ef:27:07:7b
Sending on LPF/eth1/02:18:ef:27:07:7b
Sending on Socket/fallback
DHCPDISCOVER on eth1 to 255.255.255.255 port 67 interval 3 (xid=0x43e45e0e)
DHCPOFFER of 192.168.100.159 on eth1 to 255.255.255.255 port 67 (xid=0xe5ee443)
DHCPREQUEST of 192.168.100.159 from 192.168.100.1
DHCPACK of 192.168.100.159 from 192.168.100.1
bound to 192.168.100.159 -- renewal in 6121 seconds.

```

Source IP address: 0.0.0.0

Source MAC address: 02:18:ef:27:07:7b

Destination IP address: 255.255.255.255

Destination MAC address: ff:ff:ff:ff:ff:ff

2. Take a screen shot showing the DHCP offer sent by the server. What IP address does the server offer? What is the range of addresses that the server in our experiment may offer? (You can refer to the dnsmasq configuration file.)

```
Select nicolean@gateway: ~
nicolean@gateway:~$ sudo tcpdump -i eth1 -n -e -v "udp port 67 or udp port 68"
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
15:42:29.073504 02:18:ef:27:07:7b > ff:ff:ff:ff:ff:ff, ethertype IPv4 (0x0800), length 342: (tos 0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17), length 328)
0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 02:18:ef:27:07:7b, length 300, xid 0x43e45e0e, Flags [none]
Client-Ethernet-Address 02:18:ef:27:07:7b
Vendor-rfc1048 Extensions
Magic Cookie 0x63825363
DHCP-Message Option 53, length 1: Discover
Hostname Option 12, length 10: "<hostname>"
Parameter-Request Option 55, length 13:
Subnet-Mask, BR, Time-Zone, Default-Gateway
Domain-Name, Domain-Name-Server, Option 119, Hostname
Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
NTP
15:42:32.074331 02:18:69:96:22:8f > 02:18:ef:27:07:7b, ethertype IPv4 (0x0800), length 342: (tos 0xc0, ttl 64, id 25445, offset 0, flags [none], proto UDP (17), length 328)
192.168.100.1.67 > 192.168.100.159.68: BOOTP/DHCP, Reply, length 300, xid 0x43e45e0e, Flags [none]
Your-IP 192.168.100.159
Server-IP 192.168.100.1
Client-Ethernet-Address 02:18:ef:27:07:7b
Vendor-rfc1048 Extensions
Magic Cookie 0x63825363
DHCP-Message Option 53, length 1: Offer
Server-ID Option 54, length 4: 192.168.100.1
Lease-Time Option 51, length 4: 14400
RN Option 58, length 4: 7200
RB Option 59, length 4: 12600
Subnet-Mask Option 1, length 4: 255.255.255.0
BR Option 28, length 4: 192.168.100.255
Domain-Name-Server Option 6, length 4: 192.168.100.1
Default-Gateway Option 3, length 4: 192.168.100.1

Select nicolean@client-1: ~
nicolean@client-1:~$ sudo dhclient -d eth1
Internet Systems Consortium DHCP Client 4.2.4
Copyright 2004-2012 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth1/02:18:ef:27:07:7b
Sending on LPF/eth1/02:18:ef:27:07:7b
Sending on Socket/fallback
DHCPDISCOVER on eth1 to 255.255.255.255 port 67 interval 3 (xid=0x43e45e0e)
DHCPREQUEST of 192.168.100.159 on eth1 to 255.255.255.255 port 67 (xid=0xe45e443)
DHCPOFFER of 192.168.100.159 from 192.168.100.1
DHCPACK of 192.168.100.159 from 192.168.100.1
bound to 192.168.100.159 -- renewal in 6121 seconds.
```

The client is offered the IP address: 192.168.100.159. The client also learns the server's IP address is 192.168.100.1 and that the server should be the default gateway.

3. Take a screen shot showing the DHCP request sent by the client. What is the destination address in this request? Why?

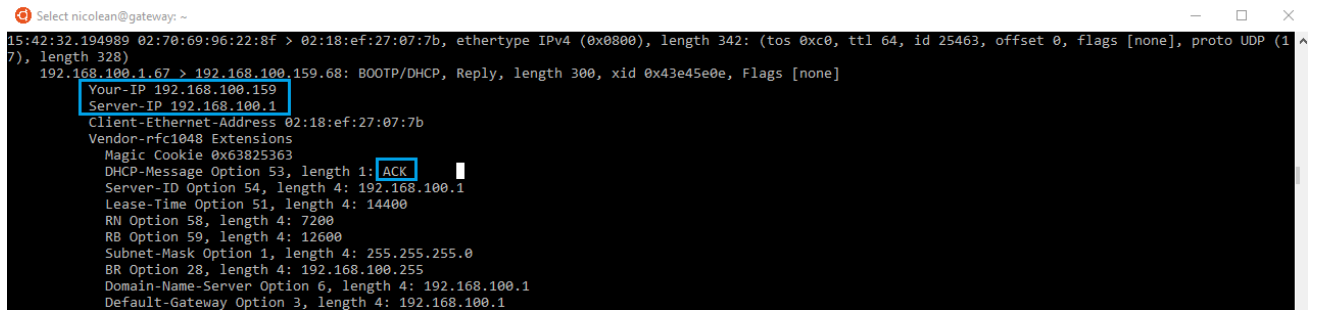
```
Select nicolean@gateway: ~
15:42:32.076906 02:18:ef:27:07:7b > ff:ff:ff:ff:ff:ff, ethertype IPv4 (0x0800), length 342: (tos 0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17), length 328)
0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 02:18:ef:27:07:7b, length 300, xid 0x43e45e0e, Flags [none]
Client-Ethernet-Address 02:18:ef:27:07:7b
Vendor-rfc1048 Extensions
Magic Cookie 0x63825363
DHCP-Message Option 53, length 1: Request
Server-ID Option 54, length 4: 192.168.100.1
Requested-IP Option 50, length 4: 192.168.100.159
Hostname Option 12, length 10: "<hostname>"
Parameter-Request Option 55, length 13:
Subnet-Mask, BR, Time-Zone, Default-Gateway
Domain-Name, Domain-Name-Server, Option 119, Hostname
Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
NTP

Select nicolean@client-1: ~
nicolean@client-1:~$ sudo dhclient -d eth1
Internet Systems Consortium DHCP Client 4.2.4
Copyright 2004-2012 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth1/02:18:ef:27:07:7b
Sending on LPF/eth1/02:18:ef:27:07:7b
Sending on Socket/fallback
DHCPDISCOVER on eth1 to 255.255.255.255 port 67 interval 3 (xid=0x43e45e0e)
DHCPREQUEST of 192.168.100.159 on eth1 to 255.255.255.255 port 67 (xid=0xe45e443)
DHCPOFFER of 192.168.100.159 from 192.168.100.1
DHCPACK of 192.168.100.159 from 192.168.100.1
bound to 192.168.100.159 -- renewal in 6121 seconds.
```

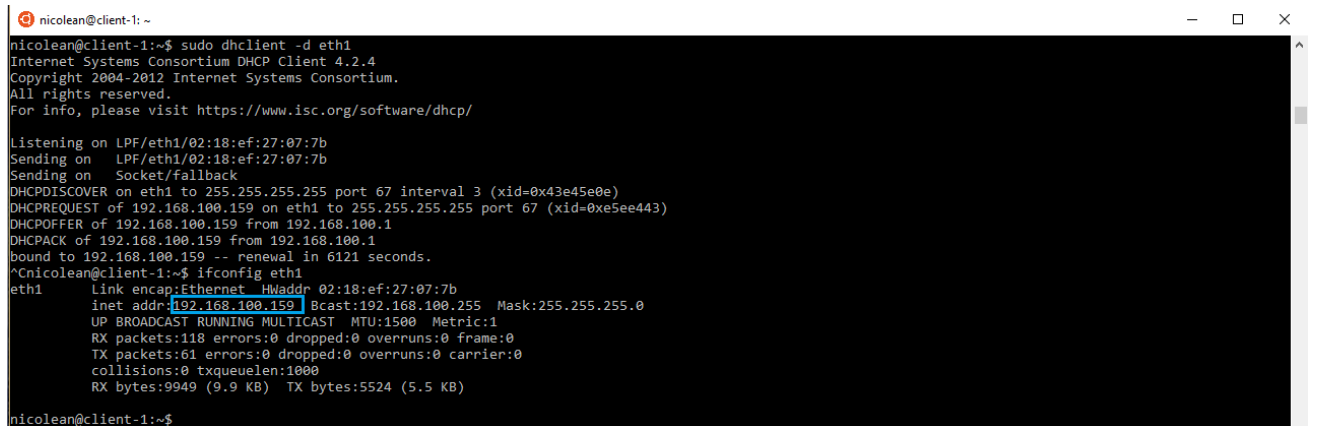
In this example, the client was given only one offer. Therefore, it requested the IP address 192.168.100.159 from the server at 192.168.100.1.

4. Take a screen shot showing the DHCP acknowledgement sent by the server to complete the configuration.



```
Select nicolean@gateway: ~
15:42:32.194989 02:70:69:96:22:8f > 02:18:ef:27:07:7b, ethertype IPv4 (0x0800), length 342: (tos 0xc0, ttl 64, id 25463, offset 0, flags [none], proto UDP (17), length 328)
192.168.100.1.67 > 192.168.100.159.68: BOOTP/DHCP, Reply, length 300, xid 0x43e45e0e, Flags [none]
Your-IP 192.168.100.159
Server-IP 192.168.100.1
Client-Ethernet-Address 02:18:ef:27:07:7b
Vendor-rfc1048 Extensions
Magic Cookie 0x63825363
DHCP-Message Option 53, length 1: ACK
Server-ID Option 54, length 4: 192.168.100.1
Lease-Time Option 51, length 4: 14400
RN Option 58, length 4: 7200
RB Option 59, length 4: 12600
Subnet-Mask Option 1, length 4: 255.255.255.0
BR Option 28, length 4: 192.168.100.255
Domain-Name-Server Option 6, length 4: 192.168.100.1
Default-Gateway Option 3, length 4: 192.168.100.1
```

5. Step 5: Take a screen shot showing the configuration of the client's `eth1` interface after receipt of the DHCP acknowledgement. How can you tell what IP address the client is using? Does this correspond to the IP address in the request and in the acknowledgement?



```
nicolean@client-1: ~
nicolean@client-1:~$ sudo dhclient -d eth1
Internet Systems Consortium DHCP Client 4.2.4
Copyright 2004-2012 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth1/02:18:ef:27:07:7b
Sending on   LPF/eth1/02:18:ef:27:07:7b
Sending on   Socket/fallback
DHCPDISCOVER on eth1 to 255.255.255.255 port 67 interval 3 (xid=0x43e45e0e)
DHCPPREREQUEST of 192.168.100.159 on eth1 to 255.255.255.255 port 67 (xid=0xe5ee443)
DHCPOFFER of 192.168.100.159 from 192.168.100.1
DHCPACK of 192.168.100.159 from 192.168.100.1
bound to 192.168.100.159 -- renewal in 6121 seconds.
^Cnicolean@client-1:~$ ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 02:18:ef:27:07:7b
          inet addr:192.168.100.159  Bcast:192.168.100.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:118 errors:0 dropped:0 overruns:0 frame:0
          TX packets:61 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9949 (9.9 KB)  TX bytes:5524 (5.5 KB)

nicolean@client-1:~$
```

The `ifconfig eth1` command lets us see IP address associated with `eth1`. In this case, the client's `eth1` is shown to be 192.168.100.159. This matches the IP address that was offered, requested, and acknowledged in the previous steps. In addition to receiving a new IP, the client received the default gateway, DNS nameserver, and subnet mask.

Exercise 2.2

In the section “Observe a DNS request and response:”

1. Sample answer to this part is the following:

- (a) Hostname = website.Lab2Grier.ch-geni-net.instageni.colorado.edu
- (b) Type = A
- (c) Address = 192.12.245.168
- (d) ns.emulab.net (155.98.32.70); ns.instageni.colorado.edu (192.12.245.132)
- (e) 192.168.100.1 (gateway acts as DNS server)

```
client-1:~$ dig website.Lab2Grier.ch-geni-net.instageni.colorado.edu

; <<>> DiG 9.11.3-lubuntu.15-Ubuntu <<>> website.Lab2Grier.ch-geni-net.instageni.colorado.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44555
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: aa81e3d79eb6a9ff60fc157b617ed1c63344a61f2624bd75 (good)
;; QUESTION SECTION:
;website.Lab2Grier.ch-geni-net.instageni.colorado.edu. IN A

;; ANSWER SECTION:
website.Lab2Grier.ch-geni-net.instageni.colorado.edu. 1 IN CNAME pcvm2-12.instageni.colorado.edu.
pcvm2-12.instageni.colorado.edu. 30 IN A      192.12.245.168

;; AUTHORITY SECTION:
instageni.colorado.edu. 30      IN      NS      ns.emulab.net.
instageni.colorado.edu. 30      IN      NS      ns.instageni.colorado.edu.

;; ADDITIONAL SECTION:
ns.instageni.colorado.edu. 30      IN      A      192.12.245.132

;; Query time: 3 msec
;; SERVER: 192.168.100.1#53(192.168.100.1)
;; WHEN: Sun Oct 31 11:26:33 MDT 2021
;; MSG SIZE rcvd: 208

tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
11:24:04.139420 IP (tos 0x0, ttl 64, id 34139, offset 0, flags [none], proto UDP (17), length 121)
  192.168.100.109.37160 > 192.168.100.1.53: 40984+ [lau] A? website.Lab2Grier.ch-geni-net.instageni.colorado.edu. (93)
```

Figure 7: Dig and TCP dump output

2. Sample answer to this part is the following:

For the hierarchical DNS resolution with +trace, take a screen shot of the dig command and its output. Draw a diagram showing how the hostname was resolved recursively, starting from the implied “ ” at the end and moving toward the beginning.


```

<>> dig 9.11.3-ubuntu1.15-Ubuntu <>> +trace website.Lab2Grier.ch-geni-net.instageni.colorado.edu
;; global options: +cmd
3600000 IN NS E.ROOT-SERVERS.NET.
3600000 IN NS H.ROOT-SERVERS.NET.
3600000 IN NS K.ROOT-SERVERS.NET.
3600000 IN NS L.ROOT-SERVERS.NET.
3600000 IN NS D.ROOT-SERVERS.NET.
3600000 IN NS I.ROOT-SERVERS.NET.
3600000 IN NS C.ROOT-SERVERS.NET.
3600000 IN NS A.ROOT-SERVERS.NET.
3600000 IN NS G.ROOT-SERVERS.NET.
3600000 IN NS J.ROOT-SERVERS.NET.
3600000 IN NS B.ROOT-SERVERS.NET.
3600000 IN NS M.ROOT-SERVERS.NET.
3600000 IN NS F.ROOT-SERVERS.NET.
;; Received 881 bytes from 192.12.245.132#53(192.12.245.132) in 1 ms

edu. 172800 IN NS l.edu-servers.net.
edu. 172800 IN NS b.edu-servers.net.
edu. 172800 IN NS c.edu-servers.net.
edu. 172800 IN NS d.edu-servers.net.
edu. 172800 IN NS e.edu-servers.net.
edu. 172800 IN NS f.edu-servers.net.
edu. 172800 IN NS g.edu-servers.net.
edu. 172800 IN NS a.edu-servers.net.
edu. 172800 IN NS h.edu-servers.net.
edu. 172800 IN NS i.edu-servers.net.
edu. 172800 IN NS j.edu-servers.net.
edu. 172800 IN NS k.edu-servers.net.
edu. 172800 IN NS m.edu-servers.net.
edu. 86400 IN DS 28065 8 2 4172496CDE85534E51129040355BD04B1FCFEBAE996DFDDE652006F6 F8B2CE76
edu. 86400 IN RRSIG DS 8 1 86400 20211113170000 20211031160000 14748 . IZIDWagfecn09v3LlIYldCzw0o0uYs8Lit+4KjhyuKV
9nAlmbS+EK j3yKeL256EwyDN013Ck18P4g4FkWT+4r1Zj3AudLY2Iio9zhtzw8k6n2 NRYVSmQqU7pqe0Xvy7u16qHCG+BVmaEV4HctryfifHaoCeiWymnUS2D K08rHuIBEspWok7A7JTLH
7QWp+EXoK1xvGyIFC5s8911JRTCD3IegFoFu +p1Lh8eu/Tmin0wETv1qF01/hrZgDqNHxGr2HpbziFvOeSnK//KLrE AhRE0AKS+ACVdenoSRSZoUSf2BLOhrfliuJMGzNgj8swSFgxQ/TI
9KIUU WlPhvQ==
;; Received 1211 bytes from 192.5.5.241#53(F.ROOT-SERVERS.NET) in 2 ms

colorado.edu. 172800 IN NS boulder.colorado.edu.
colorado.edu. 172800 IN NS otis.colorado.edu.
colorado.edu. 172800 IN NS oldduke.colorado.edu.
9DMS4EF8S88FF8NF06HEK0048QK77.edu. 86400 IN NSEC3 1 1 0 - 9D04071UB4JHDMF7CJMFNMQR40F7M NS SOA RRSIG DNSKEY NSEC3PARAM
9DMS4EF8S88FF8NF06HEK0048QK77.edu. 86400 IN RRSIG NSEC3 8 2 86400 20211104084048 20211030043048 48248 edu. ZcFXpMpTuch2cmC5g0h4CqHpg2LKERFHH
1i13lp55zIP3apq4aDmf QcdHtGNz0LjHd18200XYFwhKp00i5JIGUgUc/q15Bwz0RIFc04Vob NkMaYegUUnWj2AIOE1Y1F0pYXHO2BganJQ4H03fckxS8EwUCS6HxuaL Ffr6OI+n+
48UXBLMFLBWS18B1A3cUc6c1IPnF2FFayv==
LK0ACNHV5IOEKKNI23QG9MIREOTG7JT2.edu. 86400 IN NSEC3 1 1 0 - LUVUUTD2PSN4F7KSFBGETTDDI3E20J2 NS DS RRSIG
LK0ACNHV5IOEKKNI23QG9MIREOTG7JT2.edu. 86400 IN RRSIG NSEC3 8 2 86400 20211105054106 20211029043106 48248 edu. QHutuJJ+WqcnMAJb11/HLwNof4VxI93iFGu
ZFG/x9kubd94DgukZik12 OgJ76cL5mwFKyitEohcf9161hRA1u0shKF6RulciUcnb7vqAq2aqv0LT 7g/gLi7n2SbW4S6gzYCFVHrIdTcvckmsat6fIyWzq0fj4dba7X081yNS 3kXsPHTLL
8BqM+ga6ZqaEuq5imM1S8zFzFGQgMhtPmt4A==
;; Received 741 bytes from 192.52.178.30#53(k.edu-servers.net) in 26 ms

instageni.colorado.edu. 3600 IN NS ns.emulab.net.
instageni.colorado.edu. 3600 IN NS ns.instageni.colorado.edu.
;; Received 185 bytes from 128.138.129.76#53(otis.colorado.edu) in 1 ms

website.Lab2Grier.ch-geni-net.instageni.colorado.edu. 1 IN CNAME pcvm2-12.instageni.colorado.edu.
pcvm2-12.instageni.colorado.edu. 30 IN A 192.12.245.168
instageni.colorado.edu. 30 IN NS ns.emulab.net.
instageni.colorado.edu. 30 IN NS ns.instageni.colorado.edu.
;; Received 224 bytes from 155.98.32.70#53(ns.emulab.net) in 12 ms

```

Figure 8: Screen shot of the dig command and its output

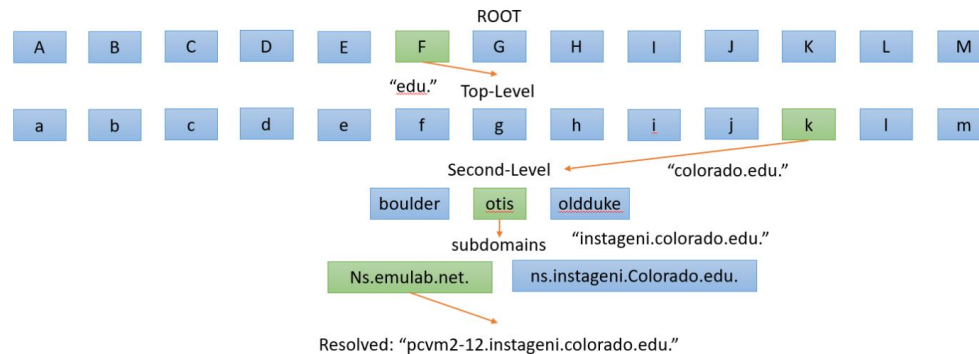


Figure 9: Sample diagram showing how the hostname was resolved recursively

Exercise 2.3

Follow the instructions in the section “Use NAT” set up the gateway to use NAT. For your lab report:

1. Take screen shots to show the three-way TCP handshake for a connection between client and website as seen by tcpdump at the website, and as seen by tcpdump at the gateway (on the LAN). Make sure you can see the IP addresses and port numbers used in the connection. sample answer would be similar to figure 10.
2. Draw a diagram showing how NAT is used between client and website, similar to this diagram but with the IP addresses, hostnames, and ports from your experiment.

Gateway (LAN)

```
wgrier@gateway:~$ sudo tcpdump -i eth1 -n "tcp port 80"
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
11:45:37.065296 IP 192.168.100.109.55128 > 192.12.245.168.80: Flags [S], seq 144419505, win 64240, options [
11:45:37.066277 IP 192.12.245.168.80 > 192.168.100.109.55128: Flags [S.], seq 1675455245, ack 144419506, win
11:45:37.066999 IP 192.168.100.109.55128 > 192.12.245.168.80: Flags [.], ack 1, win 502, options [nop,nop,TS va
```

Website (WAN)

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
11:45:30.028765 IP 172.17.4.29.55128 > 192.12.245.168.80: Flags [S], seq 144419505, win 64240, options [mss
length 0
11:45:30.028853 IP 192.12.245.168.80 > 172.17.4.29.55128: Flags [S.], seq 1675455245, ack 144419506, win 65
6045596,nop,wscale 7], length 0
11:45:30.030186 IP 172.17.4.29.55128 > 192.12.245.168.80: Flags [.], ack 1, win 502, options [nop,nop,TS va
```

Figure 10: 3-Way Handshake

The diagram should be similar to figure 11.

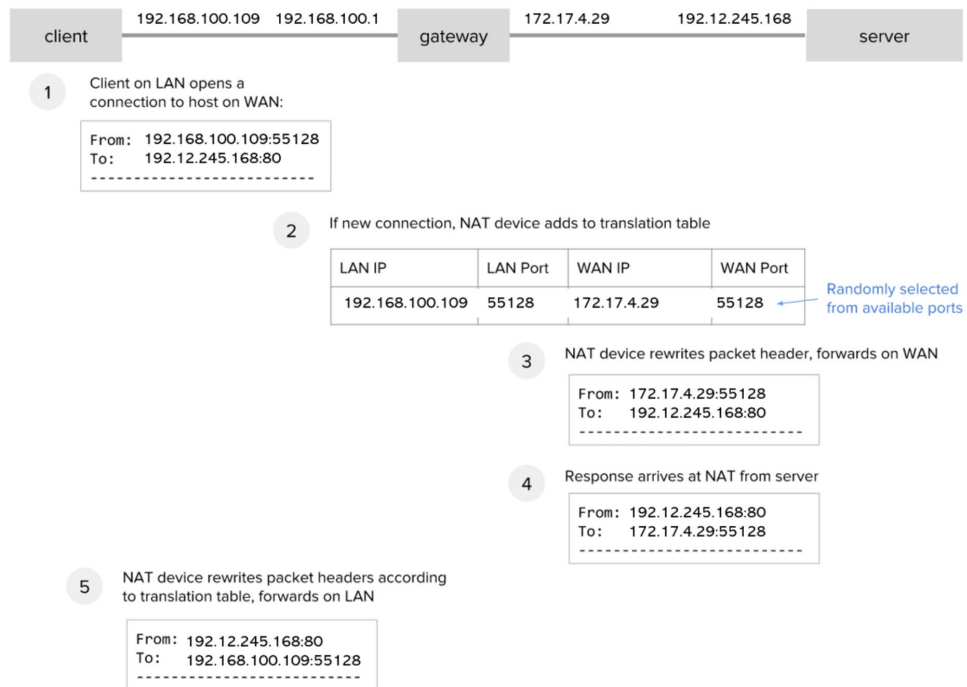


Figure 11: Nat diagram