

# Internet Networking - Homework 3

Yosef Goren & Ori Evron

May 26, 2023

## Contents

<b>1</b>	<b>BGP Business Relations</b>	<b>1</b>
<b>2</b>	<b>Multicast DNS LAB</b>	<b>1</b>
2.1	.....	1
2.2	.....	2
2.3	.....	2
2.4	.....	3
<b>3</b>	<b>Stiner Tree</b>	<b>3</b>
<b>4</b>	<b>Mininet BGP LAB</b>	<b>3</b>
4.1	.....	3
4.2	.....	3
4.3	.....	4
4.4	.....	4
4.5	.....	5
4.6	.....	5
4.7	.....	5
4.8	.....	5
4.9	.....	6
4.10	.....	7

## 1 BGP Business Relations

## 2 Multicast DNS LAB

Solutions in this sections based on specification RFC-6762.

### 2.1

The packets were sent to the IPv4 address 224.0.0.251, this IPv4 address is reserved for mDNS, which means a host can't have it.

*eth0							
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help							
mdns							
No.	Time	Source	Destination	Protocol	Length	Info	
16	14.895673567	fe80::4ba:ccb2:92b5:f5b6	ff02::fb	MDNS	211	Standard	query 0x0000 ANY 6.
17	14.895842012	10.0.2.15	224.0.0.251	MDNS	245	Standard	query 0x0000 ANY 6.
19	15.145960297	fe80::4ba:ccb2:92b5:f5b6	ff02::fb	MDNS	211	Standard	query 0x0000 ANY 6.
20	15.146125816	10.0.2.15	224.0.0.251	MDNS	245	Standard	query 0x0000 ANY 6.
22	15.464008670	fe80::4ba:ccb2:92b5:f5b6	ff02::fb	MDNS	211	Standard	query 0x0000 ANY 6.
23	15.464454723	10.0.2.15	224.0.0.251	MDNS	245	Standard	query 0x0000 ANY 6.
24	15.663825782	10.0.2.15	224.0.0.251	MDNS	227	Standard	query response 0x00
25	15.664022402	fe80::4ba:ccb2:92b5:f5b6	ff02::fb	MDNS	199	Standard	query response 0x00
26	16.794209356	10.0.2.15	224.0.0.251	MDNS	227	Standard	query response 0x00
27	16.794666140	fe80::4ba:ccb2:92b5:f5b6	ff02::fb	MDNS	199	Standard	query response 0x00
28	18.900328955	10.0.2.15	224.0.0.251	MDNS	227	Standard	query response 0x00
29	18.900977688	fe80::4ba:ccb2:92b5:f5b6	ff02::fb	MDNS	199	Standard	query response 0x00

  

▶ Frame 16: 211 bytes on wire (1688 bits), 211 bytes captured (1688 bits) ▶ Ethernet II, Src: PcsCompu_c7:e1:36 (08:00:27:c7:e1:36), Dst: IPv6mcast_ ▶ Internet Protocol Version 6, Src: fe80::4ba:ccb2:92b5:f5b6, Dst: ff02::f ▶ User Datagram Protocol, Src Port: 5353, Dst Port: 5353 ▶ Multicast Domain Name System (query)	0000 33 33 00 00 00 fb 08 00 0010 d8 1c 00 9d 11 ff fe 80 0020 cc b2 92 b5 f5 b6 ff 02 0030 00 00 00 00 00 fb 14 e9 0040 00 00 00 02 00 00 00 02 0050 01 66 01 35 01 62 01 32 0060 01 63 01 61 01 62 01 34 0070 01 30 01 30 01 30 01 30
---	--

## 2.2

- **Advantage:** Other hosts now know and can cache the MDNS to IP mapping so that when they require it - they do not need to request for it; moreover - they will be able to answer queries of others regarding the same hostname.
- **Disadvantage:** Sending the response using broadcast can and will cause un-necessary traffic since it will often send the response packets to hosts that already know the mapping or ones that otherwise will not make use of it.

## 2.3

The main reason we can think of is that broadcast generally cannot be sent outside of the same LAN<sup>1</sup>, as opposed to multicast which can - so using broadcast would limit the availability of the MDNS network to just one LAN, or would

<sup>1</sup>broadcast is generally not possible from outside networks since enabling it would expose the receiving network to DOS attacks - hence it is often disallowed at the firewall.

require solving further issues that would come with using broadcast in remote networks.

## 2.4

There are generally 3 options for how the end users or their client applications might be able to know the addresses of internet devices.

1. Standard DNS.
2. Manually searching for the IP address allocated by DHCP.
3. Using MDNS.

**Option 1** would not be practical for the purpose of IOT devices, as it would require each such device to have its own hostname at the global DNS (which would be very expensive) - or would cause the user to manually configure a local DNS service which is not a reasonable request from the end user.

Similarly - **Option 2** also requires manual configuration and knowledge that the end user does not have to implement.

Hence the best option we are left with is **Option 3** (MDNS) - which enables 'Plug and Play' usage of the IOT devices.

## 3 Stiner Tree

## 4 Mininet BGP LAB

### 4.1

The address is 192.168.1.1.

### 4.2

The routing table consists of a list of pathways for the way to each network as shown:

```

R1> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

C>* 1.1.1.1/32 is directly connected, lo
B>* 2.2.2.0/24 [20/0] via 192.168.1.2, R1-eth1, 00:03:16
B>* 3.3.3.0/24 [20/0] via 192.168.1.2, R1-eth1, 00:02:46
B>* 4.4.4.0/24 [20/0] via 192.168.1.2, R1-eth1, 00:02:16
B>* 5.5.5.0/24 [20/0] via 192.168.1.2, R1-eth1, 00:01:46
B>* 6.6.6.0/24 [20/0] via 192.168.1.2, R1-eth1, 00:02:46
B>* 7.7.7.0/24 [20/0] via 192.168.1.2, R1-eth1, 00:02:16
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.1.0/24 is directly connected, R1-eth1
R1> enable

```

It appears all of the routes are through 192.168.1.2, except the loopback connection, and the 192.168.1.0 device.

Different entries in the table are controlled by different applications or methods: 1.1.1.1, loopback and 192.168.1.0 have been added to the table since they are directly connected to the device, while the rest of the entries in the table have been added by the BGP protocol.

The timestamp at the end of the BGP entries shown when BGP has added the specific entry to the table.

### 4.3

Different parts of the table are filled in different ways; the entries for directly connected devices are added automatically - likely by the operating system or an associated daemon, while the indirect routings marked with 'B' have been added by the local BGP application (bgpd).

Generally - different entries in the table are inserted from different sources, and each such source might add its own entries based on how the specific protocol is defined.

### 4.4

The data stored in the BGP table represents the next hop, which is the next router on our network where we want our messages to be sent. In our case, where AS1 is only connected to AS2, all our messages need to be routed through AS2 to reach their destinations. We can also see the path the packets need to go through in the net...

## 4.5

The route from R1 to R5 shown in the BGP table is shown as "0 2 3 4 5 i", which means the route goes through R3 and R4.

The reason the path going through R6 and R7 is not shown is because it is was not considered an optimal path by BGP <sup>2</sup> and hence it was not saved.

## 4.6

R2 shows two routes to R5, one through R6 and the other through R3; the one through R3 is the one used, as it is the shortest one considering the AS-hop-count metric.

## 4.7

If all metrics are equal to zero, then the routing is dictated by taking the shortest path in terms of the number of hops between AS's, if that does not break the tie - then random / lexicographic selection is used.

## 4.8

It seems the path in the bgp table from R5 to R1 0 4 7 6 2 1 3 1 i.

---

<sup>2</sup>The metric is determined by the configuration - the number of hops between AS's and the weight/cost of each hop

```
"Node: R5"
Hello, this is Quagga (version 0.99.22.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

User Access Verification

Password:
bgpd-R5> show ip bgp
BGP table version is 0, local router ID is 5.5.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
               r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network        Next Hop        Metric LocPrf Weight Path
*> 1.1.1.0/24      192.168.4.1          0  4 7 6 2 1 3 1 i
*> 2.2.2.0/24      192.168.4.1          0  4 3 2 i
*> 3.3.3.0/24      192.168.4.1          0  4 3 i
*> 4.4.4.0/24      192.168.4.1          0    4 i
*> 5.5.5.0/24      0.0.0.0             0 32768 i
*> 6.6.6.0/24      192.168.4.1          0  4 7 6 i
*> 7.7.7.0/24      192.168.4.1          0  4 7 i

Total number of prefixes 7
bgpd-R5>
```

#### 4.9

The AS manager added the prefix 3 1 to

```

!
hostname bgpd-R1
password en
enable password en
log stdout
!
router bgp 1
  bgp router-id 1.1.1.1
  network 1.1.1.0/24
  neighbor 192.168.1.2 remote-as 2
  neighbor 192.168.1.2 route-map rm_peer_1_out out
!
ip prefix-list pl_peer_1_out seq 5 permit 1.1.1.0/24
!
route-map rm_peer_1_out permit 5
  match ip address prefix-list pl_peer_1_out
  set as-path prepend 3 1
!
line vty
!

```

#### 4.10

```

R4> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

B>* 1.1.1.0/24 [20/0] via 192.168.7.2, R4-eth3, 01:44:32
B>* 2.2.2.0/24 [20/0] via 192.168.3.1, R4-eth1, 01:45:10
B>* 3.3.3.0/24 [20/0] via 192.168.3.1, R4-eth1, 01:45:10
C>* 4.4.4.4/32 is directly connected, lo
B>* 5.5.5.0/24 [20/0] via 192.168.4.2, R4-eth2, 01:45:07
B>* 6.6.6.0/24 [20/0] via 192.168.7.2, R4-eth3, 01:45:02
B>* 7.7.7.0/24 [20/0] via 192.168.7.2, R4-eth3, 01:45:02
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.3.0/24 is directly connected, R4-eth1
C>* 192.168.4.0/24 is directly connected, R4-eth2
C>* 192.168.7.0/24 is directly connected, R4-eth3
R4>

```

we can see in the line 192.168.7.0 is directly connected, R4-