LPIC-1 102-400 – Lesson 14

109.1 Fundamentals of internet protocols

The *TCP/IP* Protocol Suite

- The TCP/IP (Transport Control Protocol/Internet Protocol) protocol suite, is a collection of protocols used in the Internet and Networking in general
- All modern operating systems have their own “TCP/IP stack”
- The TCP/IP stack consist of 4 (in some literature 5) layers: **Application**, **Transport**, **Internet** and **Link**
- Most protocols belong to one layer but there are some protocols that cross layers
- The basic protocols we will examine are: TCP, UDP, IP (IPv4, IPv6), ICMP and others

TCP/IP model

Layer	Description	Protocols
Application	Provides communication between network applications, session management and data presentation in a human readable form	HTTP, SMTP, DNS, DHCP
Transport	Provides transport of messages and service separation using **ports**. Provides reliability, error correction flow control and data segmentation	TCP, UDP
Internet	Responsible for routing data packets in an internetwork. IP addresses are defined here	IP (IPv4, IPv6), ICMP
Link	It is responsible for converting the data for transport into the physical elements of a network. Not realy a part of TCP/IP	Ethernet, Wi-Fi Token Ring, PPP, SLIP

The format of an IPv4 address

1010110000001111 0001100000000110 - 172.16.24.6

- An IPv4 address is 32 bit long and represented in 4 octets (bytes) in decimal separated by dots (dotted-decimal notation).
- Every address is separated in two portions:
  - **Network portion**: defines the network where the address belongs
  - **Host portion**: defines the unique host id that represents the host in an IPv4 network
- Avery address is unique in a network
- Addresses are assigned statically or dynamically through the DHCP protocol

The format of an IPv6 address

2001:0db8:0000:dead.0000:0000:0000:beef

2002:db8.0.dead::beef

- An IPv6 address is128 bit long and represent in 8 hextets (16 bit words) in hexadecimal separated by colon
- Every address is separated in two portions:
  - **Network portion**: defines the network where the address belongs
  - **Host portion**: defines the unique host id that represents the host in an IPv6 network
- Avery address is unique in a network
- Addresses are assigned statically or dynamically through SLAAC (stateless), DHCP (stateful) or a combination of both

Finding the boundaries of an IPv4 network using subnet masks

- We have the following IPv4 address:
  215.25.17.45 with mask **255.255.255.192 (/26)**

```
$ ipcalc 215.25.17.45 255.255.255.192
Address:   215.25.17.45          11010111.00011001.00010001.00 101101
Netmask:   255.255.255.192 = 26 11111111.11111111.11111111.11 000000
=>
Network:   215.25.17.0/26        11010111.00011001.00010001.00 000000
HostMin:   215.25.17.1           11010111.00011001.00010001.00 000001
HostMax:   215.25.17.62          11010111.00011001.00010001.00 111110
Broadcast: 215.25.17.63          11010111.00011001.00010001.00 111111
Hosts/Net: 62                     Class C
```

- The 26 most significant bits of 215.25.17.0 represent the network and the other 6 represent the unique network id

Subnetting a bigger network

- If we want to segment the **215.25.17.0/255.255.255.0** networks in smaller subnets using the mask **255.255.255.224** we will get 8 subnets

```
$ ipcalc  215.25.17.0 255.255.255.0 255.255.255.224 | grep -A1 '[1-8]\.$'
 1.
Network:   215.25.17.0/27       11010111.00011001.00010001.000 00000
 2.
Network:   215.25.17.32/27      11010111.00011001.00010001.001 00000
 3.
Network:   215.25.17.64/27      11010111.00011001.00010001.010 00000
 4.
Network:   215.25.17.96/27      11010111.00011001.00010001.011 00000
 5.
Network:   215.25.17.128/27     11010111.00011001.00010001.100 00000
 6.
Network:   215.25.17.160/27     11010111.00011001.00010001.101 00000
 7.
Network:   215.25.17.192/27     11010111.00011001.00010001.110 00000
 8.
Network:   215.25.17.224/27     11010111.00011001.00010001.111 00000
```

Variable length subnet masks

- It is not always efficient to use the same subnet mask for all the networks because we may have different needs in each one. Let's segment the network **215.25.17.0/255.255.255.0** in other subnets with different masks

```
$ ipcalc  215.25.17.0 255.255.255.192 | egrep -i "(network|broadcast)"
Network:   215.25.17.0/26       11010111.00011001.00010001.00 000000
Broadcast: 215.25.17.63         11010111.00011001.00010001.00 111111

$ ipcalc  215.25.17.64 255.255.255.224 | egrep -i "(network|broadcast)"
Network:   215.25.17.64/27      11010111.00011001.00010001.010 00000
Broadcast: 215.25.17.95         11010111.00011001.00010001.010 11111

$ ipcalc  215.25.17.96 255.255.255.224 | egrep -i "(network|broadcast)"
Network:   215.25.17.96/27      11010111.00011001.00010001.011 00000
Broadcast: 215.25.17.127        11010111.00011001.00010001.011 11111

$ ipcalc  215.25.17.128 255.255.255.248 | egrep -i "(network|broadcast)"
Network:   215.25.17.128/29     11010111.00011001.00010001.10000 000
Broadcast: 215.25.17.135        11010111.00011001.00010001.10000 111
```

Private IP Addresses

- Because of the exhaustion of IPv4 addresses internationally, the private IPv4 address were created. These IP addresses are not routable over the Internet and they are supposed to be used only on internal networks
- If a computer from a private network wished to access the Internet, from a private network, its private address must be “translates” to a Public IPv4 address using the “Network Address Translation” (NAT) mechanism.
- Every private network that accesses the Internet should have one or more public ipv4 addresses. The private addresses, which are usually more than the public addresses, are overloaded to the public address. That means on public address may represent more than one private address

Private IP Addresses

Address block	IPv4 Address range	Number of addresses
10.0.0.0/8 (255.0.0.0)	10.0.0.0 – 10.255.255.255	16.777.216
172.16.0.0/12 (255.240.0.0)	172.16.0.0 – 172.31.255.255	1.048.576
192.168.0.0/16 (255.255.0.0)	192.168.0.0 – 192.168.255.255	65.536

The solution: IPv6

- Version 6 of IP (IPv6) was created to counter the problem of IPv4 exhaustion
- There are many improvements but the most important in the use of 128 bit address which come up to: $2^{128} = 3,4 \times 10^{38}$ addresses!
- The addresses are represented in hexadecimal and separated with "**:**" in 16 bit words, e.g.:
- **2001:0db8:85a3:0000:0000:8a2e:0370:7334 -> 2001:db8:85a3:0:0:8a2e:370:7334 -> 2001:db8:85a3::8a2e:370:7334**
- **0:0:0:0:0:0:0:1 -> ::1 (loopback address) , 0:0:0:0:0:0:0:0 -> :: (any address)**
- **2001:db8:a::/64** (/64 is the network prefix)

Special addresses

- **loopback**: the network **127.0.0.0/8** is used to test the health of the TCP/IP stack. The **127.0.0.1** is set on the **lo** interface and the **localhost** hostname resolves to 127.0.0.1. The 127.0.0.0/8 can not be used for routing neither on the Internet or internal networks
- **link-local (APIPA)**: Addresses from **169.254.1.0** to **169.254.254.255** are used for the automatic assignment of an IP on a network interface card (e.g. Ethernet) when there is no DHCP in the network. The nodes with link-local IPs can only communicate within the same network segment with other link-local IPs. They cannot reach nodes outside their network because link-local is not supposed to be routable

TCP/IP protocols

- **IP (Internet Protocol)**: it is the backbone of TCP/IP and used by almost every other protocol
  - It’s basic task is routing data from one network to the other using IP addresses.
  - It is unreliable i.e. it does not provide error correction, re-transitions works on the best-effort principle
  - It does not provide flow control)
  - It is connectionless
  - It is implemented at the Internet layer

TCP/IP protocols

- **TCP (Transport Control Protocol)**: it is the basic protocol for creating connections between applications. It transports data using ports, which are essentially service ids.
  - It is reliable
  - It provides flow control
  - It is connection-oriented
  - It has a bigger overhead comparing to UDP
  - It only supports **unicast** i.e. communication between two nodes
  - Implemented at the transport layer of the TCP/IP model and uses IP for routing

TCP/IP protocols

- **UDP (User Datagram Protocol)**: it is implemented at the transport layer and just like TCP it uses ports to send data in the form of datagrams
  - It is unreliable
  - It provides no flow control
  - It is connectionless
  - It is faster than TCP because of the lower overhead
  - Supports unicast, broadcast and multicast
  - It is implemented at the transport layer and uses IP for routing

TCP/IP protocols

- **ICMP (Internet Control Message Protocol)**: it is used for troubleshooting and notifying other protocols about the behavior of the network
  - Flow control: notifies TCP about network congestions
  - Notifies other protocol about unreachable destinations (Destination Unreachable)
  - Re-routing of network paths (Route Redirection)
  - Checking remote destination e.g. using the **ping** command
  - It is connectionless
  - It is implemented at the Internet layer and uses IP for routing

Ports and services

- Ports are implemented at the transport layer by **TCP** and **UDP**, for identifying services on a network node
- To transfer a data packet, the application need to know the IP address of the node and the id of the service (aka port) for which the data is destined to
- A node (server) can offer many services and these are distinguished by ports
- Services (aka applications) are implemented at the application layer while ports are implemented at the transport layer
- Some examples of application protocols of TCP/IP: ftp, ssh, http, dns etc
- The **/etc/services** file contains a list of well-known services and the ports they use

Well-known ports and services

Service	Port(s)	Description
FTP (File Transfer Protocol)	20 (data), 21 (control)/TCP	FTP is used for file transfers over the Internet
SSH (Secure Shell)	22/TCP	Secure Remote control protocol
TELNET	23/TCP	Insecure remote control protocol
SMTP, SMTPS (Simple Mail Transfer Protocol)	25, 465 (TLS), 587 (STARTTLS)/TCP	Mail Sending Protocol
DNS (Domain Name Service)	53/TCP-UDP	Resolving hostnames to IPs
DHCP	67 (Server), 68 (Client)/UDP	Automatic IP Address assignment
HTTP/HTTPS (HyperText Transfer Protocol)	80, 443 (TLS)/TCP	World Wide Web
POP3, POP3S (Post Office Protocol)	110, 995 (TLS)/TCP	Receiving mail locally
Netbios	139/TCP-UDP	File/printer sharing for Windows networks
IMAP, IMAPS (Internet Message Access Protocol)	143, 993 (SSL)	Corporate Mail receiving (on a cental server)
SNMP (Simple Netowork Management Protocol)	161, 162/TCP-UDP	Monitoring and Netork Management
LDAP (Lightweight Directory Access Protocol)	389, 636 (TLS)/TCP	Directory Information/Authentication Protocol
Syslog (System Log Protocol)	514/UCP, 6514 (TLS)/TCP	Sending logs over the network

Connect to FTP servers with `ftp`

- The **ftp** command is a client for connecting to ftp servers via CLI
- **`# ftp ftp.debian.org`** `# connect to ftp.debian.org`
- **`# ftp -v ftp.debian.org`** `# connect in verbose mode`
- Commands:

```
ftp> ls # list files/directories
ftp> cd dir # change into directory dir
ftp> get file1 # get file file1
ftp> mget file[1-9] # get multiple files file1,
file2, ..., file9
ftp> put file2 # upload file file2 from local
working directory
ftp> mput file[a-f] # upload multiple files
ftp> pwd # print working directory on server
ftp> quit # = exit. Exit ftp server
```

Connect to services with `telnet`

- The **telnet** command was used in the past for shell access on remote nodes. Because of its inhered weakness to send everything in cleartext, it was replaced by **ssh** which supports encryption
- Nevertheless it is a useful troubleshooting tool for non encrypted services like HTTP, SMTP etc

- **$ telnet telehack.com** # connect to telehack.com
- **$ telnet www.debian.org 80** # connect to the debian webserver for
              # checking the service
  GET
- **$ telnet mail.theo-andreou.org 25** # connect to mailserver for basic
       # health check
  quit
- **$ telnet towel.blinkenlights.nl** # try and see! :)

Query DNS servers with `host`

- The **host** command queries DNS servers for DNS records
- **$ host theo-andreou.org** # query for A, CNAME (and MX if exist) records
- **$ host theo-andreou.org 8.8.8.8**# send query to a public DNS sever instead of the local resolver
- **$ host www.ubntucy.org** # CNAME (alias) example
- **$ host -v google.com** # verbose mode
- **$ host -t SOA theo-andreou.org** # search for the authoritative DNS server of the theo-andreou.org domain
- **$ host -t NS theo-andreou.org** # look for the **theo-andreou.org** DNS servers (aka nameservers)

Query DNS servers with `dig`

- The **dig** command is a somewhat more powerfull alternative to **host**
- **`$ dig theo-andreou.org`** `# show A, CNAME and NS records`
- **`$ dig theo-andreou.org @8.8.8.8`** `# send query to the 8.8.8.8 DNS server instead of the default system resolver`
- **`$ dig www.ubuntucy.org`** `# CNAME (alias) example`
- **`$ dig mx theo-andreou.org`** `# find mail servers for theo-andreou.org`
- **`$ dig ns theo-andreou.org`** `# find DNS servers for theo-andreou.org`
- **`$ dig soa theo-andreou.org`** `# find authoritative DNS server for theo-andreou.org`
- **`$ dig -x 8.8.8.8`** `# Reverse DNS (PTR) query to find the hostname, given the IP address`

Check network availability with `ping`

- The **ping** command check the availability of network nodes using the **ICMP** protocol

- **$ ping 127.0.0.1** # check the local TCP/IP stack
```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.031 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.051 ms
^C                           # <= *** Ctrl-C to terminate ***
--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.031/0.041/0.051/0.010 ms
```
- **$ ping www.google.com** # check the response of www.google.com
```
PING www.l.google.com (173.194.69.147) 56(84) bytes of data.
64 bytes from bk-in-f147.1e100.net (173.194.69.147): icmp_req=1
ttl=47 time=100 ms
64 bytes from bk-in-f147.1e100.net (173.194.69.147): icmp_req=2
ttl=47 time=100 ms
64 bytes from bk-in-f147.1e100.net (173.194.69.147): icmp_req=3
ttl=47 time=102 ms
^C                           # <= *** Ctrl-C to terminate ***
--- www.l.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 100.609/101.233/102.330/0.820 ms
```

Check network availability with `ping`

- **$ ping -c4 2.1.1.1** # send only 4 ICMP packets to IP 2.1.1.1
  PING 2.1.1.1 (2.1.1.1) 56(84) bytes of data.
  # no responce
  --- 2.1.1.1 ping statistics ---
  4 packets transmitted, 0 received, 100% packet loss, time 2999ms
- **$ ping -c4 192.168.2.8** # send only 4 ICMP packets to ICMP σε 192.168.2.8
  PING 192.168.2.8 (192.168.2.8) 56(84) bytes of data.
  From 192.168.2.11 icmp_seq=1 Destination Host Unreachable
  From 192.168.2.11 icmp_seq=2 Destination Host Unreachable
  From 192.168.2.11 icmp_seq=3 Destination Host Unreachable
  From 192.168.2.11 icmp_seq=4 Destination Host Unreachable
  # reply from 192.168.2.11 that 192.168.2.8 is offline
  --- 192.168.2.8 ping statistics ---
  4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3013ms

Check network paths with `traceroute` and `tracepath`

- These commands are used to check the path of a route until a certain destination. The results will display the intermediate nodes and if there is a problem, we will know exactly where the problem is. **traceroute** has more options than **tracepath** but the latter is default for most distributions
- There is also **mtr** which combines the results of **traceroute/tracepath** and **ping** and continues producing results until we press Ctrl-C

Check network paths with `traceroute` and `tracepath`

```
$ traceroute malena.theo-andreou.org # find path to malena.theo-andreou.org

traceroute to malena.theo-andreou.org (37.247.48.150), 30 hops max, 60 byte packets
 1  gateway (192.168.10.1)  0.231 ms * *
 2  gw.primeoffice.thunderworx.net (78.158.142.254)  1.094 ms  1.078 ms  1.424 ms
 3  gw.ip.primehome.com (46.21.57.254)  49.094 ms  49.110 ms  49.089 ms
 4  j1.lim-2.nsp-transit.net (78.158.134.118)  49.069 ms  49.066 ms j1.lim.nsp-transit.net
   (78.158.134.250)  49.031 ms
 5  v3068.j1.fra.prime-tel.net (78.158.141.157)  97.865 ms  100.546 ms  100.570 ms
 6  213.140.39.140 (213.140.39.140)  102.387 ms  100.581 ms  102.761 ms
 7  5.53.5.253 (5.53.5.253)  115.164 ms  115.172 ms  111.848 ms
 8  5.53.4.28 (5.53.4.28)  102.615 ms  112.005 ms  113.865 ms
 9  be12956.agr41.fra03.atlas.cogentco.com (130.117.14.117)  129.002 ms  128.962 ms
   128.894 ms
10  be3187.ccr42.fra03.atlas.cogentco.com (130.117.1.118)  131.774 ms  107.528 ms  105.803
   ms
11  be2960.ccr22.muc03.atlas.cogentco.com (154.54.36.254)  112.813 ms
   be2959.ccr21.muc03.atlas.cogentco.com (154.54.36.54)  113.138 ms
   be2960.ccr22.muc03.atlas.cogentco.com (154.54.36.254)  112.830 ms
12  be3072.ccr51.zrh02.atlas.cogentco.com (130.117.0.17)  125.978 ms  125.930 ms  125.661
   ms
13  be3586.rcr21.mil01.atlas.cogentco.com (154.54.60.114)  126.047 ms  125.913 ms
   be2043.rcr21.mil01.atlas.cogentco.com (154.54.38.102)  125.998 ms
14  be3459.nr51.b019138-1.mil01.atlas.cogentco.com (154.25.12.74)  126.769 ms  128.092 ms
   127.883 ms
15  prometeus.demarc.cogentco.com (149.14.134.122)  131.175 ms  131.180 ms  133.485 ms
16  37.247.50.20 (37.247.50.20)  129.294 ms 37.247.50.54 (37.247.50.54)  130.344 ms
   130.225 ms
17  malena.theo-andreou.org (37.247.48.150)  130.919 ms  117.366 ms 37.247.50.51
   (37.247.50.51)  130.660 ms
```

Check network paths with `traceroute` and `tracepath`

```
$ tracepath malena.theo-andreou.org
 1?: [LOCALHOST]                      pmtu 1500
 1:  gateway                                               0.640ms
 1:  gateway                                               0.635ms
 2:  gw.primeoffice.thunderworx.net                        1.337ms
 3:  gw.ip.primehome.com                                  48.664ms asymm  4
 4:  j1.lim.nsp-transit.net                               49.124ms asymm  5
 5:  v3068.j1.fra.prime-tel.net                          115.959ms asymm  6
 6:  213.140.39.140                                      129.760ms
 7:  5.53.5.253                                          103.743ms
 8:  5.53.4.28                                           109.318ms
 9:  be12956.agr41.fra03.atlas.cogentco.com              141.998ms asymm 11
10:  be3186.ccr41.fra03.atlas.cogentco.com               120.809ms asymm 12
11:  be2960.ccr22.muc03.atlas.cogentco.com               127.217ms asymm 13
12:  be3073.ccr52.zrh02.atlas.cogentco.com               131.622ms asymm 14
13:  be3586.rcr21.mil01.atlas.cogentco.com               148.614ms asymm 15
14:  be3459.nr51.b019138-1.mil01.atlas.cogentco.com      141.369ms asymm 16
15:  prometeus.demarc.cogentco.com                       139.371ms asymm 17
16:  37.247.50.20                                        133.596ms asymm 12
17:  malena.theo-andreou.org                             133.715ms reached
     Resume: pmtu 1500 hops 17 back 12
```

Search for domain and IP owners with `whois`

- The **whois** command sends queries in domain registries and IP assigning authorities for finding who is responsible
- **$ whois lpi.org** # search for people or organizations responsible for the lpi.org domain
- **$ whois 8.8.8.8** # search for people or organizations responsible for the 8.8.8.8 domain
- **$ whois ellak.org.cy** # alas it does not work for .cy domains!

License