DevOps external course

Networking using Linux. Lection 3

Lecture 6.3

Module 6 **Linux Networking**

Serge Prykhodchenko

How To Task1 Video

https://www.youtube.com/watch?v=qso9Wy875ek

Agenda

- DNS
- Routing
- Q&A

DNS

Networking. iptables vs nftables, ipfw, pf

nftables is the new packet classification framework that replaces the existing **{ip,ip6,arp,eb}_tables** infrastructure. In a nutshell:

- It is available in Linux kernels >= 3.13.
- It comes with a new command line utility **nft** whose syntax is different to **iptables**.
- It also comes with a compatibility layer that allows you to run iptables commands over the new **nftables** kernel framework.
- It provides generic set infrastructure that allows you to construct maps and concatenation. You can use this new feature to arrange your ruleset in multidimensional tree which **drastically** reduces the number of rules that need to be inspected until you find the final action on the packet

https://wiki.nftables.org/wiki-nftables/index.php/Main_Page

Networking. iptables vs nftables, ipfw, pf

Main differences with iptables

• The main differences between *nftables* and *iptables* from the user point of view are:

• The **syntax**. The *iptables* command line tool uses a getopt_long()-based parser where keys are always preceded by double minus, eg. *--key* or one single minus, eg. *-p tcp*. In that regard, *nftables* uses nicer, more intuitive and more compact syntax which is inspired by *tcpdump*.

• **Tables and chains are fully configurable**. In *nftables*, tables are container of chains with no specific semantics. Note that *iptables* comes with tables with a predefined number of base chains, you get them in an all or nothing fashion. Thus, all chains are registered even if you only need one of them. We got reports in the past that unused base chains are harming performance, even if you add no rules at all. With this new approach, you can just register the chains that you need depending on your setup. Moreover, you can also model your pipeline using the chain priorities in the way you need and select any name for your tables and chains.

• . In *nftables*, the *expressions* are the basic building block of rule, thus, a rule is basically a composite of expressions that is linearly evaluated from left to right: if the first expression matches, then the next expression is evaluated and so on until we reach the last expression that is part of the rule. An expression can match some specific payload field, packet/flow metadata and any action.

• **You can specify several actions in one single rule**. In *iptables* you can only specify one single target. This has been a longstanding limitation that users resolve by jumping to custom chains at the cost of making the rule-set structure slightly more complex.

• **No built-in counter per chain and rules**. In *nftables*, these are optional so you can enable counters on demand.

Networking. iptables vs nftables, ipfw, pf

Main differences with iptables (2)

- **Better support for dynamic ruleset updates**. In nftables, if you add a new rule, the remaining existing ones are left untouched since the ruleset is represented in a linked-list contrary to the monolithic blob representation in which the maintainance of the internal state information is complicated when performing ruleset updates.
- **Simplified dual stack IPv4/IPv6 administration**, through the new inet family which allows you to register base chains that see both IPv4 and IPv6 traffic. Thus, you don't need to rely on scripts to duplicate your ruleset anymore.
- **Generic set and map infrastructure**. This new infrastructure integrates tightly into the nftables core and it allows advanced configurations such as dictionaries, maps and intervals to achieve performance-oriented packet classification. The most important thing is that you can use any supported selector to classify traffic.
- **Support for concatenations**. Since Linux kernel 4.1, you can concatenate several keys and combine them with dictionaries and maps. The idea is to build a tuple whose values are hashed to obtain the action to be performed nearly O(1) (approximately constant).
- **New supported protocols** without kernel upgrades. Kernel upgrades can be a timeconsuming and daunting task. Specifically if you have to maintain more than one single firewall in your network. Distributors usually include a bit older Linux kernel versions for stability reasons. With the new nftables virtual machine approach, you will most likely not need such upgrade to support a new protocol. A relatively simple nft userspace software update should be enough to support new protocols

Networking. iptables vs nftables, ipfw, pf

nftables.conf EXAMPLE

```
#!/usr/sbin/nft -f

flush ruleset

# List all IPs and IP ranges of your traffic filtering proxy source.
define SAFE_TRAFFIC_IPS = {
    x.x.x.x/xx,
    x.x.x.x/xx,
    x.x.x.x,
    x.x.x.x
}

table inet firewall {

    chain inbound {

        # By default, drop all traffic unless it meets a filter
        # criteria specified by the rules that follow below.
        type filter hook input priority 0; policy drop;

        # Allow traffic from established and related packets.
        ct state established,related accept

        # Drop invalid packets.
        ct state invalid drop

        # Allow loopback traffic.
        iifname lo accept
```

```
        # Allow all ICMP and IGMP traffic, but enforce a rate limit
        # to help prevent some types of flood attacks.
        ip protocol icmp limit rate 4/second accept
        ip6 nexthdr ipv6-icmp limit rate 4/second accept
        ip protocol igmp limit rate 4/second accept

        # Allow SSH on port 22.
        tcp dport 22 accept

        # Allow HTTP(S).
        # -- From anywhere
        tcp dport { http, https } accept
        udp dport { http, https } accept
        # -- From approved IP ranges only
        # tcp dport { http, https } ip saddr $SAFE_TRAFFIC_IPS accept
        # udp dport { http, https } ip saddr $SAFE_TRAFFIC_IPS accept

        # Uncomment to allow incoming traffic on other ports.
        # -- Allow Jekyll dev traffic on port 4000.
        # tcp dport 4000 accept
        # -- Allow Hugo dev traffic on port 1313.
        # tcp dport 1313 accept

        # Uncomment to enable logging of denied inbound traffic
        # log prefix "[nftables] Inbound Denied: " flags all counter drop

    }
```

Networking. iptables vs nftables, ipfw, pf

nftables.conf EXAMPLE

```
    chain forward {

        # Drop everything (assumes this device is not a router)
        type filter hook forward priority 0; policy drop;

        # Uncomment to enable logging of denied forwards
        # log prefix "[nftables] Forward Denied: " flags all counter drop

    }

    chain outbound {

        # Allow all outbound traffic
        type filter hook output priority 0; policy accept;

    }

}
```

DNS

You can set up four different types of DNS servers:

- A master DNS server for your domain(s), which stores authoritative records for your domain.

- A slave DNS server, which relies on a master DNS server for data.

- A caching-only DNS server, which stores recent requests like a proxy server. It otherwise refers to other DNS servers.

- A forwarding-only DNS server, which refers all requests to other DNS servers.

DNS

BIND which stands for “*Berkely Internet Name Domain*” is a free and Opensource software which is widely used in Linux servers for translating Domain names to IP address. BIND performs both of the main DNS server roles – acting as an authoritative name server for one or more specific domains, and acting as a recursive resolver for the DNS system generally. The current version of BIND is BIND 9.

Dnsmasq is a lightweight DNS, TFTP, PXE, router advertisement and DHCP server. It is intended to provide coupled DNS and DHCP service to a LAN. Dnsmasq accepts DNS queries and either answers them from a small, local, cache or forwards them to a real, recursive, DNS server. It loads the contents of /etc/hosts so that local hostnames which do not appear in the global DNS can be resolved and also answers DNS queries for DHCP configured hosts.

PowerDNS, founded in the late 1990s, is a premier supplier of open source DNS software, services, and support. According to PowerDNS, there are two PowerDNS nameserver products: the Authoritative Server and the Recursor. While most other nameservers fully combine these functions, PowerDNS offers them separately but can mix both authoritative and recursive usage seamlessly. What this means is that if you download different packages depending on your need. If you would wish to have an authoritative DNS, then get the authoritative package and the same goes for the recursive counterpart.

Unbound is a free, open source validating, recursive, caching DNS resolver software under the BSD license. It is a recently developed DNS System that came into the DNS space to bring a fast and lean system that incorporates modern features based on open standards.

Dnsmasq (implementation.step1 [preparation])

4. 192.168.0.103 (student)

```
            · MobaXterm 12.2 ·
    (SSH client, X-server and networking tools)

  ➤ SSH session to student@192.168.0.103
    · SSH compression : ✔
    · SSH-browser     : ✔
    · X11-forwarding  : ✔  (remote display is forwarded through SSH)
    · DISPLAY         : ✔  (automatically set on remote server)

  ➤ For more info, ctrl+click on help or visit our website

Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-131-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

196 packages can be updated.
143 updates are security updates.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Thu Aug 27 09:15:55 2020
student@ubuntu16srvr:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group defaul
    link/ether 08:00:27:ac:1b:56 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feac:1b56/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group defaul
    link/ether 08:00:27:4c:53:00 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.1/24 brd 10.10.10.255 scope global enp0s8
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe4c:5300/64 scope link
       valid_lft forever preferred_lft forever
student@ubuntu16srvr:~$
```

port MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

4. 192.168.0.103 (student)

```
student@ubuntu16srvr:~$ sudo apt update
[sudo] password for student:
Hit:1 http://ua.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://ua.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://ua.archive.ubuntu.com/ubuntu xenial-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
190 packages can be upgraded. Run 'apt list --upgradable' to see them.
student@ubuntu16srvr:~$ sudo apt install dnsmasq
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  dnsmasq
0 upgraded, 1 newly installed, 0 to remove and 190 not upgraded.
Need to get 16.0 kB of archives.
After this operation, 71.7 kB of additional disk space will be used.
Get:1 http://ua.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 dnsmasq all 2.75-1ubuntu0.16.04.5 [16.0 kB]
Fetched 16.0 kB in 0s (353 kB/s)
Selecting previously unselected package dnsmasq.
(Reading database ... 62182 files and directories currently installed.)
Preparing to unpack .../dnsmasq_2.75-1ubuntu0.16.04.5_all.deb ...
Unpacking dnsmasq (2.75-1ubuntu0.16.04.5) ...
Processing triggers for systemd (229-4ubuntu21.4) ...
Processing triggers for ureadahead (0.100.0-19) ...
Setting up dnsmasq (2.75-1ubuntu0.16.04.5) ...
Processing triggers for systemd (229-4ubuntu21.4) ...
Processing triggers for ureadahead (0.100.0-19) ...
student@ubuntu16srvr:~$
```

Dnsmasq (implementation.step2 [configuring DHCP])

```
4. 192.168.0.103 (student)
 GNU nano 2.5.3                     File: /etc/dnsmasq.conf

#address=/www.thekelleys.org.uk/fe80::20d:60ff:fe36:f83

# Add the IPs of all queries to yahoo.com, google.com, and their
# subdomains to the vpn and search ipsets:
#ipset=/yahoo.com/google.com/vpn,search

# You can control how dnsmasq talks to a server: this forces
# queries to 10.1.2.3 to be routed via eth1
# server=10.1.2.3@eth1

# and this sets the source (ie local) address used to talk to
# 10.1.2.3 to 192.168.1.1 port 55 (there must be a interface with that
# IP on the machine, obviously).
# server=10.1.2.3@192.168.1.1#55

# If you want dnsmasq to change uid and gid to something other
# than the default, edit the following lines.
#user=
#group=

# If you want dnsmasq to listen for DHCP and DNS requests only on
# specified interfaces (and the loopback) give the name of the
# interface (eg eth0) here.
# Repeat the line for more than one interface.
interface=enp0s8
# Or you can specify which interface _not_ to listen on
#except-interface=
# Or which to listen on by address (remember to include 127.0.0.1 if
# you use this.)
#listen-address=
# If you want dnsmasq to provide only DNS service on an interface,
# configure it as shown above, and then use the following line to
# disable DHCP and TFTP on it.
#no-dhcp-interface=

# On systems which support it, dnsmasq binds the wildcard address,
# even when it is listening on only some interfaces. It then discards
# requests that it shouldn't reply to. This has the advantage of
# working even when interfaces come and go and change address. If you
# want dnsmasq to really bind only the interfaces it is listening on,
# uncomment this option. About the only time you may need this is when
# running another nameserver on the same machine.
#bind-interfaces

# If you don't want dnsmasq to read /etc/hosts, uncomment the
# following line.
#no-hosts
# or if you want it to read another file, as well as /etc/hosts, use
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text    ^J Justify    ^C Cur Pos     ^Y Prev Pa
^X Exit       ^R Read File   ^\ Replace    ^U Uncut Text  ^T To Spell   ^_ Go To Line  ^V Next Pa
```

```
4. 192.168.0.103 (student)
 GNU nano 2.5.3                     File: /etc/dnsmasq.conf                          Modified

# If you don't want dnsmasq to read /etc/hosts, uncomment the
# following line.
#no-hosts
# or if you want it to read another file, as well as /etc/hosts, use
# this.
#addn-hosts=/etc/banner_add_hosts

# Set this (and domain: see below) if you want to have a domain
# automatically added to simple names in a hosts-file.
#expand-hosts

# Set the domain for dnsmasq. this is optional, but if it is set, it
# does the following things.
# 1) Allows DHCP hosts to have fully qualified domain names, as long
#     as the domain part matches this setting.
# 2) Sets the "domain" DHCP option thereby potentially setting the
#    domain of all systems configured by DHCP
# 3) Provides the domain part for "expand-hosts"
#domain=thekelleys.org.uk

# Set a different domain for a particular subnet
#domain=wireless.thekelleys.org.uk,192.168.2.0/24

# Same idea, but range rather then subnet
#domain=reserved.thekelleys.org.uk,192.68.3.100,192.168.3.200

# Uncomment this to enable the integrated DHCP server, you need
# to supply the range of addresses available for lease and optionally
# a lease time. If you have more than one network, you will need to
# repeat this for each network on which you want to supply DHCP
# service.
dhcp-range=10.10.10.10,10.10.10.20,12h

# This is an example of a DHCP range where the netmask is given. This
# is needed for networks we reach the dnsmasq DHCP server via a relay
# agent. If you don't know what a DHCP relay agent is, you probably
# don't need to worry about this.
#dhcp-range=192.168.0.50,192.168.0.150,255.255.255.0,12h

# This is an example of a DHCP range which sets a tag, so that
# some DHCP options may be set only for this network.
#dhcp-range=set:red,192.168.0.50,192.168.0.150

# Use this DHCP range only when the tag "green" is set.
#dhcp-range=tag:green,192.168.0.50,192.168.0.150,12h

# Specify a subnet which can't be used for dynamic address allocation,
# is available for hosts with matching --dhcp-host lines. Note that
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text    ^J Justify    ^C Cur Pos     ^Y Prev Page   M-\ First Line
^X Exit       ^R Read File   ^\ Replace    ^U Uncut Text  ^T To Spell   ^_ Go To Line  ^V Next Page   M-/ Last Line
```

Dnsmasq (implementation.step3 [enabling DNS])

5. 192.168.0.103 (student)

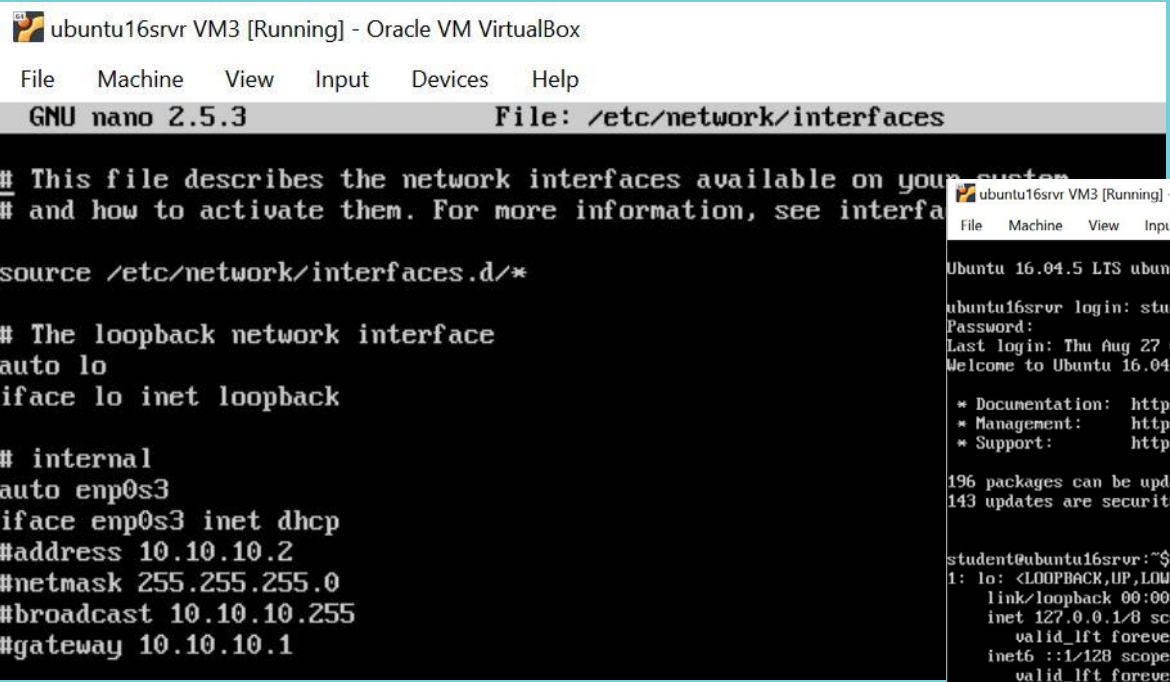
```
GNU nano 2.5.3                    File: /etc/resolv.conf

# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.0.1
```

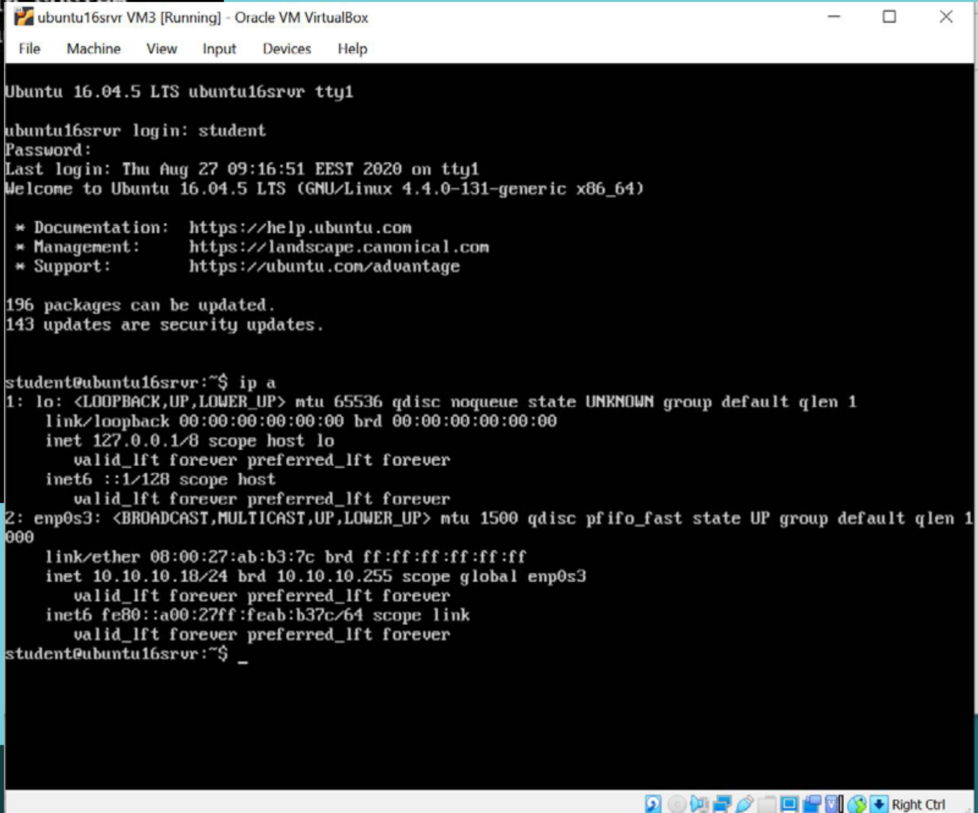
uick connect...

5. 192.168.0.103 (student)

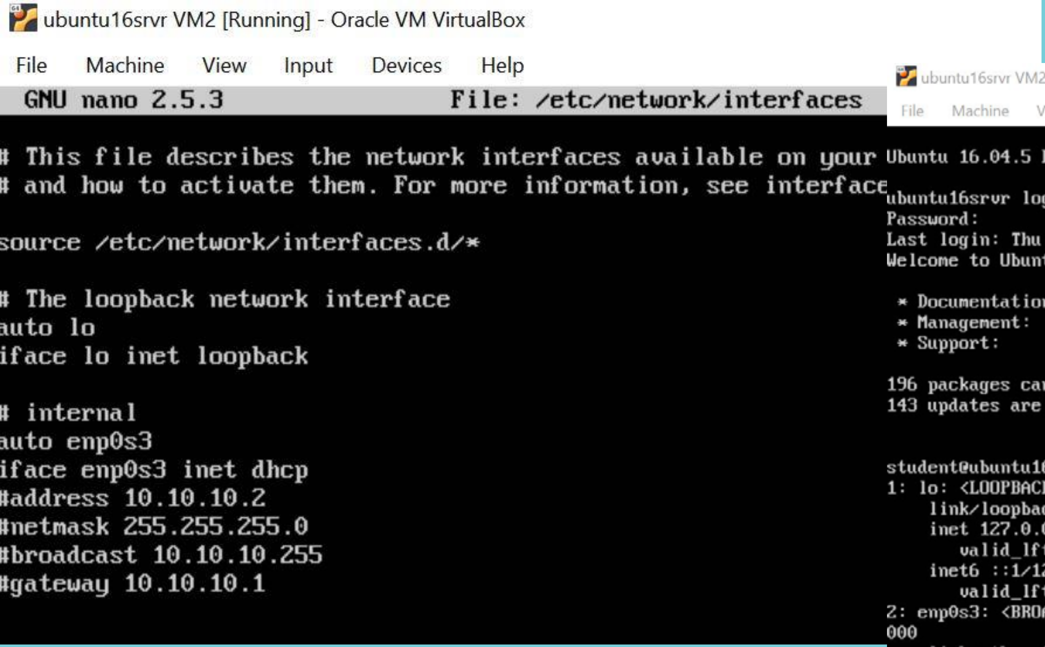
```
GNU nano 2.5.3              File: /etc/dhcp/dhclient.conf                  Modified

# Configuration file for /sbin/dhclient.
#
# This is a sample configuration file for dhclient. See dhclient.conf's
#       man page for more information about the syntax of this file
#       and a more comprehensive list of the parameters understood by
#       dhclient.
#
# Normally, if the DHCP server provides reasonable information and does
#       not leave anything out (like the domain name, for example), then
#       few changes must be made to this file, if any.
#

option rfc3442-classless-static-routes code 121 = array of unsigned integer 8;

send host-name = gethostname();
request subnet-mask, broadcast-address, time-offset, routers,
        domain-name, domain-name-servers, domain-search, host-name,
        dhcp6.name-servers, dhcp6.domain-search, dhcp6.fqdn, dhcp6.sntp-servers,
        netbios-name-servers, netbios-scope, interface-mtu,
        rfc3442-classless-static-routes, ntp-servers;

#send dhcp-client-identifier 1:0:a0:24:ab:fb:9c;
#send dhcp-lease-time 3600;
#supersede domain-name "fugue.com home.vix.com";
prepend domain-name-servers 127.0.0.1;
#require subnet-mask, domain-name-servers;

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

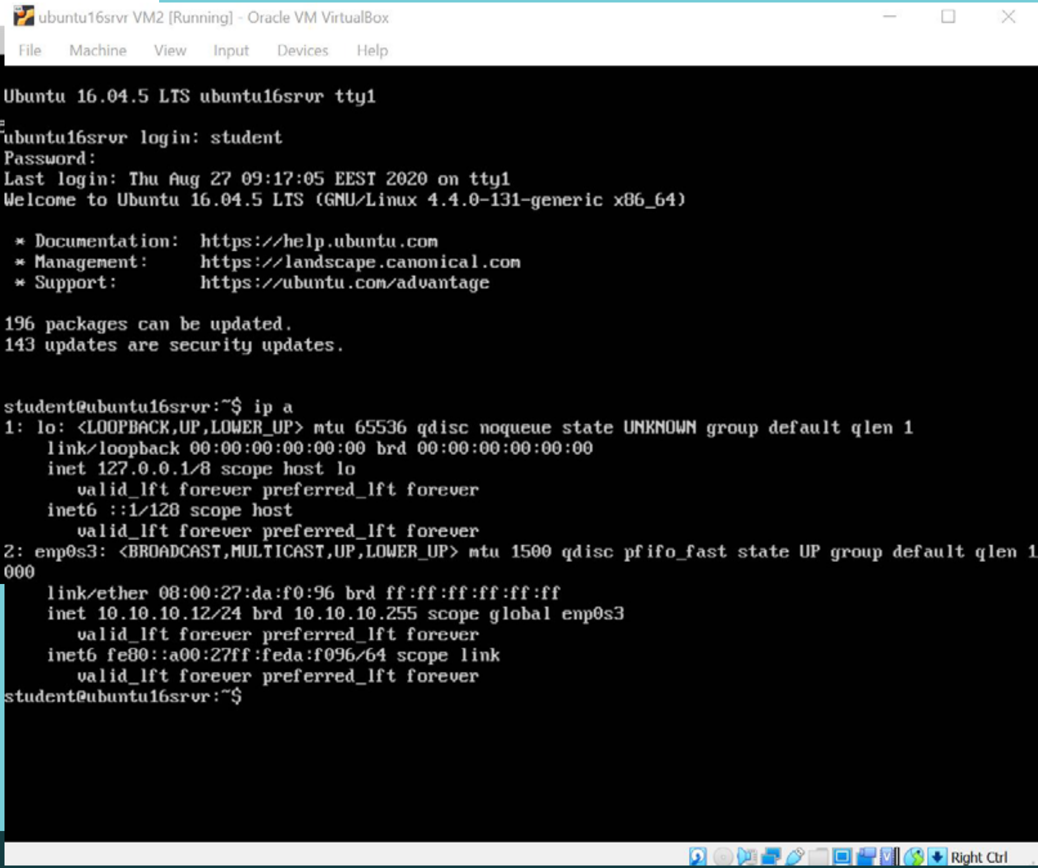
Dnsmasq (implementation.step4a [checking client1])





Dnsmasq (implementation.step4b [checking client2])

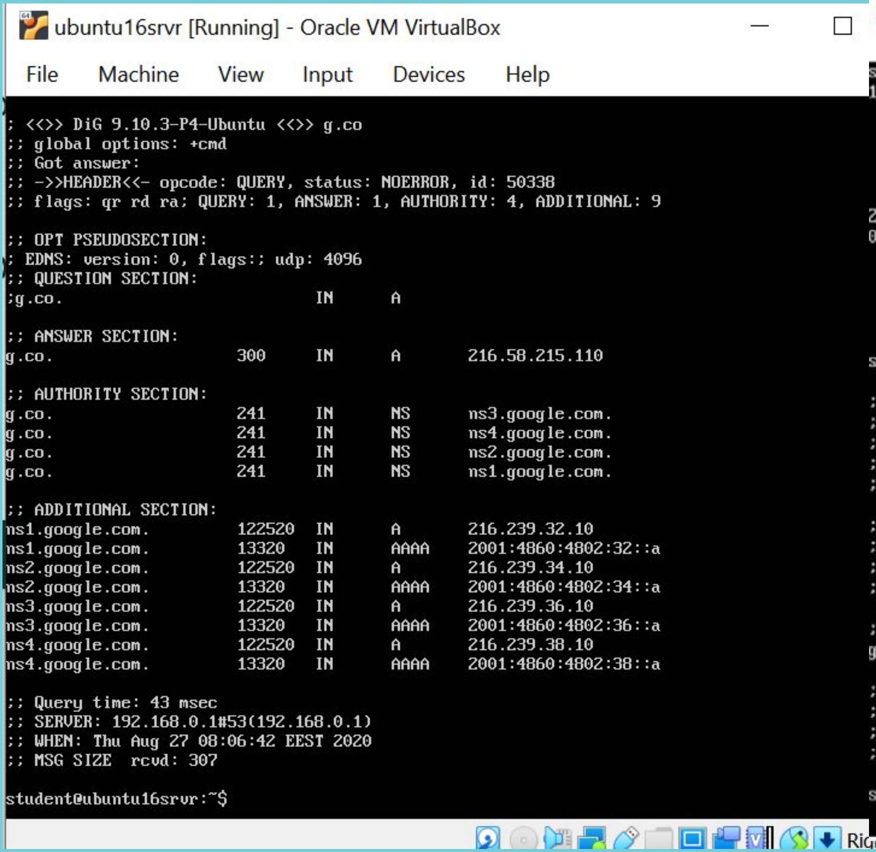


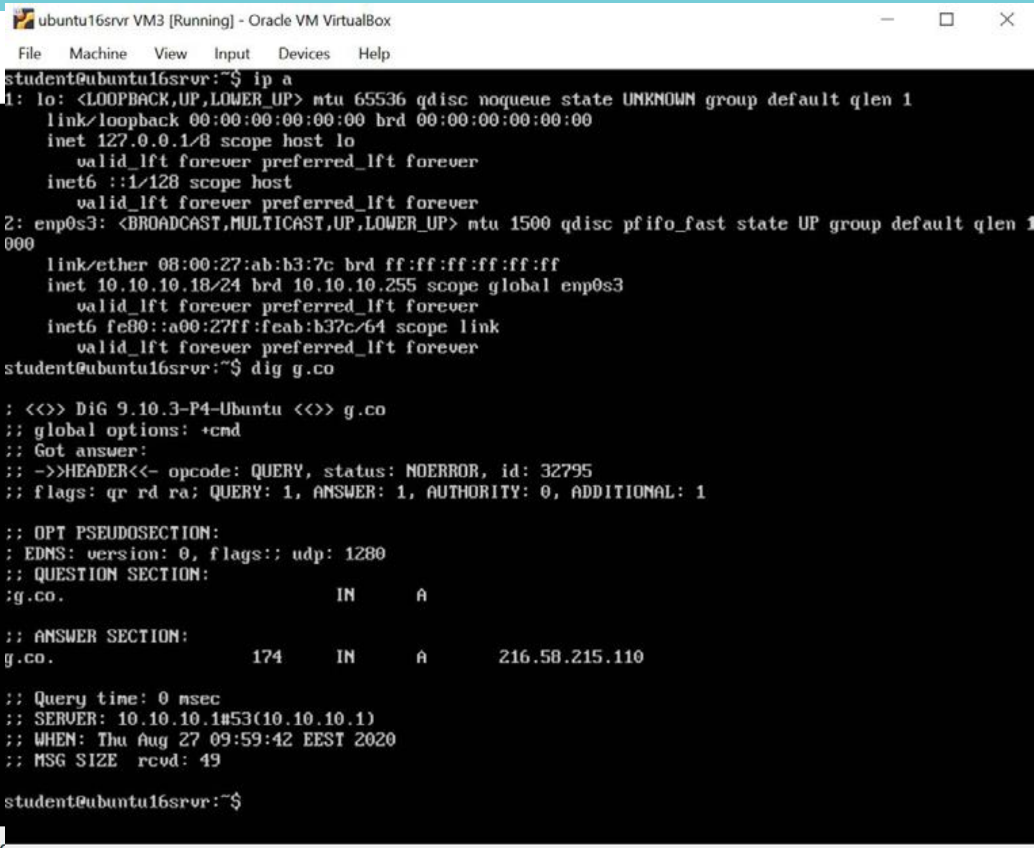


Dnsmasq (implementation.step5[keeping in mind forwarding])

```
4. 192.168.0.103 (student)

                 · MobaXterm 12.2 ·
       (SSH client, X-server and networking tools)

 ➤ SSH session to student@192.168.0.103
   · SSH compression : ✔
   · SSH-browser     : ✔
   · X11-forwarding  : ✔  (remote display is forwarded through SSH)
   · DISPLAY         : ✔  (automatically set on remote server)

 ➤ For more info, ctrl+click on help or visit our website

Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-131-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

196 packages can be updated.
143 updates are security updates.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Aug 25 15:18:12 2020
student@ubuntu16srvr:~$ sudo iptables -S
[sudo] password for student:
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
student@ubuntu16srvr:~$ sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
student@ubuntu16srvr:~$ sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -m state --state RELATED,ESTABLISHED -j ACCEPT
student@ubuntu16srvr:~$ sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT
student@ubuntu16srvr:~$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A FORWARD -i enp0s8 -o enp0s3 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT
student@ubuntu16srvr:~$
```

Dnsmasq (implementation.step6[Checking results])





ROUTING

Routing (static)

Static routes are for traffic that must not, or should not, go through the default gateway. Routing is often handled by devices on the network dedicated to routing (although any device can be configured to perform routing). Therefore, it is often not necessary to configure static routes Linux servers or clients. Exceptions include traffic that must pass through an encrypted VPN tunnel or traffic that should take a specific route for reasons of cost or security. The default gateway is for any and all traffic which is not destined for the local network and for which no preferred route is specified in the routing table. The default gateway is traditionally a dedicated network router.

Configuring Static Routes Using the Command Line

If static routes are required, they can be added to the routing table by means of the ip route add command and removed using the ip route del command. The more frequently used ip route commands take the following form:

ip route [add | del | change | append | replace] destination-address

Routing (dynamic)

Features of routing *protocols*

Routers use routing protocols:

- To know all the available paths of the network
- To select the best and fastest path for each destination in the network
- To select a single and fastest path if more than one path exists for a single destination

Functions of routing protocols:

The main functions of routing protocols are the following:

- Learn routing information from neighboring routers
- Advertise local routing information to neighboring routers
- Calculate the best route for each subnet of the network
- Provide a virtual map of all routes of the network
- Calculate the cost of each route and help the router choose the best and fastest route
- Detect any change in the network and update all routers about that change

Routing

Types of routing protocols

There are three types of routing protocols:

- distance-vector,
- link-state,
- hybrid.

RIPv1 and IGRP are examples of distance-vector routing protocols while OSPF is an example of a link-state routing protocol. Examples of hybrid routing protocols include RIPv2, EIGRP, and BGP

Routing (distance-vector)

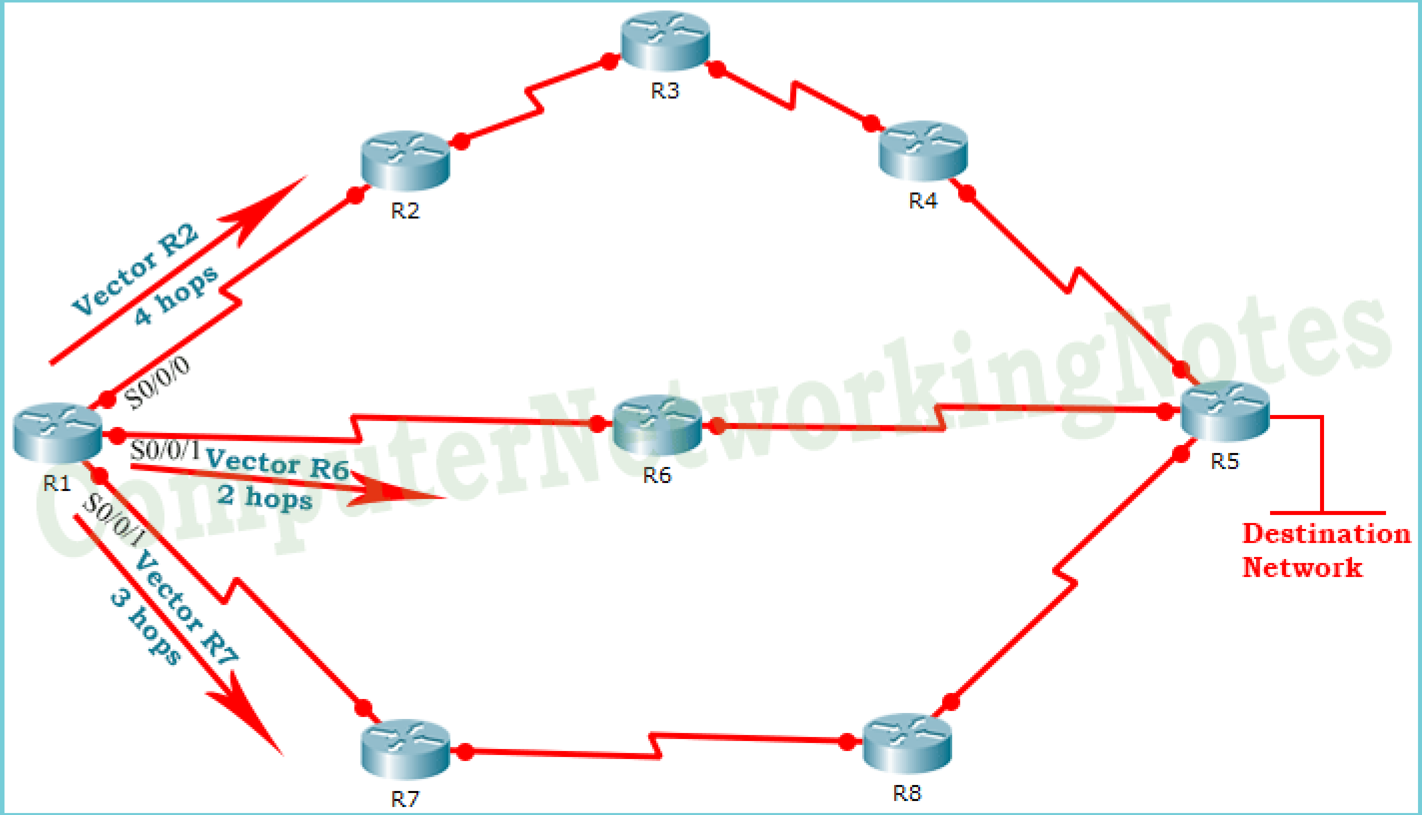
Distance-vector routing protocols

- Routers running distance-vector routing protocols periodically broadcast routing and reachability information from all active interfaces. They also receive the same information from their neighbors on their active interfaces.
- Distance-vector protocols use timers to broadcast routing information. Once their periodic timer expires, they broadcast their routing information from all active interfaces, no matter whether the routing information has changed since the previous broadcast or not.

Calculating/selecting the best route

- Distance-vector protocols use distance and direction to calculate and select the best route for each subnet of the network. Distance is the number of routers that a packet crosses to reach its destination.
- Distance is measured in terms of hops. Each instance where a packet goes through a router is called a hop. For example, if a packet crosses four routers to reach its destination, the number of hops is 4. The route with the least number of hops is selected as the best route.
- The vector indicates the direction that a packet uses to reach its destination

Routing (distance-vector)



https://www.computernetworkingnotes.com/ccna-study-guide/basic-routing-concepts-and-protocols-explained.html

Routing (distance-vector)

In this network, the router R1 has three routes to the destination network. These routes are the following.

The four-hop route (distance) through R2 (vector)

The one-hop route (distance) through R6 (vector)

The two-hop route (distance) through R7 (vector)

Since the second route has the lowest hop count, the router R1 uses this route to forward all packets of the destination network.

Key points:

Distance-vector protocols do not perform any mechanism to know who their neighbors are.

Distance-vector protocols learn about their neighbors by receiving their broadcasts.

Distance-vector protocols do not perform any formal handshake or hello process with neighbors before broadcasting routing information.

Distance-vector protocols do not verify whether neighbors received routing updates or not.

Distance-vector protocols assume that if a neighbor misses an update, it will learn about the change in the next broadcast update

Routing (Link-state)

- Unlike distance-vector routing protocols, the link-state routing protocols do not share routing and reachability information with anyone. Routers running link-state protocols share routing information only with neighbors. To discover neighbors, link-state protocols use a special protocol known as the hello protocol.
- After discovering all neighbors, the link-state protocols create three separate tables. One of these tables keeps track of directly attached neighbors, one determines the topology of the entire internetwork, and one is used as the routing table.
- From all available routes, to select the best route for each destination of the network, the link-state protocols use an algorithm called the Shortest Path First (SPF) algorithm.

Differences between distance-vector routing protocols and link-state routing protocols

- Unlike distance-vector routing protocols that broadcast the entire routing table periodically whether there are any changes or not, link-state routing protocols do not exchange routing information periodically. They exchange information only when they detect any change in the network.
- Distance-vector protocols use local broadcasts, which are processed by every router on the same segment, while linkstate protocols use multicasts which are processed only by the routers running the link-state protocol.
- Distance-vector protocols do not verify routing broadcasts. They don't care whether the neighboring routers received them or not. Link-state protocols verify routing updates. A destination router, when receiving a routing update, will respond to the source router with an acknowledgment

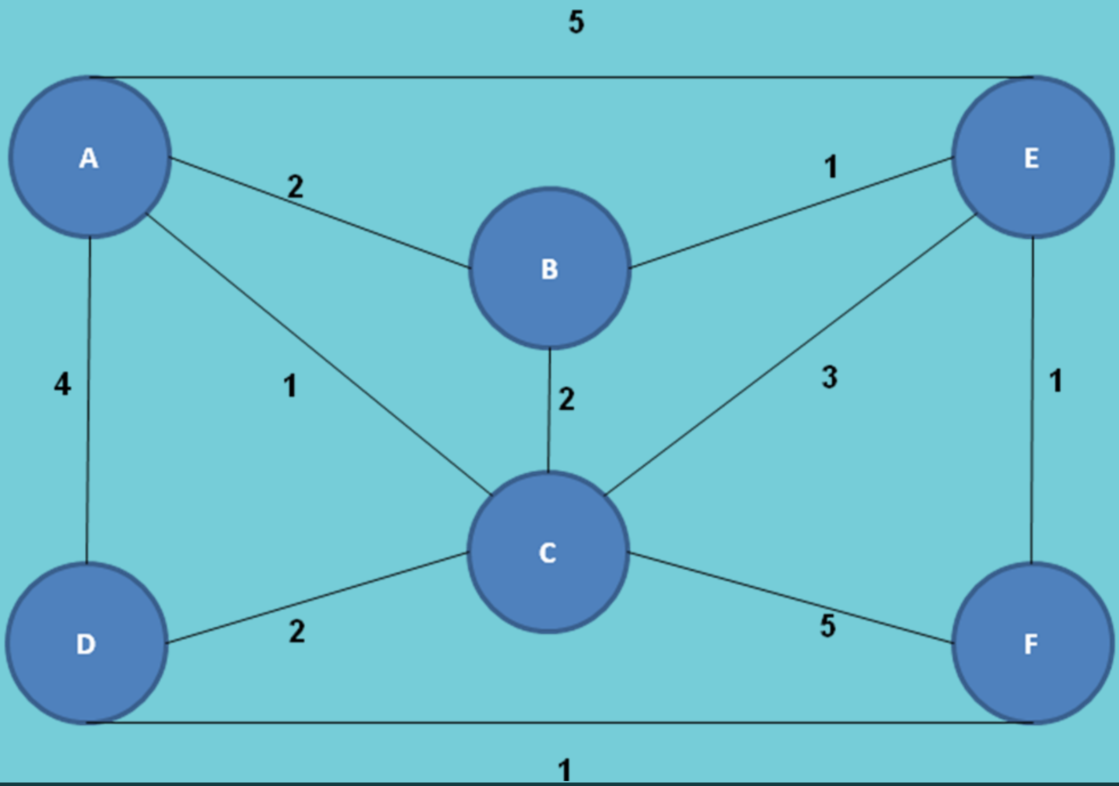
Routing (Hybrid)

- Hybrid routing protocols are the combination of both distance-vector and link-state protocols. Hybrid routing protocols are based on distance-vector routing protocols but contain many of the features and functions of link-state routing protocols.

- Hybrid routing protocols are built upon the basic principles of a distance-vector protocol but act like a link-state routing protocol. Hybrid protocols use a Hello protocol to discover neighbors and form neighbor relationships. Hybrid protocols also send updates only when a change occurs.

- Hybrid routing protocols reduce the CPU and memory overhead by functioning like a distance-vector protocol when it comes to processing routing updates; but instead of sending out periodic updates like a distance-vector protocol, hybrid routing protocols send out incremental, reliable updates via multicast messages, providing a more network- and router-friendly environment.

Routing Algorithms



Routing Algorithms

Iteration Count	New node to which least-cost route known	B Cost/ route	C Cost/ route	D Cost/ route	E Cost/ route	F Cost/ route
Init	A	2/AB	1/AC	4/AD	5/AE	∝
1	AC	2AB	**1/AC√**	**3/ACD**	**4/ACE**	**6/ACF**
2	ACB	**2/AB√**	√	3/ACD	**3/ABE**	6/ACF
3	ACBD	√	√	**3/ACD√**	3/ABE	**5/ADF**
4	ACBDE	√	√	√	**3ABE√**	**4/ABEF**
5	ACBDEF	√	√	√	√	**4/ABEF√**

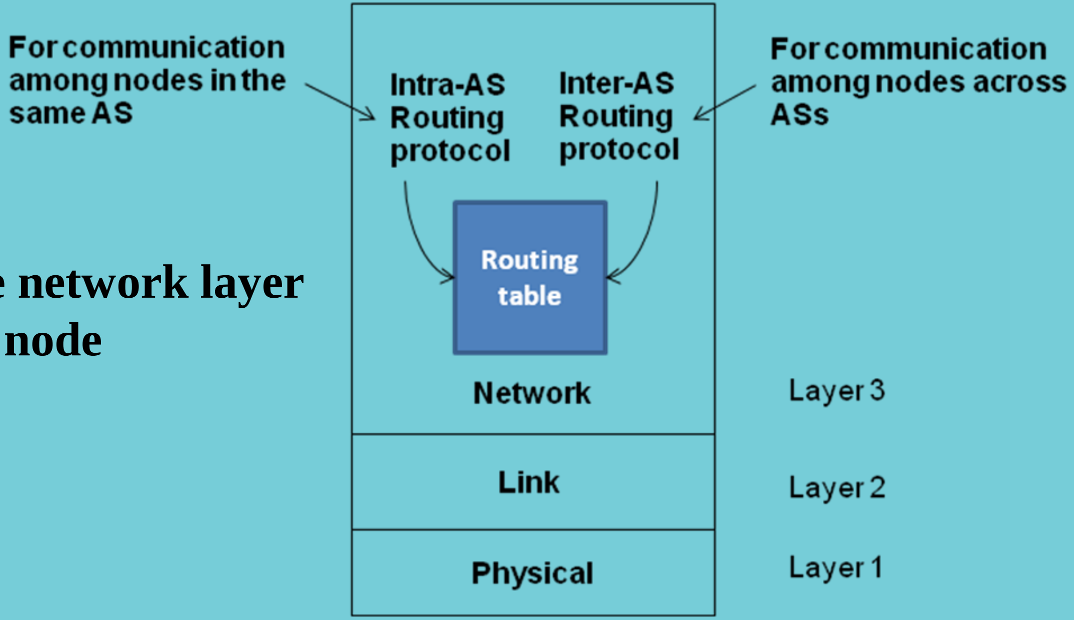
Routing Algorithms

Destination	A	B	C	F
A	5(EA)	3(BA)	4(ECA)	5(EFDCA)
B	7(EAB)	1(EB)	5(ECB)	6(EFDCB
C	6(EAC)	3(EBC)	3(EC)	4(EFDC)
D	8(EACD)	4(EBEFD)	5(ECD)	2(EFD)
F	9(EABEF)	2(EBEF)	7(ECBEF)	1(EF)

DV Table for Node E

Routing on the Internet

- Network of networks
- Scale, dynamism
- Autonomous Systems (AS)
  - Allows for evolution
  - Gateway node for inter-AS routing

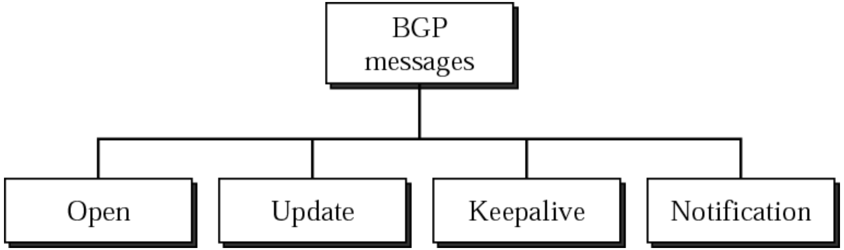


Details of the network layer in a gateway node

Border Gateway Protocol (RFC 1771)

- Based on the path vector routing.
- Distance-vector protocol not preferred for inter-AS routing (exterior routing protocol)
  - Assumes all routers have a common distance metrics to judge route preferences.
    - If routers have different meanings of a metric, it may not be possible to create stable, loop free routes.
  - A given AS may have different priorities from another AS.
  - Gives no information about the ASs that will be visited.
- Link-state routing protocol
  - Different metrics.
  - Flooding is not realistic.
- Path vector routing
  - No metrics,
  - Information about which networks can be reached by a given router and ASs to be crossed.
- Differs from DVA
  - Path vector approach does not include a distance or cost estimate
  - Lists all of the ASs visited to reach destination network.

BGP (continued)



- Messages are sent over TCP connections on port 179.
- Functional procedures
  - Neighbor acquisition (open message, acceptance through Keepalive message)
  - Neighbor reachability (periodic Keepalive messages)
  - Network reachability (broadcast an update message)
    - Each routers maintains a database of networks that can be reached
    - + preferred route to this network.
- RFC does not address
    - How a router knows the address of another router.
    - Up to network admin.

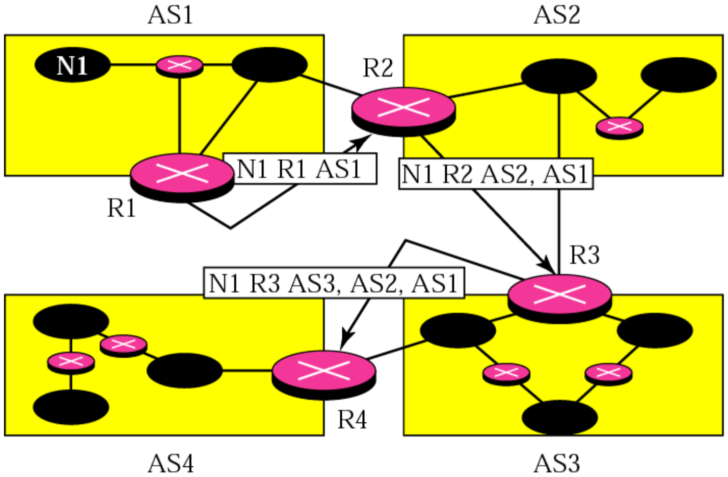
BGP (cont.)

Example of Network Reachability

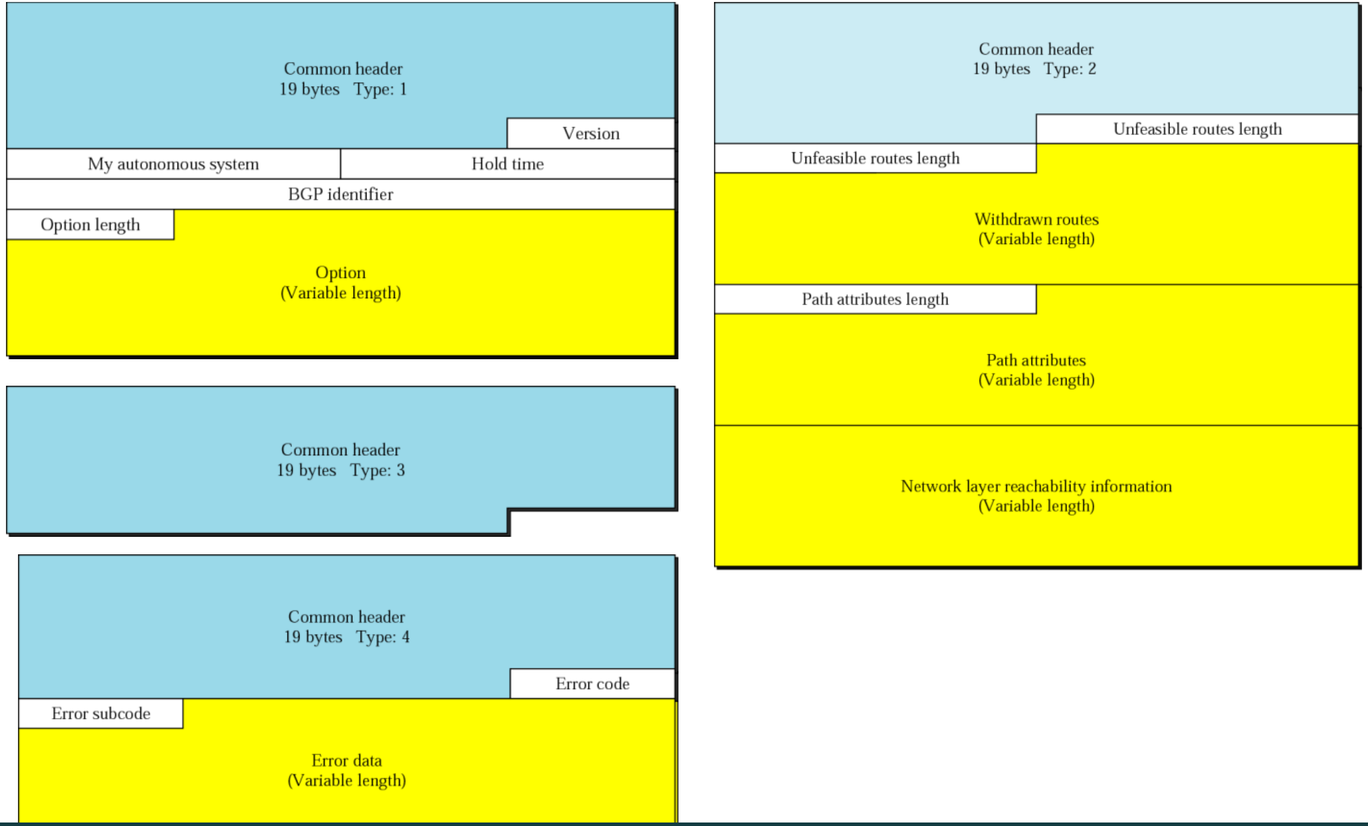
Network	Next router	Path
N1	R1	AS14,AS23,AS67
N2	R5	AS22,AS67,AS5,AS89
N3	R6	AS67,AS89,AS9,AS34
N4	R12	AS62,AS2,AS9

- Loop Prevention in BGP:
  - Checks the Path before updating its database. (If its AS is in the path ignore the message)
- Policy Routing:
  - If a path consist of an AS against the policy of the current AS, message discarded.

Example of Message adverstisements

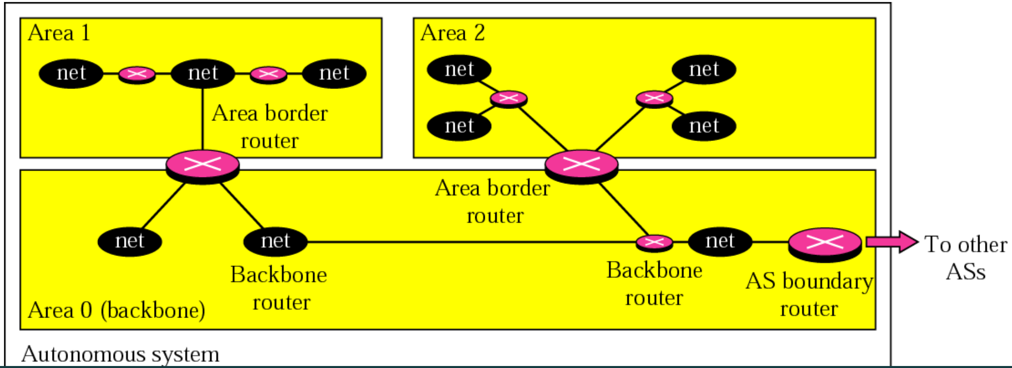


BGP message format (Open, Keepalive, Update, Notification

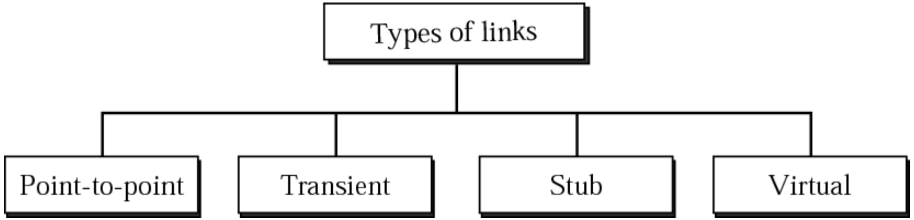


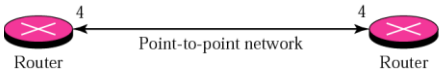
Open Shortest Path First (RFC 1247)

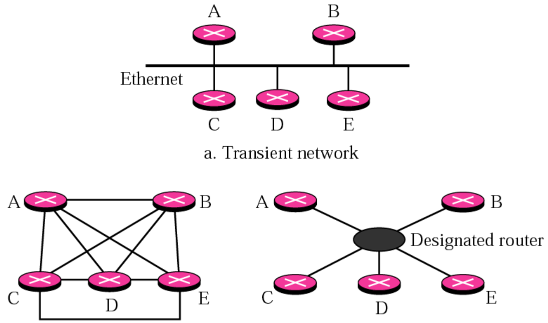
- Uses IP, has a value in the IP Header (8 bit protocol field)
- Interior routing protocol, its domain is also an autonomous system
- Special routers (autonomous system boundary routers) or backbone routers responsible to dissipate information about other AS into the current system.
- Divides an AS into areas
- Metric based on type of service
  - Minimum delay (rtt), maximum throughput, reliability, etc..



OSPF (type of links)







a. Transient network

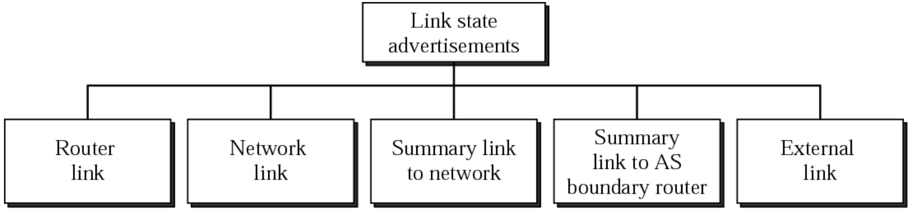
b. Unrealistic representation

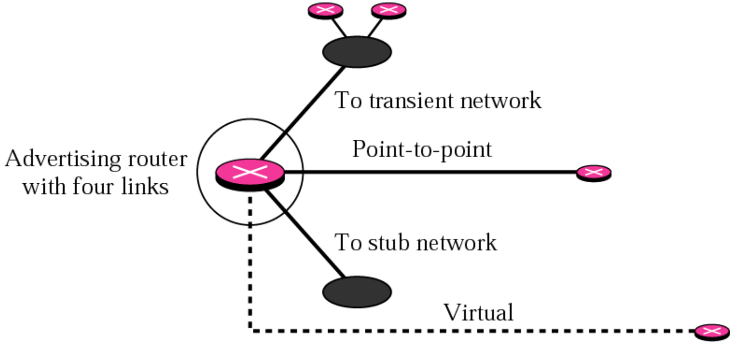
c. Realistic representation

a. Stub network

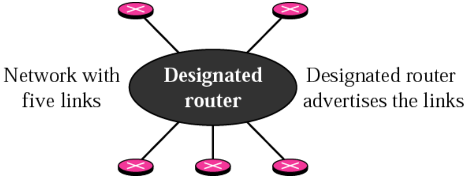
b. Representation

OSPF (link state advertisement)



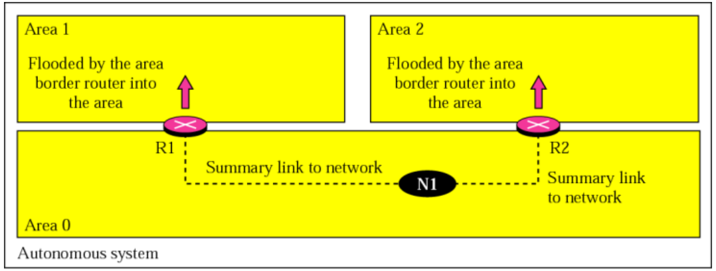


Router Link



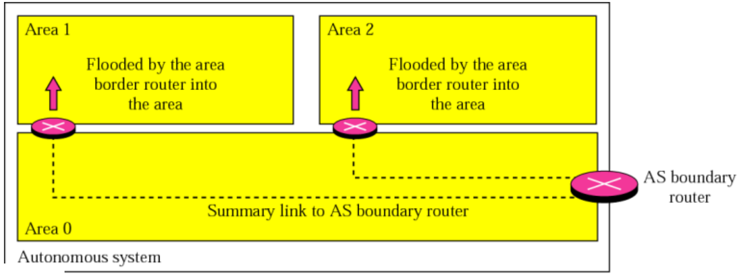
Network Link

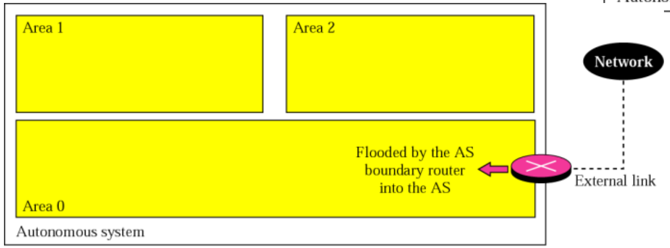
OSPF (LSA cont.)



Summary link to Network

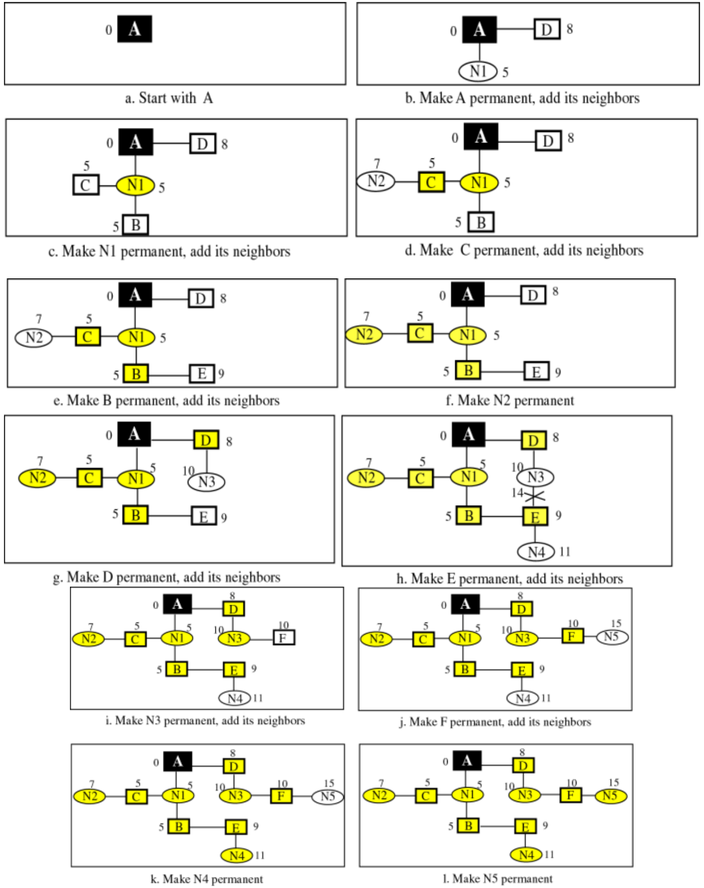
Summary link to AS boundary router



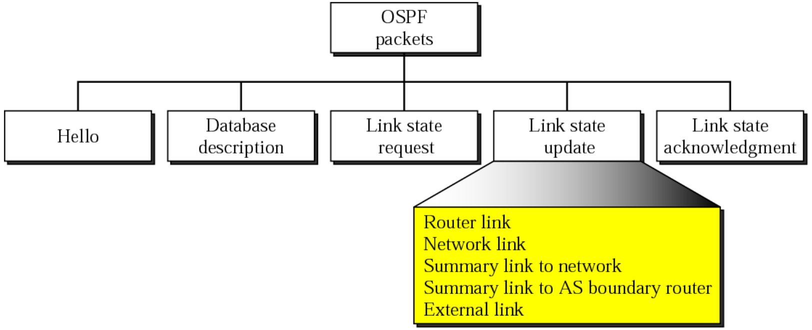


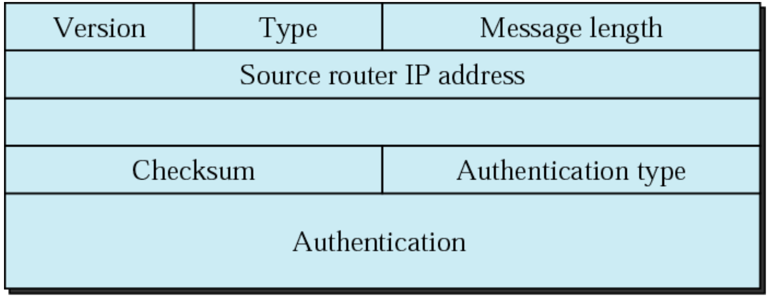
External Link

Shortest Path Calculation



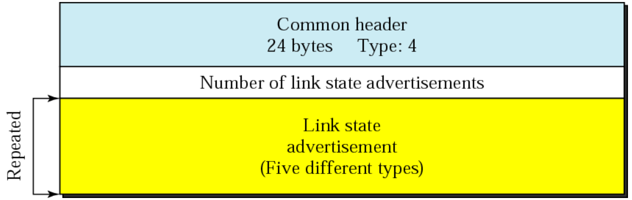
Types of OSPF packets and header format

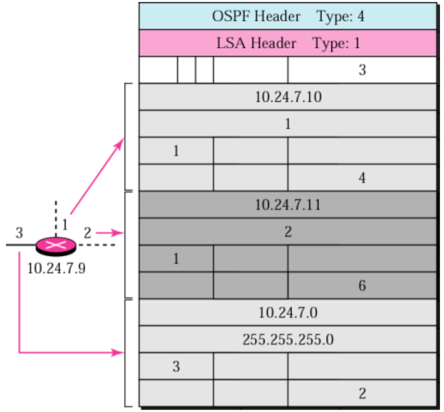


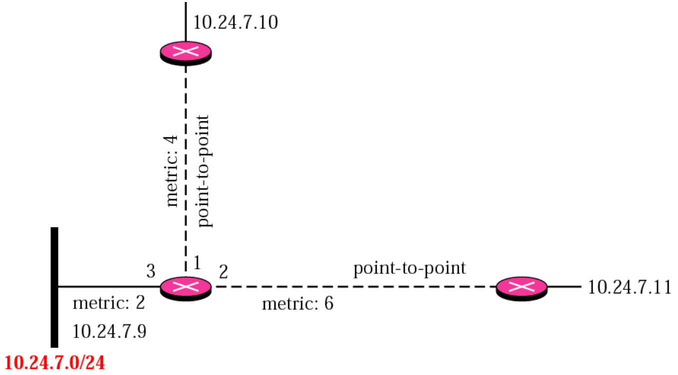


Link State Update Packet
A router link example

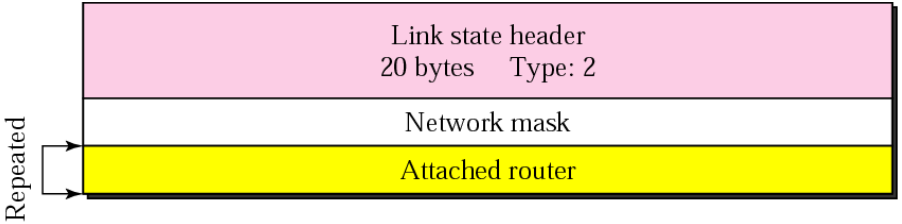
LSA header not covered
Refer to RFC 1247

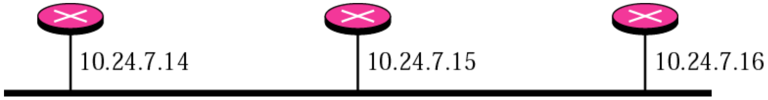






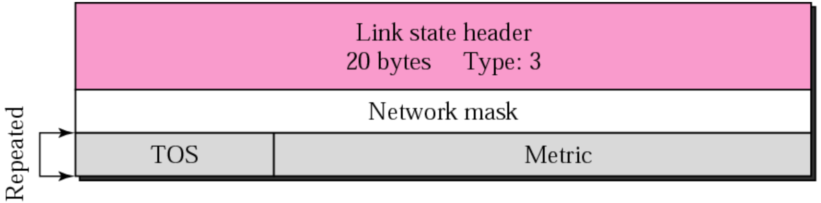
A Network Link Example



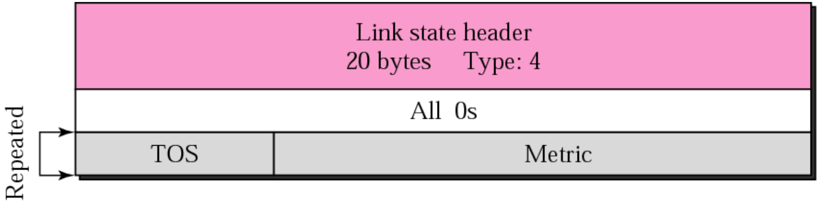


OSPF Header Type: 4
LSA Header Type: 2
255.255.255.0
10.24.7.14
10.24.7.15
10.24.7.16

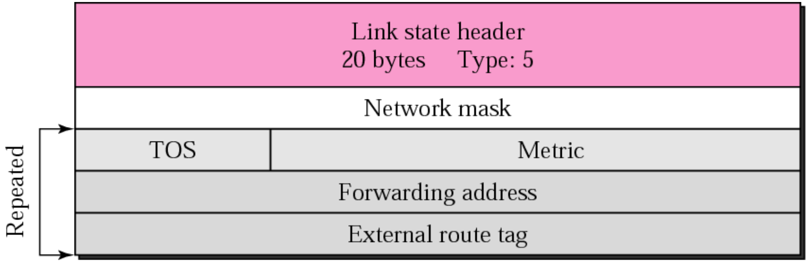
Summary Links state Advertisements



Summary link to network



Summary link to AS boundary



External Link

Quagga

Quagga is a routing software suite, providing implementations of OSPFv2, OSPFv3, RIP v1 and v2, RIPng and BGP-4 for Unix platforms, particularly FreeBSD, Linux, Solaris and NetBSD. Quagga is a fork of GNU Zebra which was developed by Kunihiro Ishiguro.

The Quagga architecture consists of a core daemon, *zebra*, which acts as an abstraction layer to the underlying Unix kernel and presents the Zserv API over a Unix or TCP stream to Quagga clients. It is these Zserv clients which typically implement a routing protocol and communicate routing updates to the zebra daemon. Existing Zserv implementations are:

IPv4	IPv6	
zebra		- kernel interface, static routes, zserv server
ripd	ripngd	- RIPv1/RIPv2 for IPv4 and RIPng for IPv6
ospfd	ospf6d	- OSPFv2 and OSPFv3
bgpd		- BGPv4+ (including address family support for multicast and IPv6)
isisd		- IS-IS with support for IPv4 and IPv6

Quagga

To convert the initial scenario into one with route server, first we must modify the configuration of routers RA, RB and RC. Now they must not peer between them, but only with the route server. For example, RA’s configuration would turn into:

```
# Configuration for router 'RA'
!
hostname RA
password ****
!
router bgp 65001
  no bgp default ipv4-unicast
  neighbor 2001:0DB8::FFFF remote-as 65000
!
  address-family ipv6
    network 2001:0DB8:AAAA:1::/64
    network 2001:0DB8:AAAA:2::/64
    network 2001:0DB8:0000:1::/64
    network 2001:0DB8:0000:2::/64
    neighbor 2001:0DB8::FFFF activate
    neighbor 2001:0DB8::FFFF soft-reconfiguration inbound
  exit-address-family
!
line vty
!
```

Which is logically much simpler than its initial configuration, as it now maintains only one BGP peering and all the filters (route-maps) have disappeared.

QUESTIONS & ANSWERS

THANK YOU!