Diskusi NetWorking V1
FRRouting dan VXLAN

Faisal Reza

Agendanya ngapain aja?

- Kenalan dulu yuk!
- Trend Open Networking
- FRRouting apaan sih?
- Konsep Routing Static dan Dynamic
  - BGP
- Advanced Data Center Routing
  - BGP EVPN
- Q&A

@si_faisal +62 85926900755

Tentang Saya

Sertifikasi Professional :

- **MikroTik** MTC [NA, RE, TCE, WE, IPv6E, INE]
- VCA-Cloud from **VMware**
- 2Xpert Certified from www.2x.com
- ZTE-NextGen Network from **ZTE** Coorporation
- **Microsoft** MTA
- **EC-Council** C|SCU dan C|ND
- **MikroTik Certified Trainer** (Venice, Italy, March 2014)
- **Huawei Certified Instructor (HCSI)** HCNA-RS, HCNA-Storage, HCNP-RS
- Certifed **EC-Council** Instructor **(CEI)**
- **H3CNE H3C** Certified Netork Engineer & Instructor

FAISAL REZA

IT Professional since 2009
(10+ years experience)

Certified Instructor

Background : Engineer >>> Instructor

MikroTik

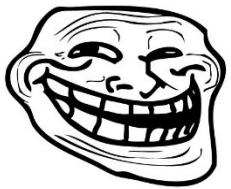
Certified Trainer

HUAWEI

Routing & Switching
Certified System Instructor

EC-Council

Certified Instructor



Network specialist Gado-gado?

H3C
The Leader in Digital Solutions

Certified Network Instructor

Latest Reputable Training Activities

MikroTik Academy Training
INSTITUT TEKNOLOGI BANDUNG
Sekolah Tinggi Elektro dan Informatika (STEI)
Juni 2018

Coorporate Inhouse Training
Gasnet & PGNcom Batam

H3C Partner & Customers
Pakistan

Latest Reputable Project Activities

Server Infrastructure & Venue Network Service
Indonesia Asian Para Games 2018
supporting INAPGOC and WSL MSC Sdn Bhd
(August– Oktober 2018)

One Stop Platform to **Grow StartUp**

www.idcloudhost.com

Introducing Us

Formed in 2015, IDCloudHost is fast growing Cloud Hosting Company focused on providing products and services to any industrial part and professional that are needed to go digital, not to mention startup industry.

26 best employees

60000+ customers

4 data center
> Indonesia (IDC3D, Bali Tower& CDC Cyber 1)
> International (Singapore, London)

Nominee of Cloud Journey of The Year DCD Awards 2017

Our **Products**

Cloud Hosting

cPanel plesk

VPS

Virtual Private Server

Colocation Server

DS

Dedicated Server

Domain

License

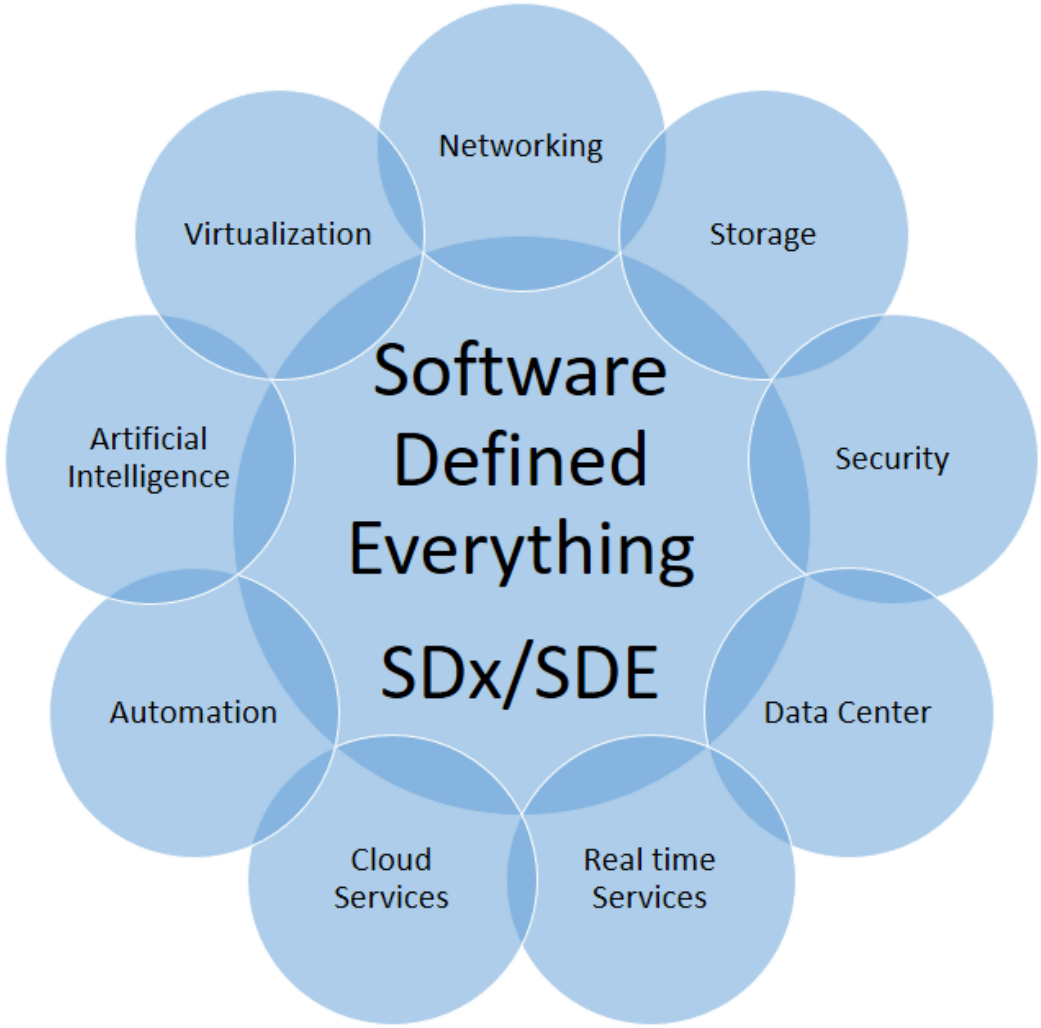
SSL

G Suite

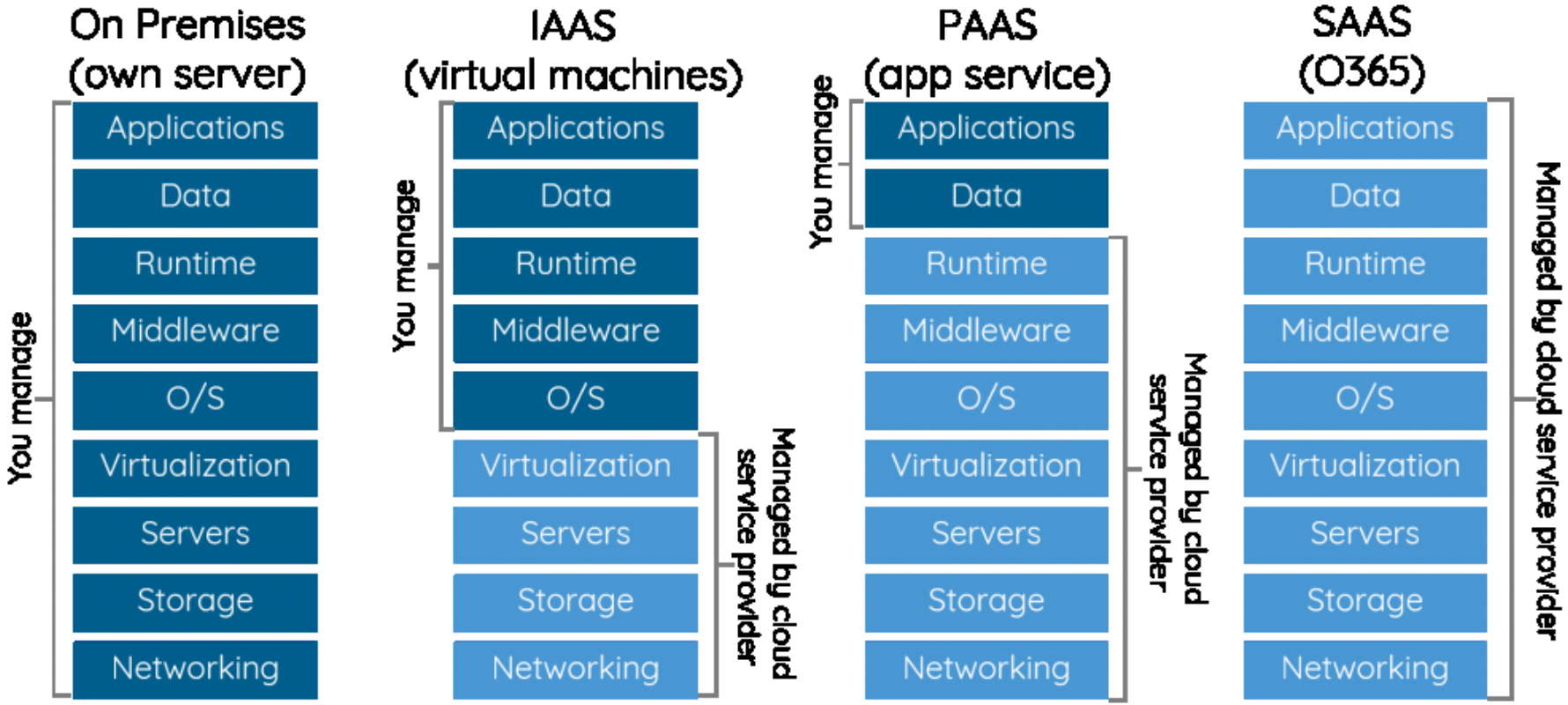
Featured **Media**

Satisfied **Customer**

Certification & Partners

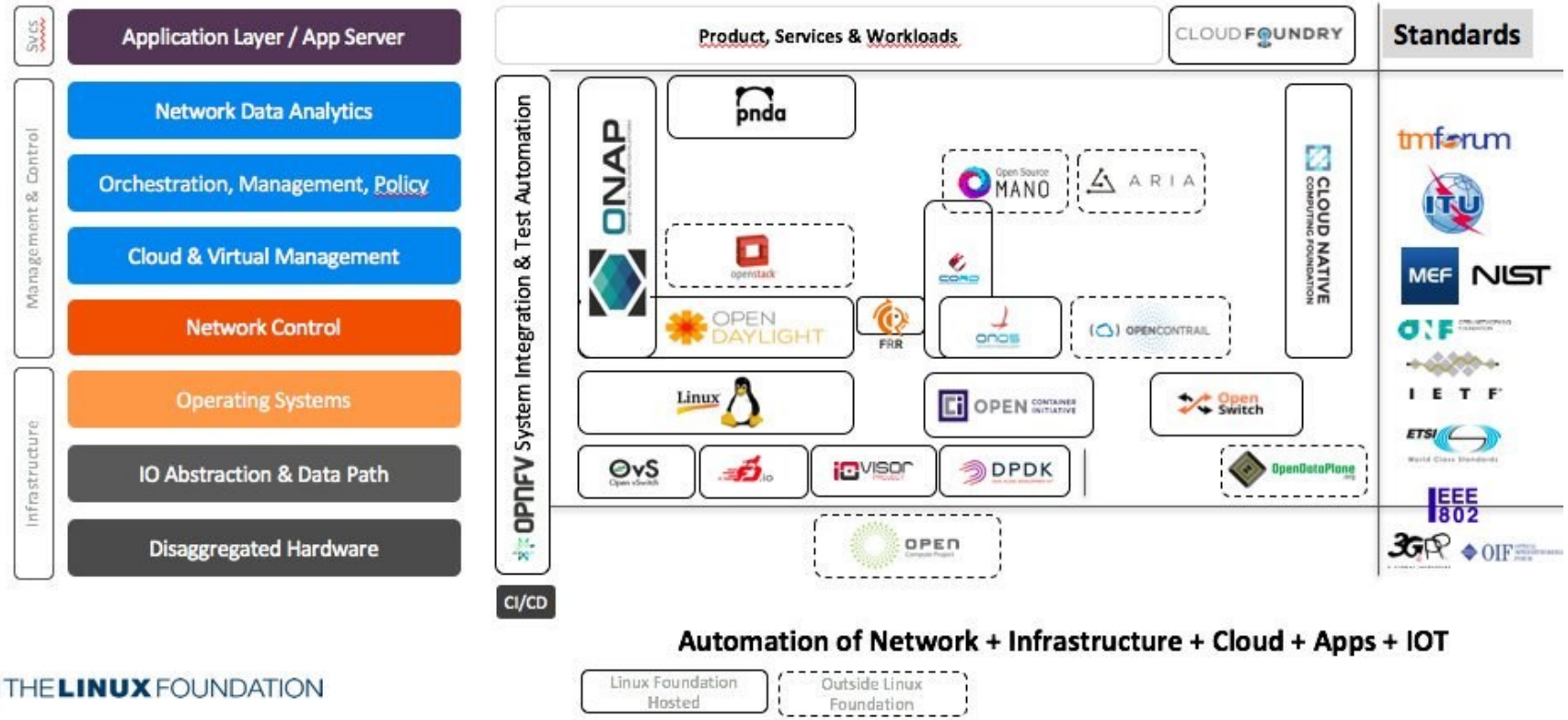

Networking
Virtualization
Storage
Artificial Intelligence
Software Defined Everything
Security
SDx/SDE
Automation
Data Center
Cloud Services
Real time Services

Ada apa dengan Cloud?



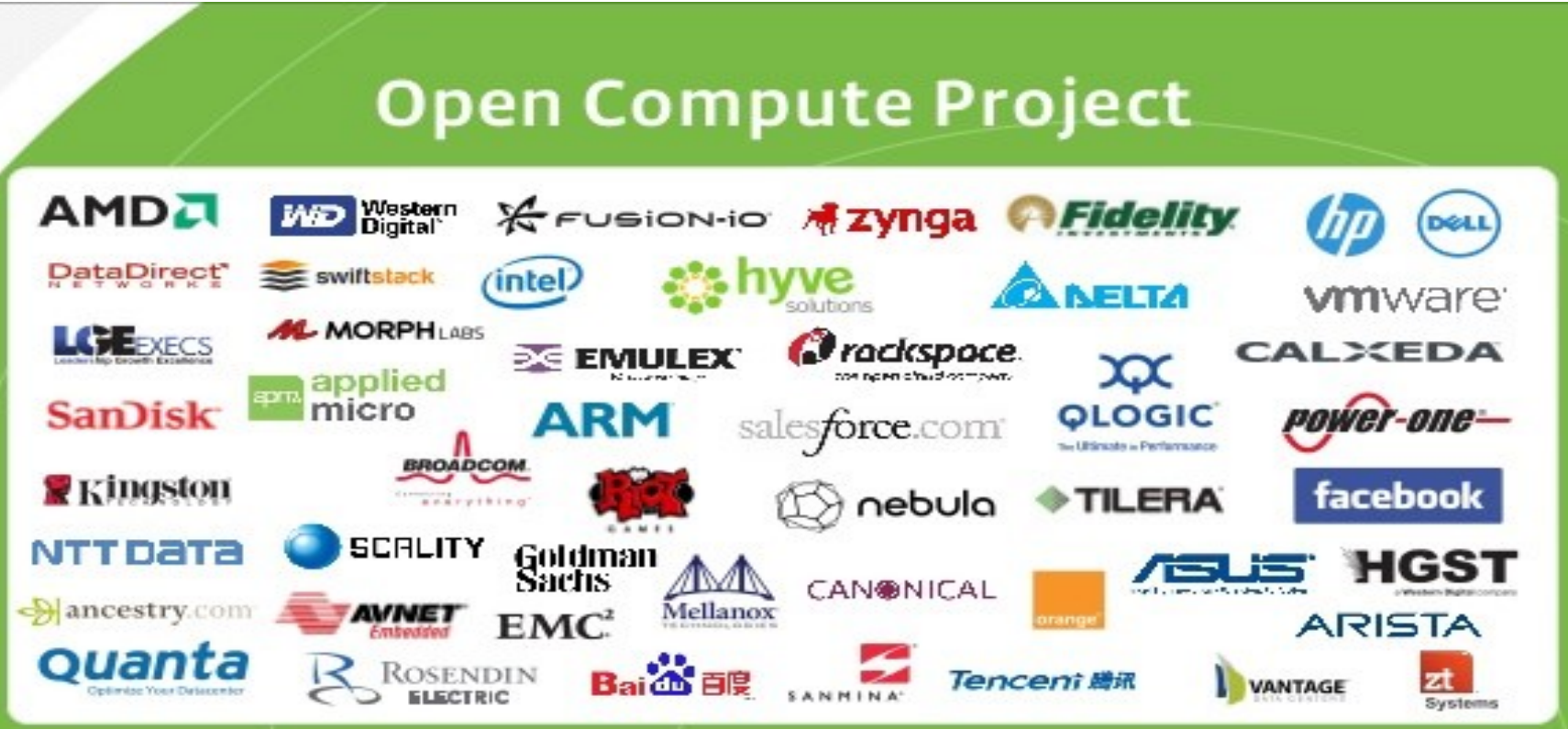
https://www.lfnetworking.org/

Open Source Networking

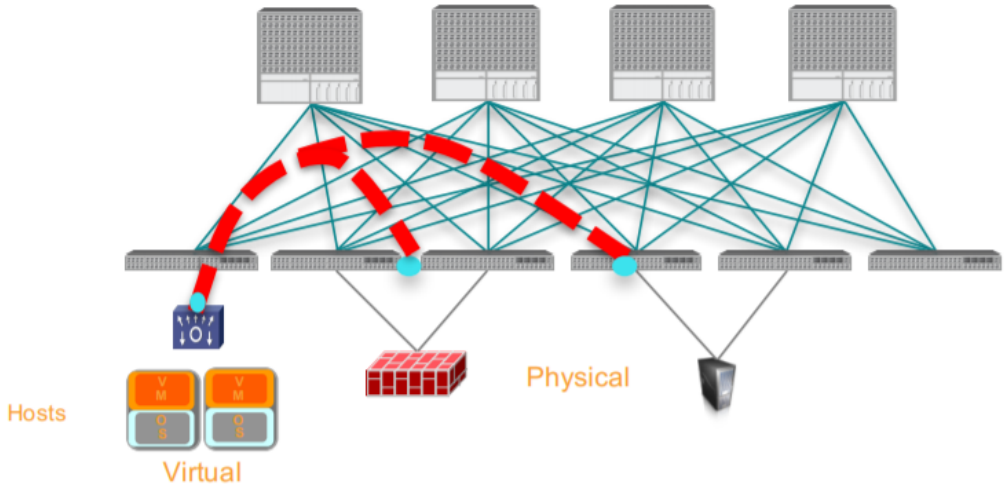


Era ‘Openness’ https://www.opencompute.org/

https://www.slideshare.net/yandex/001-john-keneveyopencomputeprojecthistoryvaluepropositionandfuturetrajectory



Trend: Flexible Data Center Fabrics

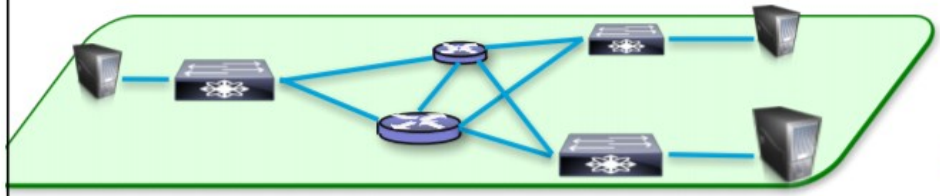


Use VXLAN to Create DC Fabrics

Mobility
Segmentation + Policy
Scale
Automated & Programmable
Full Cross Sectional BW
L2 + L3 Connectivity
Physical + Virtual

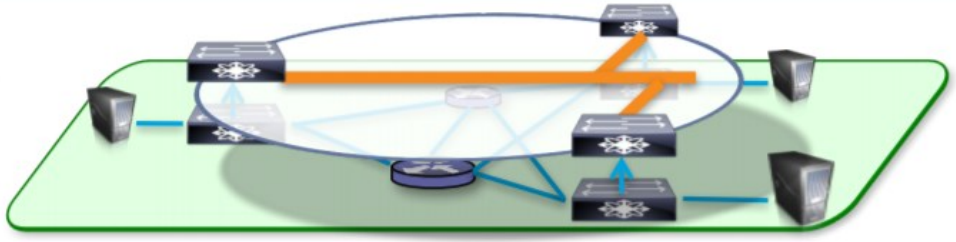
Why Overlays?

Seek well integrated best in class Overlays and Underlays



Robust Underlay/Fabric

- High Capacity Resilient Fabric
- Intelligent Packet Handling
- Programmable & Manageable

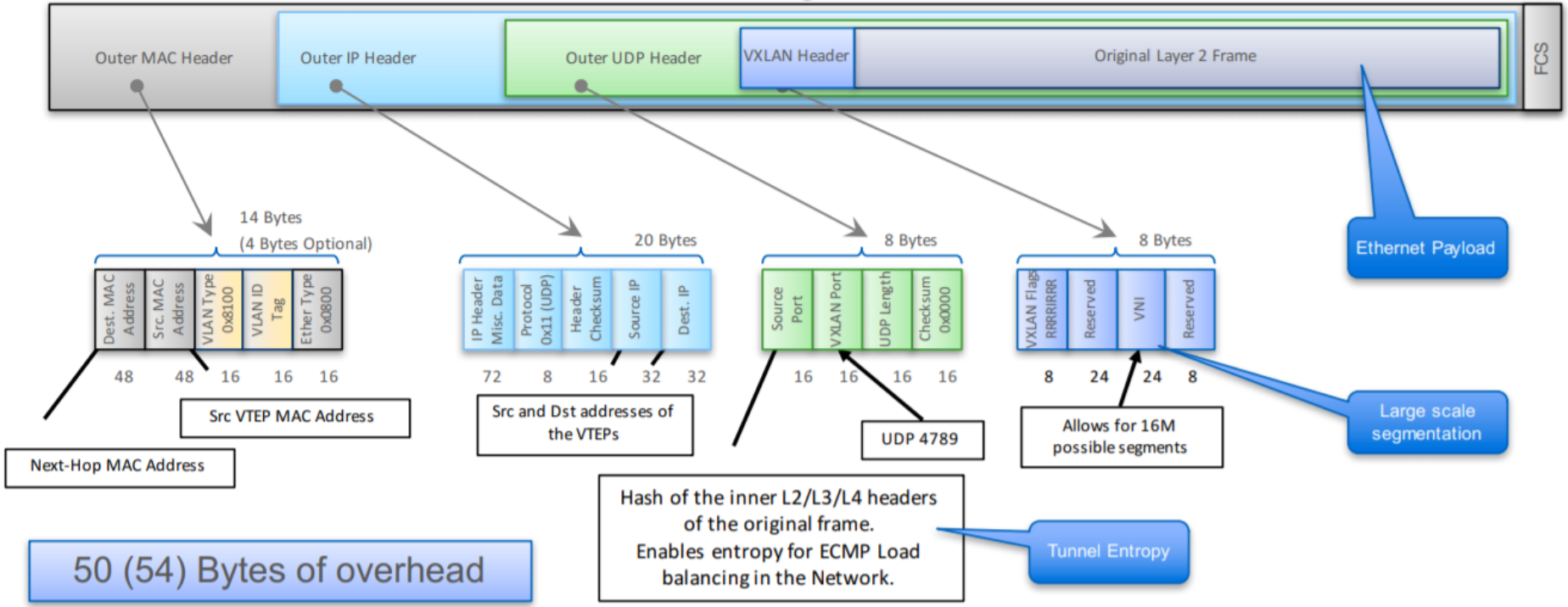


Flexible Overlay Virtual Network

- Mobility – Track end-point attach at edges
- Segmentation
- Scale – Reduce core state
  - Distribute and partition state to network edge
- Flexibility/Programmability
  - Reduced number of touch points

VXLAN Packet Structure

Ethernet in IP with a shim for scalable segmentation



FRRouting apaan ya?

- Memiliki kemampuan yang 'sama' dengan vendor pada umumnya
  - Di develop dan disupport oleh banyak perusahaan teknologi jaringan
    - https://frrouting.org
- Cara konfigurasi nya yang mirip dengan 'vendor mainstream'
  - Configuration file dan interactive CLI
- Berjalan secara native di atas OS Linux dan OS lainnya
  - Fitur di design dan berjalan sepenuhnya di Linux
- Mengutilisasi Routing stack pada kernel Stack untuk melakukan packet forwarding
- Lisensi GPLv2+

Sejarah Singkat FRR

LINUX FOUNDATION COLLABORATIVE PROJECTS

~1996 - Zebra development dimulai

↓

~2002 - Quagga forked dari Zebra

↓

2016 - FRR forked dari Quagga

↓

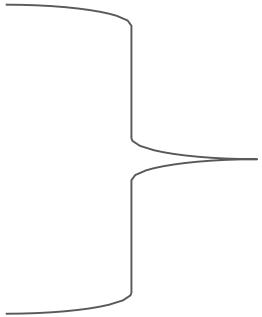
Jan. 2017 – Rilis Pertama - FRR 2.0

↓

July 30, 2019 - Latest Stable - FRR 7.1

Fitur FRRouting

- BABEL - RFC 6126
- EIGRP - RFC 7868
- NHRPD - RFC 2332
- PBR
- LDPD - RFC 5561, 5918, 5919, 6667, 7473



- BGP - Large Communities, EVPN (type 2,3 and 5), RPKI, MPLS Based VPN's, VPN/VRF Route Leaking, RFC 5549, Hostname Support, Shutdown Message, FlowSpec
- ISISD - RFC 5120(MT), SPF Backoff
- OSPF - Experimental SR, RFC 5549
- PIM - RFC 4611, 7761(Sparse Mode)
- VRF - supports both l3mdev and Namespace, not all daemons support VRF yet, BGP, ZEBRA, PIM, and OSPF

Informasi lengkap FRRouting wiki: https://github.com/FRRouting/frr/wiki

Gimana cara setupnya?

- Source - https://github.com/frrouting/frr
  - kompilasi dari source code http://docs.frrouting.org/projects/dev-guide/
- Releases - https://github.com/FRRouting/frr/releases
  - Dapat berupa packet DEB dan RPM, instalasi lebih mudah

Available Distributions/Builds On

- Debian
  - 8.1, 9.1, 10
- Ubuntu
  - 12.04, 14.04, 16.04 dan 18.04
- Redhat
  - 6, and 7
- BSD - FreeBSD 9,10, 11. NetBSD 6. OpenBSD 6
  - tidak full fitur!
- Solaris(Omnios)
  - tidak full fitur!

Kernel linux yang diperlukan ?

- VRF – memerlukan l3mdev
  - 4.4 (fungsi terbatas), 4.8 untuk full fitur
- BGP EVPN – memerlukan NTF_EXT_LEARNED dan ARP Suppression
  - 4.17 dan 4.14
- PIM-SM – memerlukan IGMPMSG_WRVIFWHOLE dan RTN_MULTICAST netlink messages
  - 4.18
- Fitur ini hanya tersedia di kernel linux (tidak di support untuk os lainnya)

Start /Stop FRR daemon

- systemctl <start|restart|reload|stop> frr
  - Berupa system service, ketika reload akan mengaplikasikan kofigurasi di file /etc/frr/frr.conf
- vtysh
  - Interactive shell berupa perintah CLI

```
root@bitbox:/home/# systemctl start frr
root@bitbox:/home/# vtysh

Hello, this is FRRouting (version 7.0).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

bitbox# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D -
SHARP,
       F - PBR,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 [0/0] via 10.50.11.1, eth0, 00:00:06
C>* 10.50.11.0/24 is directly connected, eth0, 00:00:06
C>* 192.168.210.0/24 is directly connected, swp1, 00:00:06
C>* 192.168.213.0/24 is directly connected, swp2, 00:00:06
C>* 192.168.214.0/24 is directly connected, swp3, 00:00:06
C>* 192.168.240.1/32 is directly connected, lo, 00:00:06
```

Logging

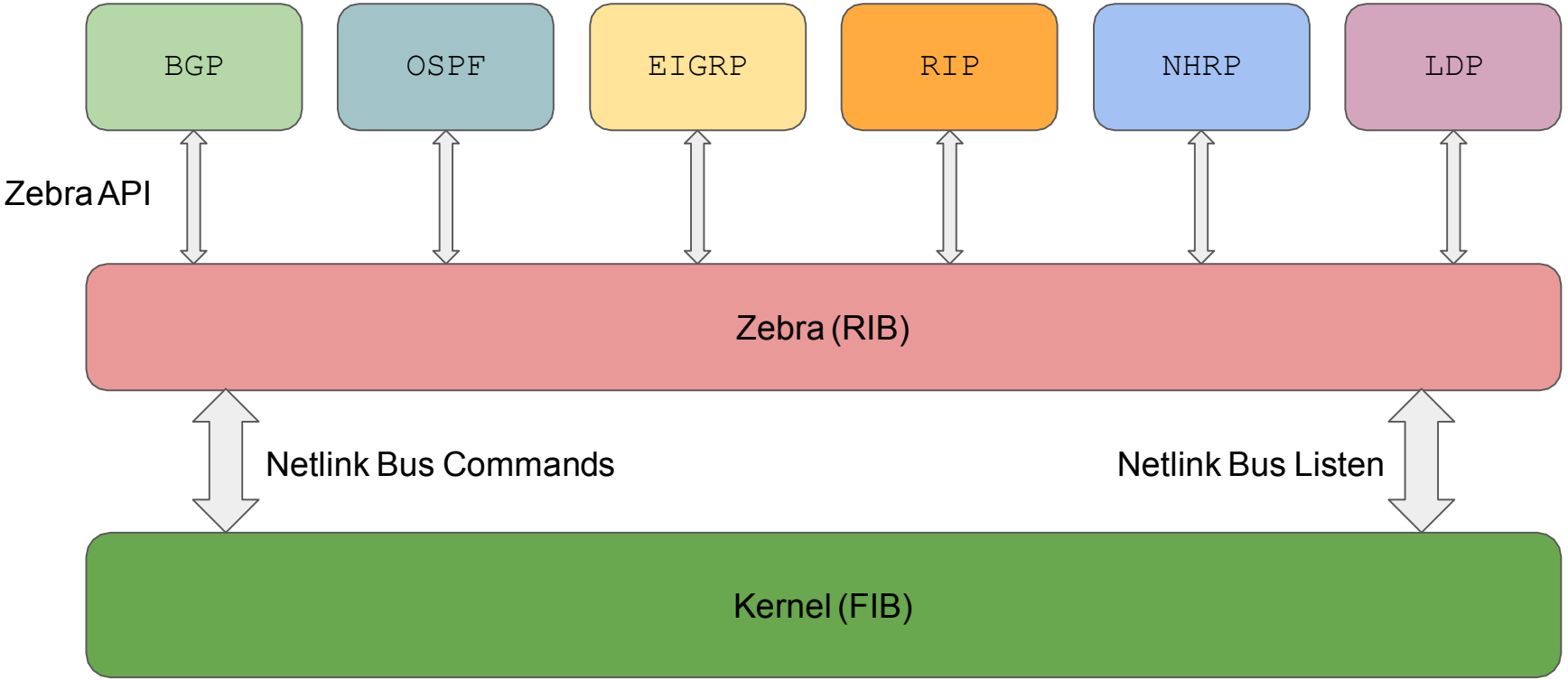
- Lokasi logfile dimana sih?
  - **`show logging`** di vtysh
  - logging destinations:
    1. Syslog - **`log syslog [level]`**
    2. File - **`log file [name]`**
    3. Standard output - **`log stdout`**
- Default nya log nya di set senyap
  - **debug [protocol] ?** Jika ingin menyalakan atau melakukan troubleshooting lebih detail

```
frr(config)# log file
/var/log/frr/frr.log
  frr(config)# log syslog
frr(config)# log stdout
frr(config)#
frr# show logging

Logging configuration for zebra:
Syslog logging: level debugging, facility
daemon, ident zebra
Stdout logging: level debugging
Monitor logging: level debugging
File logging: disabled
Protocol name: ZEBRA
Record priority: disabled
Timestamp precision: 6

...
```

FRR Architecture



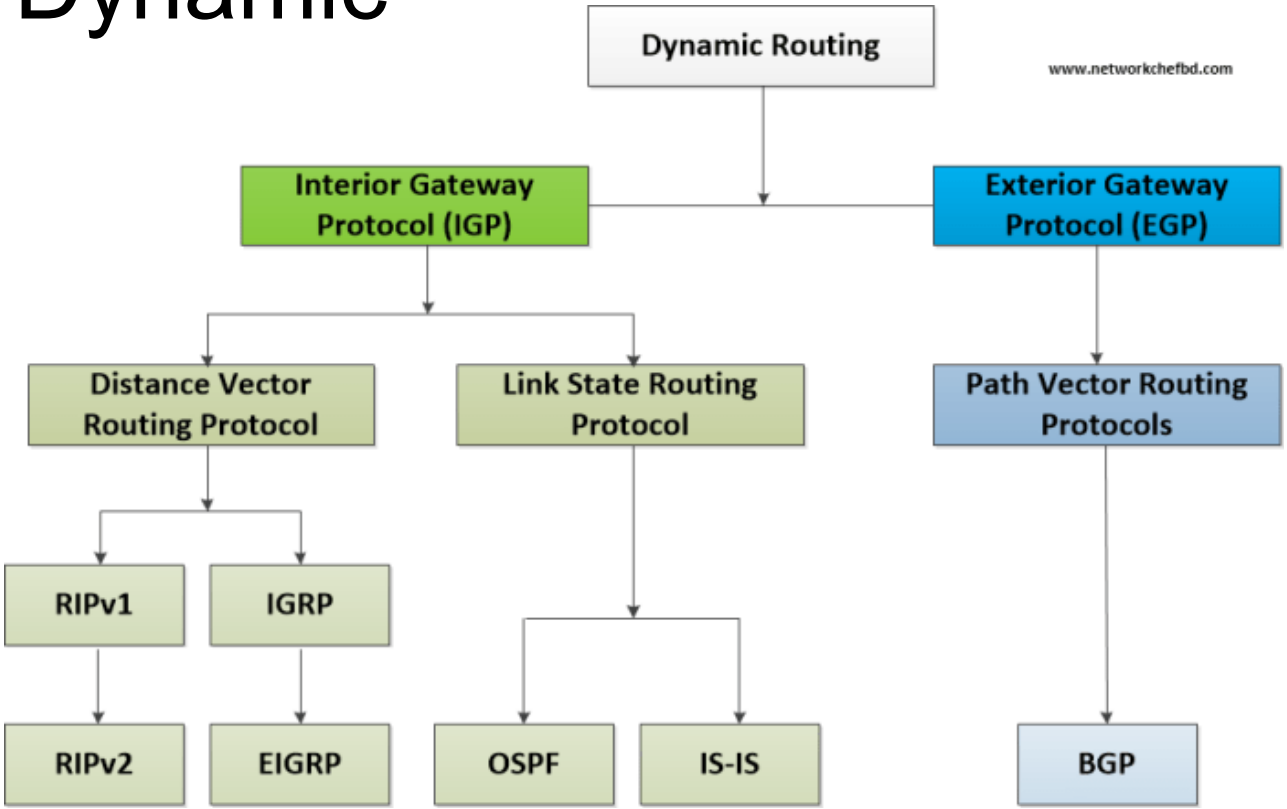
Konsep Routing

Static

Informasi routing yang di set oleh administrator jaringan

contoh: default gateway

Dynamic



Basic Routing

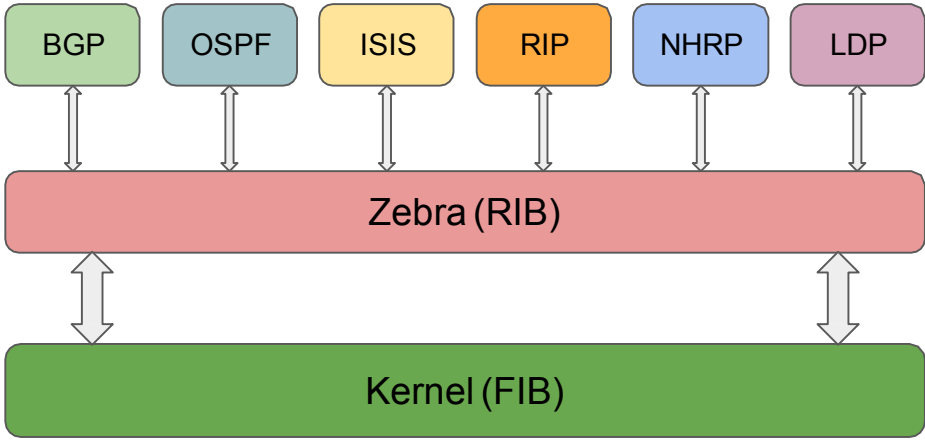
- Why?
- RIB -vs- FIB
- Metric
- Administrative Distance
- VRF
- Basic Topology and Config being used
- BGP Overview
- OSPF Overview
- IS-IS Overview

Why Routing?

- Connecting digital world
- Not getting simpler with what people want to do with them
  - VM’s, Containers, etc.
- Routing allows us to control this chaos
  - Linux now has the ability to interact with the entirety of the network via a standards based approach

RIB -vs- FIB

- Routing Information Base
  - A.k.a ‘Control Plane’
  - This is in FRRouting
- Forwarding Information Base
  - A.k.a ‘Data plane’
  - This is in the Kernel



FRR

```
bitbox# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR,
       > - selected route, * - FIB route

K * 0.0.0.0/0 [0/100] via 10.0.2.2, enp0s3 inactive, 01:24:15
D   4.3.2.1/32 [150/0] via 192.168.209.44, enp0s9, 00:00:02
K>* 4.3.2.1/32 [0/14000] is directly connected, enp0s8, 00:01:05
S   4.3.2.1/32 [1/0] is directly connected, enp0s10, 16:30:20
K * 4.3.2.1/32 [255/8192] is directly connected, enp0s9, 16:33:31
```

Kernel

```
root@bitbox~> ip route show
default via 10.0.2.2 dev enp0s3 proto static metric 100
4.3.2.1 dev enp0s8 scope link metric 14000
4.3.2.1 dev enp0s9 scope link metric 4278198272
```

What is a Metric?

- Value used to figure out the ‘best’ route towards a destination
- Kernel has a metric it receives for a route
  - Assigned by the process that installs it
- Zebra receives metrics from each routing protocol for routes

Routing Protocol	Measurement	Range
RIP	Hop Count	1-16
EIGRP	Composite Metric	0-4.2 billion
OSPF	Link State Distance	> 100

- How do you determine the best metric to use?

Administrative Distance

- Value assigned to Routing Protocols to define precedence when comparing between them
- Lower value wins
- Default values assigned to each protocol [0-255]

```
/* From zebra/zebra_rib.c - Each route type's string
 * and default distance value.
 */
static const struct {
        int key;
        int distance;
} route_info[ZEBRA_ROUTE_MAX] = {
        [ZEBRA_ROUTE_SYSTEM] = {ZEBRA_ROUTE_SYSTEM, 0},
        [ZEBRA_ROUTE_KERNEL] = {ZEBRA_ROUTE_KERNEL, 0},
        [ZEBRA_ROUTE_CONNECT] = {ZEBRA_ROUTE_CONNECT, 0},
        [ZEBRA_ROUTE_STATIC] = {ZEBRA_ROUTE_STATIC, 1},
        [ZEBRA_ROUTE_RIP] = {ZEBRA_ROUTE_RIP, 120},
        [ZEBRA_ROUTE_OSPF] = {ZEBRA_ROUTE_OSPF, 110},
        [ZEBRA_ROUTE_ISIS] = {ZEBRA_ROUTE_ISIS, 115},
        [ZEBRA_ROUTE_BGP] = {ZEBRA_ROUTE_BGP, 20 /* IBGP is 200. */},
        [ZEBRA_ROUTE_EIGRP] = {ZEBRA_ROUTE_EIGRP, 90},
        [ZEBRA_ROUTE_NHRP] = {ZEBRA_ROUTE_NHRP, 10},
        [ZEBRA_ROUTE_TABLE] = {ZEBRA_ROUTE_TABLE, 150},
        [ZEBRA_ROUTE_LDP] = {ZEBRA_ROUTE_LDP, 150},
        [ZEBRA_ROUTE_BABEL] = {ZEBRA_ROUTE_BABEL, 100},

        /* no entry/default: 150 */
};
```

Admin Distance: Determining what to install

FRR

```
bitbox# conf t
bitbox(config)# ip route 4.3.2.1/32
enp0s10  bitbox(config)# end
bitbox# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 [0/100] via 10.0.2.2, enp0s3, 00:04:13
S>* 4.3.2.1/32 [1/0] is directly connected, enp0s10, 00:00:04
K * 4.3.2.1/32 [255/8192] is directly connected, enp0s9, 00:03:15
C>* 10.0.2.0/24 is directly connected, enp0s3, 00:04:13
C>* 192.168.208.0/24 is directly connected, enp0s8, 00:04:13
C>* 192.168.209.0/24 is directly connected, enp0s9, 00:04:13
C>* 192.168.210.0/24 is directly connected, enp0s10, 00:04:13
bitbox# exit
```

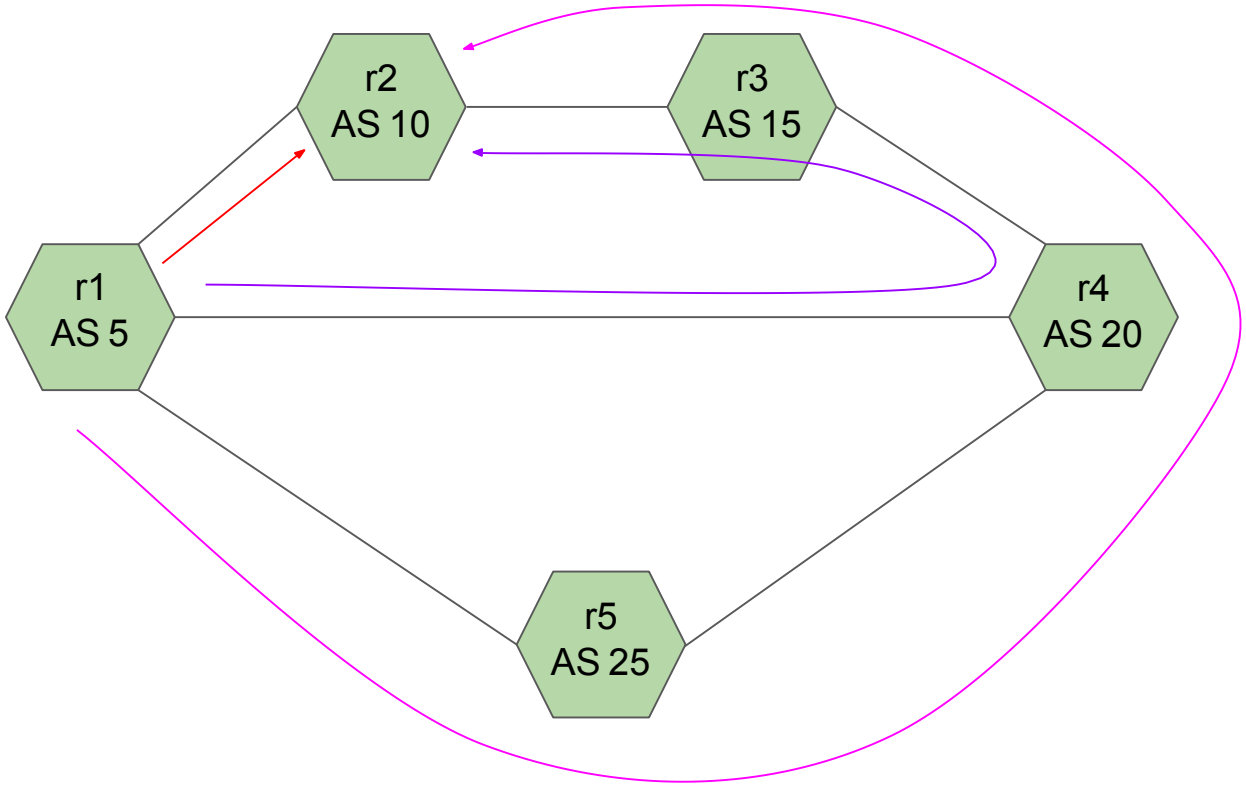
Kernel

```
root@bitbox ~> ip route show
default via 10.0.2.2 dev enp0s3 proto static metric 100
4.3.2.1 dev enp0s10 proto static metric 20
4.3.2.1 dev enp0s9 scope link metric 4278198272
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15
metric 100
192.168.208.0/24 dev enp0s8 proto kernel scope link src
192.168.208.1 metric 100
192.168.209.0/24 dev enp0s9 proto kernel scope link src
192.168.209.1 metric 100
192.168.210.0/24 dev enp0s10 proto kernel scope link src
192.168.210.1 metric 100
```

BGP Overview

- Used for routing on the Internet
  - Connects AS (Autonomous Systems) together
  - Policy Language is rich and featureful to allow operators a great level of control
- 2 modes of operation
  - IBGP (Route Reflector or Full Mesh)
  - EBGP
- Considered a path vector protocol
  - Uses AS Path to determine routes to install
  - AS Path also used for loop avoidance (don't accept a path our our AS in it)
  - What route chosen is a complicated process and simplified here for purposes of discussion
- Uses TCP/IP for connections
- "BGP in the Data Center", Dinesh Dutt (O'Reilly)
- https://tools.ietf.org/html/rfc1771 - This is the starter RFC, too many to list

BGP AS Path determination for Routing r1-r2

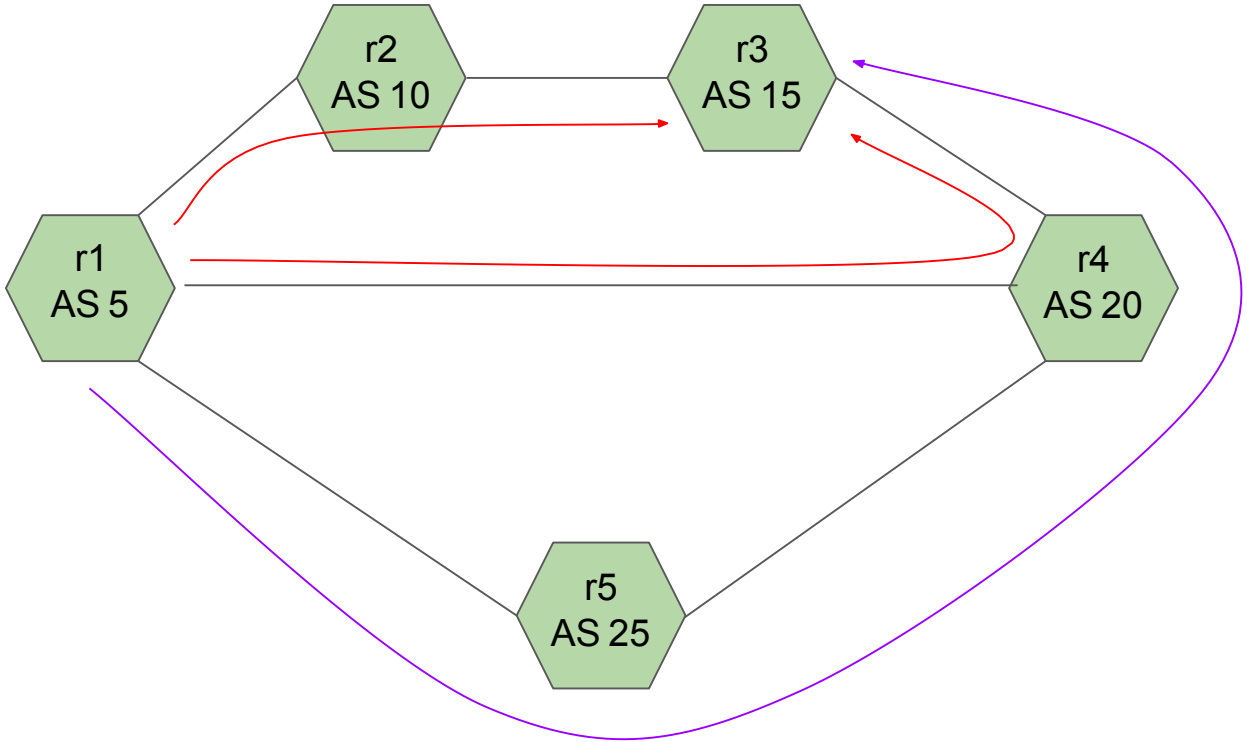


From r1:

To r2: (10),
(20, 15, 10)
(25, 20, 15, 10)

Shortest AS-Path Wins

BGP AS Path determination for Routing r1-r2

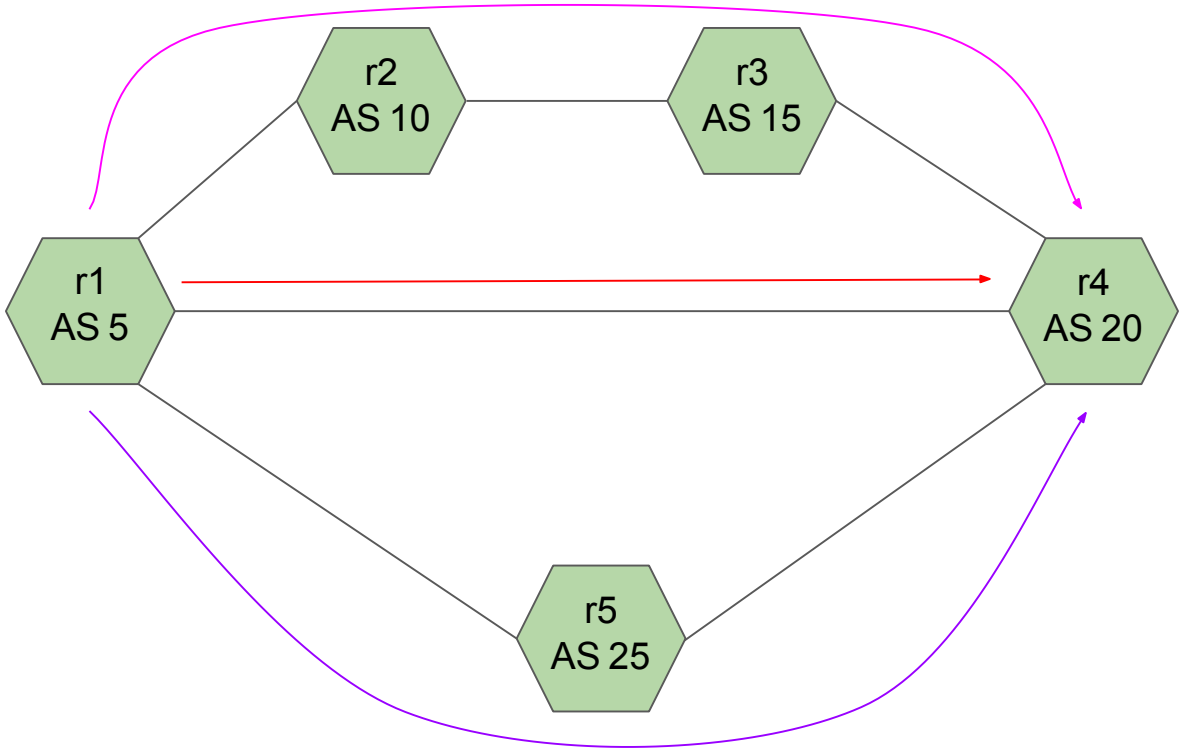


From r1:

To r3: (10, 15)
(20, 15)
(25, 20, 15)

ECMP Shortest
AS-Path Wins

BGP AS Path determination for Routing r1-r2

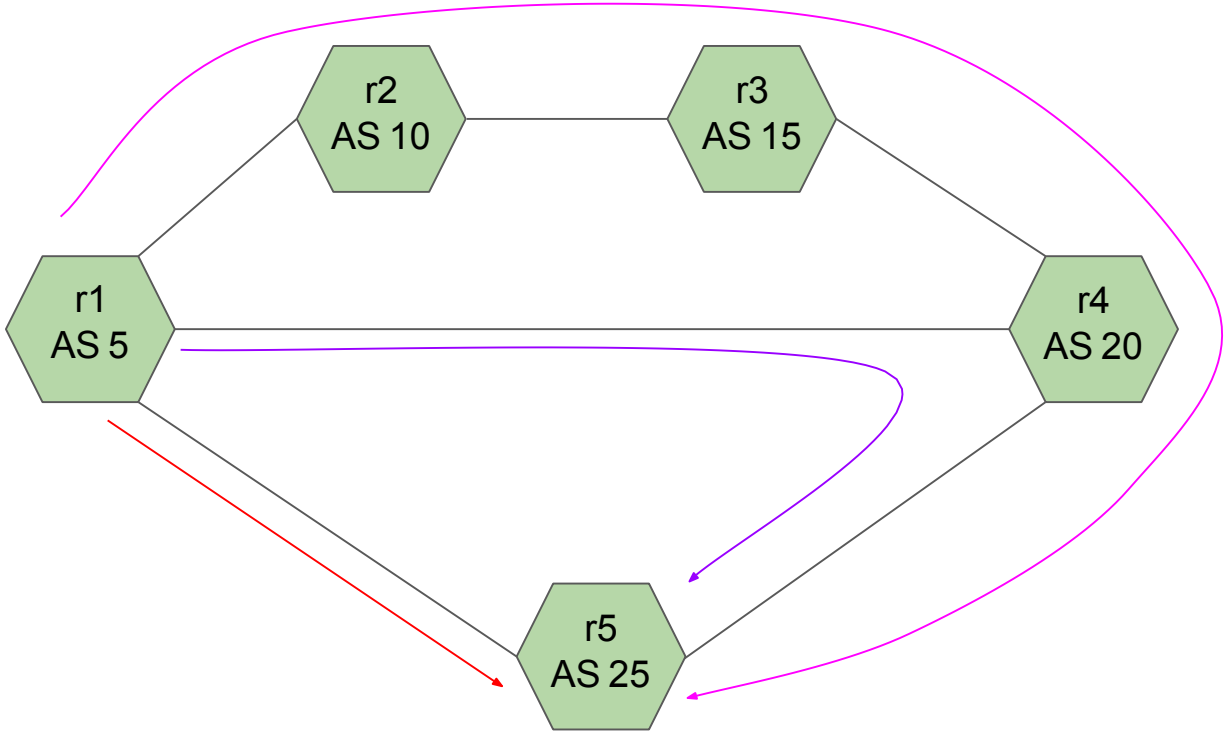


From r1:

To r4: (20)
(10, 15, 20)
(25, 20)

Shortest AS-Path Wins

BGP AS Path determination for Routing r1-r2



From r1:

To r5: (25)
(10, 15, 20, 25)
(20, 25)

Shortest AS-Path Wins

BGP Basic Setup

```
!
router bgp 5
 bgp bestpath as-path multipath-relax
 neighbor 192.168.210.2 remote-as 10
 neighbor 192.168.213.4 remote-as 20
 neighbor 192.168.214.5 remote-as 25
 !
 address-family ipv4 unicast
  redistribute connected
 exit-address-family
!
```

- Neighbors
  - Who to peer with
- Routes
  - What routes do I originate?
    - Default is to originate nothing
  - Must either `redistribute XXX` or use a `network A.B.C.D/M` statement
    - Network statements only import existing prefixes already existing in the RIB
    - To turn off this check use `no bgp network import-check`
- Policy
  - Not discussing policy here but flexible
- multipath-relax allows ecmp to work with different AS-Paths but the same length
- Each router needs to have the neighbor ip address set appropriately

BGP Working Examples

```
r1.rdu.bitbox# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 [0/0] via 10.50.11.1, eth0, 00:41:06
C>* 10.50.11.0/24 is directly connected, eth0, 00:41:06
C>* 192.168.210.0/24 is directly connected, swp1, 00:41:06
B>* 192.168.211.0/24 [20/0] via 192.168.210.2, swp1, 00:13:44
B>* 192.168.212.0/24 [20/0] via 192.168.213.4, swp2, 00:13:25
C>* 192.168.213.0/24 is directly connected, swp2, 00:41:06
C>* 192.168.214.0/24 is directly connected, swp3, 00:41:06
B>* 192.168.215.0/24 [20/0] via 192.168.213.4, swp2, 00:10:53
  *                         via 192.168.214.5, swp3, 00:10:53
C>* 192.168.240.1/32 is directly connected, lo, 00:41:06
B>* 192.168.240.2/32 [20/0] via 192.168.210.2, swp1, 00:13:44
B>* 192.168.240.3/32 [20/0] via 192.168.210.2, swp1, 00:10:53
  *                         via 192.168.213.4, swp2, 00:10:53
B>* 192.168.240.4/32 [20/0] via 192.168.213.4, swp2, 00:13:25
B>* 192.168.240.5/32 [20/0] via 192.168.214.5, swp3, 00:13:16
```

```
root@r1:/home/bitbox# ip route show
default via 10.50.11.1 dev eth0
10.50.11.0/24 dev eth0 proto kernel scope link src 10.50.11.194
192.168.210.0/24 dev swp1 proto kernel scope link src
192.168.210.1
192.168.211.0/24 via 192.168.210.2 dev swp1 proto bgp metric 20
192.168.212.0/24 via 192.168.213.4 dev swp2 proto bgp metric 20
192.168.213.0/24 dev swp2 proto kernel scope link src
192.168.213.1
192.168.214.0/24 dev swp3 proto kernel scope link src
192.168.214.1
192.168.215.0/24 proto bgp metric 20
        nexthop via 192.168.213.4 dev swp2 weight 1
        nexthop via 192.168.214.5 dev swp3 weight 1
192.168.240.2 via 192.168.210.2 dev swp1 proto bgp    metric 20
192.168.240.3 proto bgp metric 20
          nexthop via 192.168.210.2 dev swp1 weight 1
          nexthop via 192.168.213.4 dev swp2 weight 1
  192.168.240.4 via 192.168.213.4 dev swp2  proto bgp  metric 20
   192.168.240.5 via 192.168.214.5 dev swp3 proto bgp  metric 20
```

BGP Debugging

- show bgp ipv4 unicast summary
  - Shows neighbor status and number of prefixes received from peers
- show bgp ipv4 unicast
  - Shows bgp routing table and best path selections
- debug bgp neighbor events
  - Indicates why peering is not coming up, look in log file
- debug bgp updates
  - Indicates what is happening for routes received, look in log file

My most common mistake is not setting up the peering relationship correctly

BGP Debugging Examples

```
r1.rdu.bitbox# show bgp ipv4 unicast summary
BGP router identifier 192.168.240.1, local AS number 5 vrf-id 0
BGP table version 11
RIB entries 11, using 1672 bytes of memory
Peers 3, using 58 KiB of memory

Neighbor                        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
r2.rdu.bitbox(swp1) 4                     10     883     884        0    0    0 00:43:05            4
r4.rdu.bitbox(swp2) 4                     20     784     787        0    0    0 00:38:44            5
r5.rdu.bitbox(swp3) 4                     25     748     747        0    0    0 00:36:54            4

Total number of neighbors 3
```

BGP Debugging Examples Continued

```
frr# show bgp ipv4 uni
BGP table version is 11, local router ID is 192.168.240.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  10.50.11.0/24    swp3                     0             0 25 ?
*                   swp2                     0             0 20 ?
*                   swp1                     0             0 10 ?
*>                  0.0.0.0                  0         32768 ?
*> 192.168.240.1/32 0.0.0.0                  0         32768 ?
*  192.168.240.2/32 swp2                                  0 20 15 10 ?
*>                  swp1                     0             0 10 ?
*  192.168.240.3/32 swp3                                  0 25 20 15 ?
*=                  swp2                                  0 20 15 ?
*>                  swp1                                  0 10 15 ?
*  192.168.240.4/32 swp3                                  0 25 20 ?
*>                  swp2                     0             0 20 ?
*                   swp1                                  0 10 15 20 ?
*> 192.168.240.5/32 swp3                     0             0 25 ?
*                   swp2                                  0 20 25 ?

Displayed 6 routes and 15 total paths
```

What Routing Protocol to Use?

- Use what is most familiar and meets your needs
- BGP Scales better
  - This is mostly due to lack of link state flooding
- BGP can handle many afi/safi combinations that are missing in other routing protocols
  - Can be used as both Overlay and Underlay for vpn networks

Which Routing Protocol should I use?

- PIM
  - If you want to do non Link Local Multicast Routing (`224.0.0.0/4`)
- RIP/EIGRP
  - RIP is ancient and limited in scope, EIGRP is not production ready
- NHRP
  - Wickedly complicated and has a special use case that is not common
- PBR
  - Policy Based Routing, limited use cases - “static routing with a twist”
- BABEL
  - Wireless mesh and home routing
- LDP
  - Label distribution for MPLS

What does FRR Provide?

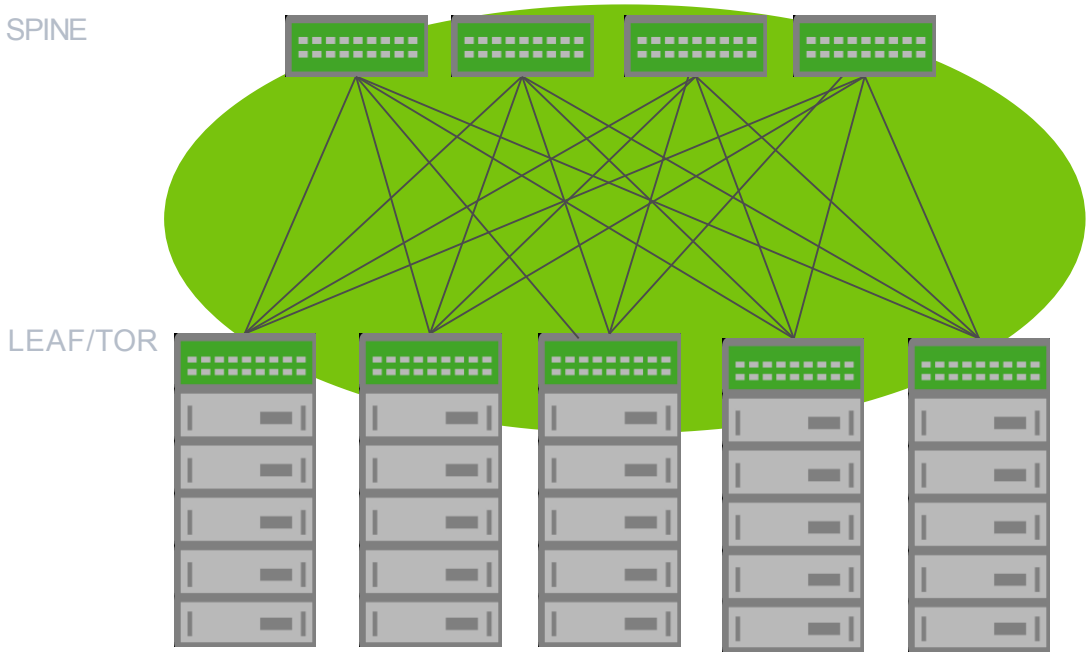
- Provides ability to run routing anywhere in your network
  - Hosts
  - VM’s
  - Containers
- Connect to closed source vendors
  - Using standards based routing protocols

Advanced Data Center Routing

Advanced Data Center Design

- Modern Data Center Architecture
- RFC 5549 (Adv IPv4 NLRI with IPv6 Nexthop)
- BGP EVPN

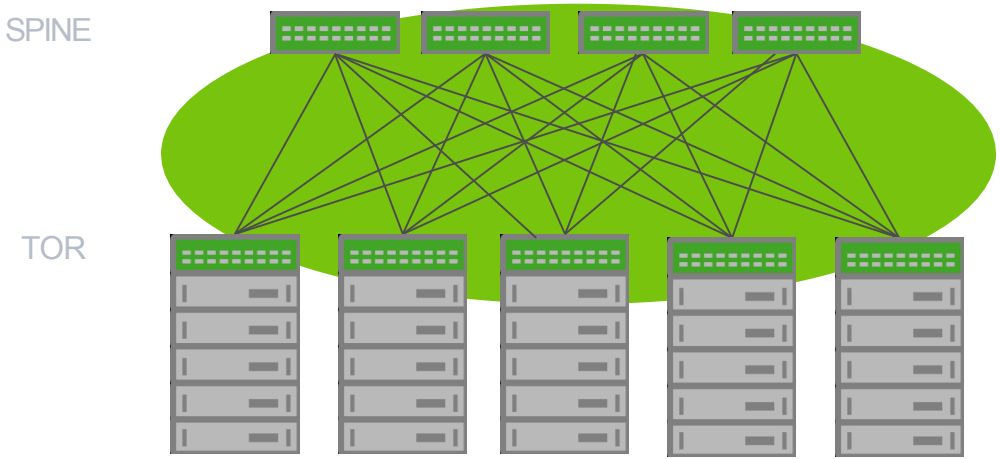
Modern Data Center Architecture



- What's the Point of Clos?
  - Known Latency
  - Guaranteed Bandwidth
  - Easy to build
  - Limited Failure scope
- How to Connect Pods together
  - Just add more Spines and call the middle layers leafs!
- ECMP
  - No STP!
- MLAG is hard(L2 is evil)

RFC 5549 - The what and why

- Simplicity of Config for BGP
- V4 routes with v6 nexthops
- Ideal for datacenter point-to-point links
- Do not have to put a v4 address on every interface
- Commonly Referred to as ‘Unnumbered’



- 45 ipv4 addresses down to 9
- Ratios change based upon the number of links in your CLOS

OSPF Routing Results

```
r1.rdu.bitbox# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 [0/0] via 10.50.11.1, eth0, 00:07:33
C>* 10.50.11.0/24 is directly connected, eth0, 00:07:33
O>* 192.168.215.0/24 [110/1767] via 192.168.240.4, swp2 onlink, 00:00:32
O 192.168.240.1/32 [110/0] is directly connected, lo, 00:02:36
C * 192.168.240.1/32 is directly connected, swp3, 00:07:33
C * 192.168.240.1/32 is directly connected, swp2, 00:07:33
C * 192.168.240.1/32 is directly connected, swp1, 00:07:33
C>* 192.168.240.1/32 is directly connected, lo, 00:07:33
O>* 192.168.240.2/32 [110/100] via 192.168.240.2, swp1 onlink, 00:02:59
O>* 192.168.240.3/32 [110/200] via 192.168.240.2, swp1 onlink, 00:00:34
  *                  via 192.168.240.4, swp2 onlink, 00:00:34
O>* 192.168.240.4/32 [110/100] via 192.168.240.4, swp2 onlink, 00:00:34
O>* 192.168.240.5/32 [110/100] via 192.168.240.5, swp3 onlink, 00:00:07
r1.rdu.bitbox#
```

```
root@r1:/home/bitbox# ip route show
default via 10.50.11.1 dev eth0
10.50.11.0/24 dev eth0 proto kernel scope link src 10.50.11.194
192.168.215.0/24 via 192.168.240.4 dev swp2 proto ospf metric 20 onlink
192.168.240.2 via 192.168.240.2 dev swp1 proto ospf metric 20 onlink
192.168.240.3 proto ospf metric 20
        nexthop via 192.168.240.2 dev swp1 weight 1 onlink
        nexthop via 192.168.240.4 dev swp2 weight 1 onlink
4.      via 192.168.240.4 dev swp2 proto ospf metric 20 onlink
5.      via 192.168.240.5 dev swp3 proto ospf metric 20 onlink
```

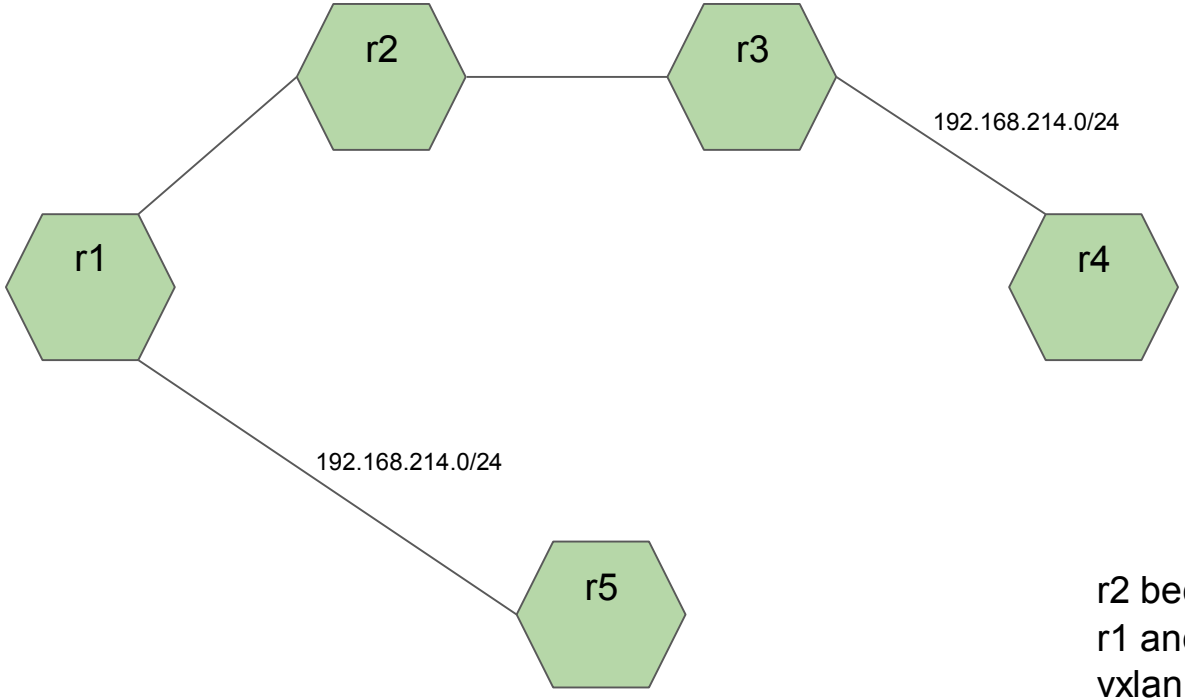
Why L2 is needed in a Data Center

- Legacy Apps still need L2 connectivity
  - Link Local Multicast
    - Service Discovery
  - Only have MAC address of whom to talk to
  - Desire to run applications on non-directly connected hosts
  - Assumption that IP addresses stay the same even when a endpoint is destroyed and recreated

VxLan Encapsulation

- L2 Segmentation over a L3 network
  - Leverage all Data Center Links in the infrastructure
    - STP blocks links
  - Tunnels!
    - More complex control plane to allow higher availability

VxLan and BGP EVPN Setup



r2 becomes the spine
r1 and r3 become leaf/tor with a vxlan interface
r4 and r5 become hosts

```
# r1
# The loopback network interface
auto lo
iface lo inet loopback
    address 192.168.240.1/32

auto swp1
iface swp1

auto swp2
iface swp2

auto swp3
iface swp3
    bridge-access 100

 auto vxlan10100
 iface vxlan10100
    vxlan-id 10100
    vxlan-local-tunnelip 192.168.240.1
    vxlan-learning off
    bridge-access 100

auto br1
iface br1
    bridge-ports swp3 vxlan10100
    bridge-vids 100
```

```
# r2
#The loopback network interface
auto lo
iface lo inet loopback
    address 192.168.240.2/32

auto swp1
iface swp1

auto swp2
iface swp2
```

```
# r3
# The loopback network interface
auto lo
iface lo inet loopback
    address 192.168.240.3/32

auto swp1
iface swp1

auto swp2
iface swp2
    bridge-access 100

 auto vxlan10100
 iface vxlan10100
    vxlan-id 10100
    vxlan-local-tunnelip 192.168.240.3
    vxlan-learning off
    bridge-access 100

auto br1
iface br1
    bridge-ports swp2 vxlan10100
    bridge-vids 100
```

```
# r4
# The loopback network interface
auto lo
iface lo inet loopback
    address 192.168.240.4/32

auto swp1
iface swp1
    address 192.168.214.4/24
```

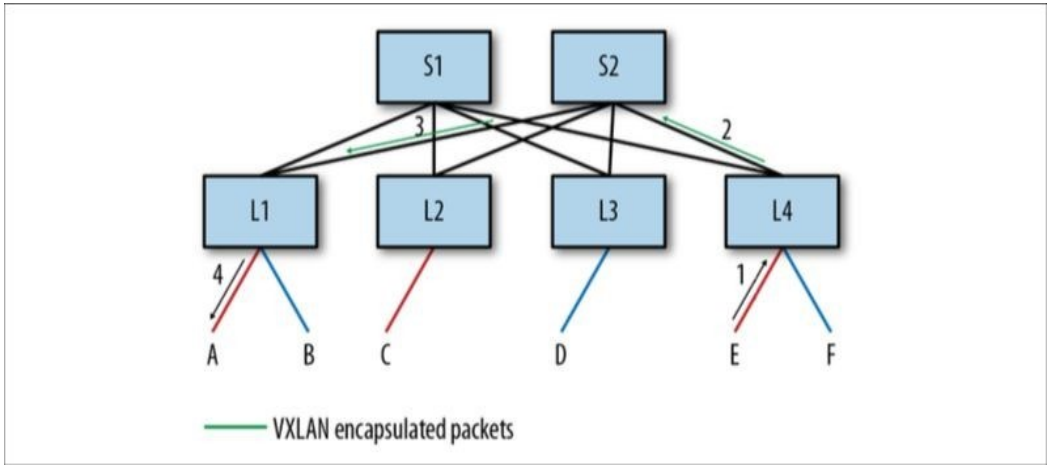
```
# r5
# The loopback network interface
auto lo
iface lo inet loopback
    address 192.168.240.5/32

auto swp1
iface swp1
    address 192.168.214.5/24

auto swp2
iface swp2
    address 192.168.215.5/24
```

BGP EVPN - What is it?

- Simply Connect L2 networks across L3 Underlay Networks
- Unified Control Plane for VxLan
- Auto Distribution of MAC Addresses
  - Allows knowledge of where to send traffic
- Allows for MAC Mobility



https://www.netdevconf.org/2.2/slides/prabhu-linuxbridge-tutorial.pdf

BGP EVPN Setup

```
#r1
!
router bgp 5
 neighbor swp1 interface remote-as external
 !
 address-family ipv4 unicast
  redistribute connected
 exit-address-family
 !
 address-family l2vpn evpn
  neighbor swp1 activate
  advertise-all-vni
 exit-address-family
!
```

```
#r2
!
router bgp 10
 neighbor swp1 interface remote-as external
 neighbor swp2 interface remote-as external
 !
 address-family ipv4 unicast
  redistribute connected
 exit-address-family
 !
 address-family l2vpn evpn
  neighbor swp1 activate
  neighbor swp2 activate
 exit-address-family
!
```

```
#r3
!
router bgp 15
 neighbor swp1 interface remote-as 10
 !
 address-family ipv4 unicast
  redistribute connected
 exit-address-family
 !
 address-family l2vpn evpn
  neighbor swp1 activate
  advertise-all-vni
 exit-address-family
!
```

BGP EVPN Configured

```
r1.rdu.bitbox# show bgp l2vpn evpn summ
BGP router identifier 192.168.240.1, local AS number 5 vrf-id 0
BGP table version 0
RIB entries 3, using 456 bytes of memory
Peers 1, using 19 KiB of memory
Neighbor                        V         AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
r2.rdu.bitbox(swp1) 4                     10    1732    1729        0    0    0 01:21:29            2

Total number of neighbors 1  r2.rdu.bitbox# show bgp
l2vpn evpn summ
BGP router identifier 192.168.240.2, local AS number 10 vrf-id 0
BGP table version 0
RIB entries 3, using 456 bytes of memory
Peers 2, using 39 KiB of memory

Neighbor                        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
a.rdu.bitbox(swp1)              4           5    1794    1796        0    0    0 01:25:00            2
r3.rdu.bitbox(swp2)             4          15    1131    1129        0    0    0 00:54:23            2

Total number of neighbors 2  r3.rdu.bitbox# show bgp
l2vpn evpn summ
BGP router identifier 192.168.240.3, local AS number 15 vrf-id 0
BGP table version 0
RIB entries 3, using 456 bytes of memory
Peers 1, using 19 KiB of memory

Neighbor                        V          AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd
r2.rdu.bitbox(swp1) 4                     10    1131    1137        0    0    0 00:54:41            2

Total number of neighbors 1
```

BGP EVPN Configured

```
r1.rdu.bitbox# show bgp l2vpn evpn  Route
Distinguisher: ip 192.168.240.1:2

 *> [2]:[0]:[0]:[48]:[08:00:27:46:bc:5f]
                     192.168.240.1                          32768 i
 *> [3]:[0]:[32]:[192.168.240.1]
                     192.168.240.1                          32768 i
 Route Distinguisher: ip 192.168.240.3:2

 *> [2]:[0]:[0]:[48]:[08:00:27:70:38:6e]
                     192.168.240.3                              0 10 15 i
 *> [3]:[0]:[32]:[192.168.240.3]
                     192.168.240.3                              0 10 15 i

Displayed 4 out of 4 total prefixes
r1.rdu.bitbox# show evpn mac vni all

VNI 10100 #MACs (local and remote) 2

MAC                         Type     Intf/Remote VTEP      VLAN
08:00:27:46:bc:5f         local     swp3
08:00:27:70:38:6e        remote 192.168.240.3
```

- Type 2 route is the MAC route
- Type 3 route specifies how to send BUM Traffic

Sumber informasi

- Website
  - https://frrouting.org/
- Wiki
  - https://github.com/FRRouting/frr/wiki
- Dokumentasi
  - http://docs.frrouting.org
- GitHub
  - https://github.com/FRRouting

Q & A ?
Terima kasih!

Faisal Reza

B1TBOX OPEN NETWORKING APPLIANCE itu apa ya?

server “RAKITAN”

Di design running 24x7
routing, firewall, server banyak fungsi

Konsep Openness, bisa diinstall / dipasang OS/Aplikasi sesuai keperluan

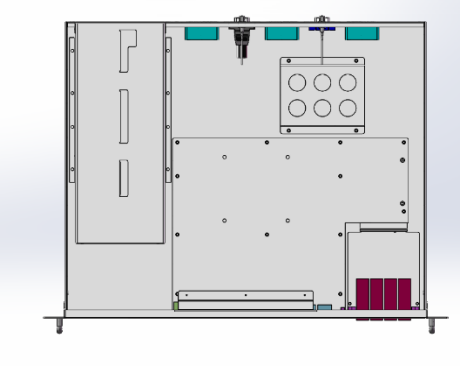
IoT Gateway dan
Embedded System

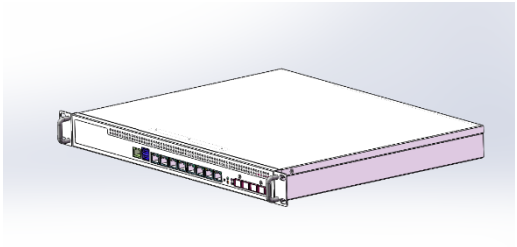
Harga kompetitif

Gimana cerita awalnya?

Pengen punya router & firewall ekonomis untuk di pasang di infrastruktur yang di manage sendiri

Design

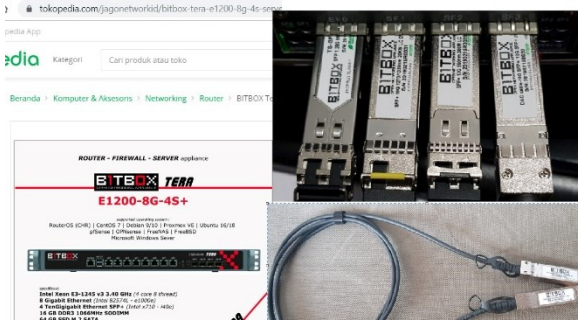




Pergi ke SZX

Factory visit!

Testing Launch



ROUTER - FIRELWALL - SERVER appliance

B1TBOX OPEN NETWORKING APPLIANCE **TERA**

E1200-8G-4S+

supported operating system:

RouterOS (CHR) | CentOS Linux 7 | Debian Linux 9/10 | Ubuntu 16/18
pfSense | OPNsense | FreeNAS | FreeBSD
Proxmox VE | VMware ESXi | CoreOS
Microsoft Windows Server **

spesifikasi:

Intel Xeon E3-1245 v3 3.40 GHz *(4 core 8 threads)*
4 TenGigabit SFP+ *(Intel x710 - i40e)* ***C****onverged* ***N****etwork* ***A****dapter* - ***SR-IOV*** *supported*
8 Gigabit Ethernet *(Intel 82574L - e1000e) - support lan bypass*
16 GB DDR3 SODIMM
128 GB SSD mSATA
Support SATA HDD/SSD 3.5/2.5 inch
Console & USB 2.0 ports
VGA port
Rackmount kit
220 Watt Single AC Power Supply

** belum termasuk license

TERA

intel XEON inside

Trus dipake buat apa BITBOX nya?

support Intel VT-x Virtualization Technology

NANO

BITBOX High Performance Networking Appliance

+

MikroTik BIRD FRROUTING VyOS = **Router**

pfsense OPNsense Your next open source firewall untangle = **Firewall**
Unified Threat Management

CentOS debian ubuntu FreeBSD PROXMOX Windows Server = **Server**
Web, DNS, Mail, NAS, Application
VIRTUALIZATION

YOUR LOGO HERE OS/aplikasi anda = ***Embedded***
System and Application

***dapat melakukan request OS yang diinstall saat pembelian**

RouterOS (CHR) | CentOS Linux 7 | Debian Linux 9/10 | Proxmox VE | Ubuntu 16/18
pfSense | OPNsense | FreeNAS | FreeBSD
Microsoft Windows Server**

contoh aplikasi / implementasi

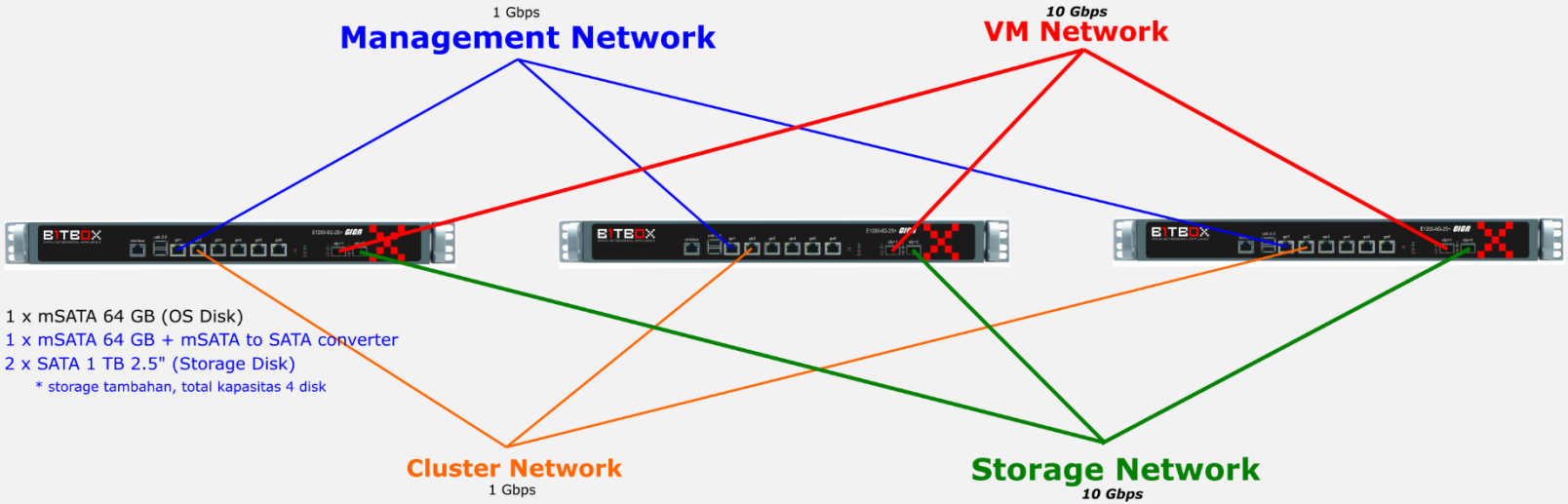
Linux Virtualization Cluster

small deployment lab/production

PROXMOX

openstack CLOUD SOFTWARE

kubernetes



intel XEON inside

ceph hadoop GLUSTER