Problem / Overview

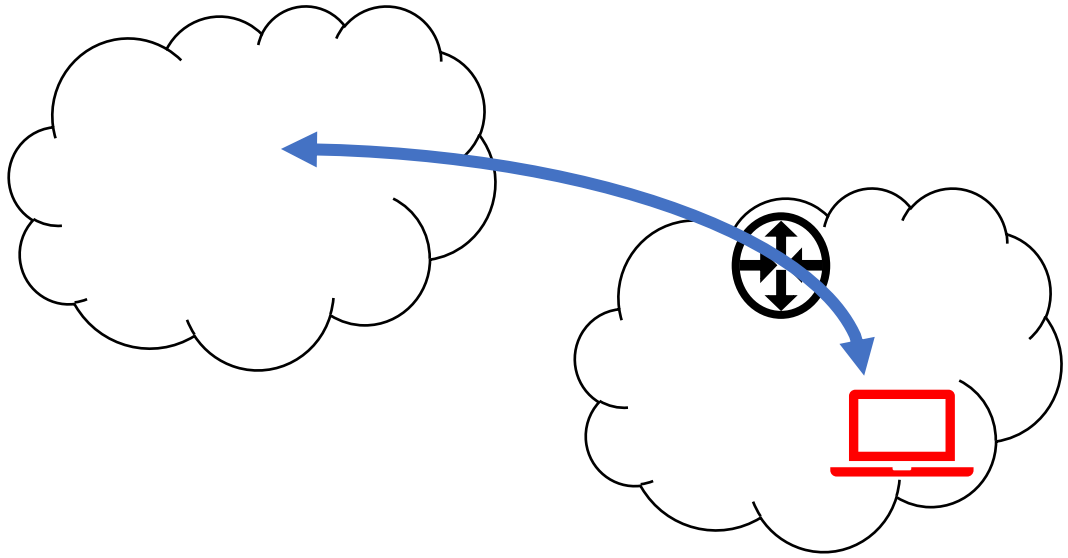
Course: Networking Principles in Practice – Linux Networking
Module: IP Layer with Linux Networking

University of Colorado **Boulder**

Where does IP Layer Come In?

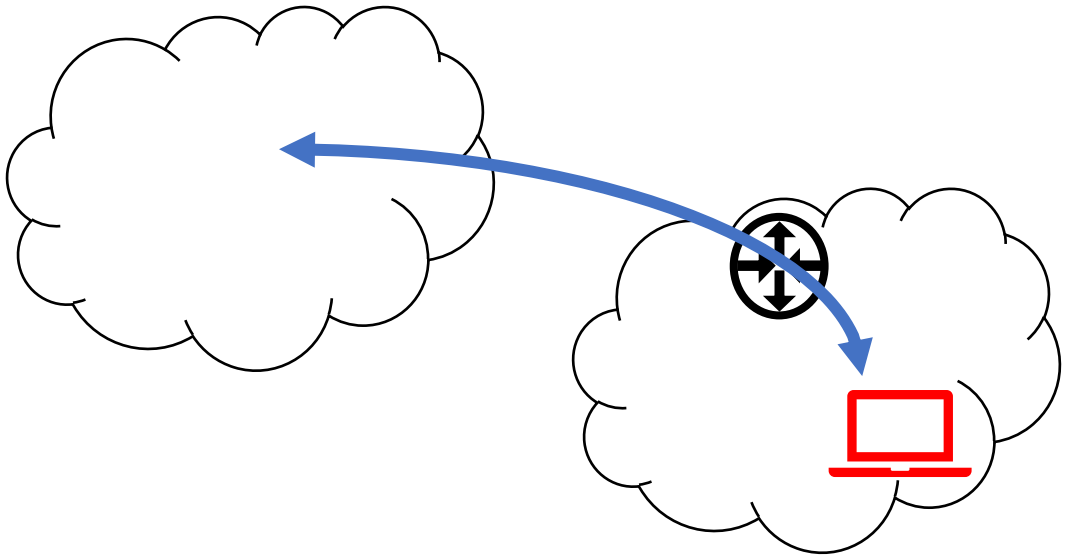
Linux as end host (either as client or server)

- Needs an IP Address
- Needs to be able to know where to send traffic



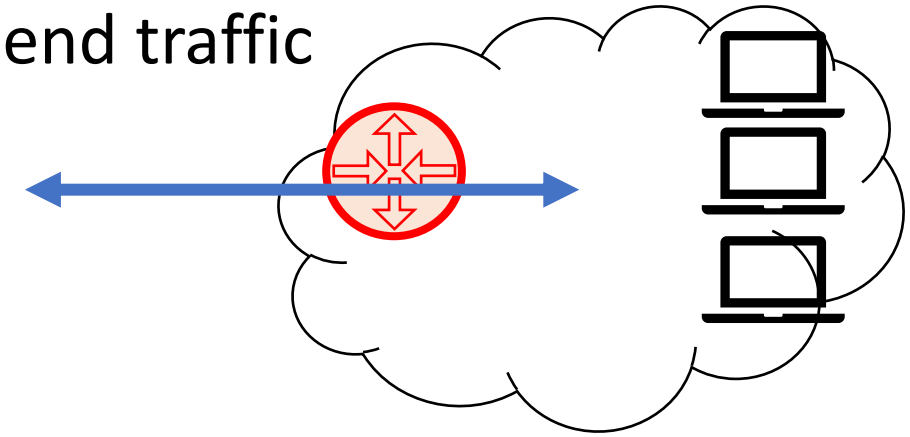
Linux as end host w/ VPN

- We saw how to create tunnel interfaces with ip link
- Need to know which IP traffic to send through the tunnel (e.g., all traffic for work), which to send without tunneling (e.g., general Internet)



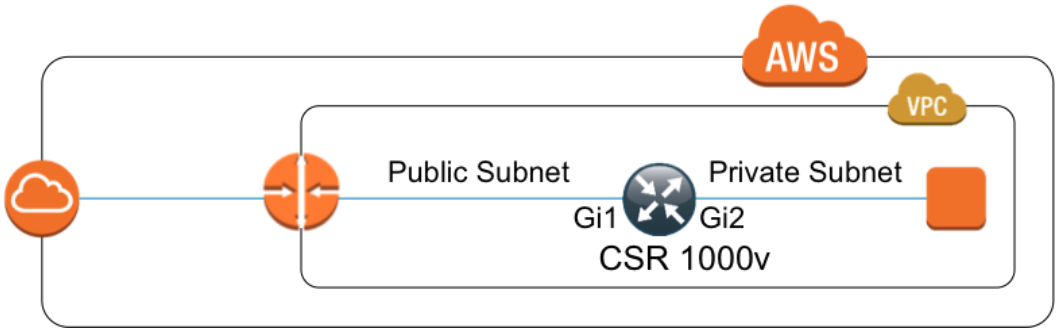
Linux as a Firewall or Load Balancer

- Sitting at the edge of a network
  - One NIC for external (e.g., Internet)
  - One NIC for internal
- Need addresses
- Need to know where to send traffic



Linux as a Virtual Router

Virtual Router in the Cloud



Network Operating System

Motivation – Linux Covers the Whole Stack

Management Plane
(Linux Utilities)

Control Plane
(Linux Ecosystem)

Data Plane
(Linux Kernel)

Upcoming Lessons

- Linux Utilities for IP Layer
- Example IP Layer Walkthroughs
- Routing in Linux
- Routing Walkthrough with Bird
- Larger Routing Experimentation

University of Colorado **Boulder**

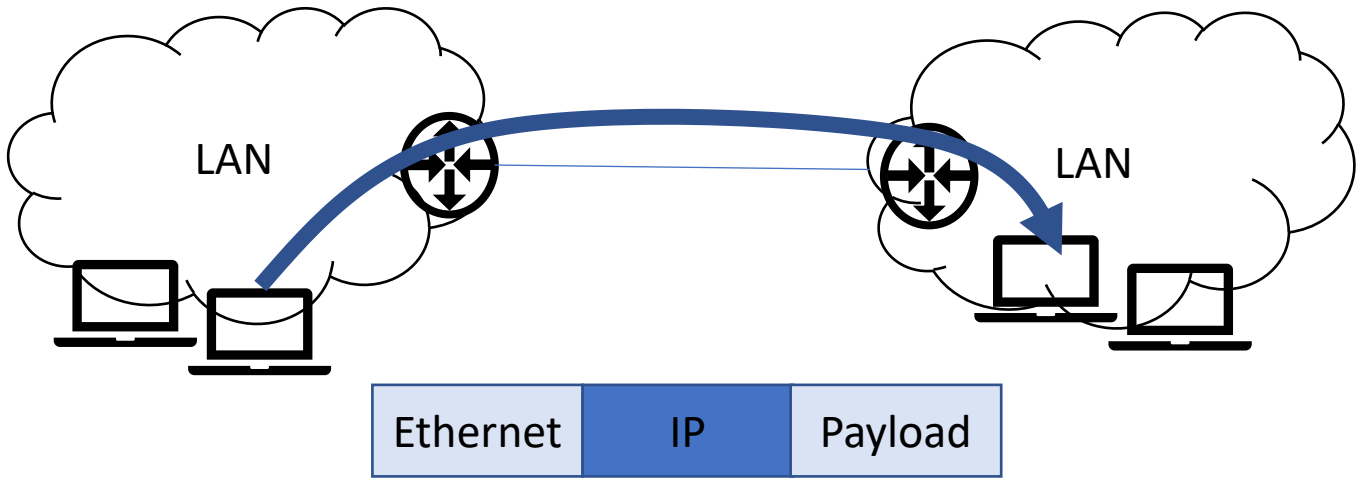
Linux Utilities for IP Layer

Course: Networking Principles in Practice – Linux Networking
Module: IP Layer with Linux Networking

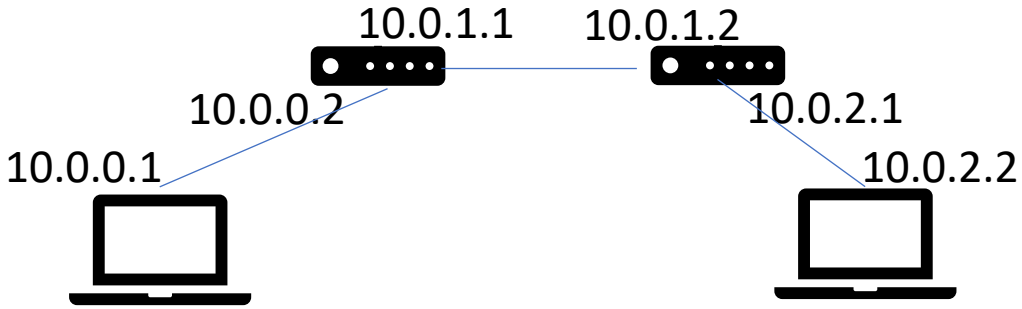
University of Colorado **Boulder**

Internetworking

- Enable communication between multiple, heterogeneous networks
- Key: Router at edge of each network
  (called Gateway in 1974 paper from Cerf & Kahn “A Protocol for Packet Network Intercommunication”)



Addressing in IP



- Each interface gets an IP address
- IPv4 - 32 bits in dotted decimal
  - 192.168.0.1
- IPv6 - 128 bits in hextets (16 bits) separated by a colon
  - *2001:0db8:0000:0000:0000:ff00:0042:8329*
  - *Can drop leading zeros*
    2001:0db8:0:0:0:ff00:0042:8329
  - *Can replace consecutive zeros with ::*
    2001:0db8::ff00:0042:8329

ip addr

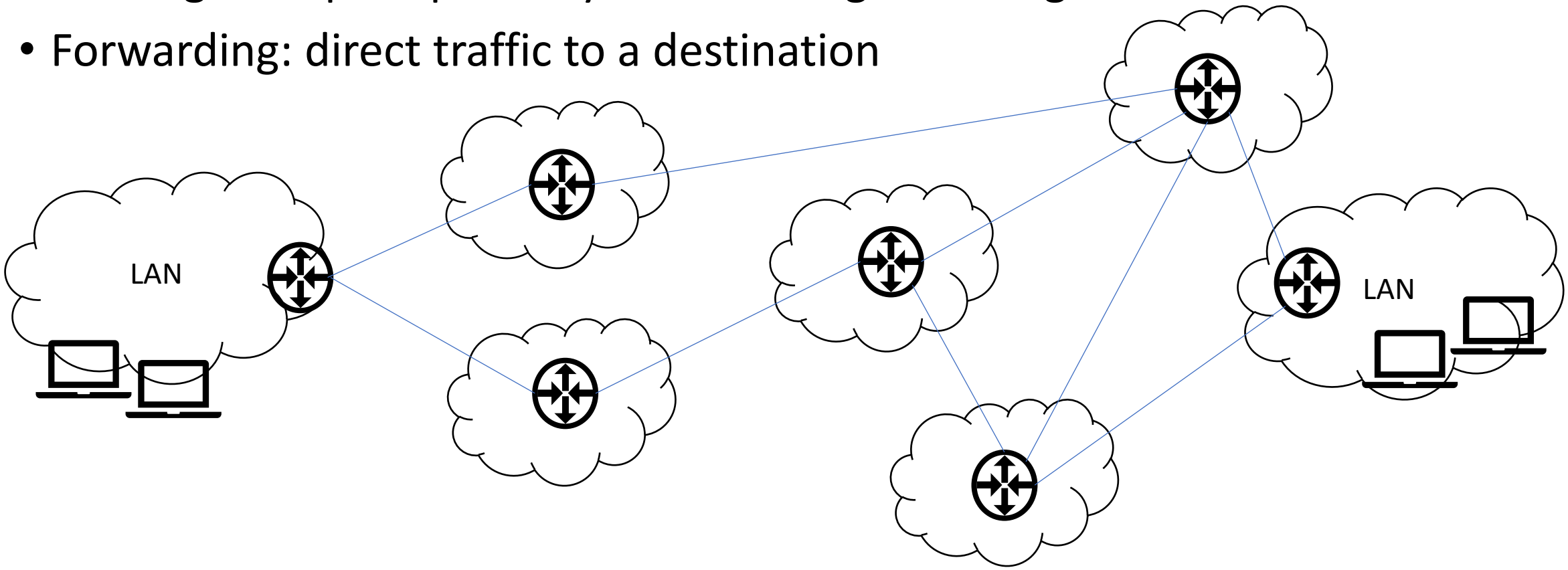
- Each device must have at least one address to use the corresponding protocol.
- It is possible to have several different addresses attached to one device.

ip addr - Utility to display, add, remove, change addresses of devices
ip addr add dev eth1 10.0.1.2

https://man7.org/linux/man-pages/man8/ip-address.8.html

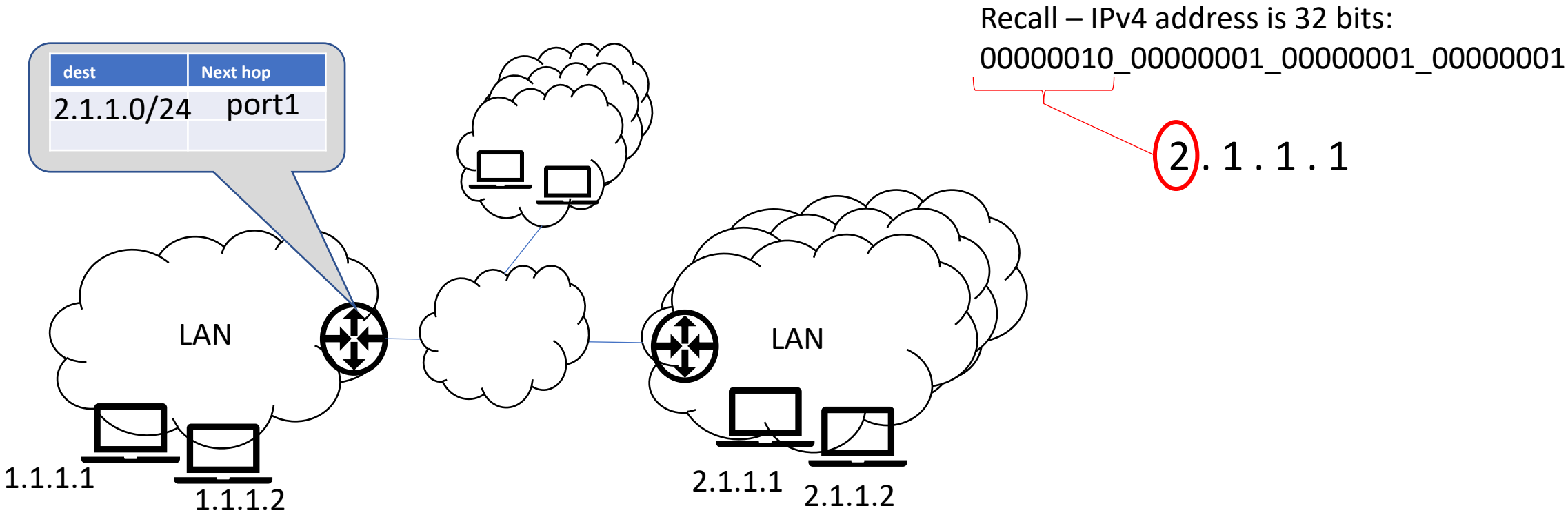
The Role of a Router

- Routing: Compute paths by coordinating with neighbors
- Forwarding: direct traffic to a destination



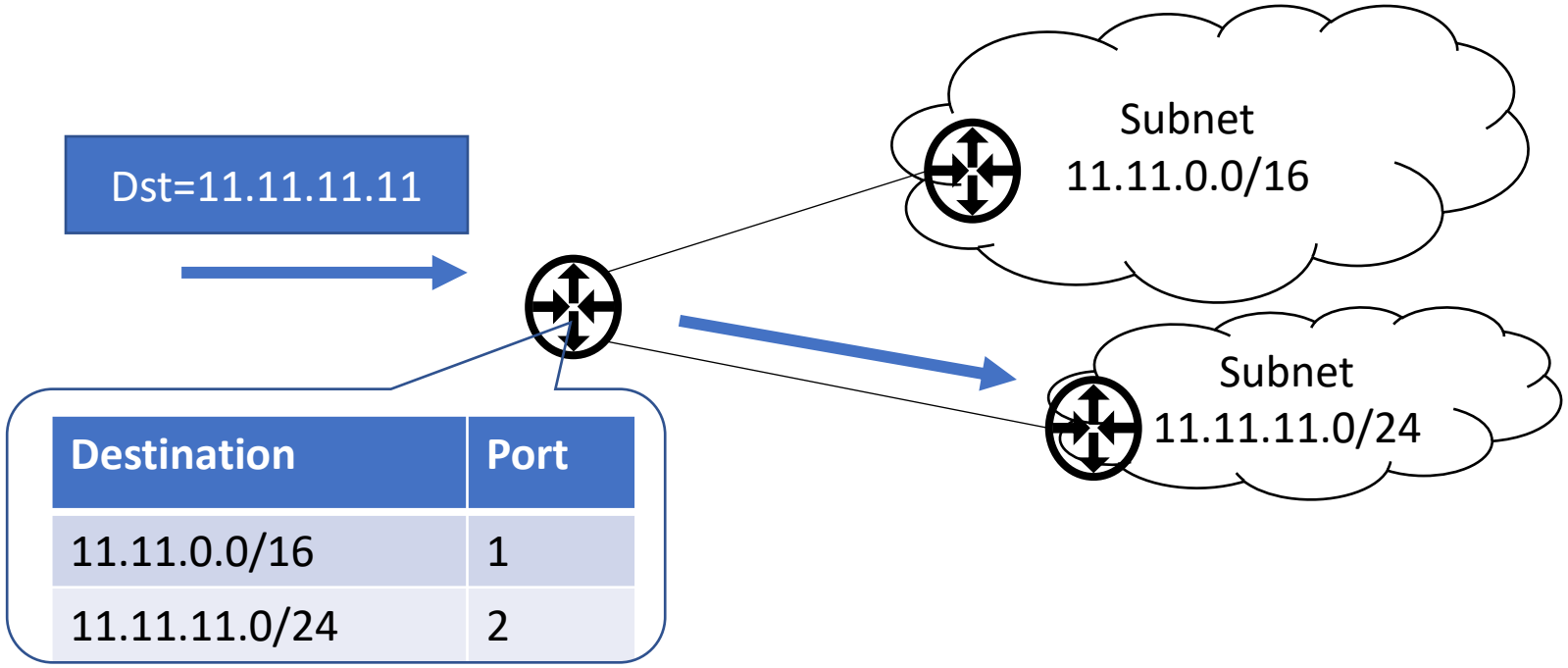
Address Structure Adds Scalability

- IP Prefix is of the form a.b.c.d / z (e.g., 2.1.1.0/24)
  z indicated how many upper order bits uniquely specify the group

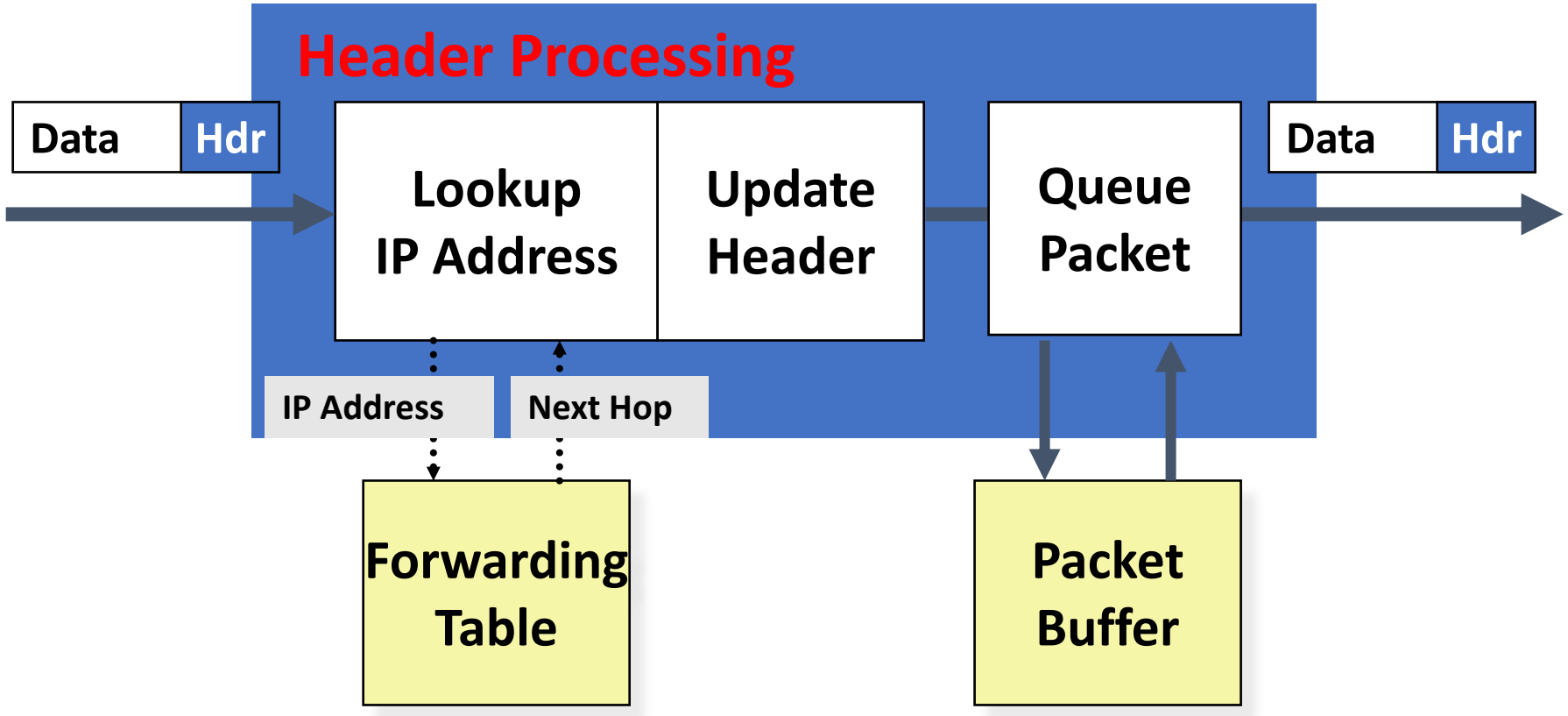


Longest Prefix Match (LPM)

- Find the most specific prefix that matches the destination



Generic Router Architecture



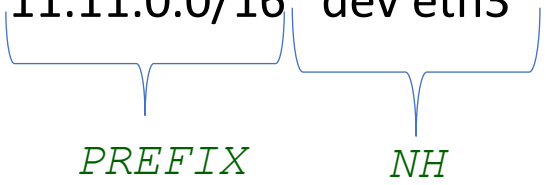
Slide from Princeton COS 461 (Mike Freedman, Jen Rexford, Nick Feamster)

ip route
Utility for Linux forwarding table management.

https://man7.org/linux/man-pages/man8/ip-route.8.html

```
ip route { add | del | change | append | replace } ROUTE

SELECTOR := [ root PREFIX ] [ match PREFIX ] [ exact PREFIX ] [ table TABLE_ID ]
[ vrf NAME ] [ proto RTPROTO ] [ type TYPE ] [ scope SCOPE ]

ROUTE := NODE_SPEC [ INFO_SPEC ]

NODE_SPEC := [ TYPE ] PREFIX [ tos TOS ] [ table TABLE_ID ] [ proto RTPROTO ]
[ scope SCOPE ] [ metric METRIC ] [ ttl- propagate { enabled | disabled } ]

INFO_SPEC := { NH | nhid ID } OPTIONS FLAGS [ nexthop NH ] ...

NH := [ encap ENCAP ] [ via [ FAMILY ] ADDRESS ] [ dev STRING ] [ weight NUMBER ] NHFLAGS
```

ip route add 11.11.0.0/16 dev eth3

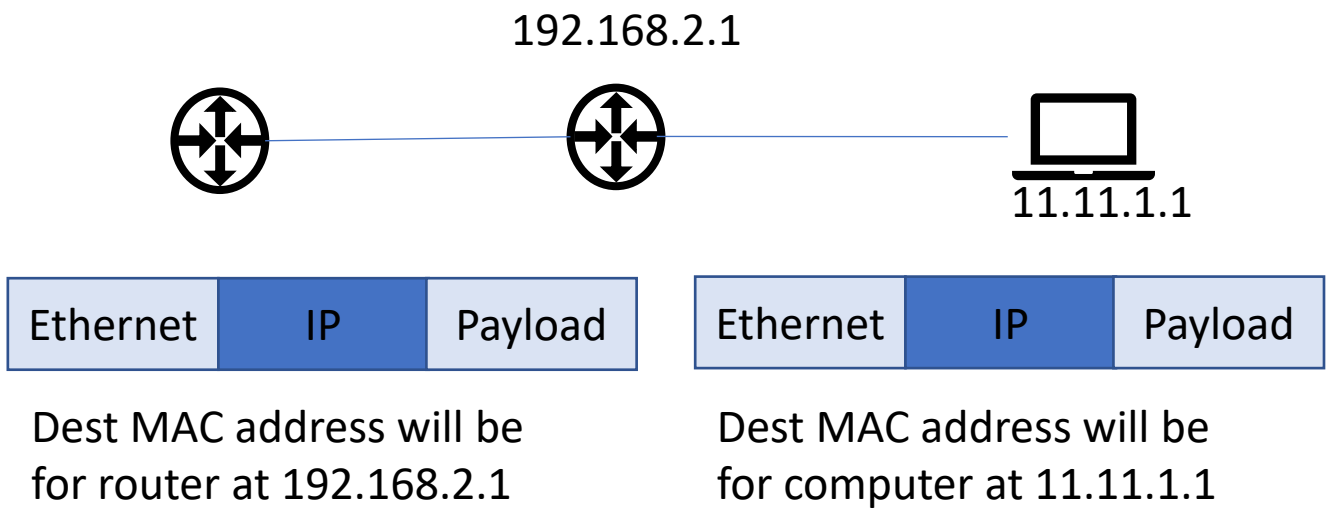


What does via do?

ip route add 11.11.0.0/16 via 192.168.2.1 dev eth3

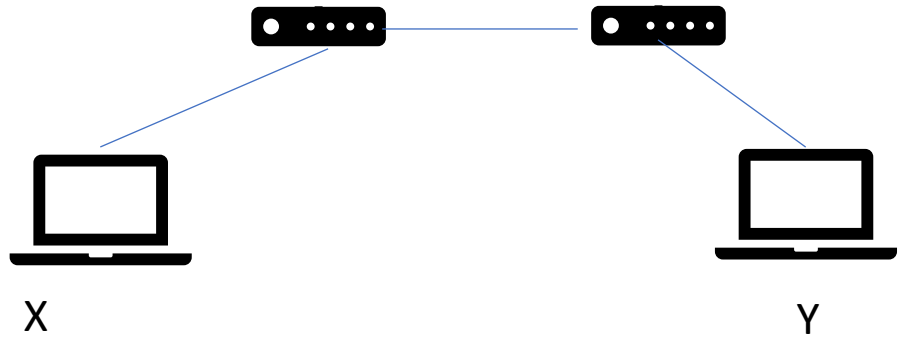
vs

ip route add 11.11.0.0/16 dev eth3

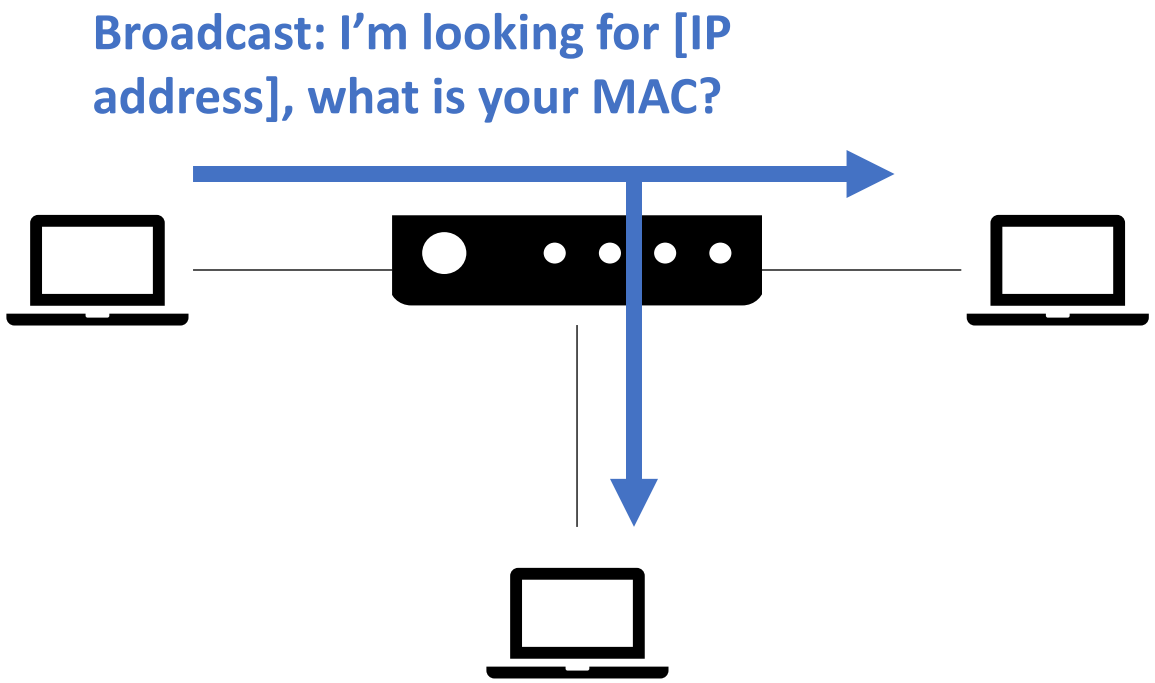


Mapping IP to MAC

- X wants to talk to Y on a LAN
- X more likely knows IP address of Y
  - Name of server (we'll cover DNS)
  - Consistent communication in larger network
- But, link layer comm is done with MAC addresses

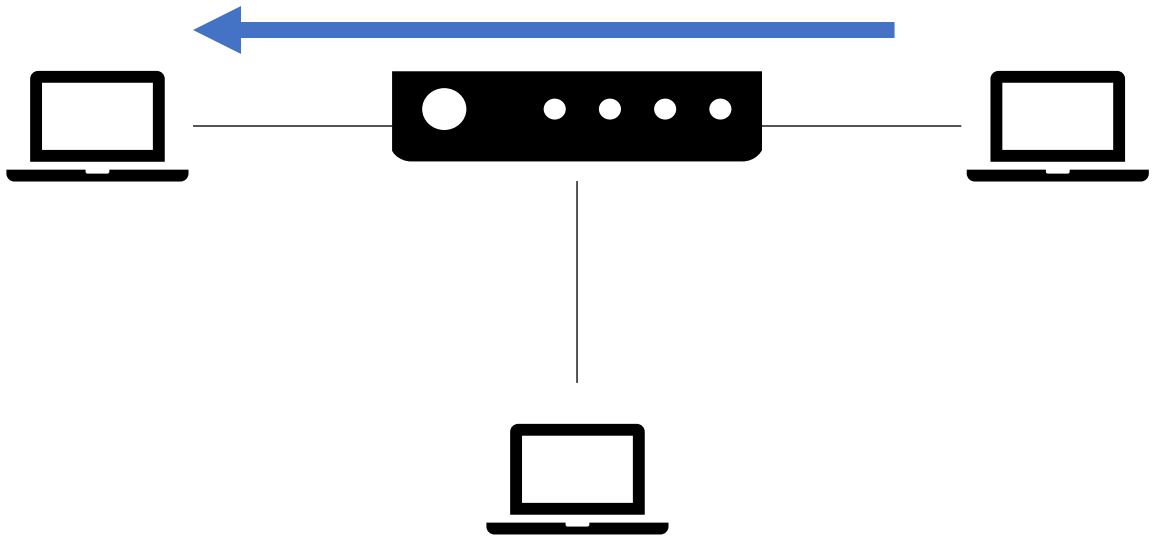


ARP – address resolution protocol

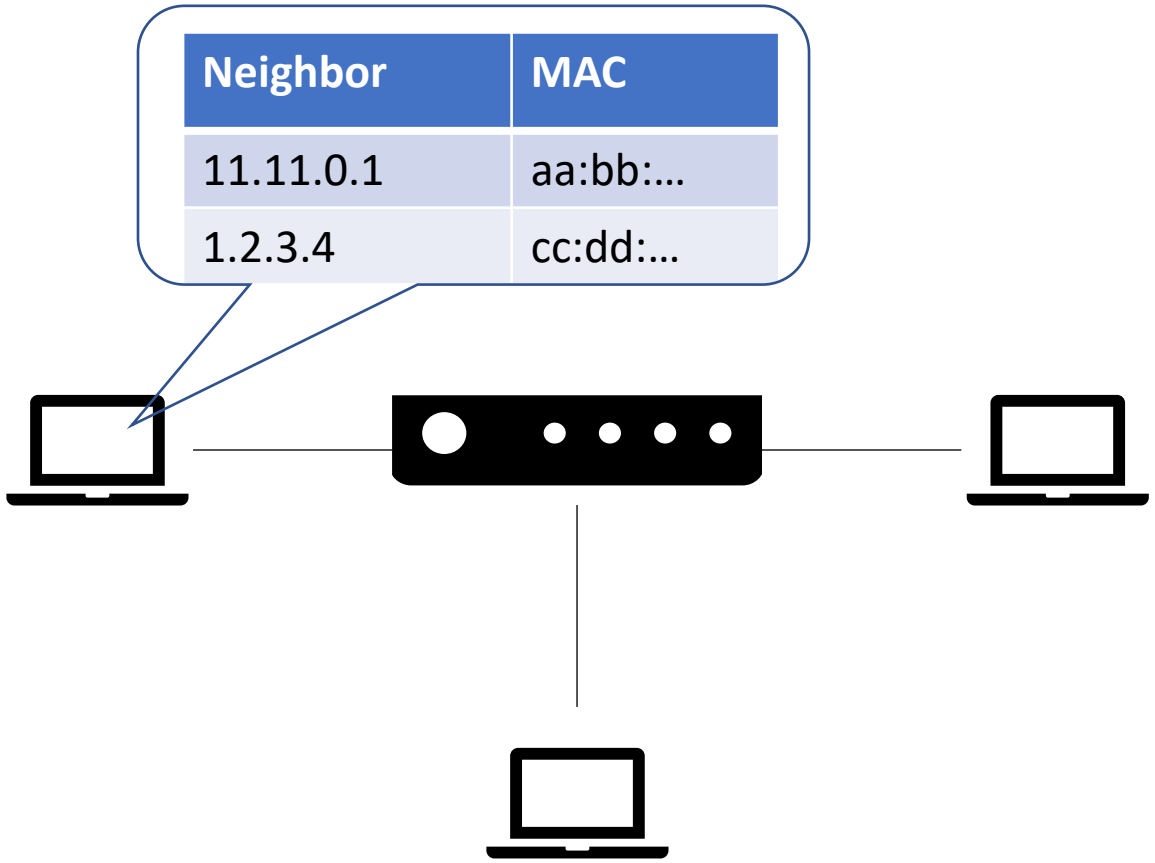


ARP – address resolution protocol

That's me. My MAC is [MAC Addr]



ARP / Neighbor / MAC Address Table



ip neigh
neighbor / arp table management utility for Linux

- https://man7.org/linux/man-pages/man8/ip-neighbour.8.html

```
[tecmint@tecmint]$ sudo ip neigh add 172.19.1.0 lladdr 02:42:e3:40:a6:b1 dev eth2 nud stale
[tecmint@tecmint]$ sudo ip neigh add 172.19.2.0 lladdr 02:42:e3:40:a6:b2 dev eth2 nud stale
[tecmint@tecmint]$ sudo ip neigh add 172.19.3.0 lladdr 02:42:e3:40:a6:b3 dev eth2 nud stale
[tecmint@tecmint]$
[tecmint@tecmint]$ ip neigh show
172.19.3.0 dev eth2 lladdr 02:42:e3:40:a6:b3 STALE
172.19.1.0 dev eth2 lladdr 02:42:e3:40:a6:b1 STALE
172.19.2.0 dev eth2 lladdr 02:42:e3:40:a6:b2 STALE
[tecmint@tecmint]$
[tecmint@tecmint]$ sudo ip neigh flush all
[tecmint@tecmint]$
[tecmint@tecmint]$ ip neigh show
[tecmint@tecmint]$
```

https://www.tecmint.com/ip-command-examples/

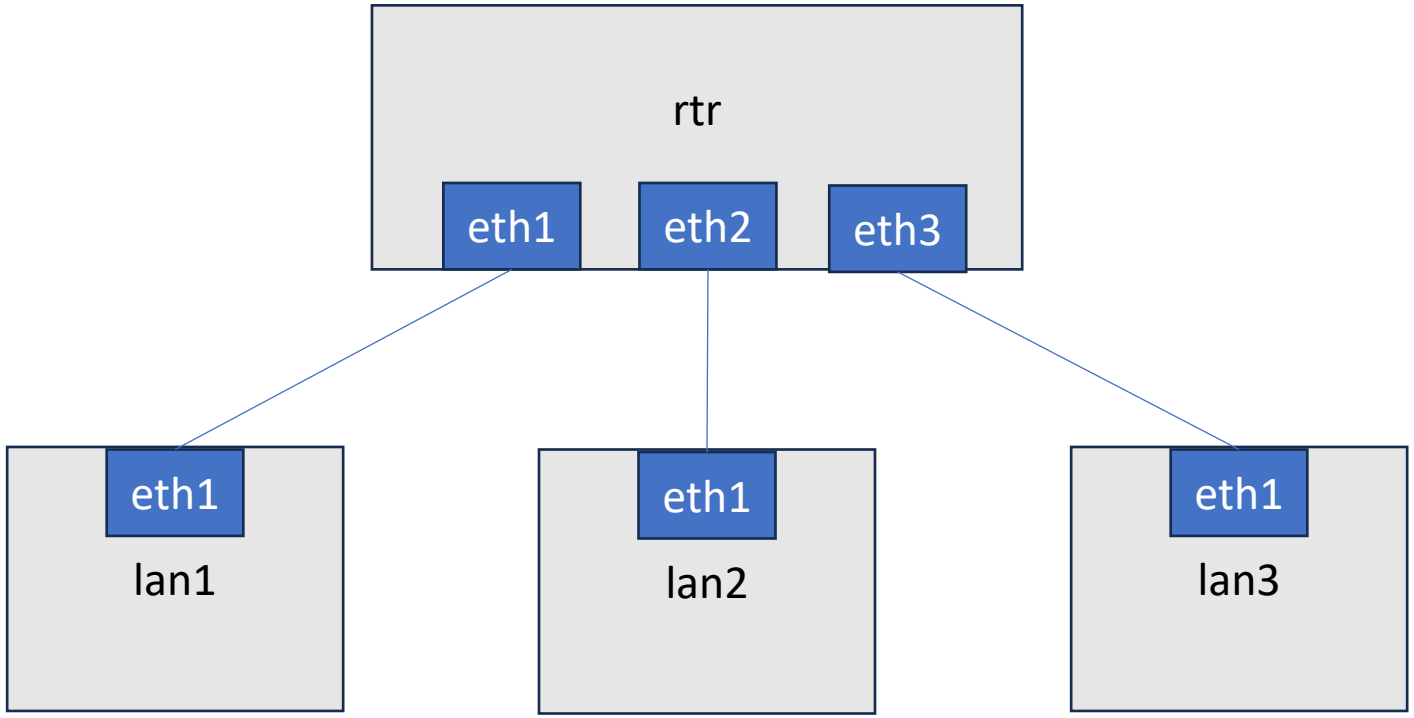
University of Colorado **Boulder**

Example IP Layer Walkthroughs

Course: Networking Principles in Practice – Linux Networking
Module: IP Layer with Linux Networking

University of Colorado **Boulder**

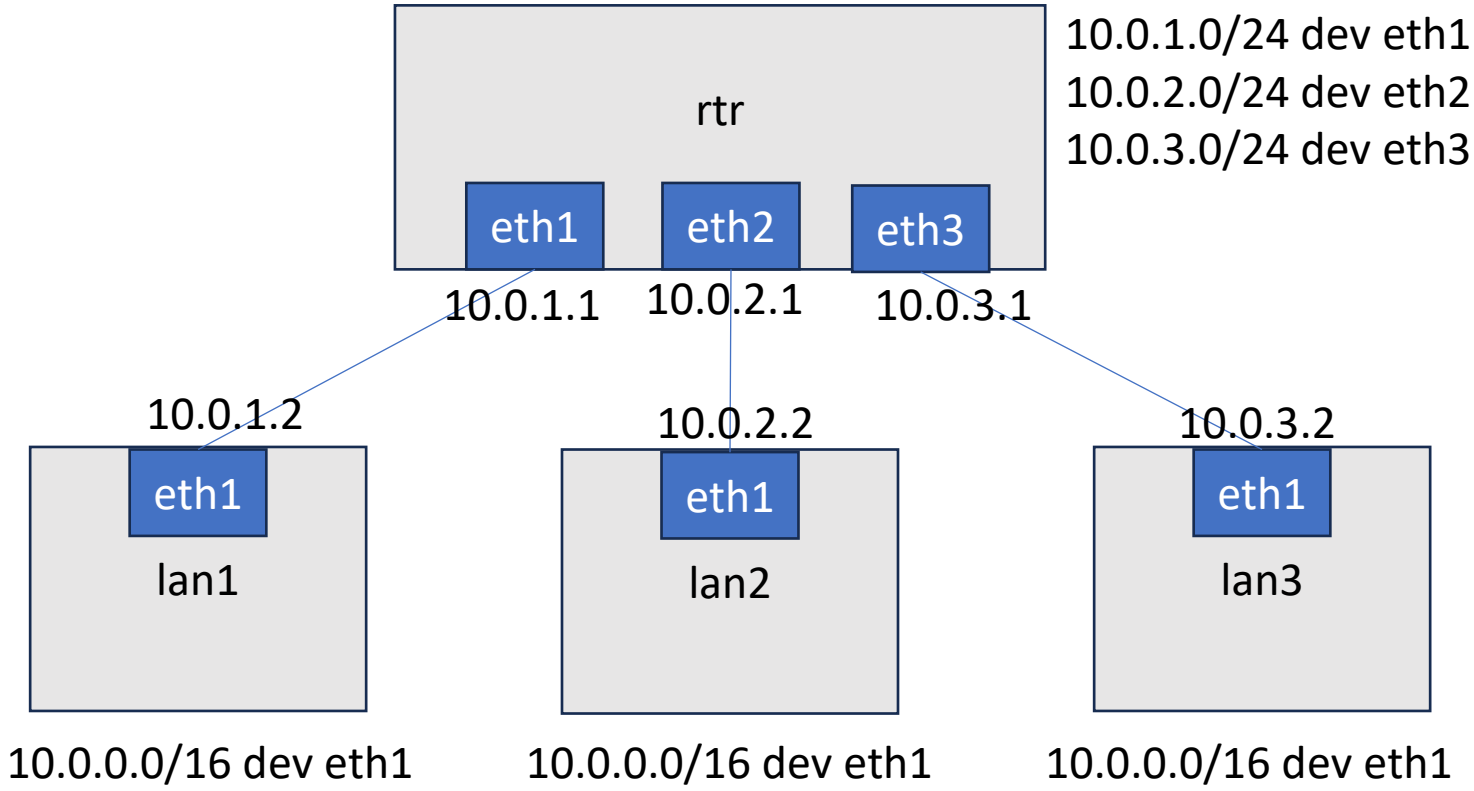
Lab Setup - 3lan-mod2.clab.yml



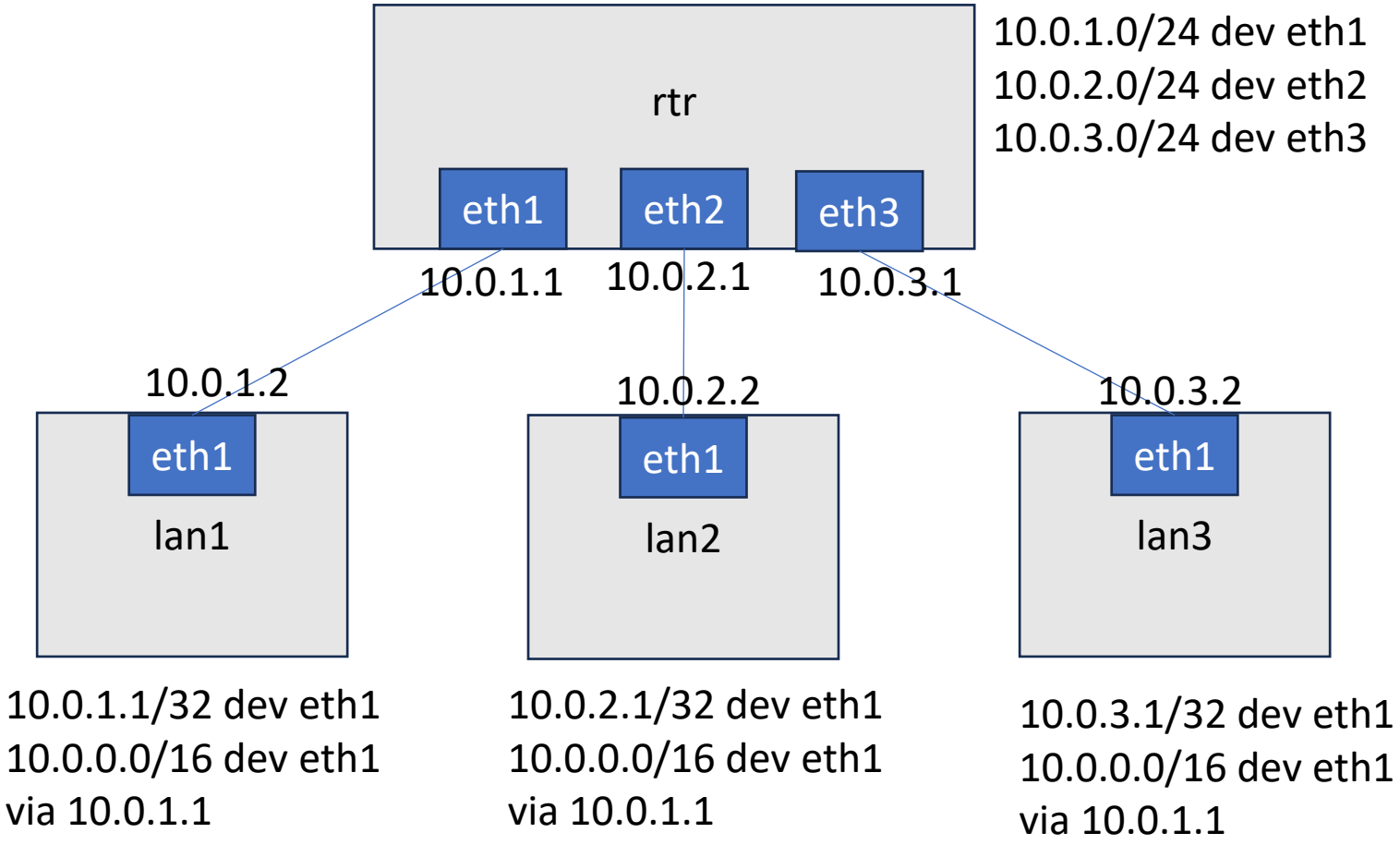
Show

- Container Lab config
- setup-ip-ex1.sh – not using via
- setup-ip-ex2.sh – using via
- setup-ip-ex3.sh – multiple IP addresses per interface
- setup-ip-ex4.sh – GRE Tunnel

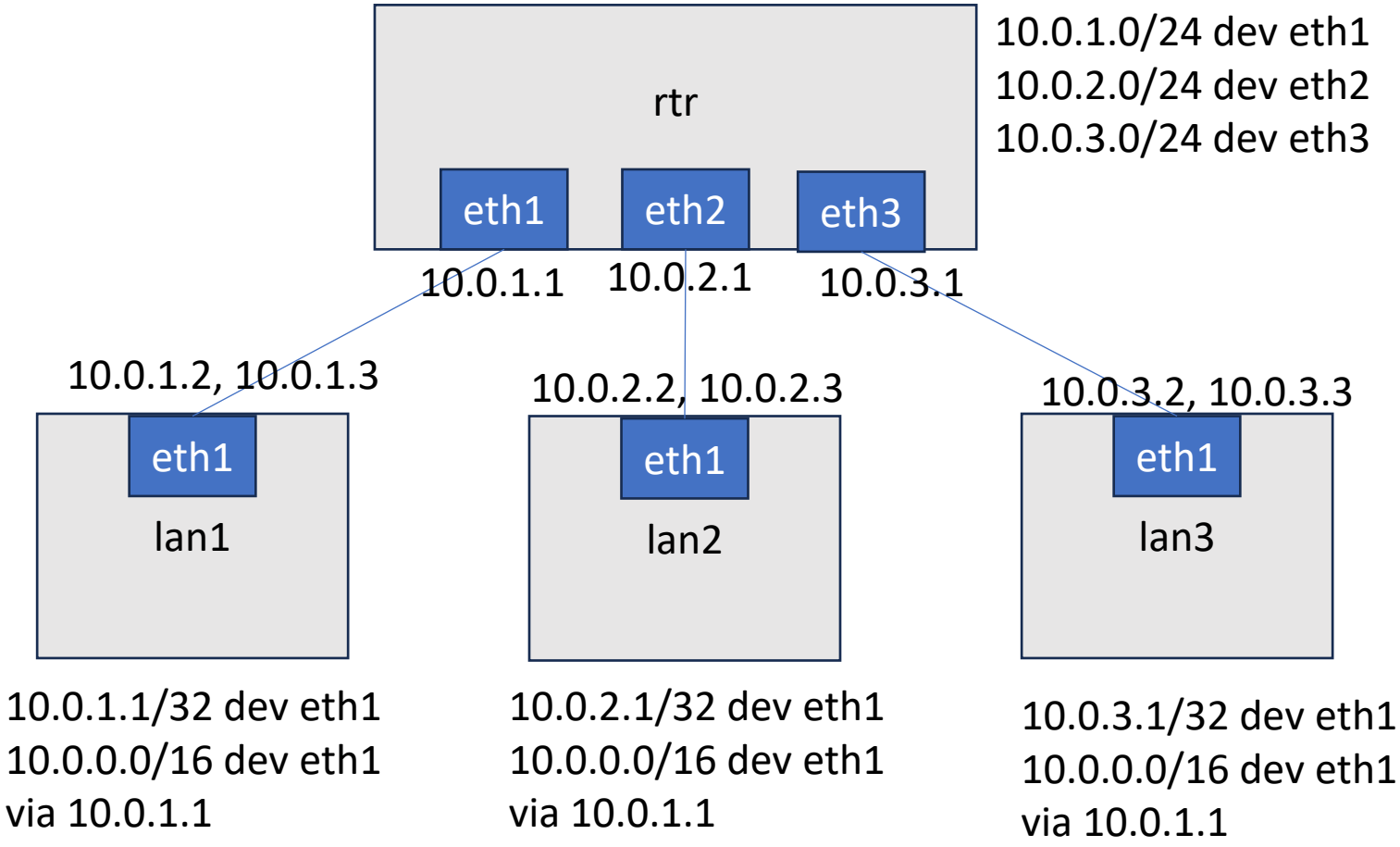
Lab Setup - setup-ip-ex1.sh – not using via



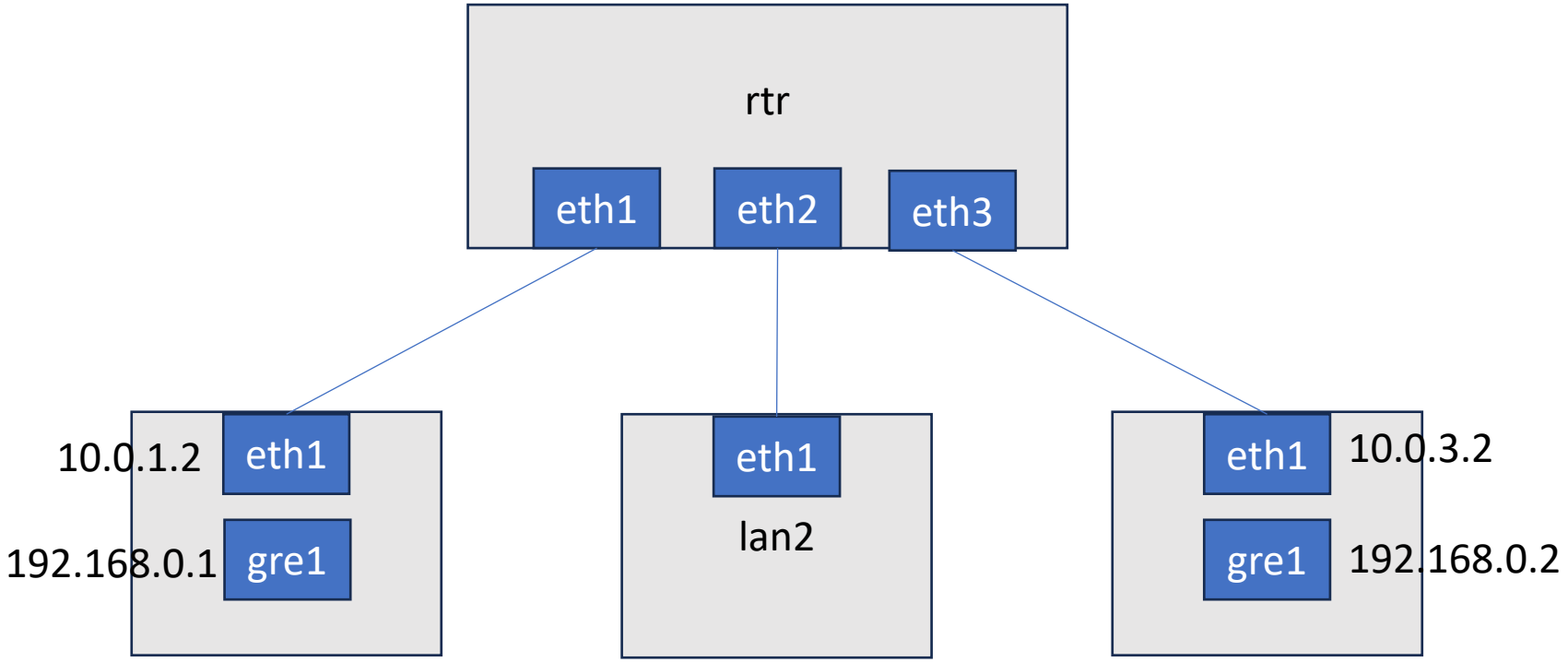
Lab Setup - setup-ip-ex2.sh –using via



Lab Setup - setup-ip-ex3.sh – multiple addrs



GRE



ip tunnel add gre1 mode gre local 10.0.1.2 remote 10.0.3.2 ttl 255

ip addr add 192.168.0.1/30 dev gre1

ping 192.168.0.2 – routing table will say that goes out dev gre1
[IP src=192.168.0.1, dest=192.168.0.2][ICMP Echo]

Gre1 device will tunnel it: local (10.0.1.2) remote (10.0.3.2)
[IP src=10.0.1.2, dst=10.0.3.2][GRE][IP src=192.168.0.1, dest=192.168.0.2][ICMP Echo]

Routing table will say dest 10.0.3.2 goes out eth1

University of Colorado **Boulder**

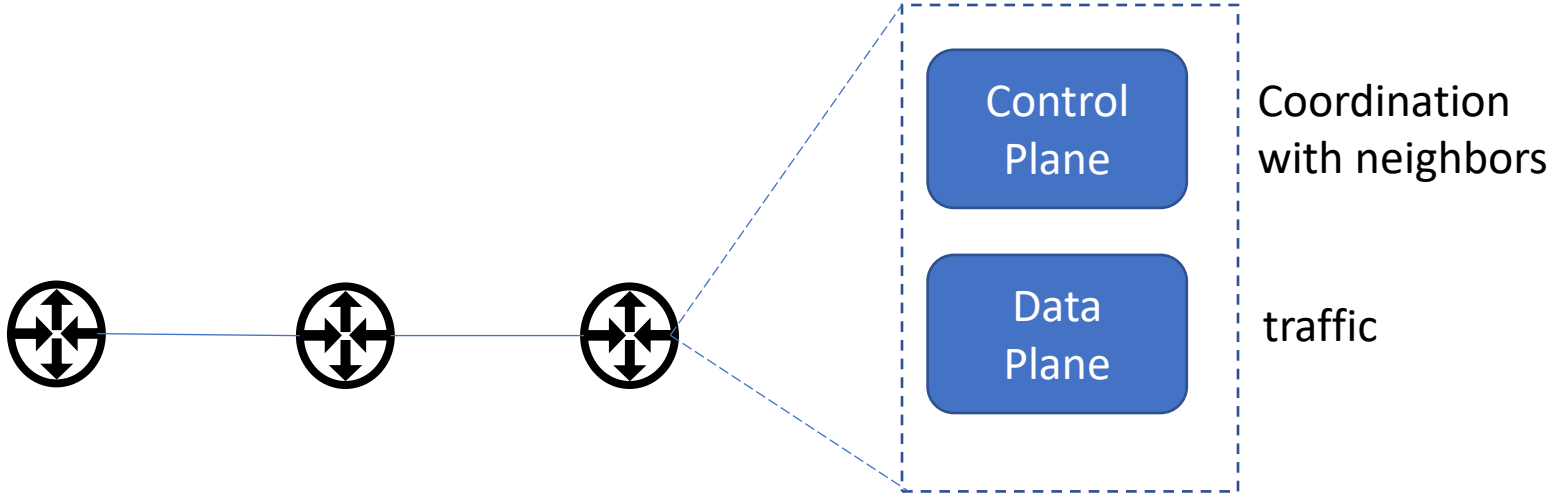
Routing in Linux

Course: Networking Principles in Practice – Linux Networking
Module: IP Layer with Linux Networking

University of Colorado **Boulder**

Forwarding vs Routing

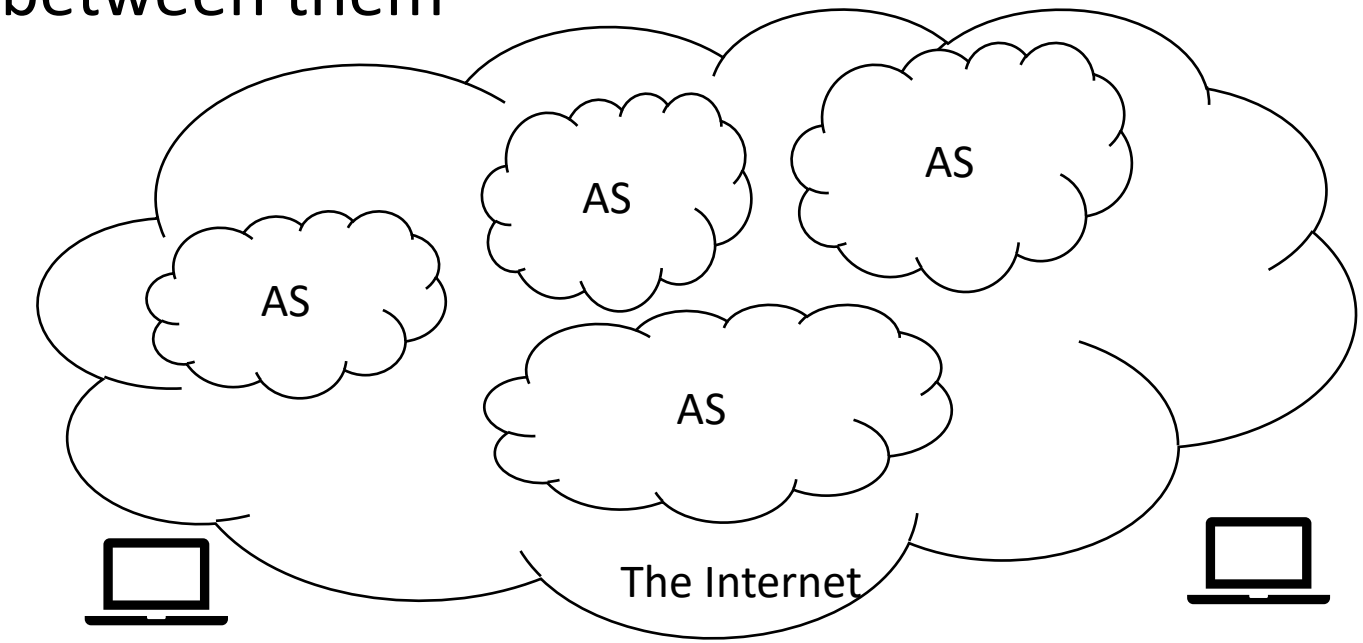
- Forwarding: Data plane
  - Direct a data packet to an output port/link
  - Uses a forwarding table
- Routing: Control plane
  - Computes paths by coordinating with neighbors
  - Creates the forwarding table



BGP Refresher: Overview

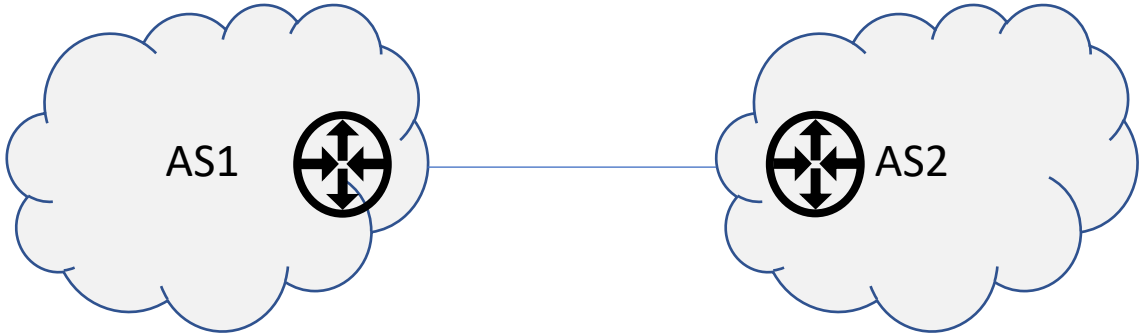
Internet is many inter-connected networks called Autonomous Systems

Border Gateway Protocol (BGP) is the protocol used to coordinate between them

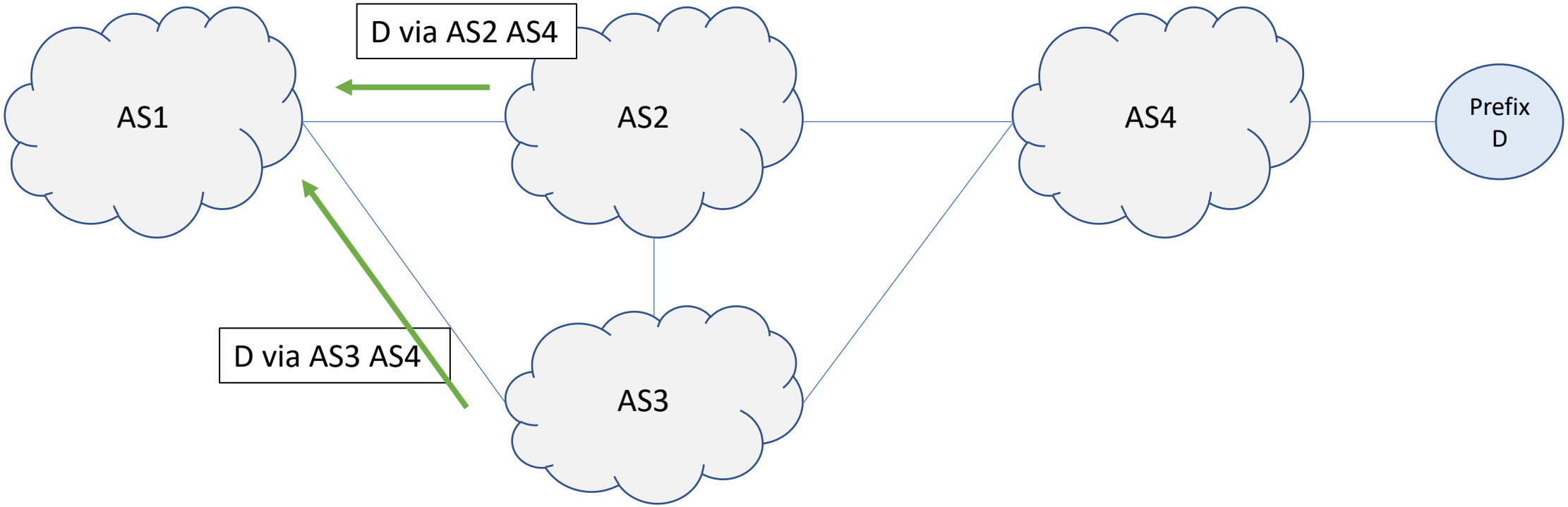


BGP Refresher: Peering

- Routers will “peer” with a neighbor
  - Establish a TCP connection
  - Establish some properties about each other (AS Number, capabilities, etc.)
  - State machine leads to the “Established” state once peered



BGP Refresher: Message Exchange



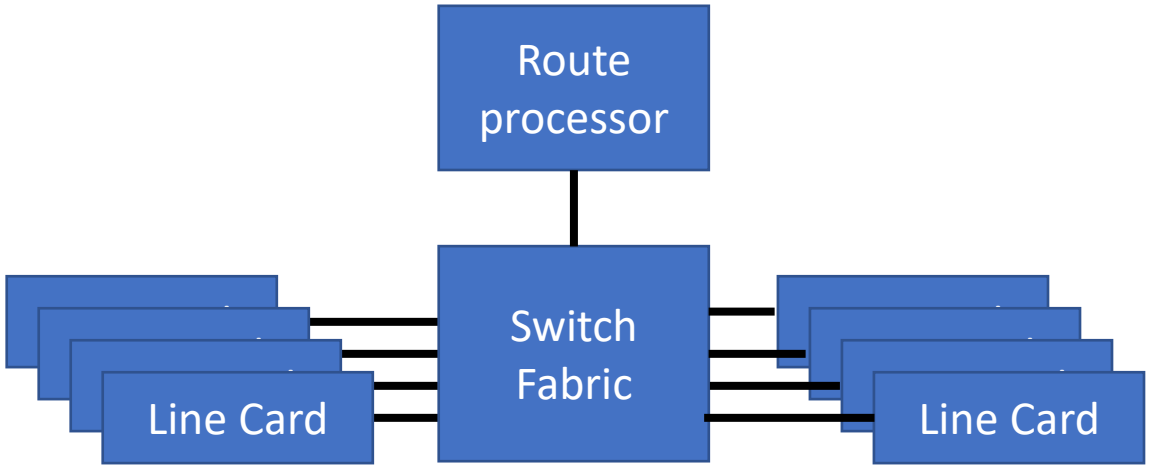
- Announcements: tells neighbors of the availability of a path to some destination
- Withdrawals: tells neighbors route no longer available (e.g., upon failure)

BGP Refresher: Route Decision

- Routers learn about multiple routes for the same prefix, but choose 1 (to set in the forwarding table, and to announce to neighbors)

Local preference	numerical value assigned by routing policy.
AS path length	number of AS-level hops in the path
Multiple exit discriminator (MED)	allows one AS to specify that one exit point preferred
eBGP over iBGP	Learned through external neighbor over internal
Shortest IGP path cost	Exit this network as quickly as possible
Router ID	arbitrary tiebreaker

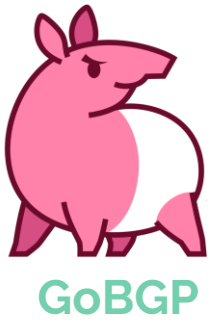
BGP Refresher RIB / FIB



- Control Plane
  - Runs routing protocols (BGP)
  - Routing table (RIB)
- Data Plane
  - Forwards packets
  - Forwarding table (FIB)

Linux Routing Software

- Quagga (https://www.nongnu.org/quagga/)
- Bird (https://bird.network.cz/)
- FRR (https://frrouting.org/)
- GoBGP (https://osrg.github.io/gobgp/)
- …



Bird configuration (/etc/bird/bird.conf)

- Protocol blocks related to Linux
- Protocol block - BGP
- Import / Export Filters
- Templates
- Tables
- Logging
- Debug

Protocol blocks related to Linux

- Serves as a module for getting information about network interfaces from the kernel

```
protocol device {
}
```

- Synchronizes BIRD tables with the OS kernel. One instance per table

```
protocol kernel {
        ipv4 {                  # Connect protocol to IPv4 table by channel
#               table master4;  # Default IPv4 table is master4
#               import all;     # Import to table, default is import all
                export all;     # Export to protocol. default is export none
        };
#       kernel table 10;        # Kernel table to synchronize with (default: main)
}
```

Protocol Block – BGP (setting peering)

```
protocol bgp uplink1 {
        description "My BGP uplink";
        local 198.51.100.1 as 65000;
        neighbor 198.51.100.10 as 64496;
        hold time 90;                   # Default is 240
        password "secret";# Password used for MD5 authentication
…
```

Protocol Block – BGP Filters

Add to protocol block:

```
protocol bgp uplink1 {
        … (peering)…
        ipv4 {                  # regular IPv4 unicast (1/1)
                import filter rt_import;
                export where source ~ [ RTS_STATIC, RTS_BGP ];
        };
…
```

Protocol Block – BGP Filters

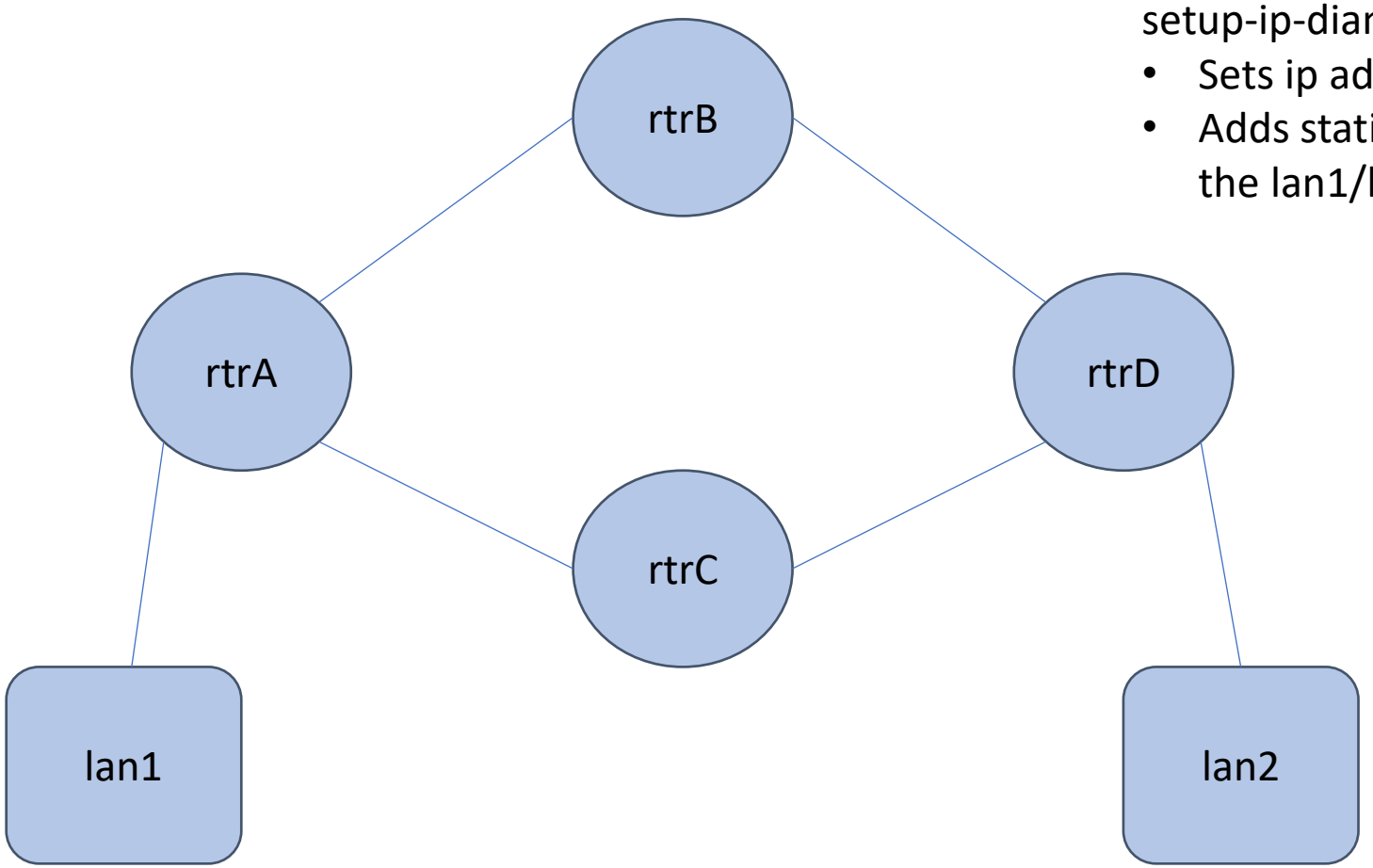
And define filter (note: can use functions as well):

```
filter rt_import
 {
        if bgp_path.first != 64496 then accept;
        if bgp_path.len > 64 then accept;
        if bgp_next_hop != from then accept;
        reject;
}
```

Other configuration

- Templates – can specify a protocol block as template, then extend it (useful if all the configuration is the same, except for, e.g., IP address)
- Tables – there exist default tables, but you can specify (at the top) and reference (in a protocol block) custom tables.
- Logging – indicate what logging level you'd like, and where to output
- Debug – turn on/off debug for individual protocols

(future lesson) Running bird



setup-ip-diamond.sh

- Sets ip addresses for each interface
- Adds static routes on rtrA and rtrD to the lan1/lan2

make_aliases.sh

* Sets aliases for “docker exec …”

diamond-mod2.clab.yml

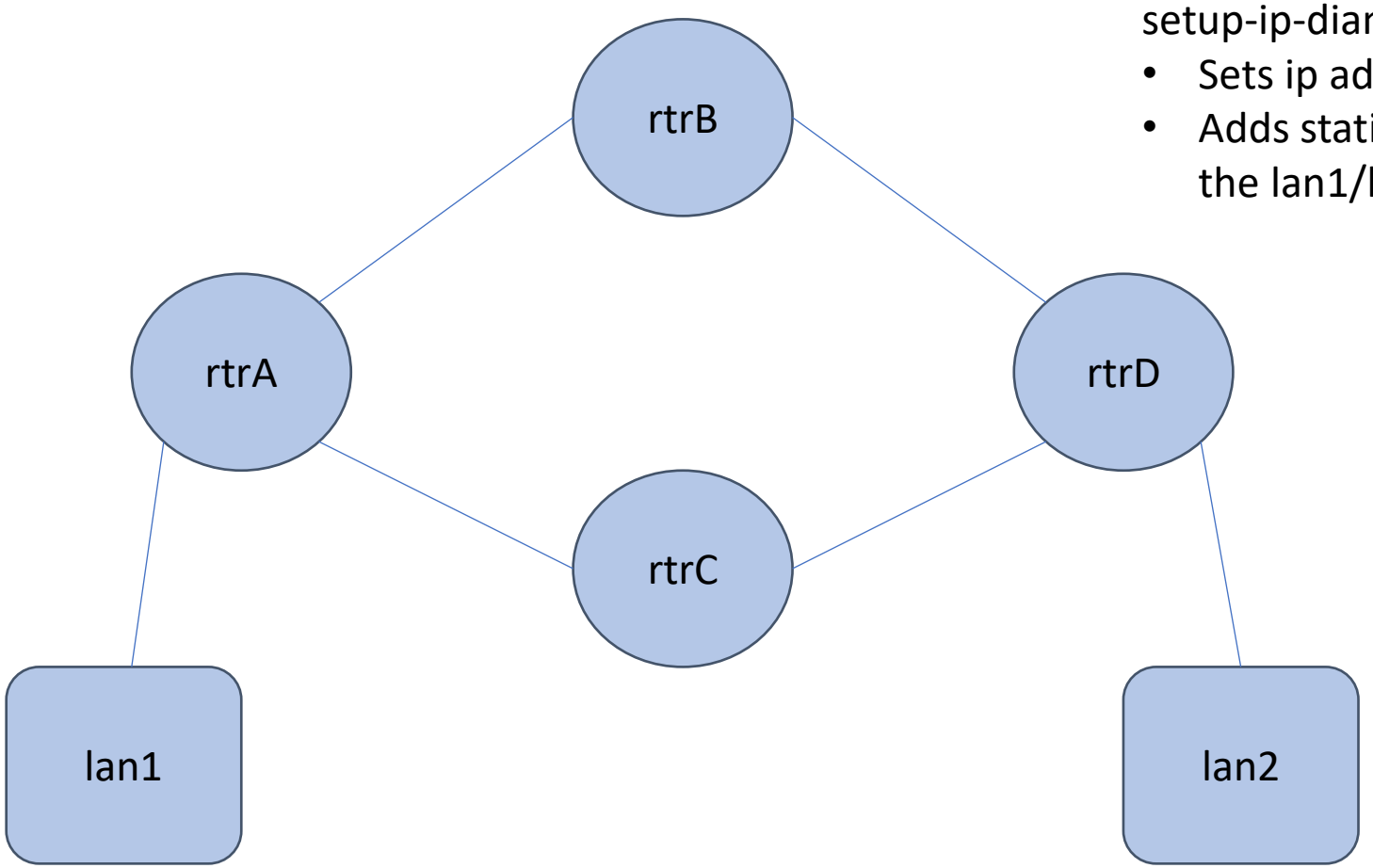
University of Colorado **Boulder**

Routing Walkthrough with Bird

Course: Networking Principles in Practice – Linux Networking
Module: IP Layer with Linux Networking

University of Colorado **Boulder**

Topology - Standard



setup-ip-diamond.sh

- Sets ip addresses for each interface
- Adds static routes on rtrA and rtrD to the lan1/lan2

make_aliases.sh

* Sets aliases for “docker exec …”

diamond-mod2.clab.yml

Configuration / Operation

For each container mount ./mod2-bird-confs to /etc/bird

- rtrA-bird.conf
- Note: BGP config. Note: Kernel protocol (will pick up static routes)

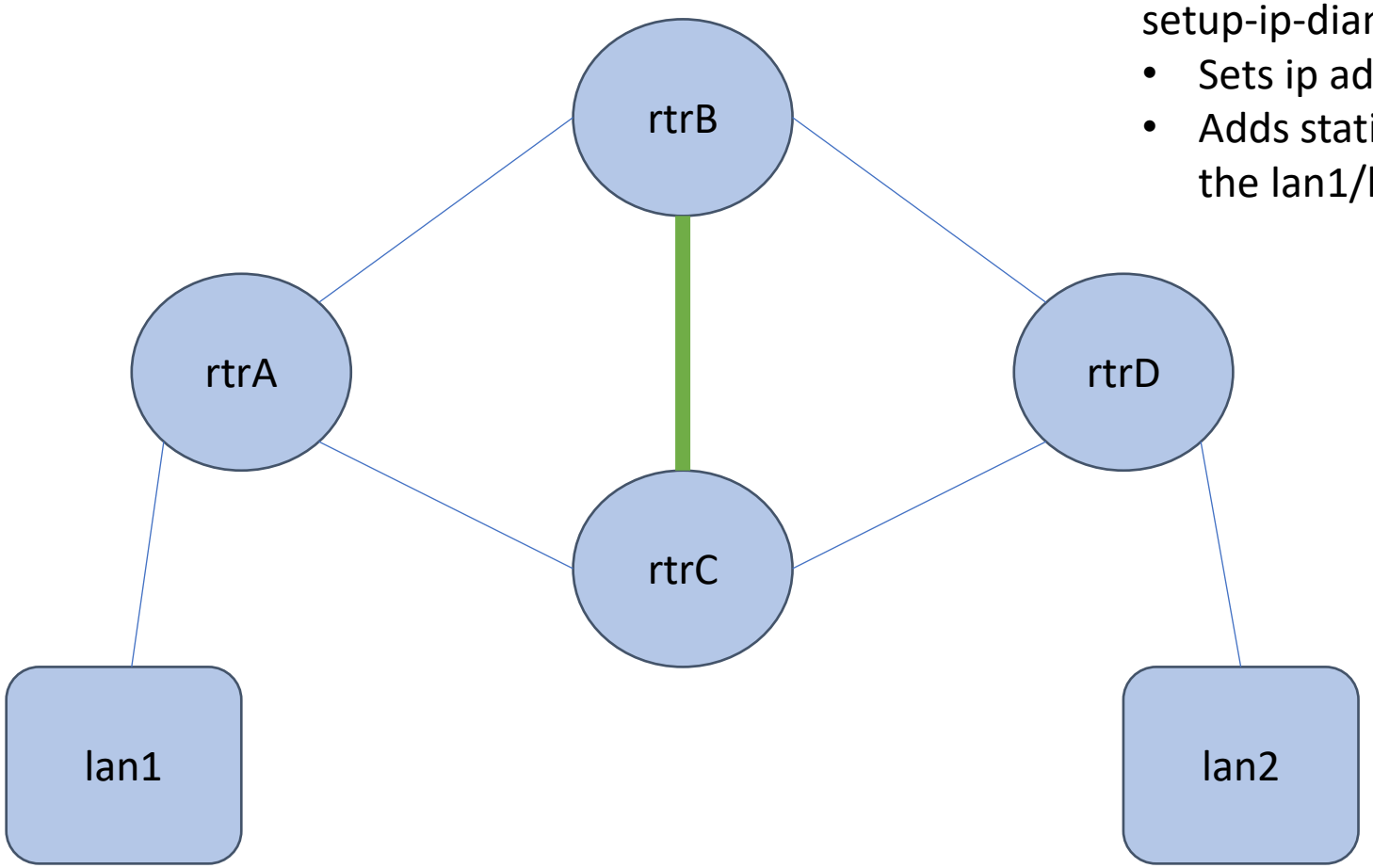
start-bird.sh

- bird -c /etc/bird-alt/rtrA-bird.conf

birdc – utility (in the container) to interact with bird

- birdc show protocols
- birdc show route all

Lab1 – step 1 (add peering)



setup-ip-diamond.sh

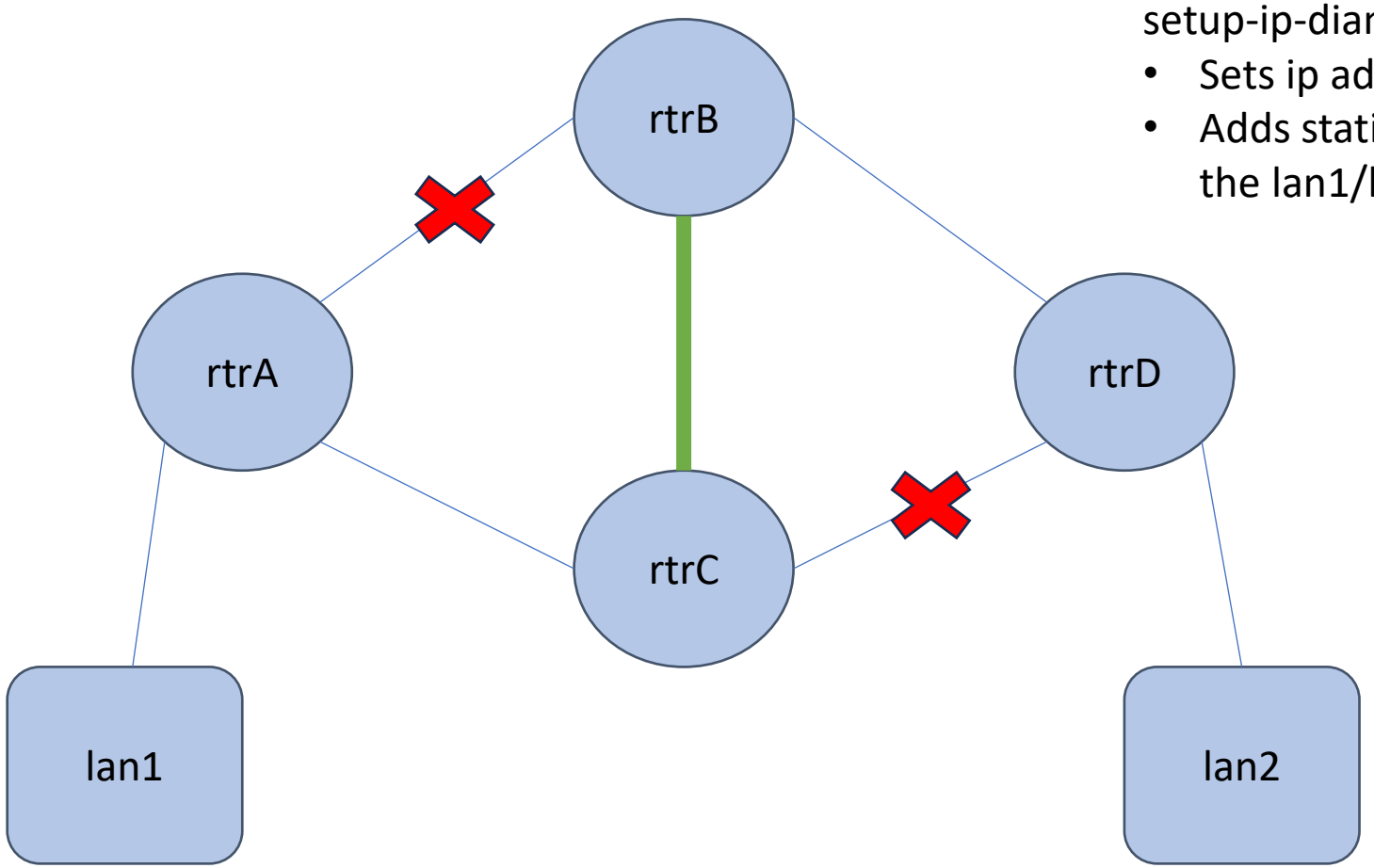
- Sets ip addresses for each interface
- Adds static routes on rtrA and rtrD to the lan1/lan2

make_aliases.sh

* Sets aliases for “docker exec …”

diamond-mod2.clab.yml

Lab1 – step 2 (fail links)



setup-ip-diamond.sh

- Sets ip addresses for each interface
- Adds static routes on rtrA and rtrD to the lan1/lan2

make_aliases.sh

* Sets aliases for “docker exec …”

diamond-mod2.clab.yml

University of Colorado **Boulder**

Larger Routing Experimentation

Course: Networking Principles in Practice – Linux Networking
Module: IP Layer with Linux Networking

University of Colorado **Boulder**

Larger Experiments

- How do we get real routing tables and announcements?
- How can we use those to inject announcements in a programmatic manner?
- Containerlab is nice, but it’s pretty manual… can we get something with richer capabilities?

Larger Experiments

- How do we get real routing tables and announcements?
  - RIPE - Routing data collector
- How can we use those to inject announcements in a programmatic manner?
  - ExaBGP
- Containerlab is nice, but it’s pretty manual… can we get something with richer capabilities?
  - SEED Internet Emulator

RIPE (https://www.ripe.net/)

- RIPE NCC is the regional Internet registry for Europe, the Middle East and parts of Central Asia.
- RIS is a routing data collection platform. https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris

RIPE (https://www.ripe.net/)

- Example: Routing Table Dump
- wget https://data.ris.ripe.net/rrc03/2023.08/bview.20230803.0800.gz
  - stored in MRT format
  - These files can be read using bgpdump (https://github.com/RIPE-NCC/bgpdump)

```
docker run -v $(pwd)/dump:/dump fusl/ripencc-bgpdump  -m -O
/dump/bview-out /dump/bview.20230803.0800
```

TABLE_DUMP2|1691049600|B|80.249.211.217|8455|**33.0.0.0/8**|**8455 3356 749**|
IGP|80.249.211.217|0|0|8455:5998|NAG||

ExaBGP (https://github.com/Exa-Networks/exabgp)

- BGP agent meant for testing - has high programmability
- Can Add static routes in the configuration WITH AS Paths

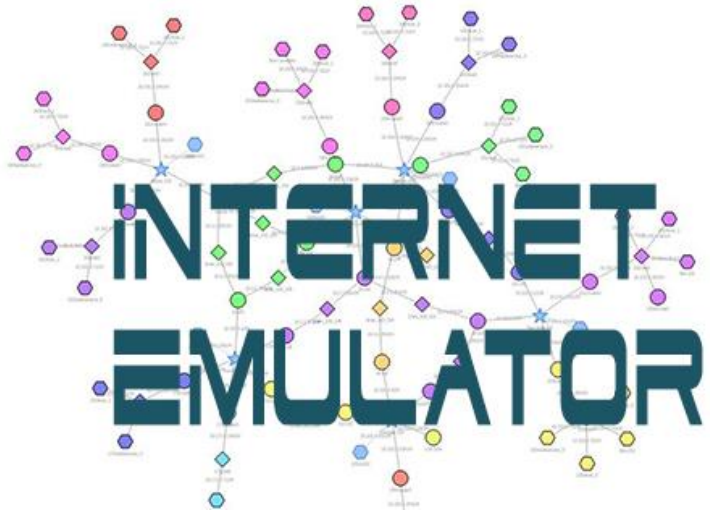
```
neighbor 10.10.0.1 {
   router-id 10.100.0.2;
   local-address 10.10.0.2;
   local-as 150;
   peer-as 65000;
   static {
      route 7.0.0.0/8 next-hop 10.10.0.2 as-path [ 49432 48362 9002 3356 749 ];
      route 11.0.0.0/8 next-hop 10.10.0.2 as-path [ 49432 48362 9002 3356 749 ];
      …
```

ExaBGP - Python programmable BGP agent

API:
exabgpcli neighbor 1.2.3.4 announce route 10.0.0.1/24 next-hop self extended-community [123456:666]

```
#!/usr/bin/env python
import sys
import time
messages = [
    'announce route 1.1.0.0/24 next-hop 101.1.101.1’,
    'announce route 1.1.0.0/25 next-hop 101.1.101.1’,
    'withdraw route 1.1.0.0/24 next-hop 101.1.101.1', ]

time.sleep(2)
while messages:
   message = messages.pop(0)
    …
```

SEED Internet Emulator

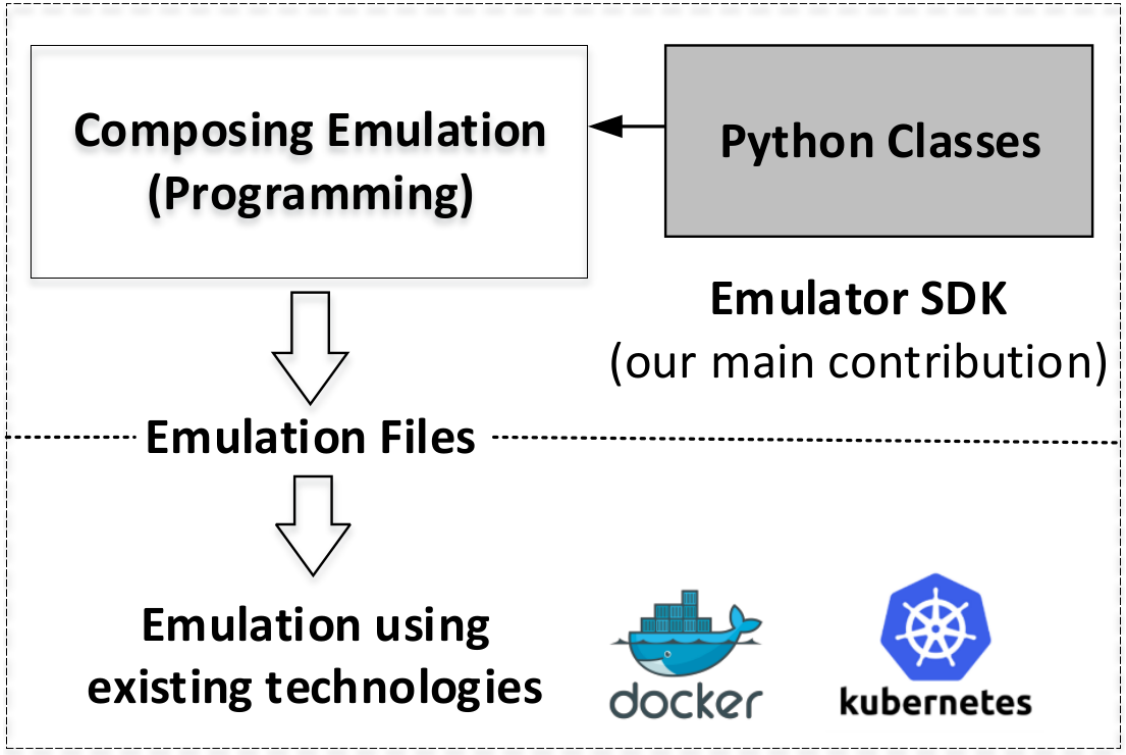


- **Founders**
  - Kevin Du
  - Honghao Zeng (MS student)
- **History**
  - 2018 – 2020: Investigation & Design
  - August 2020: Implementation
  - July 2021: First release

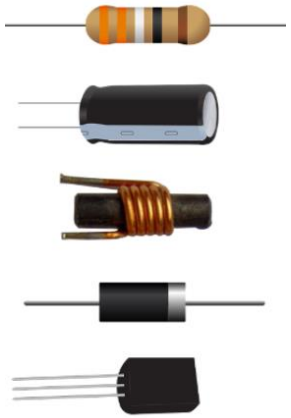
https://github.com/seed-labs/seed-emulator

Slide from: https://seedsecuritylabs.org/emulator/emulator_01_2022.pptx

Design



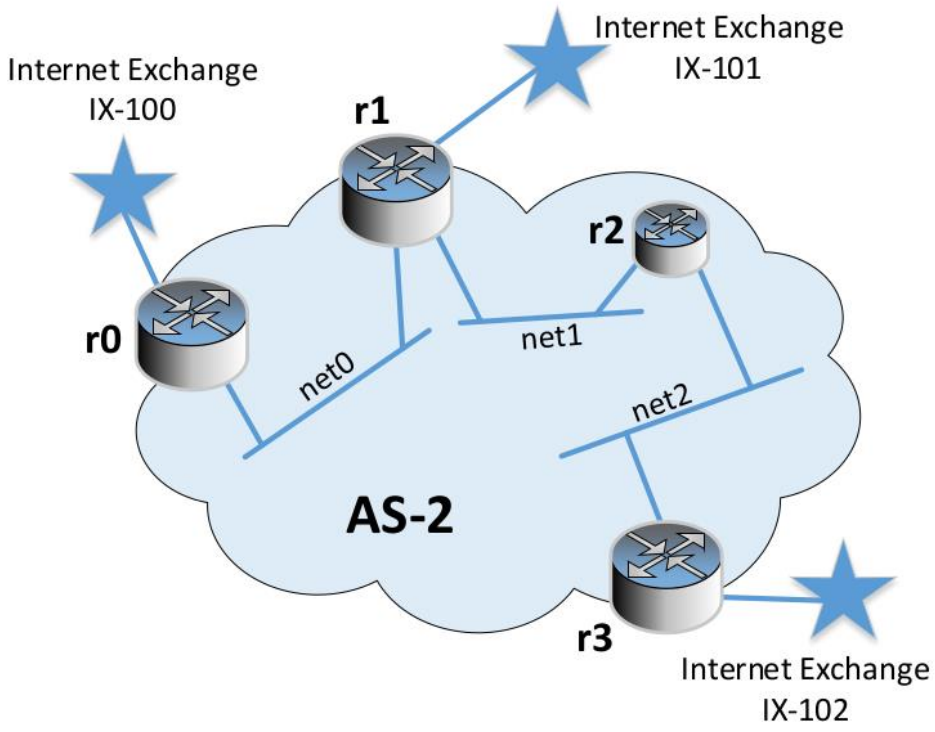
Primitives (Classes)



- Autonomous System
- Internet Exchange
- Network
- Router, BGP speaker
- Host
- Service
- etc.

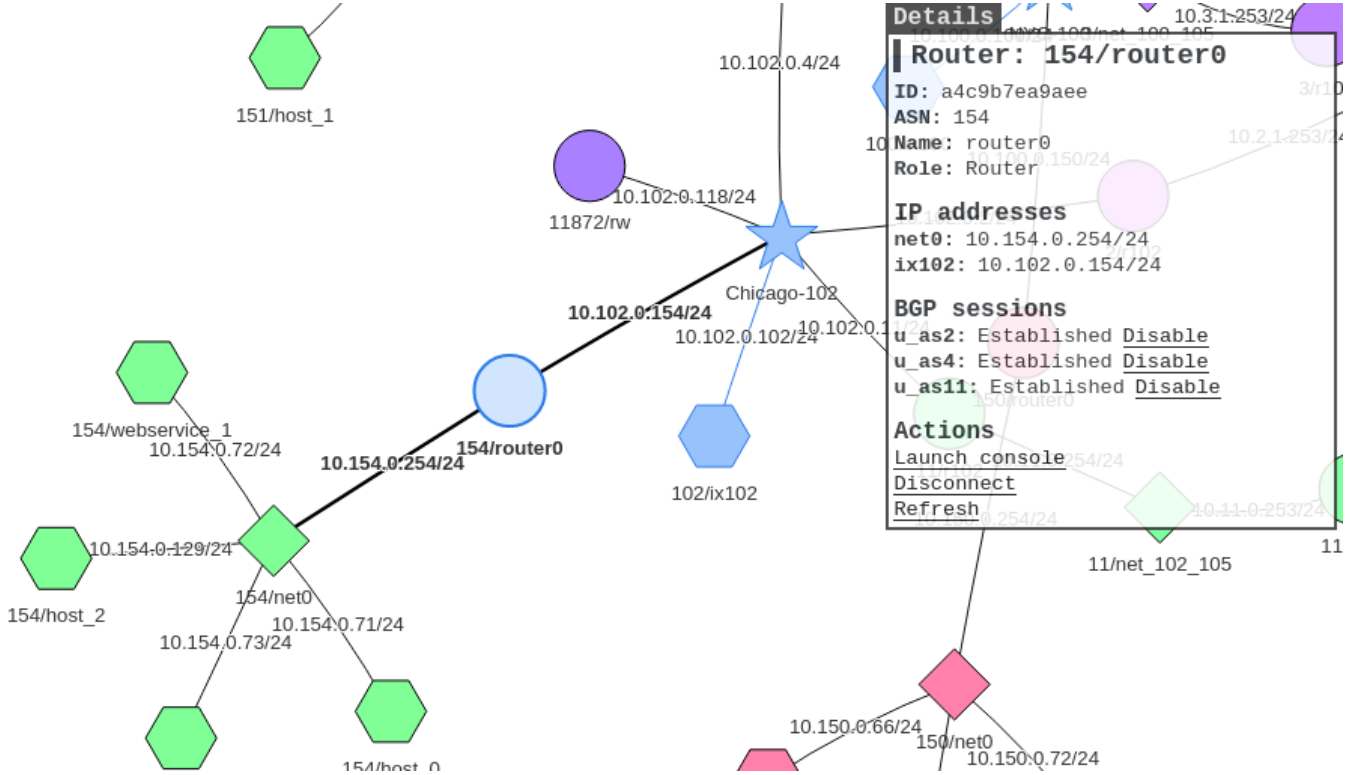
Slide from: https://seedsecuritylabs.org/emulator/emulator_01_2022.pptx

Example: Create a Transit AS



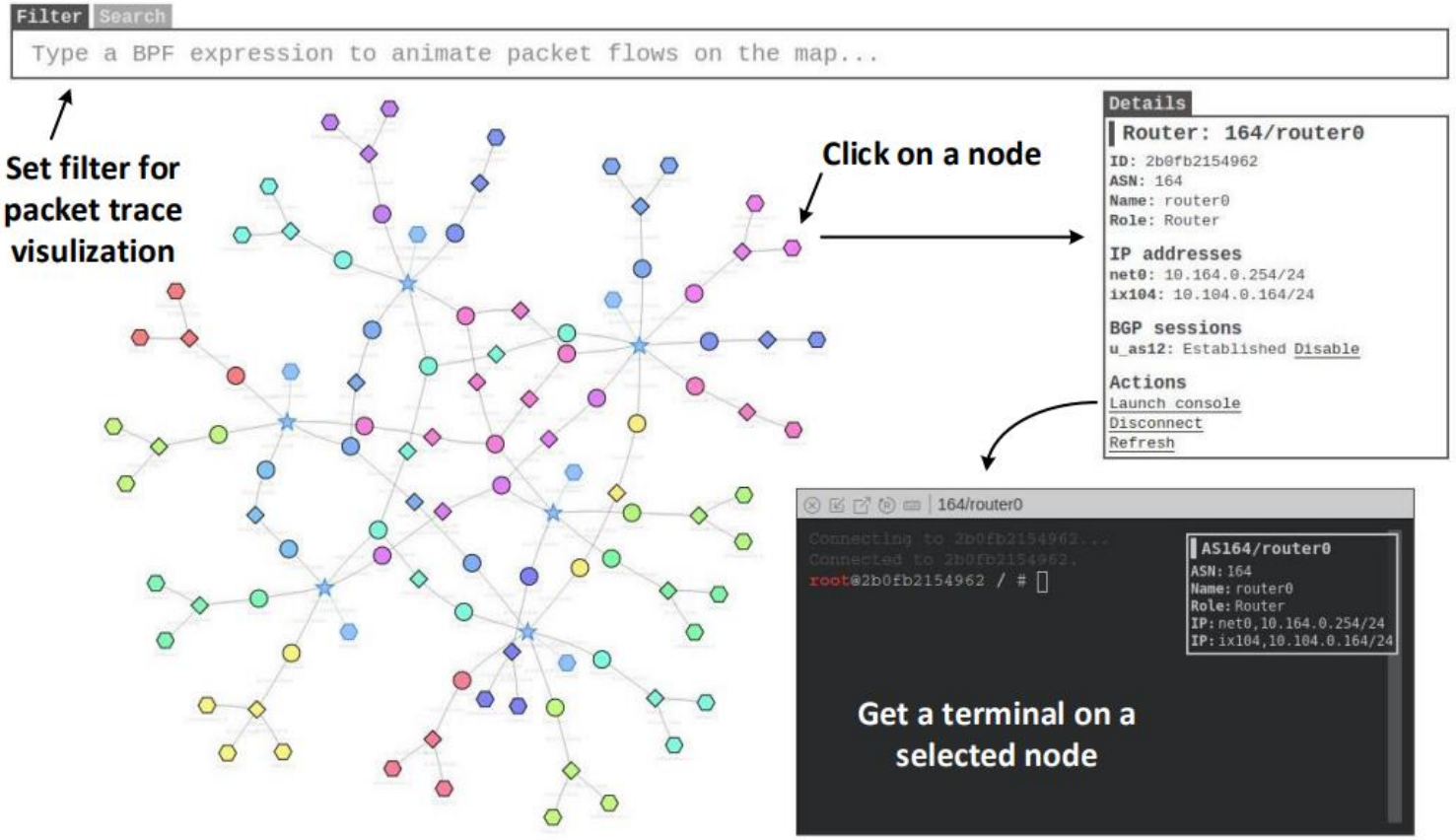

```
# Create the autonomous system (asn = 2)
as2 = base.createAutonomousSystem(2)

# Create 3 internal networks
as2.createNetwork('net0')
as2.createNetwork('net1')
as2.createNetwork('net2')

# Create 4 routers
as2.createRouter('r0').joinNetwork('ix100')
                      .joinNetwork('net0')
as2.createRouter('r1').joinNetwork('net0')
                      .joinNetwork('ix101')
                      .joinNetwork('net1')
as2.createRouter('r2').joinNetwork('net1')
                      .joinNetwork('net2')
as2.createRouter('r3').joinNetwork('net2')
                      .joinNetwork('ix102')
```

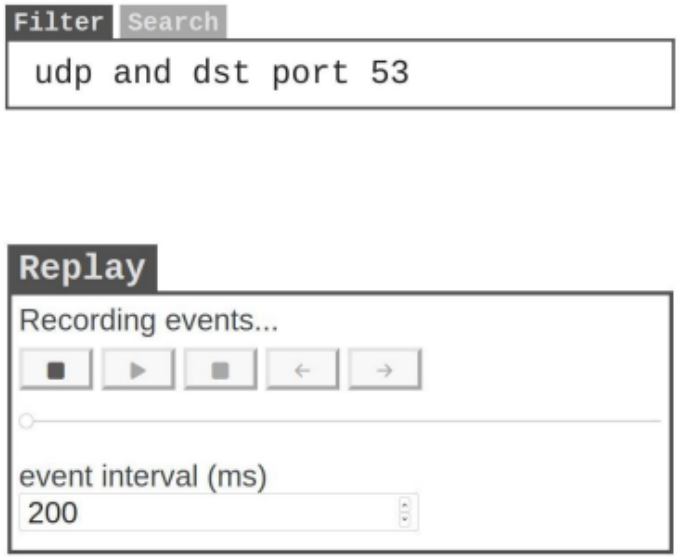
Example: BGP Peering




```
ebgp.addPrivatePeerings(102, [2, 4],  [11, 154], PeerRelationship.Provider)
ebgp.addPrivatePeerings(102, [11], [154, 11872], PeerRelationship.Provider)
```

Visualization Tool: the Map





Slide from: https://seedsecuritylabs.org/emulator/emulator_01_2022.pptx

Additional Information

SEED Website: **https://seedsecuritylabs.org/**

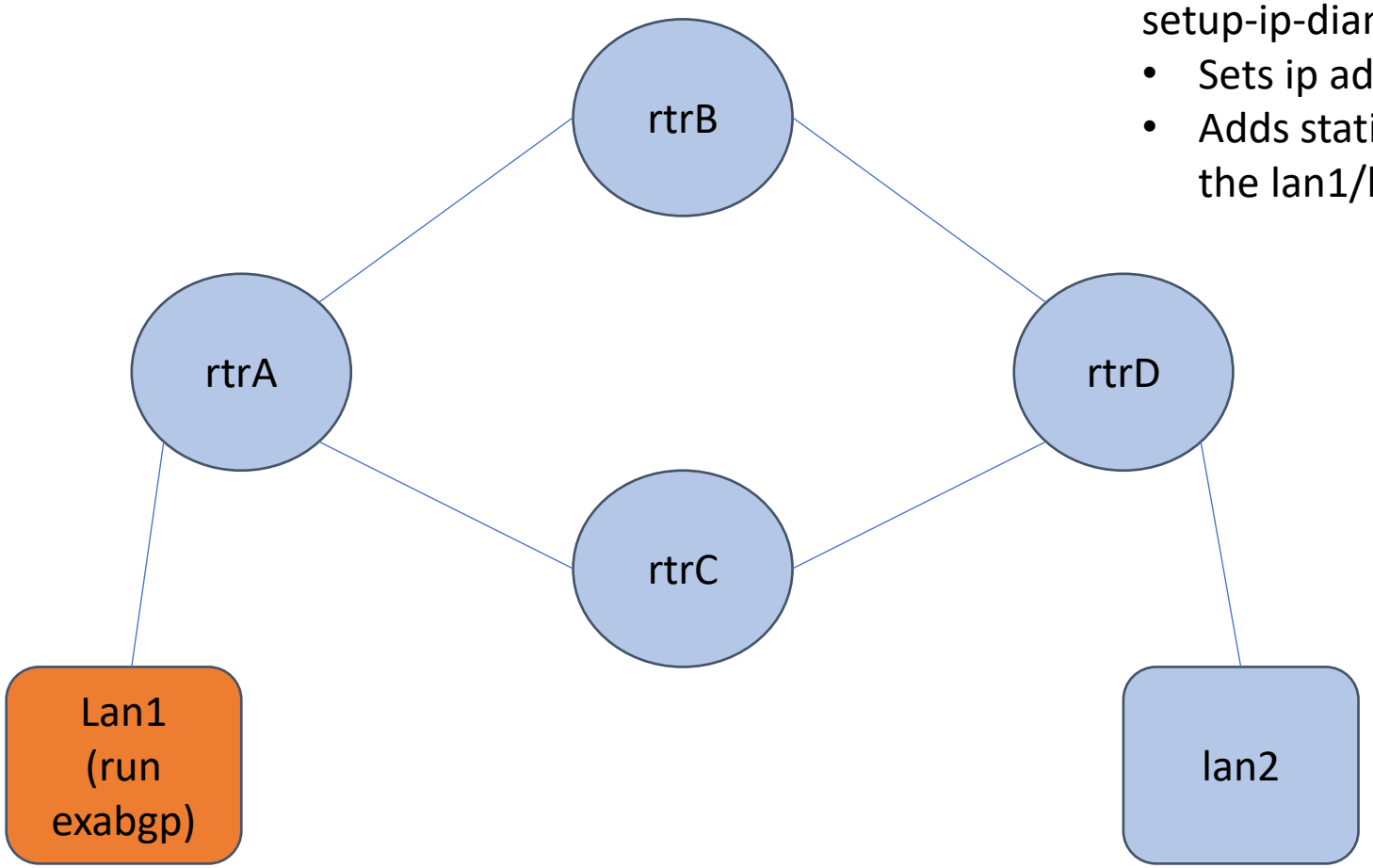
SEED Internet Emulator

We have developed an open-source Python framework, which can be used to create emulation of the Internet. It opens a door for many new activities that are difficult to perform in the current SEED platform, including BGP attacks, large-scale DNS attacks, Blockchain, Botnet, Dark-net, etc. We welcome everybody to join us in this project. More details about the Internet emulator and labs can be found here.

Emulator-Based Labs | Videos | Code and Documentation

Slide from: https://seedsecuritylabs.org/emulator/emulator_01_2022.pptx

Topology – with ExaBGP



setup-ip-diamond.sh

- Sets ip addresses for each interface
- Adds static routes on rtrA and rtrD to the lan1/lan2

diamond-mod2.clab.yml

Configuration / Operation

For each container mount ./mod2-bird-confs to /etc/bird

- rtrA-bird.conf
- exa.conf

start-bird.sh

- bird -c /etc/bird-alt/rtrA-bird.conf

Run exabgp

* lan1   exabgp -v /etc/bird-alt/exa.conf

birdc – utility (in the container) to interact with bird

- ip route show
- …

University of Colorado **Boulder**