Free University of Brussels

Lab 1: Introduction to Routing & Inter-VLAN Addressing

Using Cisco GNS3 on Linux

Lab Group: 2

BOTTON David	000615056
SUNDARESAN Sankara Narayanan	000599921
MESSAOUDI Leila	000442388
CHOUGULE Gaurav	000619973

Program: MS Cybersecurity M-SECUC:1

Course: ELEC-H504 - Network Security

Instructor: Navid Ladner

Report Due Date: April 7, 2025

Table of Content

1 Initial Topology **3**
1.1 GNS3 WebUI . . . 3
1.2 Router Assessment . . . 4
1.3 Host Assessment . . . 5

2 Mission 0 - GNS3 Basics **6**
2.1 Implementation Walkthrough . . . 6
2.1.1 Host Configuration . . . 6
2.1.2 Router Configuration . . . 7
2.2 Debugging and Validation . . . 8
2.2.1 Connectivity Testing . . . 8
2.2.2 Wireshark Analysis . . . 8

3 Mission 1 - RIPv2 Implementation **11**
3.1 Extended Topology . . . 11
3.2 Implementation Walkthrough . . . 12
3.2.1 Extended Configuration for R1 . . . 13
3.2.2 Router Configuration for R2 . . . 13
3.2.3 ipterm-2 configuration . . . 14
3.2.4 Validation . . . 14
3.3 RIPv2 Implementation and Analysis . . . 14
3.3.1 RIPv2 Configuration . . . 14
3.3.2 Traceroute Operation . . . 15
3.3.3 Static Route Cleanup Validation . . . 15
3.3.4 MD5 Authentication Implementation . . . 16
3.4 Redundancy and Dynamic Re-routing . . . 17

4 Mission 2 - Inter-VLAN Routing Analysis **19**
4.1 Inter-VLAN Traffic Analysis . . . 19
4.2 Inter-VLAN Routing Configuration and Verification . . . 20

Abstract

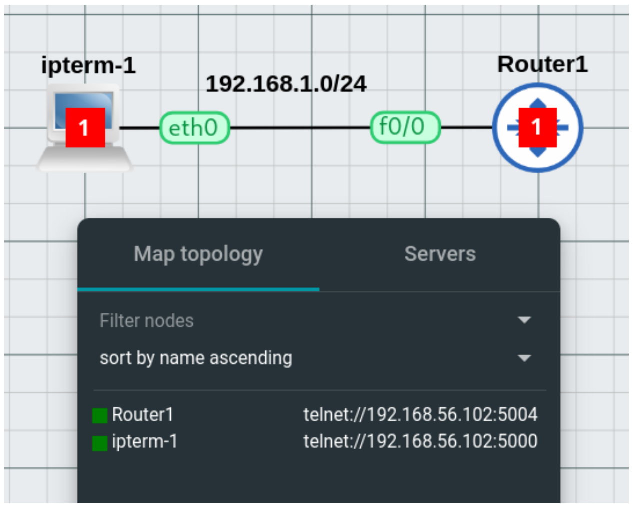
This technical report documents a methodical progression through a series of labs geared towards network security conducted at ULB. This first paper treats the foundations of network administration with Cisco virtual appliances as well as GNS3's hybrid orchestration using its native WebUI and a virtual machine serving Cisco IOS images on-the-fly. This particular lab is not complicated, it fits the level of a first hands-on exercise for practicing common networking operations in a typical design. A short primary network assessment is done, then the rest of the work is seen to be split in three distinct missions; Mission 0, starts from the smallest possible network in order to learn basic competencies in Cisco CLI operations and later validate point-to-point connectivity. Mission 1 is to set up a legacy `Interior Gateway Protocol`*, RIPv2, for dynamic routing of our packets in subnetted network. An essential step in understanding further mechanisms still in use inside all of our networks nowadays. Mission 2 goes one more step ahead with VLAN, a layer 2 logical network segmentation technique which brings very interesting concepts like trunk links, isolation and frame tagging under the IEEE's dot1q standard. I conclude last mission by ensuring inter-VLAN routing is indeed effective. Readers seeking deeper Cisco networking knowledge may find the complete Jeremy's IT Lab CCNA video series on YouTube, truly an amazing companion resource.*

1 Initial Topology

Starting with a minimal topology (single host—router) without any preconfigured interfaces or routes, is established a controlled environment to validate each layer of network functionality from the ground up. This approach mirrors real-world provisioning workflows, where engineers must explicitly enable and secure interfaces, allowing granular observation of GNS3's packet-handling behavior and Cisco IOS baseline states.

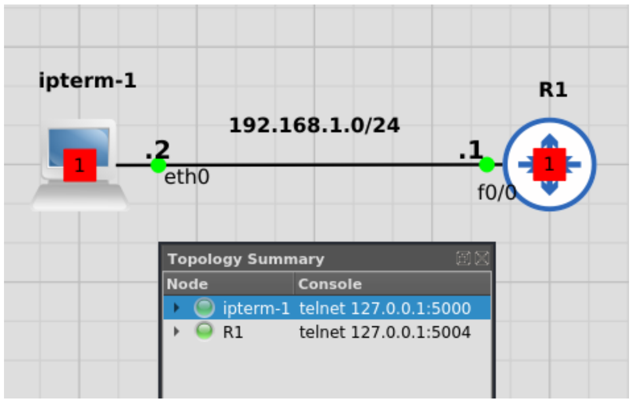
1.1 GNS3 WebUI

The first step in topology represented in Fig.1 shows the point-to-point connection between ipterm-1 and Router R1. This configuration utilizes nested KVM virtualization on an Ubuntu virtual machine. Its primary host-only network interface card (NIC) is responsible for the communication between GNS3 server and WebUI while the secondary NIC, which has network address translation enabled, connects to the outside world. The Cisco IOS images run in KVM using raw socket passthrough right to the kernel network stack of the VM, where individual Layer 2 frames are created for protocol dissection.



🟩 Router1	telnet://192.168.56.102:5004
🟩 ipterm-1	telnet://192.168.56.102:5000

Figure 1: WebUI: Initial network topology

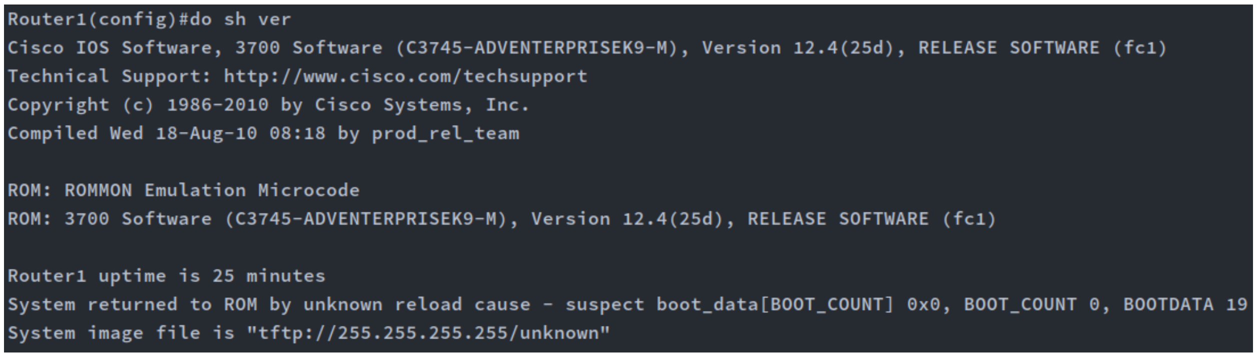


Node	Console
ipterm-1	telnet 127.0.0.1:5000
R1	telnet 127.0.0.1:5004

Figure 2: GNS3: Initial network topology

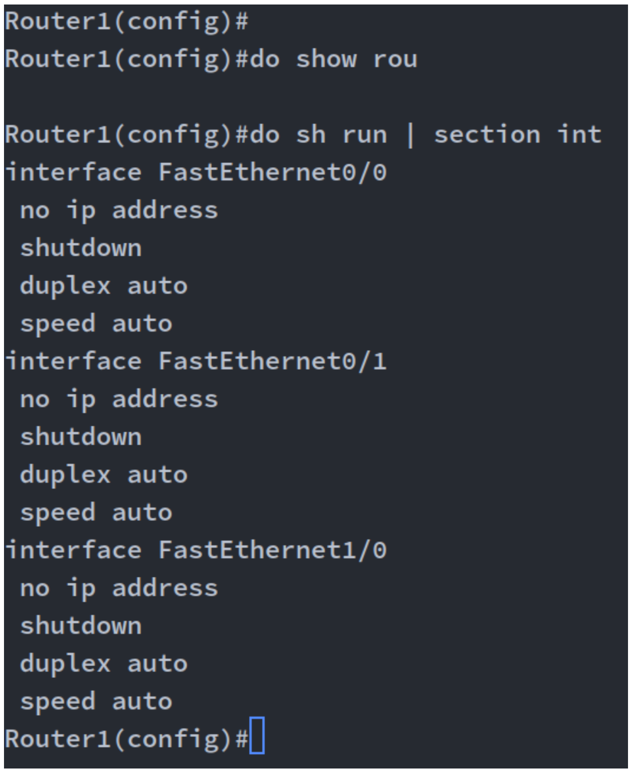
1.2 Router Assessment

- Router R1 (Fig.2-3) Assessment:
  - `IOS version 12.4(25d)` confirmed feature parity with lab requirements (RIPv2 support)
  - `show ip route` returned empty table, expected for an unconfigured router
  - All physical interfaces `administratively down` (Cisco's default secure state)
  - FastEthernet0/0, 0/1, and 1/0 all showed `unassigned` IP status
  - No VLANs or trunking interfaces configured in baseline state



```
Router1(config)#do sh ver
Cisco IOS Software, 3700 Software (C3745-ADVENTERPRISEK9-M), Version 12.4(25d), RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2010 by Cisco Systems, Inc.
Compiled Wed 18-Aug-10 08:18 by prod_rel_team

ROM: ROMMON Emulation Microcode
ROM: 3700 Software (C3745-ADVENTERPRISEK9-M), Version 12.4(25d), RELEASE SOFTWARE (fc1)

Router1 uptime is 25 minutes
System returned to ROM by unknown reload cause - suspect boot_data[BOOT_COUNT] 0x0, BOOT_COUNT 0, BOOTDATA 19
System image file is "tftp://255.255.255.255/unknown"
```

Figure 3: R1 version

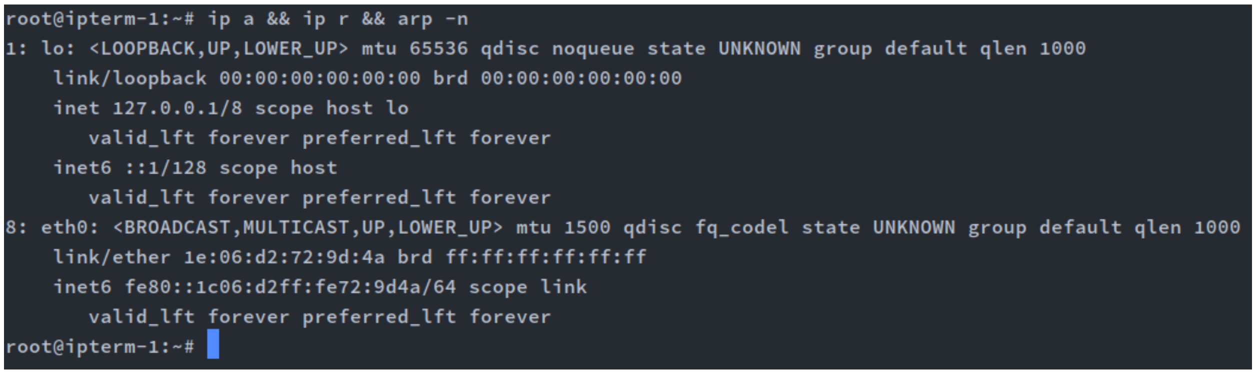


```
Router1(config)#
Router1(config)#do show rou

Router1(config)#do sh run | section int
interface FastEthernet0/0
 no ip address
 shutdown
 duplex auto
 speed auto
interface FastEthernet0/1
 no ip address
 shutdown
 duplex auto
 speed auto
interface FastEthernet1/0
 no ip address
 shutdown
 duplex auto
 speed auto
Router1(config)#
```

Figure 4: R1 routes and interfaces

1.3 Host Assessment

- Host ipterm-1 (Fig.4) Assessment:
  - `eth0` showed no IPv4 assignment
  - ARP cache completely empty (no L2 adjacencies)
  - Routing table empty
  - No default gateway configured
  - Interface maximum transmission unit (MTU) size defaulted to 1500 bytes



```
root@ipterm-1:~# ip a && ip r && arp -n
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    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
8: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 1e:06:d2:72:9d:4a brd ff:ff:ff:ff:ff:ff
    inet6 fe80::1c06:d2ff:fe72:9d4a/64 scope link
       valid_lft forever preferred_lft forever
root@ipterm-1:~#
```

Figure 5: ipterm-1 network situation report

2 Mission 0 - GNS3 Basics

2.1 Implementation Walkthrough

These practical labs take advantage of Cisco's hierarchical CLI architecture that has its own learning curve in comparison to most conventional shells. Cisco IOS is mode-based and handles separation of concerns and layered access control at the command-line level. For example, Global config mode is used for system parameters, interface mode for context-specific IP assignment, privilege exec mode for verification, etc. Also, it integrates peculiar command truncation capabilities (e.g., "conf t" is the same command as "configure terminal"), configuration speed is preferred while maintaining readability through flexible command inference. *Nota Bene*: In Cisco IOS the helper is coupled with the completion system, called by the character ≪*?*≫ at any point while typing a command.

- **A few key differentiators from traditional shells quickly emerge when using Cisco IOS:**
  - `Stateful Contexts`: Cisco IOS uses modal configuration:
    * `User EXEC > Privileged EXEC > Global Config > Interface Config`
    * Bash/PowerShell use flat command structure with directory-based context
  - `Implicit Security`:
    * Cisco: Interfaces default to `shutdown` state
    * Linux: Network interfaces default to `up` with `ifconfig` auto-activation
  - `Configuration Lifecycle`:
    * Cisco: Changes reside in a first `running-config` (RAM) until written to a persistent `startup-config` (NVRAM) or flushed
    * Bash: Immediate filesystem writes (e.g., `/etc/network/interfaces`)
  - `Line Mode Features`:
    * Cisco: Context-aware command abbreviation (`conf t` = `configure terminal`)
    * Bash: Requires explicit aliases for command shortening

The GNS3 system itself needed to be carefully orchestrated between native Linux workstation for the Web UI and the VM housing the computationally demanding iOS images. Packet captures were automatically routed through the virtual interfaces of the VM. This hybrid architecture allows taking advantage of host resources while maintaining network isolation boundaries between lab instances.

2.1.1 Host Configuration

Persistent network configuration is critical for lab consistency across VM reboots and topology reloads. While transient ifconfig commands could temporarily set the IP, a choice was made in persisting host configurations through Ubuntu's `/etc/network/interfaces` file.

1. Connect to ipterm-1 via telnet:

```
$ telnet 192.168.56.102 5000
```

2. Configure interface and gateway permanently:

```
$ vi /etc/network/interfaces
      auto eth0
      iface eth0 inet static
          address 192.168.1.2
          netmask 255.255.255.0
          gateway 192.168.1.1
```

This arrangement enables three primary objectives; Boot-Time Activation: auto eth0 ensures interface activation before starting any service. Subnet Alignment: /24 netmask aligns with topology's FastEthernet0/0 interface requirement. Default Path Enforcement: Gateway declaration generates persistent route to R1.

3. Verify routing table:

```
$ ip r
  # default via 192.168.1.1 dev eth0
  # 192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.2
```

These telnet sessions utilize GNS3 NAT mapping through the VM's host-only adapter forwarding console access without lab traffic being commingled with it. That segregation prevented ARP contamination from reaching management and lab networks.

2.1.2 Router Configuration

Cisco's hierarchical configuration model requires deliberate state transitions to implement changes. The process begins with privilege escalation from user EXEC (<) to privileged EXEC (#) via `enable` command, although GNS3's auto-privilege setting bypasses this step. The critical phase starts with `configure terminal`, entering global configuration mode (`(config)#`) where interface-specific parameters are set.

1. Connect to R1 via telnet:

```
$ telnet 192.168.56.102 5004
```

2. Enter global configuration mode:

```
R1# conf t
```

3. Configure FastEthernet 0/0 interface:

```
R1(config)# int f0/0
R1(config-if)# ip add 192.168.1.1 255.255.255.0
R1(config-if)# no shu
```

4. Save configuration:

```
R1(config)# do wr
```

5. Verify interface status:

```
R1(config)# do sh ip int br
# Interface                  IP-Address      OK? Method Status                Protocol
# FastEthernet0/0            192.168.1.1     YES manual up                    up
# FastEthernet0/1            unassigned      YES unset  administratively down down
# FastEthernet1/0            unassigned      YES unset  administratively down down
```

Validation using show ip interface brief provides multi-state awareness: Status Column: Physical layer status (up/down) Protocol Column: Data link layer status Method: Checks manual (static) versus dynamic (DHCP) addressing `no shu` command shortened is a demonstration of Cisco's context-aware abbreviation strategy, where abbreviated commands are performed if unambiguous within current config mode. This is very different from Bash's exact alias or full command path requirement. Security Note: While Telnet is good for labs, production environments have to use SSH with `transport input ssh in line vty` config to prevent credential sniffing. Telnet should be disabled by default on production systems.

2.2 Debugging and Validation

Verification started with simple ICMP connectivity testing with ping 192.168.1.1 from ipterm-1 to R1, verifying L3 reachability. Cisco's debug ip packet disclosed raw packet processing, revealing normal ICMP echo flow.

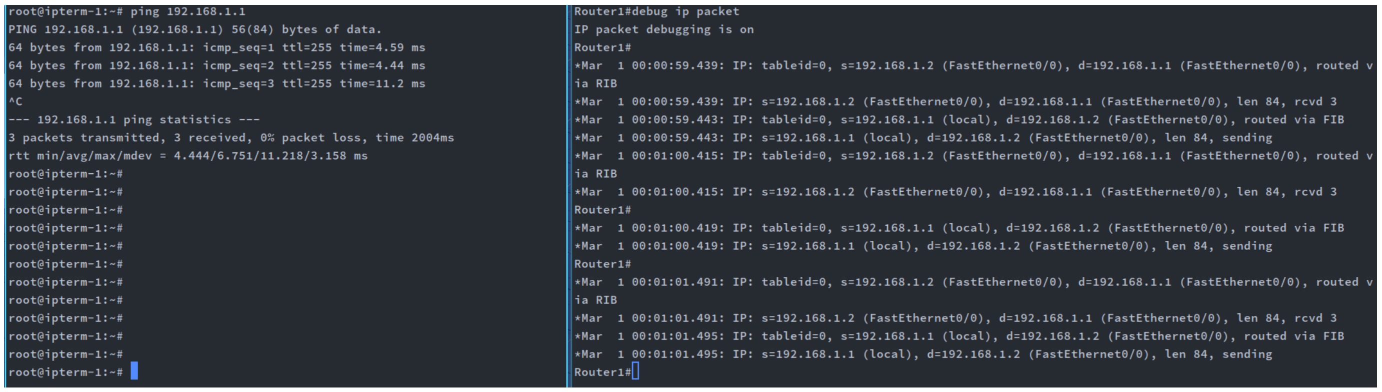
2.2.1 Connectivity Testing

1. Ping R1 from host:

```
R1# ping 192.168.1.1
```

2. Cisco packet capture on R1:

```
R1# debug ip packet
```



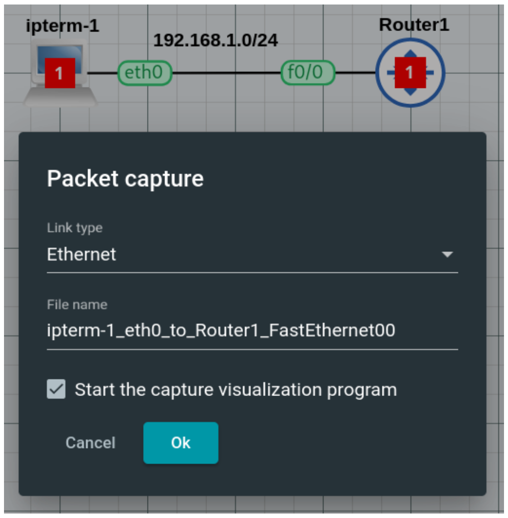
```
root@ipterm-1:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=255 time=4.59 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=255 time=4.44 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=255 time=11.2 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 4.444/6.751/11.218/3.158 ms
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
root@ipterm-1:~#
```

```
Router1#debug ip packet
IP packet debugging is on
Router1#
*Mar  1 00:00:59.439: IP: tableid=0, s=192.168.1.2 (FastEthernet0/0), d=192.168.1.1 (FastEthernet0/0), routed v
ia RIB
*Mar  1 00:00:59.439: IP: s=192.168.1.2 (FastEthernet0/0), d=192.168.1.1 (FastEthernet0/0), len 84, rcvd 3
*Mar  1 00:00:59.443: IP: tableid=0, s=192.168.1.1 (local), d=192.168.1.2 (FastEthernet0/0), routed via FIB
*Mar  1 00:00:59.443: IP: s=192.168.1.1 (local), d=192.168.1.2 (FastEthernet0/0), len 84, sending
*Mar  1 00:01:00.415: IP: tableid=0, s=192.168.1.2 (FastEthernet0/0), d=192.168.1.1 (FastEthernet0/0), routed v
ia RIB
*Mar  1 00:01:00.415: IP: s=192.168.1.2 (FastEthernet0/0), d=192.168.1.1 (FastEthernet0/0), len 84, rcvd 3
Router1#
*Mar  1 00:01:00.419: IP: tableid=0, s=192.168.1.1 (local), d=192.168.1.2 (FastEthernet0/0), routed via FIB
*Mar  1 00:01:00.419: IP: s=192.168.1.1 (local), d=192.168.1.2 (FastEthernet0/0), len 84, sending
Router1#
*Mar  1 00:01:01.491: IP: tableid=0, s=192.168.1.2 (FastEthernet0/0), d=192.168.1.1 (FastEthernet0/0), routed v
ia RIB
*Mar  1 00:01:01.491: IP: s=192.168.1.2 (FastEthernet0/0), d=192.168.1.1 (FastEthernet0/0), len 84, rcvd 3
*Mar  1 00:01:01.495: IP: tableid=0, s=192.168.1.1 (local), d=192.168.1.2 (FastEthernet0/0), routed via FIB
*Mar  1 00:01:01.495: IP: s=192.168.1.1 (local), d=192.168.1.2 (FastEthernet0/0), len 84, sending
Router1#
```

Figure 6: Cisco packet capture from ipterm-1 to R1

2.2.2 Wireshark Analysis

The native GNS3 Web interface behaved erratically under my Linux host machine. Although Figure 6 illustrates starting a capture through the GUI, actual Wireshark analysis involved terminal acrobatics. I suggest switching gns3-gui or SSH into the erver instead. For analysts preferring local tools; file exfiltration using `scp` bypassed the WebUI's export function. A welcomed workaround given UI clunkiness and other undefined behaviors during capture.



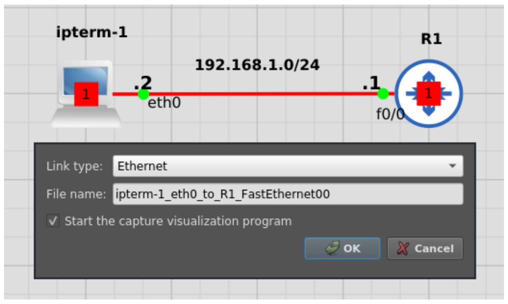
Packet capture

Link type: Ethernet

File name: ipterm-1_eth0_to_Router1_FastEthernet00

☑ Start the capture visualization program

Figure 7: Start a capture from WebUI



Link type: Ethernet

File name: ipterm-1_eth0_to_R1_FastEthernet00

✓ Start the capture visualization program

Figure 8: Start a capture from GNS3

1. Connect to gns3 server with X forwarding:

```
$ ssh -X gns3@192.168.56.102
      # save key?
      yes
      # password?
      gns3
```

2. Find pcap:

```
$ sudo find / -name "*ipterm-1_eth0_to_Router1_FastEthernet00*" -type f 2>/dev/null |
    grep -i gns3
```

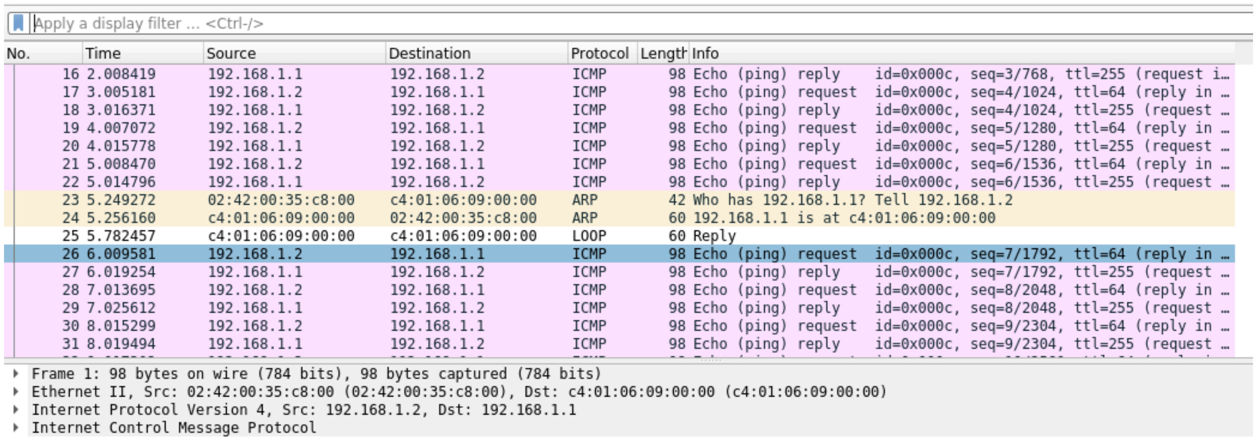
3. Open in Wireshark:

```
$ wireshark ipterm-1_eth0_to_Router1_FastEthernet00
```

Alternative: Exfiltrate pcap back to SSH client

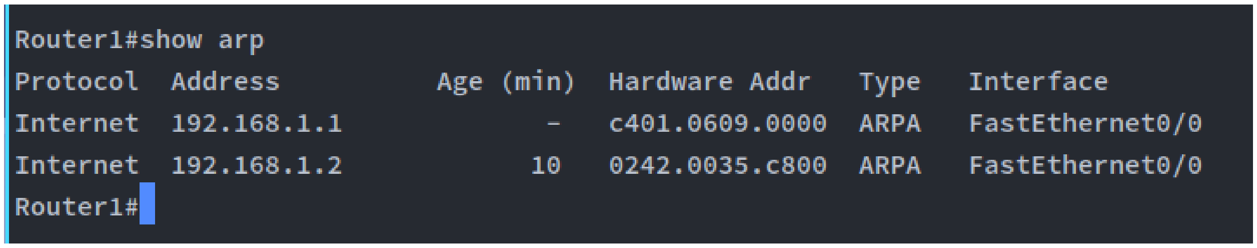
```
$ scp "ipterm-1_eth0_to_Router1_FastEthernet00" username@$(echo $SSH_CLIENT | awk '{
    print $1}'):/home/username/
```

Layer 2/3 validation was performed using a cross-tool synthesis: Wireshark confirmed 802.3 frames encapsulating ICMP payloads from ipterm-1 to R1, while Cisco's `show arp` and `show interfaces` confirmed MAC bindings (Figure 8). This hybrid approach; CLI debugs for real-time telemetry and packet captures for post-hoc analysis, is a good practice in a networking labs. The necessity of learning both vendor-specific diagnostics (e.g., debug modes of Cisco) and general packet analysis is a core skill for any network security job.



```
No.   Time      Source             Destination        Protocol Length Info
   16 2.008419  192.168.1.1        192.168.1.2        ICMP      98 Echo (ping) reply    id=0x000c, seq=3/768, ttl=255 (request i…
   17 3.005181  192.168.1.2        192.168.1.1        ICMP      98 Echo (ping) request  id=0x000c, seq=4/1024, ttl=64 (reply in …
   18 3.016371  192.168.1.1        192.168.1.2        ICMP      98 Echo (ping) reply    id=0x000c, seq=4/1024, ttl=255 (request …
   19 4.007072  192.168.1.2        192.168.1.1        ICMP      98 Echo (ping) request  id=0x000c, seq=5/1280, ttl=64 (reply in …
   20 4.015778  192.168.1.1        192.168.1.2        ICMP      98 Echo (ping) reply    id=0x000c, seq=5/1280, ttl=255 (request …
   21 5.008470  192.168.1.2        192.168.1.1        ICMP      98 Echo (ping) request  id=0x000c, seq=6/1536, ttl=64 (reply in …
   22 5.014796  192.168.1.1        192.168.1.2        ICMP      98 Echo (ping) reply    id=0x000c, seq=6/1536, ttl=255 (request …
   23 5.249272  02:42:00:35:c8:00  c4:01:06:09:00:00  ARP       42 Who has 192.168.1.1? Tell 192.168.1.2
   24 5.256160  c4:01:06:09:00:00  02:42:00:35:c8:00  ARP       60 192.168.1.1 is at c4:01:06:09:00:00
   25 5.782457  c4:01:06:09:00:00  c4:01:06:09:00:00  LOOP      60 Reply
   26 6.009581  192.168.1.2        192.168.1.1        ICMP      98 Echo (ping) request  id=0x000c, seq=7/1792, ttl=64 (reply in …
   27 6.019254  192.168.1.1        192.168.1.2        ICMP      98 Echo (ping) reply    id=0x000c, seq=7/1792, ttl=255 (request …
   28 7.013695  192.168.1.2        192.168.1.1        ICMP      98 Echo (ping) request  id=0x000c, seq=8/2048, ttl=64 (reply in …
   29 7.025612  192.168.1.1        192.168.1.2        ICMP      98 Echo (ping) reply    id=0x000c, seq=8/2048, ttl=255 (request …
   30 8.015299  192.168.1.2        192.168.1.1        ICMP      98 Echo (ping) request  id=0x000c, seq=9/2304, ttl=64 (reply in …
   31 8.019494  192.168.1.1        192.168.1.2        ICMP      98 Echo (ping) reply    id=0x000c, seq=9/2304, ttl=255 (request …

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
▸ Ethernet II, Src: 02:42:00:35:c8:00 (02:42:00:35:c8:00), Dst: c4:01:06:09:00:00 (c4:01:06:09:00:00)
▸ Internet Protocol Version 4, Src: 192.168.1.2, Dst: 192.168.1.1
▸ Internet Control Message Protocol
```

Figure 9: Captured ping analysis on the gns3 server



```
Router1#show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  192.168.1.1             -   c401.0609.0000  ARPA   FastEthernet0/0
Internet  192.168.1.2            10   0242.0035.c800  ARPA   FastEthernet0/0
Router1#
```

Figure 10: L2 addressing confirmation on R1

3 Mission 1 - RIPv2 Implementation

3.1 Extended Topology

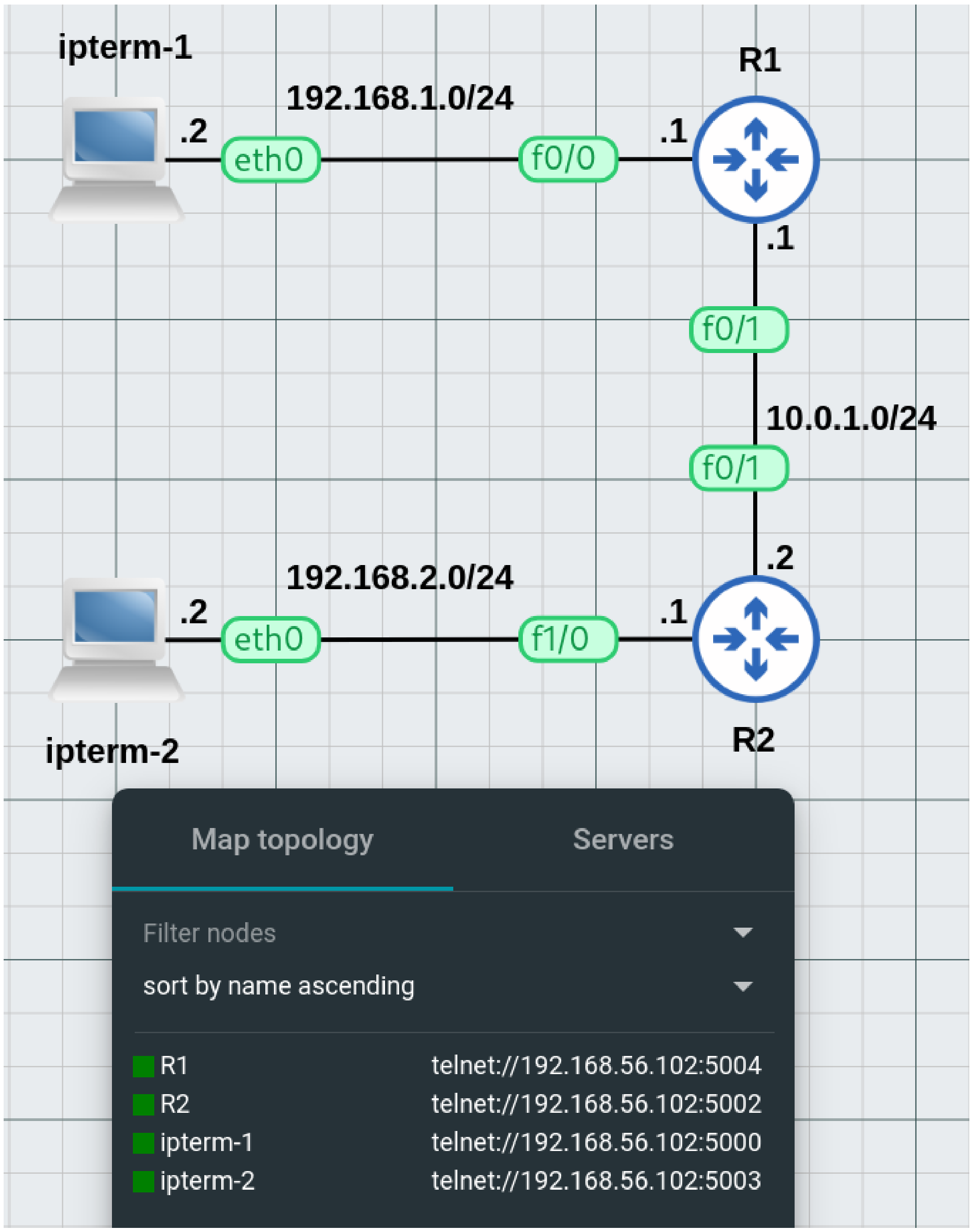
The expanded topology introduced R2 and ipterm-2 to create a multi-hop setting (Figure 9). As much as the lab needed dynamic routing through RIPv2, I took the initiative to introduce static routes anyway. Here are three insights explaining some reasons why to do so:

Protocol Prioritization: Cisco routers prefer routes based on the lowest Administrative Distance (AD) value. Static routes have a default AD at 1, they always take precedence over RIPv2-learned routes which default at 120. A poorly set up static route would effectively blackhole traffic for all eternity. In a small static lab demonstrating RIP, they can be useful as automatic backup paths in case of any routing protocol failure. By setting static routes manually, I demonstrate routers' preference for conflicting sources of routes and the failover mechanisms in downgrading to a higher AD route automatically. This is essential to understand for network troubleshooting. Thus, I set the two Static routes with an AD at 121, a requirement for both hosts in the extended topology to communicate with each other while testing RIP.

Redundancy & ECMP: Although not strictly required, static route configuration established the topology to readily accommodate Equal-Cost Multi-Path (ECMP) scenarios. In the event that redundant links prove problematic, static routes of equal metric enable load balancing, a precursor to RIPv2's hop-count-based path selection where static routes define basic paths and dynamic protocols offer failover.

Security Baseline: Static routes eliminate attack vectors inherent in RIP (e.g., route advertisement spoofing). By way of contrast and resulting RIPv2 deployment, I highlighted the security/performance tradeoff: static routes are stable but non-scalable, and RIP automatically converges at the cost of protocol susceptibility – a tradeoff central to network hardening in future labs.

This method followed defense-in-depth design, foundational principles in modern networks. It provided a controlled environment in which to observe RIP's convergence mechanisms on known-good static paths.



🟩 R1	telnet://192.168.56.102:5004
🟩 R2	telnet://192.168.56.102:5002
🟩 ipterm-1	telnet://192.168.56.102:5000
🟩 ipterm-2	telnet://192.168.56.102:5003

Figure 11: Extended topology.

3.2 Implementation Walkthrough

The deliberate application of AD manipulation (e.g. ip route 192.168.2.0 255.255.255.0 10.0.1.2 121) and passive debugging was applied to examine the behavior of RIPv2 without dismantling the static route safety net. Artificially elevating static route AD above RIP's default of 120, I simulated a ≪dynamic-primary/static-backup≫ design, forcing RIPv2 to dominate the routing table while preserving static routes as dormant failovers. This method is necessary for stress-testing RIP hold-down timers and triggered updates without endangering complete connectivity loss. Moreover, passive packet analysis maintained static route integrity while revealing raw RIPv2 mechanics: unauthenticated updates (before MD5 configuration) and split horizon enforcement even as static routes quietly managed traffic.

3.2.1 Extended Configuration for R1

1. Connect via telnet:

```
$ telnet 192.168.56.102 5004
```

2. Enter global configuration mode:

```
R1# conf t
```

3. Configure interface f0/1:

```
R1(config)# int f0/1
R1(config-if)# ip add 10.0.1.1 255.255.255.0
R1(config-if)# no shu
```

4. Configure static route to reach the 192.168.2.0/24 network:

```
R1(config-if)# ip route 192.168.2.0 255.255.255.0 10.0.1.2 121
```

5. Save configuration:

```
R1(config-if)# do wr
```

3.2.2 Router Configuration for R2

1. Connect via telnet:

```
$ telnet 192.168.56.102 5002
```

2. Enter global configuration mode:

```
R1# conf t
```

3. Configure FastEthernet 0/1 interface (WAN side):

```
R1(config)# int f0/1
R1(config-if)# ip add 10.0.1.2 255.255.255.0
R1(config-if)# no shu
```

4. Configure FastEthernet 1/0 interface (LAN side):

```
R1(config-if)# int f1/0
R1(config-if)# ip add 192.168.2.1 255.255.255.0
R1(config-if)# no shu
```

5. Configure static route to reach the 192.168.1.0/24 network:

```
R1(config-if)# ip route 192.168.1.0 255.255.255.0 10.0.1.1 121
```

6. Save configuration:

```
R1(config)# do wr
```

The static route S 192.168.1.0/24 [121/0] via 10.0.1.1 in R2's routing table confirms bidirectional L3 reachability between R2's Fa0/1 (10.0.1.2) and R1's Fa0/1 (10.0.1.1). Although directly connected C 10.0.1.0 and C 192.168.2.0 interfaces confirm local segment availability, the presence of the static route confirms that manual path injection was successful. This is contrasted with RIP-learned routes (AD 120), which are intentionally overridden here to confirm baseline IP forwarding before dynamic protocol deployment. The [121/0] metric indicates zero hop cost, as though the next-hop is directly connected, valuable piece of information when troubleshooting asymmetric routing issues.

7. A static route appears on each router if round-trip is achievable:

```
# Example R2
do sh ip ro
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 1 subnets
C       10.0.1.0 is directly connected, FastEthernet0/1
S    192.168.1.0/24 [121/0] via 10.0.1.1
C    192.168.2.0/24 is directly connected, FastEthernet1/0
```

3.2.3 ipterm-2 configuration

1. Configure interface and gateway permanently:

```
$ vi /etc/network/interfaces
      auto eth0
      iface eth0 inet static
          address 192.168.2.2
          netmask 255.255.255.0
          gateway 192.168.2.1
```

3.2.4 Validation

1. From ipterm-2, test connectivity:

```
$ ping -c 4 192.168.1.2    # ipterm-1, see ARP propagation causing packet loss
```

2. Trace routes between hosts:

```
# From ipterm-1 to ipterm-2
$ traceroute 192.168.2.2
# From ipterm-2 to ipterm-1
$ traceroute 192.168.1.2
```

3.3 RIPv2 Implementation and Analysis

3.3.1 RIPv2 Configuration

Router configurations strictly follow interface-specific network statements. For R1 (Fa0/1: 10.0.1.1, Fa0/0: 192.168.1.1):

```
R1# conf t
R1(config)# router rip
R1(config-router)# ver 2
R1(config-router)# no auto-summary  ! Disables classful behavior
R1(config-router)# net 10.0.1.0
R1(config-router)# net 192.168.1.0  ! 192.168.2.0 R2
R1(config-router)# end
R1# wr
```

Listing 1: R1 Correct RIP Configuration

R2 configuration mirrors this structure for its connected networks (also 10.0.1.0 but 192.168.2.0 instead). Verification via `show ip rip database` confirms route propagation:

```
R1# sh ip rip data
# 10.0.0.0/8        auto-summary
# 10.0.1.0/24       directly connected, FastEthernet0/1
# 192.168.1.0/24    auto-summary
# 192.168.1.0/24    directly connected, FastEthernet0/0
# 192.168.2.0/24    auto-summary
# 192.168.2.0/24
#     [1] via 10.0.1.2, 00:00:21, FastEthernet0/1
```

Listing 2: R1 RIP Database

3.3.2 Traceroute Operation

The traceroute utility follows packet direction by exploiting the Time-to-Live (TTL) field of IP headers. TTL is decremented by every hop, and an ICMP ”Time Exceeded” message is sent back by the router that discards the packet. This gives us a step-by-step reconstruction of the path. The three-hop path (ipterm-1 → R2 → R1 → ipterm-2) in our output maps directly to the advertised routes of RIPv2, confirming dynamic routing functionality. Latency increase—11ms at R2, 33ms at R1—reveals cumulative processing delays between devices. Most importantly, the absence of asymmetric paths or timeout errors verifies RIP's convergence and absence of routing loops in this simple topology. This is a formidable tool for both baseline verification and fault isolation in any dynamic routing environment.

Executing `traceroute 192.168.2.2` from ipterm-1 reveals path construction:

```
ipterm-2$ traceroute -n 192.168.1.2
# traceroute to 192.168.1.2 (192.168.1.2), 30 hops max, 60 byte packets
#  1  192.168.2.1 (192.168.2.1)  11.817 ms  11.718 ms  11.671 ms
#  2  10.0.1.1 (10.0.1.1)  33.507 ms  44.193 ms  54.406 ms
#  3  192.168.1.2 (192.168.1.2)  85.777 ms  85.746 ms  95.800 ms
```

Listing 3: Traceroute from ipterm-2 to ipterm-1

3.3.3 Static Route Cleanup Validation

To confirm RIPv2's dominance over residual static configurations, we purge any legacy static routes while maintaining traceroute functionality:

```
R1# conf t
R1(config)# no ip route 192.168.2.0 255.255.255.0 10.0.1.2 121
R1(config)# end
R1# sh ip rou | inclu 192.168.2.0
R 192.168.2.0/24 [120/1] via 10.0.1.2, 00:00:07, FastEthernet0/1 ! RIP route persists
```

Listing 4: Removing Static Route with AD 121 on R1

Post-removal, `traceroute 192.168.2.2` from ipterm-1 had the same hop pattern, showing RIPv2 autonomous path retention. *Nota Bene*: For educational purposes, static routes were originally configured with an Administrative Distance value higher than default RIP, ensuring dynamic routes will always take priority over static entries unless explicitly privileged.

3.3.4 MD5 Authentication Implementation

To harden RIPv2 against unauthorized route injection, we implemented MD5 authentication between R1 and R2 using Cisco's keychain framework. After that point, R2 will lack MD5 authentication, while R1 has it enabled. This mismatch causes a one-way RIP communication failure, also called asymmetric authentication.

```
R1# conf t
R1(config)# key chain KC
R1(config-keychain)# key 1
R1(config-keychain-key)# key-string PaSSW0rD1 ! Case-sensitive
R1(config)# int fa0/1
R1(config-if)# ip rip authentication mode md5 ! Enforce crypto validation
R1(config-if)# ip rip authentication key-chain KC
R1(config-if)# do wr
```

Listing 5: R1 Keychain Configuration

Post-configuration, RIP adjacencies broke intentionally – routes aged out within 240 seconds (default invalid timer). The following frame captured between R2 and R1 does not contain any MD5 Authentication Header, R1 discards R2's unauthenticated RIP updates.

```
Frame 3: 106 bytes on wire (848 bits), 106 bytes captured (848 bits)
Ethernet II, Src: c4:02:07:01:00:01 (c4:02:07:01:00:01), Dst: IPv4mcast_09 (01:00:5e
    :00:00:09)
Internet Protocol Version 4, Src: 10.0.1.2, Dst: 224.0.0.9
User Datagram Protocol, Src Port: 520, Dst Port: 520
Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    IP Address: 10.0.2.0, Metric: 2
    IP Address: 10.0.3.0, Metric: 1
    IP Address: 192.168.2.0, Metric: 1

No.     Time           Source               Destination          Protocol Length Info
      4 7.254034       c4:02:07:01:00:01    c4:02:07:01:00:01    LOOP     60     Reply
```

Listing 6: R2 Mirror Configuration

```
R2(config)# key chain KC ! Identical key-ID/string required
R2(config-keychain)# key 1
R2(config-keychain-key)# key-string PaSSW0rD1 ! Mismatch = silent drop
R2(config-keychain-key)# int f0/1
R2(config-if)# ip rip authen mod md
```

Listing 7: Setup MD5 authenticaton on R2

When RIPv2 MD5 authentication is employed, Wireshark captures indicate three phases:

- Pre-authentication: RIPv2 updates are not authenticated, and route tables are in the clear.

- Post-MD5: Updates include an Authentication Header (Keyed Message Digest) with Key ID 1, showing MD5 authentication—though routes remain unencrypted.

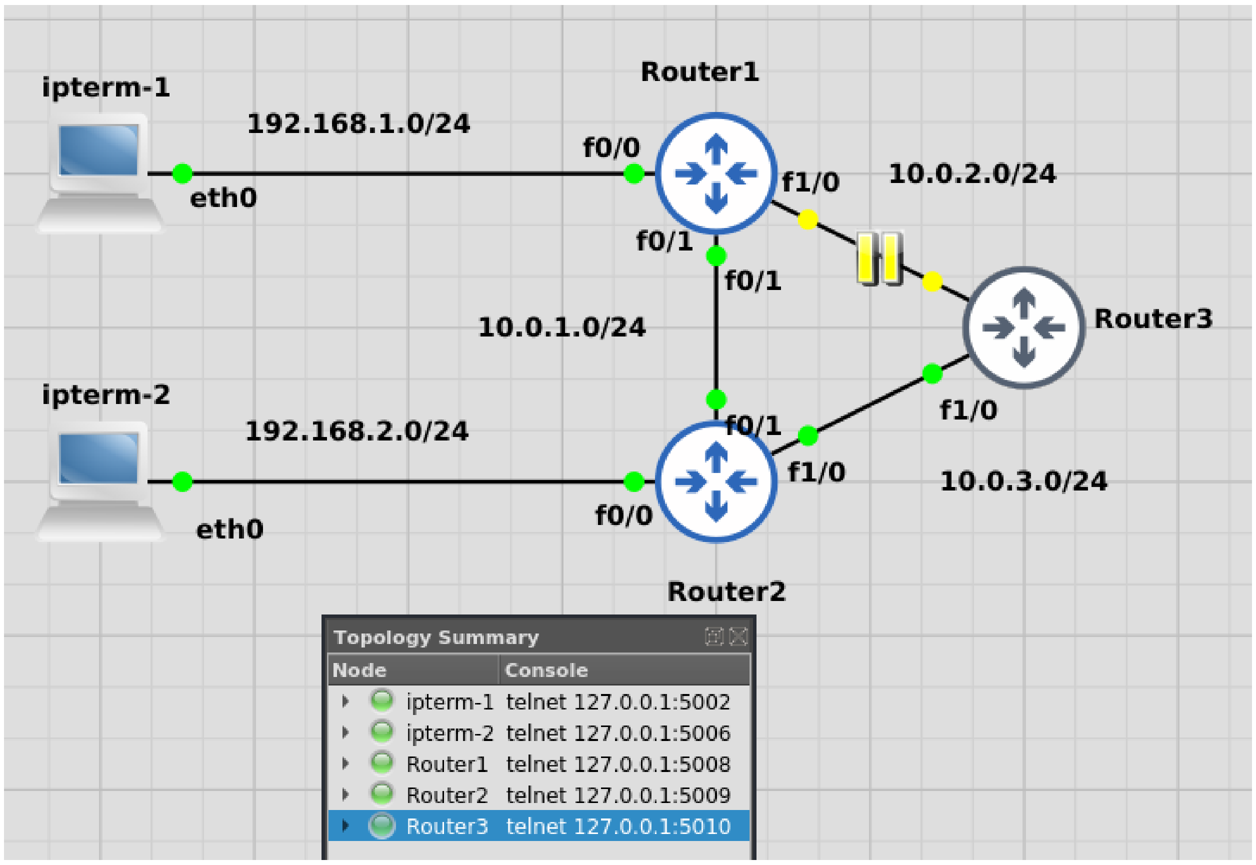
- Invalid keys: Mismatched keys cause silent discard of updates (no detectable errors in captures), causing routes to age out totally from tables ("stealthy failure").

```
Frame 8: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)
Ethernet II, Src: c4:01:06:e3:00:01 (c4:01:06:e3:00:01), Dst: IPv4mcast_09 (01:00:5e
    :00:00:09)
Internet Protocol Version 4, Src: 10.0.1.1, Dst: 224.0.0.9
User Datagram Protocol, Src Port: 520, Dst Port: 520
Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    Authentication: Keyed Message Digest
    IP Address: 10.0.2.0, Metric: 1
    IP Address: 192.168.1.0, Metric: 1

No.     Time           Source                Destination           Protocol Length Info
      9 21.678626      c4:02:07:01:00:01     c4:02:07:01:00:01     LOOP     60     Reply


Frame 12: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits)
Ethernet II, Src: c4:02:07:01:00:01 (c4:02:07:01:00:01), Dst: IPv4mcast_09 (01:00:5e
    :00:00:09)
Internet Protocol Version 4, Src: 10.0.1.2, Dst: 224.0.0.9
User Datagram Protocol, Src Port: 520, Dst Port: 520
Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    Authentication: Keyed Message Digest
    IP Address: 10.0.3.0, Metric: 1
    IP Address: 192.168.2.0, Metric: 1

No.     Time           Source                Destination           Protocol Length Info
     13 35.020738      c4:01:06:e3:00:01     c4:01:06:e3:00:01     LOOP     60     Reply
```

Listing 8: MD5 authenticaton R1 to R2

3.4 Redundancy and Dynamic Re-routing

R3 can be added in redundancy so that RIPv2 dynamically load-balances traffic across dual links (R1→R2 or R1→R3→R2). Traceroutes in the beginning show the shortest path via R1-R2. When this link is disabled, RIPv2 convergence is triggered (30s), diverting the traffic perfectly via R3, demonstrating fault tolerance. Trace routes post-failure will reveal an additional hop via R3 but remain connected, demonstrating enterprise-grade resilience. Redundancy + dynamic routing ensures uptime—critical for production networks.



Topology Summary

Node	Console
ipterm-1	telnet 127.0.0.1:5002
ipterm-2	telnet 127.0.0.1:5006
Router1	telnet 127.0.0.1:5008
Router2	telnet 127.0.0.1:5009
Router3	telnet 127.0.0.1:5010

Figure 12: Redundant topology

The traceroutes verify dynamic redundancy:

- Pre-failure: Direct route through R1-R2 (3 hops).

- Post-link disable: Traffic routes around via R3 (R2→R3→R1→ipterm-1), indicated by the new hop at 10.0.3.2 (R3). The asterisks (*) represent transient convergence delays, typical in actual failovers.

- R3's Role: Successful rerouting proves R3's RIPv2 MD5 authentication (key-ID/password sync with R1/R2), preventing unauthorized route propagation.

In conclusion, redundancy works. R3's authenticated contribution ensures smooth rerouting, with delivery of resilience specifications despite brief packet loss when re-convergence occurs.

```
root@ipterm-2:~# traceroute 192.168.1.2
traceroute to 192.168.1.2 (192.168.1.2), 30 hops max, 60 byte packets
 1  192.168.2.1 (192.168.2.1)  20.979 ms  31.931 ms  41.956 ms
 2  10.0.1.1 (10.0.1.1)  52.576 ms  75.561 ms  85.811 ms
 3  192.168.1.2 (192.168.1.2)  96.400 ms  107.171 ms  117.547 ms

# Disabling link R1 to R2 here...

root@ipterm-2:~# traceroute 192.168.1.2
traceroute to 192.168.1.2 (192.168.1.2), 30 hops max, 60 byte packets
 1  192.168.2.1 (192.168.2.1)  2.976 ms * *
 2  10.0.3.2 (10.0.3.2)  33.465 ms  44.046 ms  54.389 ms
 3  * * *
 4  * * *
```

```
5  * * *
6  * * *
7  * * *
8  * 192.168.1.2 (192.168.1.2)  64.579 ms  75.081 ms
```
Listing 9: traceroute proving redundancy between both hosts

4 Mission 2 - Inter-VLAN Routing Analysis

4.1 Inter-VLAN Traffic Analysis

For this section a different topology is used. There are four VPCS devices with two of them being a part of VLAN 20 and the other two a part of VLAN 30. The VLANs are implemented through the Dot1q standard. The VPCS devices are connected in ACCESS mode with the respective VLAN tags while router R1 is connected in TRUNK mode without a tag.

- TRUNK mode: Traffic from multiple VLANS is allowed simultaneously.
- ACCESS mode: Traffic from only one VLAN is allowed on the port. Hence it's suitable for end devices.

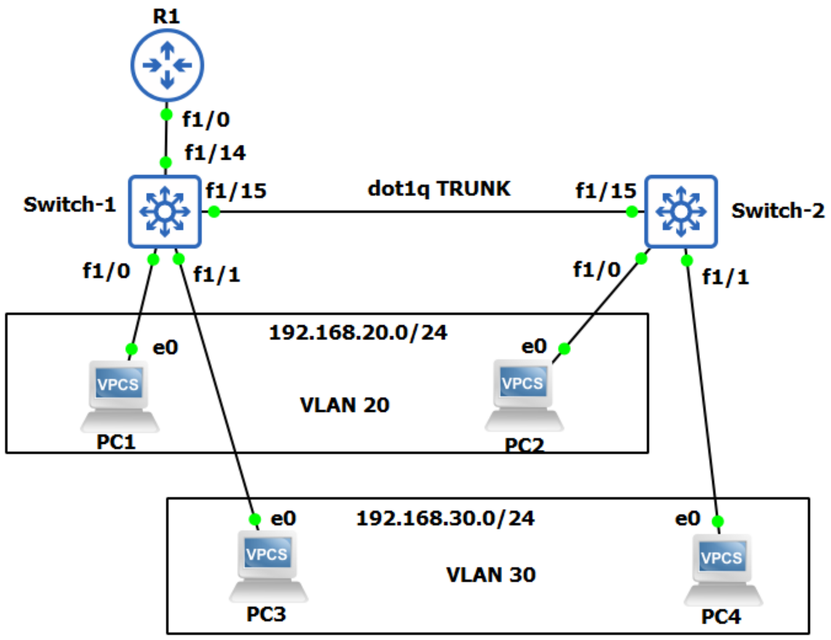


Figure 13: Inter-VLAN topology

Is important to take into account that switches are layer 2 devices, ie the way their configuration is only relevant for a local network, but once out in the internet the layer 2 configurations are invisible and if properly configured should be irrelevant. Since only one router is necessary to connect to the internet it makes sense to use switches to inter-connect devices inside the local subnetwork(s).

```
R1(config)# no ip routing
R1(config)# end
```
Listing 10: Setting up R1

```
Switch-1(config)# vlan database
Switch-1(config-vlan) # vlan 20
Switch-1(config-vlan) # vlan 30
Switch-1(config-vlan) # exit
```

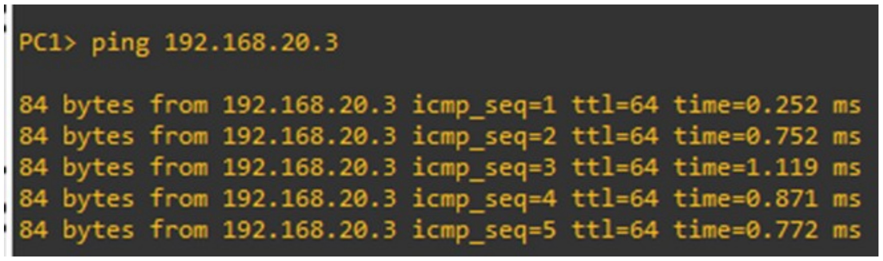
Listing 11: Setting up Switch-1

```
Switch-1(config)# interface fastEthernet 1/0
Switch-1(config-if) # switchport mode access
Switch-1(config-if) # switchport access vlan 20
Switch-1(config-if) # exit
Switch-1(config)# interface fastEthernet 1/1
Switch-1(config-if) # switchport mode access
Switch-1(config-if) # switchport access vlan 30
Switch-1(config-if) # exit
```

Listing 12: Configuring ports to support VLAN

```
Switch-1(config)# interface fastEthernet 1/15
Switch-1(config-if) # switchport mode trunk
Switch-1(config-if) # switchport trunk allowed vlan all
Switch-1(config-if) # no shutdown
Switch-1(config-if) # end
```

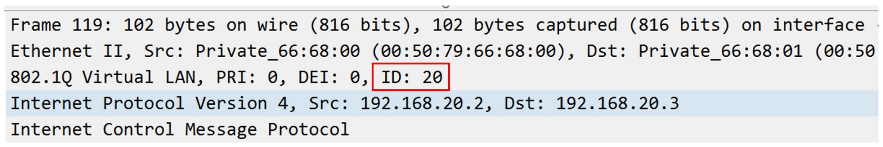
Listing 13: Allowing Trunking



```
PC1> ping 192.168.20.3

84 bytes from 192.168.20.3 icmp_seq=1 ttl=64 time=0.252 ms
84 bytes from 192.168.20.3 icmp_seq=2 ttl=64 time=0.752 ms
84 bytes from 192.168.20.3 icmp_seq=3 ttl=64 time=1.119 ms
84 bytes from 192.168.20.3 icmp_seq=4 ttl=64 time=0.871 ms
84 bytes from 192.168.20.3 icmp_seq=5 ttl=64 time=0.772 ms
```

Figure 14: Pinging PC2 from PC1

Trunking is using the same physical (or virtual) interface to address several subnetworks between switches. In this case the interface f1/15 of each switch is used to direct traffic between their connected PCs even though these switches are effectively connected in different subnetworks.
Once the trunking and subnetworks are properly set up it is possible to have communication within the network without any problem (since the router is not configured yet).Further examination of the packets allows us to identify the tag ID which indicates which VLAN the communication comes from.



```
Frame 119: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface
Ethernet II, Src: Private_66:68:00 (00:50:79:66:68:00), Dst: Private_66:68:01 (00:50
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 20
Internet Protocol Version 4, Src: 192.168.20.2, Dst: 192.168.20.3
Internet Control Message Protocol
```

Figure 15: Pinging PC2 from PC1

4.2 Inter-VLAN Routing Configuration and Verification

Now we finally configure the router to flow the traffic between the VLANs. The router here basically allows for inter-VLAN routing.

```
R1(config-)# interface fastEthernet 1/0
R1(config-if)# no shutdown
R1(config-if)# exit
R1(config-)# interface fastEthernet 1/0.20
R1(config-if)# encapsulation dot1Q 20
R1(config-if)# ip address 192.168.20.1 255.255.255.0
R1(config-if)# exit
R1(config-)# interface fastEthernet 1/0.30
R1(config-if)# encapsulation dot1Q 30
R1(config-if)# ip address 192.168.30.1 255.255.255.0
R1(config-if)# exit
```

Listing 14: Configuring R1

The command *interface fastEthernet 1/0.20* creates a subinterface, FastEthernet 1/0.20, under the physical interface FastEthernet 1/0 and the command *encapsulation dot1Q 20* specifies that the subinterface FastEthernet 1/0.20 will use IEEE 802.1Q encapsulation for VLAN tagging, with VLAN ID 20. This means that any traffic on this subinterface will be tagged as belonging to VLAN 20. The same is replicated for the other VLAN.
After setting up R1 we can now send pings from PC1 to the router.

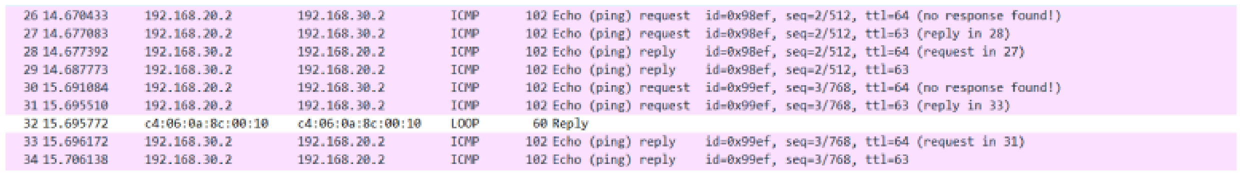
```
84 bytes from 192.168.20.1 icmp_seq=1 ttl=255 time=9.129 ms
84 bytes from 192.168.20.1 icmp_seq=2 ttl=255 time=10.947 ms
84 bytes from 192.168.20.1 icmp_seq=3 ttl=255 time=7.178 ms
84 bytes from 192.168.20.1 icmp_seq=4 ttl=255 time=8.879 ms
84 bytes from 192.168.20.1 icmp_seq=5 ttl=255 time=6.088 ms
```

Figure 16: Pinging router from PC1

```
84 bytes from 192.168.30.2 icmp_seq=1 ttl=63 time=32.095 ms
84 bytes from 192.168.30.2 icmp_seq=2 ttl=63 time=20.609 ms
84 bytes from 192.168.30.2 icmp_seq=3 ttl=63 time=15.068 ms
84 bytes from 192.168.30.2 icmp_seq=4 ttl=63 time=17.106 ms
84 bytes from 192.168.30.2 icmp_seq=5 ttl=63 time=16.882 ms
```

Figure 17: Pinging PC3 from PC1

With the router set up for inter-vlan routing, the packets travel from PC1 (192.168.20.2/24) to the router, where they are placed on the subnet of PC3 (192.168.30.0/24). From there they then make their way to PC3 (192.168.30.2/24). The result of this is that all packets traveling from VLAN 20 to VLAN 30 need to pass through the stick line between EtherSwitch1 and the router, which can limit the bandwidth. In terms of security inter-vlan routing allows one to set up a firewall in the router to control which packets are allowed to travel from one vlan network to the other. When we capture the packets flowing through the stick line, we see the following.



```
26 14.670433   192.168.20.2       192.168.30.2       ICMP   102 Echo (ping) request  id=0x98ef, seq=2/512, ttl=64 (no response found!)
27 14.677083   192.168.20.2       192.168.30.2       ICMP   102 Echo (ping) request  id=0x98ef, seq=2/512, ttl=63 (reply in 28)
28 14.677392   192.168.30.2       192.168.20.2       ICMP   102 Echo (ping) reply    id=0x98ef, seq=2/512, ttl=64 (request in 27)
29 14.687773   192.168.30.2       192.168.20.2       ICMP   102 Echo (ping) reply    id=0x98ef, seq=2/512, ttl=63
30 15.691084   192.168.20.2       192.168.30.2       ICMP   102 Echo (ping) request  id=0x99ef, seq=3/768, ttl=64 (no response found!)
31 15.695510   192.168.20.2       192.168.30.2       ICMP   102 Echo (ping) request  id=0x99ef, seq=3/768, ttl=63 (reply in 33)
32 15.695772   c4:06:0a:8c:00:10  c4:06:0a:8c:00:10  LOOP    60 Reply
33 15.696172   192.168.30.2       192.168.20.2       ICMP   102 Echo (ping) reply    id=0x99ef, seq=3/768, ttl=64 (request in 31)
34 15.706138   192.168.30.2       192.168.20.2       ICMP   102 Echo (ping) reply    id=0x99ef, seq=3/768, ttl=63
```

Figure 18: Wireshark Capture

We observe every ping request and reply 2 times, once when they are traveling towards the router, and once when they return. The four packets we see are

- ICMP Echo Request from VPCS-1 to Router (VLAN 20 subinterface): This packet is an ICMP Echo Request (ping) generated by PC1.
- Router forwards ICMP Echo Request to the appropriate subinterface (VLAN 30): The router checks the routing table and sees that the destination IP address (192.168.30.x) is on VLAN 30. It forwards the packet through the subinterface for VLAN 30.
- ICMP Echo Reply from VPCS-3 to Router (VLAN 30 subinterface): This is the ICMP Echo Reply sent by VPCS-3 back to the router.
- Router forwards ICMP Echo Reply to VPCS-1 (VLAN 20 subinterface): The router forwards the ICMP Echo Reply from VPCS-3 to VPCS-1.