CMPU 4050: Systems Integration
Week 2 — Networking Fundamentals and Routing

Today's Roadmap

1. TCP/IP Stack Review
   - Layers and their roles
   - How services use the stack
2. Packet Structure Deep Dive
   - Frames, packets, segments
   - Encapsulation process
3. Linux Networking Tools
   - The `ip` command suite
   - Testing and diagnostics
4. Routing and NAT
   - Routing tables and decisions
   - Configuring Linux as a router
   - IP masquerading
5. VirtualBox Networking
6. Practice Scenarios

Learning Outcomes

After this lecture you will:

Understand packet structure and encapsulation
Configure Linux networking and routing
Implement NAT for internet sharing
Troubleshoot network issues systematically

Assignment Relevance

Assignment 1: DNS packet structure
Assignment 2: Router configuration

The Core Problem of Networking

Fundamental Challenge

We have N computers that need to communicate reliably across different networks. The TCP/IP stack solves this systematically.

Layer Responsibilities:

Application	Services (DNS, DHCP, HTTP)
Transport	Process-to-process (TCP/UDP)
Network	Host-to-host routing (IP)
Link	Local network delivery (Ethernet)

Addressing at Each Layer:

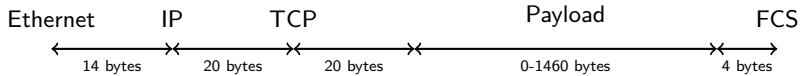
- Application: URLs, service names
- Transport: Port numbers (16-bit)
- Network: IP addresses (32-bit IPv4)
- Link: MAC addresses (48-bit)

Key Insight

Each layer solves one specific problem, adding its own headers to carry addressing information.

Packet Encapsulation - Structured View

Layer	Data Structure
Application	HTTP Request Data
Transport	TCP Header | HTTP Request Data
Network	IP Header | TCP Header | HTTP Request Data
Data Link	Ethernet | IP | TCP | Data | FCS



How the Layers Work Together

Application Layer: Communication between services
Transport Layer: Communication between processes via ports
Network Layer: Communication between hosts on different networks via IP
Link Layer: Communication on a single network segment via MAC

Key Concept

Each layer adds its own header (and sometimes trailer) to the data from the layer above, creating a nested structure that enables end-to-end communication across networks.

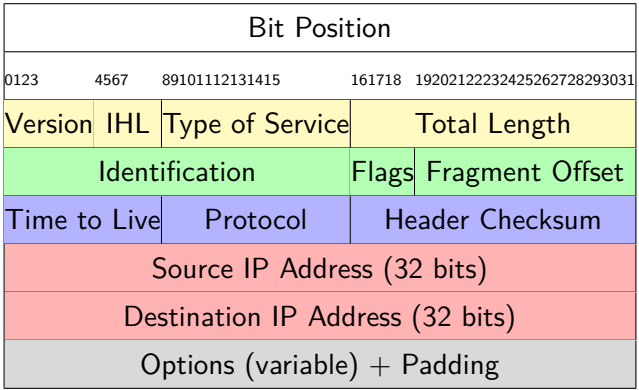
Ethernet Frame Structure

Ethernet II Frame Format				
Dest MAC	Src MAC	Type	Payload	FCS
6 bytes	6 bytes	2 bytes	46-1500 bytes	4 bytes

Field	Description
Destination MAC	Hardware address of recipient
Source MAC	Hardware address of sender
Type/Length	0x0800 = IPv4, 0x0806 = ARP, 0x86DD = IPv6
Payload	Higher layer protocol data
FCS	Frame Check Sequence (CRC-32)

- MAC addresses only work on local network (same subnet)
- EtherType tells receiver how to interpret payload

IP Packet Header Structure



Critical Fields:

- **TTL**: Prevents routing loops
- **Protocol**: 6=TCP, 17=UDP, 1=ICMP
- **Source/Dest IP**: 32-bit addresses

For Your Assignments:

- DNS uses Protocol=17 (UDP)
- Router modifies TTL on forward
- NAT changes Source IP

TCP Segment Structure

TCP Header (20 bytes minimum)	
Source Port (16 bits)	Destination Port (16 bits)
Sequence Number (32 bits)	
Acknowledgment Number (32 bits)	
Data Offset (4) | Reserved (3) | Flags (9)	Window Size (16 bits)
Checksum (16 bits)	Urgent Pointer (16 bits)
Options (variable)	
Data	

TCP Flags (9 bits): NS | CWR | ECE | URG | ACK | PSH | RST | SYN | FIN

- **Ports**: 16-bit service identifiers
- **Sequence**: Order reassembly
- **Flags**: SYN, ACK, FIN for connection

Common Ports to Know:

- 22: SSH 53: DNS
- 80: HTTP 443: HTTPS

The Encapsulation Process

Layer	Protocol Data Unit
Application	HTTP Request Data
Transport	TCP Hdr | HTTP Request Data
Network	IP Hdr | TCP | HTTP Data
Data Link	Eth Hdr | IP | TCP | Data | FCS

Layer	Added Headers	Key Information
Transport	Src Port: 54321, Dst Port: 443	Process identification
Network	Src IP: 192.168.1.100, Dst IP: 142.250.80.46	Host addressing
Data Link	Dst MAC: Router, Src MAC: Your NIC	Local delivery

This is what Assignment 1 does

Your DNS resolver: DNS data → UDP segment → IP packet → Socket send

Modern Network Management: The 'ip' Command

The `ip` command replaces legacy tools (`ifconfig`, `route`, `arp`):

View Configuration:

```
# Interfaces and addresses
ip addr show
ip -br addr  # brief

# Routing table
ip route show

# ARP cache (neighbors)
ip neigh show

# Link status
ip link show
```

Modify Configuration:

```
# Add IP address
sudo ip addr add \
  192.168.1.10/24 dev enp0s3

# Add route
sudo ip route add \
  10.0.0.0/24 via 192.168.1.1

# Bring interface up
sudo ip link set enp0s3 up
```

Making Network Changes Permanent

Ubuntu uses Netplan (YAML configuration):

Edit `/etc/netplan/01-netcfg.yaml`:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:               # First adapter (NAT)
      dhcp4: true
    enp0s8:               # Second adapter (Host-only)
      dhcp4: false
      addresses: [192.168.1.1/24]
      routes:
        - to: 10.0.0.0/24
          via: 192.168.1.254
```

Apply changes:

```
sudo netplan apply
```

Network Testing and Diagnostics

Systematic Troubleshooting Approach:

1. **Physical/Link Layer:**

```
ip link show          # Interface UP?
ethtool enp0s3       # Cable connected?
```

2. **Network Layer:**

```
ip addr show         # IP assigned?
ping -c 3 <gateway>   # Gateway reachable? [from ip route show]
```

3. **End-to-end:**

```
ping 8.8.8.8         # Internet connectivity?
nslookup google.com # DNS resolution?
```

4. **Service Level:**

```
nc -zv <host> 22       # Port open on <host> ip?
ss -tulpn            # Services listening?
```

Understanding Routing Tables

Every packet needs a routing decision based on destination IP:

```
$ ip route show
default via 192.168.1.1 dev enp0s3 proto dhcp metric 100
10.0.0.0/24 via 192.168.1.254 dev enp0s3
192.168.1.0/24 dev enp0s3 proto kernel scope link src 192.168.1.10
```

How Linux Makes Routing Decisions

1. Check for most specific match (longest prefix)
2. If no match, use default route
3. Determine output interface and next hop
4. Decrement TTL, forward packet

Assignment 2 Context

Your server must have routes for both client network and internet access

Enabling Linux as a Router

Three essential steps to make Linux route packets:

Step 1: Enable IP Forwarding

```
# Temporary (lost on reboot)
sudo sysctl -w net.ipv4.ip_forward=1

# Permanent - edit /etc/sysctl.conf:
net.ipv4.ip_forward=1

# Apply permanent changes
sudo sysctl -p
```

Step 2: Configure Routing Tables

```
# Ensure routes exist for both networks
ip route show  # Verify client and internet routes
```

Step 3: Set Up NAT (next slide)

Network Address Translation (NAT)

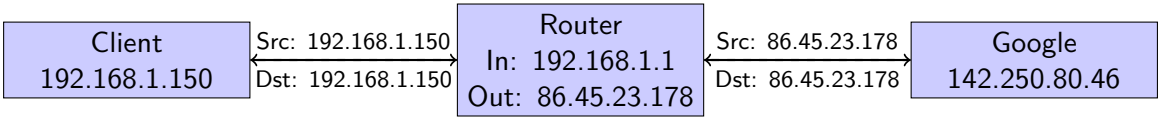
Private IP addresses (192.168.x.x) cannot be routed on the internet. NAT rewrites packet headers to use the router's public IP.

Configure NAT with iptables:

```
# Enable masquerading on outgoing interface
sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
# Allow forwarding between interfaces
sudo iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT
sudo iptables -A FORWARD -i enp0s3 -o enp0s8 \
  -m state --state RELATED,ESTABLISHED -j ACCEPT
# Save rules (Ubuntu)
sudo apt install iptables-persistent
sudo netfilter-persistent save
```

Outgoing: Changes source IP from 192.168.1.x to router's public IP
Incoming: Changes destination IP back to original private IP

IP Masquerading Visualized



NAT rewrites headers here

Assignment 2 Implementation

This is exactly what your server must do: enable clients with private IPs to access the internet

Understanding Linux Firewall Architecture

Netfilter/iptables Framework

Linux implements packet filtering through the Netfilter framework, controlled via iptables

Three Primary Tables:

- `filter`: Packet filtering (default)
- `nat`: Network address translation
- `mangle`: Packet alteration

Filter Table Chains:

- `INPUT`: Packets destined for host
- `FORWARD`: Packets being routed
- `OUTPUT`: Locally generated packets

Packet Flow Through Chains:

1. Incoming packet hits `INPUT` or `FORWARD`
2. Rules evaluated sequentially
3. First matching rule determines action
4. Default policy applies if no match

Rule Actions:

- `ACCEPT`: Allow packet
- `DROP`: Silently discard
- `REJECT`: Discard with error

Essential iptables Commands

```
iptables [-t table] -A CHAIN [criteria] -j ACTION
```

Viewing Current Rules:

```
# List all rules with line numbers
sudo iptables -L -n -v --line-numbers
# List NAT table rules
sudo iptables -t nat -L -n -v
```

Basic Firewall Rules:

```
# Allow established connections
sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# Allow SSH from specific network
sudo iptables -A INPUT -p tcp --dport 22 -s 192.168.1.0/24 -j ACCEPT
# Allow DNS queries
sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 53 -j ACCEPT
# Block specific IP address
sudo iptables -A INPUT -s 10.0.0.5 -j DROP
```

VirtualBox Network Adapter Types

NAT Mode

- VM gets internet
- Host cannot reach VM
- VMs isolated from each other
- Always 10.0.2.15
- Router at 10.0.2.2

Bridged Mode

- VM on physical network
- Gets IP from network DHCP
- Visible to all devices
- Like a real computer
- May not work on WiFi

Host-only Mode

- Private VM network
- Host can access VMs
- No internet access
- Typically 192.168.56.x
- Perfect for labs

Assignment 2 Architecture

- Server: Adapter 1 (NAT) + Adapter 2 (Host-only)
- Client: Adapter 1 (Host-only only)
- Server routes between networks

Troubleshooting Network Issues

Common Problems:

1. No connectivity
2. Can ping IP but not hostname
3. Local works, internet doesn't
4. Intermittent connection

Diagnostic Approach:

1. Check physical/link status
2. Verify IP configuration
3. Test gateway connectivity
4. Check DNS resolution

Key Takeaways

Conceptual Understanding:

- Packets are encapsulated at each layer
- Each layer adds addressing headers
- Routers work at Network layer (IP)
- NAT modifies packet headers

Practical Skills:

- Configure Linux networking with `ip`
- Enable and verify IP forwarding
- Implement NAT with iptables
- Troubleshoot systematically

Assignment Connections:

Assignment 1 (DNS Resolver):

- Creates UDP segments on port 53
- Encapsulates in IP packets
- System handles Ethernet framing

Assignment 2 (Router Config):

- Implements complete routing
- Configures NAT for clients
- Integrates multiple services

This Week's Lab and Next Week

Week 2 Lab: Networking and Routing

- Examine network configuration on your VM
- Use tcpdump to inspect packet headers
- Configure basic routing between interfaces
- Test NAT configuration
- Practice with different VirtualBox network modes

Week 3: DNS Part 1

Building on today's packet knowledge:

- How DNS queries traverse the network stack
- DNS protocol details and record types
- Configuring DNS clients and testing resolution
- Beginning Assignment 1 (DNS Resolver)

Quick Reference Card

Viewing:

```
ip addr show
ip route show
ip link show
ip neigh show
ss -tulpn
```

Testing:

```
ping -c 4 host
traceroute host
nc -zv host port
nslookup domain
tcpdump -i any
```

Configuration:

```
# Add IP
ip addr add IP/mask dev IF
# Add route
ip route add net via gw
# Enable forwarding
sysctl -w \
  net.ipv4.ip_forward=1
# Apply netplan
netplan apply
```

NAT Setup:

```
# Masquerade
iptables -t nat \
  -A POSTROUTING \
  -o OUT_IF -j MASQUERADE
# Forward
iptables -A FORWARD \
  -i IN_IF -j ACCEPT
# Save
netfilter-persistent save
```

Common Port Numbers

22 (SSH), 53 (DNS), 67-68 (DHCP), 80 (HTTP), 443 (HTTPS), 2049 (NFS)

What You'll See in tcpdump:

1. **Frame Level:** Ethernet II, Src: 08:00:27:xx:xx:xx, Dst: 52:54:00:xx:xx:xx
2. **Network Level:** Internet Protocol Version 4, Src: 192.168.1.10, Dst: 8.8.8.8
3. **Transport Level:** User Datagram Protocol, Src Port: 54321, Dst Port: 53
4. **Application Level:** Domain Name System (query)