Networking Refresher
Outline

• Network Architecture
  • Components
  • Functionalities
  • Packet switching

• Network Layers

• Basics of Routing
Network Architecture
Main Goal

Remote processes communicating with each other.

Hello! Send me index.html

Sure. Here it is!
Main Goal: Two Requirements

Remote processes communicating with each other.

Hello! Send me index.html

Sure. Here it is!

1 Network Infra.

2 APIs
What is the Internet? A Component View

- **Millions of connected computing devices:**
  - *hosts = end systems*
  - running *network apps*

- **Communication links**
  - fiber, copper, radio, satellite
  - transmission rate: *bandwidth*

- **Packet switches:** forward packets (chunks of data)
  - *routers and switches*
The Network Core

- Mesh of interconnected routers
- Packet switching: Lines are not reserved by connections
- *Store-and-forward*: Routers only forward packets when the whole packet is received
- What happens if a router receives/stores too much data?
  - Initially, packets are delayed (buffered)
  - Eventually, packets must be dropped

![Network Diagram]

- $R = 100$ Mb/s
- $R = 1.5$ Mb/s
- Queue of packets waiting for output link
Internet Structure: Network of networks!

• End systems connect to Internet via access ISPs (Internet Service Providers)
  • Residential, company and university ISPs

• Access ISPs in turn must be interconnected
  • So that any two hosts can send packets to each other

• Resulting network of networks is complex
  • Evolution was driven by economics and national policies

• CDNs can use their own networks outside of the system
Network Layers
Protocol Layers

• Every packet has a series of headers, one for each layer
• Headers are read by intermediate devices for routing/filtering decisions

<table>
<thead>
<tr>
<th>HTTP, FTP, ...</th>
<th>application</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP, UDP</td>
<td>transport</td>
</tr>
<tr>
<td>IP</td>
<td>network</td>
</tr>
<tr>
<td>Ethernet</td>
<td>link</td>
</tr>
<tr>
<td></td>
<td>physical</td>
</tr>
</tbody>
</table>
Data Link Layer

• Two devices that are not directly connected want to talk to each other
• The devices are identified by MAC addresses
• Instead, they are connected by *switches*
• Switches know MAC addresses and will forward packets to the right devices through the right port

Problems:
• Scaling: Switches can’t know every MAC address
• MAC addresses do not convey logical information about the network hierarchy
Network Layer (This lecture)

• IP addresses instead of MAC addresses for the wider Internet
  • Given destination IP, any router should be able to forward the packet towards the destination without knowing the whole path
• CIDR rules give logic to IP addresses to minimize routing table size
• Interior/exterior gateway protocols to route messages (more later)

Problem:
• No guarantee that packets arrive, no guarantee of order
• Congestion is an issue
• No distinction between different services on one end device
• No concept of “connections”
Transport Layer

- TCP, UDP, QUIC
- Enables (some of) the following features:
  - Port number to distinguish between applications, or multiple connections for the same application
  - Connection establishment
  - Flow control
  - Congestion control
  - Correct order of packets
  - Guarantee delivery of packets

Not in UDP

---

application
transport
network
link
physical
Encapsulation

message
segment
datagram
frame

source

application
transport
network
link
physical

link
physical

switch

destination

application
transport
network
link
physical

network
link
physical

router
TCP/IP Protocol Suite Summary

• **application**: supporting network applications
  • FTP, SMTP, HTTP

• **transport**: process-to-process data transfer
  • TCP, UDP

• **network**: routing of datagrams from source to destination
  • IP, routing protocols

• **link**: data transfer between neighboring network elements
  • Ethernet, 802.111 (WiFi), PPP

• **physical**: bits “on the wire”
Basics of Routing
Internet Routing

• “Flat” routing not suited for the Internet
  • Scalability (as the network size increases)
    • Space complexity $\rightarrow$ Each node cannot be expected to store routes to every destination (or destination network)
    • Convergence times increase
    • Communication $\rightarrow$ Total message count increases
  • Administrative autonomy
    • Each internetwork may want to run its network independently
      • E.g., hide topology information from competitors

• Solution: Hierarchy via autonomous systems (AS’s)
Today’s Internet

• Uses hierarchy of AS’s
• Each AS:
  • A set of routers under a single technical administration
  • **Intra-domain Routing**: Use an *interior gateway protocol (IGP)* and common metrics to route packets within the AS
  • **Inter-domain Routing**: Use an *exterior gateway protocol (EGP)* to route packets to other AS’s

• IGP: OSPF, RIP
• EGP: BGP
Interconnected AS’s

- Forwarding table is populated by IGPs and EGPs
  - **Interior gateway protocols (IGPs)** determine entries for destinations within AS
  - **IGPs and exterior gateway protocols (EGPs)** determine entries for external destinations
Interior Gateway Protocol: OSPF

• Link-state algorithm

• Router floods OSPF link-state advertisements to all other routers in entire AS
  • carried in OSPF messages directly over IP
  • includes neighbors, and bandwidth information (link cost)

• Each node independently computes a topology map
  • route computation using Dijkstra’s algorithm
Exterior Gateway Protocol

Suppose router in AS1 receives datagram destined outside of AS1:
• router should forward packet to gateway router, but which one?

AS1 must:
1. learn which destinations are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1
   Job of EGP!
Exterior Gateway Protocol: BGP

• BGP (Border Gateway Protocol):
  • the de-facto EGP

• Allows a subnet to advertise its existence to rest of Internet

• BGP provides each AS a means to:
  • eBGP: obtain subnet reachability information from neighboring ASes
  • iBGP: propagate reachability information to all AS-internal routers.

• Determines “good” routes to other networks based on reachability information and policy
An Example

https://bgpview.io/asn/11105#graph
Recall: Security Goals

• Confidentiality: what can routers (and wiretappers) see?
• Integrity: what can MITMs change? What can other end devices spoof?
• Availability: is end device connectivity ensured? Can someone be “knocked off” the Internet?
Sources of Network Vulnerabilities

- Protocol-level vulnerabilities
  - Implicit trust assumptions in design

- Implementation vulnerabilities
  - Both on routers and end-hosts

- Incomplete specifications
  - Often left to the programmers
An IP Packet Journey

[Map showing IP Packet Journey with cities such as VAN, EDM, WIN, REG, MON, TOR, HAL, and Alice]
An IP Packet Journey

VAN → EDM → WIN → TOR → MON → HAL

Alice
An IP Packet Journey

source: Alice
destination: SFU
data

VAN -> EDM -> REG -> WIN

MON -> TOR -> HAL

SFU
An IP Packet Journey

src: Alice

dst: SFU

data

VAN  EDM  REG  WIN  MON  TOR  HAL
An IP Packet Journey

src: Alice
dst: SFU
data
An IP Packet Journey

src: Alice
dst: SFU
data
An IP Packet Journey

src: Alice
dst: SFU
data
What happens between two routers
What happens between two routers

VAN Forwarding Table

<table>
<thead>
<tr>
<th>DST</th>
<th>OUT_IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>IF 3</td>
</tr>
<tr>
<td>SFU</td>
<td>IF 1</td>
</tr>
</tbody>
</table>

src: Alice
dst: SFU
data
What happens between two routers

Data Plane

VAN

src: Alice
dst: SFU
data

EDM

IF 1
IF 2
IF 3
IF 4
What happens between two routers

This is called **Packet Forwarding**
- moving packets from router’s **input** to appropriate router **output**
- done by the **data-plane** component

Forwarding happens by:
- examining the **destination address**, and
- matching it with a **local** forwarding table

But, who calculates the **forwarding tables**?
Routers Have “Brains”

This brain is called the Control Plane
Routers Have “Brains”

The control plane runs a routing algorithm to:
• find routes, and
• fill the tables

VAN Forwarding Table

<table>
<thead>
<tr>
<th>DST</th>
<th>OUT_IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>IF 3</td>
</tr>
<tr>
<td>SFU</td>
<td>IF 1</td>
</tr>
</tbody>
</table>

EDM Forwarding Table

<table>
<thead>
<tr>
<th>DST</th>
<th>OUT_IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>IF 4</td>
</tr>
<tr>
<td>SFU</td>
<td>IF 2</td>
</tr>
</tbody>
</table>
**Control Plane: Two Approaches**

**Distributed Approach**: routers exchange messages with each other to calculate the tables

- Examples: OSPF, IS-IS

---

**VAN**

- **Control Plane**
  - Routing algorithm

- **VAN Forwarding Table**
  - DST: Alice, OUT_IF: IF 3
  - DST: SFU, OUT_IF: IF 1

---

**EDM**

- **Control Plane**
  - Routing algorithm

- **EDM Forwarding Table**
  - DST: Alice, OUT_IF: IF 4
  - DST: SFU, OUT_IF: IF 2

---

**REG**

- **Control Plane**
  - Routing algorithm

- **REG Forwarding Table**
  - DST: Alice, OUT_IF: IF 4
  - DST: SFU, OUT_IF: IF 2

---
Control Plane: Two Approaches

Centralized Approach: routers exchange messages with an external software
- Software-defined networking (SDN)
- Examples: OpenFlow
Router Architecture Overview

Input Ports

1
2
\ldots
N

Routing Processor

Switching Fabric

Insert forwarding rules

Control Plane

Data Plane

Output Ports

1
2
\ldots
N
IP Overview
IP is the waist of the “hourglass”

- Multiple higher-layer protocols
  - Transport and Application

- Multiple lower-layer protocols
  - Link and Physical

- Single Internet protocol
  → No need to update routers and hosts every time we have a new service!

HTTP, FTP, DNS, SMTP, …
TCP, UDP, …
Ethernet, PPP, …
CSMA, SONET, …
Copper, fiber, radio

At every router/host
# IPv4 Datagram Format

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Octet</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet</td>
<td>Bit</td>
<td>0-3</td>
<td>4-7</td>
<td>8-15</td>
<td>16-18</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Version</td>
<td>Header Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>96</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>160</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24+</td>
<td>192+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **IPv4**
- **ICMP** 0x01
- **TCP** 0x06
- **UDP** 0x11
- **IPv6** 0x29

Header & data

Fragmentation

Addressing

E.g., TCP segment

*Min. header size is 20 bytes*
Time-to-live (TTL)

- Max. number of traversed hops
  - Before a datagram is dropped (Why?)

- TTL value is set by the source
  - Linux/Mac  64
  - Windows  128
  - Solaris, Cisco IOS  255

Loops!

Often used in OS Fingerprinting tools
Time-to-live (TTL)

• When a router receives an IP datagram:
  • If TTL is 0 → drop pkt
  • Decrement TTL by 1

• Does the router need to recalculate checksum?
IPv4 Addressing

• **IP address**: 32-bit identifier for host, router interface
• **Interface**: connection between host/router and physical link

• A router typically has multiple interfaces
• A host typically has one or two interfaces

*IP addresses/subnets are associated with each interface*
IPv4 Addressing

223.1.1.1 = 11011111 00000001 00000001 00000001

223.1.1.2
223.1.1.3
223.1.1.4
223.1.2.9
223.1.2.1
223.1.2.2
223.1.3.1
223.1.3.2
223.1.3.27

223.1.3.1
223.1.3.2

How are interfaces connected?
IPv4 Addressing

223.1.1.X

223.1.1.1
223.1.1.2
223.1.1.3

223.1.2.X

223.1.2.1
223.1.2.2

223.1.3.X

223.1.3.1
223.1.3.2
Subnets

• IP address:
  • subnet part: high order bits
  • host part: low order bits

• What’s a subnet?
  • device interfaces with same subnet part of IP address
  • can physically reach each other without intervening router

This network consists of three subnets
Subnets

• How many subnets?
  • 6

• Recipe
  • to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
  • each isolated network is called a subnet
IPv4 Addressing: CIDR

- CIDR: Classless Inter Domain Routing
- IP address is composed of
  - a subnet part (or prefix)
  - a host part (or suffix)
- Address format: a.b.c.d/x, where x is # bits in subnet portion of address (called mask)

200.23.16.0/24

- Subnet part: 11001000 00010111 00010000
- Host part: 00000000

24 bits = 32 - 24 = 8 bits
IPv4 Addressing: CIDR

200.23.16.0/24

<table>
<thead>
<tr>
<th>Subnet part (Prefix)</th>
<th>Host part (Suffix)</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000.00010111.00010000.00000000</td>
<td>00000000</td>
<td>200.23.16.0</td>
</tr>
<tr>
<td>11001000.00010111.00010000.00000001</td>
<td>00000001</td>
<td>200.23.16.1</td>
</tr>
<tr>
<td>11001000.00010111.00010000.00000010</td>
<td>00000010</td>
<td>200.23.16.2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001000.00010111.00010000.11111110</td>
<td>11111110</td>
<td>200.23.16.254</td>
</tr>
<tr>
<td>11001000.00010111.00010000.11111111</td>
<td>11111111</td>
<td>200.23.16.255</td>
</tr>
</tbody>
</table>

/24 bits means that we have 8 bits to address up to 256 hosts.
**IPv4 Addressing: CIDR**

In practice, the first and last IP addresses of a prefix are reserved. → /24 can support up to 254 (=256-2) hosts.

<table>
<thead>
<tr>
<th>Subnet part (Prefix)</th>
<th>Host part (Suffix)</th>
<th>IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000.00010111.00010000.</td>
<td>00000000</td>
<td>200.23.16.0</td>
</tr>
<tr>
<td>11001000.00010111.00010000.</td>
<td>11111111</td>
<td>200.23.16.255</td>
</tr>
</tbody>
</table>

Identifies the network (host part is all 0’s)

Identifies the broadcast address (host part is all 1’s)
How to get an IP address?

How does a host get IP address?

• Hard-coded by system admin in a file
• DHCP: Dynamic Host Configuration Protocol
  • dynamically get address from a server
How to get an IP address?

How does a network get IP address?

• Gets allocated portion of its provider ISP’s address space
How to get an IP address?

**Example:** Given an ISP network called 733 with address 200.23.16.0/20. How can it allocate IP addresses for 8 customer networks?

Use additional 3 bits to allocate addresses for the 8 customer networks.

<table>
<thead>
<tr>
<th>ISP 733 block</th>
<th>11001000 00010111 00010000 00000000</th>
<th>200.23.16.0/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization 0</td>
<td>11001000 00010111 00010000 00000000</td>
<td>200.23.16.0/23</td>
</tr>
<tr>
<td>Organization 1</td>
<td>11001000 00010111 00010010 00000000</td>
<td>200.23.18.0/23</td>
</tr>
<tr>
<td>Organization 2</td>
<td>11001000 00010111 00010100 00000000</td>
<td>200.23.20.0/23</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization 7</td>
<td>11001000 00010111 00011110 00000000</td>
<td>200.23.30.0/23</td>
</tr>
</tbody>
</table>
Hierarchical IP Addressing

- IP addresses are hierarchical

200.23.0.0/16
  
  This is ISP 733
  200.23.16.0/20
  
  ISP 733 is a customer of other provider

200.23.0.0/16 is ISP 733

Other eight customer networks of 200.23.18.0/23

The eight customer networks

- 200.23.16.0/23
- 200.23.18.0/23
- 200.23.20.0/23
- 200.23.30.0/23

- 200.23.18.0/26
- 200.23.18.64/26
- 200.23.18.192/26
- 200.23.19.0/26

...
Hierarchical IP Addressing

Hierarchical addressing allows efficient advertisement of routing information:

```
Send me anything with addresses beginning 200.23.16.0/20
```

```
Send me anything with addresses beginning 199.31.0.0/16
```
Hierarchical IP Addressing

- Hierarchical addressing allows efficient advertisement of routing information:

  200.23.16.0/23
  200.23.18.0/23
  200.23.30.0/23

  ISP 733

  199.31.0.0/16

  ISP 732

  "Send me anything with addresses beginning 200.23.16.0/20"

  "Send me anything with addresses beginning 199.31.0.0/16 or 200.23.18.0/23"

Organization 0

Organization 2

Organization 7

Organization 1 moves to ISP 732
Hierarchical IP Addressing

- Routers forward a packet to its destination based on the subnet part, not the host part
  - use **longest** address prefix that matches destination address
  - This is called the **longest prefix matching**

<table>
<thead>
<tr>
<th>DST</th>
<th>OUT_IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>200.23.18.0/23</td>
<td>IF2</td>
</tr>
<tr>
<td>200.23.16.0/20</td>
<td>IF1</td>
</tr>
<tr>
<td>199.31.0.0/16</td>
<td>IF2</td>
</tr>
</tbody>
</table>

Forwarding Table

- **src:** Alice
- **dst:** 200.23.16.5
- **data**
Hierarchical IP Addressing: Summary

• Scalable forwarding tables

• Adding/removing hosts without modifying forwarding table

• Small prefix advertisement overhead
Destination-based Forwarding

- Look-up is done at the input port
- IP routers forward packets by:
  - examining the destination address, and
  - matching it with a local forwarding table

They use the longest prefix matching algorithm
Generalized Forwarding

• Large-scale networks are complex
  • They don’t have “routers” networks.
  • They include middleboxes and other devices

• Network management becomes a hard task

• Network operators need a unified way to manage all of their network devices!

• One solution: Software-defined networks
Generalized Forwarding

- Each router contains a flow table that is computed and distributed by a logically centralized controller.
OpenFlow Data Plane Abstraction

• **Flow**: defined by header fields
• **Simple packet-handling rules**
  • **Pattern**: match values in packet header fields
  • **Actions**: (for a matched pkt)
    • drop, forward, modify a matched packet or send matched packet to controller
  • **Priority**: disambiguate overlapping patterns
  • **Counters**: #bytes and #packets

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.**.*.* → send to controller
Questions?
Extras
IPv4 Fragmentation

• Different link layer protocols have different MTU
  • Maximum transmission unit

• A router can break a datagram into fragments
  • If MTU of outgoing link is less than pkt size

• A destination reassembles IP fragments
  • To be delivered to transport layer
    • Why is the reassembly done at destinations?
IPv4 Fragmentation

• Example:

Router A
MTU = 4000

4000 bytes

Router B
MTU = 1500

1500 bytes
1500 bytes
1040 bytes

Fragment 1
Fragment 2
Fragment 3
IPv4 Fragmentation

• Example:

![Diagram](image_url)

- Fragment 1: 1480 bytes
- Fragment 2: 1480 bytes
- Fragment 3: 1020 bytes
IPv4 Fragmentation

• Issues
  • All fragments must be delivered to the destination → not guaranteed!
  • Last fragment may have non-optimal size → wasting router resources
  • Destination needs to hold IP fragments in memory
  • Only first datagram contains TCP/UDP header
    • Firewalls and other network functions don’t work well with IP fragments

• In the current Internet, fragmentation is not recommended
• IPv6 does not support fragmentation
IPv6

• Initial motivation:
  • 32-bit address space soon to be completely allocated.

• Additional motivation:
  • header format helps speed processing/forwarding
  • header changes to facilitate QoS

• IPv6 datagram format:
  • fixed-length 40-byte header
  • no fragmentation allowed
IPv6 Datagram Format

<table>
<thead>
<tr>
<th>Octet</th>
<th>Bit</th>
<th>0-3</th>
<th>4-7</th>
<th>8-11</th>
<th>12-15</th>
<th>16-23</th>
<th>24-31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>Version</td>
<td>Traffic Class</td>
<td>Flow Label</td>
<td>Payload Length</td>
<td>Next Header</td>
<td>Hop Limit</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>96</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>160</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>192</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>224</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>288</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Priority/Traffic Class:** identify priority among datagrams in flow

**Flow Label:** identify datagrams in same “flow”

**Next header:** identify upper layer protocol for data
Other Changes

• **Checksum**: removed entirely to reduce processing time at each hop

• **Options**: allowed, but outside of header, indicated by “Next Header” field

• **No Fragmentation**:  
  • Packet is dropped if its size is larger than outgoing link MTU  
  • An error message is sent to the sender
IPv4 → IPv6

• Not all routers can be upgraded simultaneously
  • how will network operate with mixed IPv4 and IPv6 routers?

• Tunneling:
  • IPv6 datagram carried as payload in IPv4 datagram among IPv4 routers
IPv4 $\rightarrow$ IPv6: Tunneling

Logical View

A IPv6

B IPv6

IPv4 tunnel connecting IPv6 routers

E IPv6

F IPv6

Physical View

A IPv6

B IPv6

C IPv4

D IPv4

E IPv6

F IPv6
IPv6 Deployment

- It is hard to change the network-layer protocols!
- IPv6 was first introduced in 1995!

**Percentage of users accessing Google using IPv6**
Switching Fabric

• Transfer packet from input buffer to appropriate output buffer
• Switching rate: rate at which packets can be transfer from inputs to outputs
• Three types of switching fabrics
Input Port Functions

**Physical layer:** bit-level reception

**Data-link layer:** e.g., Ethernet

**Switching:**
- Using header field values, lookup output port using forwarding table in input port memory ("match plus action")
- **Goal:** complete input port processing at ‘line speed’
- **Queuing:** if datagrams arrive faster than forwarding rate into switch fabric
Input Port Queuing

• Fabric slower than input ports combined → queuing may occur at input queues
  • queuing delay and loss due to input buffer overflow!

• Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

Output port contention: only one red datagram can be transferred. *lower red packet is blocked*

One packet time later: green packet experiences HOL blocking
Output Ports

- **Buffering** required when packets arrive from fabric faster than the transmission rate
  - Packets can be lost due to congestion, lack of buffers

- **Scheduling discipline** chooses among queued packets for transmission
  - Priority scheduling – who gets best performance, network neutrality
Output Port Queuing

• Buffering when arrival rate via switch exceeds output line speed

• Queueing (delay) and loss due to output port buffer overflow!
Scheduling Policy

• **Scheduling**: choose next packet to send on link

• FIFO (first in first out) scheduling: send in order of arrival to queue
  • **discard policy**: if packet arrives to full queue: who to discard?
    • **tail drop**: drop arriving packet
    • **priority**: drop/remove on priority basis
    • **random**: drop/remove randomly
Scheduling Policy: Priority

• Priority scheduling: send highest priority queued packet
  • multiple classes, with different priorities
  • class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
Scheduling Policy: Other Policies

• Round Robin (RR):
  • multiple classes
  • cyclically scan class queues, sending one complete packet from each class (if available)

• Weighted Fair Queuing (WFQ):
  • generalized Round Robin
  • each class gets weighted amount of service in each cycle
Longest Prefix Matching

- Longest prefix matching: often performed using ternary content addressable memories (TCAMs)

- Content addressable:
  - present address to TCAM
  - retrieve address in one clock cycle, regardless of table size

```
<table>
<thead>
<tr>
<th>Mem. Address</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
</tr>
</tbody>
</table>
```
Longest Prefix Matching

- Longest prefix matching: often performed using ternary content addressable memories (TCAMs)
- Content addressable:
  - present address to TCAM
  - retrieve address in one clock cycle, regardless of table size

![Diagram of TCAM and SRAM with entries and search results]
Longest Prefix Matching

- Longest prefix matching: often performed using ternary content addressable memories (TCAMs)
- Content addressable:
  - present address to TCAM
  - retrieve address in one clock cycle, regardless of table size

```
Entry 1
1 1 0 X
Entry 2
1 1 X X
Entry 3
1 0 X X
Entry 4
X X X X
```

```
Search for: 1 0 1 0
```

```
Mem. Address | Port
-------------|------
00           | 1
01           | 3
10           | 5
11           | 2
```

```
Entry 2
```

```
Entry 3
```

```
Entry 4
```

```
Entry 1
```

```
Entry 2
```

```
Entry 3
```

```
Entry 4
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
TCAM Advantages and Disadvantages

• Advantages:
  • Simpler than other (trie-based) algorithms
  • Read operation is done in one clock cycle

• Disadvantages:
  • Requires larger chip area
    • E.g., a typical SRAM cell contains 6T, while a TCAM cell contains 16T!
  • High power consumption
OpenFlow: Flow Table Entries

<table>
<thead>
<tr>
<th>Rule</th>
<th>Action</th>
<th>Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch Port</td>
<td>VLAN ID</td>
<td>MAC src</td>
</tr>
<tr>
<td></td>
<td>MAC dst</td>
<td>Eth type</td>
</tr>
<tr>
<td></td>
<td>IP Src</td>
<td>IP Dst</td>
</tr>
<tr>
<td></td>
<td>IP Prot</td>
<td>TCP sport</td>
</tr>
<tr>
<td></td>
<td>TCP dport</td>
<td>Packet + byte counters</td>
</tr>
</tbody>
</table>

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify Fields

Link layer | Network layer | Transport layer
OpenFlow: Examples

Destination-based forwarding:

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>51.6.0.8</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port6</td>
</tr>
</tbody>
</table>

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Forward</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>drop</td>
</tr>
</tbody>
</table>

do not forward (block) all datagrams destined to TCP port 22

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Forward</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>128.119.1.1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>drop</td>
</tr>
</tbody>
</table>

do not forward (block) all datagrams sent by host 128.119.1.1
OpenFlow Abstraction

• **Match+action**: unifies different kinds of devices

  ▪ **Router**
    • *match*: longest destination IP prefix
    • *action*: forward out a link
  
  ▪ **Switch**
    • *match*: destination MAC address
    • *action*: forward or flood
  
  ▪ **Firewall**
    • *match*: IP addresses and TCP/UDP port numbers
    • *action*: permit or deny
  
  ▪ **NAT**
    • *match*: IP address and port
    • *action*: rewrite address and port