Assignment 1

- Hard deadline: NO EXTENSIONS
- IMPORTANT: Make sure you test your program on CSE machines as per the demo setup
- Demonstration Schedule for Week 8 is linked to assignment page
  - YOU MUST ATTEND YOUR ALLOCATED SLOT
- MONDAY: PUBLIC HOLIDAY
  - Demo in Week 9
  - Monday lab schedule will be pushed back by one week
Chapter 4: network layer

Our goals:

- understand principles behind network layer services:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - routing (path selection)
  - broadcast, multicast

- instantiation, implementation in the Internet
Network Layer: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   - datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 routing algorithms
   - link state
   - distance vector
   - hierarchical routing
4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 broadcast and multicast routing
Some background

- 1968: DARPA/ARPAnet (precursor to Internet)
  - (Defense) Advanced Research Projects Agency Network

- Mid 1970’s: new networks emerge
  - SATNet, Packet Radio, Ethernet
  - All “islands” to themselves – didn’t work together

- Big question: How to connect these networks?
Internetworking

- Cerf & Kahn in 1974,
  - “A Protocol for Packet Network Intercommunication”
  - Foundation for the modern Internet

- **Routers** forward **packets** from source to destination
  - May cross many separate networks along the way

- All packets use a common **Internet Protocol**
  - Any underlying data link protocol
  - Any higher layer transport protocol
Network layer

- transport segment from sending to receiving host
- on sending side, encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in every host, router
- router examines header fields in all IP datagrams passing through it
Two key network-layer functions

- **forwarding**: move packets from router’s input to appropriate router output

- **routing**: determine route taken by packets from source to dest.
  - **routing algorithms**

  **analogy:**
  - **routing**: process of planning trip from source to dest
  - **forwarding**: process of getting through single interchange
Quiz: When should a router perform routing? Forwarding

A: Do both when a packet arrives

B: Route in advance, forward when a packet arrives

C: Forward in advance, route when a packet arrives

D: Do both in advance

E: Some other combination
Interplay between routing and forwarding

Routing algorithm determines end-end-path through network.

Forwarding table determines local forwarding at this router.

<table>
<thead>
<tr>
<th>header value</th>
<th>output link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100</td>
<td>3</td>
</tr>
<tr>
<td>0101</td>
<td>2</td>
</tr>
<tr>
<td>0111</td>
<td>2</td>
</tr>
<tr>
<td>1001</td>
<td>1</td>
</tr>
</tbody>
</table>

Value in arriving packet’s header:

Network Layer
Connection setup

- 3rd important function in some network architectures:
  - ATM, frame relay, X.25
- before datagrams flow, two end hosts and intervening routers establish virtual connection
  - routers get involved
- network vs transport layer connection service:
  - network: between two hosts (may also involve intervening routers in case of VCs)
  - transport: between two processes
Network service model

Q: What service model for “channel” transporting datagrams from sender to receiver?

deleted

delay

delay

deleted

example services for individual datagrams:
- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

deleted

deleted

example services for a flow of datagrams:
- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing
## Network layer service models:

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Service Model</th>
<th>Guarantees?</th>
<th>Congestion feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bandwidth</td>
<td>Loss</td>
</tr>
<tr>
<td>Internet</td>
<td>best effort</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>ATM</td>
<td>CBR</td>
<td>constant rate</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed rate</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed minimum</td>
<td>no</td>
</tr>
<tr>
<td>ATM</td>
<td>UBR</td>
<td>none</td>
<td>no</td>
</tr>
</tbody>
</table>
Network Layer: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   - datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 routing algorithms
   - link state
   - distance vector
   - hierarchical routing
4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 broadcast and multicast routing
Connection, connection-less service

- **datagram** network provides network-layer *connectionless* service
- **virtual-circuit** network provides network-layer *connection* service
- analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
  - *service*: host-to-host
  - *no choice*: network provides one or the other
  - *implementation*: in network core
Virtual circuits

“source-to-dest path behaves much like telephone circuit”

- performance-wise
- network actions along source-to-dest path

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- every router on source-dest path maintains “state” for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)
VC implementation

A VC consists of:

1. *path* from source to destination
2. *VC numbers*, one number for each link along path
3. *entries in forwarding tables* in routers along path

- Packet belonging to VC carries VC number (rather than dest address)
- VC number can be changed on each link.
  - New VC number comes from forwarding table
### VC forwarding table

**forwarding table in Router R1:**

<table>
<thead>
<tr>
<th>Incoming interface</th>
<th>Incoming VC #</th>
<th>Outgoing interface</th>
<th>Outgoing VC #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>1</td>
<td>97</td>
<td>3</td>
<td>87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**VC routers maintain connection state information!**
Virtual Circuit Setup - Example

ATM Adaptation Layer (AAL): “adapts” upper layers (IP or native ATM applications) to ATM layer below.

AAL present only in end systems, not in switches.

AAL layer segment (header/trailer fields, data) fragmented across multiple ATM cells.

Analogy: TCP segment in many IP packets.

ATM Adaptation Layer (AAL) [more]

Different versions of AAL layers, depending on ATM service class:
- AAL1: for CBR (Constant Bit Rate) services, e.g. circuit emulation
- AAL2: for VBR (Variable Bit Rate) services, e.g., MPEG video
- AAL5: for data (e.g., IP datagrams)

ATM Layer Service:
- transport cells across ATM network
- analogous to IP network layer
- very different services than IP network layer

Network Architecture

<table>
<thead>
<tr>
<th>ATM Service</th>
<th>best effort</th>
<th>CBR</th>
<th>VBR</th>
<th>ABR</th>
<th>UBR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>none</td>
<td>constant rate</td>
<td>guaranteed rate</td>
<td>guaranteed rate</td>
<td>minimum</td>
</tr>
<tr>
<td>Loss</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Order</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Timing</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Congestion</td>
<td>feedback</td>
<td>no  (inferred via loss)</td>
<td>no congestion</td>
<td>no congestion</td>
<td>yes</td>
</tr>
</tbody>
</table>

Guarantees?

ATM Layer: Virtual Circuits

VC transport: cells carried on VC from source to destination.

Call setup, teardown for each call before data can flow.

Each packet carries VC identifier (not destination ID).

Every switch on source-destination path maintain “state” for each passing connection.

Link, switch resources (bandwidth, buffers) may be allocated to VC: to get circuit-like performance.

Permanent VCs (PVCs): long lasting connections, typically: “permanent” route between to IP routers.

Switched VCs (SVC): dynamically set up on per-call basis.

Network Layer

Self Study
ATM Adaptation Layer (AAL): "adapts" upper layers (IP or native ATM applications) to ATM layer below.

AAL present only in end systems, not in switches.

AAL layer segment (header/trailer fields, data) fragmented across multiple ATM cells.

Analogy: TCP segment in many IP packets.

Different versions of AAL layers, depending on ATM service class:

- AAL1: for CBR (Constant Bit Rate) services, e.g., circuit emulation
- AAL2: for VBR (Variable Bit Rate) services, e.g., MPEG video
- AAL5: for data (e.g., IP datagrams)

ATM Layer: Virtual Circuits

VC transport: cells carried on VC from source to destination.

Call setup, teardown for each call before data can flow.

Each packet carries VC identifier (not destination ID).

Every switch on source-dest path maintain "state" for each passing connection.

Link, switch resources (bandwidth, buffers) may be allocated to VC: to get circuit-like performance.

Permanent VCs (PVCs): long lasting connections, typically "permanent" route between IP routers.

Switched VCs (SVC): dynamically set up on a per-call basis.

Network Architecture:

Service Model:

- Best effort
- CBR
- VBR
- ABR
- UBR

Bandwidth:

- None
- Constant rate
- Guaranteed rate
- Guaranteed minimum
- None

Loss:

- No
- Yes
- Yes
- Yes
- No

Order:

- No
- Yes
- Yes
- Yes
- Yes

Timing:

- No
- Yes
- Yes
- No
- No

Congestion:

- Feedback
- No (inferred via loss)
- No
- Congestion
- Yes
- No

Guarantees?
Example (contd.)

Network Layer 21
Example (contd.)

Switch Y chooses to use VC # 3 for this connection on the link connecting Host A to Switch Y.
Why? – VC #'s 0, 1 and 2 may be already in use.
Example (contd.)

Network Layer
Switch Z chooses to use VC # 12 for this connection on the link connecting Switch Y to Switch Z.

Note: VC # on each link are chosen independently.
Example (contd.)

```
<table>
<thead>
<tr>
<th>In VCI</th>
<th>In port</th>
<th>Out port</th>
<th>Out VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
```

**Network Layer 25**
Host B chooses to use VC # 7 for this connection on the link connecting Switch Z to Host B
Example (contd.)

ACK message informs upstream switch about the VC # to use on out-going link.
Example (contd.)

```
<table>
<thead>
<tr>
<th>In VCI</th>
<th>In port</th>
<th>Out port</th>
<th>Out VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```

**Network Layer**
Example (contd.)

![Diagram with network nodes and tables]

Host A

Switch Z

Host B

Y's table

<table>
<thead>
<tr>
<th>In VCI</th>
<th>In port</th>
<th>Out port</th>
<th>Out VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Z's table

<table>
<thead>
<tr>
<th>In VCI</th>
<th>In port</th>
<th>Out port</th>
<th>Out VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

Network Layer 29
Example (contd.)

**Y’s table**

<table>
<thead>
<tr>
<th>In VCI</th>
<th>In port</th>
<th>Out port</th>
<th>Out VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td>12</td>
</tr>
</tbody>
</table>

**Z’s table**

<table>
<thead>
<tr>
<th>In VCI</th>
<th>In port</th>
<th>Out port</th>
<th>Out VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

Network Layer
Example (contd.)

VC Setup is now complete

ACK message informs sender (Host A) about the VC # to be used for this connection

Network Layer
Example (contd.)

Data Transmission begins

<table>
<thead>
<tr>
<th>Y's table</th>
<th>Z's table</th>
</tr>
</thead>
<tbody>
<tr>
<td>In VCI</td>
<td>In port</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

I love you! 3

Network Layer 32
Example (contd.)

Network Layer 33
Example (contd.)

Addresses are only used for connection set-up ("signalling"). Virtual Circuit Identifiers (VCIs) are used for sending messages and for "teardown." This doesn't scale! Every connection needs an entry in every switch along the way! Or does it?

Addressing properties

<table>
<thead>
<tr>
<th>In VCI</th>
<th>In port</th>
<th>Out port</th>
<th>Out VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>In VCI</th>
<th>In port</th>
<th>Out port</th>
<th>Out VCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td>12</td>
</tr>
</tbody>
</table>

Y’s table

Z’s table

Host A

Switch Z

Host B

Self Study
Example (contd.)

Addresses are only used for connection set-up (“signalling”)

Virtual Circuit Identifiers (VCIs) are used for sending messages and for “teardown”

This doesn’t scale! Every connection needs an entry in every switch along the way!

Or does it?

Network Layer
Example (contd.)

Addresses are only used for connection set-up ("signalling"). Virtual Circuit Identifiers (VCIs) are used for sending messages and for "teardown". This doesn't scale! Every connection needs an entry in every switch along the way! Or does it?

Network Layer  36
Example (contd.)

In VCI | In port | Out port | Out VCI
-------|---------|----------|--------
 12     |   0     |    3     |    7   
 3      |   0     |    4     |    12  

I love you! 7

Addressing properties

- Addresses are only used for connection set-up (“signalling”)
- Virtual Circuit Identifiers (VCIs) are used for sending messages and for “teardown”
- This doesn’t scale! Every connection needs an entry in every switch along the way!
- Or does it?
Virtual circuits: signaling protocols

- used to setup, maintain, teardown VC
- used in ATM, frame-relay, X.25
- not widely used in today’s Internet

1. initiate call
2. incoming call
3. accept call
4. call connected
5. data flow begins
6. receive data
Virtual Circuits in Action

- MPLS (Multi-protocol Label Switching): RFC 3031
  - Packets pre-fixed with "labels"
    - A 20-bit label value
    - a 3-bit Traffic Class field for Quality of Service (QoS) priority and ECN (Explicit Congestion Notification)
    - a 1-bit bottom of stack flag.
      - If this is set, it signifies that the current label is the last in the stack
    - 8-bit TTL field
  - Labels can be stacked on top of another
  - Virtual circuit (label-switched path) established between Label Edge Routers (LERs)
  - Often used to setup Virtual Private Networks (VPN)
**Datagram networks**

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of “connection”
- packets forwarded using destination host address
Datagram forwarding table

4 billion IP addresses, so rather than list individual destination address list range of addresses (aggregate table entries)

IP destination address in arriving packet’s header
<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 00000000 through 11001000 00010111 00010111 11111111</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

Q: but what happens if ranges don’t divide up so nicely?
Longest prefix matching

When looking for forwarding table entry for given destination address, use longest address prefix that matches destination address.

<table>
<thead>
<tr>
<th>Destination Address Range</th>
<th>Link interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001000 00010111 00010*** ******</td>
<td>0</td>
</tr>
<tr>
<td>11001000 00010111 00011000 ******</td>
<td>1</td>
</tr>
<tr>
<td>11001000 00010111 00011*** ******</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

Examples:

DA: 11001000 00010111 00010110 10101010
which interface?

DA: 11001000 00010111 00011000 10101010
which interface?
Datagram or VC network: why?

Internet (datagram)
- data exchange among computers
  - “elastic” service, no strict timing req.
- many link types
  - different characteristics
  - uniform service difficult
- “smart” end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at “edge”

ATM (VC)
- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- “dumb” end systems
  - telephones
  - complexity inside network
Quiz: Connection state

- Which of the following relies on connection state in routers in the network?
  - Pick one.
  - A. TCP
  - B. Internet
  - C. Virtual circuit network
  - D. UDP
  - E. A and C
Quiz: Virtual circuit

- Which of the following is true? Pick one.

A. A virtual circuit uses a different VC number for each link along a route
B. A virtual circuit uses the same VC number for all packets in a connection
C. A virtual circuit router uses the destination address (among other fields) in order to determine the outgoing interface
D. A and C
Quiz: Longest prefix matching

On which outgoing interface will a packet destined to 11011001 be forwarded?

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>1*</td>
<td>A</td>
</tr>
<tr>
<td>11*</td>
<td>B</td>
</tr>
<tr>
<td>111*</td>
<td>C</td>
</tr>
<tr>
<td>1111*</td>
<td>D</td>
</tr>
<tr>
<td>Default</td>
<td>D</td>
</tr>
</tbody>
</table>
Network Layer: outline

4.1 introduction
4.2 virtual circuit and datagram networks
4.3 what’s inside a router
4.4 IP: Internet Protocol
   - datagram format
   - IPv4 addressing
   - ICMP
   - IPv6
4.5 routing algorithms
   - link state
   - distance vector
   - hierarchical routing
4.6 routing in the Internet
   - RIP
   - OSPF
   - BGP
4.7 broadcast and multicast routing
IP Routers

- Core building block of the Internet infrastructure
- $120B+ industry
- Vendors: Cisco, Huawei, Juniper, Alcatel-Lucent (account for >90%)
Router definitions

- N = number of external router "ports"
- R = speed ("line rate") of a port
- Router capacity = N x R

N = number of external router "ports"
R = speed ("line rate") of a port
Router capacity = N x R
Networks and routers

- UNSW
- home, small business
- Optus
- core
- edge (ISP)
- IBM
- edge (enterprise)
- UoW

Network Layer
Examples of routers (core)

**Juniper T4000**
- $R = 10/40 \text{ Gbps}$
- $NR = 4 \text{ Tbps}$

**Cisco CRS**
- $R = 10/40/100 \text{ Gbps}$
- $NR = 322 \text{ Tbps}$

72 racks, 1MW
Examples of routers (edge)

Cisco ASR 1006
- R=1/10 Gbps
- NR = 40 Gbps

Juniper M120
- R= 2.5/10 Gbps
- NR = 120 Gbps
Examples of routers (small business)

Cisco 3945E
- R = 10/100/1000 Mbps
- NR < 10 Gbps
What’s inside a router?

Processes packets on their way in

Route/Control Processor

Processes packets before they leave

Transfers packets from input to output ports

Linecards (input)

1

2

N

Interconnect (Switching) Fabric

Linecards (output)

1

2

N

Input and Output for the same port are on one physical linecard

Network Layer
What’s inside a router?

1. Implement IGP and BGP protocols; compute routing tables
2. Push forwarding tables to the line cards

Diagram:
- Linecards (input)
  - 1
  - 2
  - N
- Interconnect (Switching) Fabric
- Route/Control Processor
- Linecards (output)
  - 1
  - 2
  - N

Network Layer
What’s inside a router?

Constitutes the control plane

Constitutes the data plane

Route/Control Processor

Interconnect Fabric

Linecards (input)

Linecards (output)

Network Layer
# Input Linecards

<table>
<thead>
<tr>
<th>Version</th>
<th>Header Length</th>
<th>Type of Service (TOS)</th>
<th>Total Length (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16-bit Identification</th>
<th>Flags</th>
<th>Fragment Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time to Live (TTL)</th>
<th>Protocol</th>
<th>Header Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source IP Address</th>
<th>Destination IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options (if any)</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Network Layer 58
Input Linecards

- **Tasks**
  - Receive incoming packets (physical layer stuff)
  - Update the IP header
    - TTL, Checksum, Options (maybe), Fragment (maybe)
  - Lookup the output port for the destination IP address
  - Queue the packet at the switch fabric

- **Challenge:** speed!
  - 100B packets @ 40Gbps → new packet every 20 nano secs!

- **Typically implemented with specialized hardware**
  - ASICs, specialized “network processors”
  - “exception” processing often done at control processor
IP Lookup

- IP addressing and routing (to be discussed)
  - For scalability, multiple IP addresses are aggregated
  - Routing protocols operate on IP address prefixes ("a.b.c.d/n" notation)
  - IP routing tables maintain a mapping from IP prefixes to output interfaces

- Route lookup → find the longest prefix in the table that matches the packet destination address
  - Longest Prefix Match (LPM) lookup
Longest Prefix Match Lookup

- Packet with destination address 12.82.100.101 is sent to interface 2, as 12.82.100.xxx is the longest prefix matching packet’s destination address.
Example #1: 4 Prefixes, 4 Ports

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>201.143.0.0/22</td>
<td>Port 1</td>
</tr>
<tr>
<td>201.143.4.0/24</td>
<td>Port 2</td>
</tr>
<tr>
<td>201.143.5.0/24</td>
<td>Port 3</td>
</tr>
<tr>
<td>201.143.6.0/23</td>
<td>Port 4</td>
</tr>
</tbody>
</table>
Finding the Match

Consider 11001001100011110000010111010010

- First 21 bits match 4 partial prefixes
- First 22 bits match 3 partial prefixes
- First 23 bits match 2 partial prefixes
- First 24 bits match exactly one full prefix
Finding Match Efficiently

- Testing each entry to find a match scales poorly
  - On average: $O(\text{number of entries})$

- Leverage tree structure of binary strings
  - Set up tree-like data structure

- Return to example:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>11001001100011111000000000000</td>
<td>1</td>
</tr>
<tr>
<td>110010011000111110000000010000</td>
<td>2</td>
</tr>
<tr>
<td>110010011000111110000000010100</td>
<td>3</td>
</tr>
<tr>
<td>1100100110001111100000000111111</td>
<td>4</td>
</tr>
</tbody>
</table>
Consider four three-bit prefixes

- Just focusing on the bits where all the action is….

- 0** ➔ Port 1
- 100 ➔ Port 2
- 101 ➔ Port 3
- 11* ➔ Port 4
Tree Structure

0** \rightarrow \text{Port 1}

100 \rightarrow \text{Port 2}

101 \rightarrow \text{Port 3}

11* \rightarrow \text{Port 4}

Network Layer 66
Walk Tree: Stop at Prefix Entries

- 0** → Port 1
- 100 → Port 2
- 101 → Port 3
- 11* → Port 4

Network Layer 67
Walk Tree: Stop at Prefix Entries

- 0** ➔ Port 1
- 100 ➔ Port 2
- 101 ➔ Port 3
- 11* ➔ Port 4

Network Layer 68
Slightly Different Example

- Several of the unique prefixes go to same port
  - 0** ➔ Port 1
  - 100 ➔ Port 2
  - 101 ➔ Port 1
  - 11* ➔ Port 1
Prefix Tree

0** → Port 1
100 → Port 2
101 → Port 1
11* → Port 1

Network Layer 70
LPM in real routers

- Real routers use far more advanced/complex solutions than the approaches I just described
  - but what we discussed is their starting point

- With many heuristics and optimizations that leverage real-world patterns
  - Some destinations more popular than others
  - Some ports lead to more destinations
  - Typical prefix granularities
Recap: Input linecards

- Main challenge is processing speeds

- Tasks involved:
  - Update packet header (easy)
  - LPM lookup on destination address (harder)

- Mostly implemented with specialized hardware
Output Linecard

- **Packet classification**: map each packet to a “flow”
  - Flow (for now): set of packets between two particular endpoints
- **Buffer management**: decide when and which packet to drop
- **Scheduler**: decide when and which packet to transmit
Output Linecard

- **Packet classification**: map each packet to a “flow”
  - Flow (for now): set of packets between two particular endpoints

- **Buffer management**: decide when and which packet to drop

- **Scheduler**: decide when and which packet to transmit

- **Used to implement various forms of policy**
  - Deny all e-mail traffic from ISP-X to Y (access control)
  - Route IP telephony traffic from X to Y via PHY_CIRCUIT (policy)
  - Ensure that no more than 50 Mbps are injected from ISP-X (QoS)
Simplest: FIFO Router

- No classification
- Drop-tail buffer management: when buffer is full drop the incoming packet
- First-In-First-Out (FIFO) Scheduling: schedule packets in the same order they arrive
Packet Classification

- Classify an IP packet based on a number of fields in the packet header, e.g.,
  - source/destination IP address (32 bits)
  - source/destination TCP port number (16 bits)
  - Type of service (TOS) byte (8 bits)
  - Type of protocol (8 bits)

- In general fields are specified by range
  - classification requires a multi-dimensional range search!
Scheduler

- One queue per “flow”
- Scheduler decides when and from which queue to send a packet
- Goals of a scheduling algorithm:
  - Fast!
  - Depends on the policy being implemented (fairness, priority, etc.)
Example: Priority Scheduler

- Priority scheduler: packets in the highest priority queue are always served before the packets in lower priority queues.
Example: Round Robin Scheduler

- Round robin: packets are served from each queue in turn
Switching
Shared Memory (1st Generation)

Limited by rate of shared memory

(* Slide by Nick McKeown, Stanford Univ.)
Shared Bus (2nd Generation)

Limited by shared bus

(* Slide by Nick McKeown)
Point-to-Point Switch (3rd Generation)

(*Slide by Nick McKeown)
This is called an “output queued” switch
This is called an “input queued” switch.
Two challenges with input queuing

1) Need an internal fabric scheduler!
3rd Gen. Router: Switched Interconnects

Fabric Scheduler
Two challenges with input queuing

1) Need an internal fabric scheduler!
2) Must avoid “head-of-line” blocking
Head of Line Blocking

HoL blocking limits throughput to approximately 58% of capacity.
“Virtual Output Queues”
3rd Gen. Router: Switched Interconnects
Reality is more complicated

- Commercial (high-speed) routers use
  - combination of input and output queuing
  - complex multi-stage switching topologies (Clos, Benes)
  - distributed, multi-stage schedulers (for scalability)
Summary

What we’ve covered today:
- Network layer services
- Virtual Circuit and Datagram Networks
- Router Internals

To be continued:
- IP
- IP addressing
- Fragmentation
- ICMP
- NAT
- Routing Algorithms