

**CSCE 463/612**

**Networks and Distributed Processing**

**Spring 2025**

## **Network Layer**

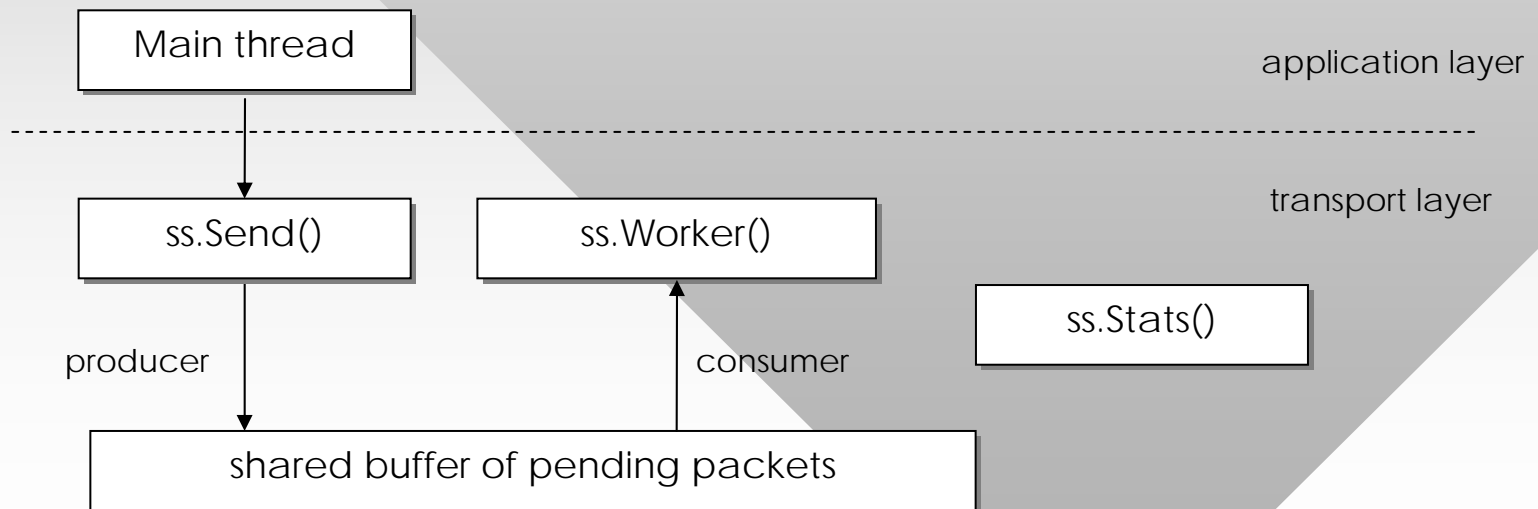
Dmitri Loguinov

Texas A&M University

March 27, 2025

## Homework #3

- Part3 requires three threads in SenderSocket
  - `ss.Send()` is the producer into a bounded buffer of  $W$  packets ( $W$  = sender window)
  - Worker thread is the consumer from this buffer (ACK arrival that moves `sndBase` by  $X$  pkts releases  $X$  slots in buffer)
  - Requires two semaphores



## Homework #3

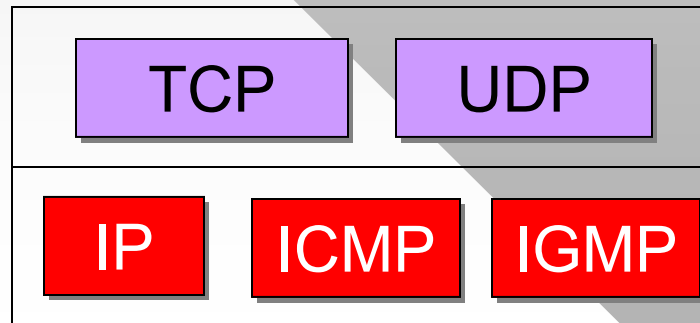
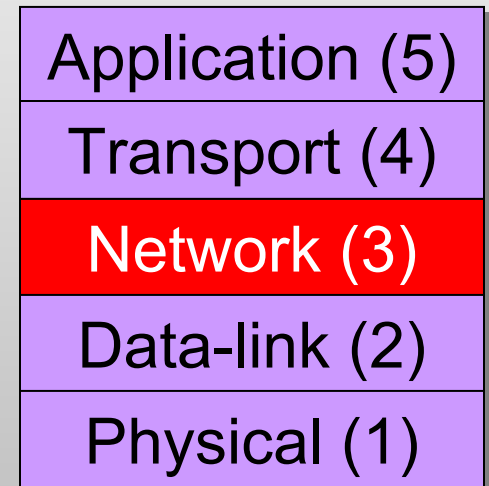
- Interesting aspect is how to release semaphore to accommodate flow control
  - Assume sndBase, nextSeq, window W are known
  - Receive ACK with sequence  $y > \text{sndBase}$ ,  $\text{recvWnd} = R$
  - By how much to release semaphore?

```
lastReleased = 0;
sndBase = -1;      // SYN-ACK 0 will move this to 0
while (not end of transfer)
{
    get ACK or SYN-ACK with sequence y, receiver window R
    if (y > sndBase)
    {
        sndBase = y
        effectiveWin = min (W, R)
        // how much we can advance the semaphore
        newReleased = sndBase + effectiveWin - lastReleased;
        ReleaseSemaphore (s, newReleased);
        lastReleased += newReleased;
    }
}
```

# Chapter 4: Network Layer

## Chapter goals:

- Understand principles behind network layer services:
  - How a router works (forwarding)
  - Routing (path selection)
  - Dealing with scale
  - Other topics: IPv6, multicasting
- Traceroute program as hw#4
- Big picture:



transport

network

# Chapter 4: Roadmap

## 4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

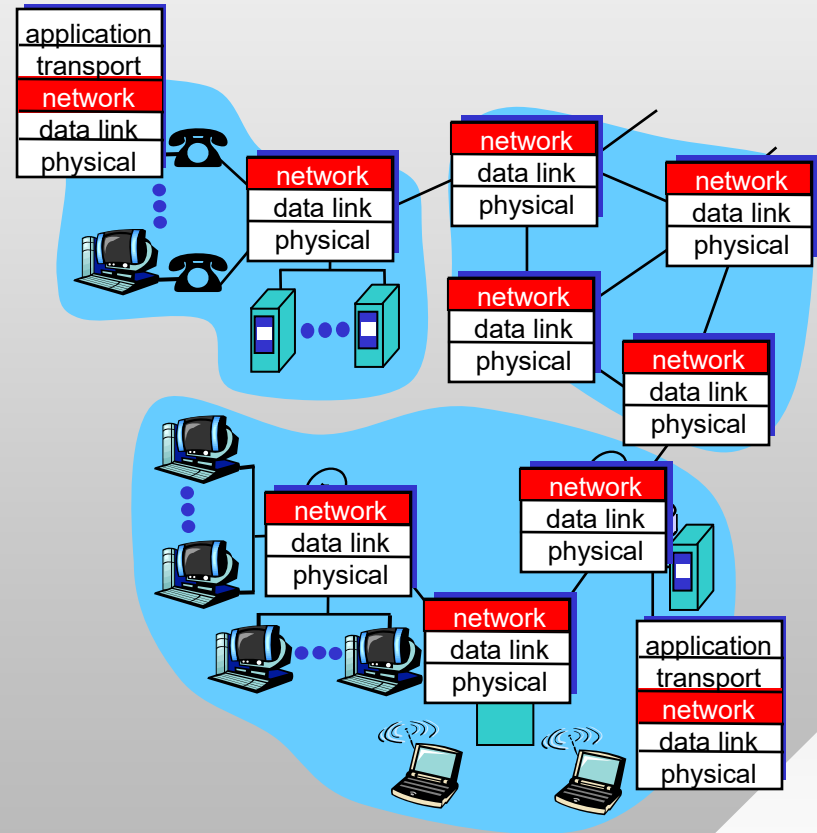
4.5 Routing algorithms

4.6 Routing in the Internet

4.7 Broadcast and multicast routing

# Network Layer = IP Layer


- Transports segments from sending to receiving host
- On the sending side, encapsulates segments into **datagrams**
- On the receiving side, delivers segments to transport layer
- Network layer protocols in **every** host and router
- Router examines header fields in all IP datagrams passing through it



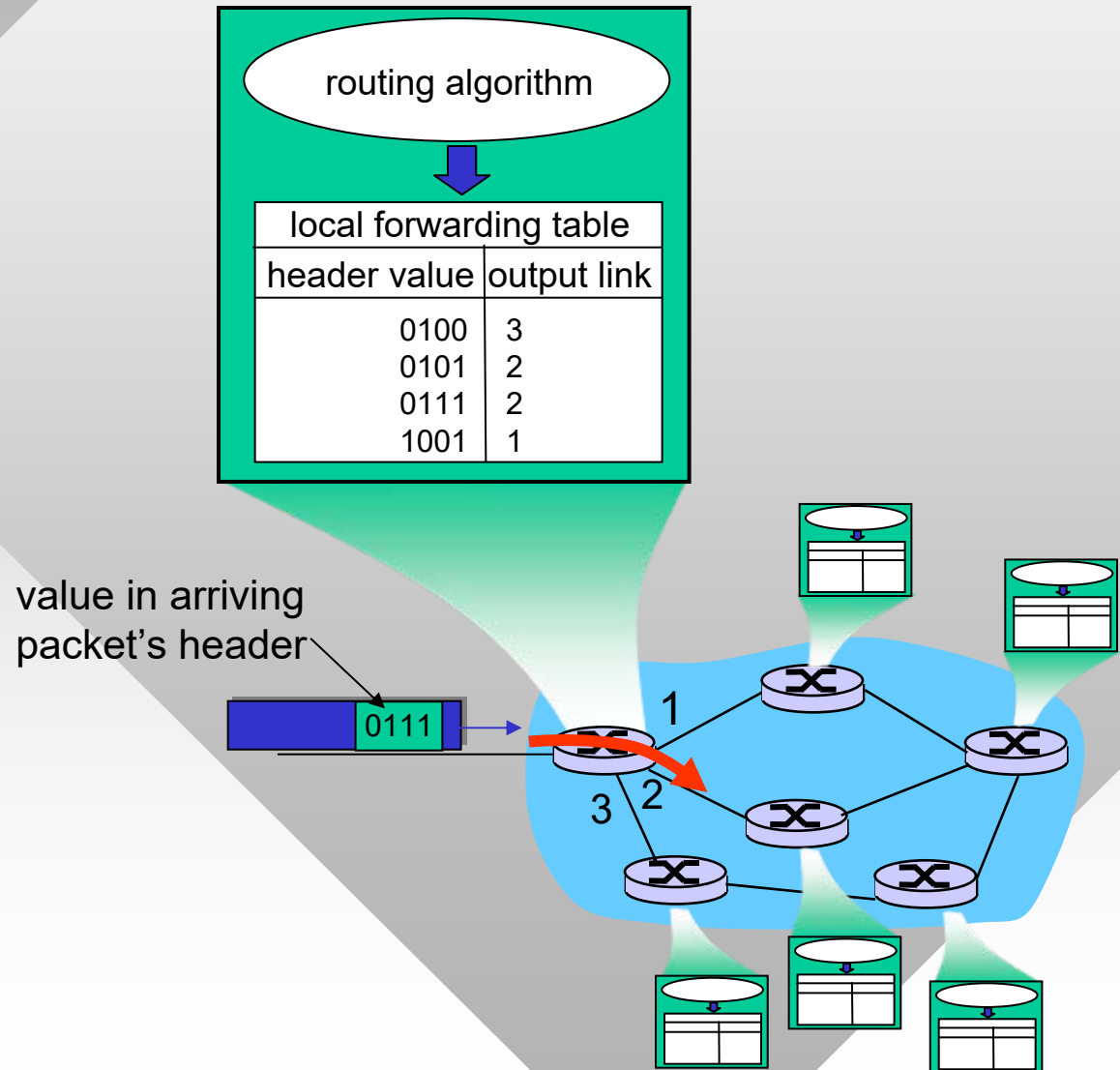
# Key Network-Layer Functions

- 1) *Routing*: determine the path taken by packets from source to dest
  - Build a minimum-cost table at each router
  - Table has next-hop neighbor for each possible destination
  - **Goal**: send packet along the least-expensive path (e.g., in terms of hops, ISPs, or peering agreements)
- 2) *Forwarding*: move packets from a router's input port to appropriate router output **port**
  - Table lookup
  - Port-to-port transfer
  - **Goal**: efficiency

physical interface  
(NIC) inside router,  
not a TCP/UDP port!



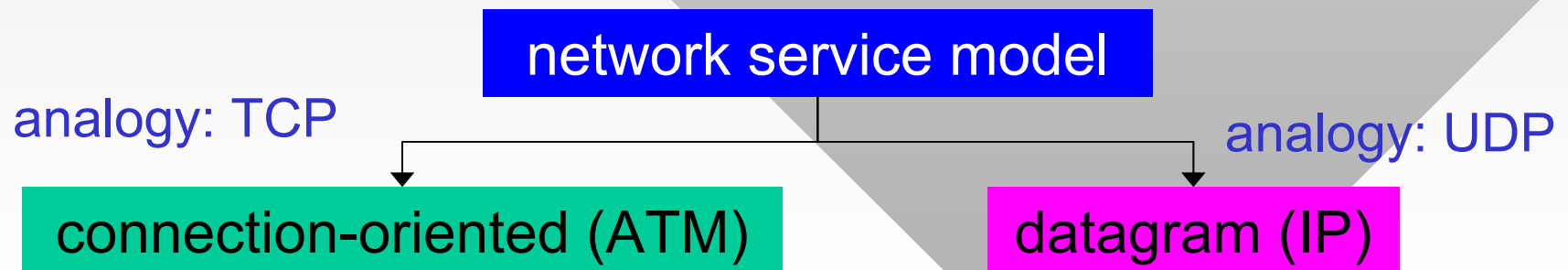
# Interplay Between Routing and Forwarding





# Connection Setup (ATM)

- 3) *Connection setup* in certain network architectures:
  - e.g., ATM (Asynchronous Transfer Mode)
- Before datagrams flow in such networks, two hosts and intermediate routers establish virtual circuit (VC)
  - Routers get involved to set up a path
- Network and transport layer connection service:
  - **Network:** between two hosts
  - **Transport:** between two sockets/processes



# Chapter 4: Roadmap

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

4.5 Routing algorithms

4.6 Routing in the Internet

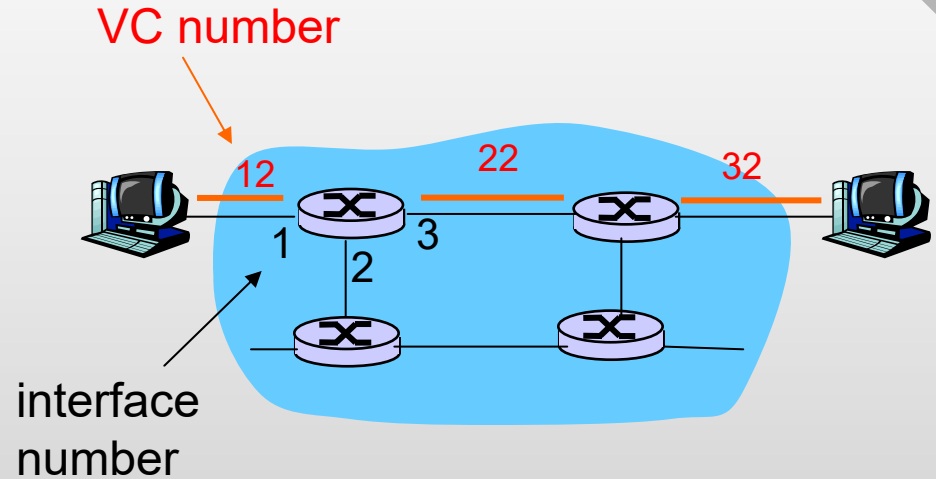
4.7 Broadcast and multicast routing

# Virtual Circuits

- VCs may create a path that behaves much like a telephone circuit (no congestion, low delay, no loss)
- Call setup for each connection *before* data can flow
  - Similar to TCP's handshake, but involves routers
- Each packet carries a **VC tag** instead of the 5-tuple <src addr, dest addr, src port, dest port, proto>
- *Every* router on source-dest path maintains “state” for each passing connection
  - Mapping from tags to next-hop router
- Fraction of router resources (bandwidth, buffers) are allocated to each VC

# Forwarding Table

Forwarding table in northwest router:

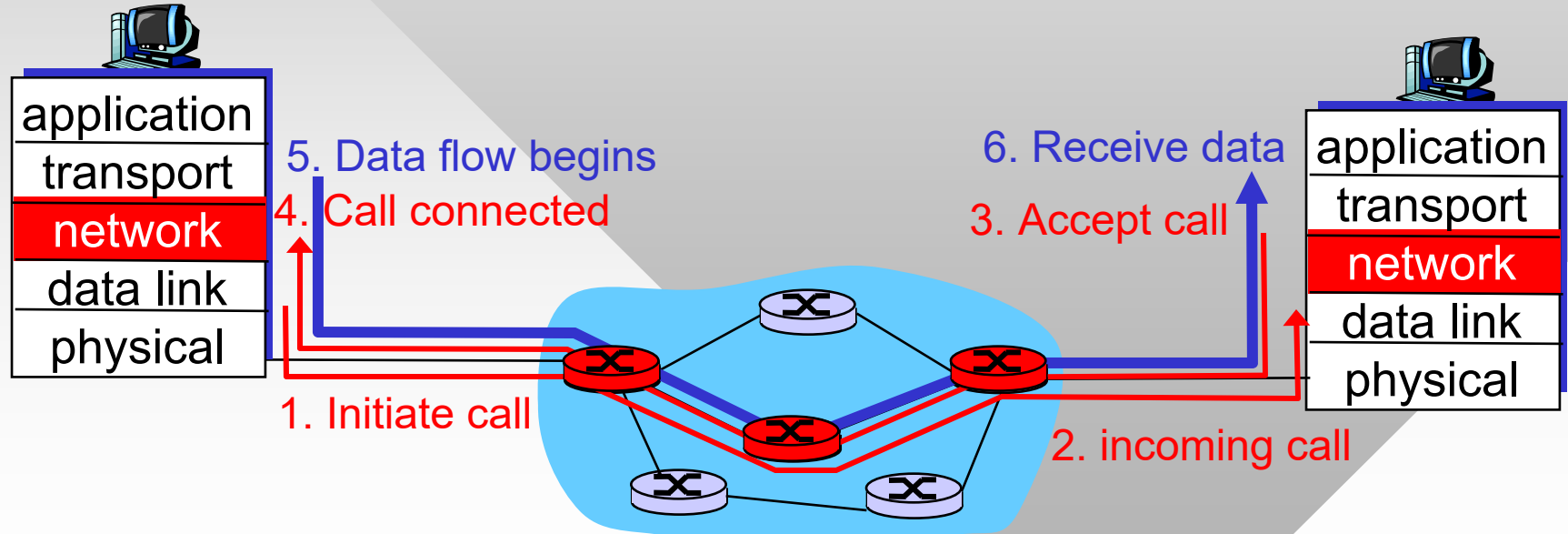


Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

**Routers maintain connection state information!**

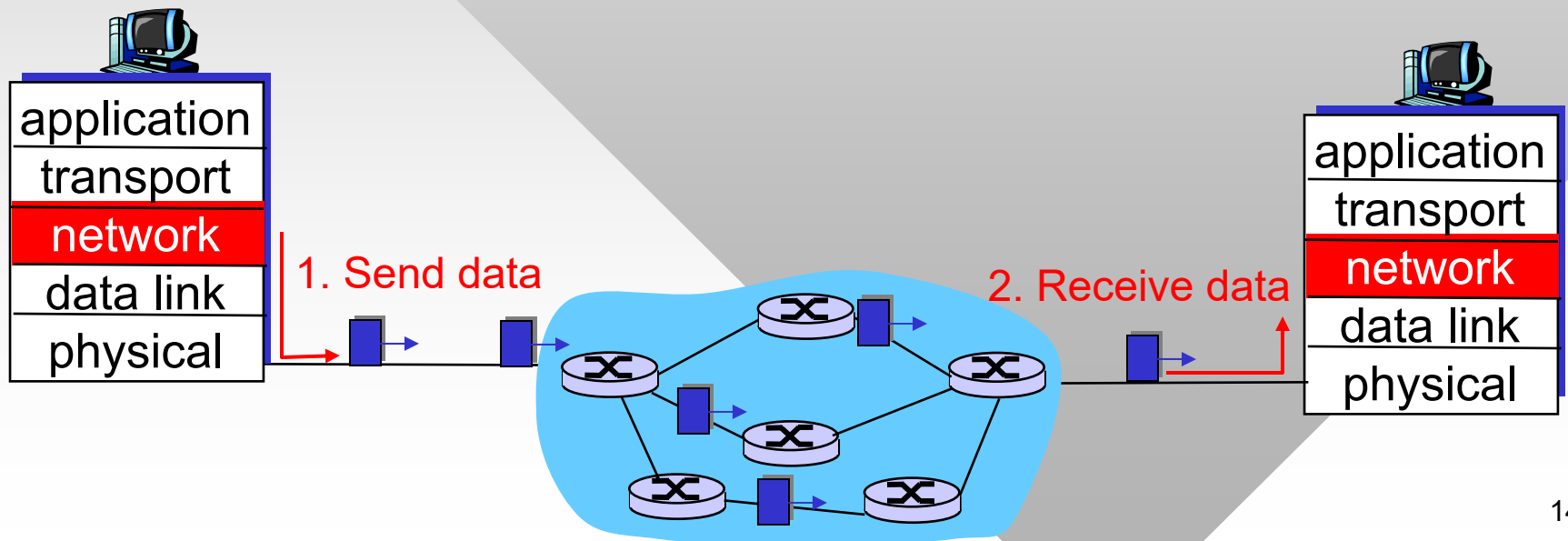
# Virtual Circuits: Signaling Protocols

- Setup, maintain, teardown VC
- Used in ATM, frame-relay, etc.
- **Not used end-to-end in today's Internet**



# 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**
  - Packets between the same source-dest pair may take different paths (**multi-path routing**)



# Datagram Forwarding Table

4 billion  
possible entries

Destination Address Range (32 bit)

Link Interface

11001000 00010111 00010000 00000000  
through  
11001000 00010111 00010111 11111111

0

11001000 00010111 00011000 00000000  
through  
11001000 00010111 00011000 11111111

1

11001000 00010111 00011000 00000000  
through  
11001000 00010111 00011111 11111111

2

otherwise

3

# Longest Prefix Matching

<u>Prefix Match</u>	<u>Link Interface</u>
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

## Examples (DA = destination address)

DA: 11001000 00010111 00010110 10100001  
DA: 11001000 00010111 00011001 10101010  
DA: 11001000 00010111 00011000 10101010

Which interface?



# Datagram or VC Network: Why?

## Internet

- Driven by data exchange among computers
  - “Elastic” service, no strict timing requirements
- “Smart” end systems (computers)
  - Can adapt, perform control, error recovery
  - Simple network core, complexity at “edge”
- Many link types
  - Different characteristics
  - Uniform service difficult

## ATM

- Evolved from telephony
- Human conversation:
  - Strict timing, bandwidth requirements
  - Need for guaranteed service
- “Dumb” end systems
  - Telephones
  - Complexity in network core