

# Introduction to Network

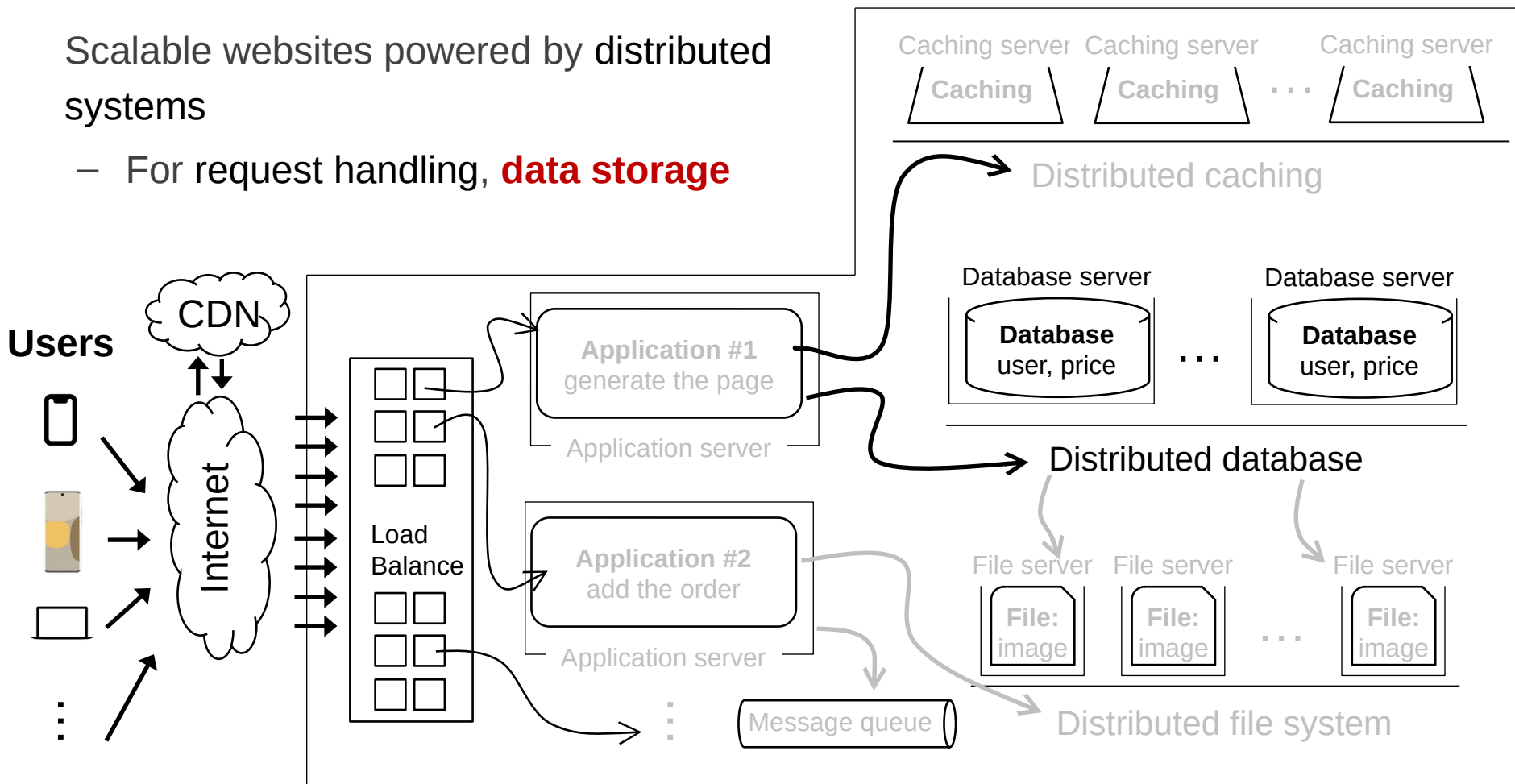
IPADS, Shanghai Jiao Tong University

<https://www.sjtu.edu.cn>

# Review: scalable websites

Scalable websites powered by distributed systems

- For request handling, **data storage**



# Layers in Network

## **Application**

- Can be thought of as a fourth layer
- Not part of the network

## **End-to-end layer**

- Everything else required to provide a comfortable application interface

## **Network layer**

- Forwarding data through intermediate points to the place it is wanted

## **Link layer**

- Moving data directly from one point to another

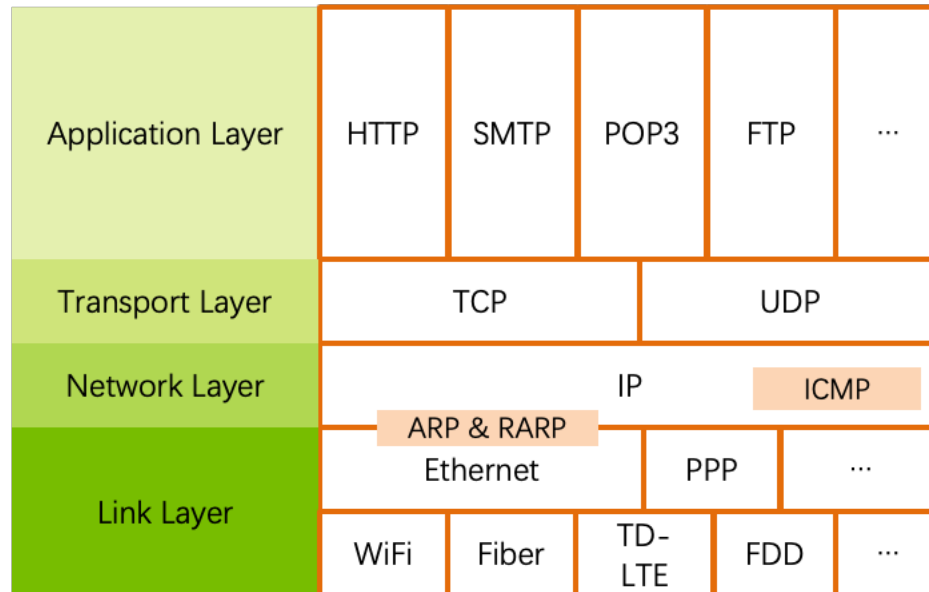
# OSI, TCP/IP & Protocol Stack

## OSI

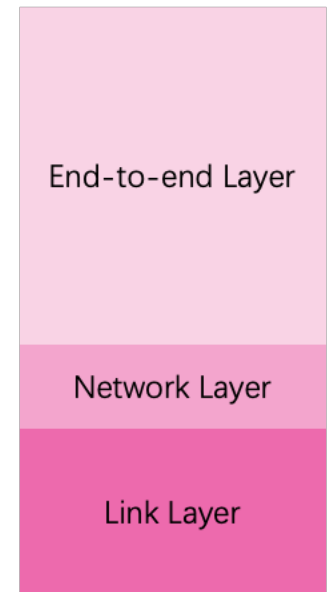
- The open systems interconnection (OSI) model
- 7-layer architecture



OSI



TCP/IP



CSE

# The Internet "Hour Glass" Protocols

## More people, more useful

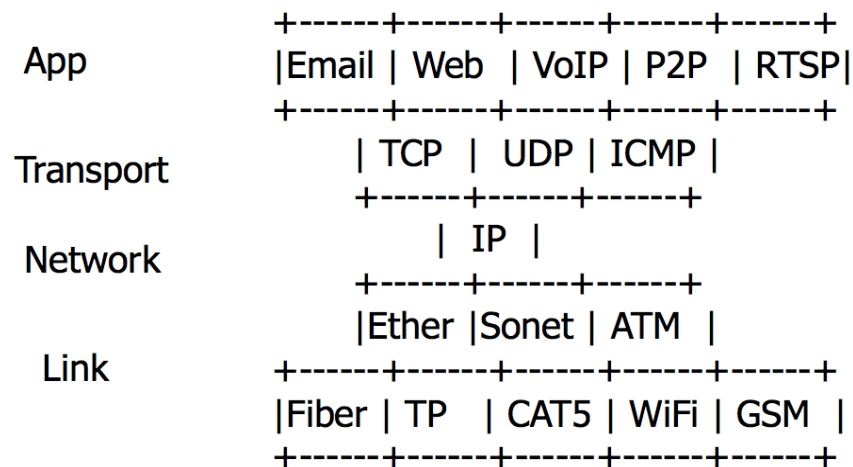
- Value to me =  $N$
- Value to society is  $N^2$

## Network, dumb vs. smart

- Standardize vs. flexibility

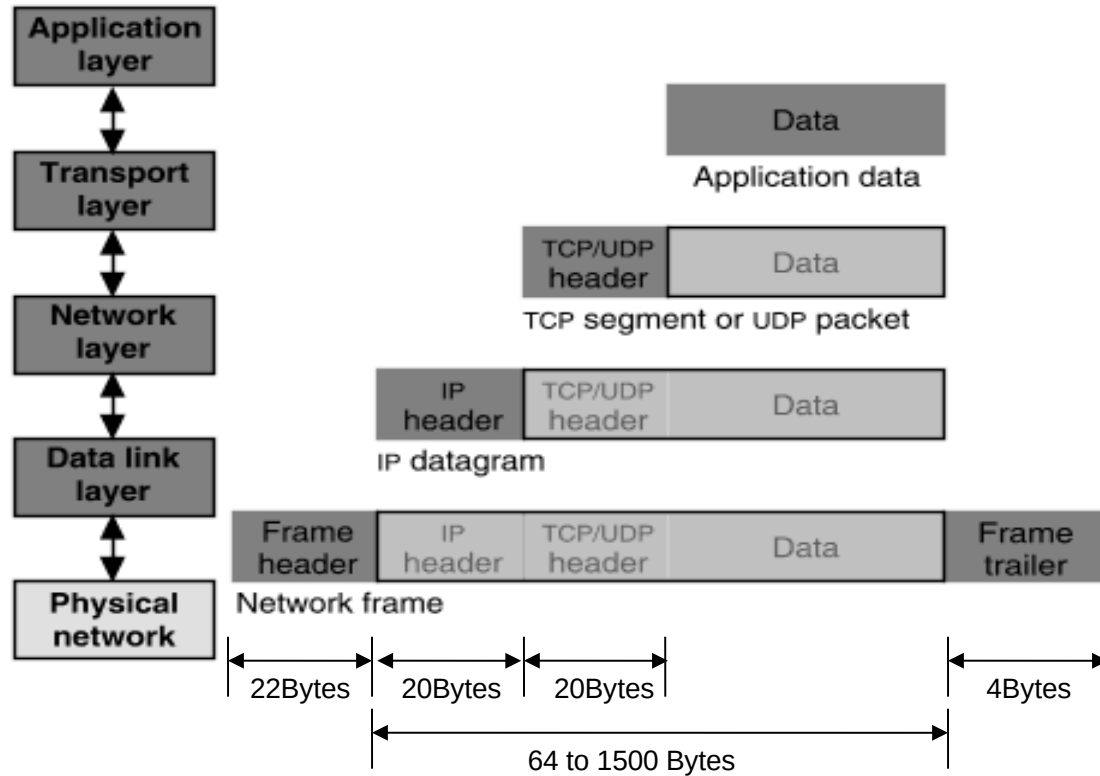
## Network is a black box

- Simplify the system that uses it



"Everything over IP, and IP over everything"

# Packet Encapsulation



# Application Layer

## Entities

- Client and server
- End-to-end connection

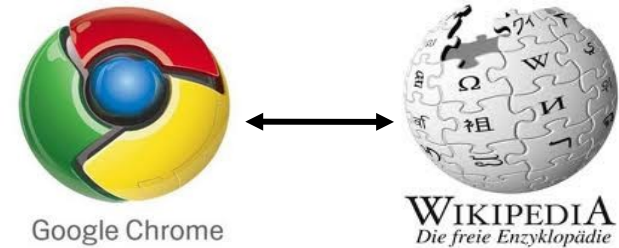
## Name space: URL

## Protocols

- HTTP, FTP, POP3, SMTP, etc.

## What to care?

- Content of the data: video, text, ...



```
<html>
  <head>
    <title>Google</title>

    <script>window.google=...
    </script>
  </head>
  <body>  ...  </body>
</html>
```

# Transport Layer

## Entities

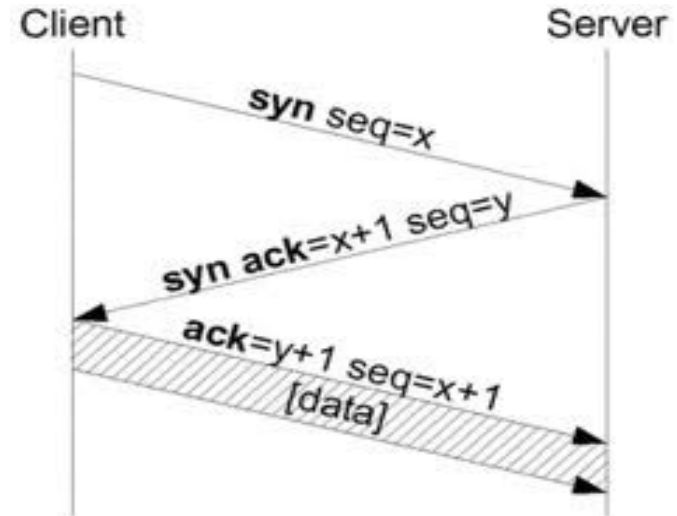
- Sender and receiver
- Proxy, firewall, etc.
- End-to-end connection

Name space: **port** number

Protocols: TCP, UDP, etc.

## What to care?

- TCP: Retransmit packet if lost
- UDP: nothing





## Packet Format of TCP & UDP

### TCP Segment Header Format

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Sequence Number							
64	Acknowledgment Number							
96	Data Offset	Res	Flags		Window Size			
128	Header and Data Checksum				Urgent Pointer			
160...	Options							

### UDP Datagram Header Format

Bit #	0	7	8	15	16	23	24	31
0	Source Port				Destination Port			
32	Length				Header and Data Checksum			

# Network Layer (the Internet Layer, IP Layer)

## Network entities

- Gateway, bridge
- Router, etc.

## Name space

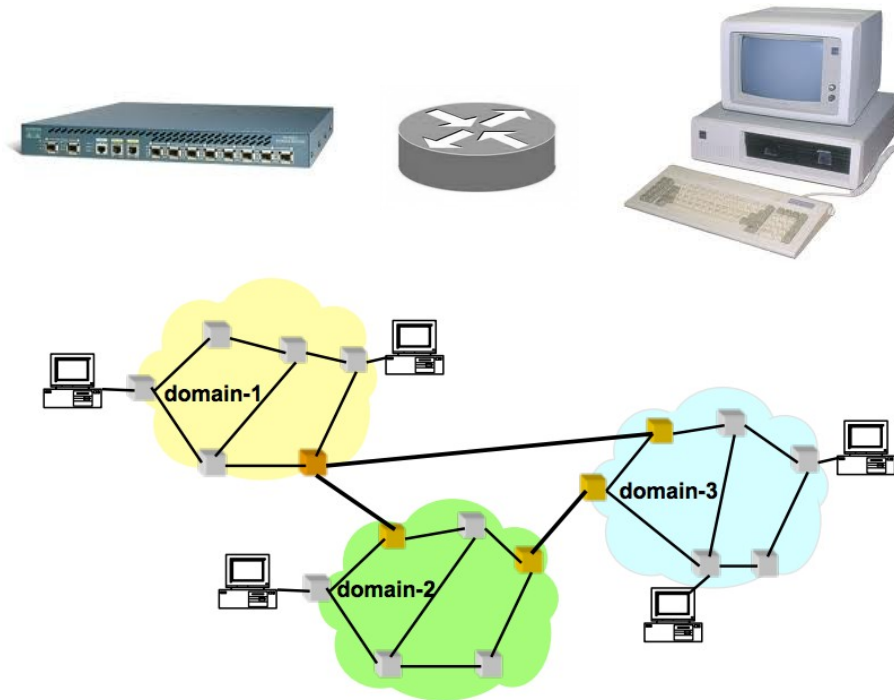
- IP address

## Protocols

- IP, ICMP (ping)

## What to care?

- Next hop decided by route table



# IP Datagram (Packet, Package)

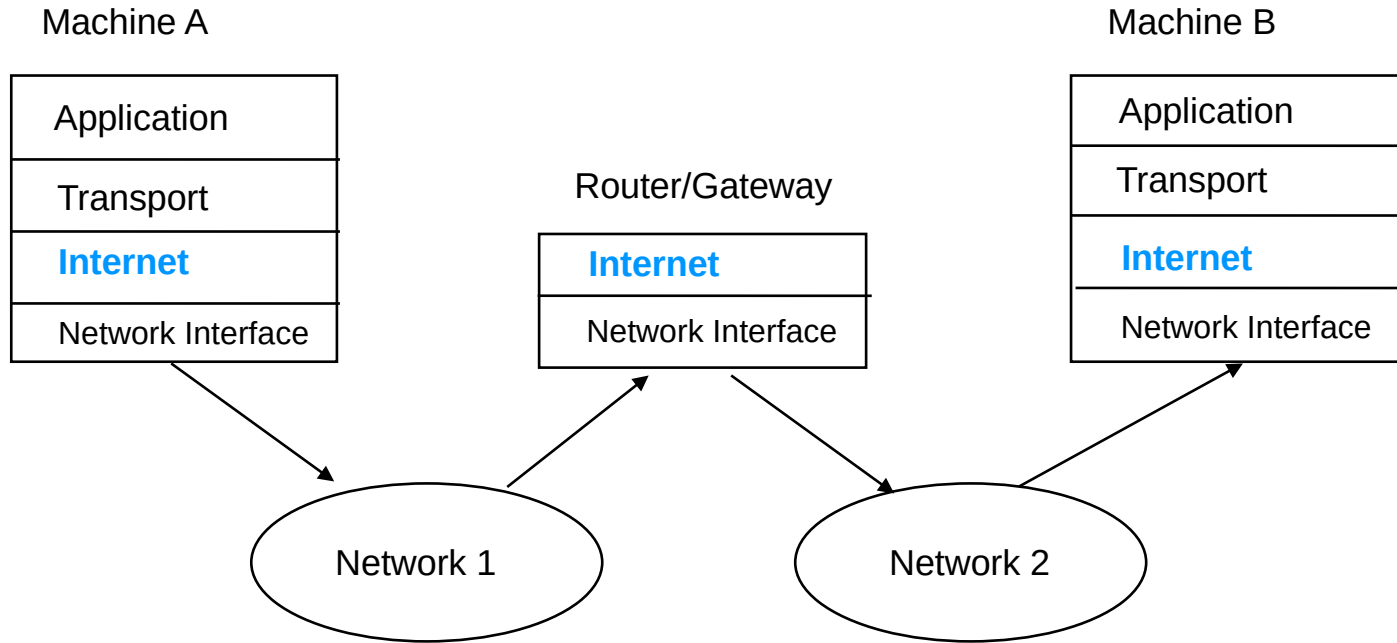
0	4	8	16	19	31
Version	IHL	Type of Service	Total Length		
Identification			Flags	Fragment Offset	
Time To Live		Protocol	Header Checksum		
Source IP Address					
Destination IP Address					
Options					Padding

Header

```
10101011101010101010010101010100101010100
11010010101010010101111111010000011101111
10100001011101010100110101011110100000101
00100000000010101000011010000111111010101
..... 1011011001010100011001001010110
```

Data

# TCP/IP Architecture



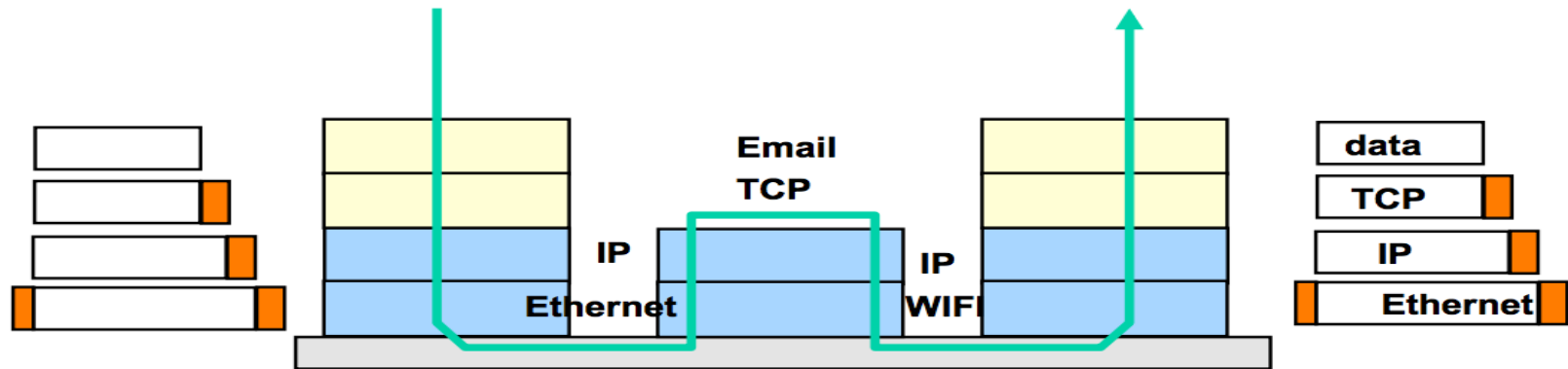
# TCP/IP Architecture

Each layer adds/strips off its own header

Each layer may split up higher-level data

Each layer multiplexes multiple higher layers

Each layer is (mostly) transparent to higher layers



# Link Layer

From a node to its physical neighbor

# The Link Layer

**The bottom-most layer of the three layers**

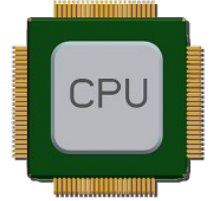
**Purpose: moving data directly from one physical location to another**

- 1. Physical transmission
- 2. Multiplexing the link
- 3. Framing bits & bit sequences
- 4. Detecting transmission errors
- 5. Providing a useful interface to the up layer

# Physical Transmission using Shared Clock

**Example-1: moving a bit from register-1 to register-2  
on the same chip**

- Run a wire to connect output of reg-1 to input of reg-2
- Wait till reg-1's output has settled & signal has propagated to reg-2
- Reg-2 read input the next clock tick
- **Assumption:** propagation can be done within one clock



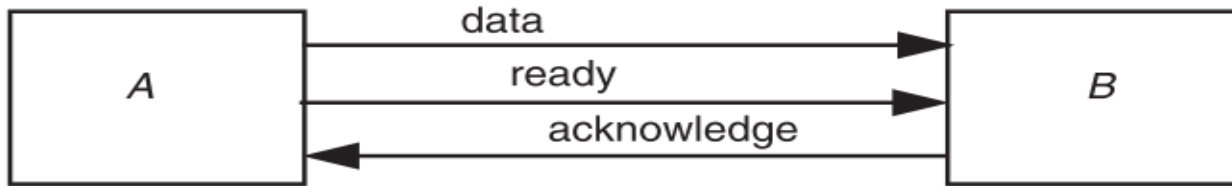
**How to send data between two modules without sharing a clock?**



# Physical Transmission without Shared Clock

## Three-wire ready/acknowledge protocol

- 1. **A** places data on data line
- 2. **A** changes value on the ready line
- **B** sees the ready line change, reads value on the data line, then changes the acknowledge line



**B:** when to look at the data line? (*ready* is set)

**A:** when to stop holding the bit value on the data line? (*ack* is set)

# Parallel Transmission

## Propagation time $\Delta t$

- It takes more than  $2\Delta t$  to send one bit
- The max data rate is  $1/(2\Delta t)$

## Parallel transmission

- Use  $N$  parallel data lines to achieve  $N/(2\Delta t)$
- E.g., SCSI, printer, etc.

# Serial Transmission

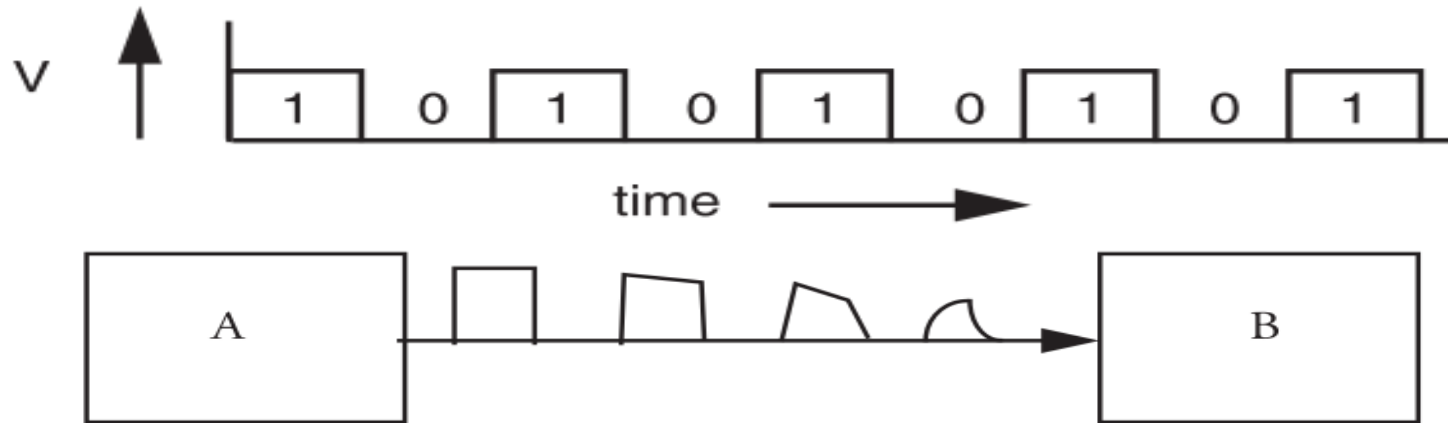
## Ready/acknowledge protocol

- $\Delta t$  grows significantly, which limits the data rate

## Serial transmission

- Send a **stream** of bits down a single line
- Without waiting for any response from the receiver
- Expect the receiver can recover the bits with no additional signal
- Higher rates, longer distance, fewer wires
- E.g., USB, SATA

## Signal Transmission on Analog Line



**It is hard for B to understand the signal**

- B doesn't have a copy of A's clock, so when to sample the signal?

# VCO: Voltage Controlled Oscillator

How to make two ends agree on the data rate without clock line?

The receiver run a VCO at about the same data rate

- VCO's output is multiplied by the voltage of incoming signal
- The product is suitably filtered and sent back to adjust the VCO
- VCO will finally be **locked** to both the frequency and phase of the arriving signal: phase-locked loop
- Then the VCO becomes a clock source for the receiver



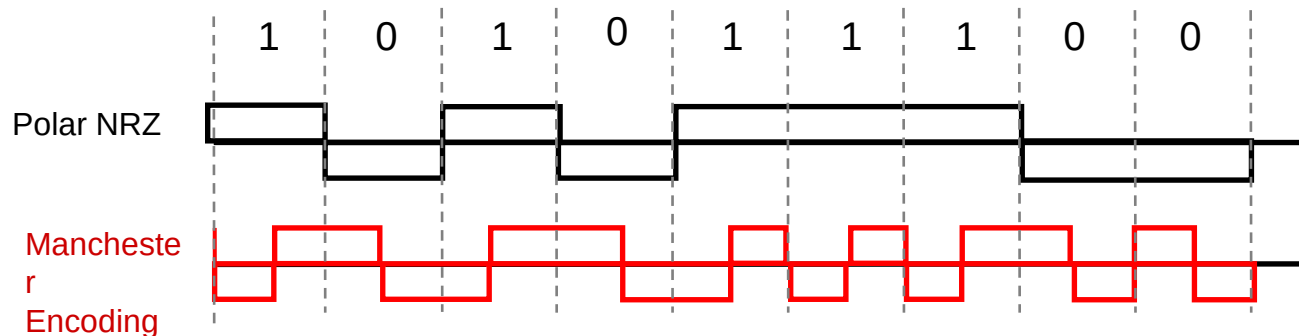
**Problem: if no transition in the stream (e.g., a lot of zero), the phase-locked loop cannot synchronize**

# Manchester Code

**Solution: sender encodes the data to ensure transitions**

**Phase encoding: at least 1 level transition for a bit**

- Manchester code: 0  $\rightarrow$  01, 1  $\rightarrow$  10
- Max data rate is only half, but simple enough



# How to Share a Connection?

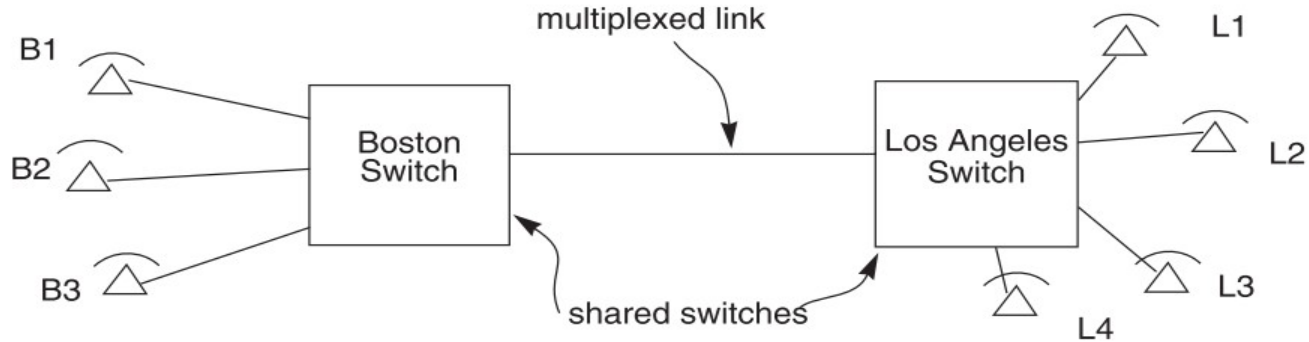
## **Isochronous communication (telephone communication)**

- Needs prior arrangement between switches
- Connection: set up and tear down
- **Stream**: continuous bits flows out of a phone

## **Asynchronous communication (data communication)**

- **Message**: burst, ill-suited to fixed size and spacing of isochronous frames
- Connectionless, asynchronous

# Isochronous Multiplexing

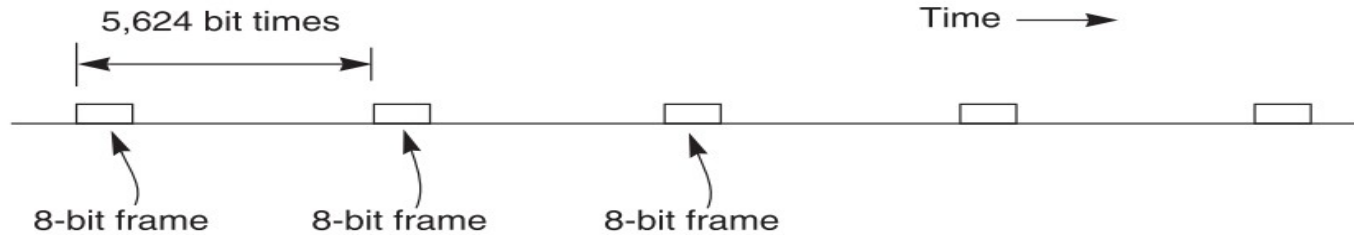


## Telephone network

- Leverage "virtual link" for connection
- "network is busy" when no available time slot



## Isochronous - TDM



64 Kbps each phone, 45 Mbps link

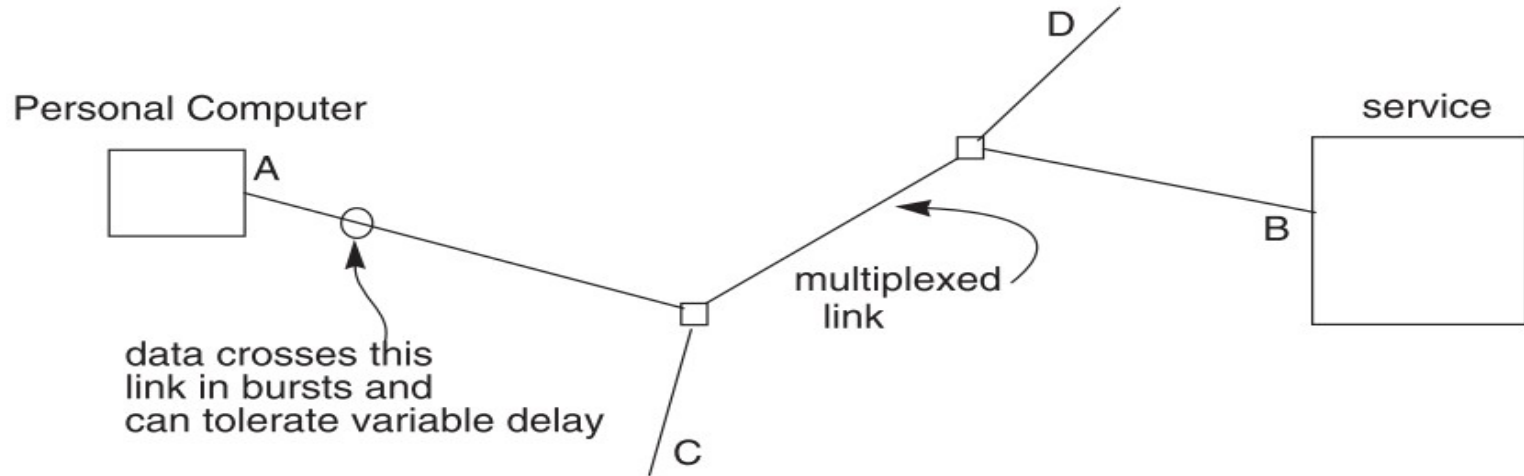
8-bit block (frame), 8000 frames per second

5624 bit times or 125 us

703 simultaneous conversations (what if there is a 704<sup>th</sup> calling?)

**Q: Why the voice is still *continuous*, instead of *fragmented*?**

# Data Communication Network



Data communication network usually contains **burst** communication

Different from the telephone network

# Frame and Packet: Asynchronous Link

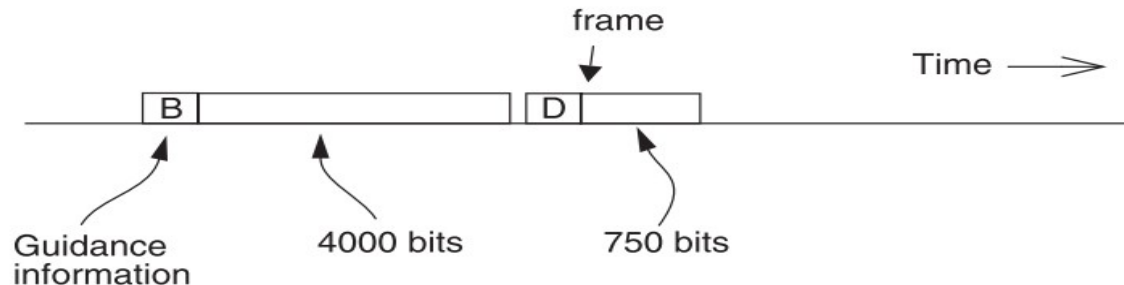
Frame can be of any length, carried at any time that the link is free

Packet: a variable-length frame with its **guidance info**

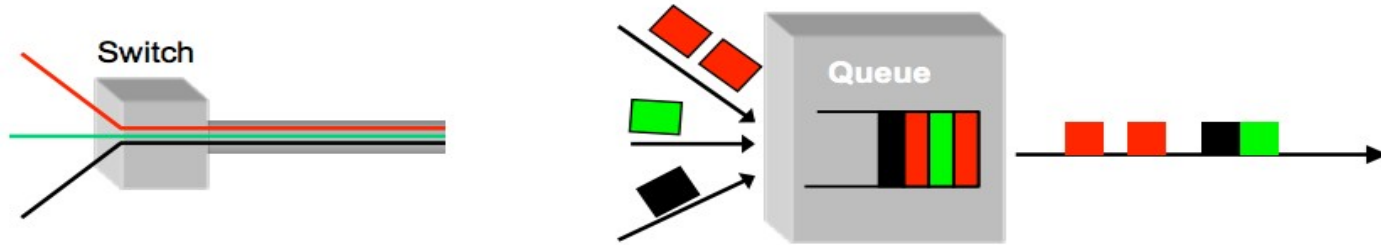
Connectionless transmission: no state maintained

Segment and reassemble

Packet voice: replacing many parts of isochronous network



# Multiplexing / Demultiplexing



**Multiplex using a queue: switch need memory/buffer**

**Demultiplex using information in packet header**

- Header has destination
- Switch has a forwarding table that contains information about which link to use to reach a destination

# Framing Frames

**Where a frame begins and ends**

**Independent from framing bits**

- Some model separates link layer to 2: one for bits and one for frames

**Simple method**

- Choose a pattern of bits, e.g., 7 one-bits in a row, as a frame-separator
- Bit stuffing: if data contains 6 ones in a row, then add an extra bit (0)

# Error Handling

## Error detection code

- Adding redundancy: e.g., checksum at the end

## What to do if detect an error

- Error correction code: with enough redundancy
  - Where noise is well understood, e.g., disk
- Ask sender to resend: sender holds frame in buffer
- Let receiver discard the frame
- Blending these techniques

# Coding: Incremental Redundancy

## Forward error correction

- Perform coding before storing or transmitting
- Later decode the data without appealing to the creator

## Hamming distance

- Number of 1 in  $A \oplus B$ ,  $\oplus$  is exclusive OR (XOR)
- If H-distance between every legitimate pair is 2
  - 000101, can only detect 1-bit flip
- If H-distance between every legitimate pair is 3
  - Can only correct 1 bit flip
- If H-distance between every legitimate pair is 4
  - Can detect 2-bit flip, correct 1-bit flip

100101  
000111

100101  
010111

## Example-1: Simple Parity Check

2 bits -> 3 bits

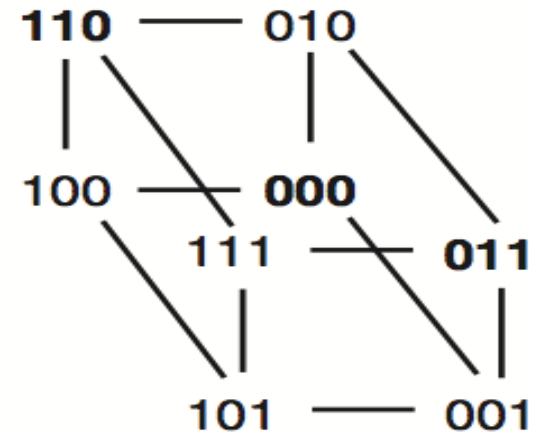
- Detect 1-bit errors
- 8 patterns total

Only 4 correct patterns

- 00 -> 00**0**
- 11 -> 11**0**
- 10 -> 10**1**
- 01 -> 01**1**

Hamming distance of this code is 2

- 1-bit flipping will cause incorrect pattern





## Example-2: 4-bit -> 7-bit

4 bits -> 7 bits (56 using only extra 7)

- 3 extra bits to distinguish 8 cases
- e.g. **1101** -> **1010101**

### Correct 1-bit errors

- **1010101** -> **1010001** : P1 & P4 not match
- **1010101** -> **110101** : P2 not match

1	2	3	4	5	6	
7						
1	0	1	0	1	0	1
1	0	1	0	1	0	1
1	0	1	0	1	0	1

$$\begin{aligned}P_1 &= P_7 \oplus P_5 \oplus P_3 \\P_2 &= P_7 \oplus P_6 \oplus P_3 \\P_4 &= P_7 \oplus P_6 \oplus P_5\end{aligned}$$

Not Match	Error
None	None
P1	P1
P2	P2
P4	P4
P1 & P2	P3
P1 & P4	P5
P2 & P4	P6
P1 & P2 & P4	P7



**NETWORK LAYER**

# IP: Best-effort Network

## 1. Best-effort network

- If it cannot dispatch, may discard a packet

## 2. Guaranteed-delivery network

- Also called [store-and-forward](#) network, no discarding data
- Work with complete messages rather than packets
- Use disk for buffering to handle peaks
- Tracks individual message to make sure none are lost

## In real world

- No absolute guarantee
- Guaranteed-delivery: higher layer; best-effort: lower layer

## Duplicate Packets and Suppression

### **Discarding packets is common case**

- Many network protocol includes timeout and resend mechanism

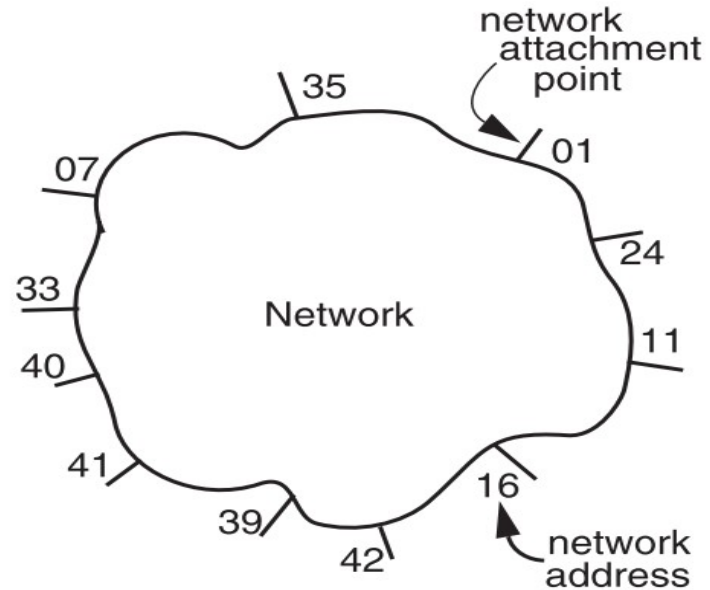
### **When a congested forwarder discards a packet**

- Client does not receive a response as quickly as originally hoped
- Users may prepared for duplicate requests and responses
- Detecting duplicates may or may not be important

# The Network Layer

## Addressing interface

- Network attachment points
- Network address
- Source & destination



**NETWORK\_SEND** (segment\_buffer, destination, network\_protocol, end\_layer\_protocol)

**NETWORK\_HANDLE** (packet, network\_protocol)

# Managing the Forwarding Table: Routing

## Routing (or path-finding)

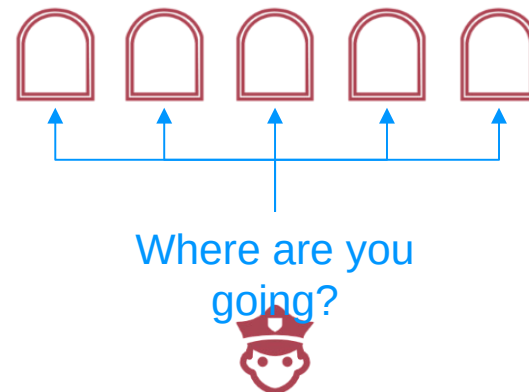
- Constructing the tables

## Impractical by hand

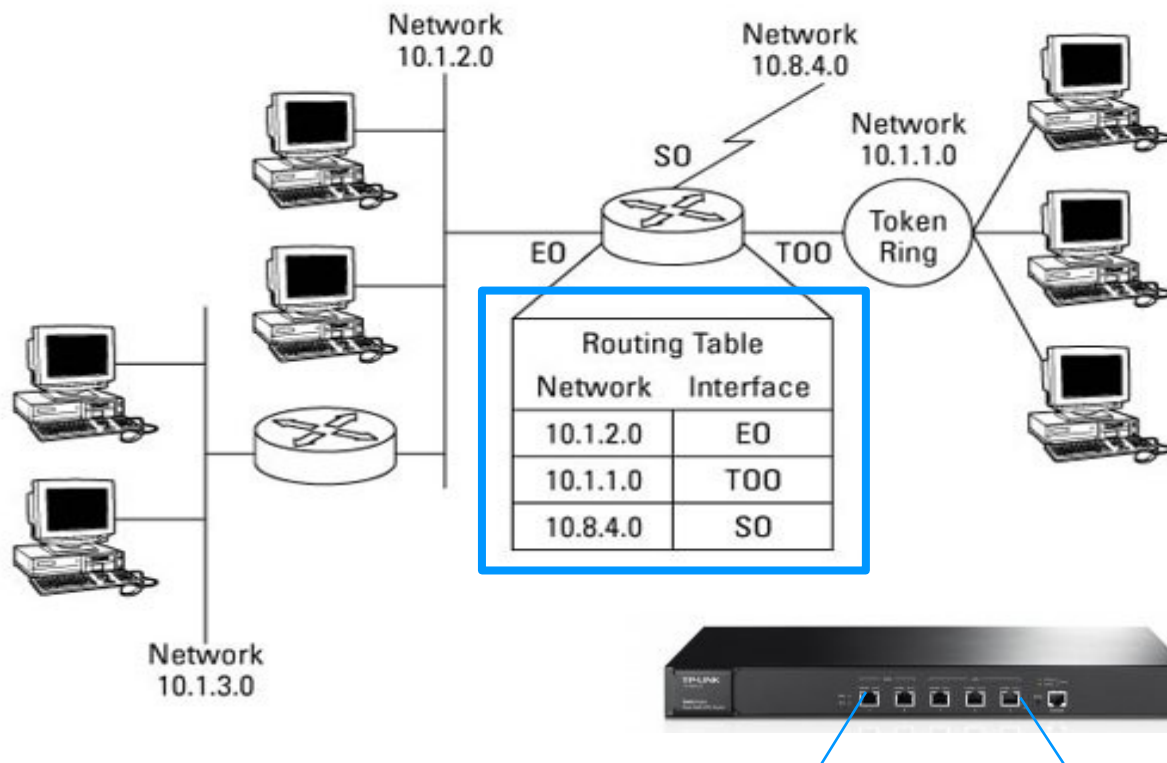
- Determining the best paths requires calculation
- Recalculating the table when links change
- Recalculating the table when link fails
- Adapt according to traffic congestion

## Static routing vs. adaptive routing

- Adaptive routing requires exchange of info



# IP Route Table



# Control-plane VS. Data-plane

## Control-plane

- Control the data flow by defining rules
- E.g., the routing algorithm

## Data-plane

- Copies data according to the rules
- Performance critical
- E.g., the IP forwarding process