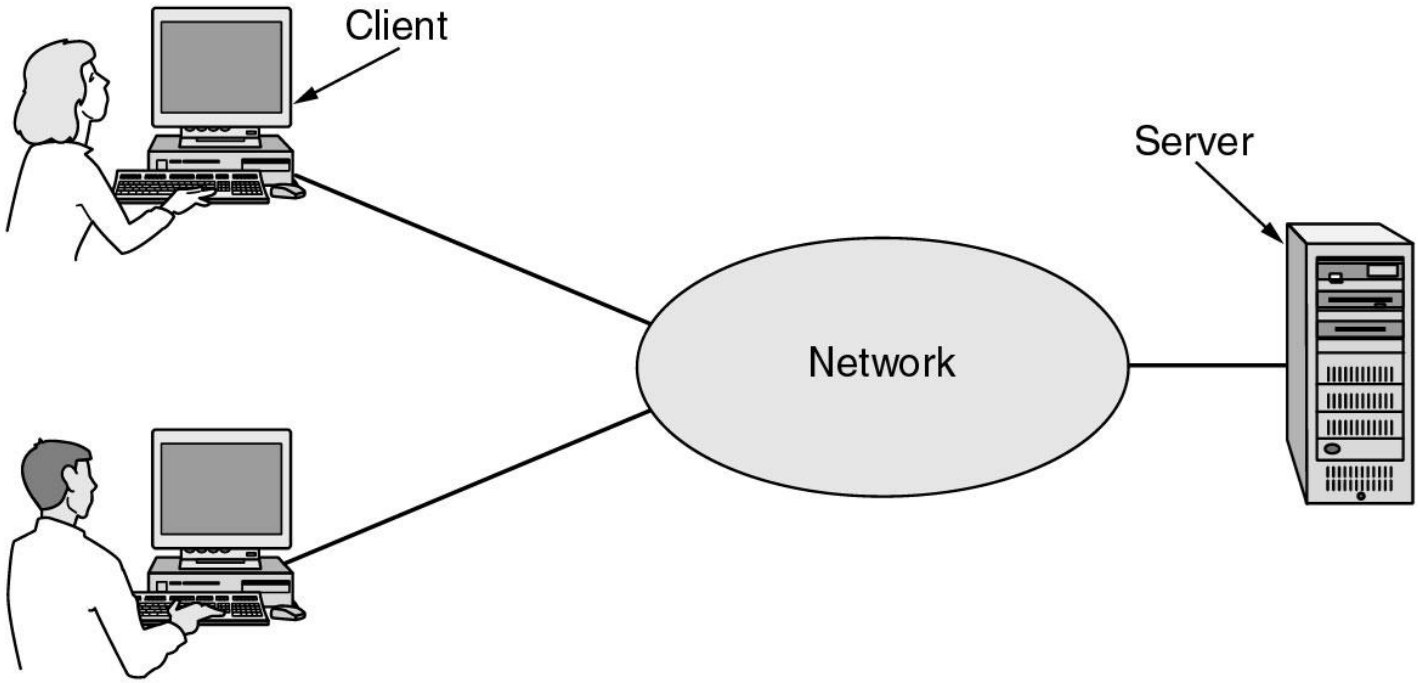
Chapter 1

Introduction

Uses of Computer Networks

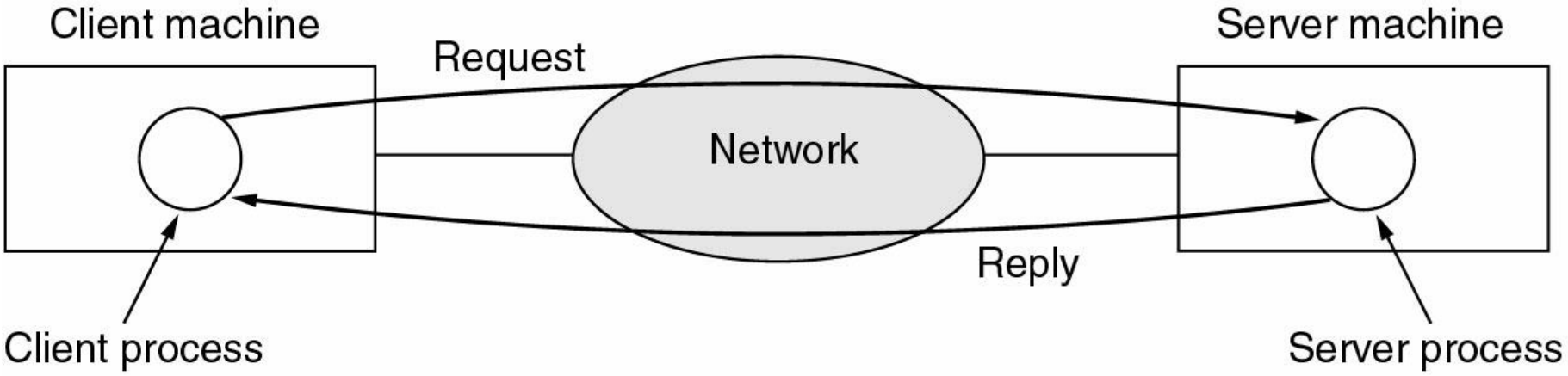
- Business Applications
- Home Applications
- Mobile Users
- Social Issues

Business Applications of Networks



A network with two clients and one server.

Business Applications of Networks (2)

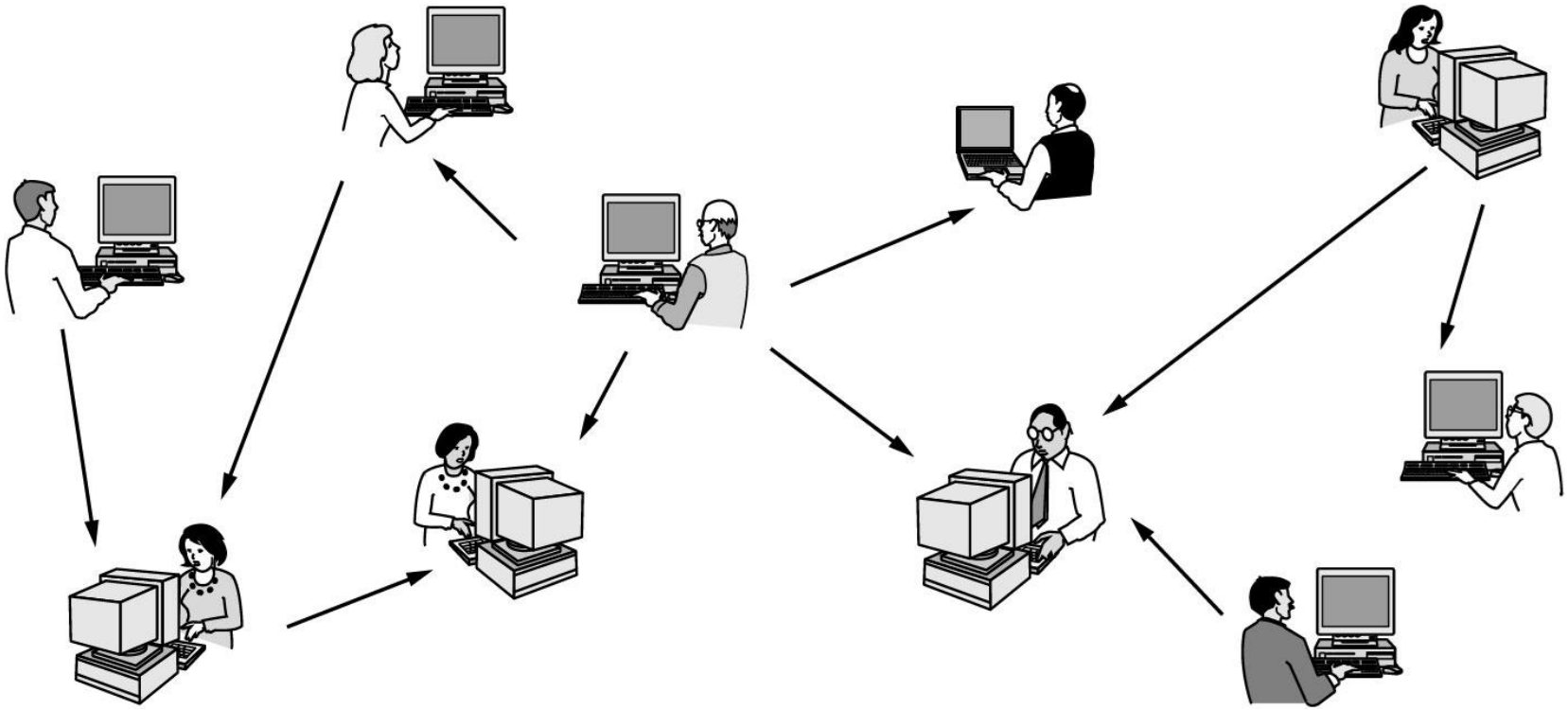


The client-server model involves requests and replies.

Home Network Applications

- Access to remote information
- Person-to-person communication
- Interactive entertainment
- Electronic commerce

Home Network Applications (2)



In peer-to-peer system there are no fixed clients and servers.

Home Network Applications (3)

Tag	Full name	Example
B2C	Business-to-consumer	Ordering books on-line
B2B	Business-to-business	Car manufacturer ordering tires from supplier
G2C	Government-to-consumer	Government distributing tax forms electronically
C2C	Consumer-to-consumer	Auctioning second-hand products on-line
P2P	Peer-to-peer	File sharing

Some forms of e-commerce.

Mobile Network Users

Wireless	Mobile	Applications
No	No	Desktop computers in offices
No	Yes	A notebook computer used in a hotel room
Yes	No	Networks in older, unwired buildings
Yes	Yes	Portable office; PDA for store inventory

Combinations of wireless networks and mobile computing.

Network Hardware

- Local Area Networks
- Metropolitan Area Networks
- Wide Area Networks
- Wireless Networks
- Home Networks
- Internetworks

Broadcast Networks

Types of transmission technology

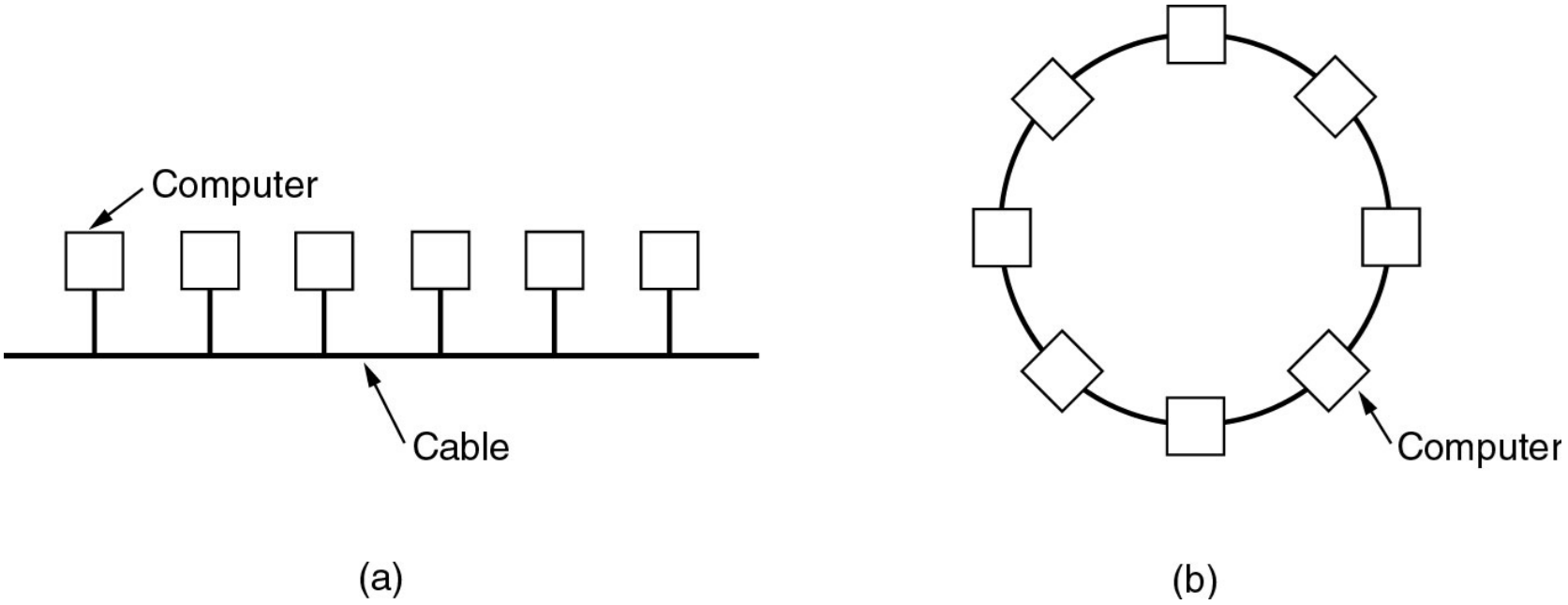
- Broadcast links
- Point-to-point links

Broadcast Networks (2)

Interprocessor distance	Processors located in same	Example
1 m	Square meter	Personal area network
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1000 km	Continent	
10,000 km	Planet	The Internet

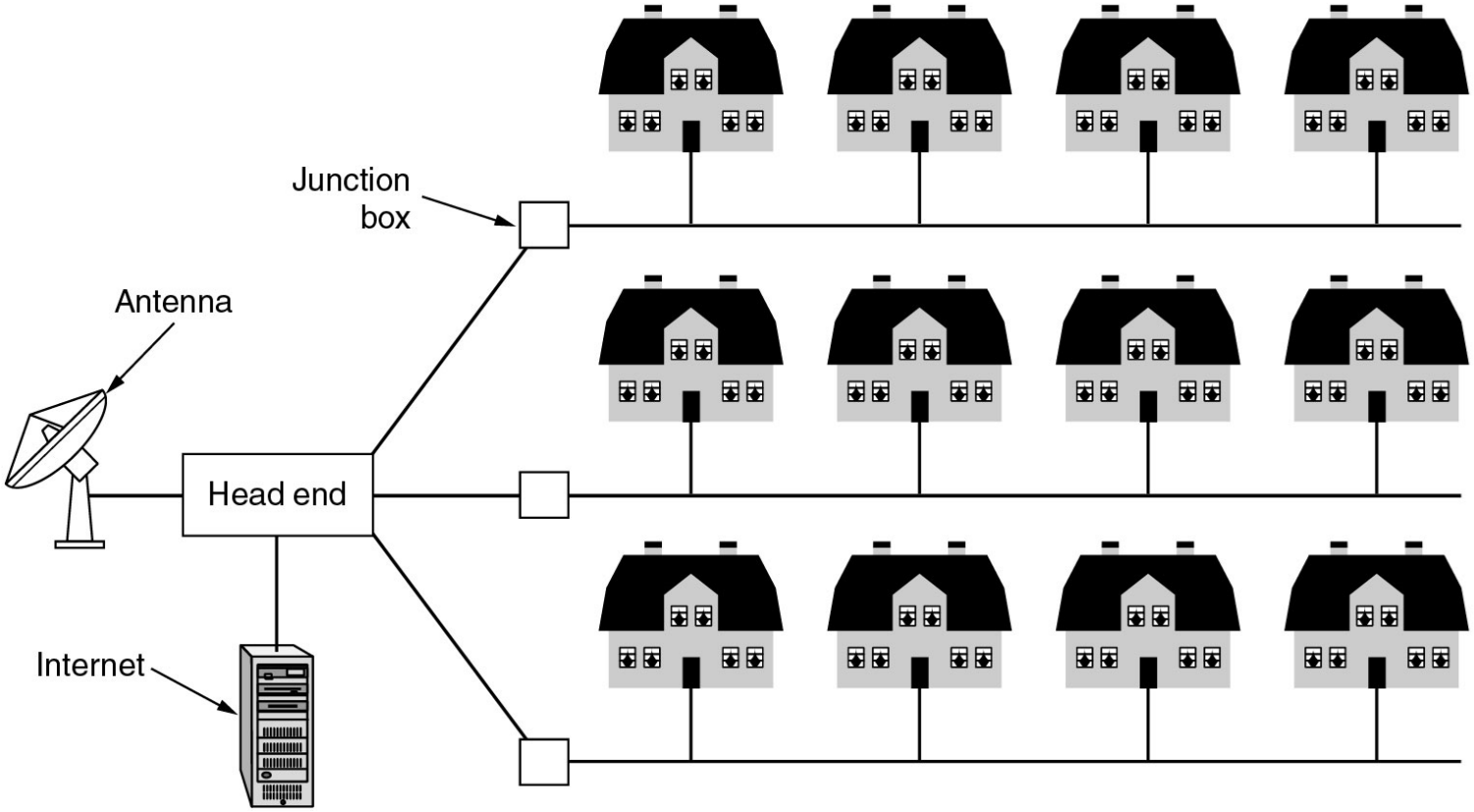
Classification of interconnected processors by scale.

Local Area Networks



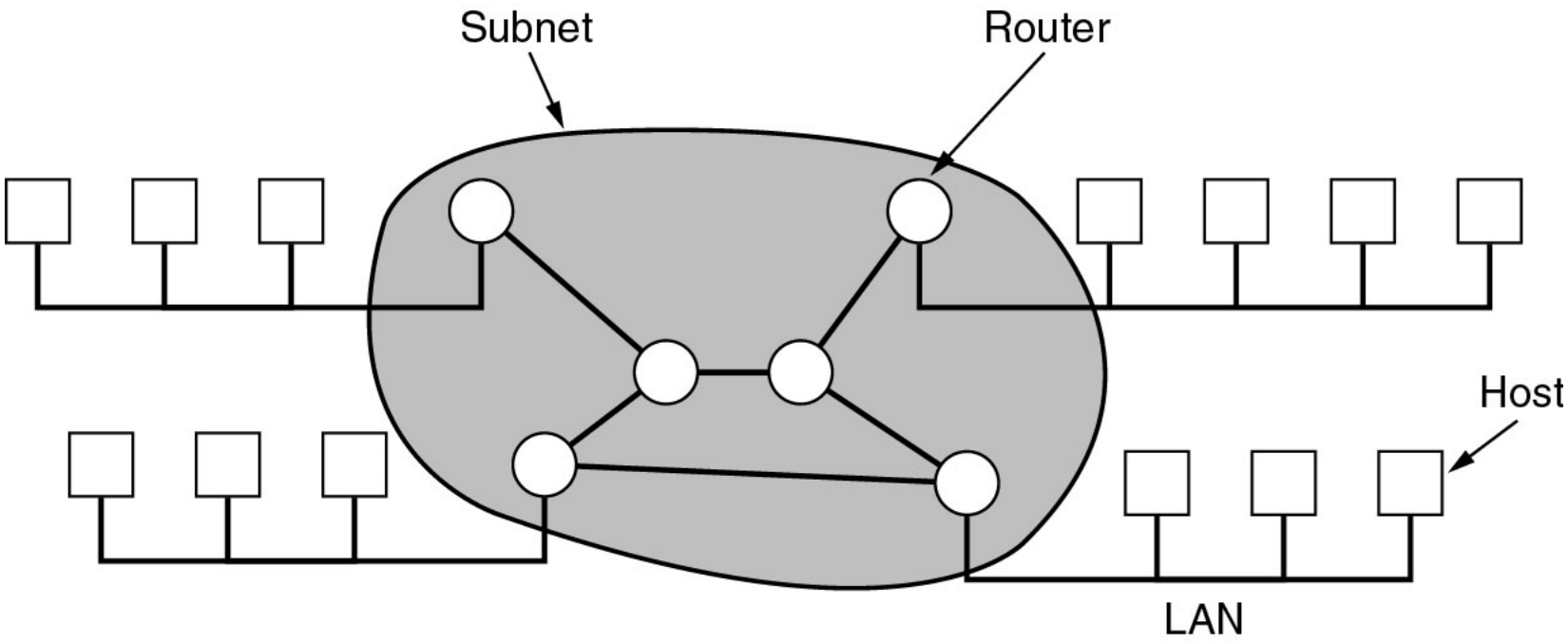
Two broadcast networks
(a) Bus
(b) Ring

Metropolitan Area Networks



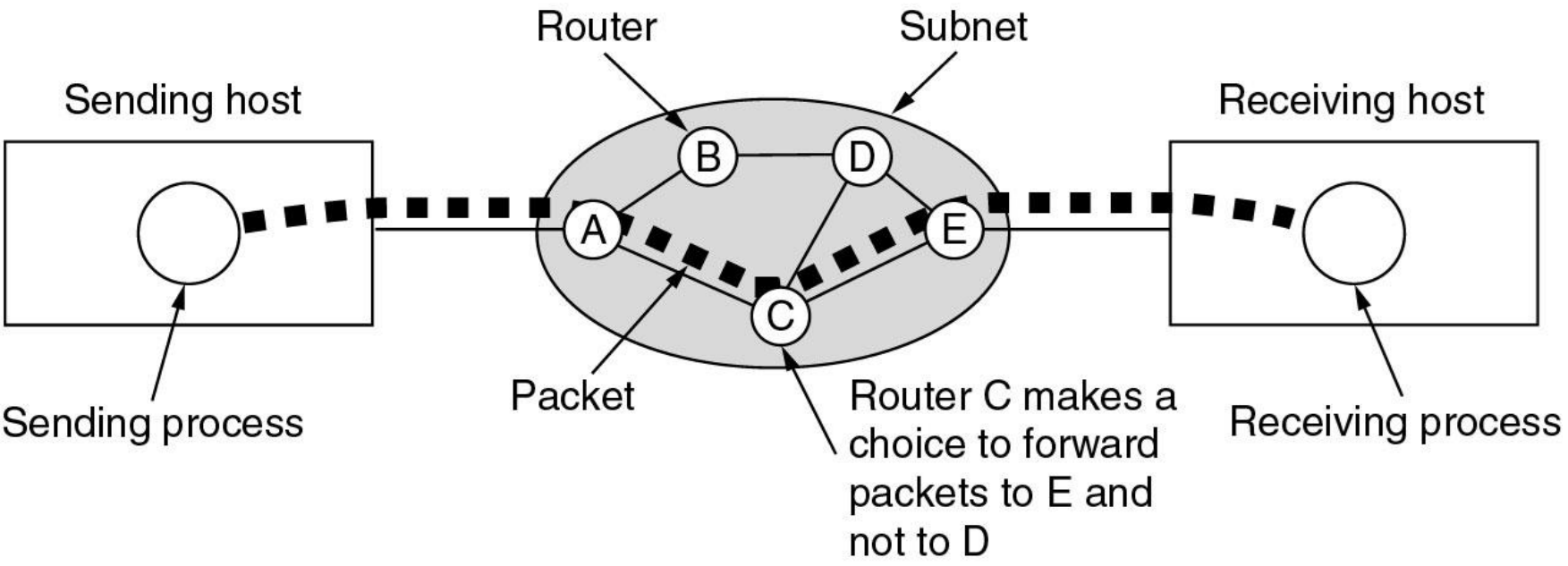
A metropolitan area network based on cable TV.

Wide Area Networks



Relation between hosts on LANs and the subnet.

Wide Area Networks (2)



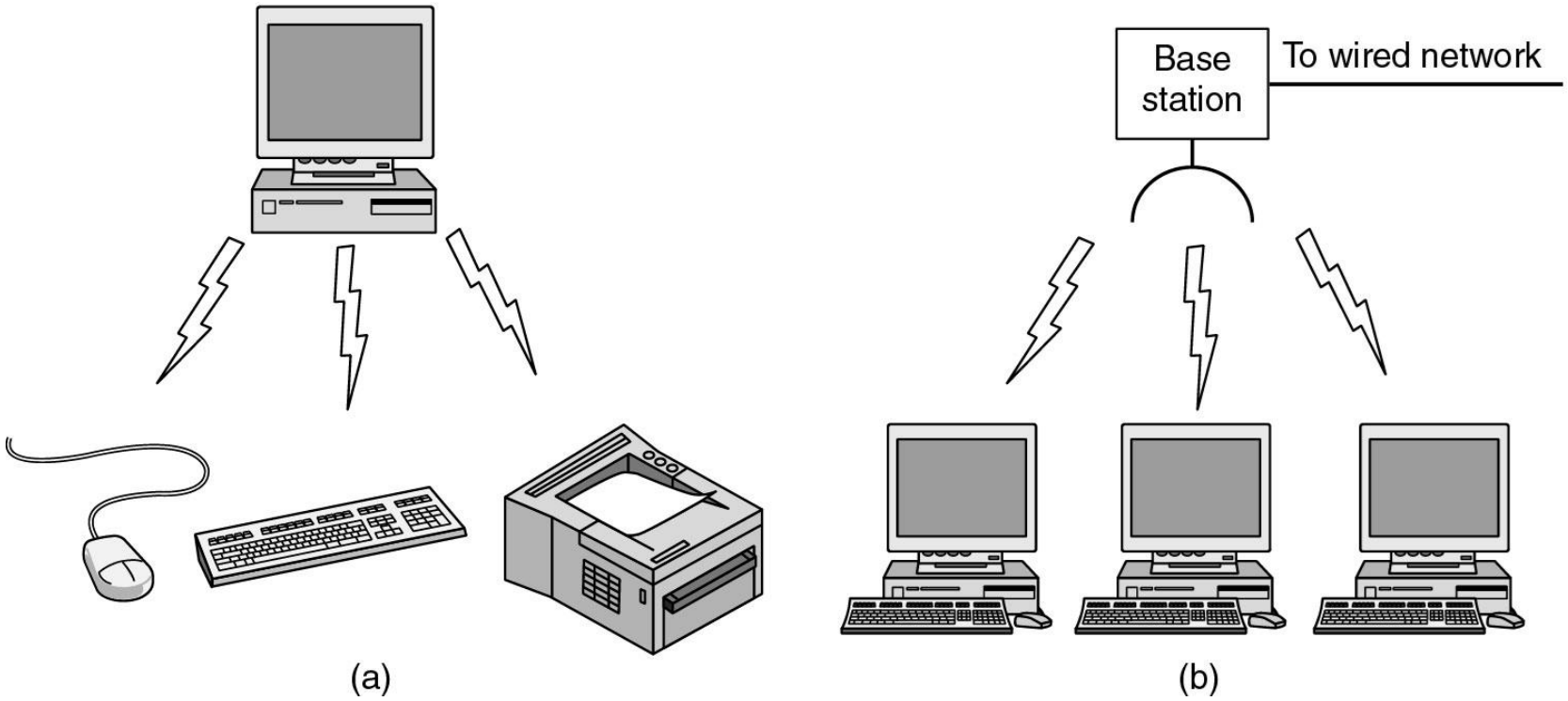
A stream of packets from sender to receiver.

Wireless Networks

Categories of wireless networks:

- System interconnection
- Wireless LANs
- Wireless WANs

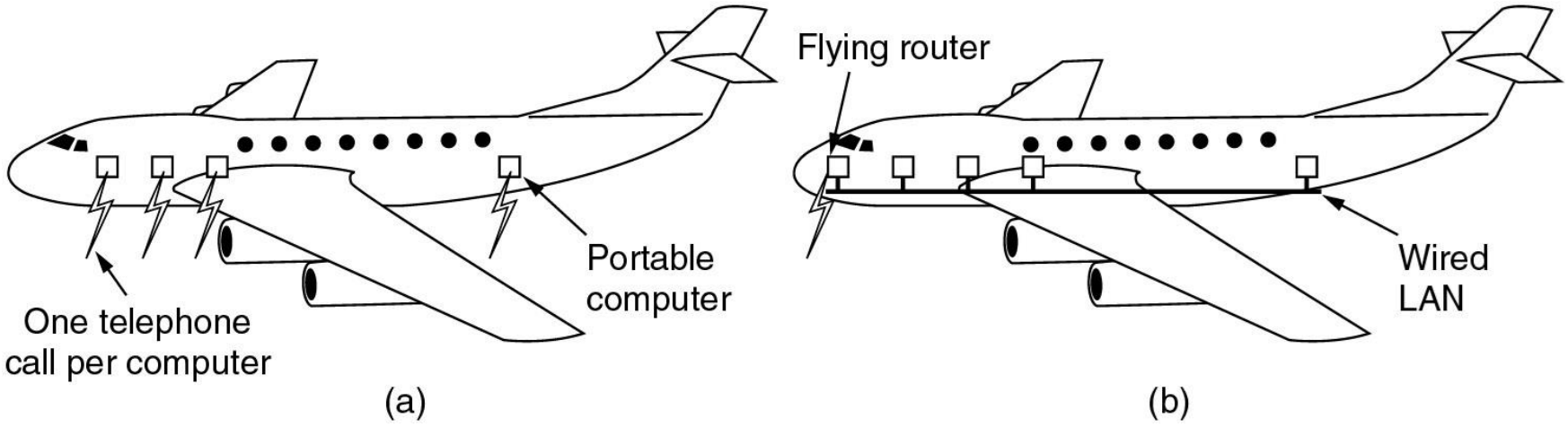
Wireless Networks (2)



(a) Bluetooth configuration

(b) Wireless LAN

Wireless Networks (3)



(a) Individual mobile computers

(b) A flying LAN

Home Network Categories

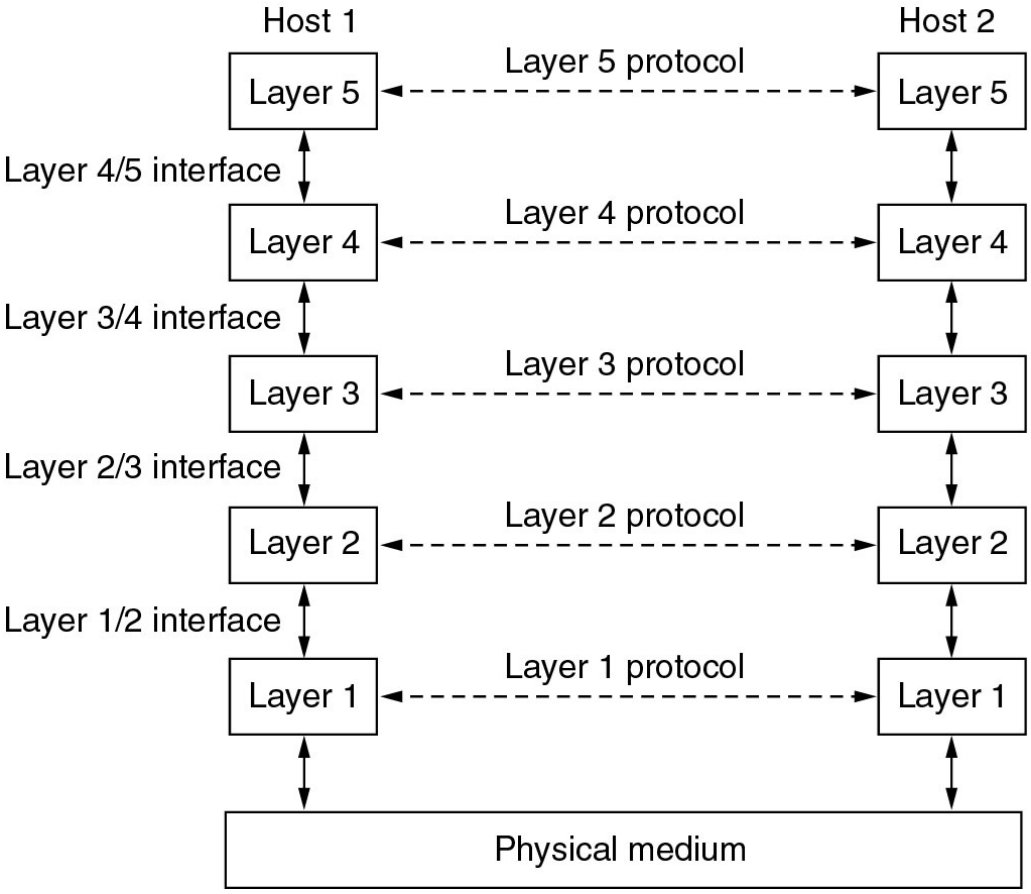
- Computers (desktop PC, PDA, shared peripherals)
- Entertainment (TV, DVD, VCR, camera, stereo, MP3)
- Telecomm (telephone, cell phone, intercom, fax)
- Appliances (microwave, fridge, clock, furnace, airco)
- Telemetry (utility meter, burglar alarm, babycam)

Network Software

- Protocol Hierarchies
- Design Issues for the Layers
- Connection-Oriented and Connectionless Services
- Service Primitives
- The Relationship of Services to Protocols

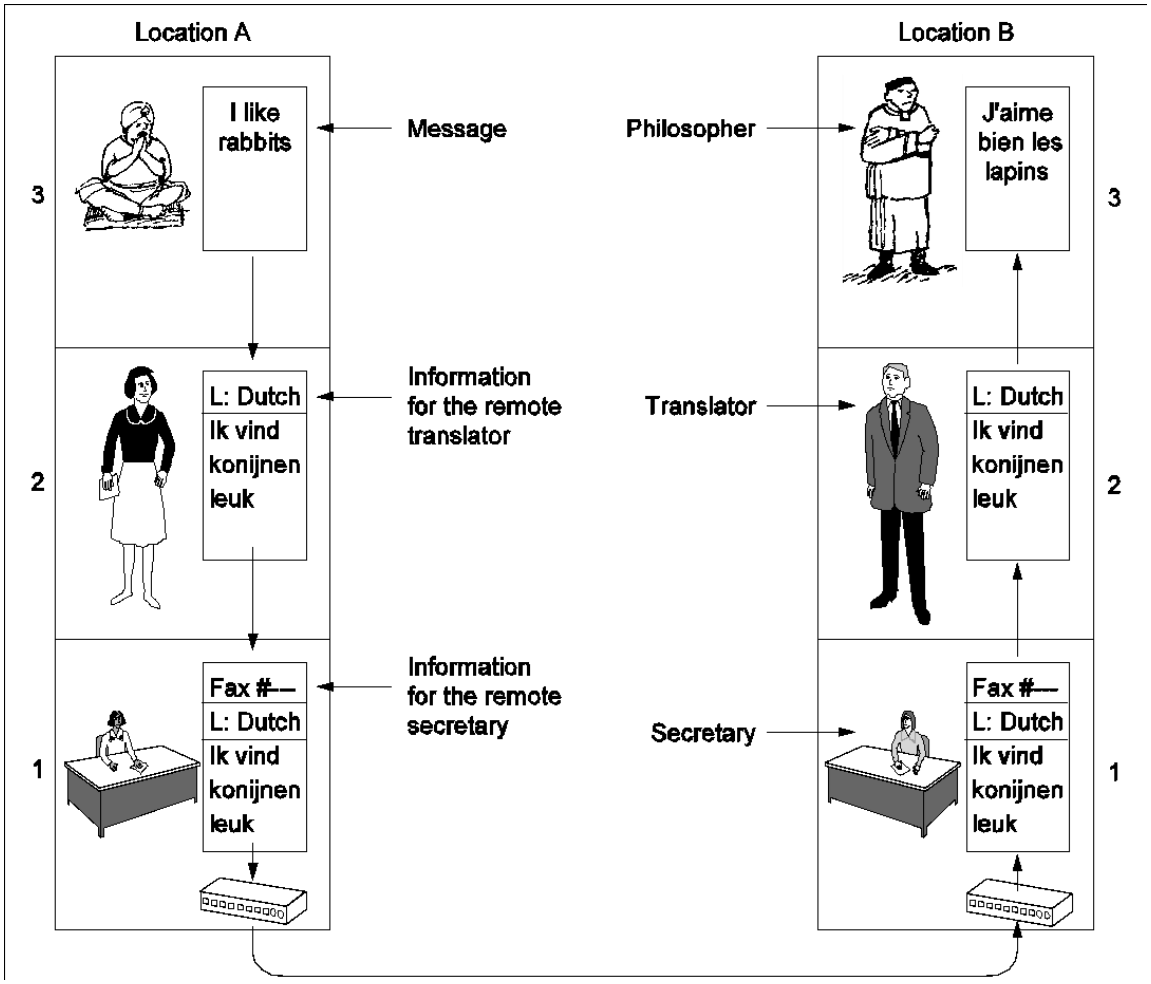
Network Software

Protocol Hierarchies



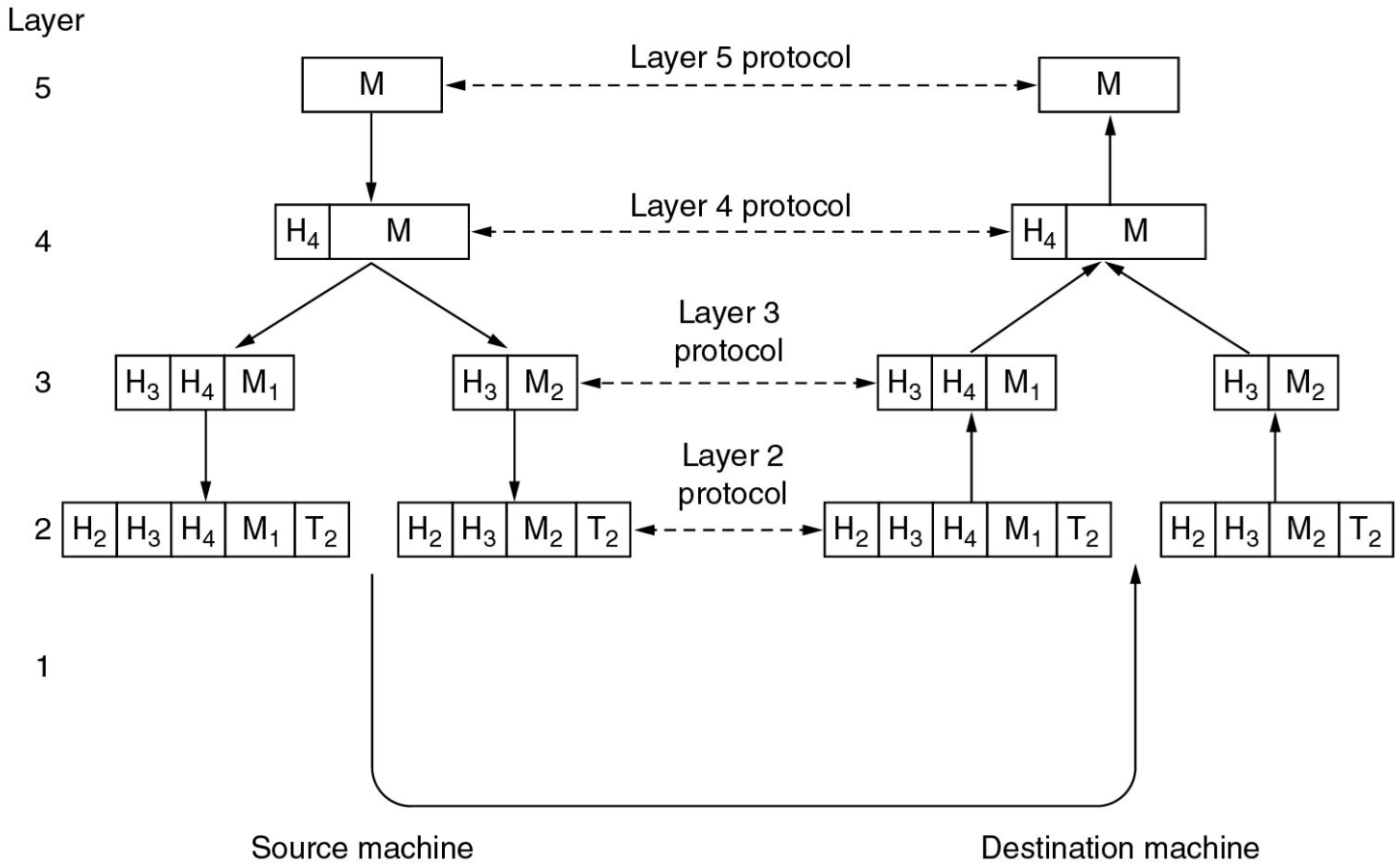
Layers, protocols, and interfaces.

Protocol Hierarchies (2)



The philosopher-translator-secretary architecture.

Protocol Hierarchies (3)



Example information flow supporting virtual communication in layer 5.

Design Issues for the Layers

- Addressing
- Error Control
- Flow Control
- Multiplexing
- Routing

Connection-Oriented and Connectionless Services

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Remote login
	Unreliable connection	Digitized voice
Connection-less	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Registered mail
	Request-reply	Database query

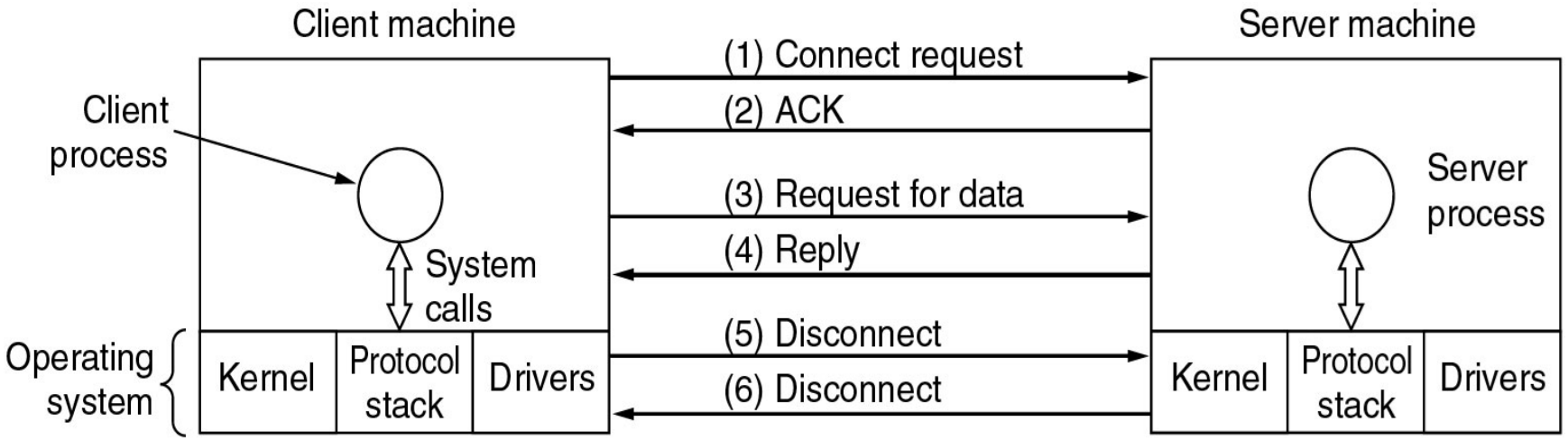
Six different types of service.

Service Primitives

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

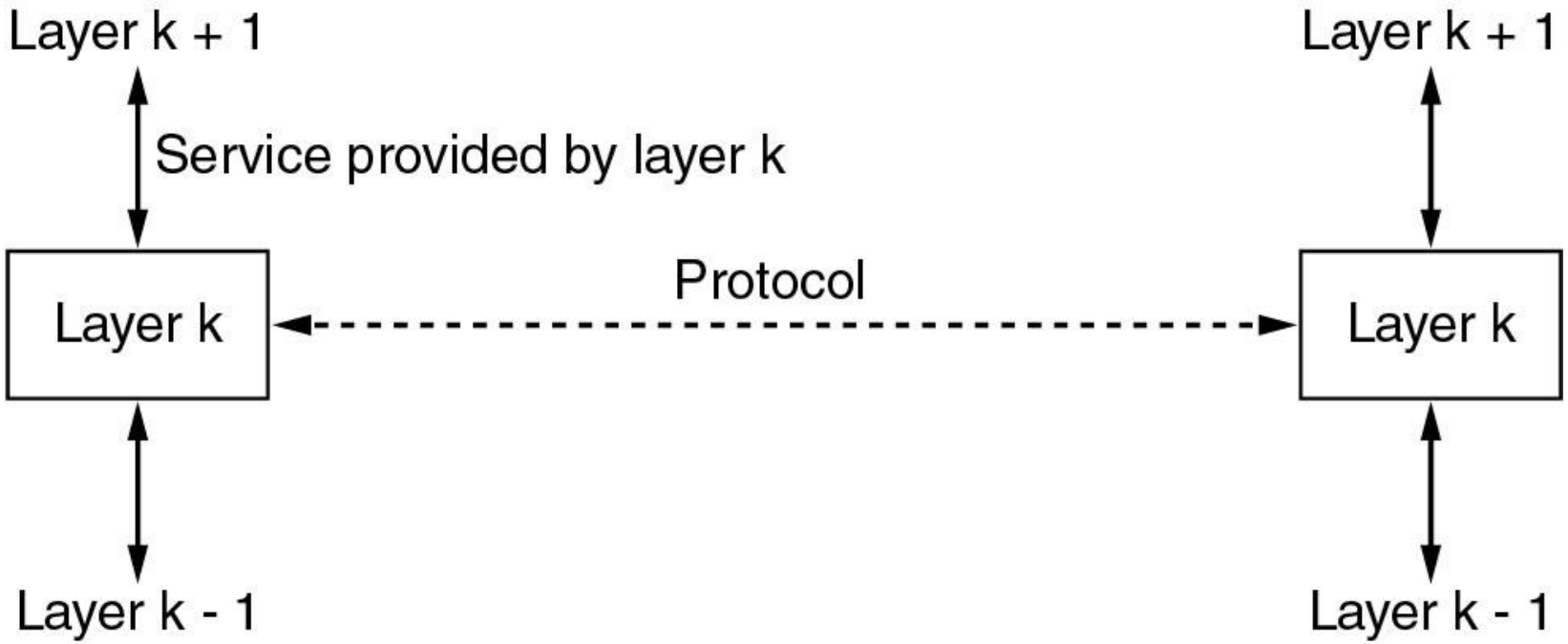
Five service primitives for implementing a simple connection-oriented service.

Service Primitives (2)



Packets sent in a simple client-server interaction on a connection-oriented network.

Services to Protocols Relationship



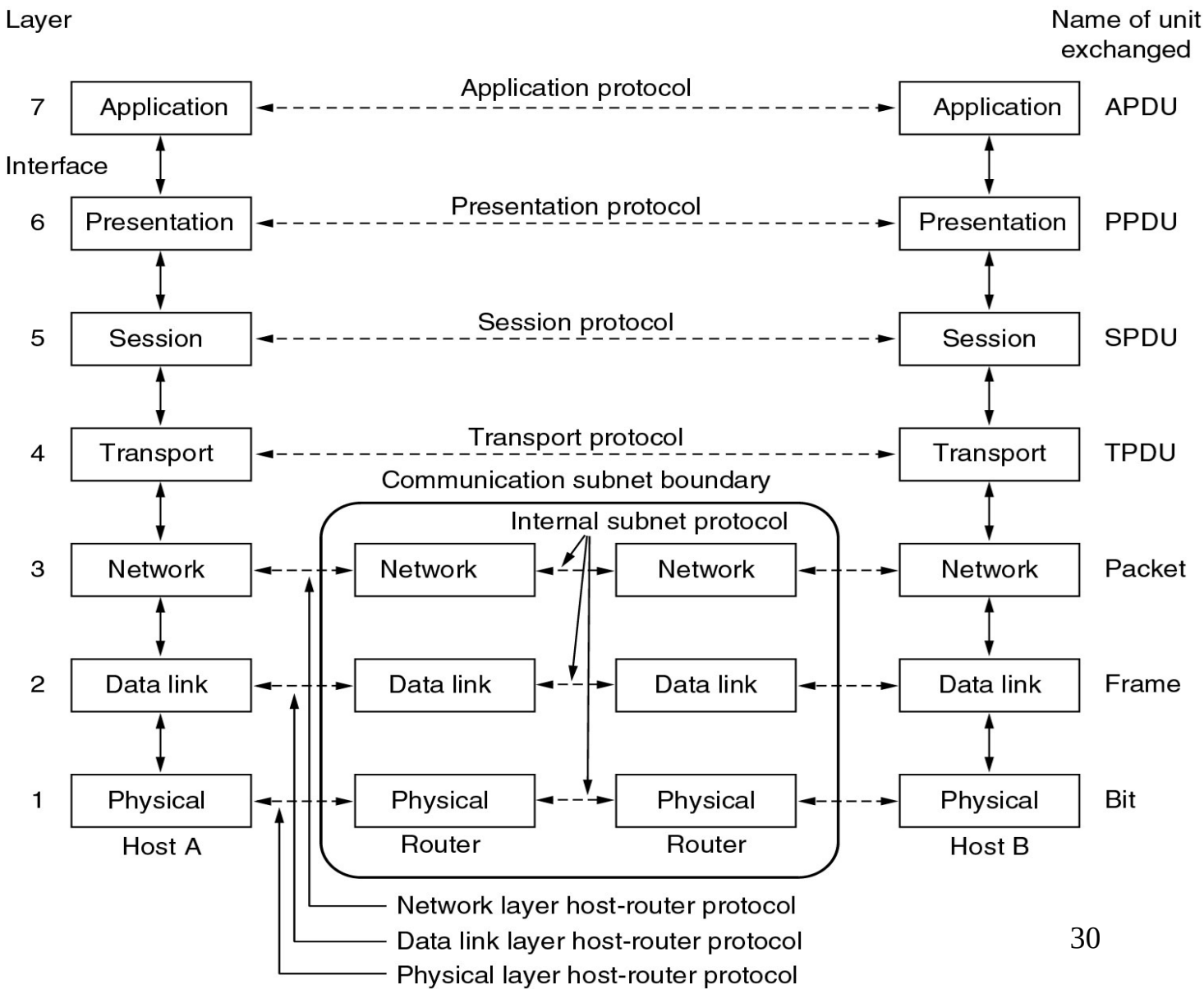
The relationship between a service and a protocol.

Reference Models

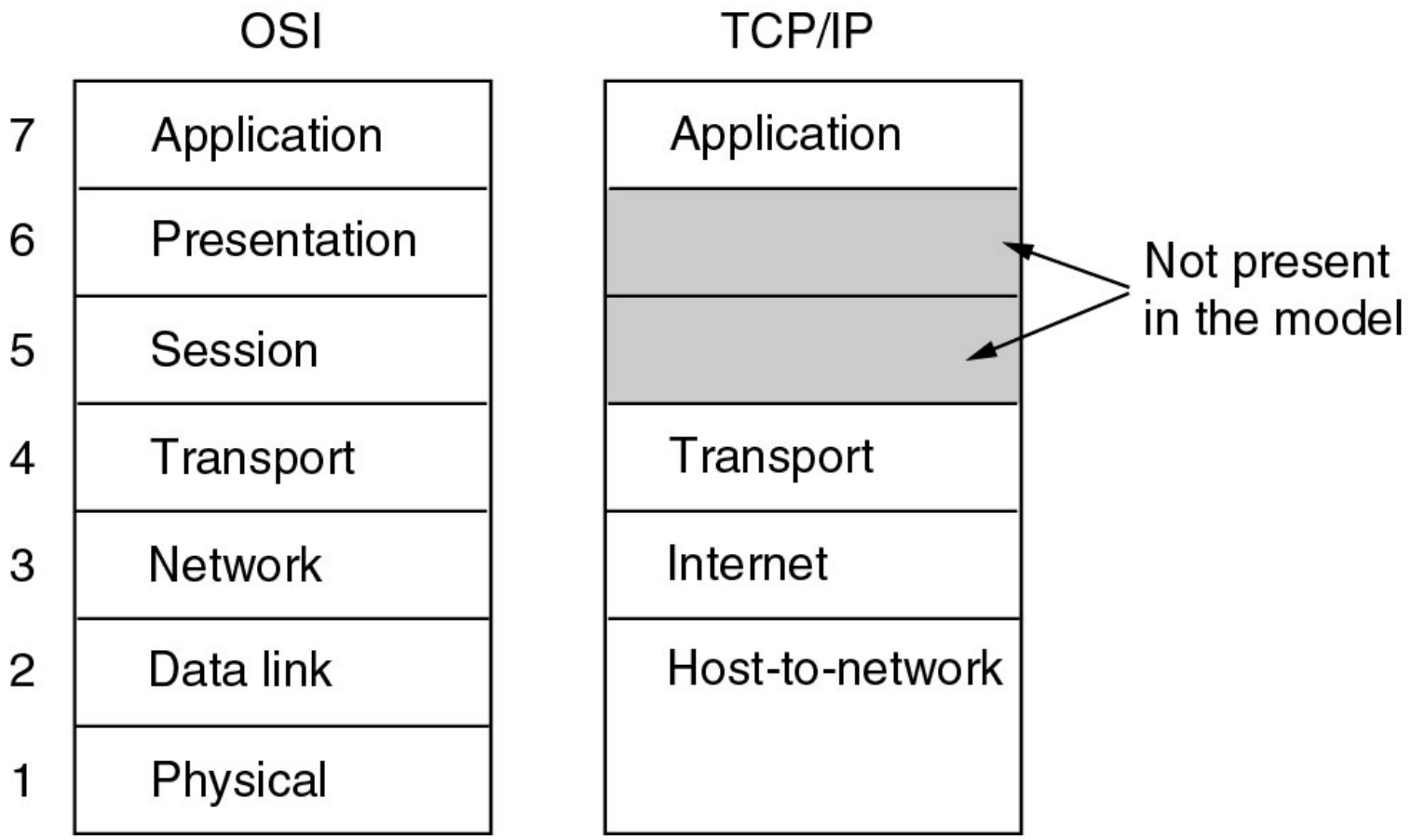
- The OSI Reference Model
- The TCP/IP Reference Model
- A Comparison of OSI and TCP/IP
- A Critique of the OSI Model and Protocols
- A Critique of the TCP/IP Reference Model

Reference Models

The OSI reference model.

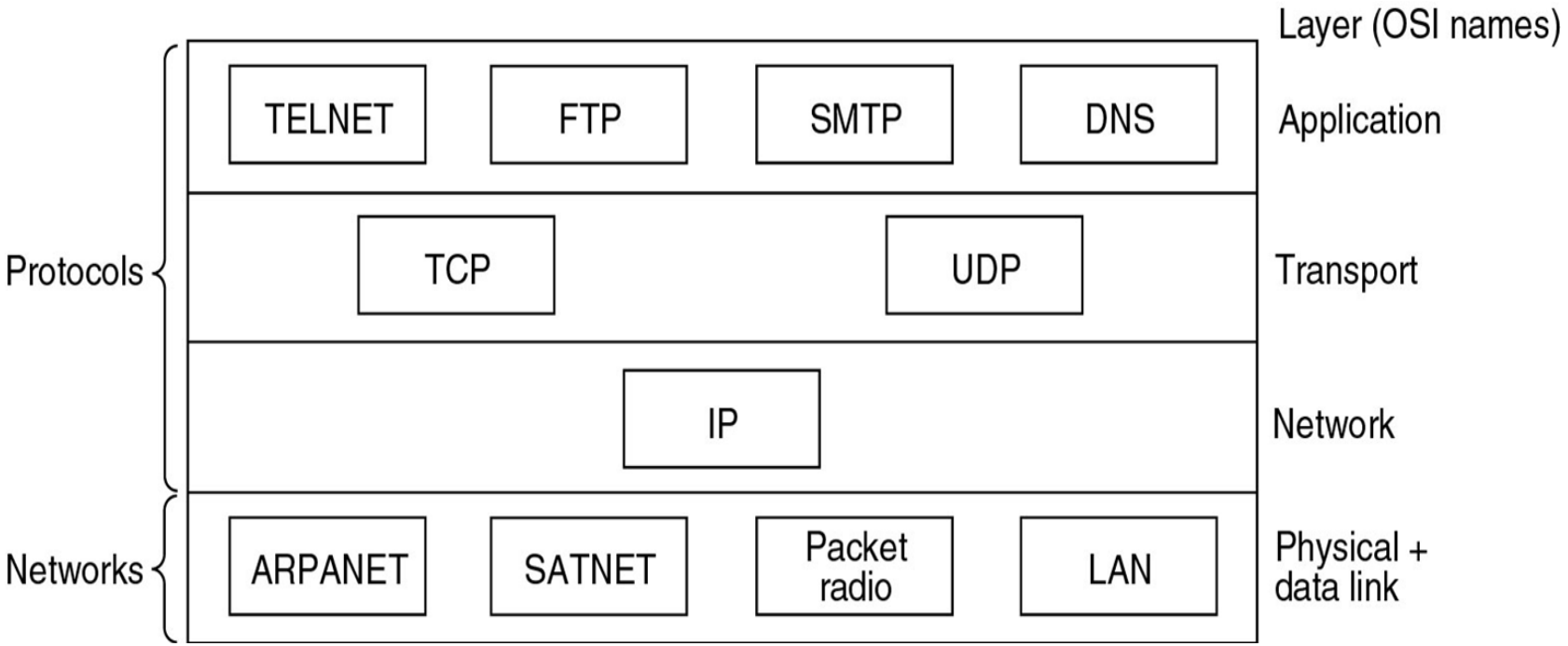


Reference Models (2)



The TCP/IP reference model.

Reference Models (3)



Protocols and networks in the TCP/IP model initially.

Comparing OSI and TCP/IP Models

Concepts central to the OSI model

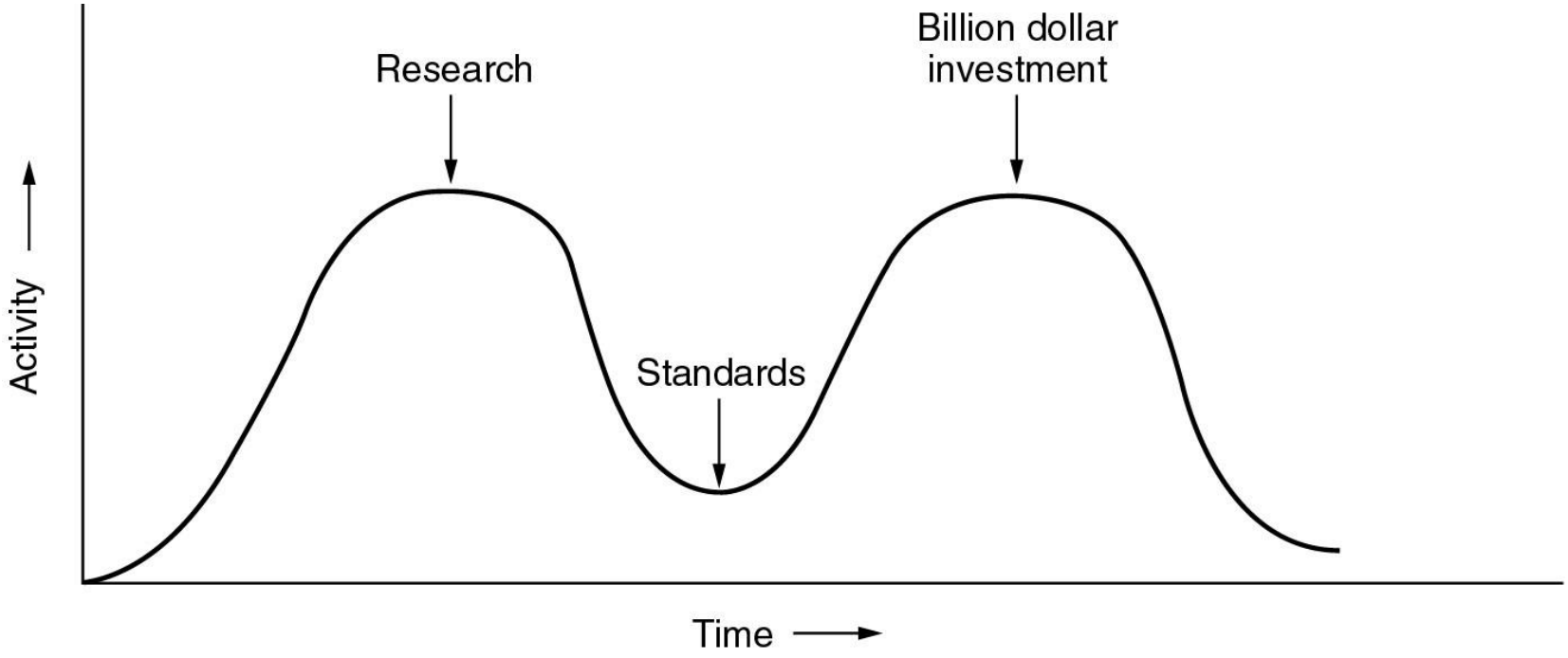
- Services
- Interfaces
- Protocols

A Critique of the OSI Model and Protocols

Why OSI did not take over the world

- Bad timing
- Bad technology
- Bad implementations
- Bad politics

Bad Timing



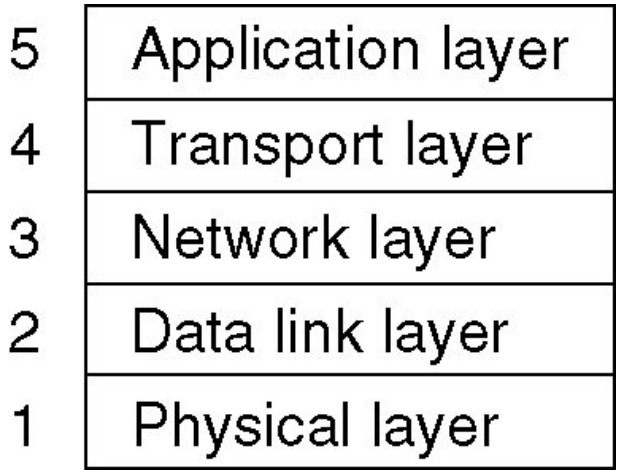
The apocalypse of the two elephants.

A Critique of the TCP/IP Reference Model

Problems:

- Service, interface, and protocol not distinguished
- Not a general model
- Host-to-network “layer” not really a layer
- No mention of physical and data link layers
- Minor protocols deeply entrenched, hard to replace

Hybrid Model

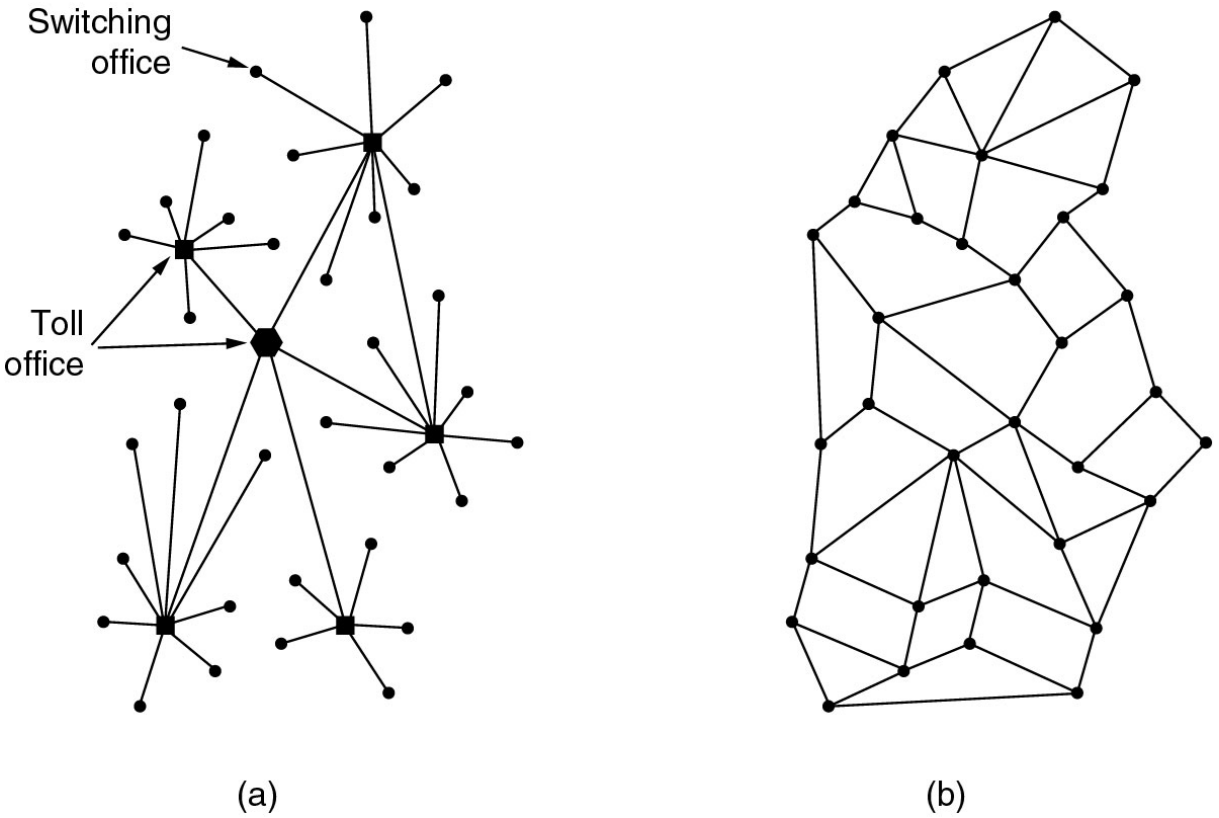


The hybrid reference model to be used in this book.

Example Networks

- The Internet
- Connection-Oriented Networks:
  X.25, Frame Relay, and ATM
- Ethernet
- Wireless LANs: 802.11

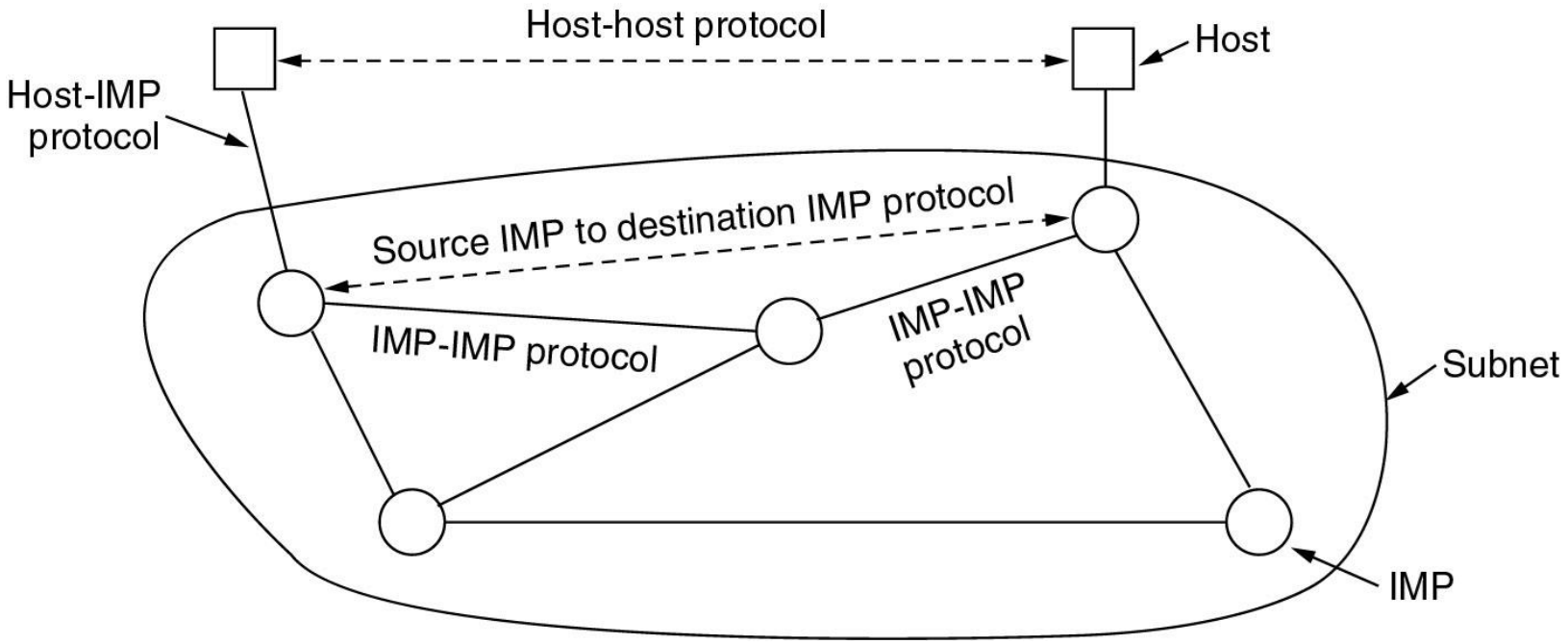
The ARPANET



(a) Structure of the telephone system.

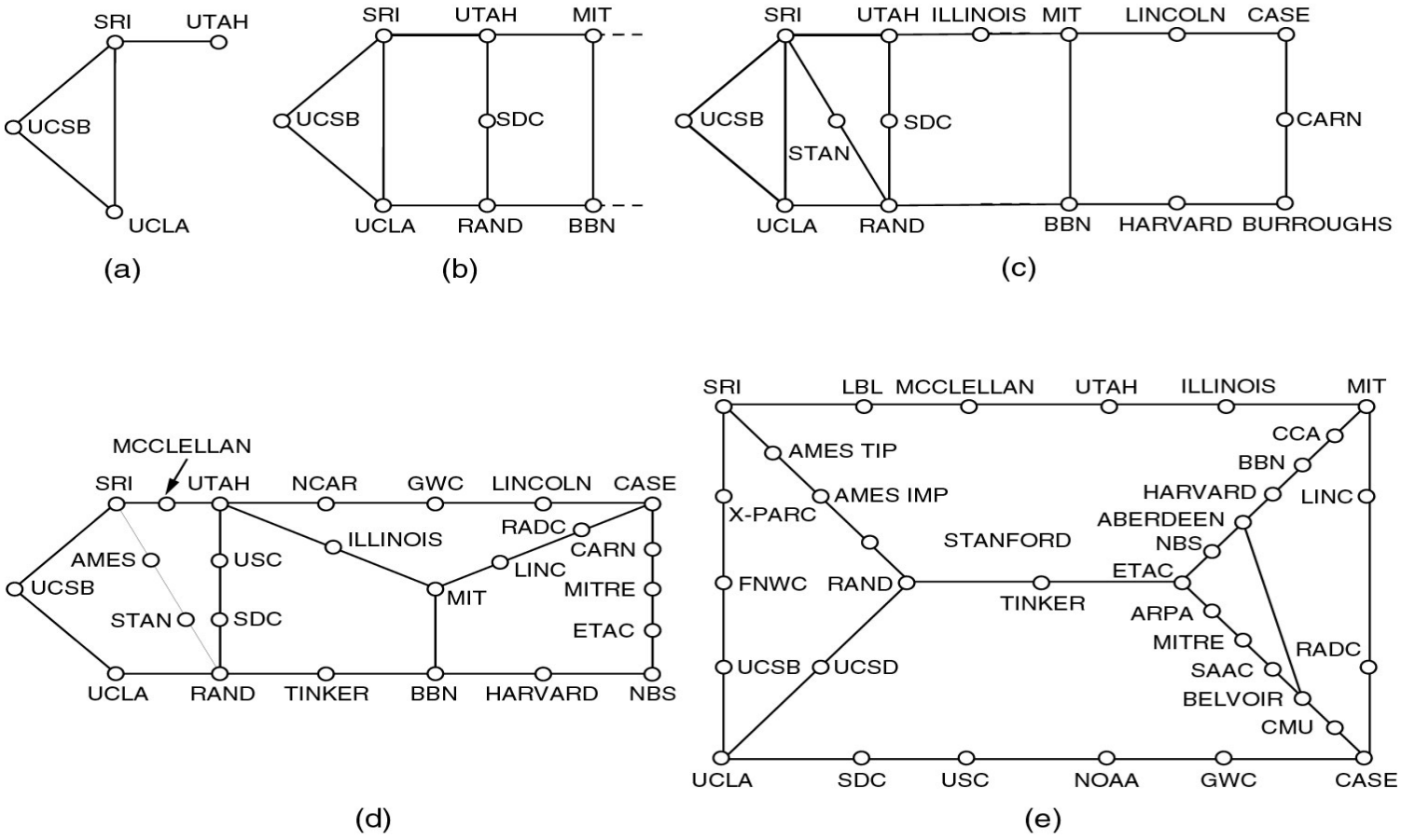
(b) Baran's proposed distributed switching system.

The ARPANET (2)



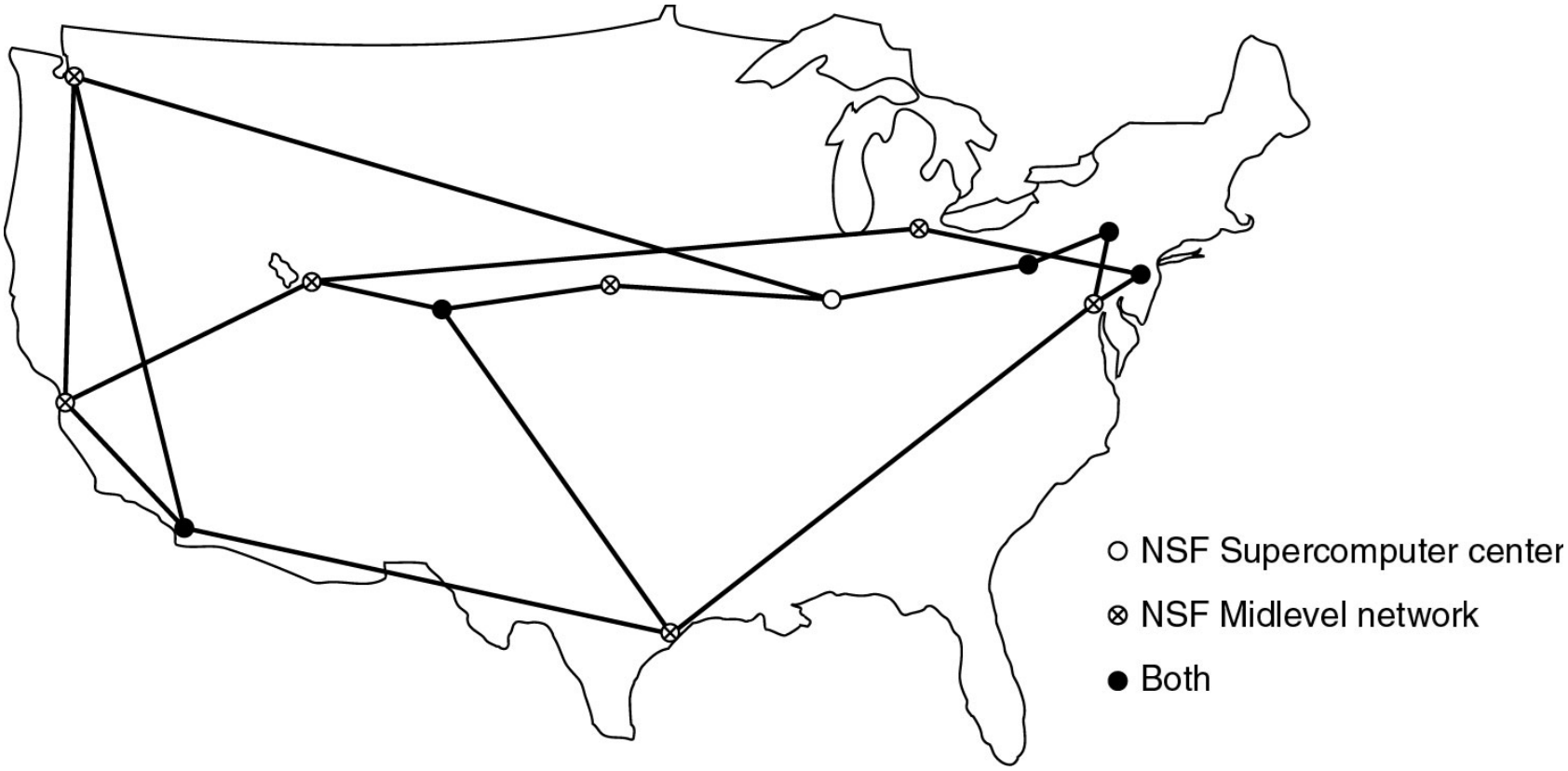
The original ARPANET design.

The ARPANET (3)



Growth of the ARPANET (a) December 1969. (b) July 1970. (c) March 1971. (d) April 1972. (e) September 1972.

NSFNET



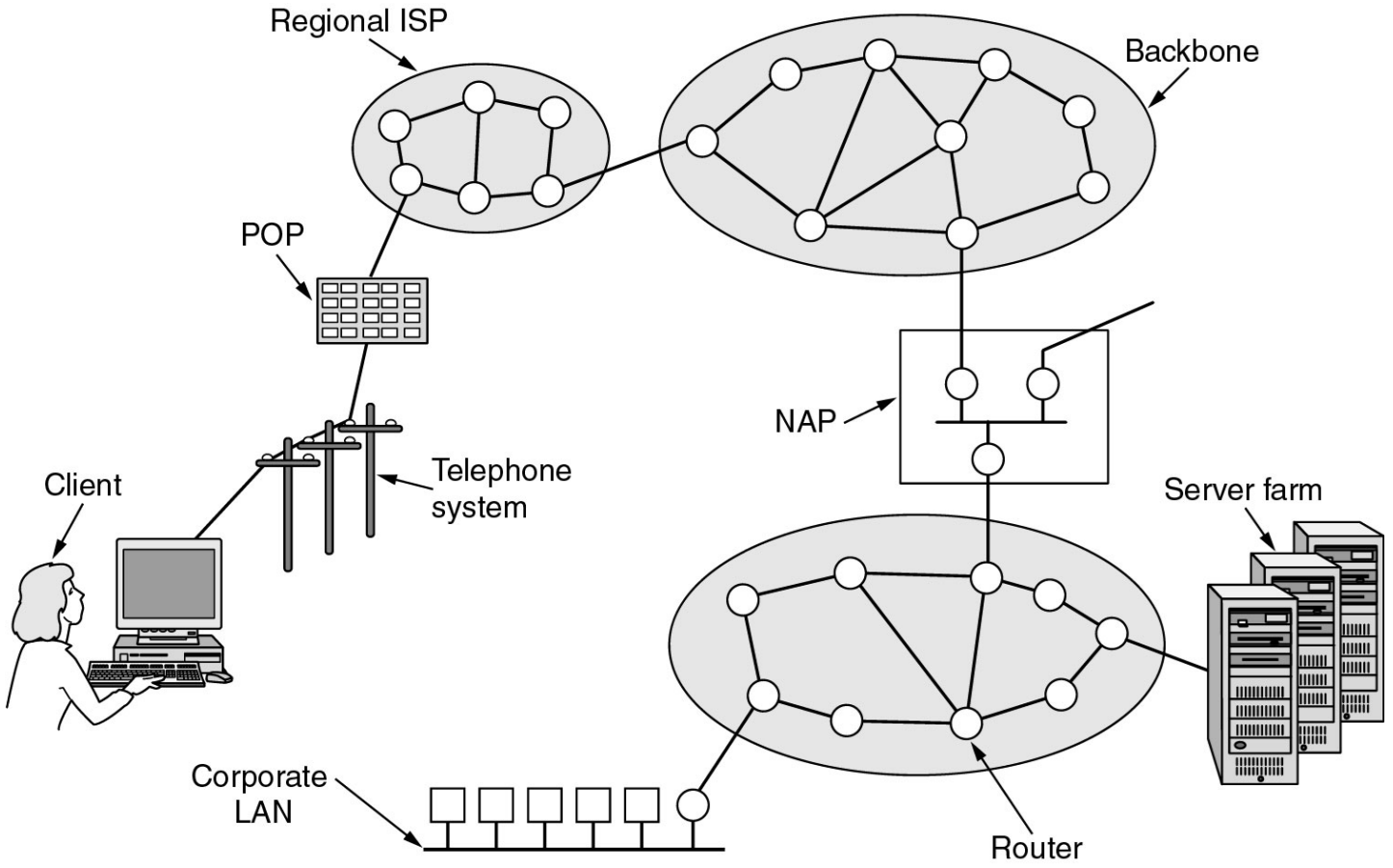
The NSFNET backbone in 1988.

Internet Usage

Traditional applications (1970 – 1990)

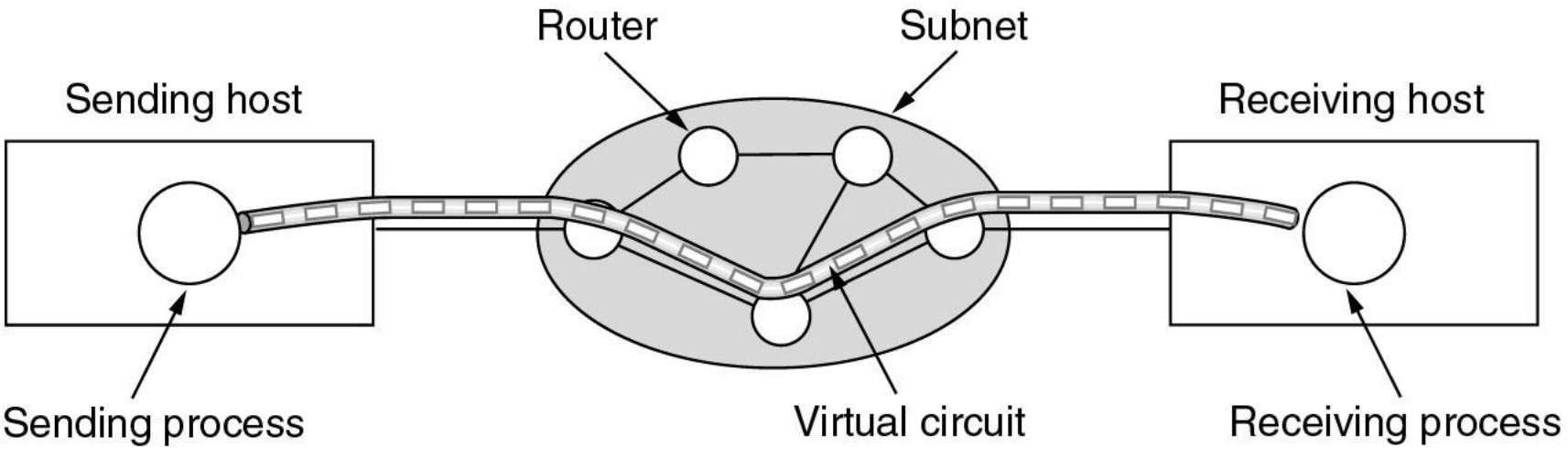
- E-mail
- News
- Remote login
- File transfer

Architecture of the Internet



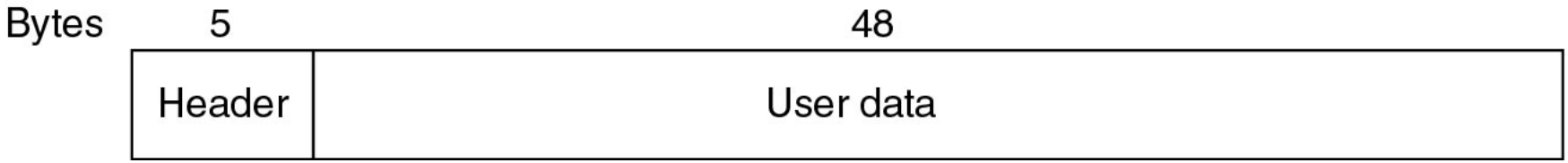
Overview of the Internet.

ATM Virtual Circuits



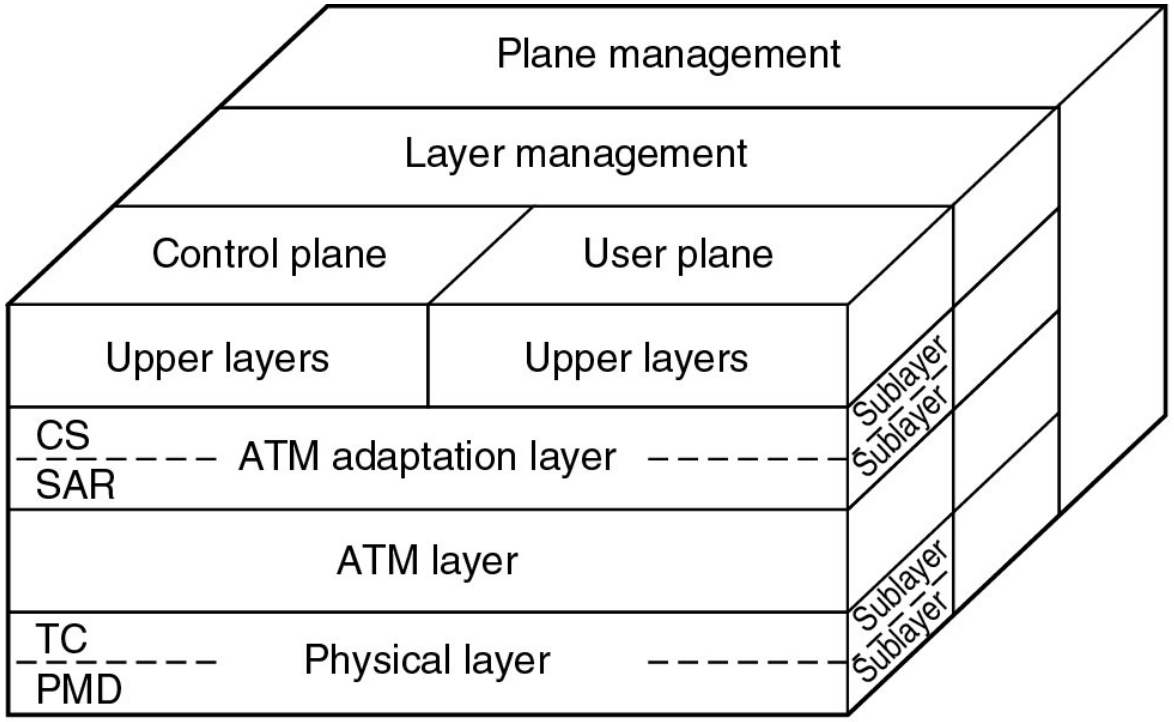
A virtual circuit.

ATM Virtual Circuits (2)



An ATM cell.

The ATM Reference Model



CS: Convergence sublayer
SAR: Segmentation and reassembly sublayer
TC: Transmission convergence sublayer
PMD: Physical medium dependent sublayer

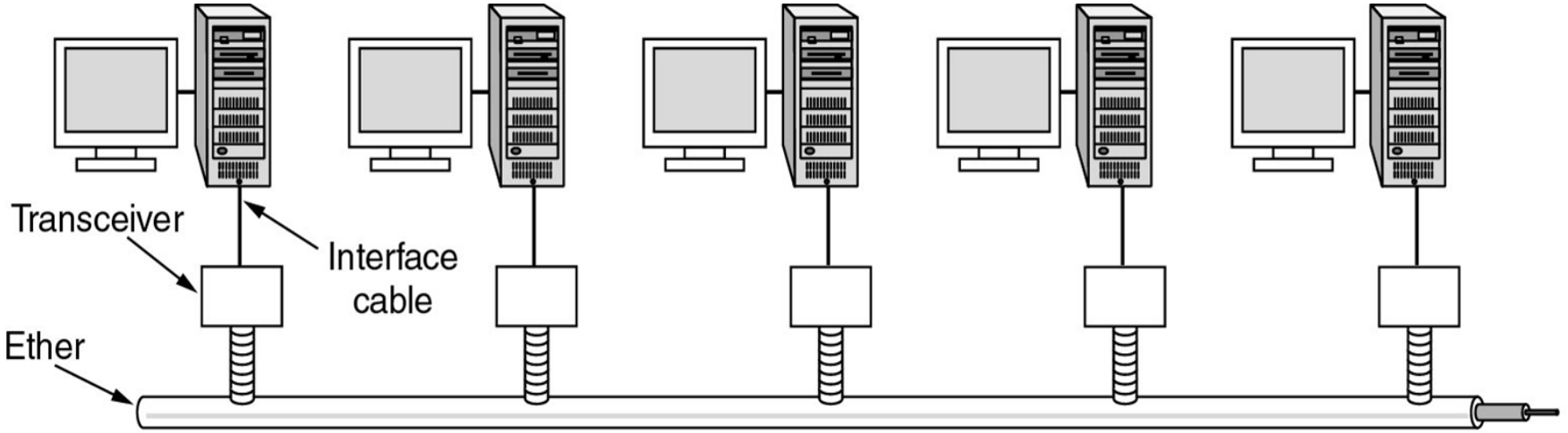
The ATM reference model.

The ATM Reference Model (2)

OSI layer	ATM layer	ATM sublayer	Functionality
3/4	AAL	CS	Providing the standard interface (convergence)
		SAR	Segmentation and reassembly
2/3	ATM		Flow control Cell header generation/extraction Virtual circuit/path management Cell multiplexing/demultiplexing
2	Physical	TC	Cell rate decoupling Header checksum generation and verification Cell generation Packing/unpacking cells from the enclosing envelope Frame generation
1		PMD	Bit timing Physical network access

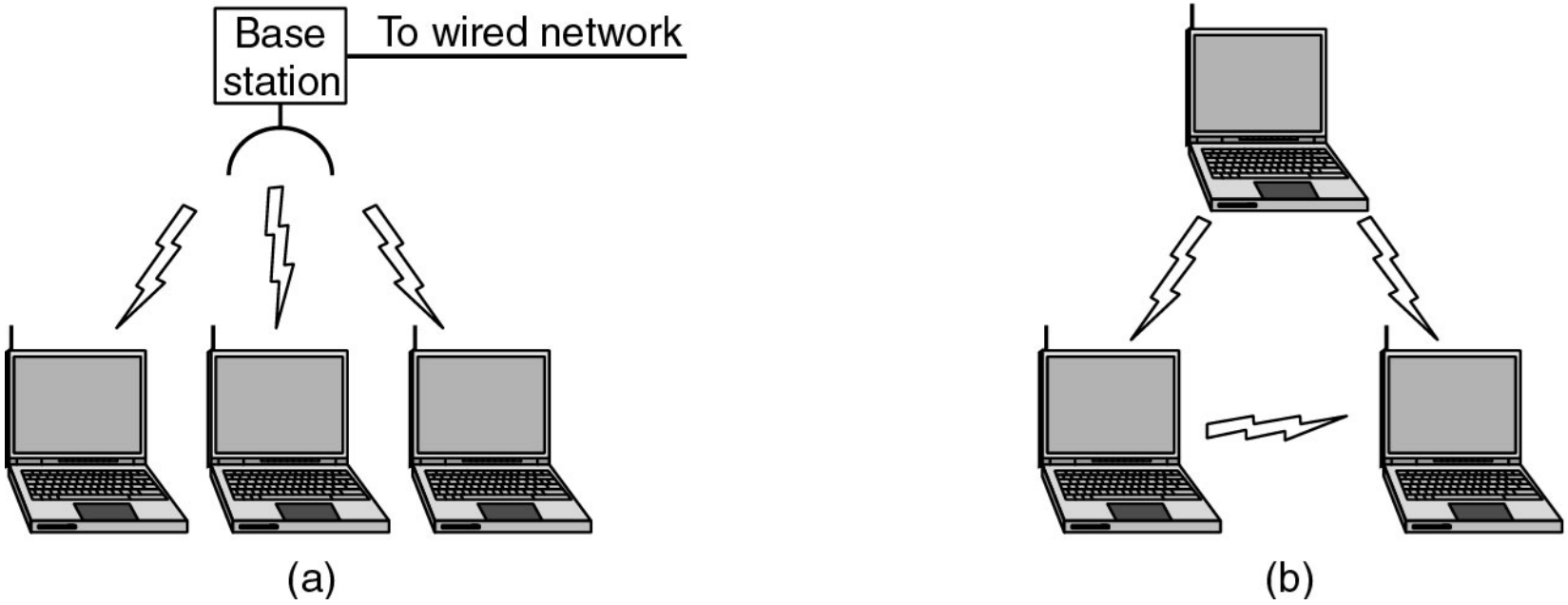
The ATM layers and sublayers and their functions.

Ethernet



Architecture of the original Ethernet.

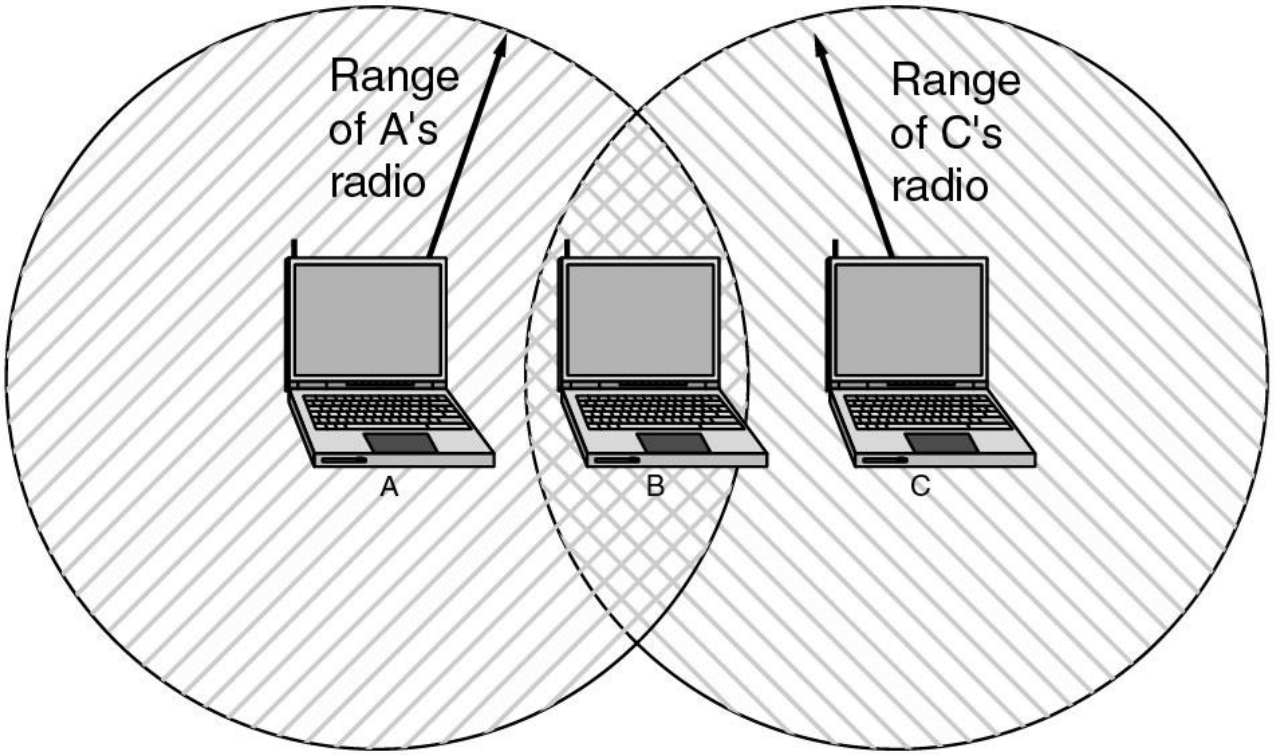
Wireless LANs



(a) Wireless networking with a base station.

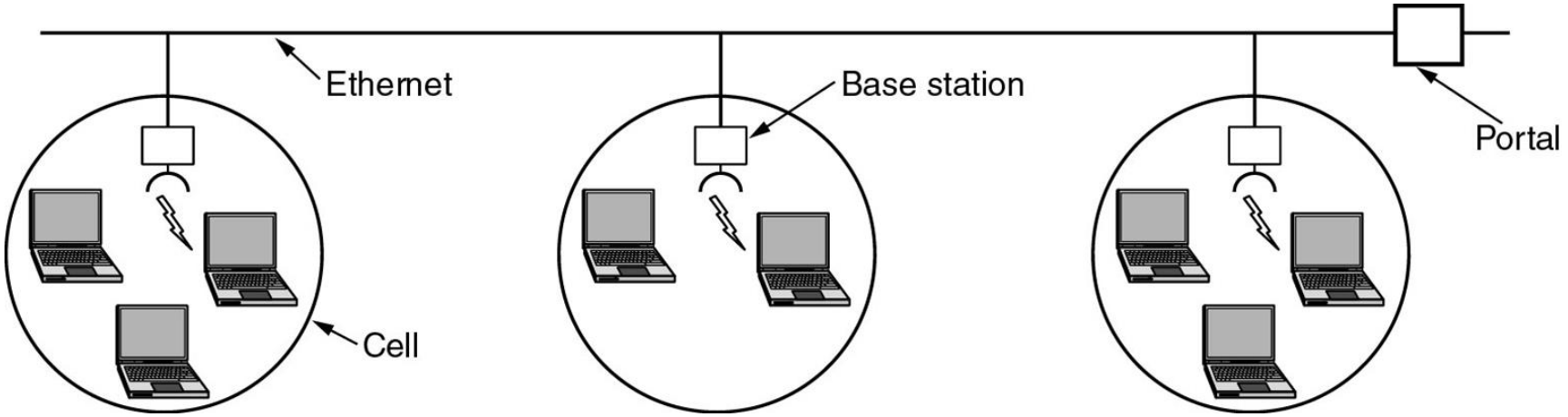
(b) Ad hoc networking.

Wireless LANs (2)



The range of a single radio may not cover the entire system.

Wireless LANs (3)



A multicell 802.11 network.

Network Standardization

- Who's Who in the Telecommunications World
- Who's Who in the International Standards World
- Who's Who in the Internet Standards World

ITU

- Main sectors
  - Radiocommunications
  - Telecommunications Standardization
  - Development
- Classes of Members
  - National governments
  - Sector members
  - Associate members
  - Regulatory agencies

IEEE 802 Standards

Number	Topic
802.1	Overview and architecture of LANs
802.2 ↓	Logical link control
802.3 *	Ethernet
802.4 ↓	Token bus (was briefly used in manufacturing plants)
802.5	Token ring (IBM's entry into the LAN world)
802.6 ↓	Dual queue dual bus (early metropolitan area network)
802.7 ↓	Technical advisory group on broadband technologies
802.8 †	Technical advisory group on fiber optic technologies
802.9 ↓	Isochronous LANs (for real-time applications)
802.10 ↓	Virtual LANs and security
802.11 *	Wireless LANs
802.12 ↓	Demand priority (Hewlett-Packard's AnyLAN)
802.13	Unlucky number. Nobody wanted it
802.14 ↓	Cable modems (defunct: an industry consortium got there first)
802.15 *	Personal area networks (Bluetooth)
802.16 *	Broadband wireless
802.17	Resilient packet ring

The 802 working groups. The important ones are marked with *. The ones marked with □ are hibernating. The one marked with † gave up.

Metric Units

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

The principal metric prefixes.

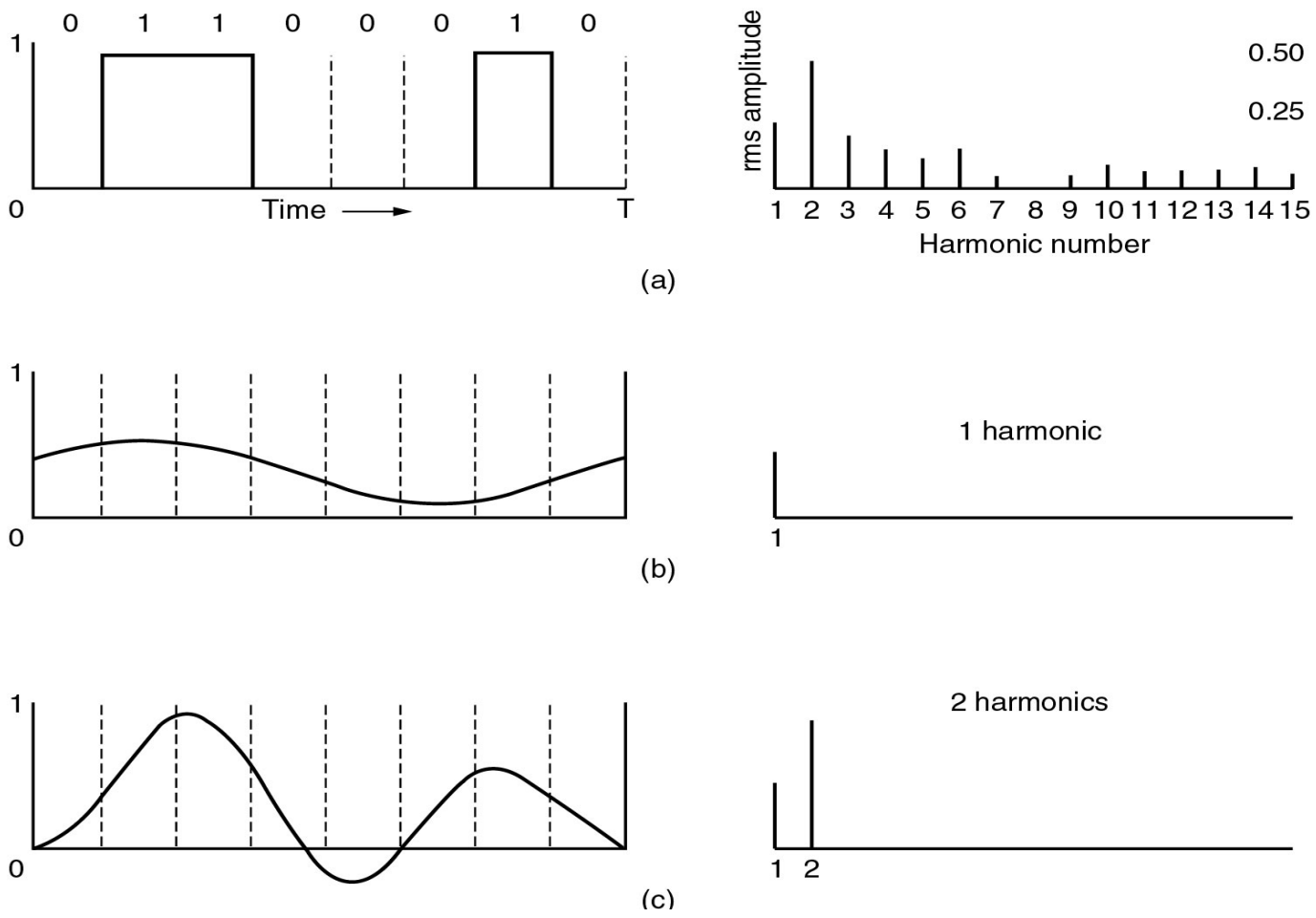
Chapter 2

The Physical Layer

The Theoretical Basis for Data Communication

- Fourier Analysis
- Bandwidth-Limited Signals
- Maximum Data Rate of a Channel

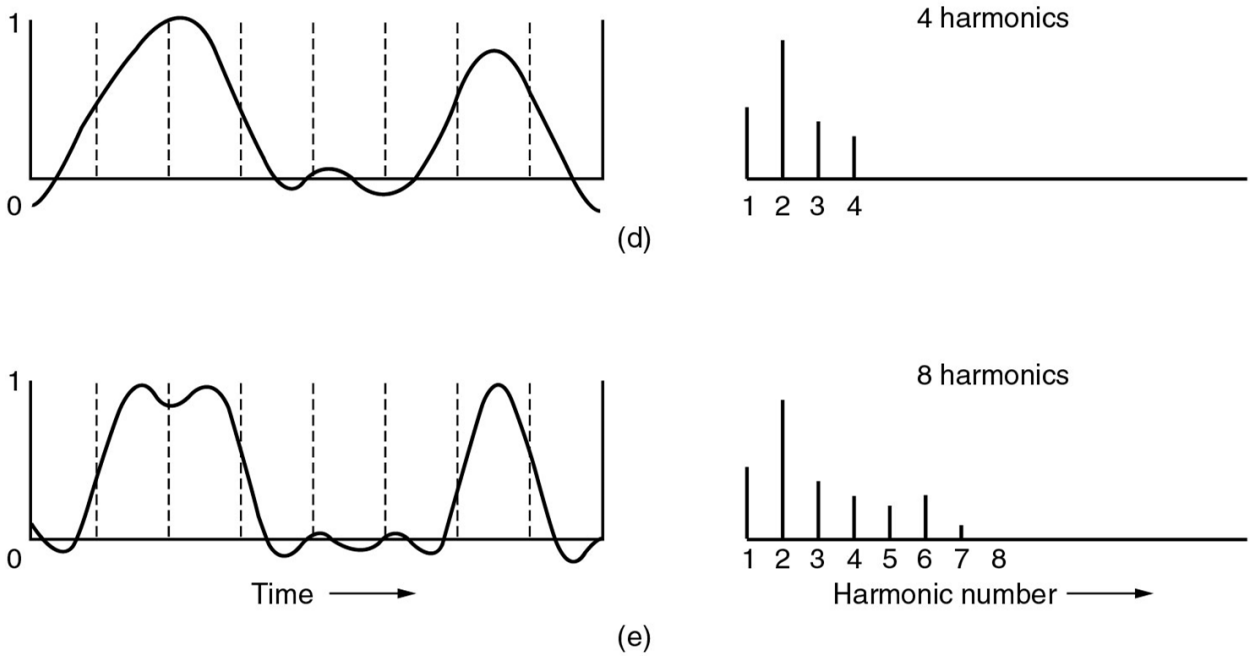
Bandwidth-Limited Signals



A binary signal and its root-mean-square Fourier amplitudes.

(b) – (c) Successive approximations to the original signal.

Bandwidth-Limited Signals (2)



(d) – (e) Successive approximations to the original signal.

Bandwidth-Limited Signals (3)

Bps	T (msec)	First harmonic (Hz)	# Harmonics sent
300	26.67	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

Relation between data rate and harmonics.

Guided Transmission Data

- Magnetic Media
- Twisted Pair
- Coaxial Cable
- Fiber Optics

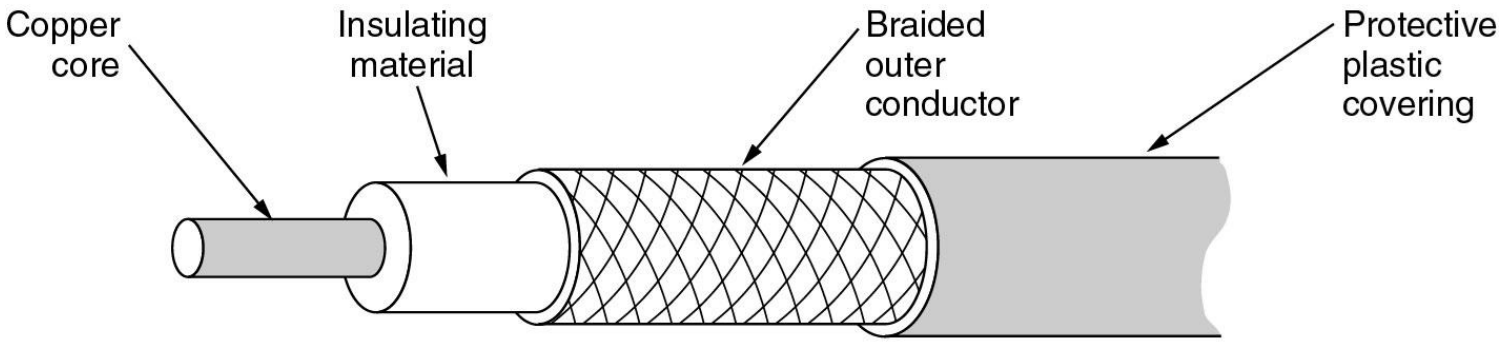
Twisted Pair

(a)

(b)

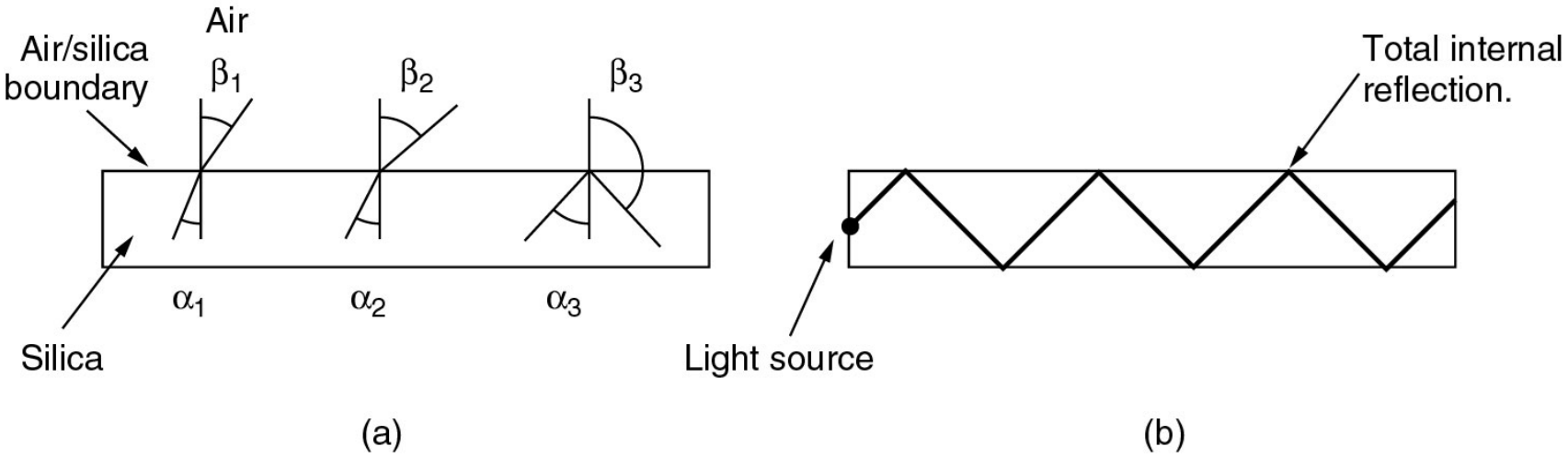
(a) Category 3 UTP.
(b) Category 5 UTP.

Coaxial Cable



A coaxial cable.

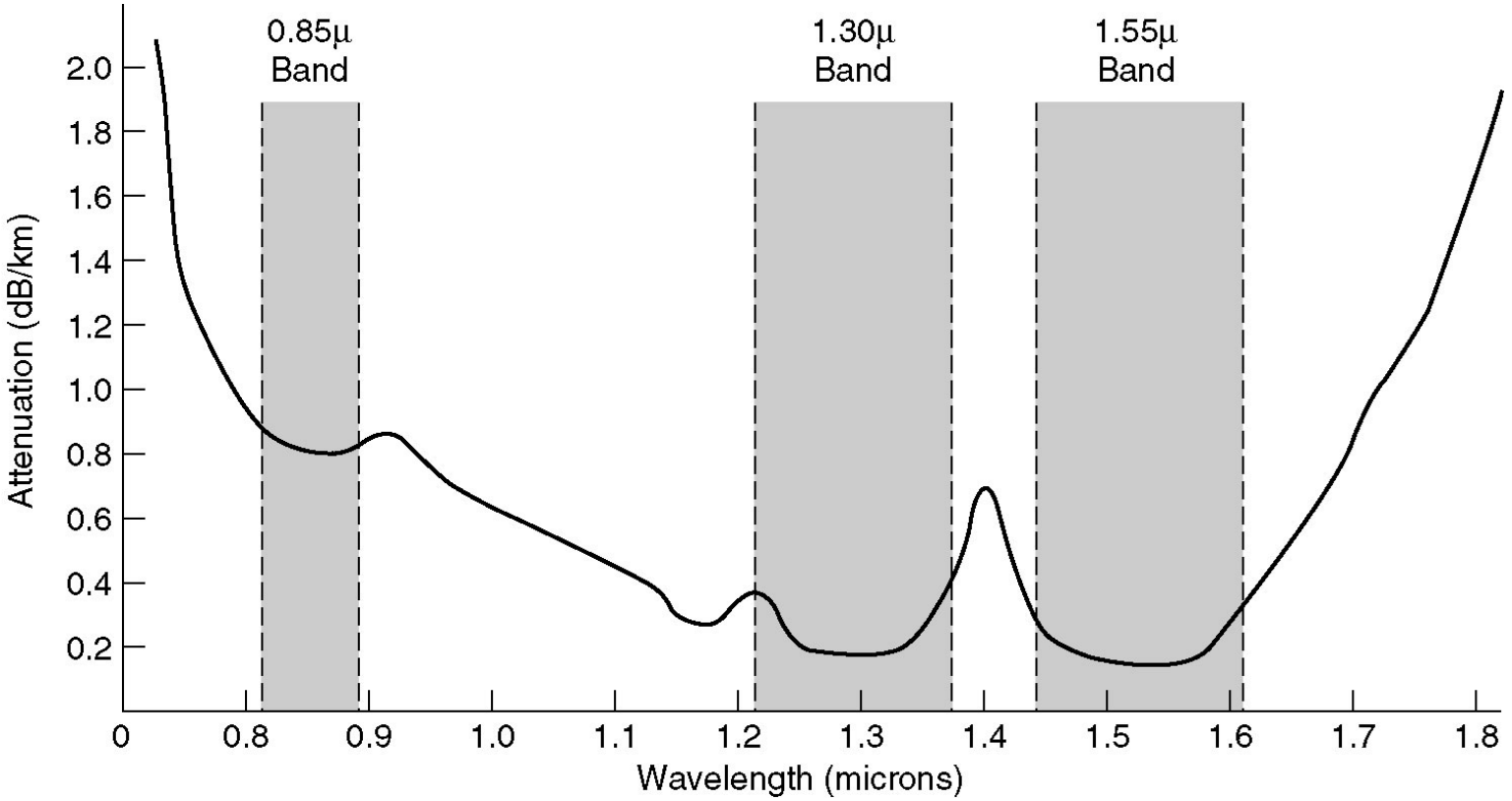
Fiber Optics



(a) Three examples of a light ray from inside a silica fiber impinging on the air/silica boundary at different angles.

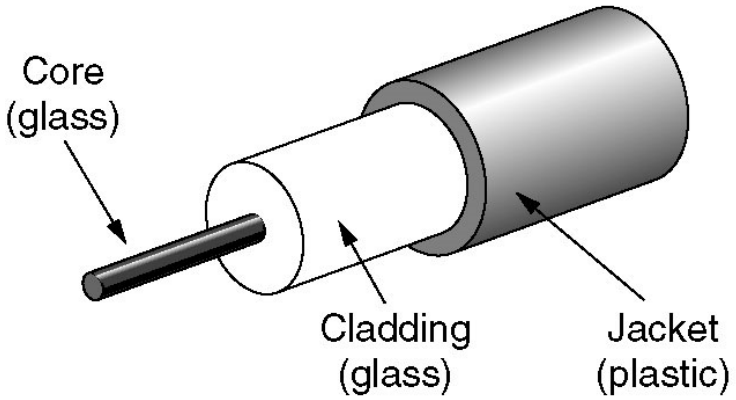
(b) Light trapped by total internal reflection.

Transmission of Light through Fiber

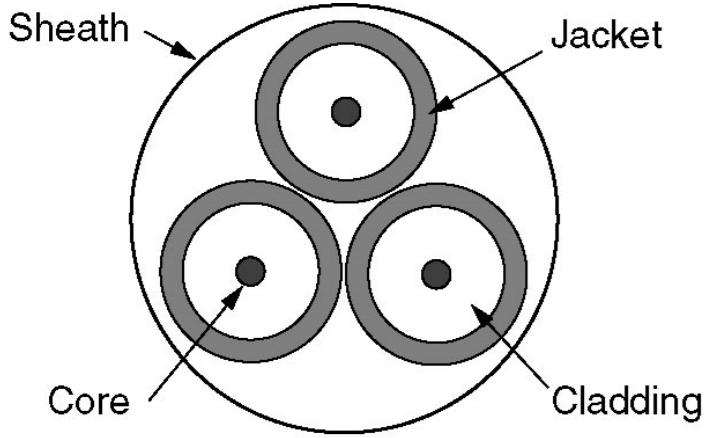


Attenuation of light through fiber in the infrared region.

Fiber Cables



(a)



(b)

(a) Side view of a single fiber.

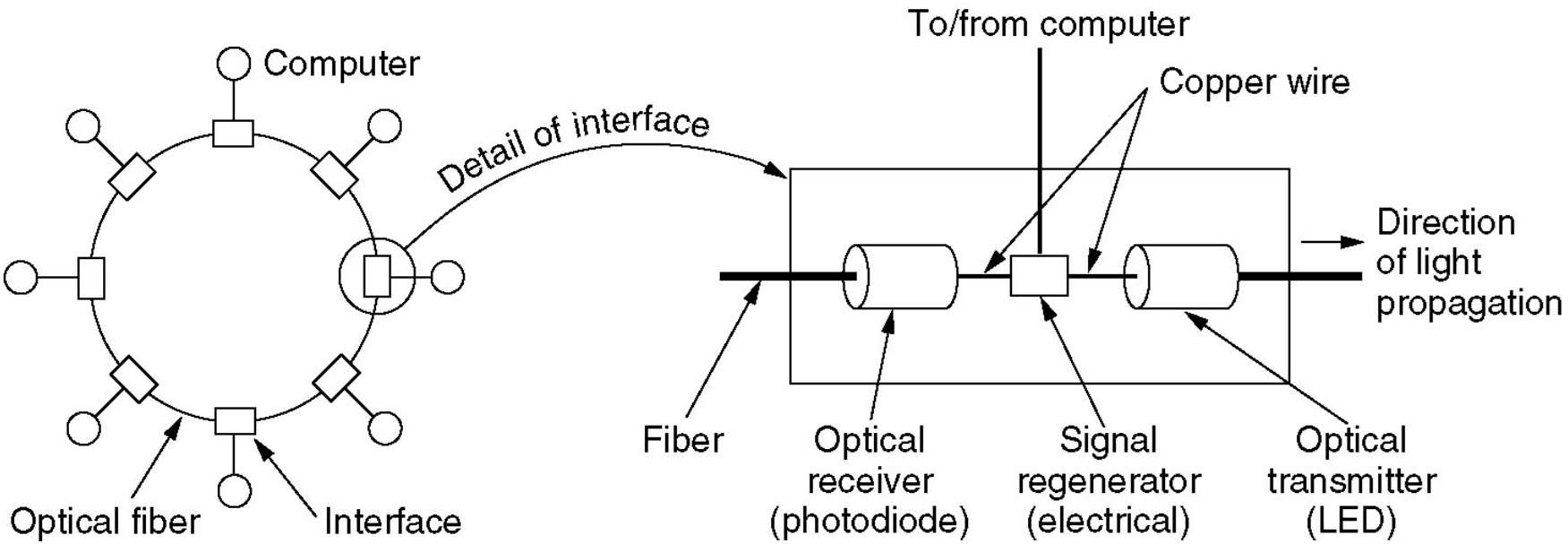
(b) End view of a sheath with three fibers.

Fiber Cables (2)

Item	LED	Semiconductor laser
Data rate	Low	High
Fiber type	Multimode	Multimode or single mode
Distance	Short	Long
Lifetime	Long life	Short life
Temperature sensitivity	Minor	Substantial
Cost	Low cost	Expensive

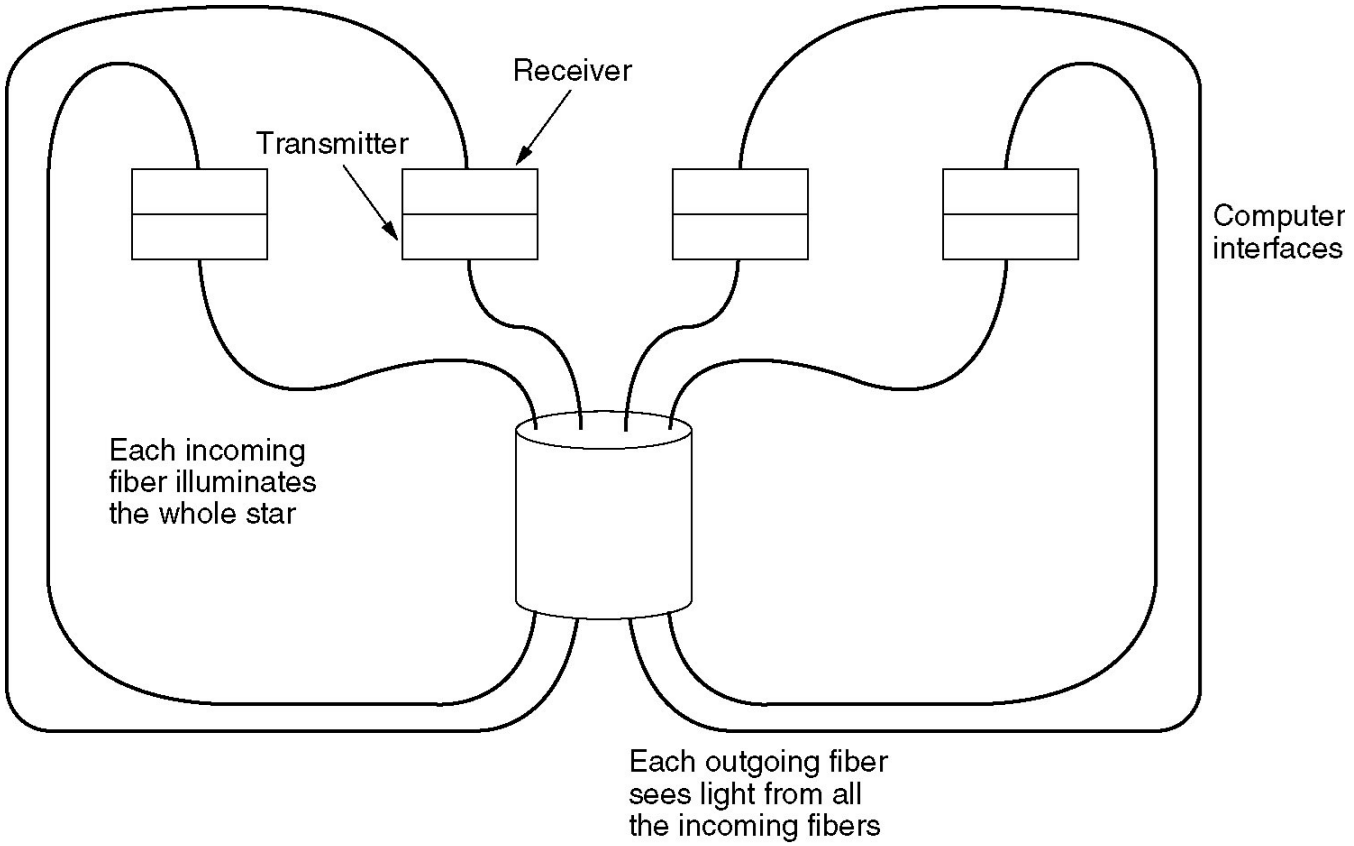
A comparison of semiconductor diodes and LEDs as light sources.

Fiber Optic Networks



A fiber optic ring with active repeaters.

Fiber Optic Networks (2)

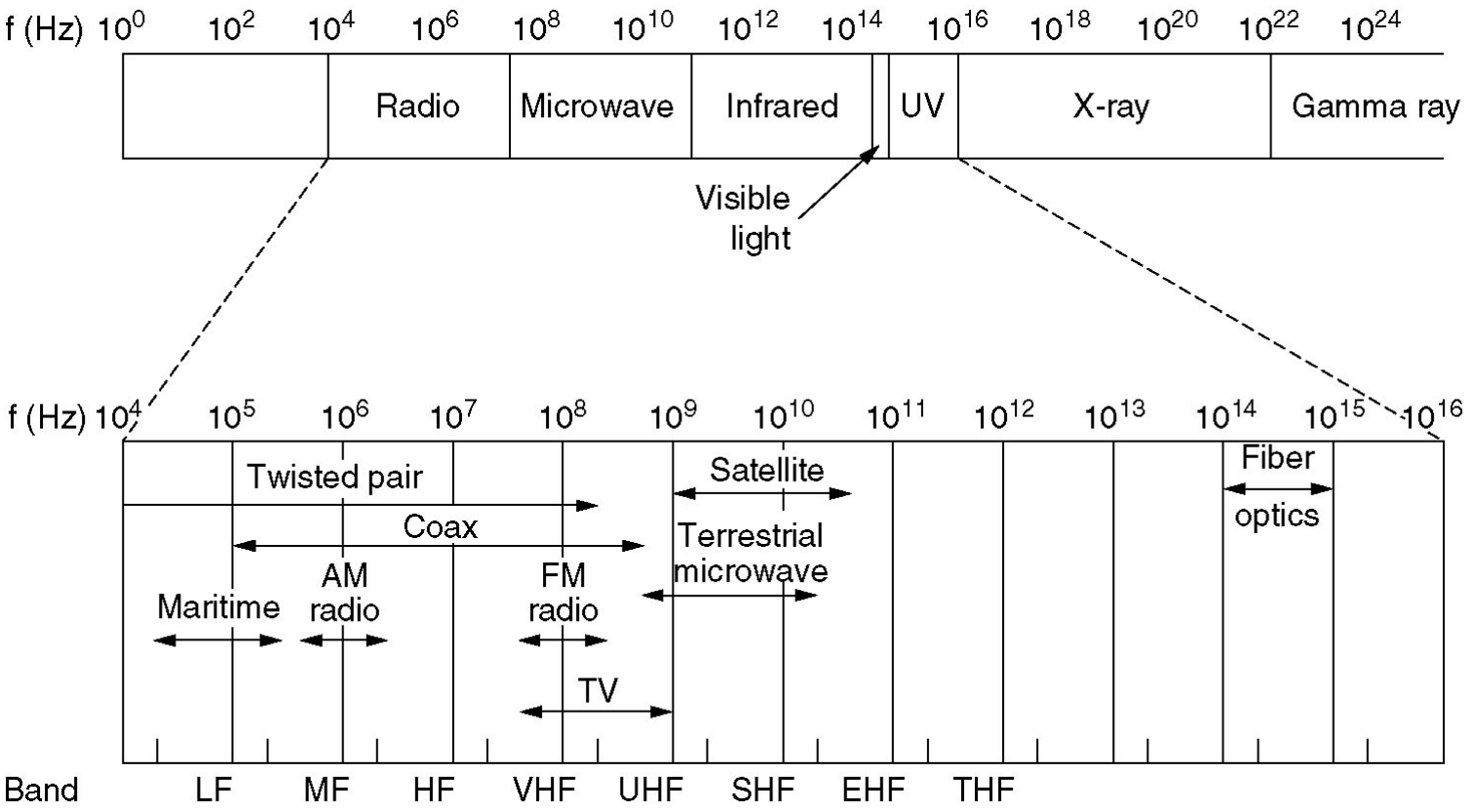


A passive star connection in a fiber optics network.

Wireless Transmission

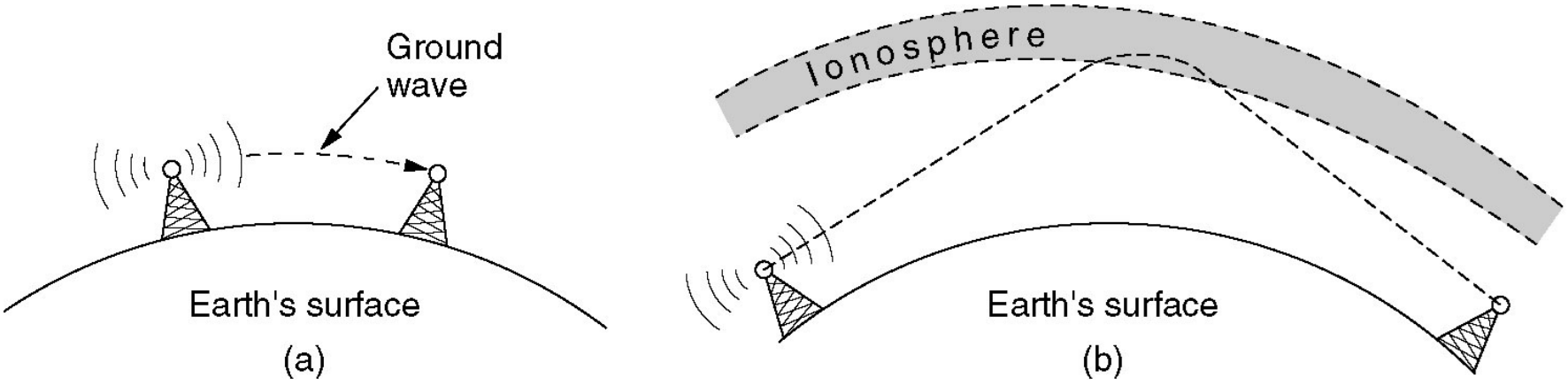
- The Electromagnetic Spectrum
- Radio Transmission
- Microwave Transmission
- Infrared and Millimeter Waves
- Lightwave Transmission

The Electromagnetic Spectrum



The electromagnetic spectrum and its uses for communication.

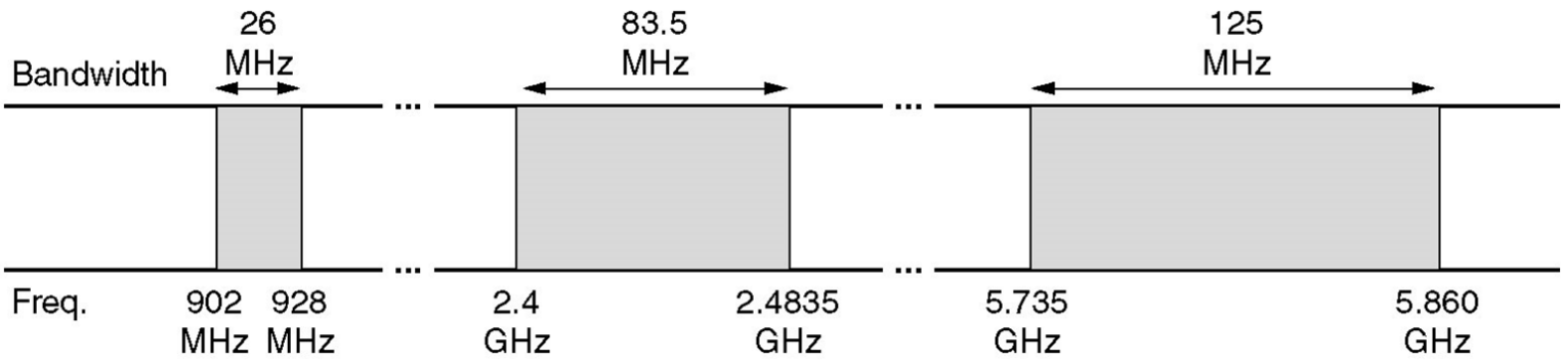
Radio Transmission



(a) In the VLF, LF, and MF bands, radio waves follow the curvature of the earth.

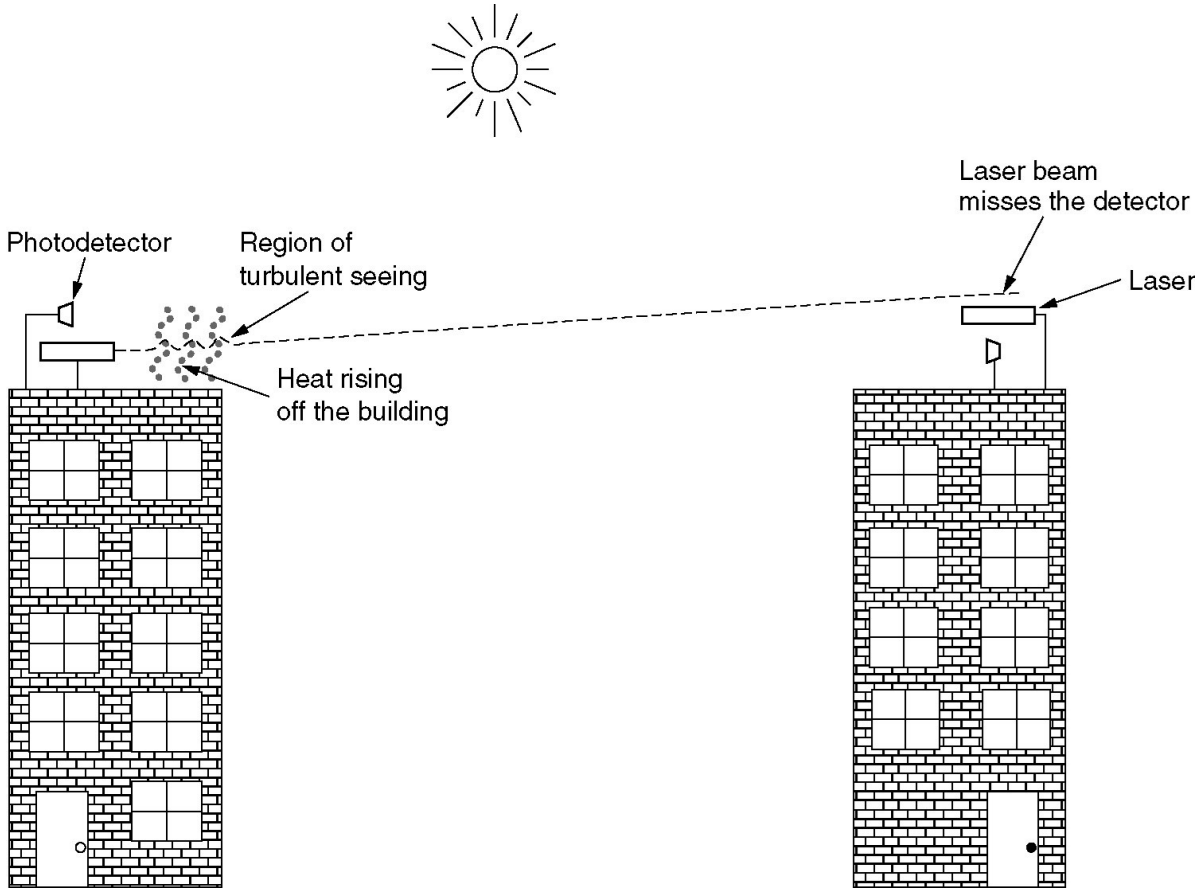
(b) In the HF band, they bounce off the ionosphere.

Politics of the Electromagnetic Spectrum



The ISM bands in the United States.

Lightwave Transmission

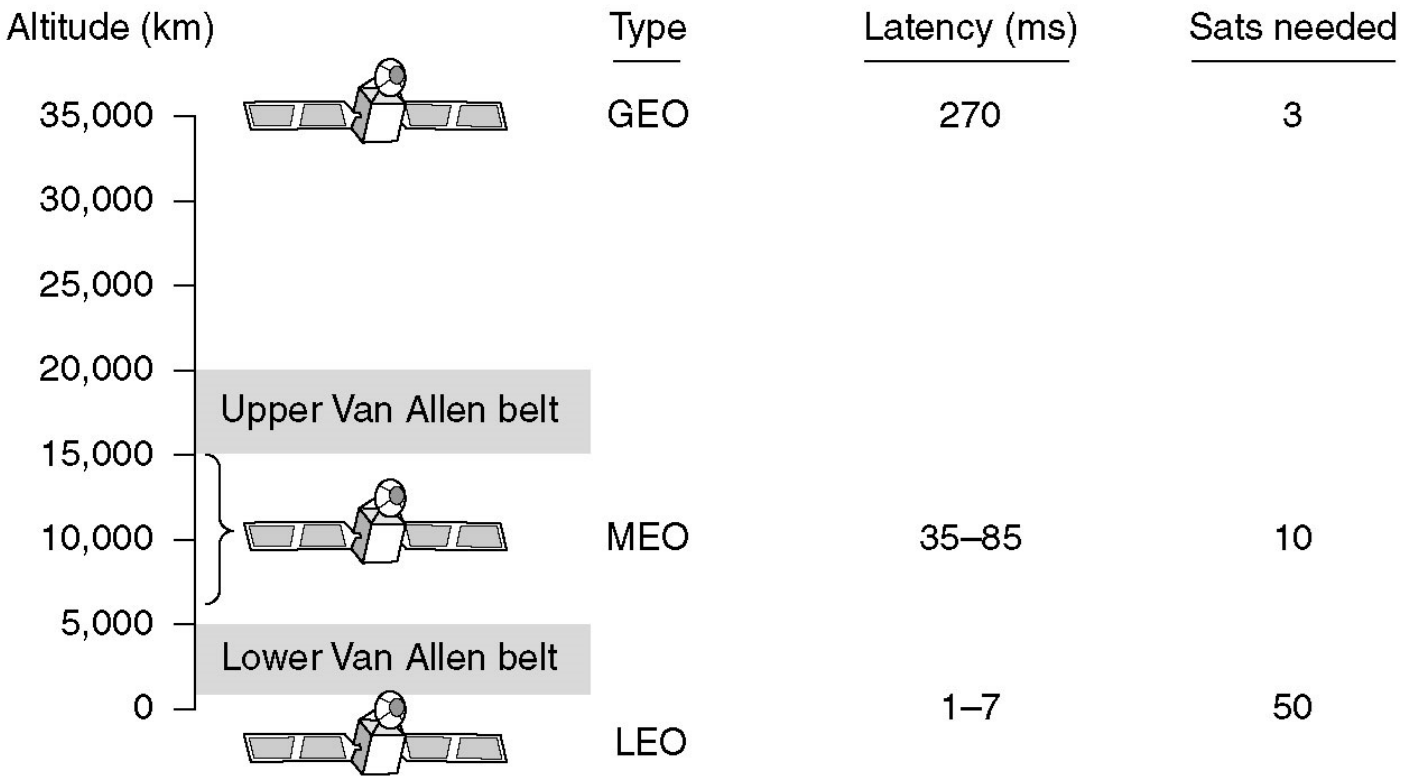


Convection currents can interfere with laser communication systems. A bidirectional system with two lasers is pictured here.

Communication Satellites

- Geostationary Satellites
- Medium-Earth Orbit Satellites
- Low-Earth Orbit Satellites
- Satellites versus Fiber

Communication Satellites



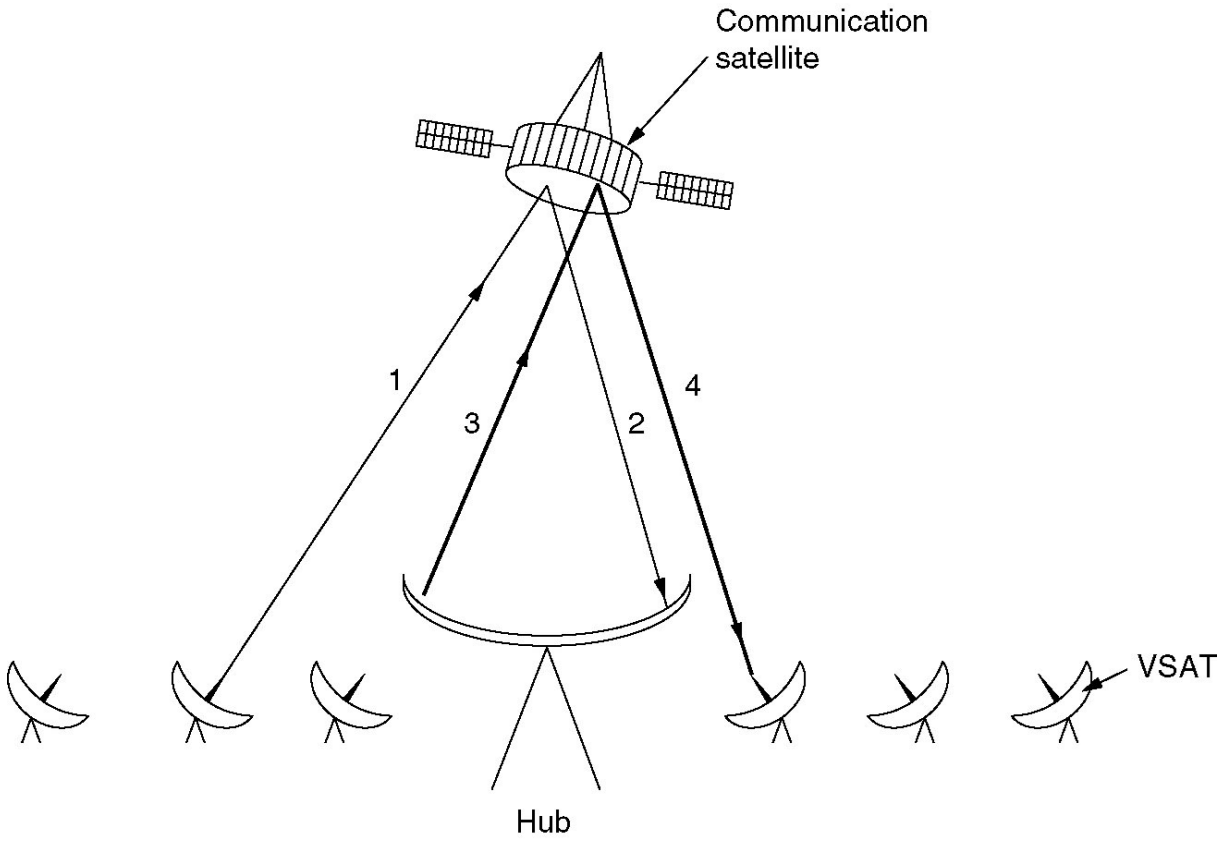
Communication satellites and some of their properties, including altitude above the earth, round-trip delay time and number of satellites needed for global coverage.

Communication Satellites (2)

Band	Downlink	Uplink	Bandwidth	Problems
L	1.5 GHz	1.6 GHz	15 MHz	Low bandwidth; crowded
S	1.9 GHz	2.2 GHz	70 MHz	Low bandwidth; crowded
C	4.0 GHz	6.0 GHz	500 MHz	Terrestrial interference
Ku	11 GHz	14 GHz	500 MHz	Rain
Ka	20 GHz	30 GHz	3500 MHz	Rain, equipment cost

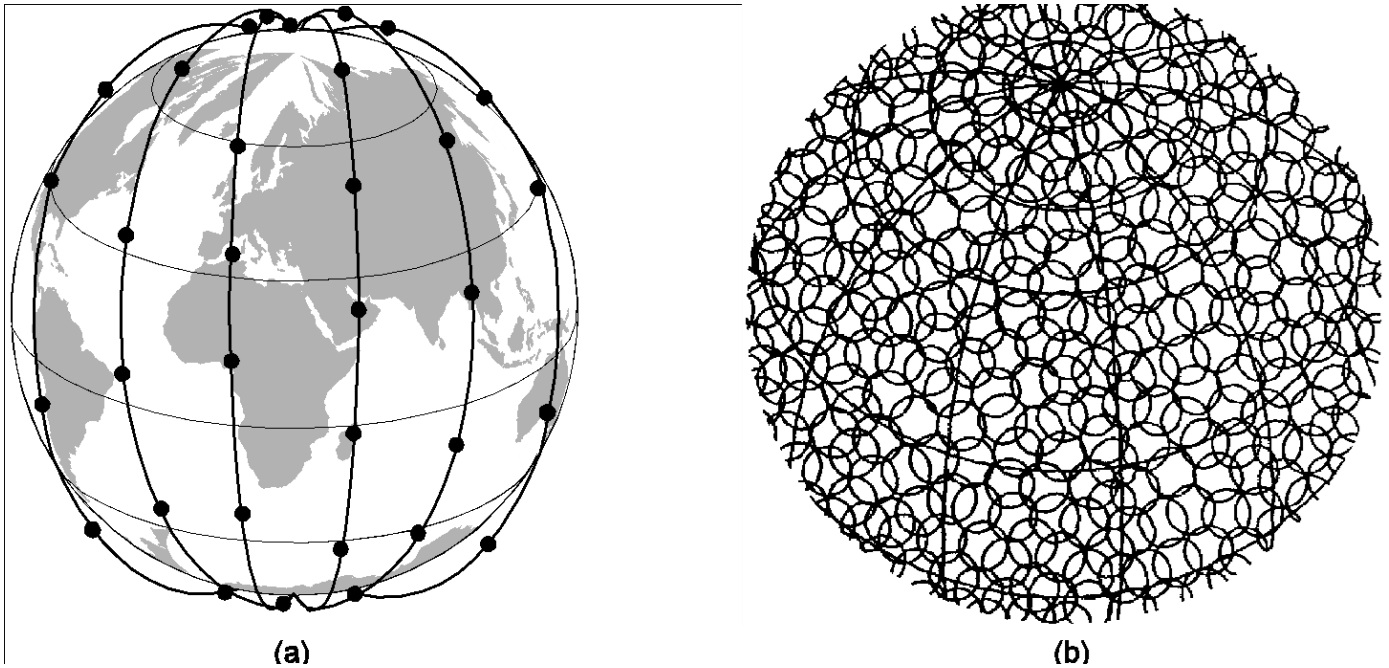
The principal satellite bands.

Communication Satellites (3)



VSATs using a hub.

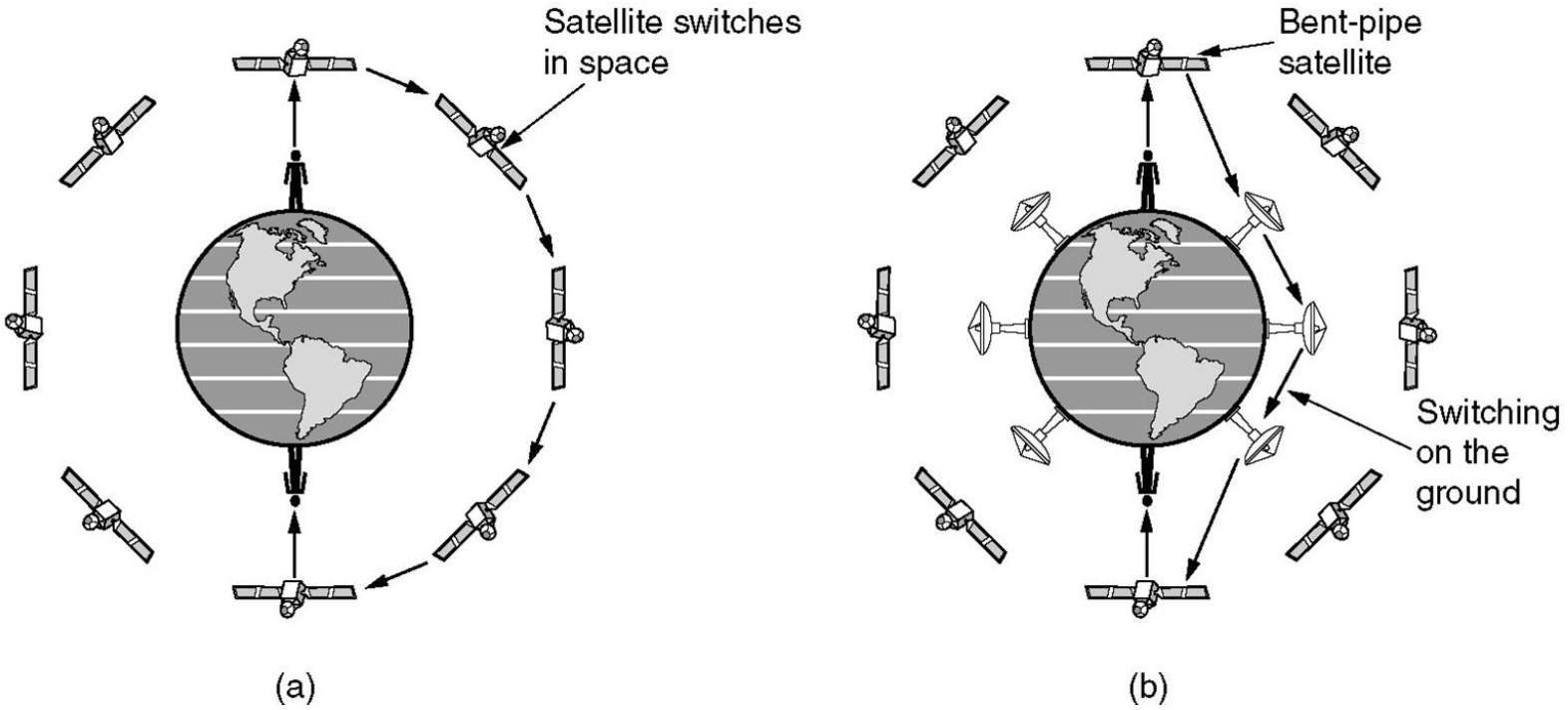
Low-Earth Orbit Satellites
Iridium



(a) The Iridium satellites from six necklaces around the earth.

(b) 1628 moving cells cover the earth.

Globalstar



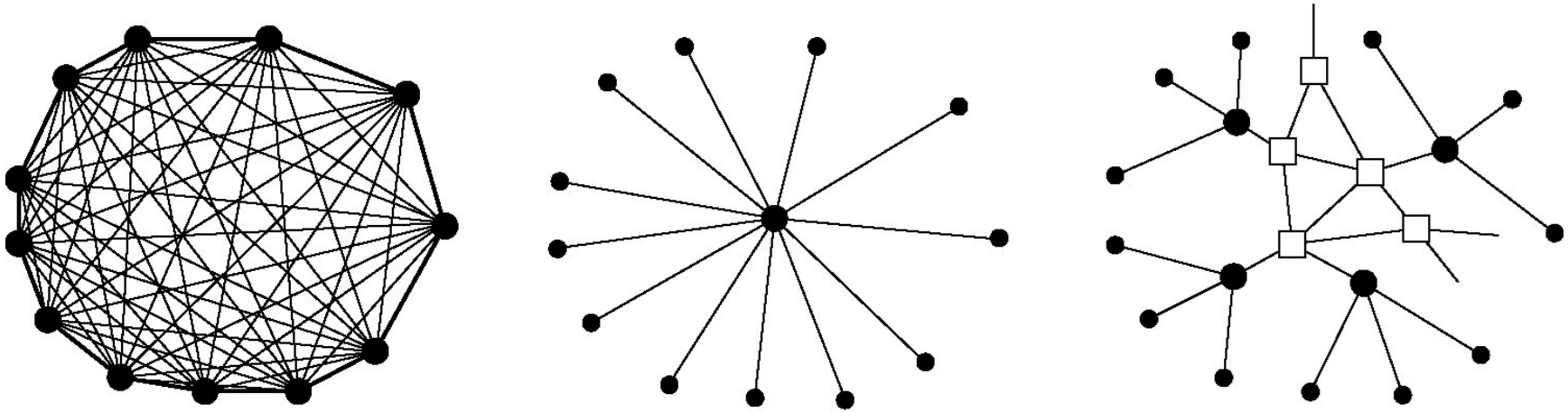
(a) Relaying in space.

(b) Relaying on the ground.

Public Switched Telephone System

- Structure of the Telephone System
- The Politics of Telephones
- The Local Loop: Modems, ADSL and Wireless
- Trunks and Multiplexing
- Switching

Structure of the Telephone System

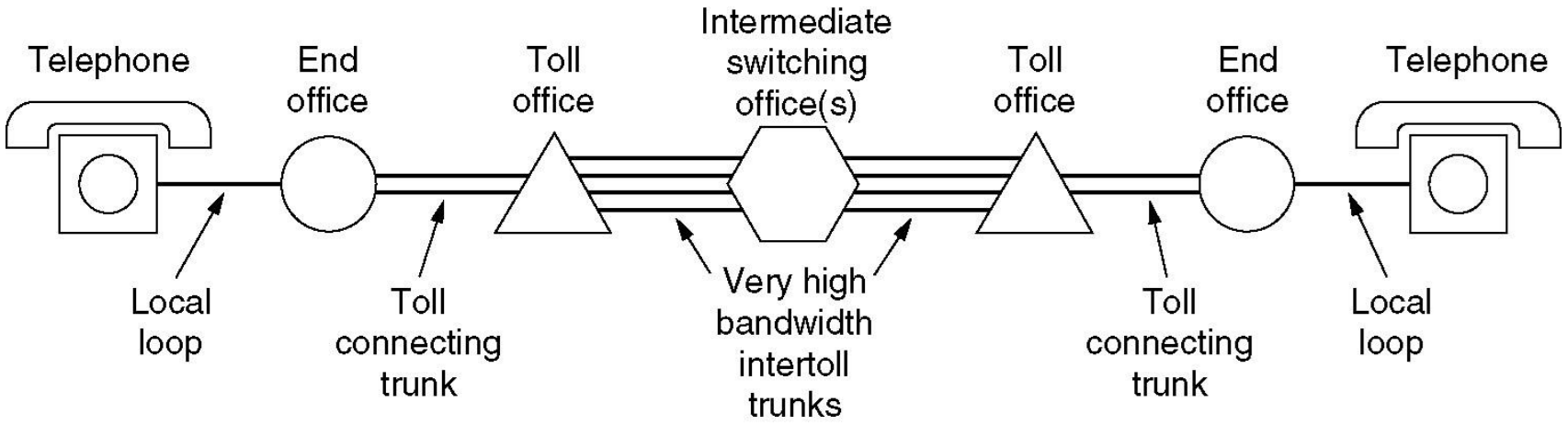


(a) Fully-interconnected network.

(b) Centralized switch.

(c) Two-level hierarchy.

Structure of the Telephone System (2)

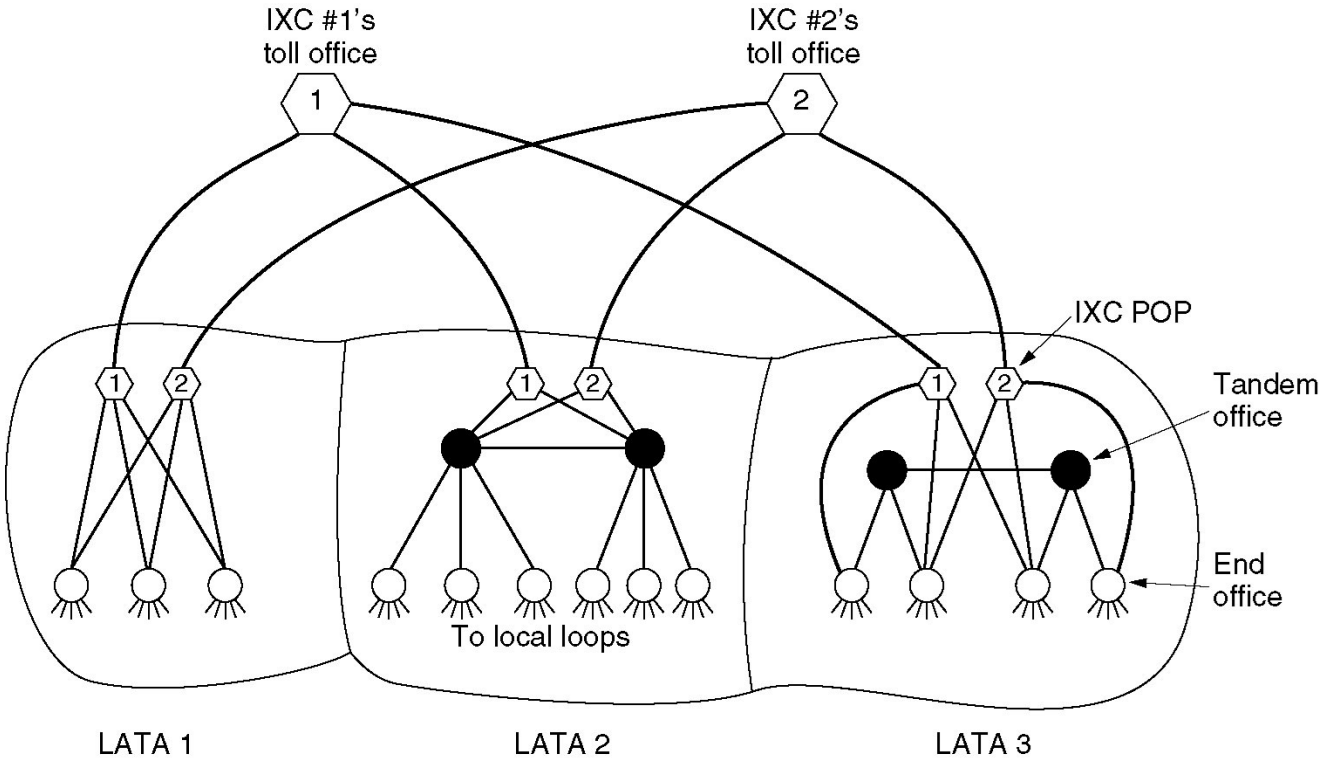


A typical circuit route for a medium-distance call.

Major Components of the Telephone System

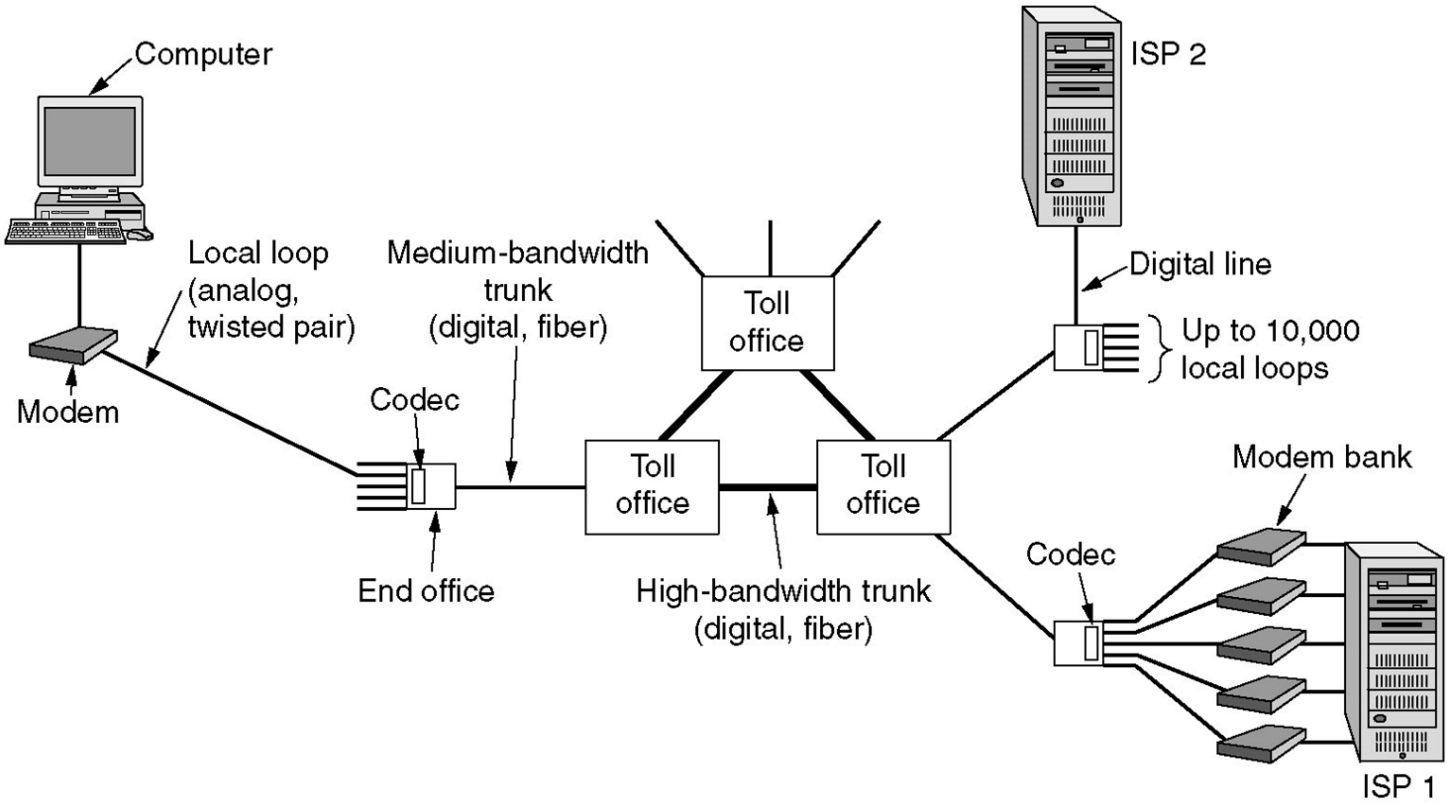
- Local loops
  - Analog twisted pairs going to houses and businesses
- Trunks
  - Digital fiber optics connecting the switching offices
- Switching offices
  - Where calls are moved from one trunk to another

The Politics of Telephones



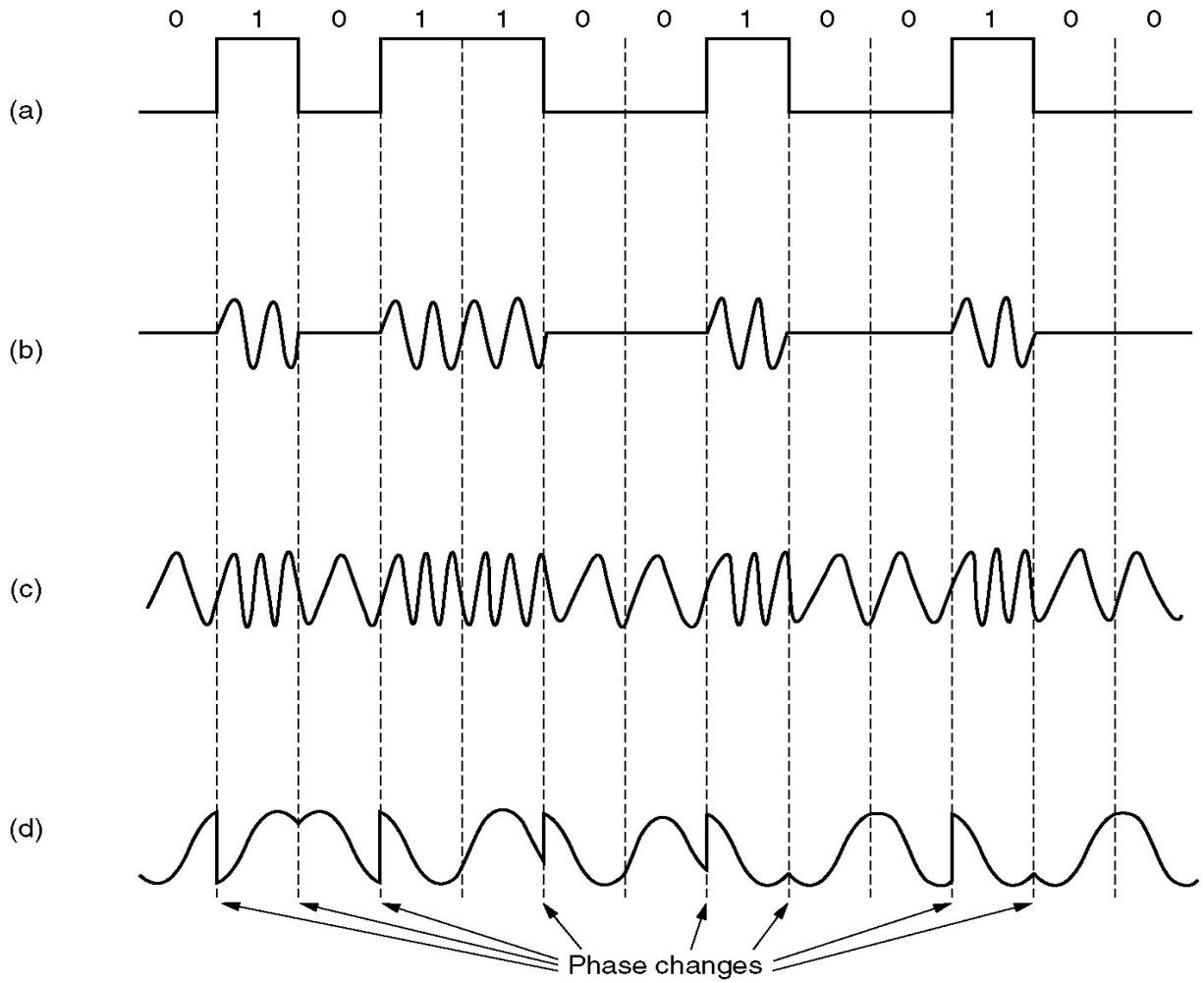
The relationship of LATAs, LECs, and IXCs. All the circles are LEC switching offices. Each hexagon belongs to the IXC whose number is on it.

The Local Loop: Modems, ADSL, and Wireless



The use of both analog and digital transmissions for a computer to computer call. Conversion is done by the modems and codecs.

Modems



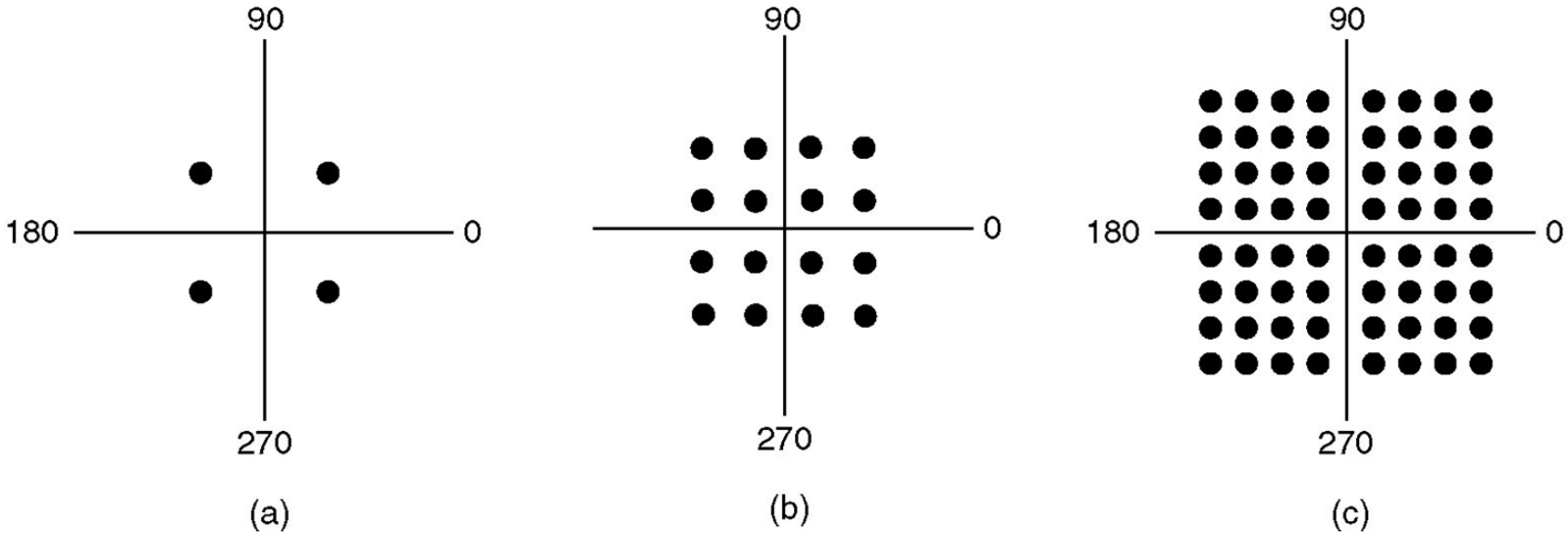
(a) A binary signal

(b) Amplitude modulation

(c) Frequency modulation

(d) Phase modulation

Modems (2)

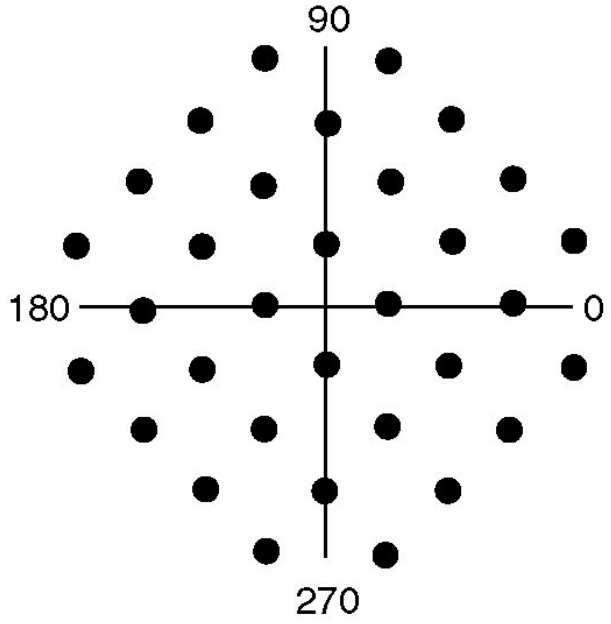


(a) QPSK.

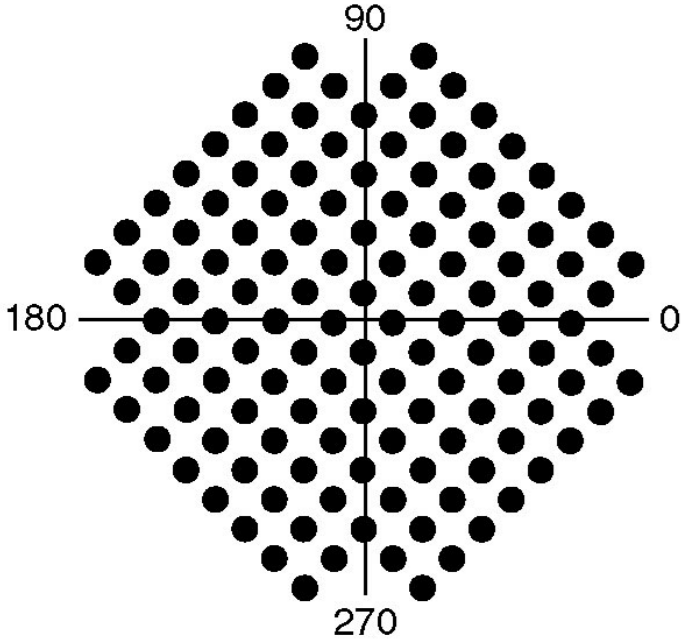
(b) QAM-16.

(c) QAM-64.

Modems (3)



(a)

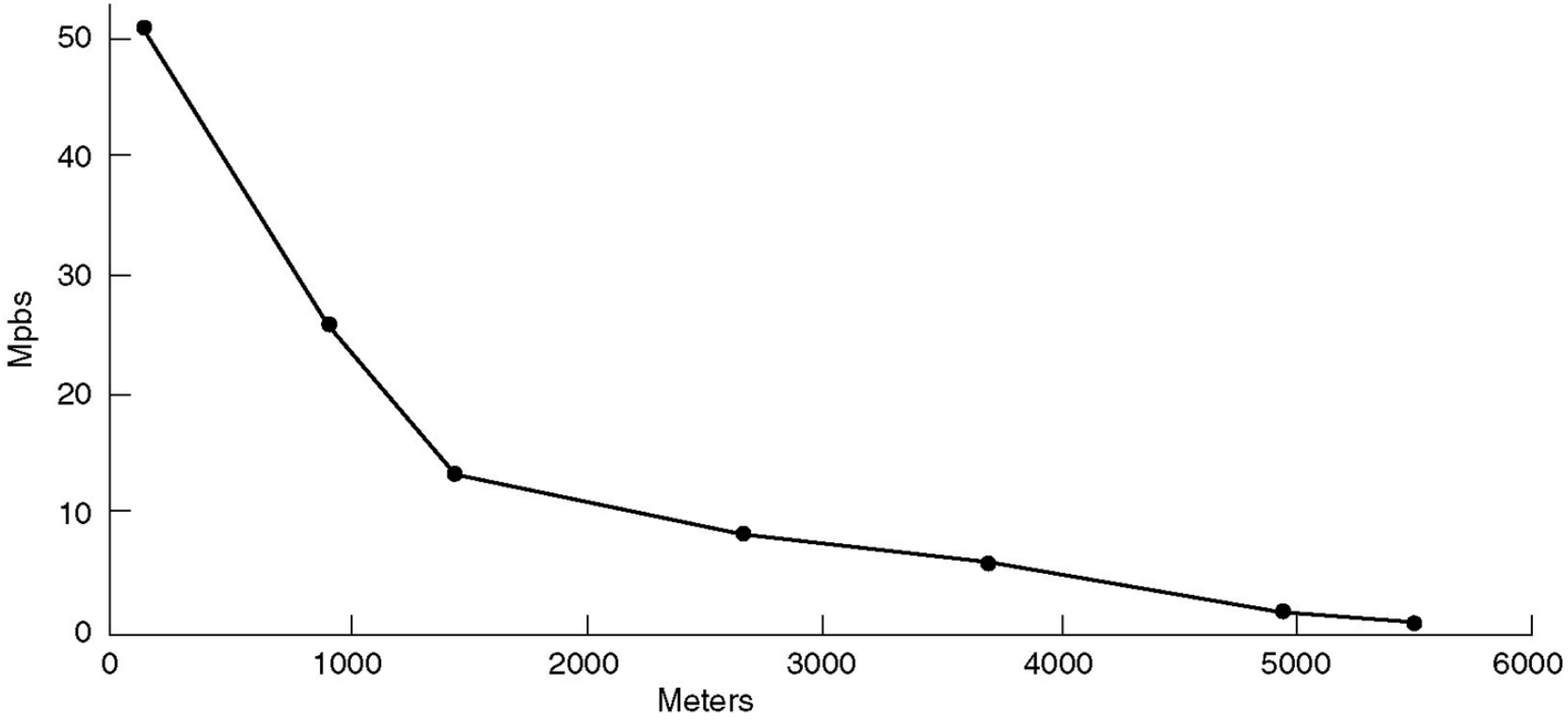


(b)

(a) V.32 for 9600 bps.

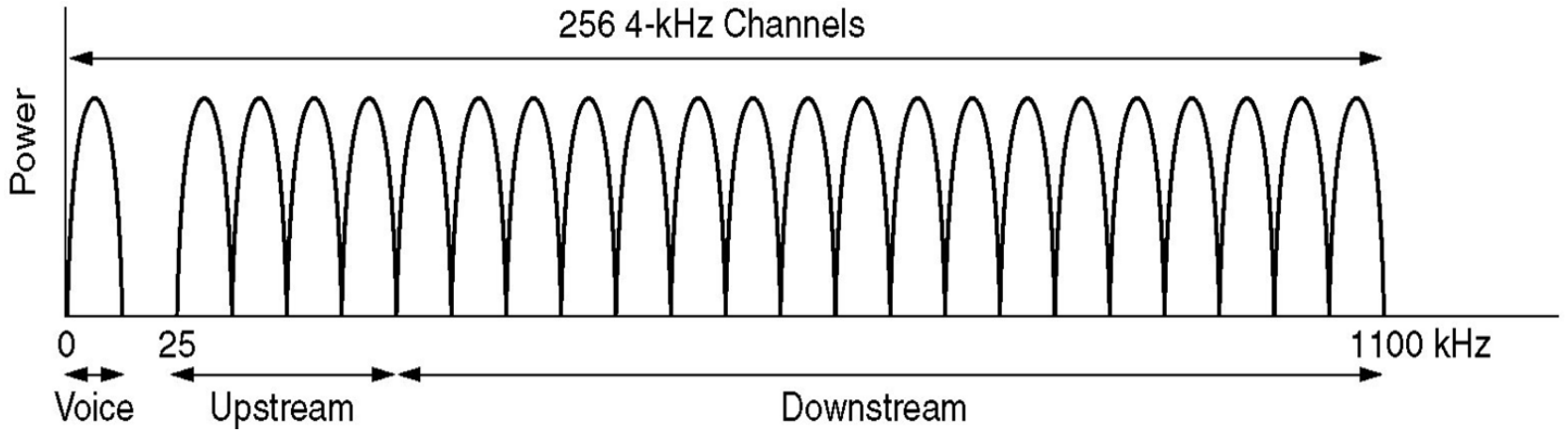
(b) V32 bis for 14,400 bps.

Digital Subscriber Lines



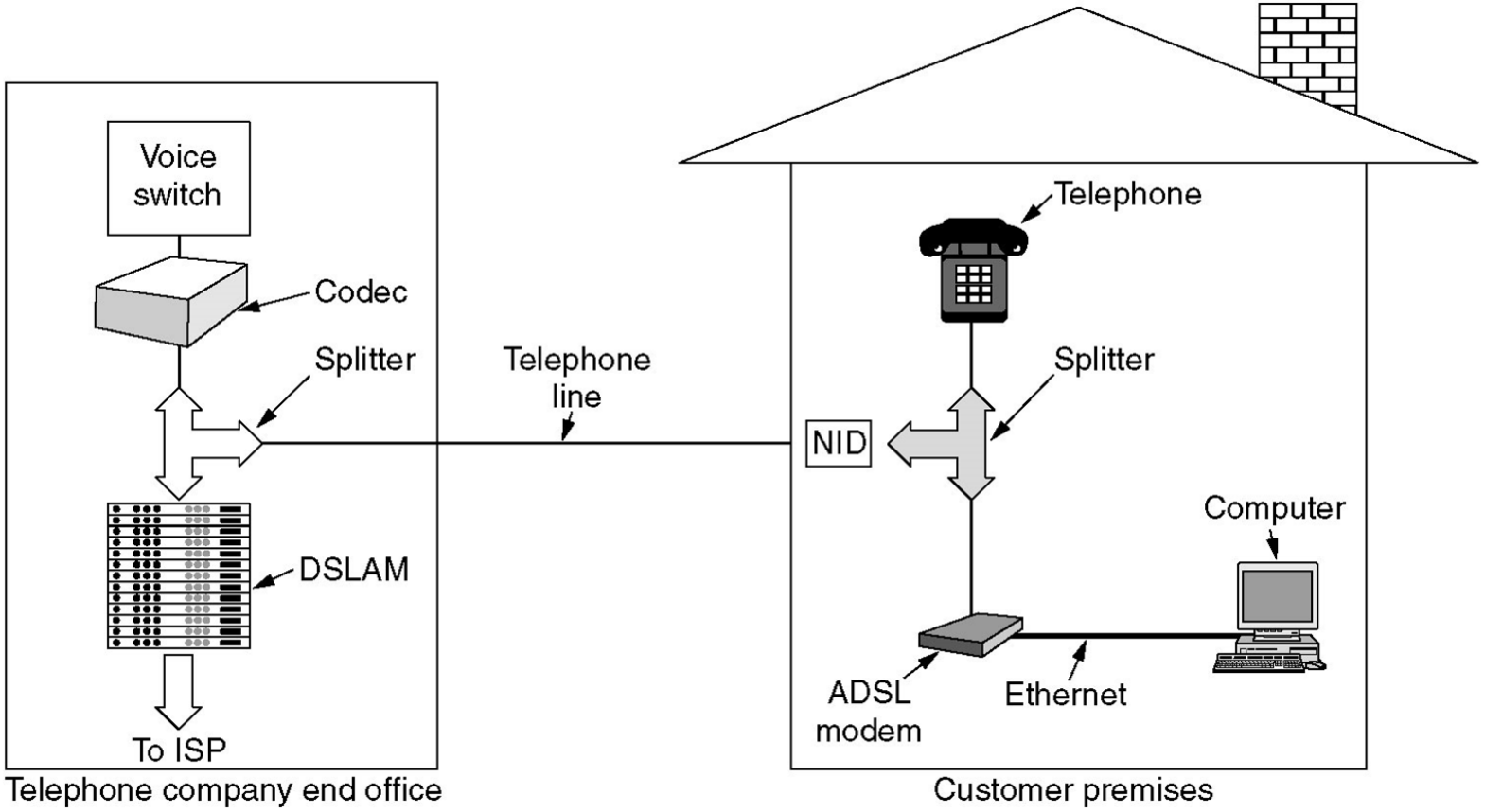
Bandwidth versus distanced over category 3 UTP for DSL.

Digital Subscriber Lines (2)



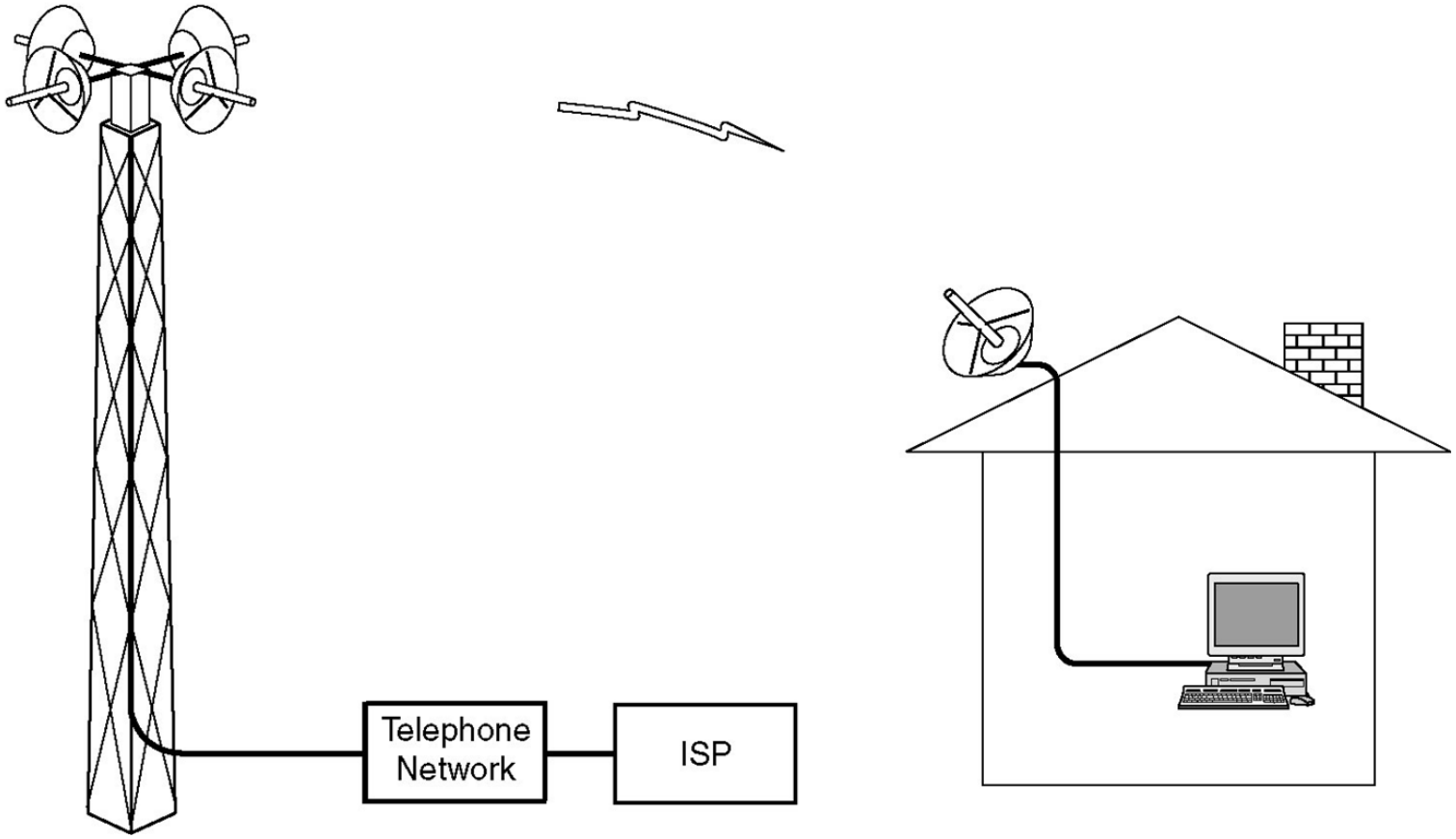
Operation of ADSL using discrete multitone modulation.

Digital Subscriber Lines (3)



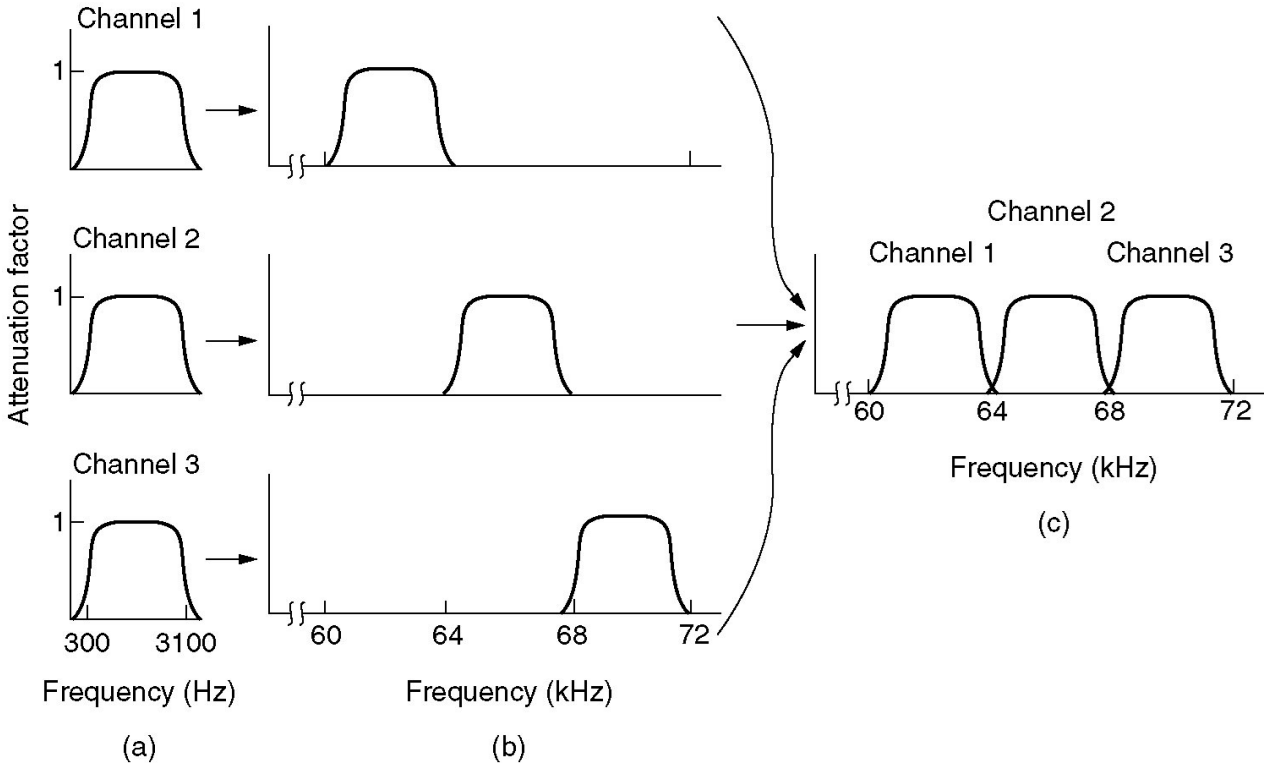
A typical ADSL equipment configuration.

Wireless Local Loops



Architecture of an LMDS system.

Frequency Division Multiplexing

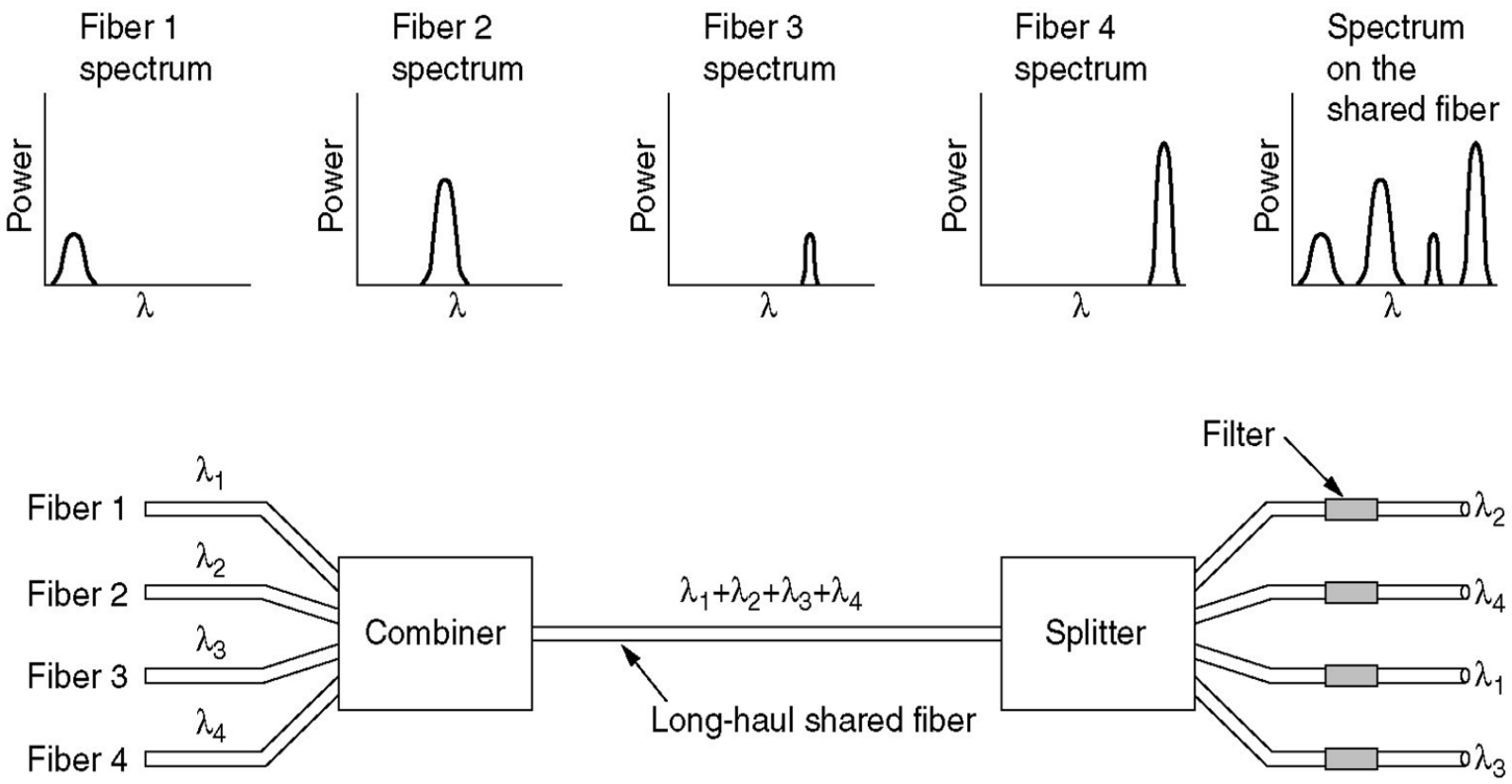


(a) The original bandwidths.

(b) The bandwidths raised in frequency.

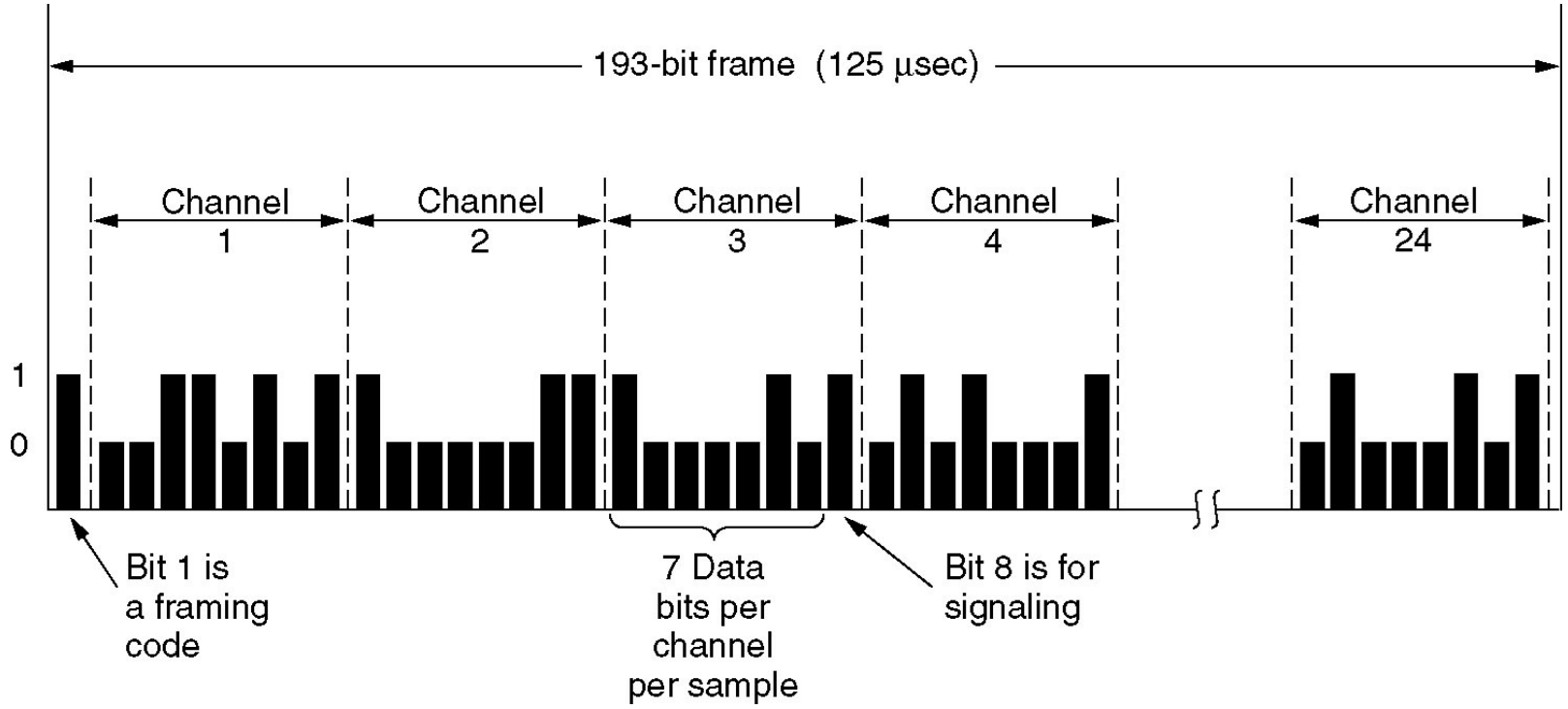
(b) The multiplexed channel.

Wavelength Division Multiplexing



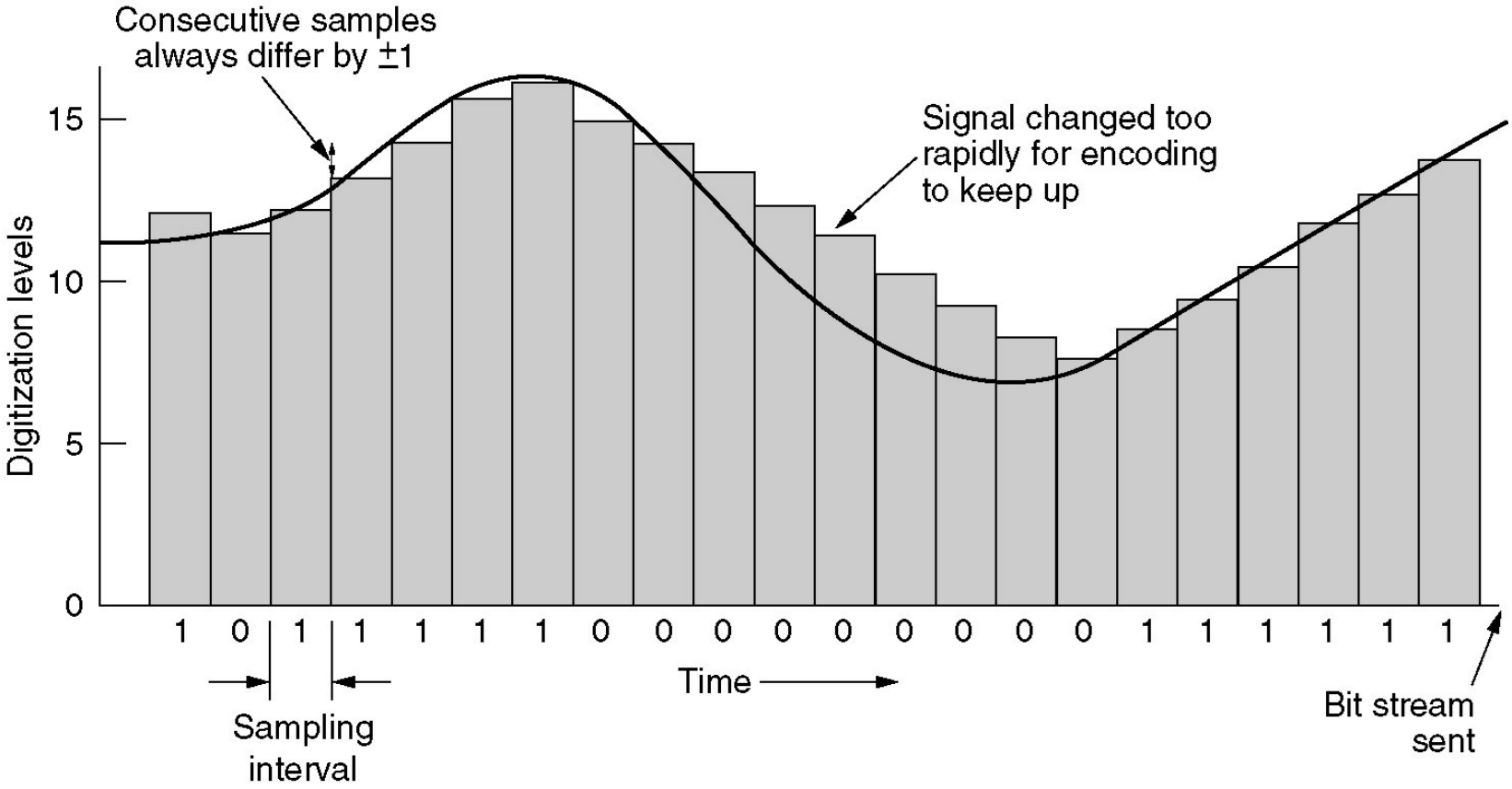
Wavelength division multiplexing.

Time Division Multiplexing



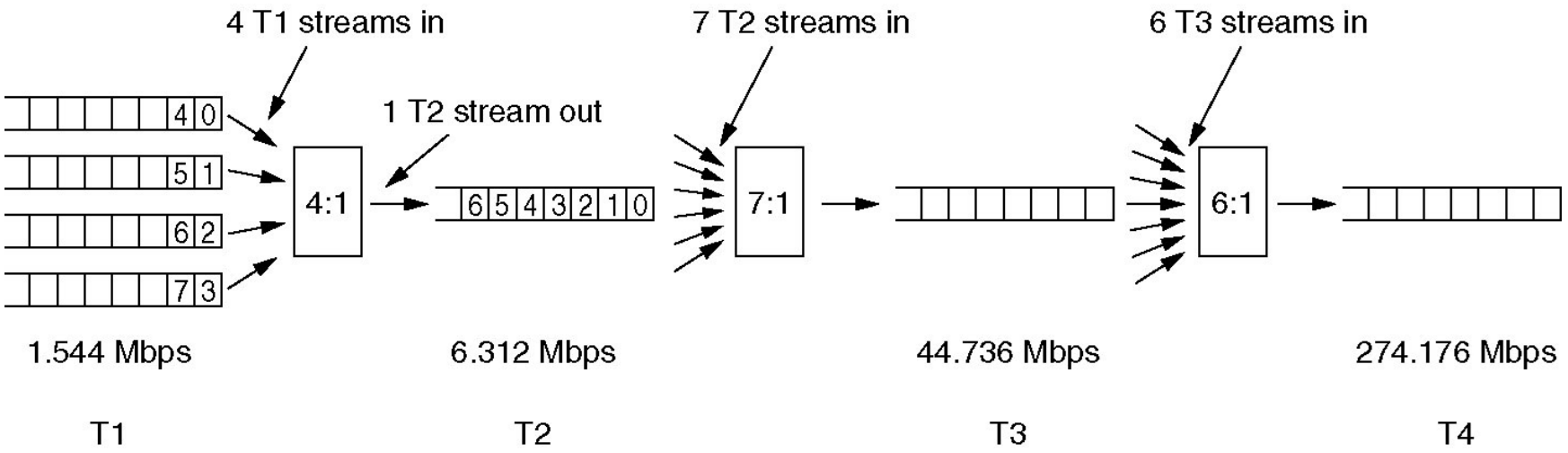
The T1 carrier (1.544 Mbps).

Time Division Multiplexing (2)



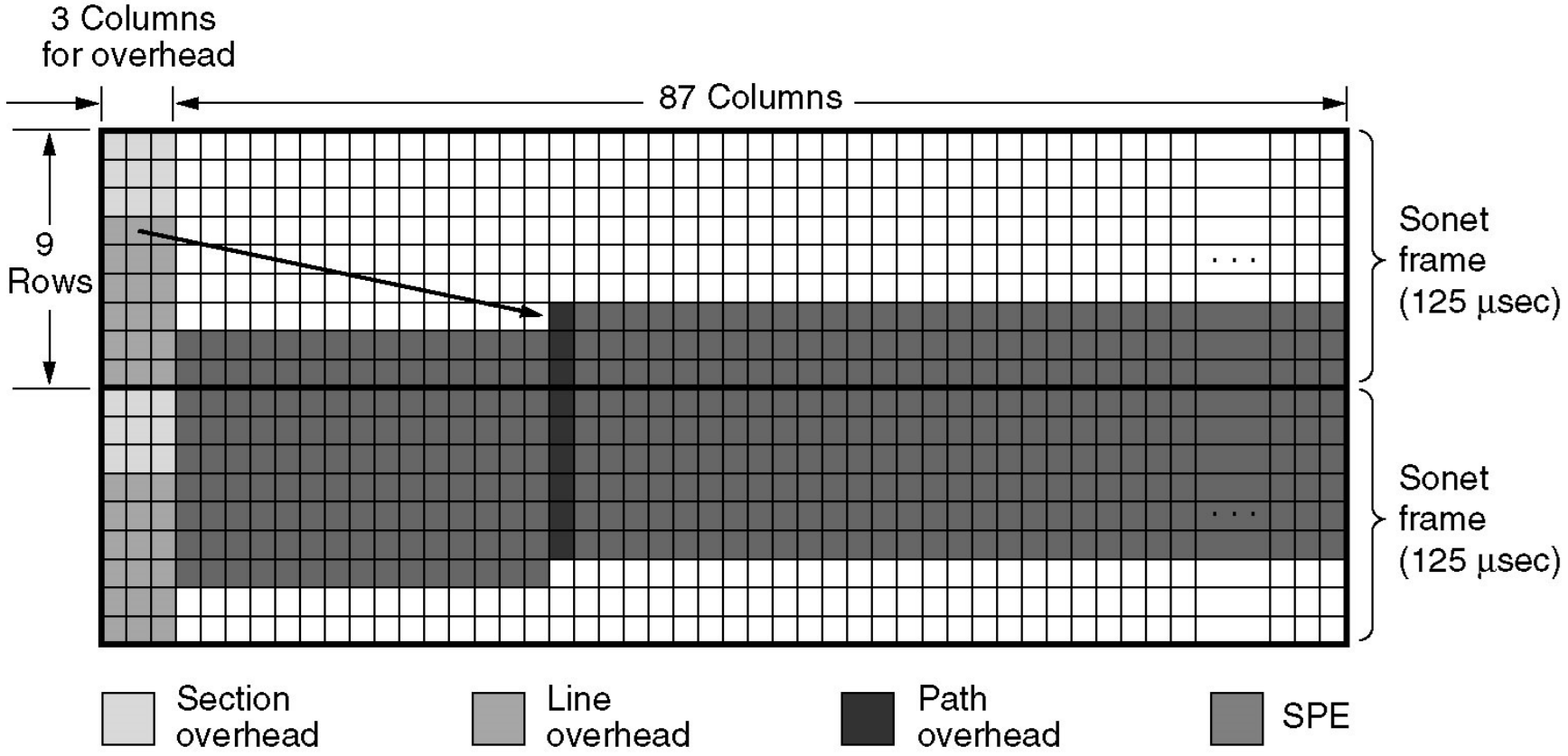
Delta modulation.

Time Division Multiplexing (3)



Multiplexing T1 streams into higher carriers.

Time Division Multiplexing (4)



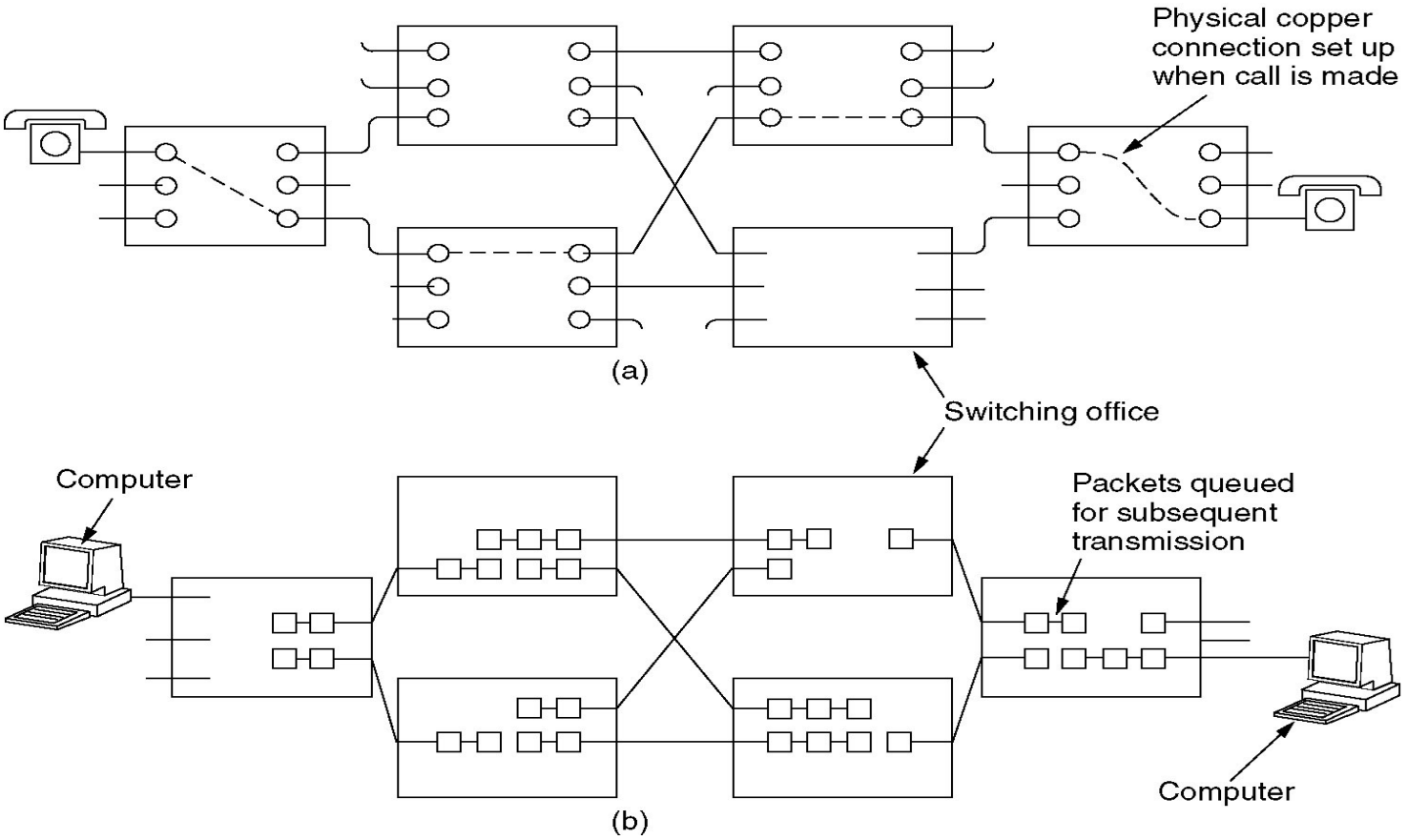
Two back-to-back SONET frames.

Time Division Multiplexing (5)

SONET		SDH	Data rate (Mbps)		
Electrical	Optical	Optical	Gross	SPE	User
STS-1	OC-1		51.84	50.112	49.536
STS-3	OC-3	STM-1	155.52	150.336	148.608
STS-9	OC-9	STM-3	466.56	451.008	445.824
STS-12	OC-12	STM-4	622.08	601.344	594.432
STS-18	OC-18	STM-6	933.12	902.016	891.648
STS-24	OC-24	STM-8	1244.16	1202.688	1188.864
STS-36	OC-36	STM-12	1866.24	1804.032	1783.296
STS-48	OC-48	STM-16	2488.32	2405.376	2377.728
STS-192	OC-192	STM-64	9953.28	9621.504	9510.912

SONET and SDH multiplex rates.

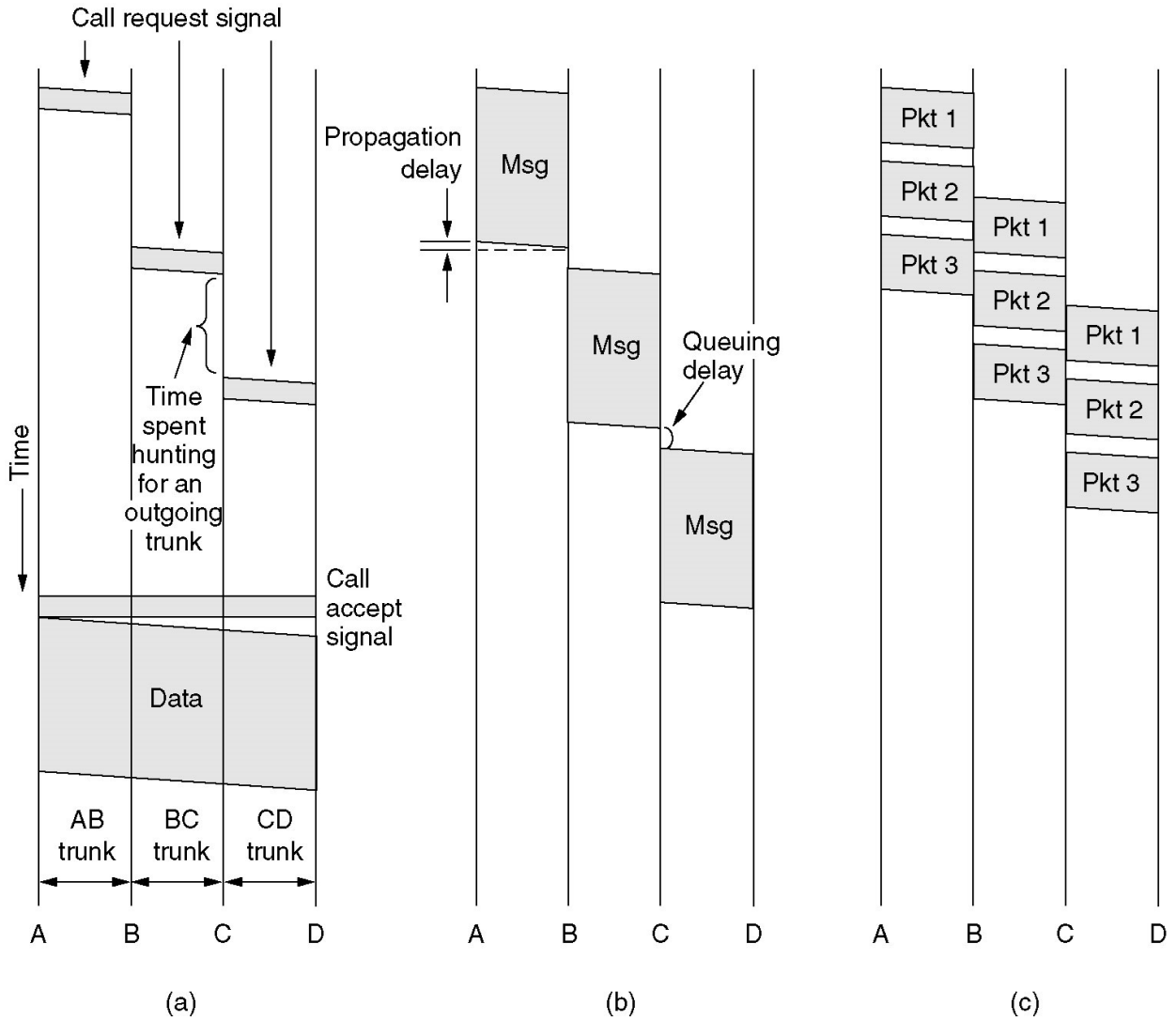
Circuit Switching



(a) Circuit switching.

(b) Packet switching.

Message Switching



(a) Circuit switching (b) Message switching (c) Packet switching

Packet Switching

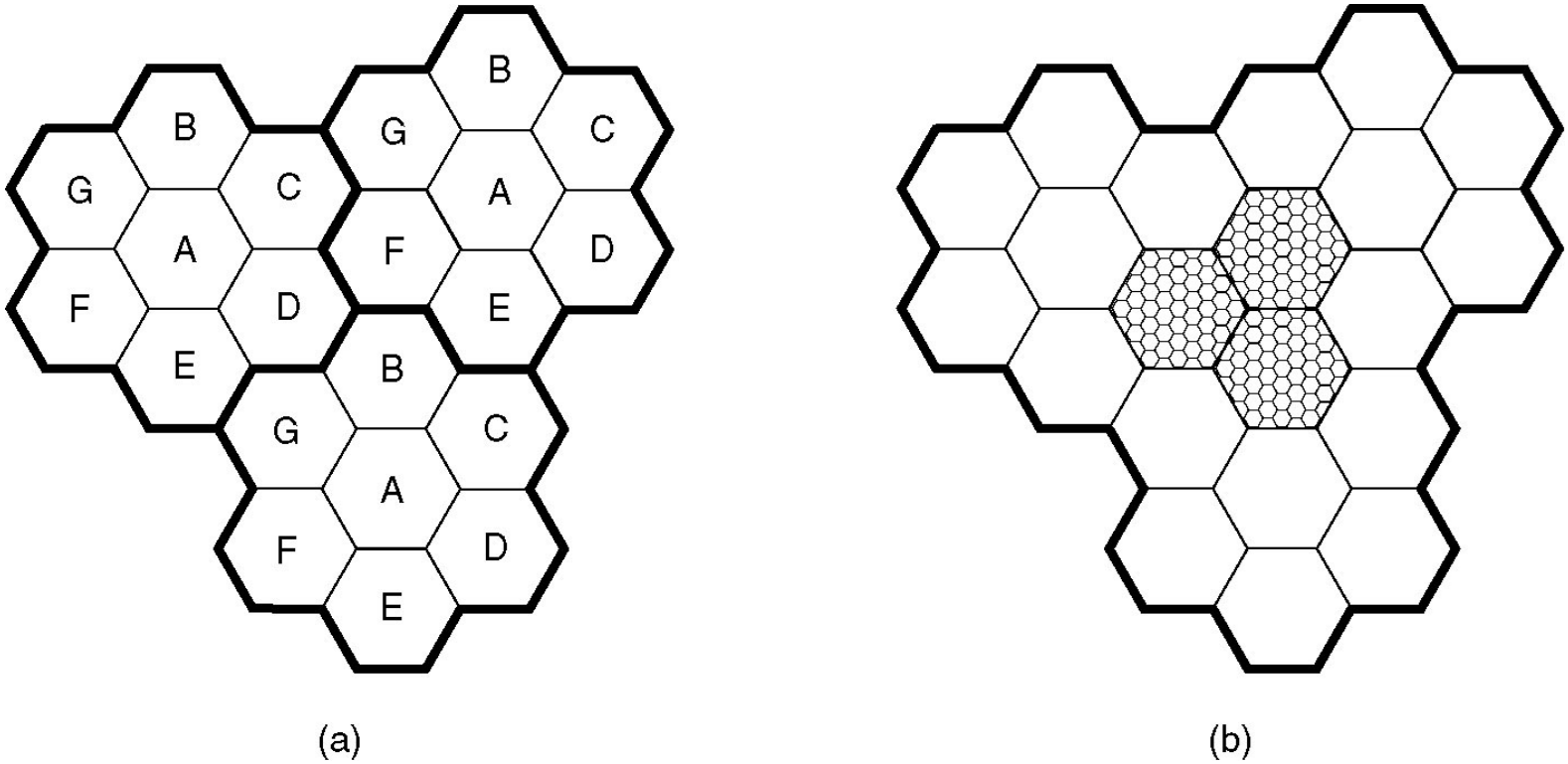
Item	Circuit-switched	Packet-switched
Call setup	Required	Not needed
Dedicated physical path	Yes	No
Each packet follows the same route	Yes	No
Packets arrive in order	Yes	No
Is a switch crash fatal	Yes	No
Bandwidth available	Fixed	Dynamic
When can congestion occur	At setup time	On every packet
Potentially wasted bandwidth	Yes	No
Store-and-forward transmission	No	Yes
Transparency	Yes	No
Charging	Per minute	Per packet

A comparison of circuit switched and packet-switched networks.

The Mobile Telephone System

- First-Generation Mobile Phones: Analog Voice
- Second-Generation Mobile Phones: Digital Voice
- Third-Generation Mobile Phones: Digital Voice and Data

Advanced Mobile Phone System



(a) Frequencies are not reused in adjacent cells.

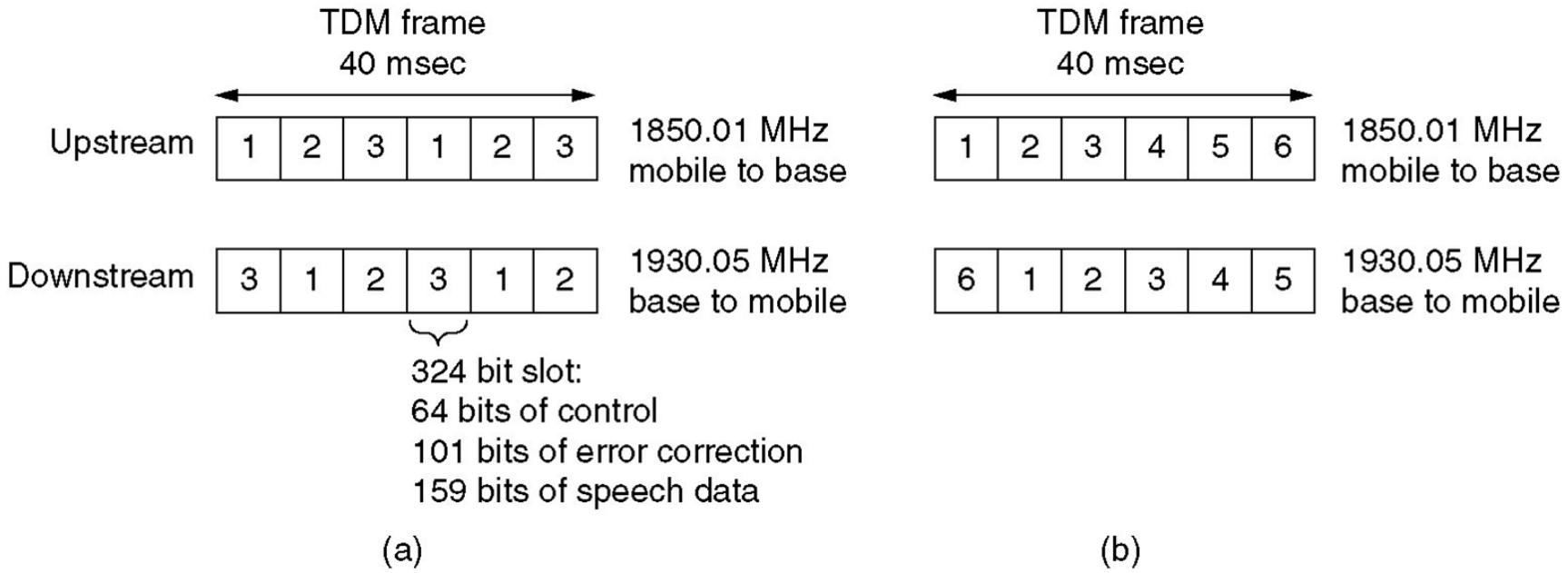
(b) To add more users, smaller cells can be used.

Channel Categories

The 832 channels are divided into four categories:

- Control (base to mobile) to manage the system
- Paging (base to mobile) to alert users to calls for them
- Access (bidirectional) for call setup and channel assignment
- Data (bidirectional) for voice, fax, or data

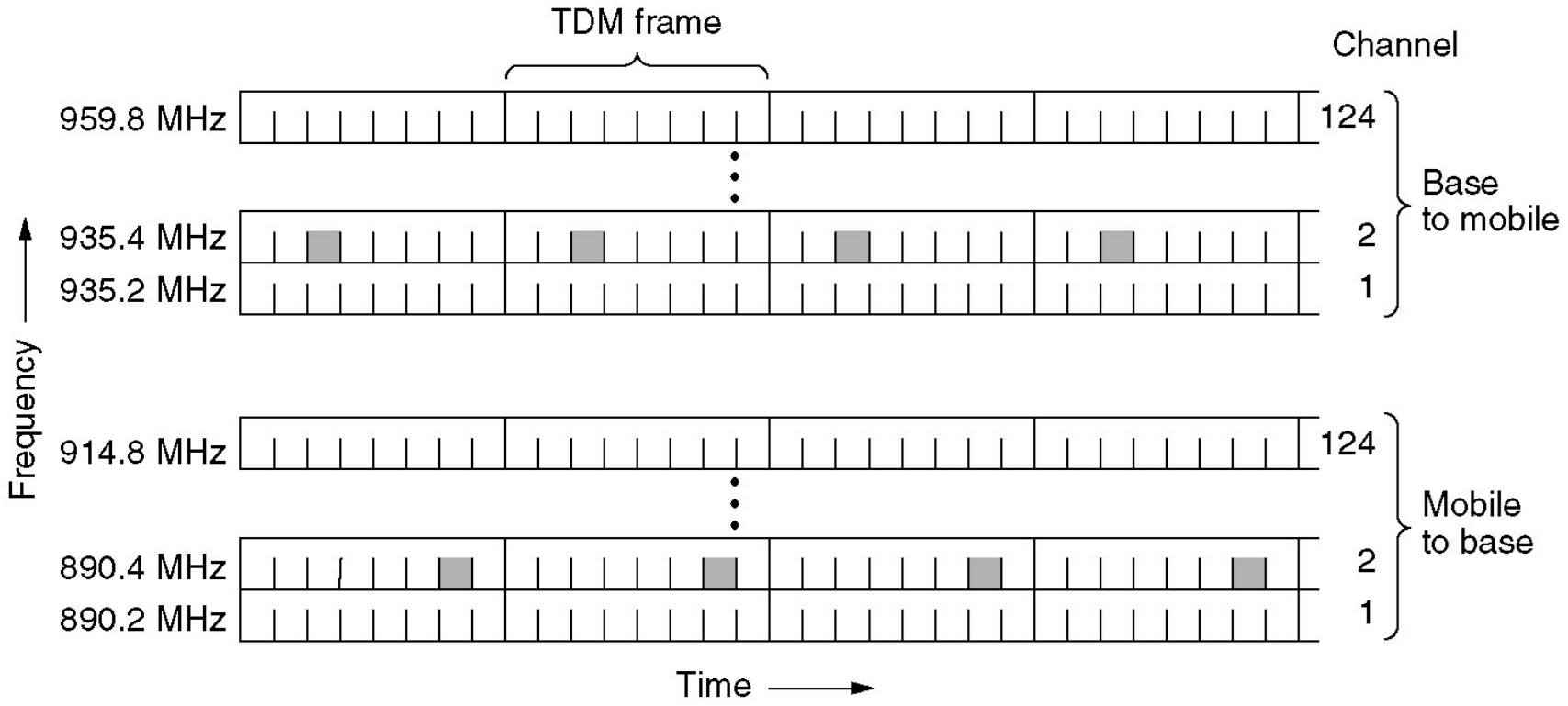
D-AMPS
Digital Advanced Mobile Phone System



(a) A D-AMPS channel with three users.

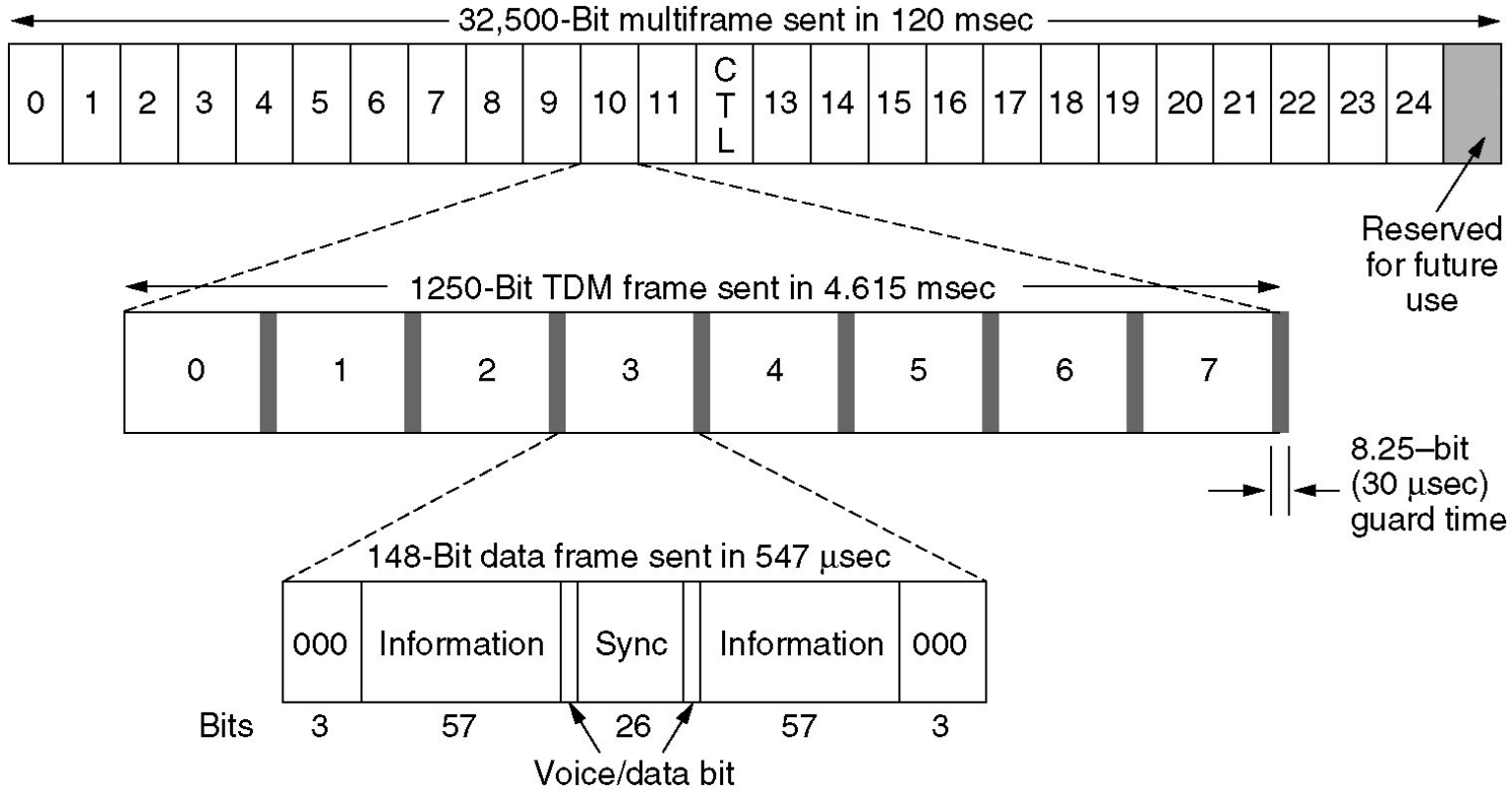
(b) A D-AMPS channel with six users.

GSM
Global System for Mobile Communications



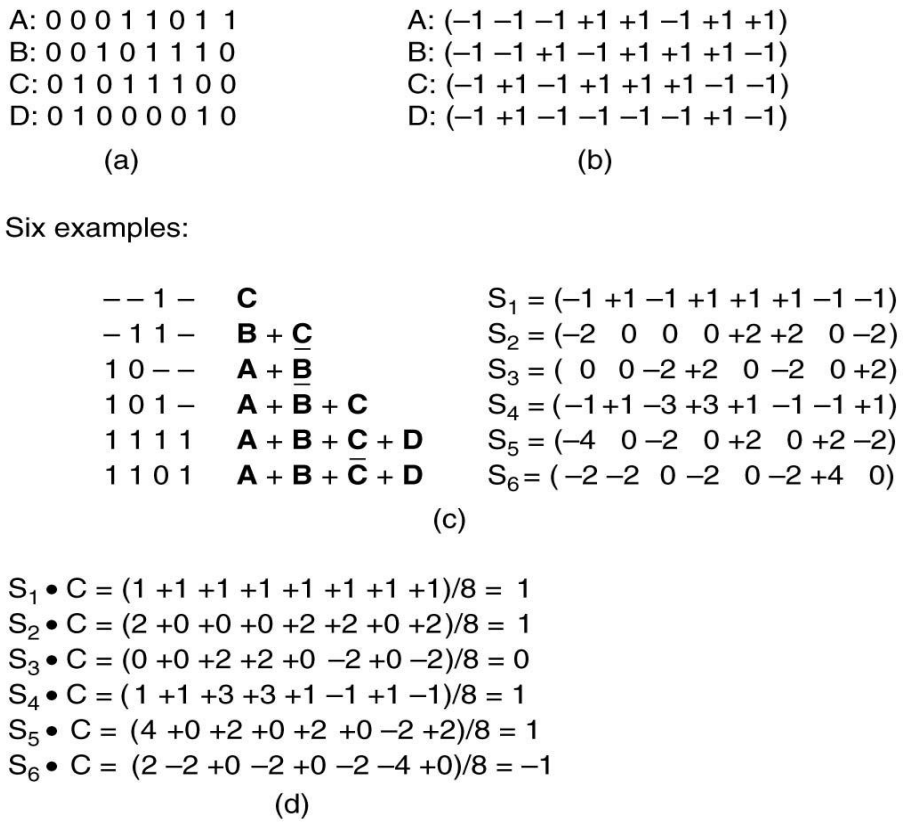
GSM uses 124 frequency channels, each of which uses an eight-slot TDM system

GSM (2)



A portion of the GSM framing structure.

CDMA – Code Division Multiple Access



(a) Binary chip sequences for four stations
(b) Bipolar chip sequences
(c) Six examples of transmissions
(d) Recovery of station C's signal

Third-Generation Mobile Phones: Digital Voice and Data

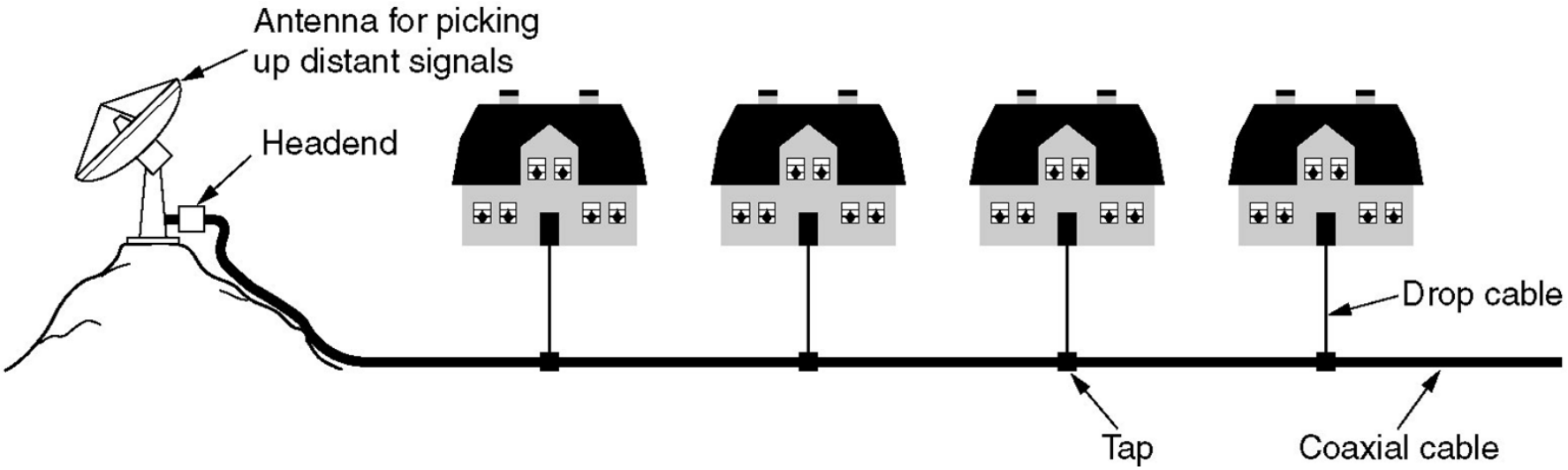
Basic services an IMT-2000 network should provide

- High-quality voice transmission
- Messaging (replace e-mail, fax, SMS, chat, etc.)
- Multimedia (music, videos, films, TV, etc.)
- Internet access (web surfing, w/multimedia.)

Cable Television

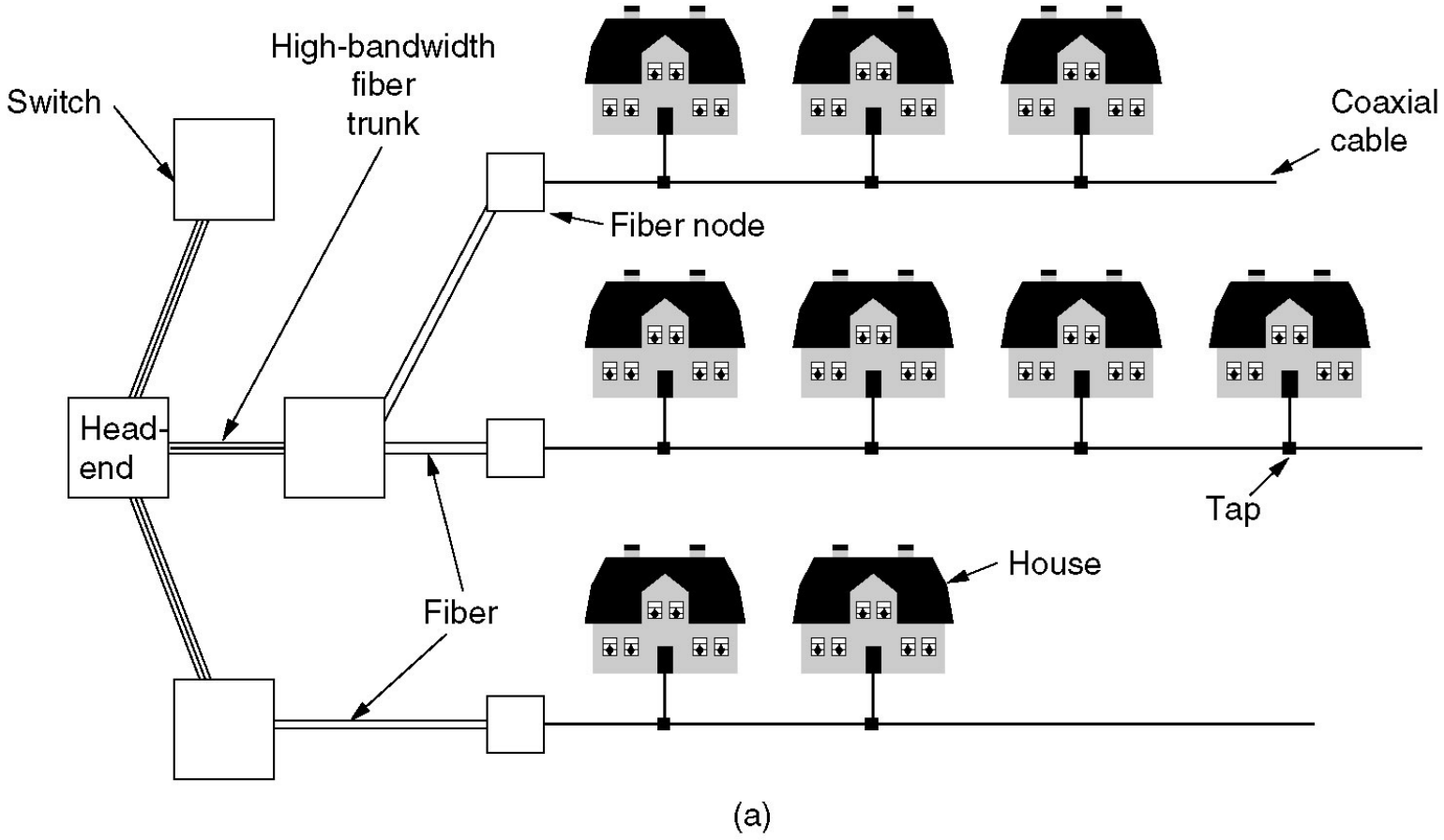
- Community Antenna Television
- Internet over Cable
- Spectrum Allocation
- Cable Modems
- ADSL versus Cable

Community Antenna Television



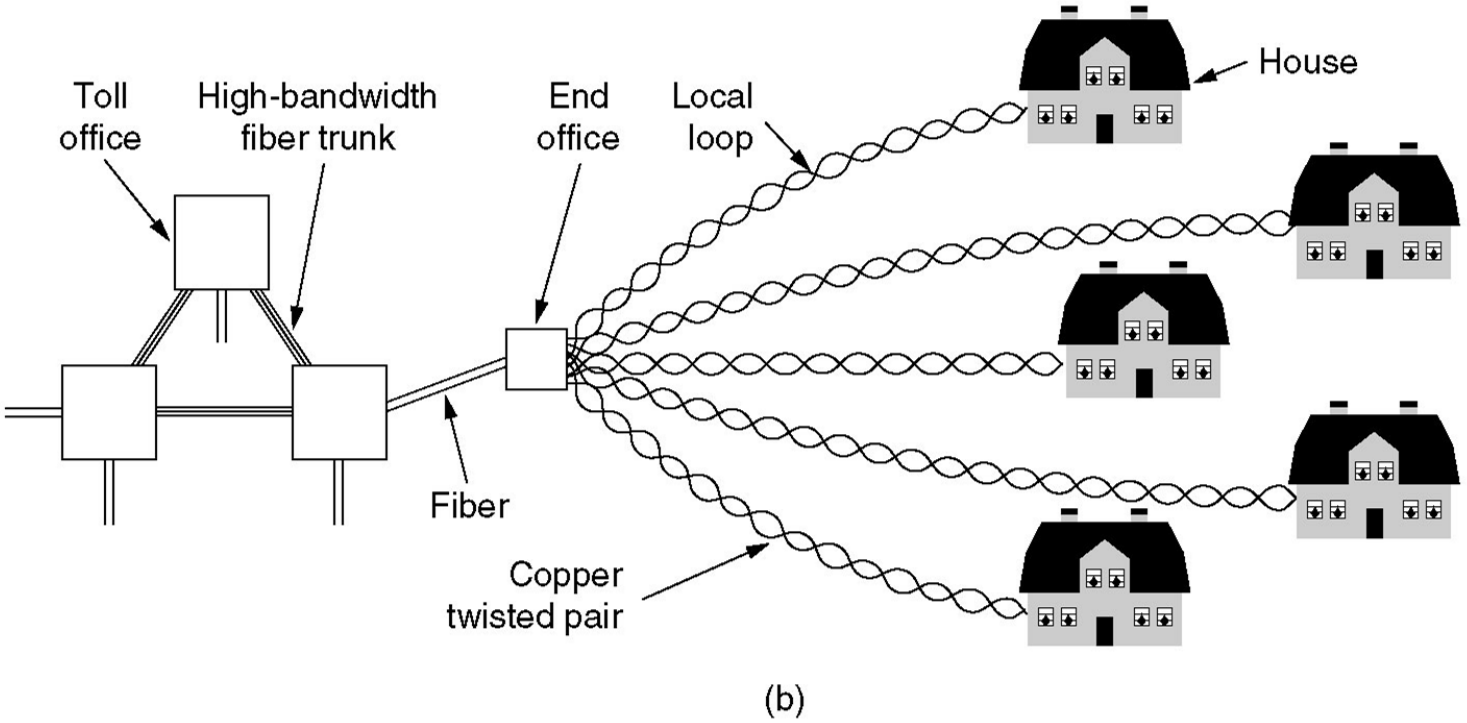
An early cable television system.

Internet over Cable



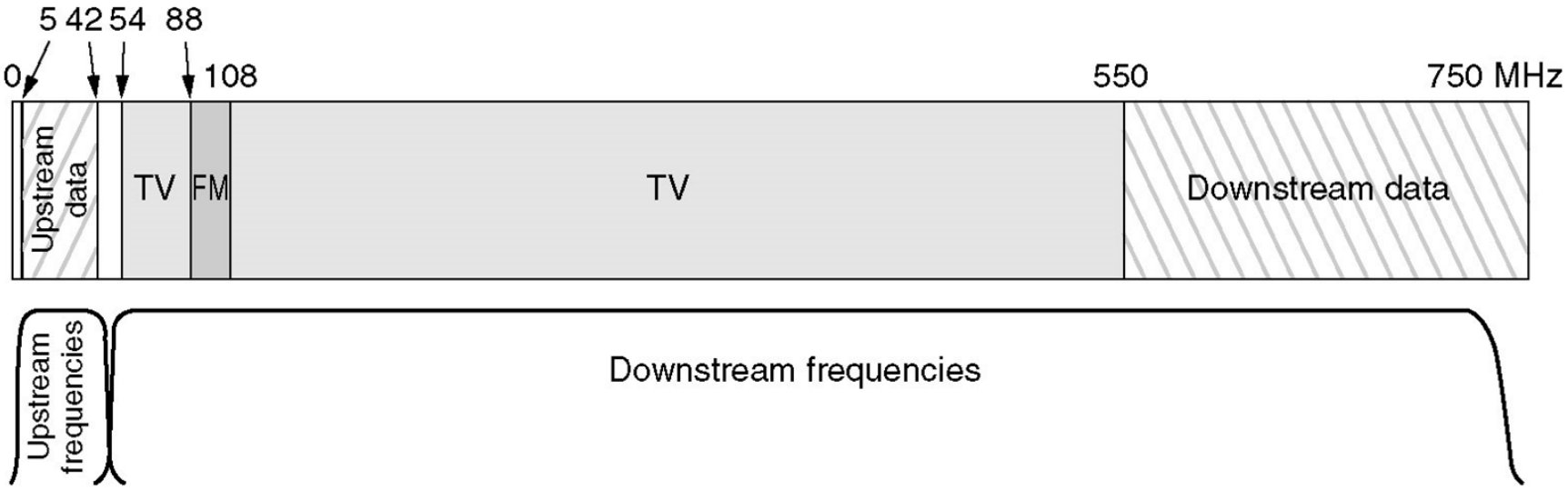
Cable television

Internet over Cable (2)



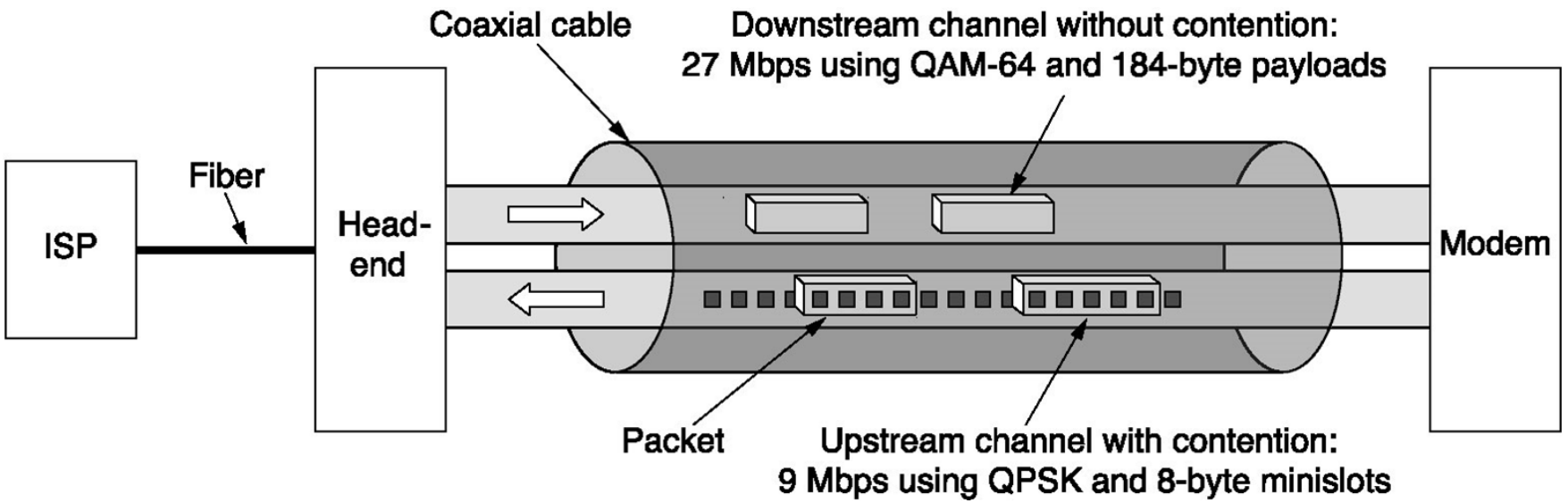
The fixed telephone system.

Spectrum Allocation



Frequency allocation in a typical cable TV system used for Internet access

Cable Modems



Typical details of the upstream and downstream channels in North America.

Chapter 3

The Data Link Layer

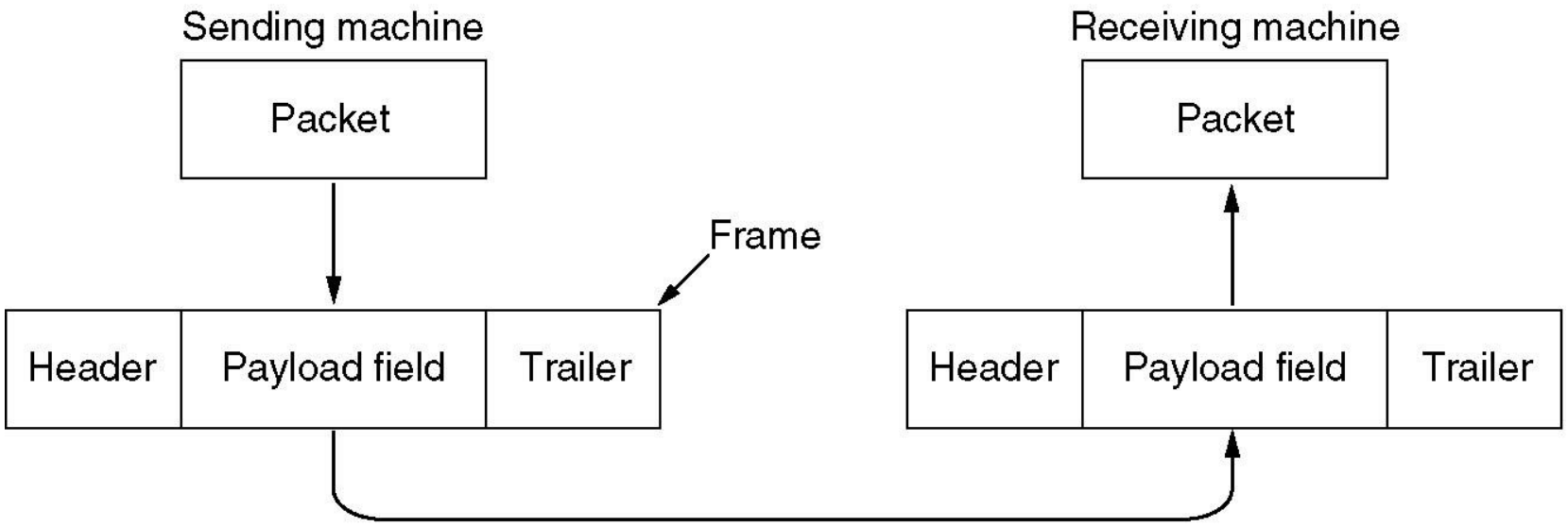
Data Link Layer Design Issues

- Services Provided to the Network Layer
- Framing
- Error Control
- Flow Control

Functions of the Data Link Layer

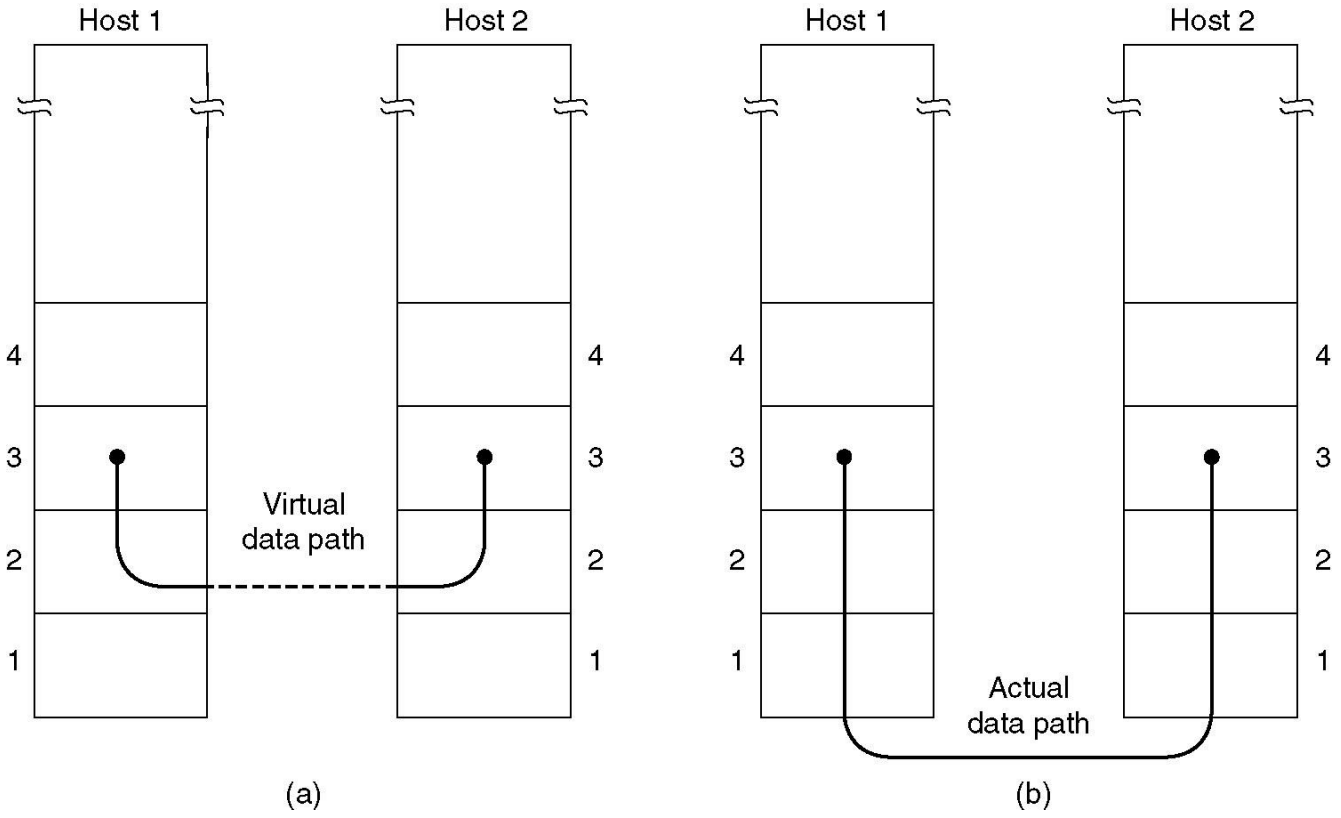
- Provide service interface to the network layer
- Dealing with transmission errors
- Regulating data flow
  - Slow receivers not swamped by fast senders

Functions of the Data Link Layer (2)



Relationship between packets and frames.

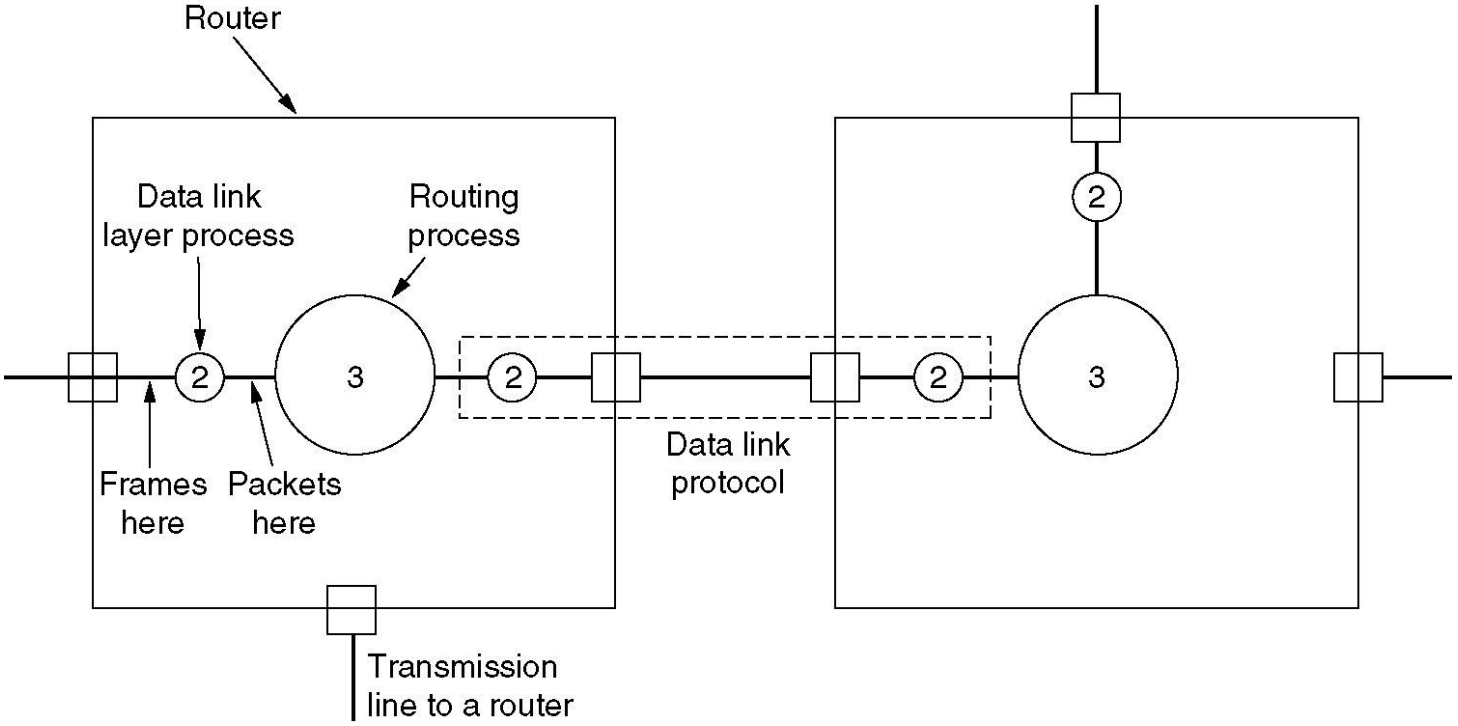
Services Provided to Network Layer



(a) Virtual communication.

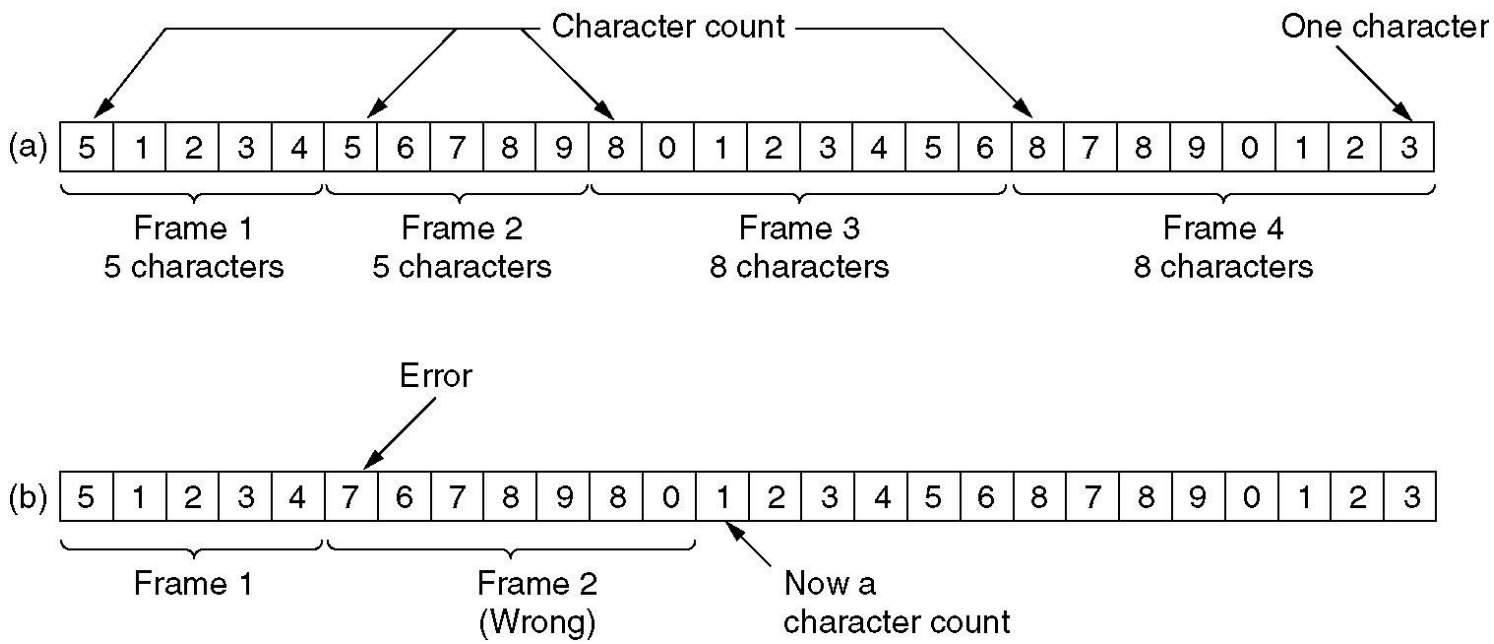
(b) Actual communication.

Services Provided to Network Layer (2)



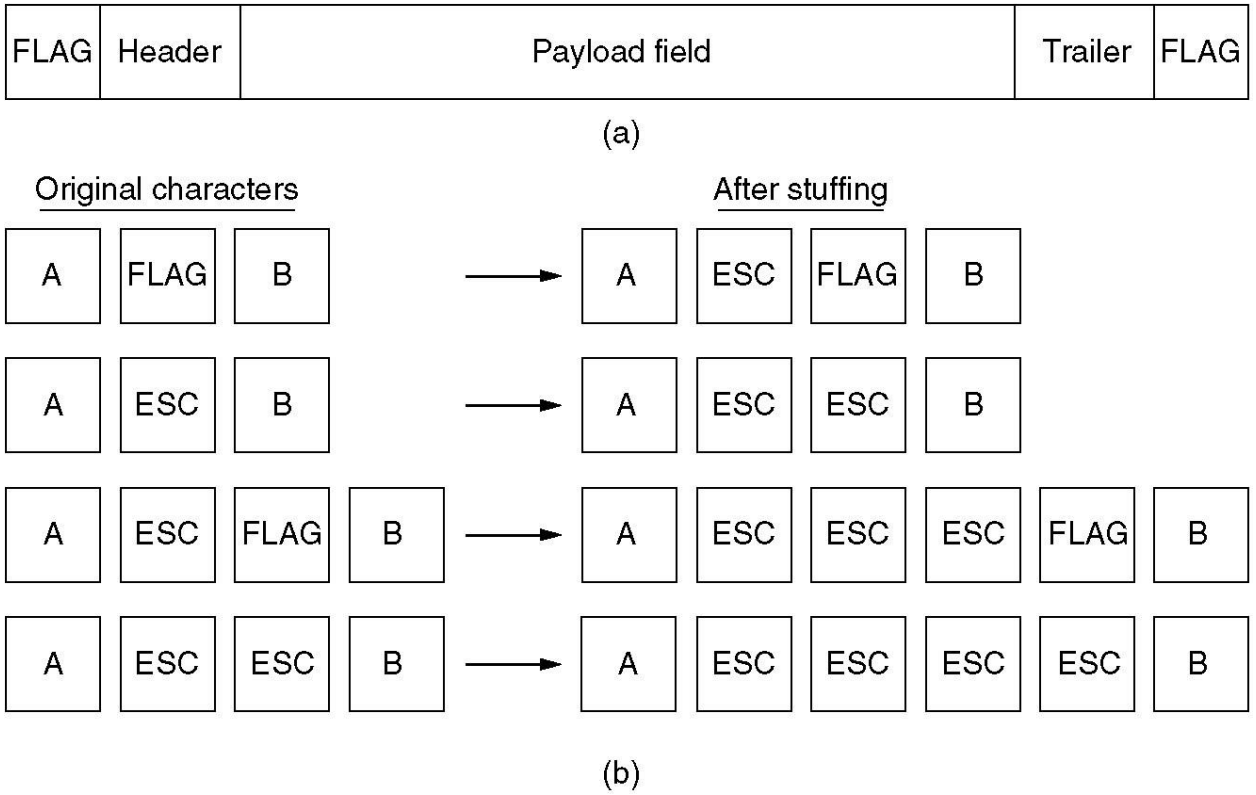
Placement of the data link protocol.

Framing



A character stream. (a) Without errors. (b) With one error.

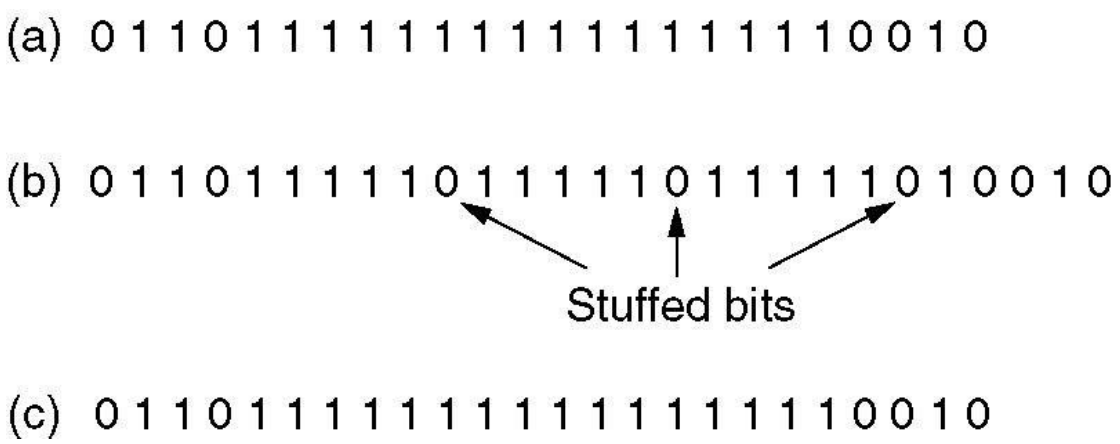
Framing (2)



(a) A frame delimited by flag bytes.

(b) Four examples of byte sequences before and after stuffing.

Framing (3)



Bit stuffing

(a) The original data.

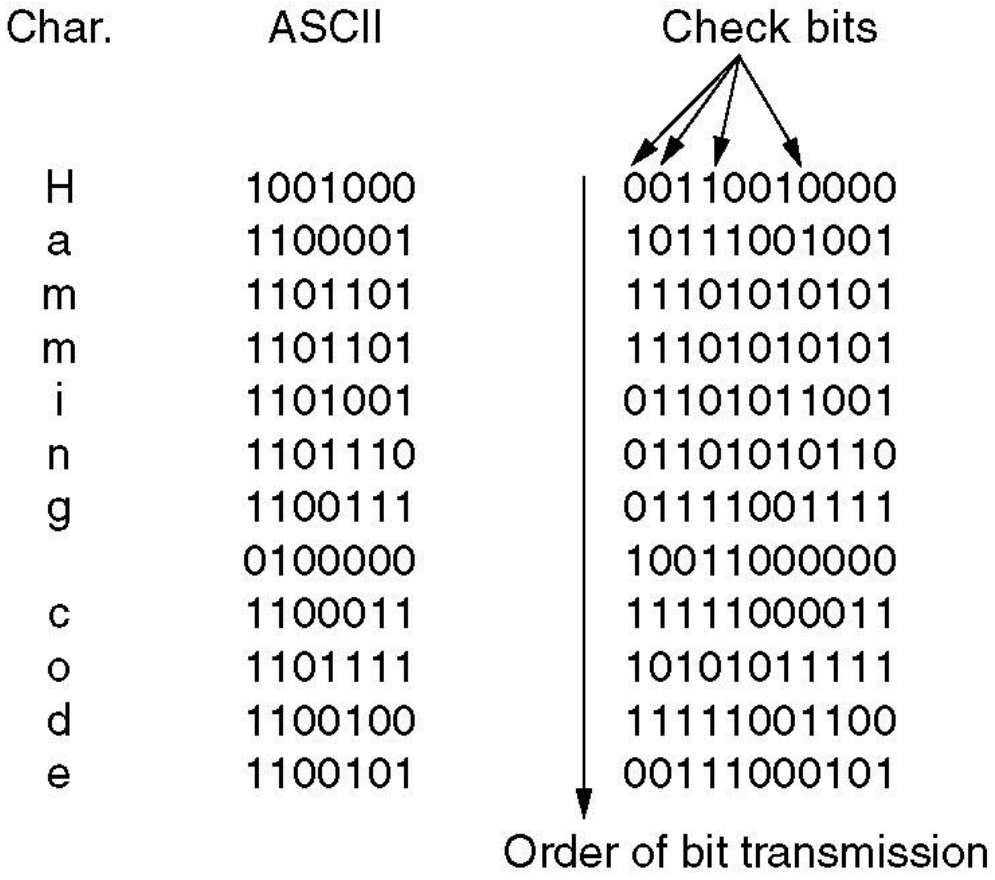
(b) The data as they appear on the line.

(c) The data as they are stored in receiver's memory after destuffing.

Error Detection and Correction

- Error-Correcting Codes
- Error-Detecting Codes

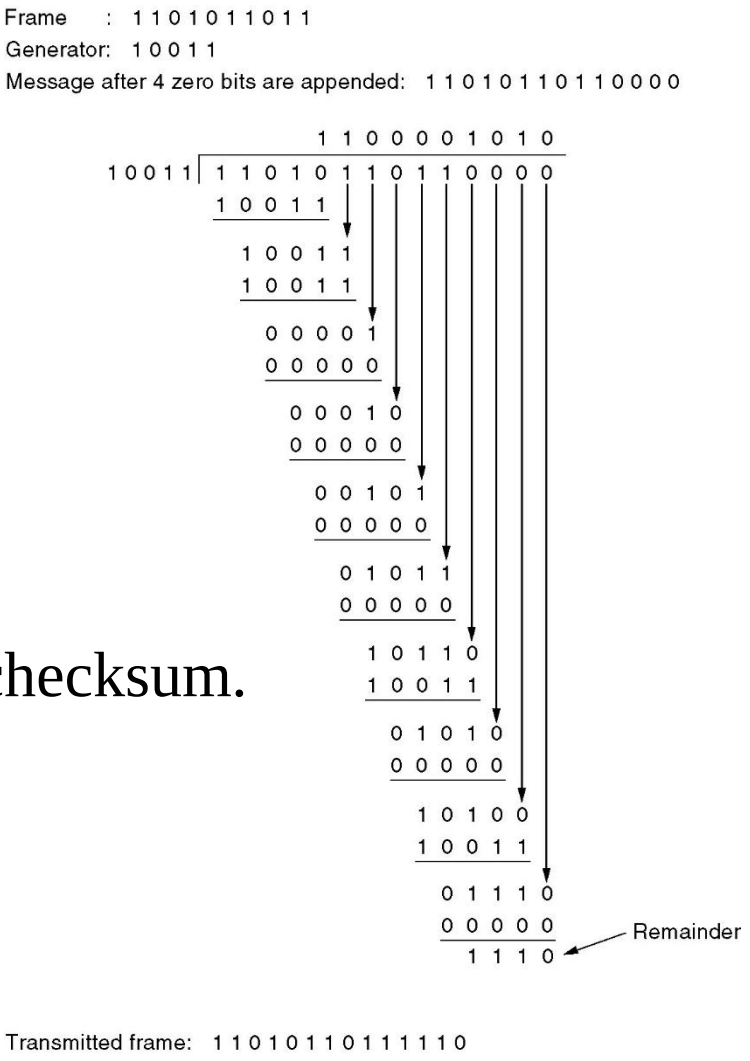
Error-Correcting Codes



Use of a Hamming code to correct burst errors.

Error-Detecting Codes

Calculation of the polynomial code checksum.



Elementary Data Link Protocols

- An Unrestricted Simplex Protocol
- A Simplex Stop-and-Wait Protocol
- A Simplex Protocol for a Noisy Channel

Protocol Definitions

```
#define MAX_PKT 1024                                  /* determines packet size in bytes */

typedef enum {false, true} boolean;                   /* boolean type */
typedef unsigned int seq_nr;                          /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT];} packet;/*    packet definition */
typedef enum {data, ack, nak} frame_kind;             /* frame_kind definition */

typedef struct {                                      /* frames are transported in this layer */
  frame_kind kind;                                    /* what kind of a frame is it? */
  seq_nr seq;                                         /* sequence number */
  seq_nr ack;                                         /* acknowledgement number */
  packet info;                                        /* the network layer packet */
} frame;
```

Continued →

Some definitions needed in the protocols to follow.
These are located in the file protocol.h.

Protocol Definitions (ctd.)

Some definitions needed in the protocols to follow. These are located in the file protocol.h.

```
/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);

/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
void enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void disable_network_layer(void);

/* Macro inc is expanded in-line: Increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

Unrestricted Simplex Protocol

```
/* Protocol 1 (utopia) provides for data transmission in one direction only, from
   sender to receiver.  The communication channel is assumed to be error free,
   and the receiver is assumed to be able to process all the input infinitely quickly.
   Consequently, the sender just sits in a loop pumping data out onto the line as
   fast as it can. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                              /* buffer for an outbound frame */
  packet buffer;                        /* buffer for an outbound packet */

  while (true) {
      from_network_layer(&buffer);      /* go get something to send */
      s.info = buffer;                  /* copy it into s for transmission */
      to_physical_layer(&s);            /* send it on its way */
  }                                     /*  * Tomorrow, and tomorrow, and tomorrow,
                                            Creeps in this petty pace from day to day
                                            To the last syllable of recorded time
                                                - Macbeth, V, v */
}

void receiver1(void)
{
  frame r;
  event_type event;                     /* filled in by wait, but not used here */

  while (true) {
      wait_for_event(&event);           /* only possibility is frame_arrival */
      from_physical_layer(&r);          /* go get the inbound frame */
      to_network_layer(&r.info);        /* pass the data to the network layer */
  }
}
```

Simplex Stop-and-Wait Protocol

```
/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be error
   free, as in protocol 1. However, this time, the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                              /* buffer for an outbound frame */
  packet buffer;                        /* buffer for an outbound packet */
  event_type event;                     /* frame_arrival is the only possibility */

  while (true) {
      from_network_layer(&buffer);      /* go get something to send */
      s.info = buffer;                  /* copy it into s for transmission */
      to_physical_layer(&s);            /* bye bye little frame */
      wait_for_event(&event);           /* do not proceed until given the go ahead */
  }
}

void receiver2(void)
{
  frame r, s;                           /* buffers for frames */
  event_type event;                     /* frame_arrival is the only possibility */
  while (true) {
      wait_for_event(&event);           /* only possibility is frame_arrival */
      from_physical_layer(&r);          /* go get the inbound frame */
      to_network_layer(&r.info);        /* pass the data to the network layer */
      to_physical_layer(&s);            /* send a dummy frame to awaken sender */
  }
}
```

A Simplex Protocol for a Noisy Channel

A positive acknowledgement with retransmission protocol.

```
/* Protocol 3 (par) allows unidirectional data flow over an unreliable channel. */

#define MAX_SEQ 1                                   /* must be 1 for protocol 3 */
typedef enum  {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;                        /* seq number of next outgoing frame */
  frame s;                                          /* scratch variable */
  packet buffer;                                    /* buffer for an outbound packet */
  event_type event;

  next_frame_to_send = 0;                           /* initialize outbound sequence numbers */
  from_network_layer(&buffer);                      /* fetch first packet */
  while (true) {
      s.info = buffer;                              /* construct a frame for transmission */
      s.seq = next_frame_to_send;                   /* insert sequence number in frame */
      to_physical_layer(&s);                        /* send it on its way */
      start_timer(s.seq);                           /* if answer takes too long, time out */
      wait_for_event(&event);                       /* frame_arrival, cksum_err, timeout */
      if (event == frame_arrival) {
          from_physical_layer(&s);                  /* get the acknowledgement */
          if (s.ack == next_frame_to_send) {
              stop_timer(s.ack);                    /* turn the timer off */
              from_network_layer(&buffer);          /* get the next one to send */
              inc(next_frame_to_send);              /* invert next_frame_to_send */
          }
      }
  }
}
```

Continued →

A Simplex Protocol for a Noisy Channel (ctd.)

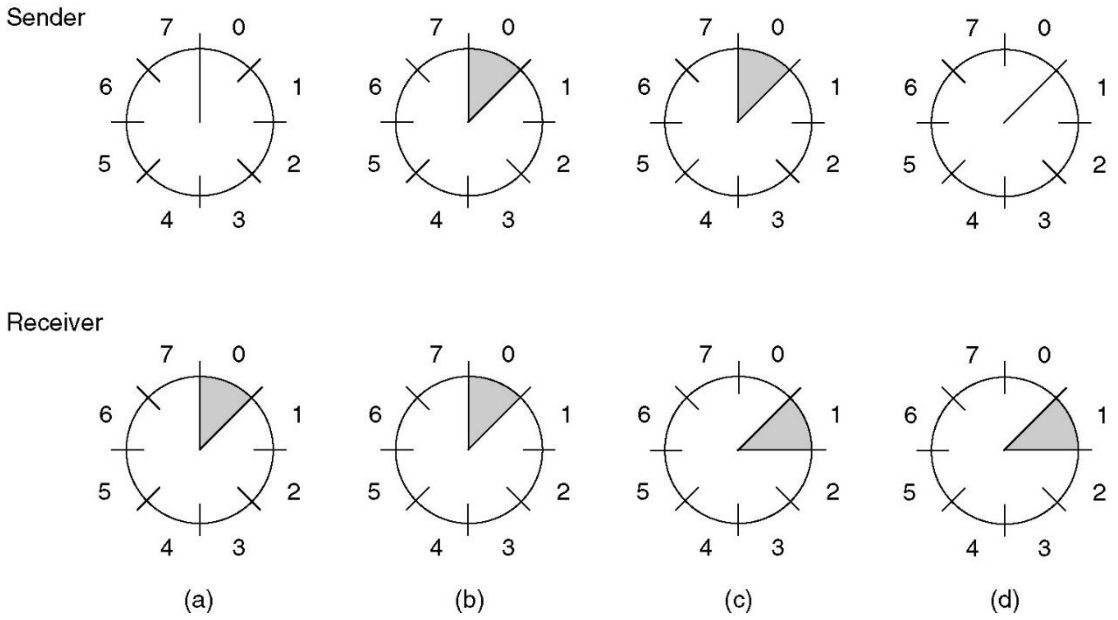
```
void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true) {
       wait_for_event(&event);                  /* possibilities: frame_arrival, cksum_err */
       if (event == frame_arrival) {            /* a valid frame has arrived. */
            from_physical_layer(&r);            /* go get the newly arrived frame */
            if (r.seq == frame  expected) {     /* this is what we have been waiting for. */
                 to_network_layer(&r.info);     /* pass the data to the network layer */
                 inc(frame_expected);           /* next time expect the other sequence nr */
            }
            s.ack = 1 – frame_expected;         /* tell which frame is being acked */
            to_physical_layer(&s);              /* send acknowledgement */
       }
  }
}
```

A positive acknowledgement with retransmission protocol.

Sliding Window Protocols

- A One-Bit Sliding Window Protocol
- A Protocol Using Go Back N
- A Protocol Using Selective Repeat

Sliding Window Protocols (2)



A sliding window of size 1, with a 3-bit sequence number.

(a) Initially.

(b) After the first frame has been sent.

(c) After the first frame has been received.

(d) After the first acknowledgement has been received.

A One-Bit Sliding Window Protocol

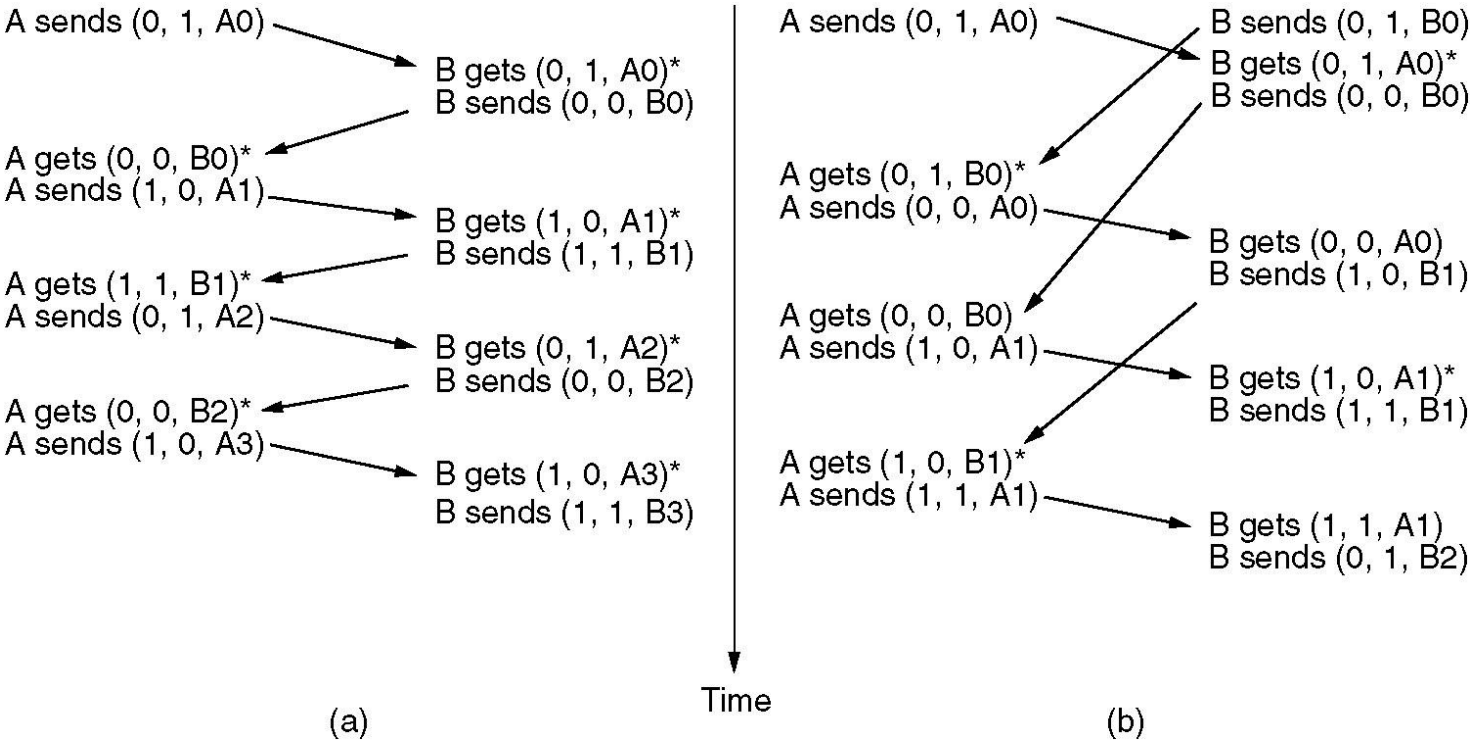
```
/* Protocol 4 (sliding window) is bidirectional. */
#define MAX_SEQ 1                                   /* must be 1 for protocol 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
{
  seq_nr next_frame_to_send;                        /* 0 or 1 only */
  seq_nr frame_expected;                            /* 0 or 1 only */
  frame r, s;                                       /* scratch variables */
  packet buffer;                                    /* current packet being sent */
  event_type event;
  next_frame_to_send = 0;                           /* next frame on the outbound stream */
  frame_expected = 0;                               /* frame expected next */
  from_network_layer(&buffer);                      /* fetch a packet from the network layer */
  s.info = buffer;                                  /* prepare to send the initial frame */
  s.seq = next_frame_to_send;                       /* insert sequence number into frame */
  s.ack = 1 – frame_expected;                       /* piggybacked ack */
  to_physical_layer(&s);                            /* transmit the frame */
  start_timer(s.seq);                               /* start the timer running */
```


Continued →

A One-Bit Sliding Window Protocol (ctd.)

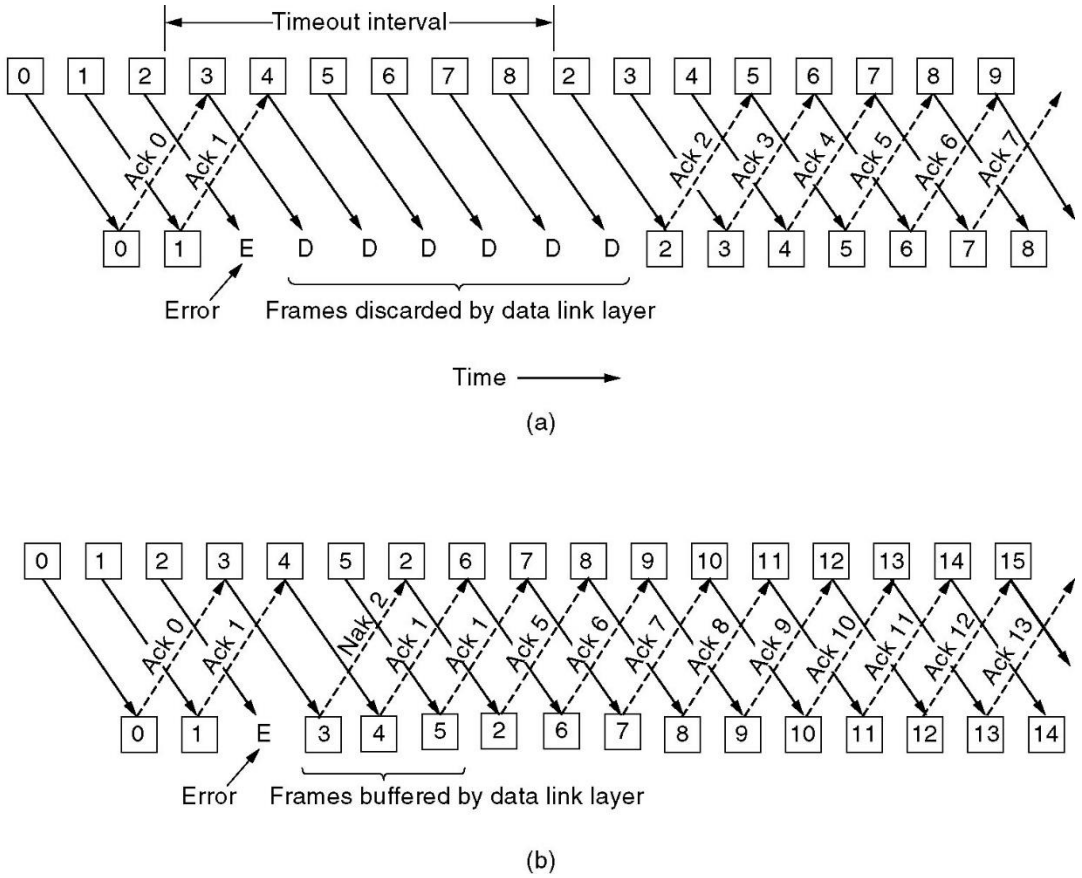
```
while (true) {
    wait_for_event(&event);                 /* frame_arrival, cksum_err, or timeout */
    if (event == frame_arrival) {           /* a frame has arrived undamaged. */
        from_physical_layer(&r);            /* go get it */

        if (r.seq == frame_expected) {      /* handle inbound frame stream. */
            to_network_layer(&r.info);      /* pass packet to network layer */
            inc(frame_expected);            /* invert seq number expected next */
        }

        if (r.ack == next_frame_to_send) {  /* handle outbound frame stream. */
            stop_timer(r.ack);              /* turn the timer off */
            from_network_layer(&buffer);    /* fetch new pkt from network layer */
            inc(next_frame_to_send);        /* invert senderís sequence number */
        }
    }
    s.info = buffer;                        /* construct outbound frame */
    s.seq = next_frame_to_send;             /* insert sequence number into it */
    s.ack = 1 – frame_expected;             /* seq number of last received frame */
    to_physical_layer(&s);                  /* transmit a frame */
    start_timer(s.seq);                     /* start the timer running */
  }
}
```

A One-Bit Sliding Window Protocol (2)



Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

A Protocol Using Go Back N



Pipelining and error recovery. Effect on an error when

(a) Receiver's window size is 1.

(b) Receiver's window size is large.

Sliding Window Protocol Using Go Back N

```
/* Protocol 5 (pipelining) allows multiple outstanding frames. The sender may transmit up
   to MAX_SEQ frames without waiting for an ack. In addition, unlike the previous protocols,
   the network layer is not assumed to have a new packet all the time. Instead, the
   network layer causes a network_layer_ready event when there is a packet to send. */

#define MAX_SEQ 7                       /* should be 2^n – 1 */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Return true if a <=b < c circularly; false otherwise. */
  if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
        return(true);
    else
        return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/* Construct and send a data frame. */
  frame s;                              /* scratch variable */

  s.info = buffer[frame_nr];            /* insert packet into frame */
  s.seq = frame_nr;                     /* insert sequence number into frame */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);/* piggyback ack */
  to_physical_layer(&s);                /* transmit the frame */
  start_timer(frame_nr);                /* start the timer running */
}
```


Continued →

Sliding Window Protocol Using Go Back N

```
void protocol5(void)
{
  seq_nr next_frame_to_send;          /* MAX_SEQ > 1; used for outbound stream */
  seq_nr ack_expected;                /* oldest frame as yet unacknowledged */
  seq_nr frame_expected;              /* next frame expected on inbound stream */
  frame r;                            /* scratch variable */
  packet buffer[MAX_SEQ + 1];         /* buffers for the outbound stream */
  seq_nr nbuffered;                   /* # output buffers currently in use */
  seq_nr i;                           /* used to index into the buffer array */
  event_type event;

  enable_network_layer();             /* allow network_layer_ready events */
  ack_expected = 0;                   /* next ack expected inbound */
  next_frame_to_send = 0;             /* next frame going out */
  frame_expected = 0;                 /* number of frame expected inbound */
  nbuffered = 0;                      /* initially no packets are buffered */
```

Continued →

Sliding Window Protocol Using Go Back N

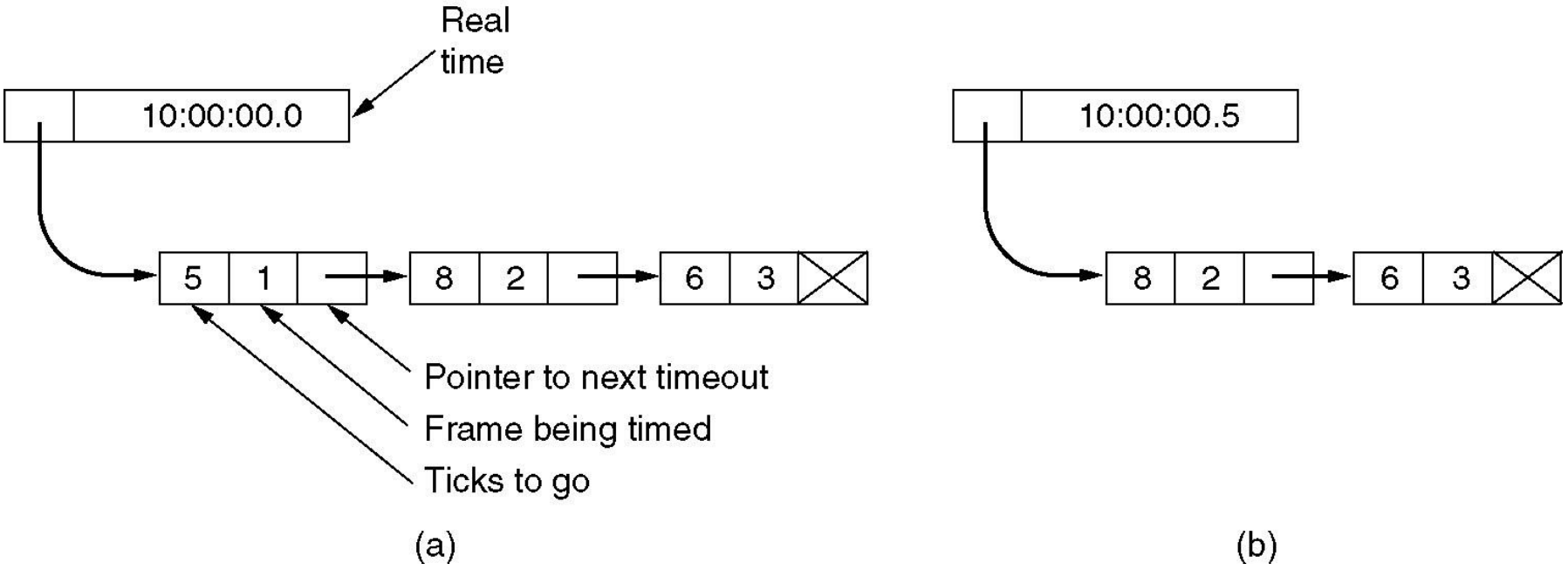
```
while (true) {
  wait_for_event(&event);              /* four possibilities: see event_type above */

  switch(event) {
    case network_layer_ready:          /* the network layer has a packet to send */
         /* Accept, save, and transmit a new frame. */
         from_network_layer(&buffer[next_frame_to_send]); /* fetch new packet */
         nbuffered = nbuffered + 1;    /* expand the sender's window */
         send_data(next_frame_to_send, frame_expected, buffer);/* transmit the frame */
         inc(next_frame_to_send);      /* advance sender's upper window edge */
         break;

    case frame_arrival:                /* a data or control frame has arrived */
         from_physical_layer(&r);      /* get incoming frame from physical layer */

         if (r.seq == frame_expected) {
              /* Frames are accepted only in order. */
              to_network_layer(&r.info);  /* pass packet to network layer */
              inc(frame_expected);   /* advance lower edge of receiver's window */
          }
```


Continued →

Sliding Window Protocol Using Go Back N

```
            /* Ack n implies n – 1, n – 2, etc.  Check for this. */
            while (between(ack_expected, r.ack, next_frame_to_send)) {
                /* Handle piggybacked ack. */
                nbuffered = nbuffered    1; /* one frame fewer buffered */
                stop_timer(ack_expected); /* frame arrived intact; stop timer */
                inc(ack_expected);       /* contract sender's window */
            }
            break;

        case cksum_err: break;              /* just ignore bad frames */

        case timeout:                       /* trouble; retransmit all outstanding frames */
            next_frame_to_send = ack_expected;     /* start retransmitting here */
            for (i = 1; i <= nbuffered; i++) {
                send_data(next_frame_to_send, frame_expected, buffer);/* resend 1 frame */
                inc(next_frame_to_send);   /* prepare to send the next one */
            }

    }

    if (nbuffered < MAX_SEQ)
        enable_network_layer();
    else
        disable_network_layer();
  }
}
```

Sliding Window Protocol Using Go Back N (2)



Simulation of multiple timers in software.

A Sliding Window Protocol Using Selective Repeat

```
/* Protocol 6 (nonsequential receive) accepts frames out of order, but passes packets to the
   network layer in order. Associated with each outstanding frame is a timer. When the timer
   expires, only that frame is retransmitted, not all the outstanding frames, as in protocol 5. */

#define MAX_SEQ 7                                   /* should be 2^n – 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;
#include "protocol.h"
boolean no_nak = true;                              /* no nak has been sent yet */
seq_nr oldest_frame = MAX- SEQ + 1;                 /* initial value is only for the simulator */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Same as between in protocol5, but shorter and more obscure. */
  return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/* Construct and send a data, ack, or nak frame. */
  frame s;                                          /* scratch variable */

  s.kind = fk;                                      /* kind == data, ack, or nak */
  if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
  s.seq = frame_nr;                                 /* only meaningful for data frames */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
  if (fk == nak) no_nak = false;                    /* one nak per frame, please */
  to_physical_layer(&s);                            /* transmit the frame */
  if (fk == data) start_timer(frame_nr % NR_BUFS);
  stop_ack_timer();                                 /* no need for separate ack frame */
}
```


Continued →

A Sliding Window Protocol Using Selective Repeat (2)

```
void protocol6(void)
{
  seq_nr ack_expected;                          /* lower edge of sender's window */
  seq_nr next_frame_to_send;                    /* upper edge of sender's window + 1 */
  seq_nr frame_expected;                        /* lower edge of receiver's window */
  seq_nr too_far;                               /* upper edge of receiver's window + 1 */
  int i;                                        /* index into buffer pool */
  frame r;                                      /* scratch variable */
  packet out_buf[NR_BUFS];                      /* buffers for the outbound stream */
  packet in_buf[NR_BUFS];                       /* buffers for the inbound stream */
  boolean arrived[NR_BUFS];                     /* inbound bit map */
  seq_nr nbuffered;                             /* how many output buffers currently used */
  event_type event;

  enable_network_layer();                       /* initialize */
  ack_expected = 0;                             /* next ack expected on the inbound stream */
  next_frame_to_send = 0;                       /* number of next outgoing frame */
  frame_expected = 0;
  too_far = NR_BUFS;
  nbuffered = 0;                                /* initially no packets are buffered */
  for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
```


Continued →

A Sliding Window Protocol Using Selective Repeat (3)

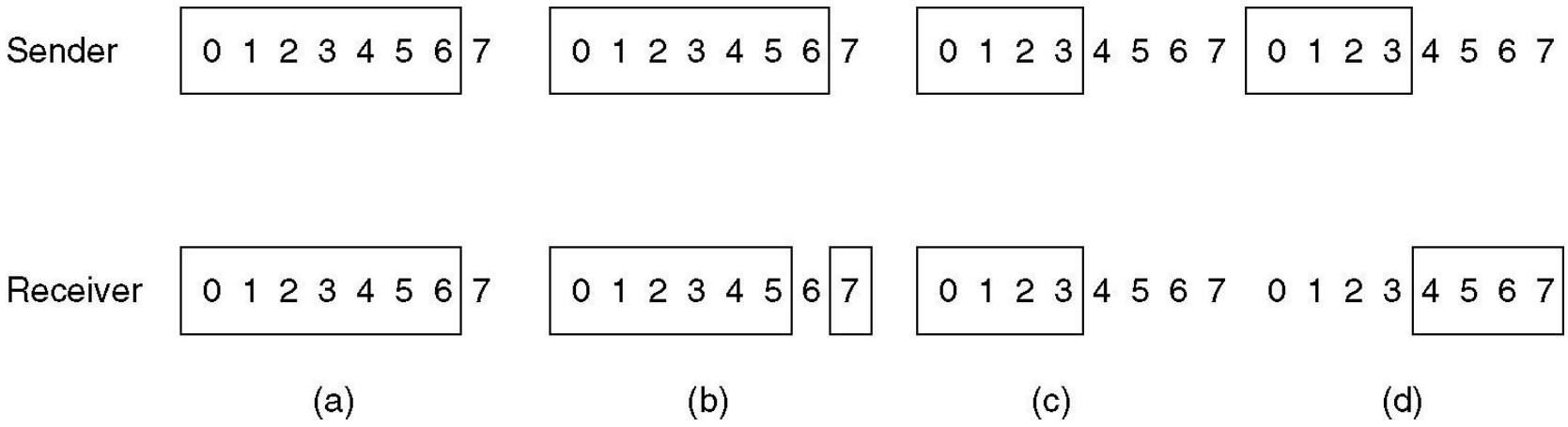
```
while (true) {
  wait_for_event(&event);                          /* five possibilities: see event_type above */
  switch(event) {
    case network_layer_ready:                      /* accept, save, and transmit a new frame */
         nbuffered = nbuffered + 1;                /* expand the window */
         from_network_layer(&out  buf[next_frame_to_send % NR_BUFS]); /* fetch new packet */
         send_frame(data, next_frame_to_send, frame_expected, out_buf);/* transmit the frame */
         inc(next_frame_to_send);                  /* advance upper window edge */
         break;

    case frame_arrival:                            /* a data or control frame has arrived */
         from_physical_layer(&r);                  /* fetch incoming frame from physical layer */
         if (r.kind == data) {
              /* An undamaged frame has arrived. */
              if ((r.seq != frame_expected) && no_nak)
                 send_frame(nak, 0, frame_expected, out_buf); else start_ack_timer();
              if (between(frame_expected, r.seq, too_far) && (arrived[r.seq%NR_BUFS] == false)) {
                   /* Frames may be accepted in any order. */
                   arrived[r.seq % NR_BUFS] = true;       /* mark buffer as full */
                   in_buf[r.seq % NR_BUFS] = r.info;      /* insert data into buffer */
                   while (arrived[frame_expected % NR_BUFS]) {
                        /* Pass frames and advance window. */
                        to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                        no_nak = true;
                        arrived[frame_expected % NR_BUFS] = false;
                        inc(frame_expected);      /* advance lower edge of receiver's window */
                        inc(too_far);             /* advance upper edge of receiver's window */
                        start_ack_timer();        /* to see if a separate ack is needed */
                   }
              }
         }
```

Continued →

A Sliding Window Protocol Using Selective Repeat (4)

```
        if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1),next frame to send))
            send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);

        while (between(ack_expected, r.ack, next_frame_to_send)) {
            nbuffered = nbuffered    1;          /* handle piggybacked ack */
            stop_timer(ack_expected % NR_BUFS);     /* frame arrived intact */
            inc(ack_expected);                  /* advance lower edge of sender's window */
        }
        break;
    case cksum_err:
        if (no_nak) send_frame(nak, 0, frame_expected, out_buf);/* damaged frame */
        break;
    case timeout:
        send_frame(data, oldest_frame, frame_expected, out_buf);/* we timed out */
        break;
    case ack_timeout:
        send_frame(ack,0,frame_expected, out_buf);      /* ack timer expired; send ack */
    }
    if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
  }
}
```

A Sliding Window Protocol Using Selective Repeat (5)



(a) Initial situation with a window size seven.

(b) After seven frames sent and received, but not acknowledged.

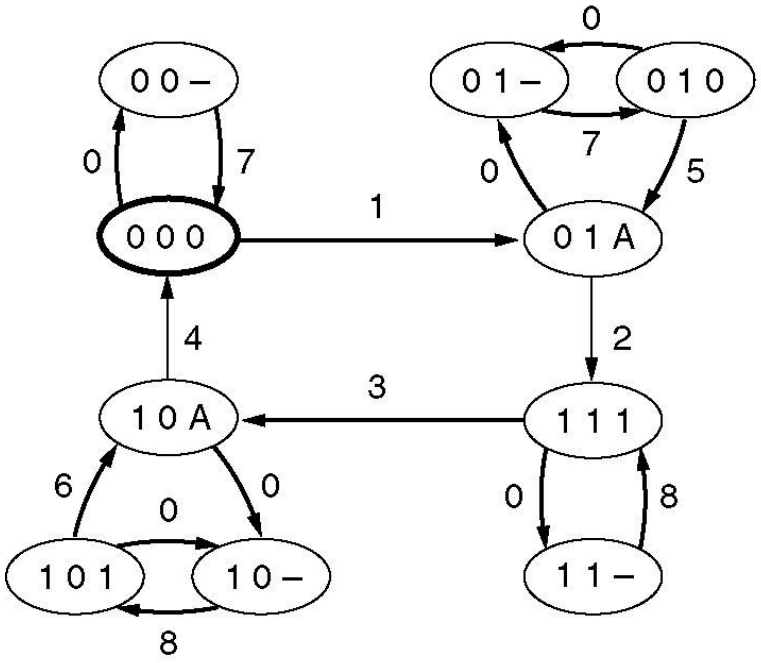
(c) Initial situation with a window size of four.

(d) After four frames sent and received, but not acknowledged.

Protocol Verification

- Finite State Machined Models
- Petri Net Models

Finite State Machined Models



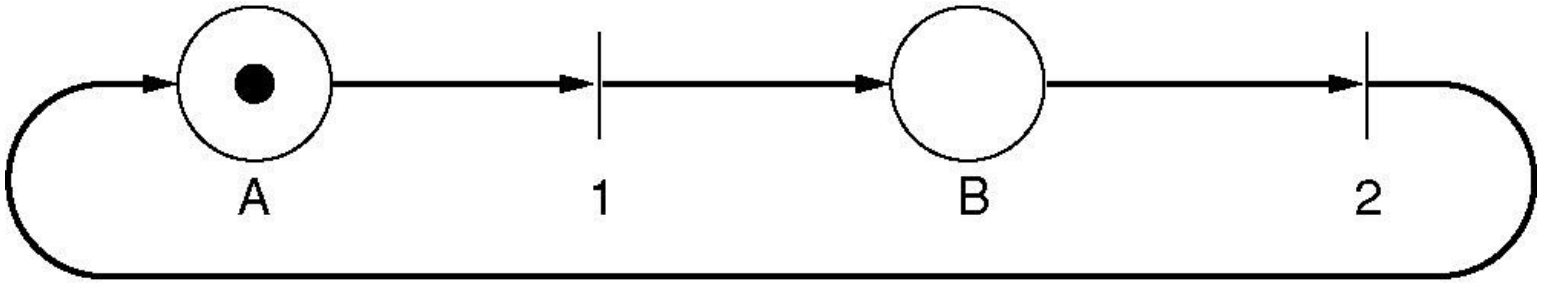
(a)

Transition	Who runs?	Frame accepted	Frame emitted	To network layer
0	–	(frame lost)		–
1	R	0	A	Yes
2	S	A	1	–
3	R	1	A	Yes
4	S	A	0	–
5	R	0	A	No
6	R	1	A	No
7	S	(timeout)	0	–
8	S	(timeout)	1	–

(b)

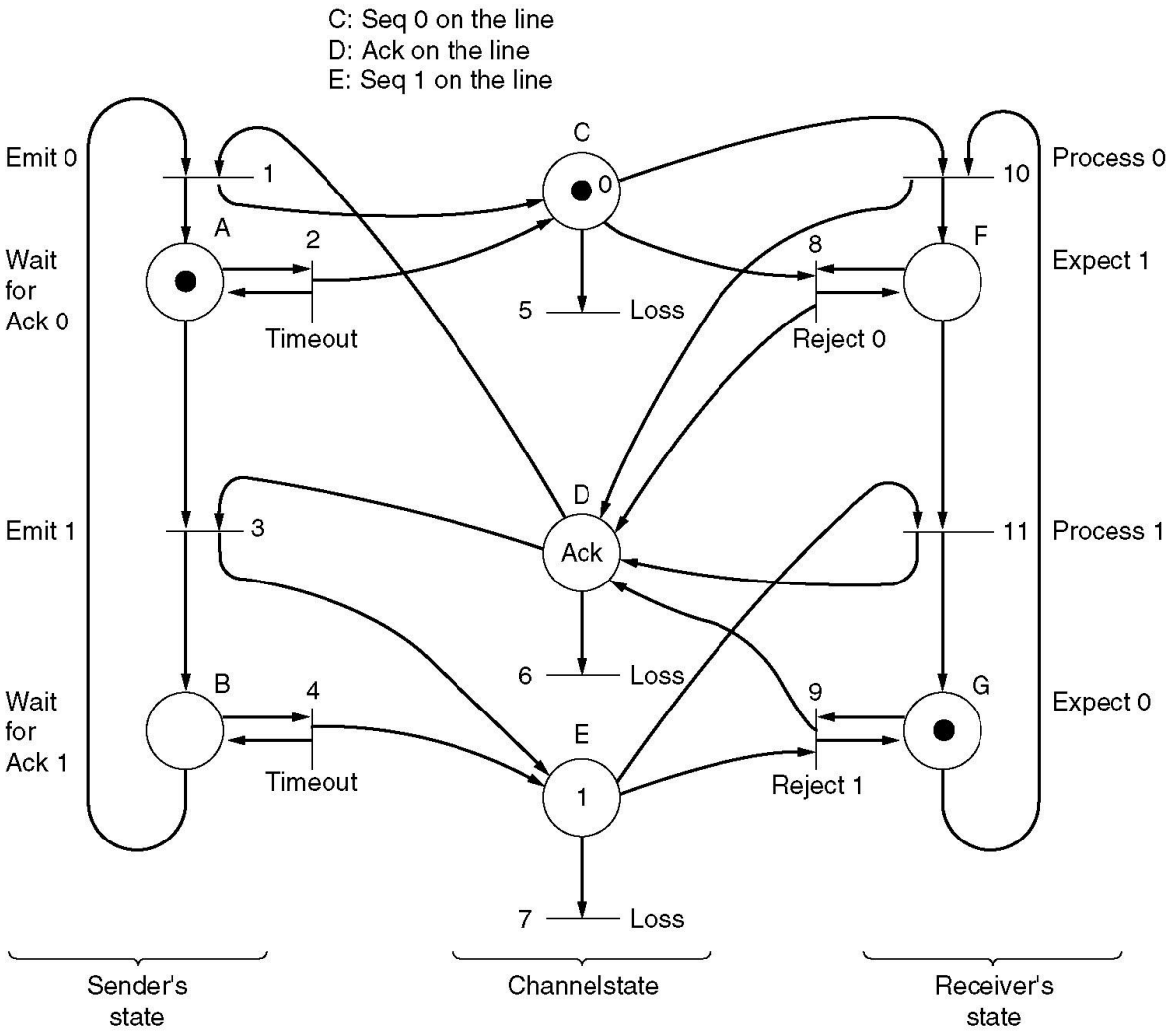
(a) State diagram for protocol 3. (b) Transmissions.

Petri Net Models



A Petri net with two places and two transitions.

Petri Net Models (2)

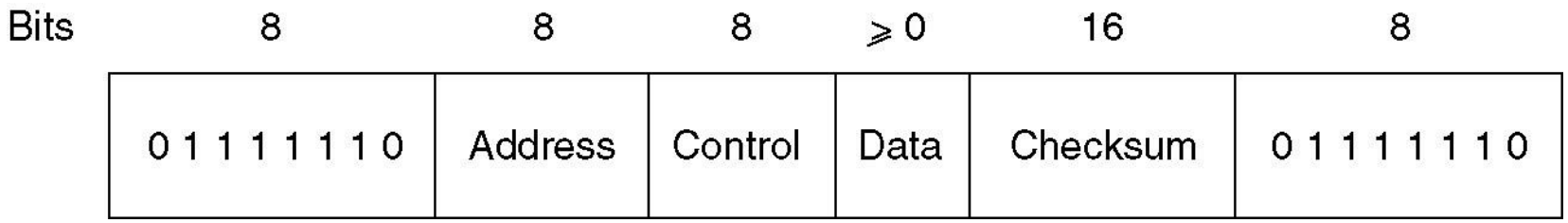


A Petri net model for protocol 3.

Example Data Link Protocols

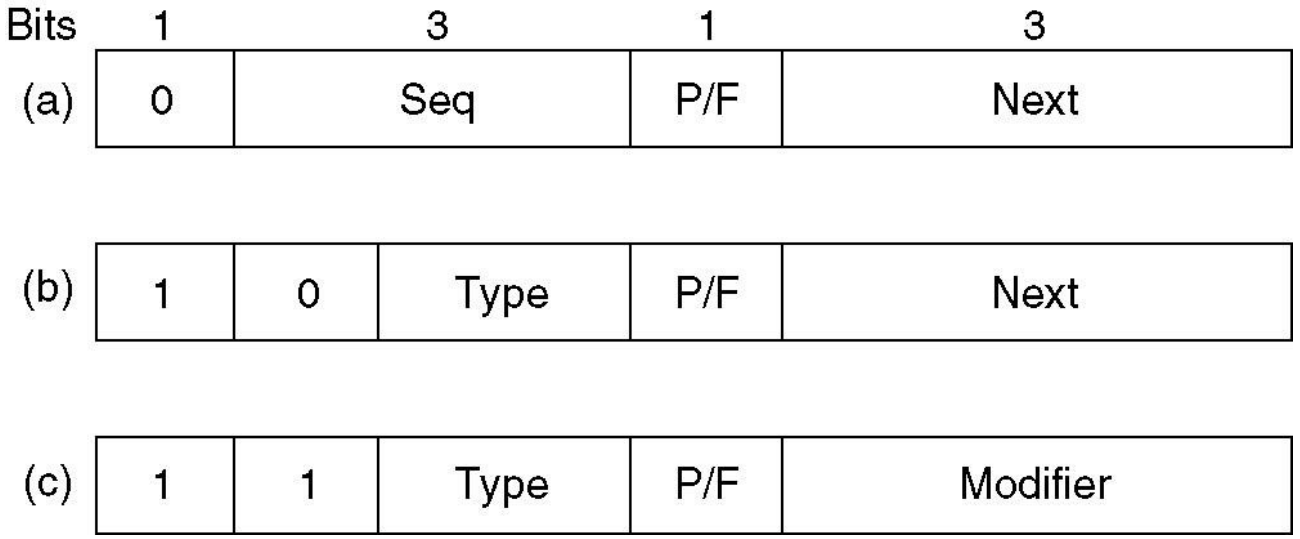
- HDLC – High-Level Data Link Control
- The Data Link Layer in the Internet

High-Level Data Link Control



Frame format for bit-oriented protocols.

High-Level Data Link Control (2)



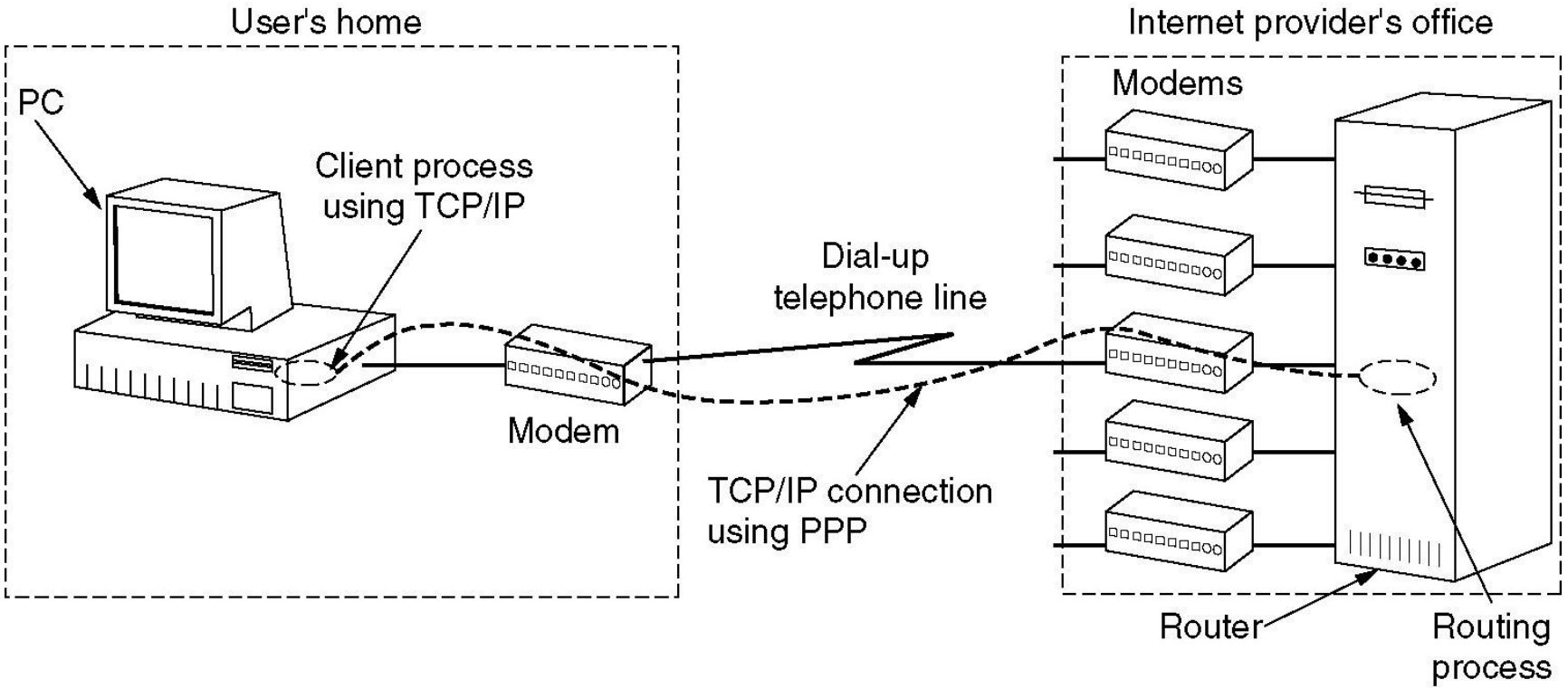
Control field of

(a) An information frame.

(b) A supervisory frame.

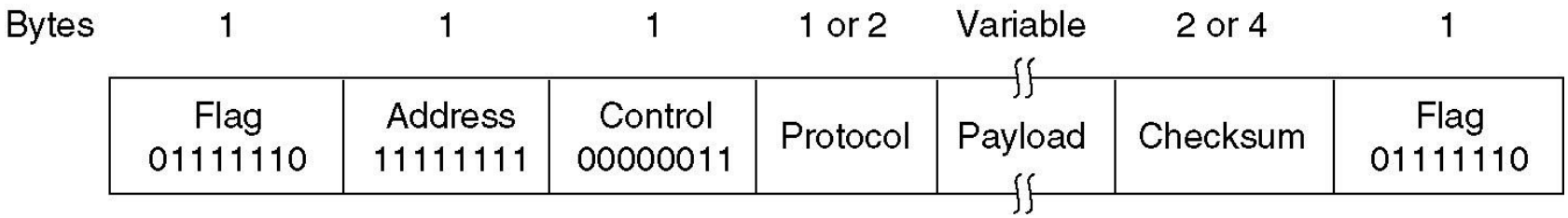
(c) An unnumbered frame.

The Data Link Layer in the Internet



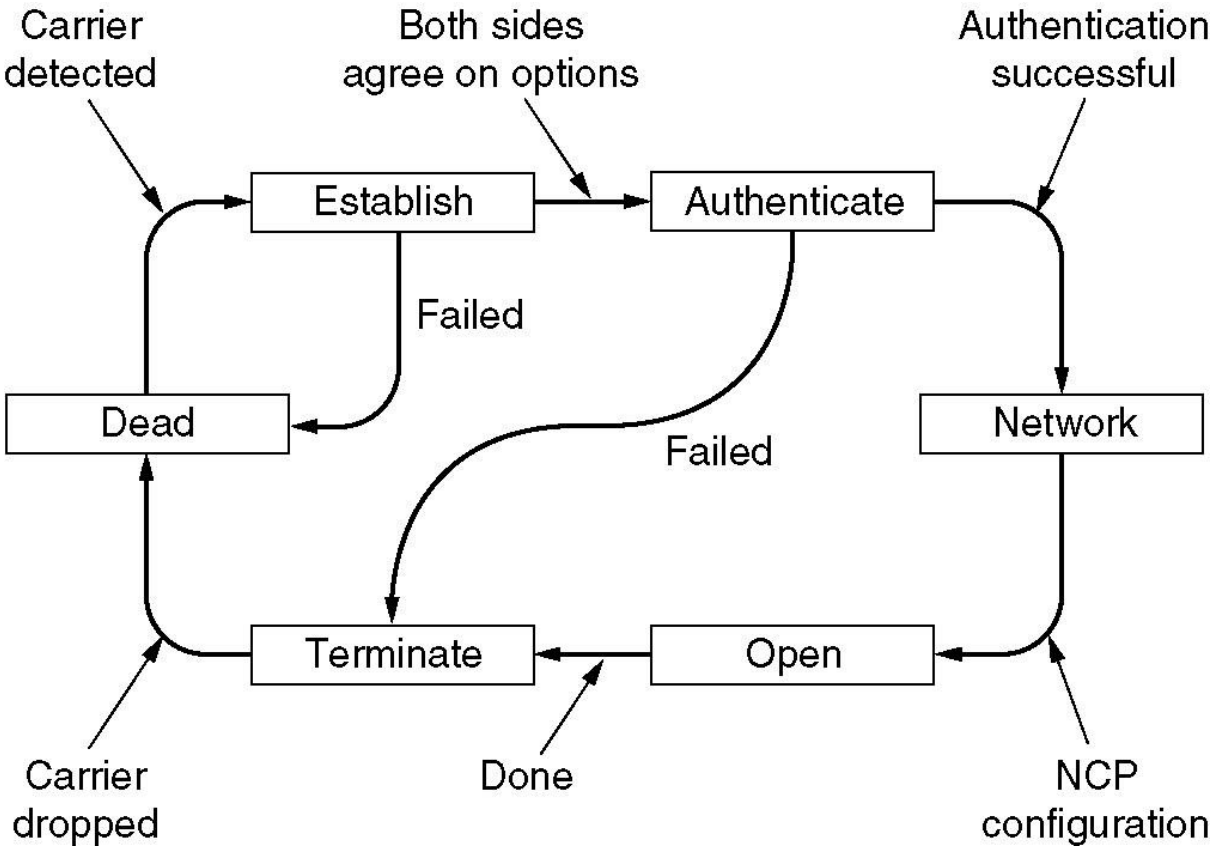
A home personal computer acting as an internet host.

PPP – Point to Point Protocol



The PPP full frame format for unnumbered mode operation.

PPP – Point to Point Protocol (2)



A simplified phase diagram for bring a line up and down.

PPP – Point to Point Protocol (3)

Name	Direction	Description
Configure-request	I → R	List of proposed options and values
Configure-ack	I ← R	All options are accepted
Configure-nak	I ← R	Some options are not accepted
Configure-reject	I ← R	Some options are not negotiable
Terminate-request	I → R	Request to shut the line down
Terminate-ack	I ← R	OK, line shut down
Code-reject	I ← R	Unknown request received
Protocol-reject	I ← R	Unknown protocol requested
Echo-request	I → R	Please send this frame back
Echo-reply	I ← R	Here is the frame back
Discard-request	I → R	Just discard this frame (for testing)

The LCP frame types.

Chapter 4

The Medium Access Control Sublayer

The Channel Allocation Problem

- Static Channel Allocation in LANs and MANs
- Dynamic Channel Allocation in LANs and MANs

Dynamic Channel Allocation in LANs and MANs

1. Station Model.

2. Single Channel Assumption.

3. Collision Assumption.

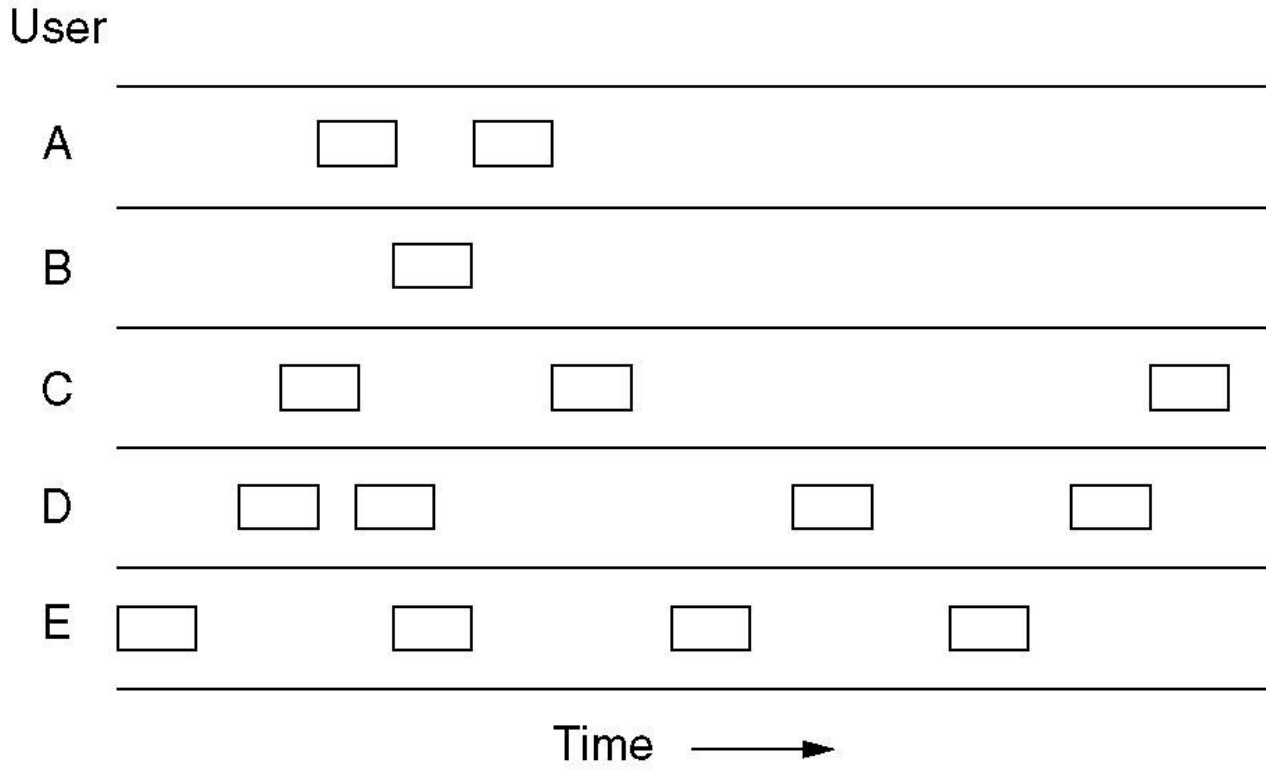
4. (a) Continuous Time.
   (b) Slotted Time.

5. (a) Carrier Sense.
   (b) No Carrier Sense.

Multiple Access Protocols

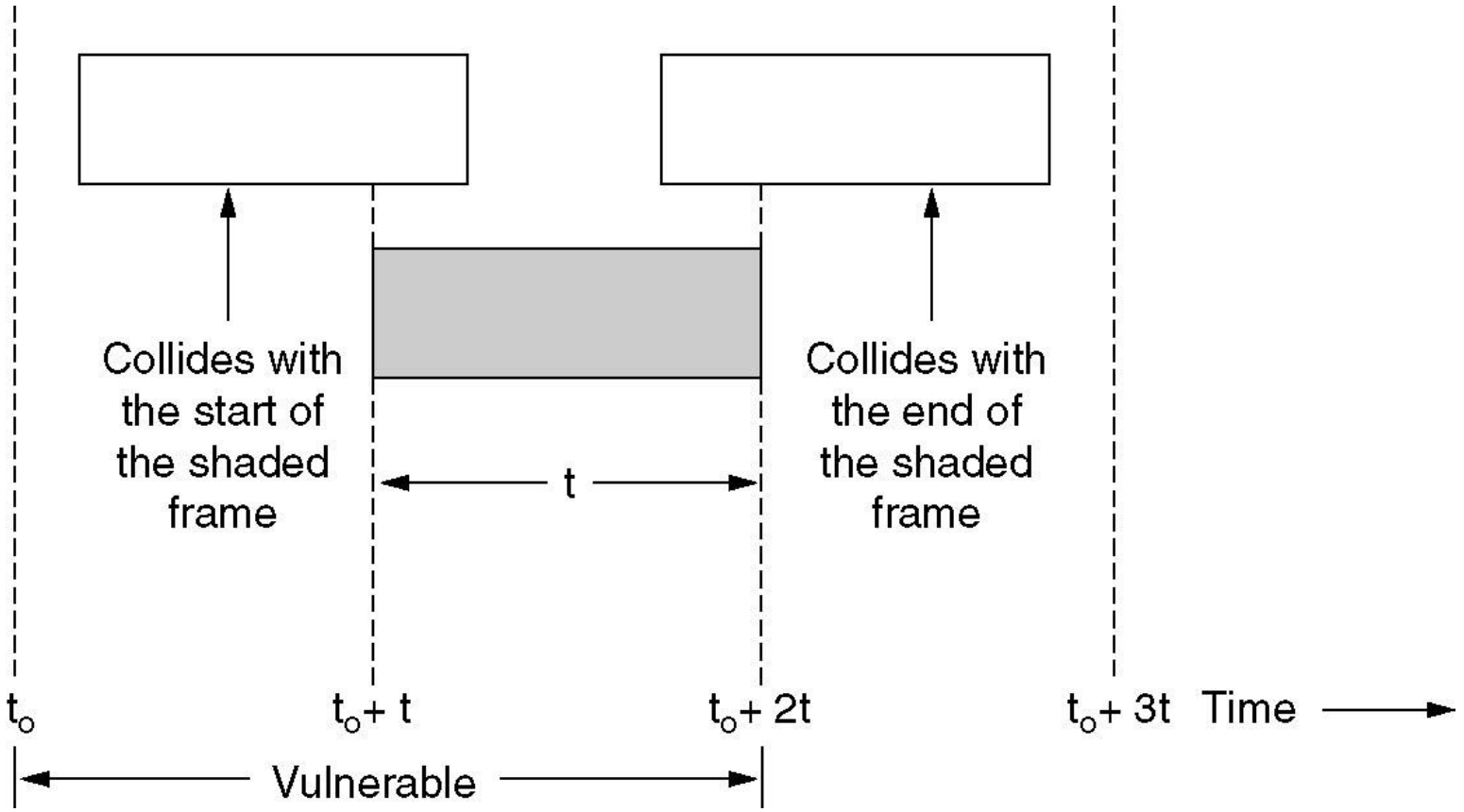
- ALOHA
- Carrier Sense Multiple Access Protocols
- Collision-Free Protocols
- Limited-Contention Protocols
- Wavelength Division Multiple Access Protocols
- Wireless LAN Protocols

Pure ALOHA



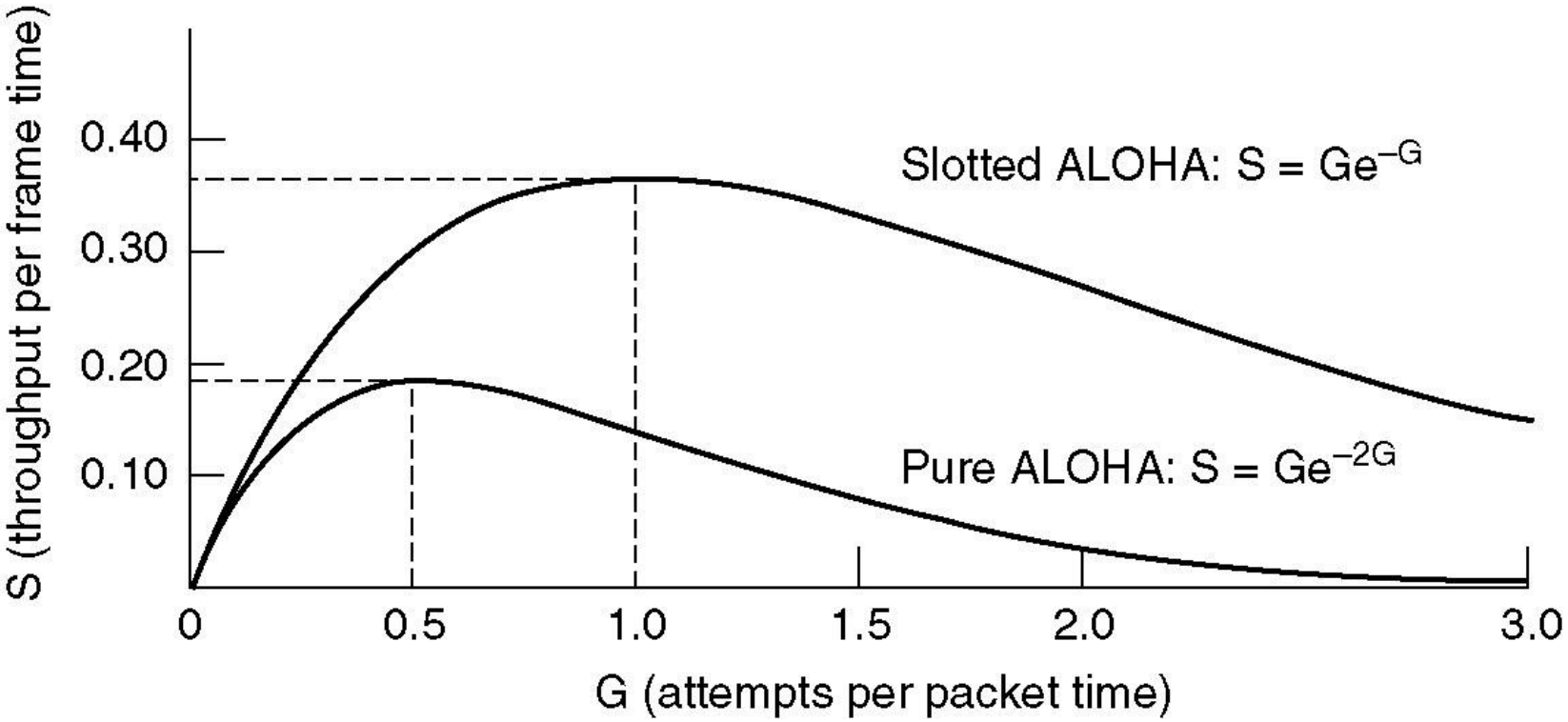
In pure ALOHA, frames are transmitted at completely arbitrary times.

Pure ALOHA (2)



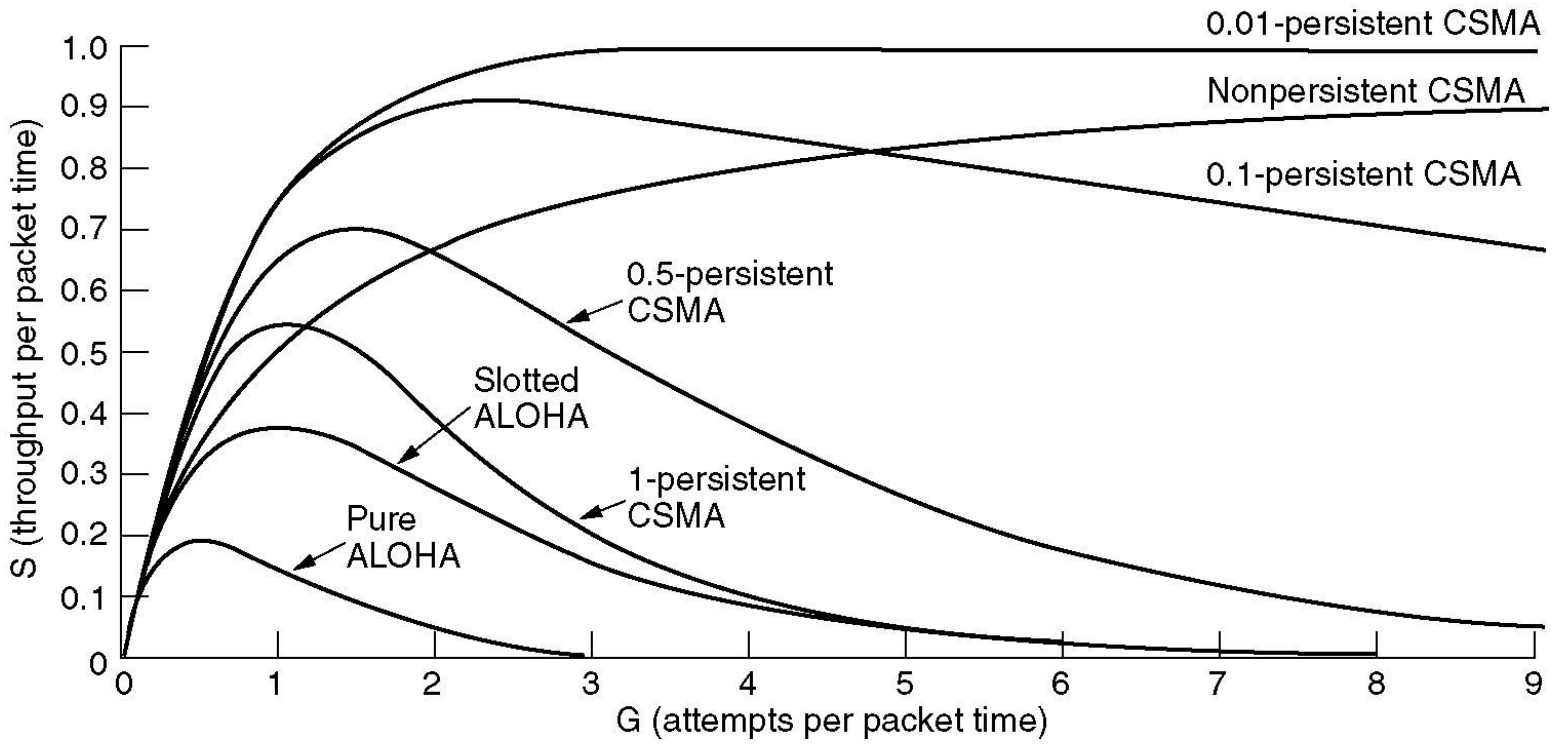
Vulnerable period for the shaded frame.

Pure ALOHA (3)



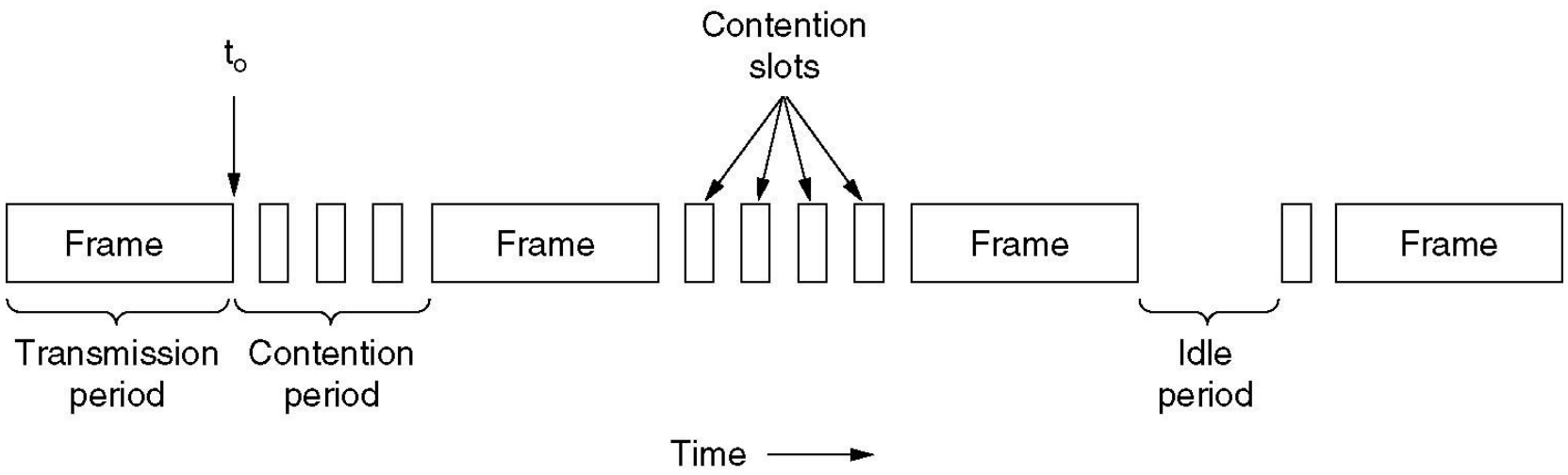
Throughput versus offered traffic for ALOHA systems.

Persistent and Nonpersistent CSMA



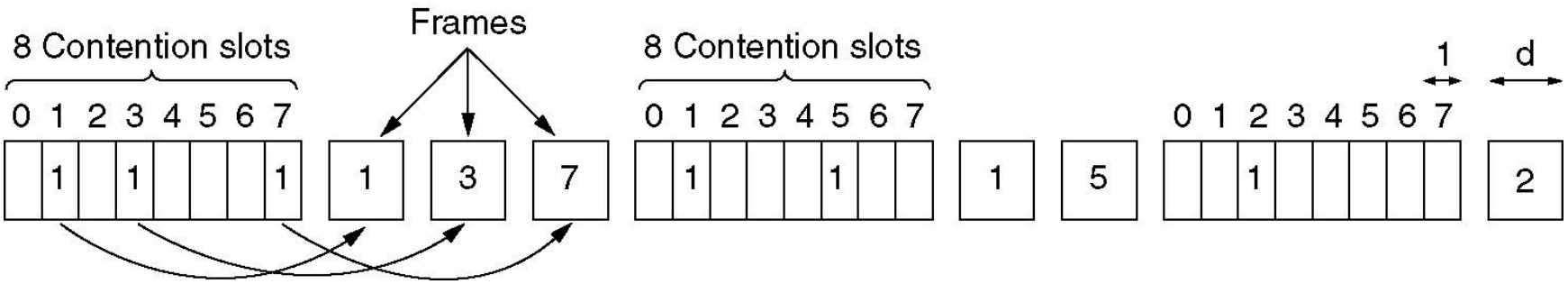
Comparison of the channel utilization versus load for various random access protocols.

CSMA with Collision Detection



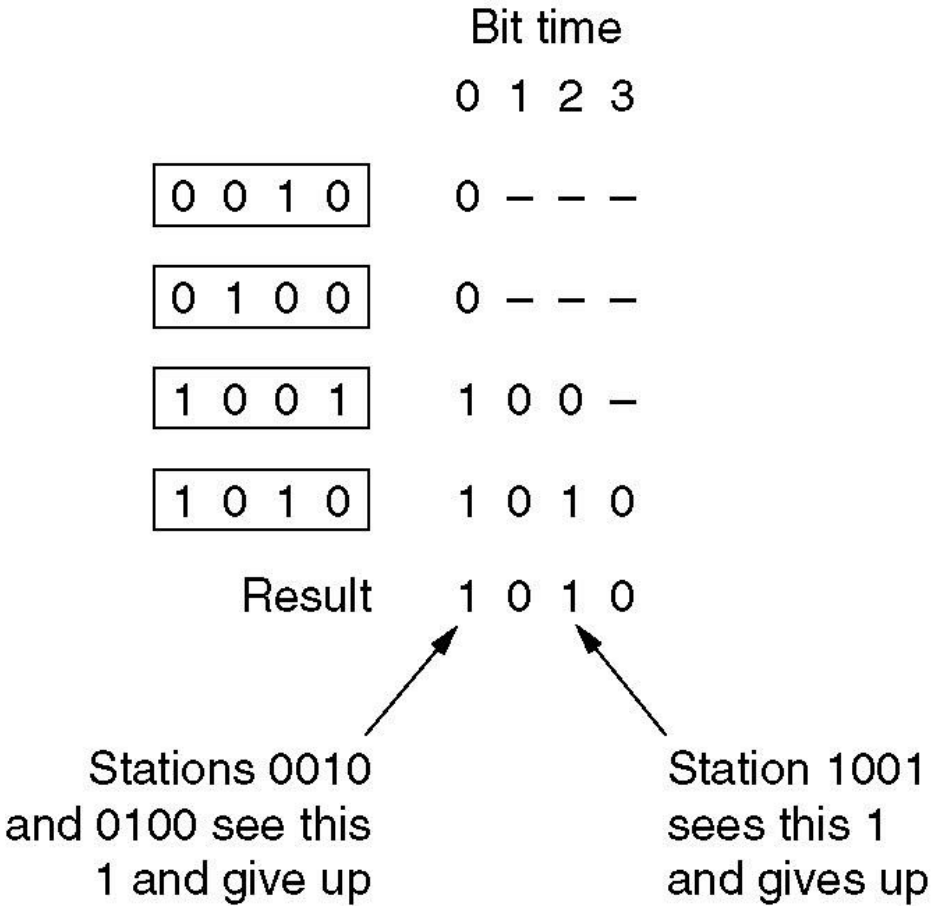
CSMA/CD can be in one of three states: contention, transmission, or idle.

Collision-Free Protocols



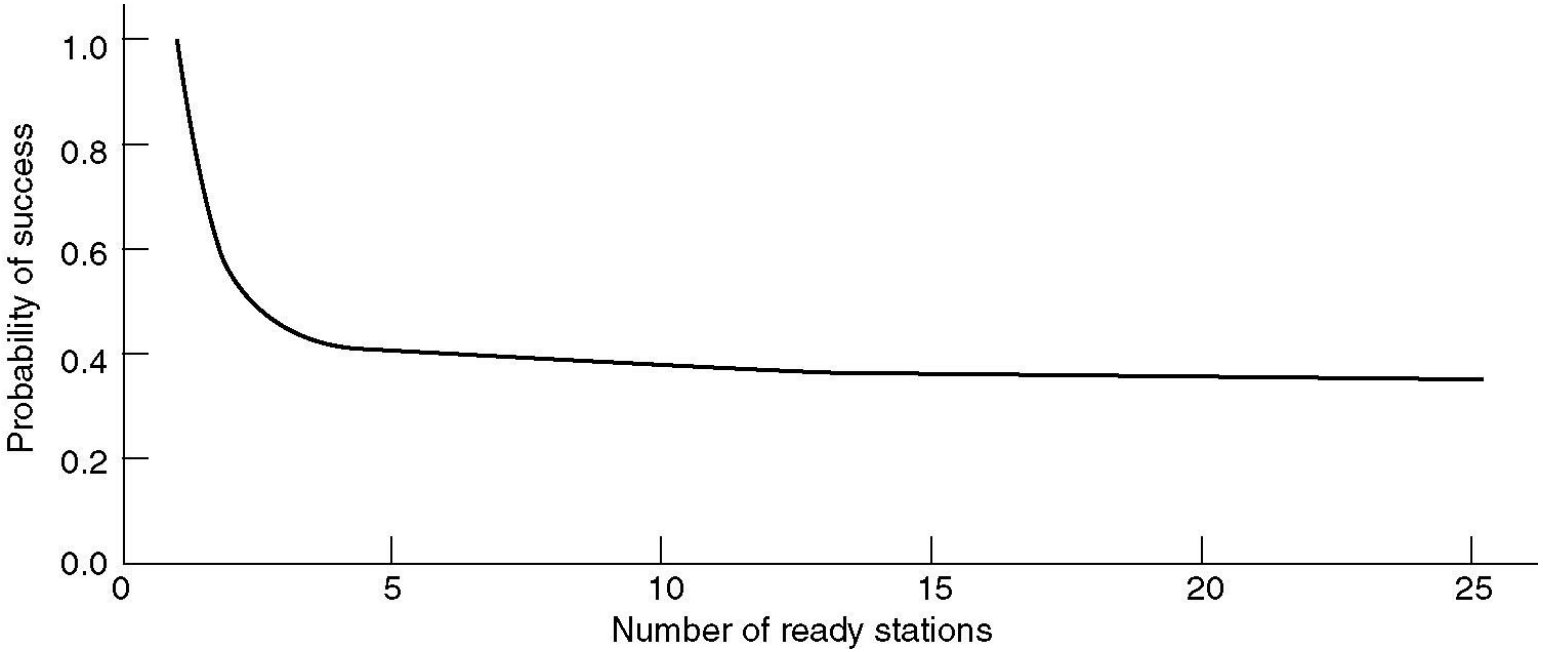
The basic bit-map protocol.

Collision-Free Protocols (2)



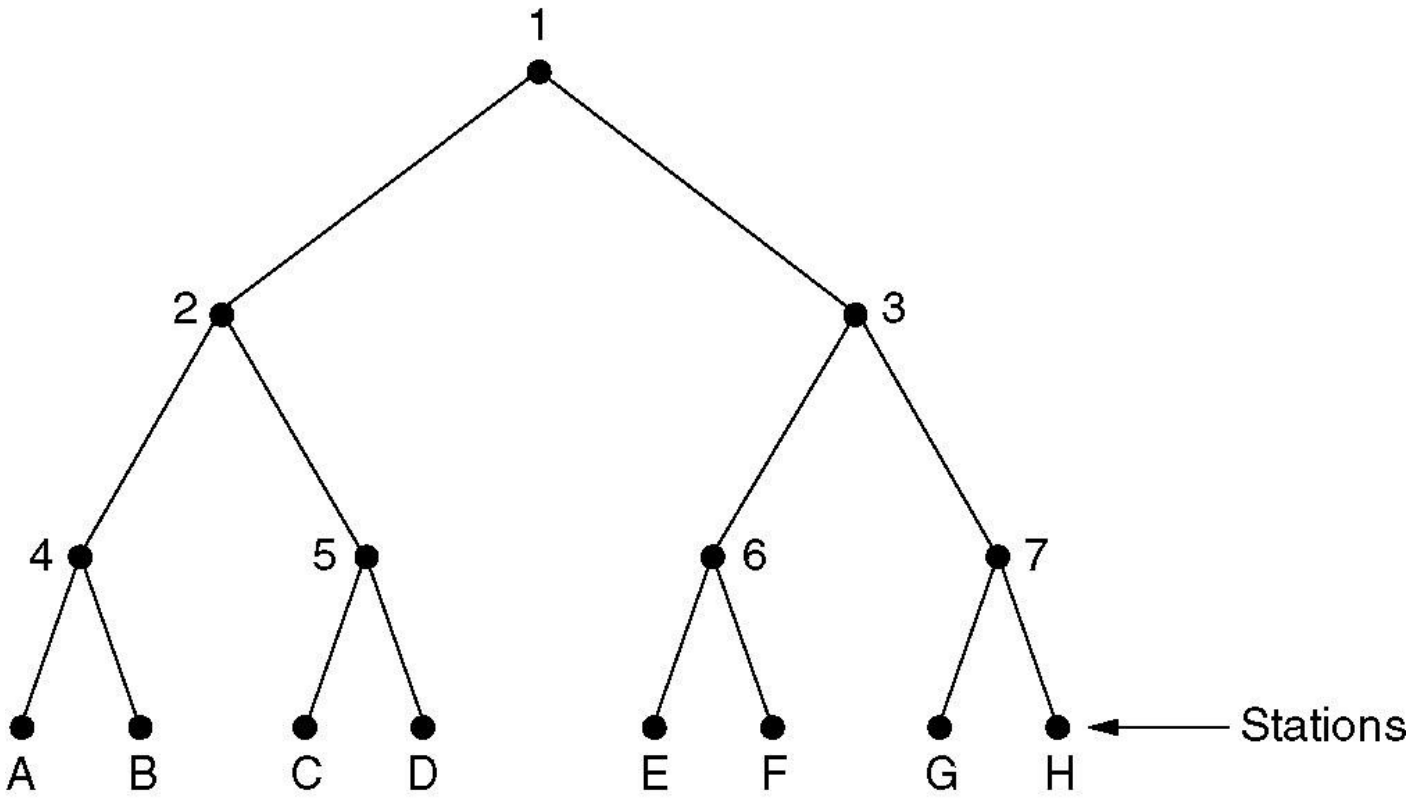
The binary countdown protocol. A dash indicates silence.

Limited-Contention Protocols



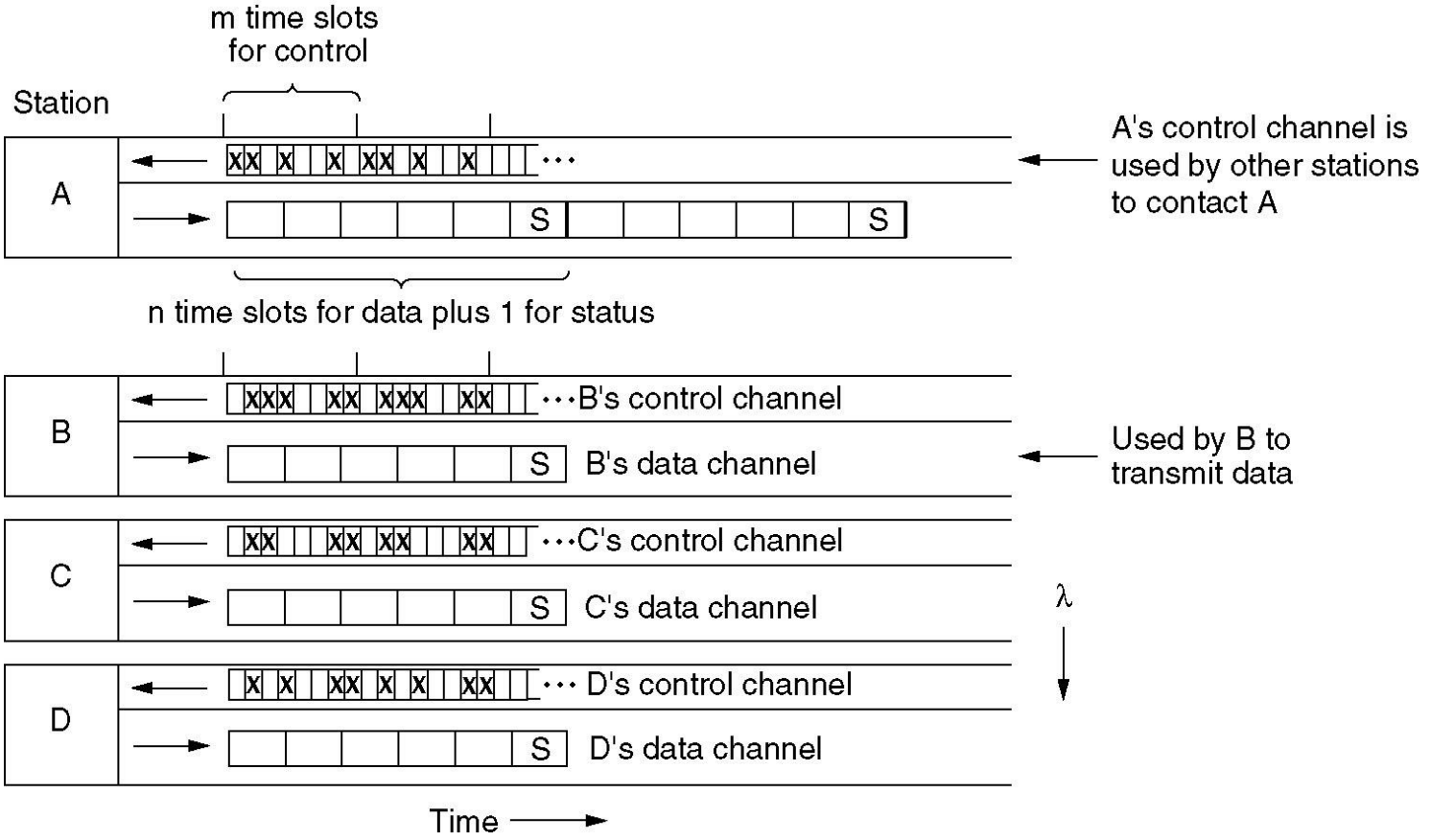
Acquisition probability for a symmetric contention channel.

Adaptive Tree Walk Protocol



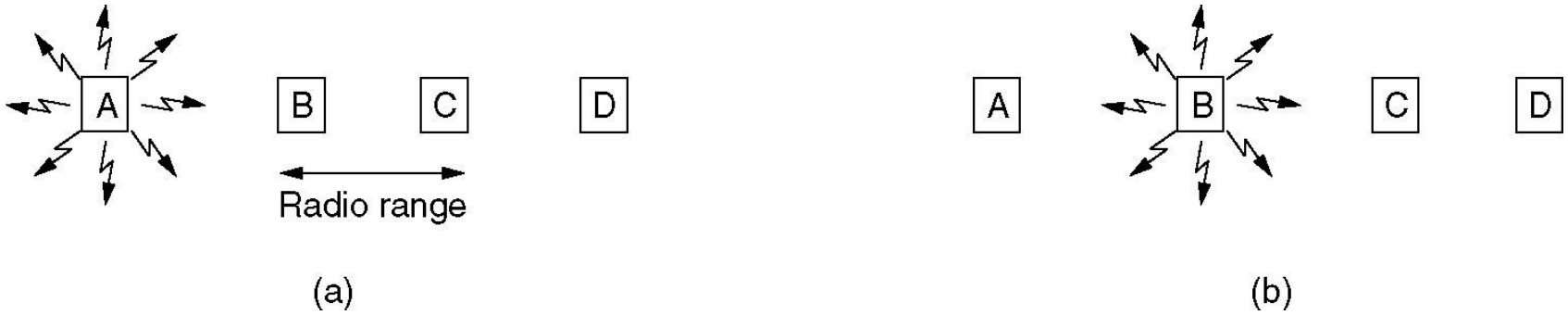
The tree for eight stations.

Wavelength Division Multiple Access Protocols



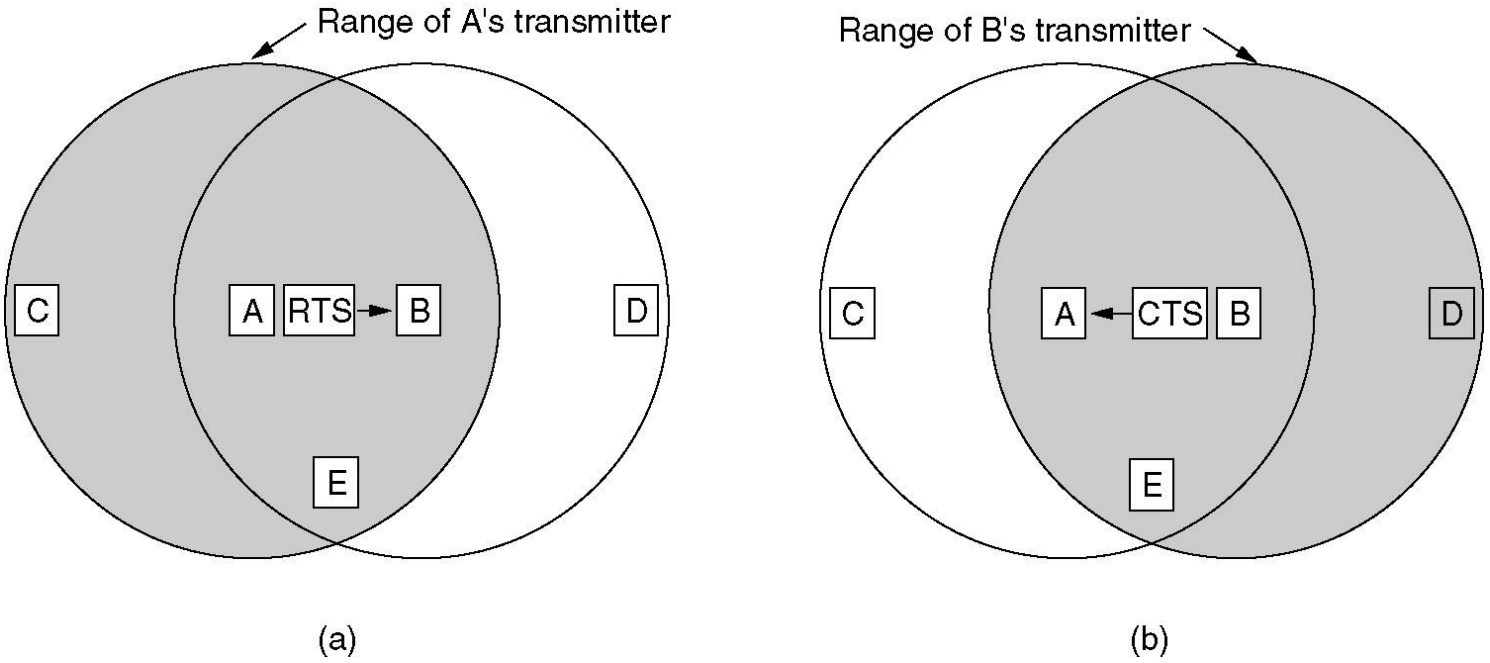
Wavelength division multiple access.

Wireless LAN Protocols



A wireless LAN. (a) A transmitting. (b) B transmitting.

Wireless LAN Protocols (2)



The MACA protocol. (a) A sending an RTS to B. (b) B responding with a CTS to A.

Ethernet

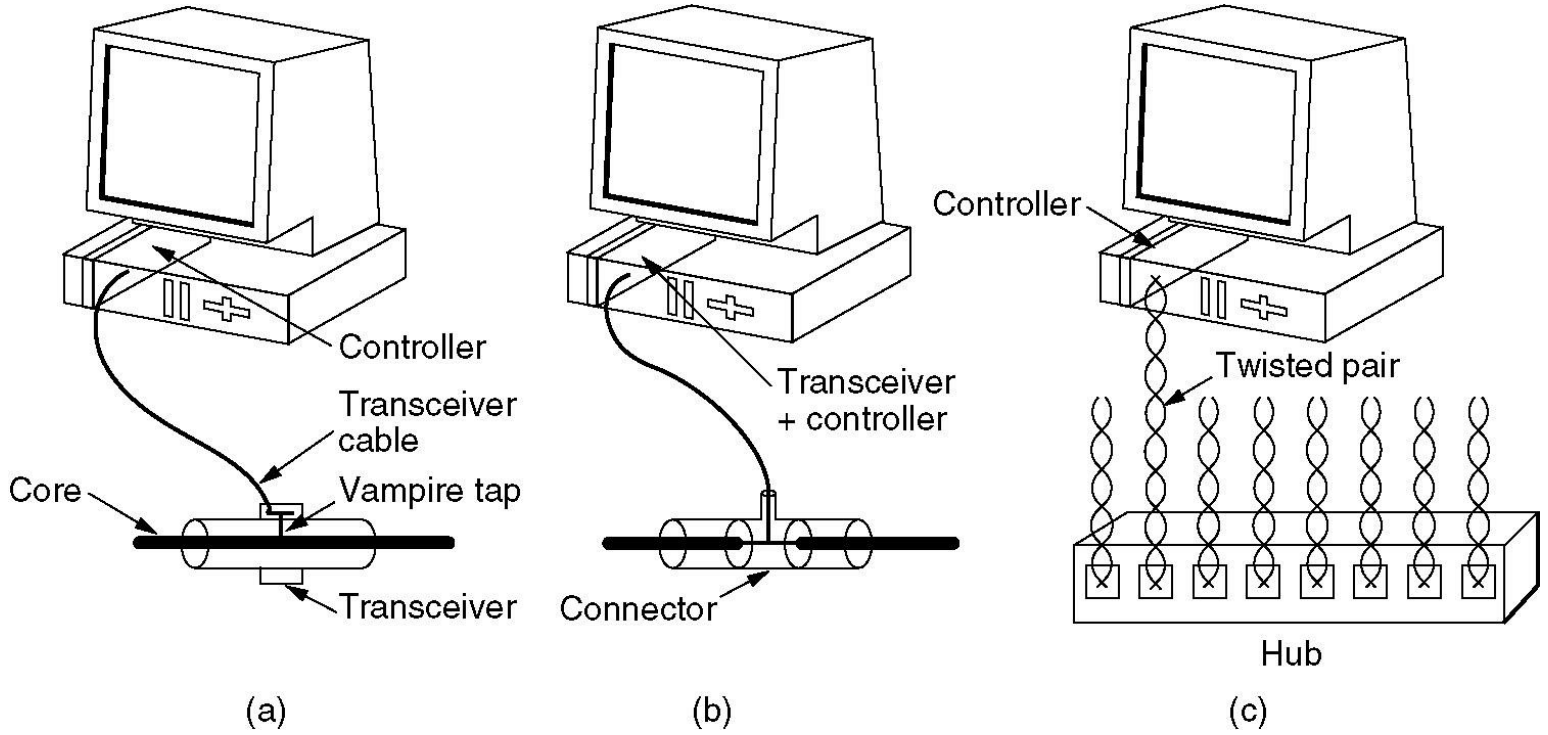
- Ethernet Cabling
- Manchester Encoding
- The Ethernet MAC Sublayer Protocol
- The Binary Exponential Backoff Algorithm
- Ethernet Performance
- Switched Ethernet
- Fast Ethernet
- Gigabit Ethernet
- IEEE 802.2: Logical Link Control
- Retrospective on Ethernet

Ethernet Cabling

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

The most common kinds of Ethernet cabling.

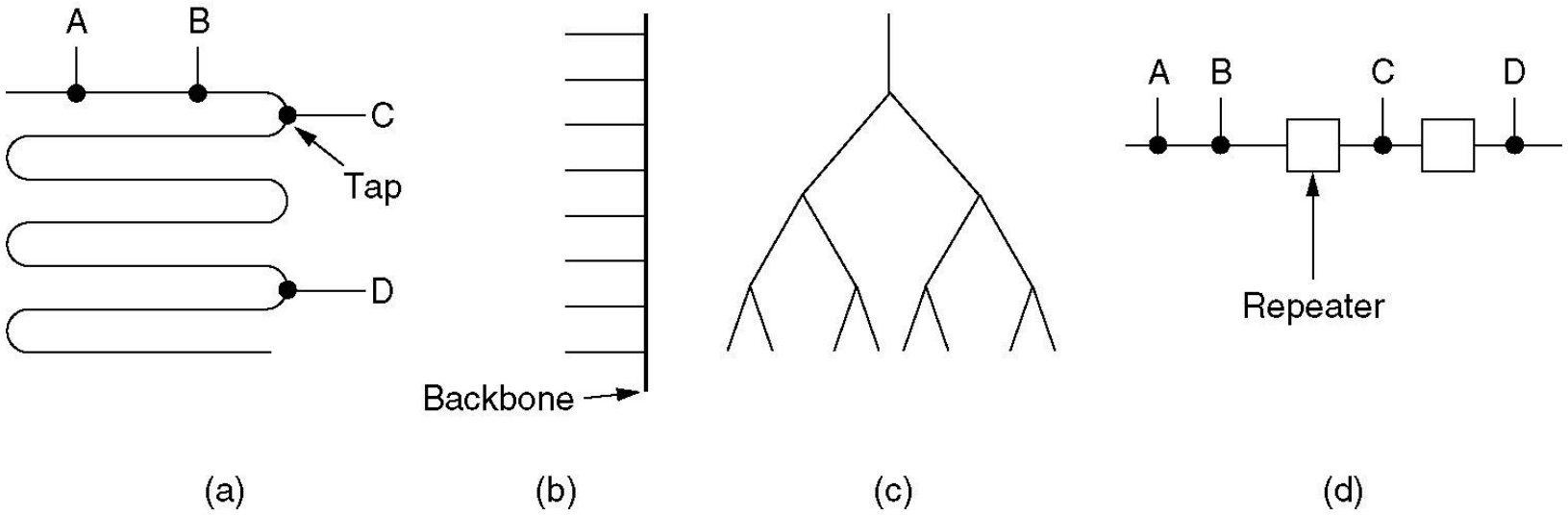
Ethernet Cabling (2)



Three kinds of Ethernet cabling.

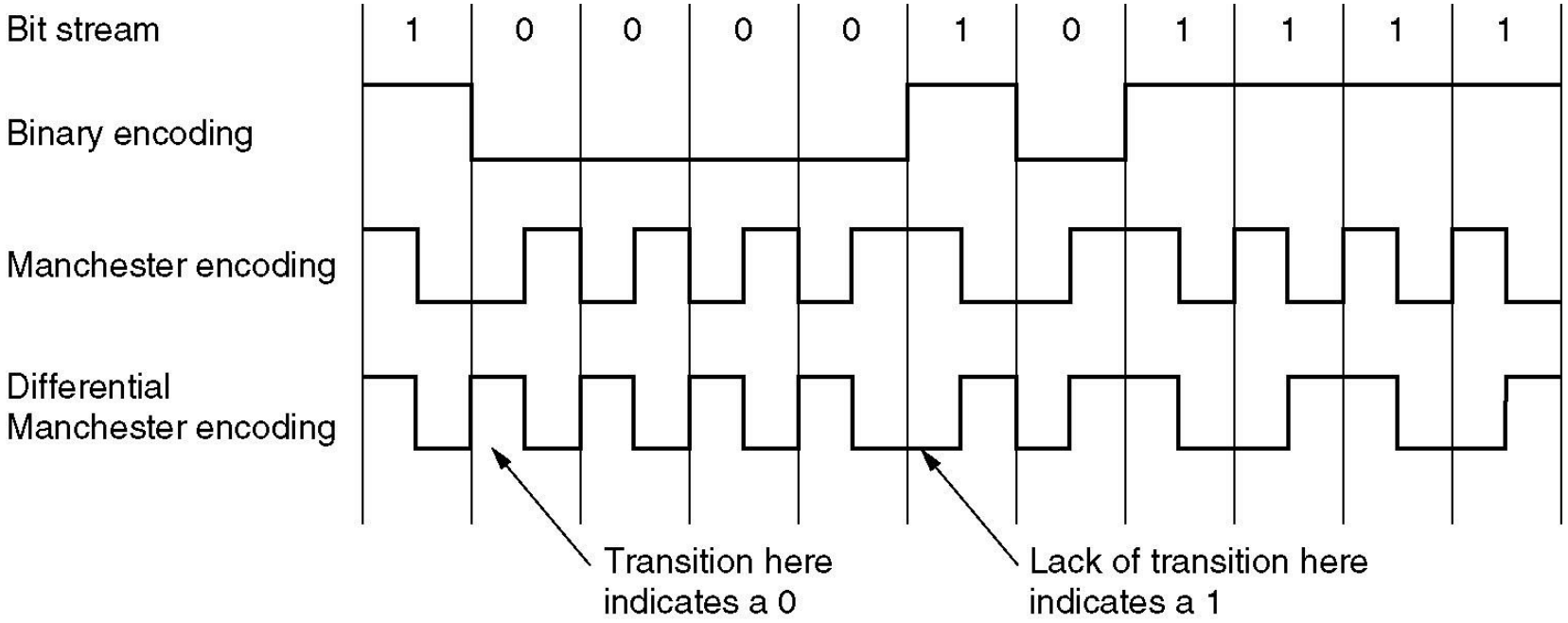
(a) 10Base5, (b) 10Base2, (c) 10Base-T.

Ethernet Cabling (3)



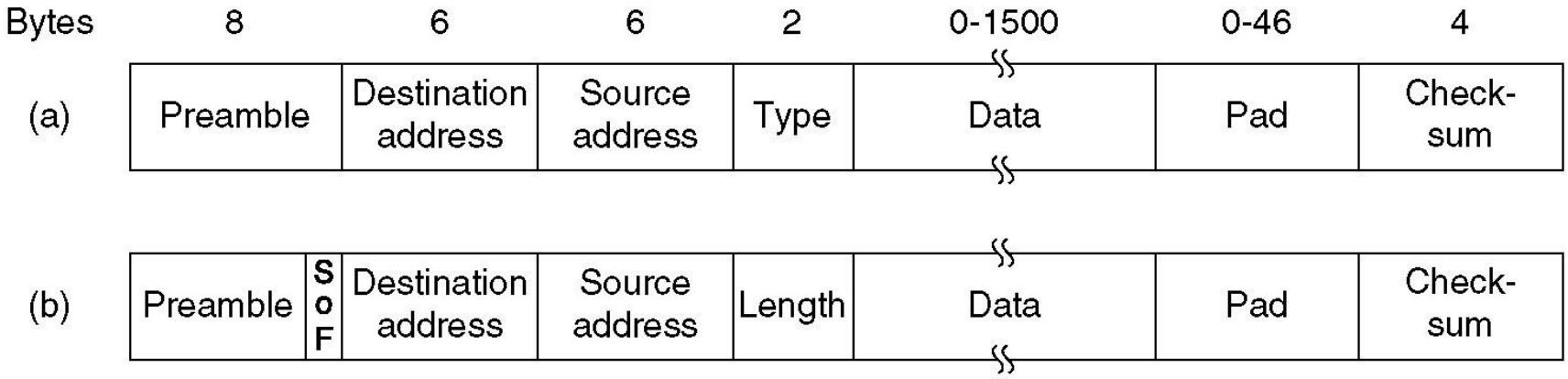
Cable topologies. (a) Linear, (b) Spine, (c) Tree, (d) Segmented.

Ethernet Cabling (4)



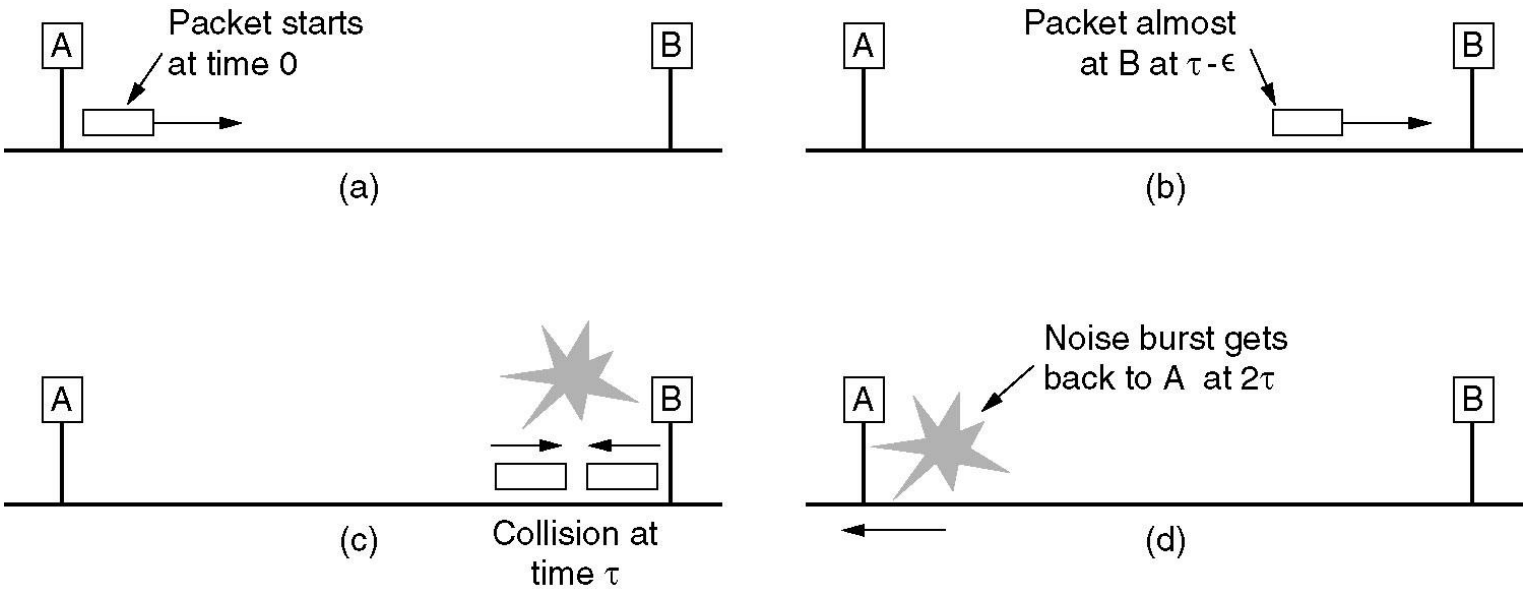
(a) Binary encoding, (b) Manchester encoding,
(c) Differential Manchester encoding.

Ethernet MAC Sublayer Protocol



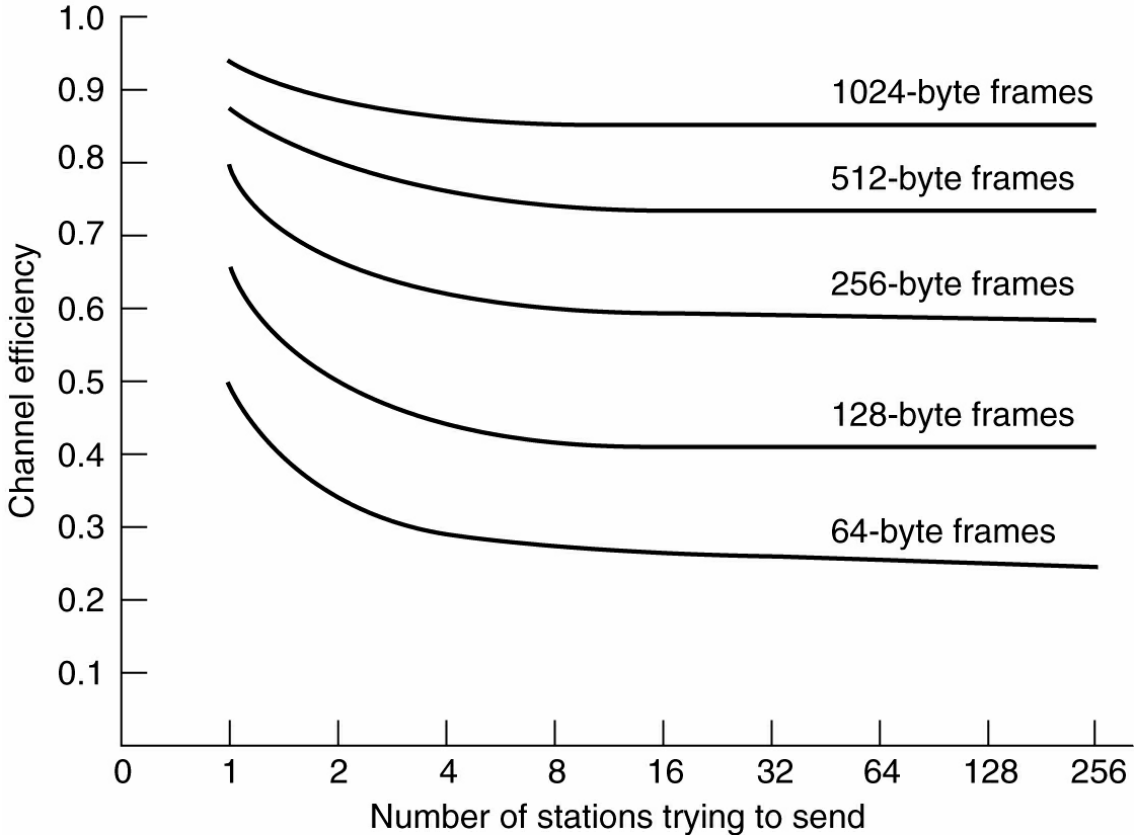
Frame formats. (a) DIX Ethernet, (b) IEEE 802.3.

Ethernet MAC Sublayer Protocol (2)



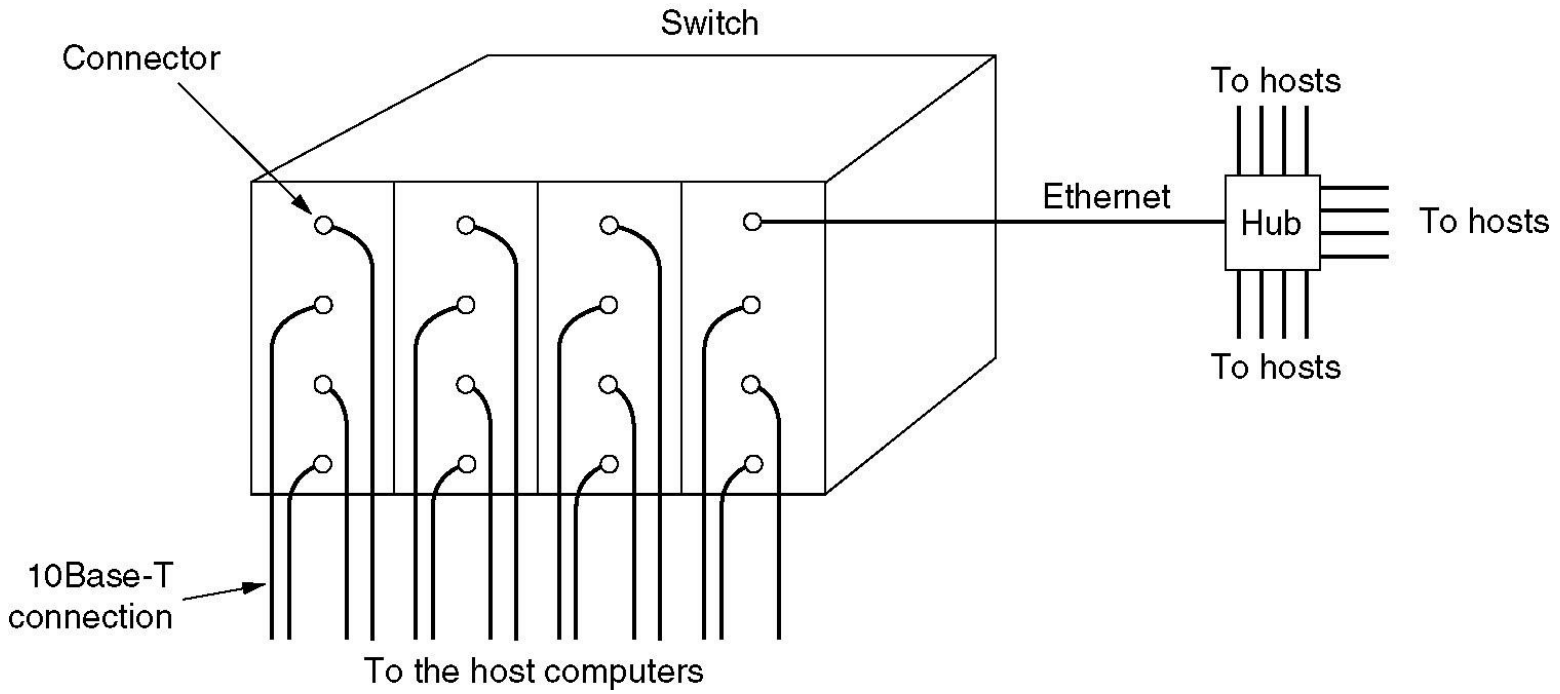
Collision detection can take as long as 2τ.

Ethernet Performance



Efficiency of Ethernet at 10 Mbps with 512-bit slot times.

Switched Ethernet



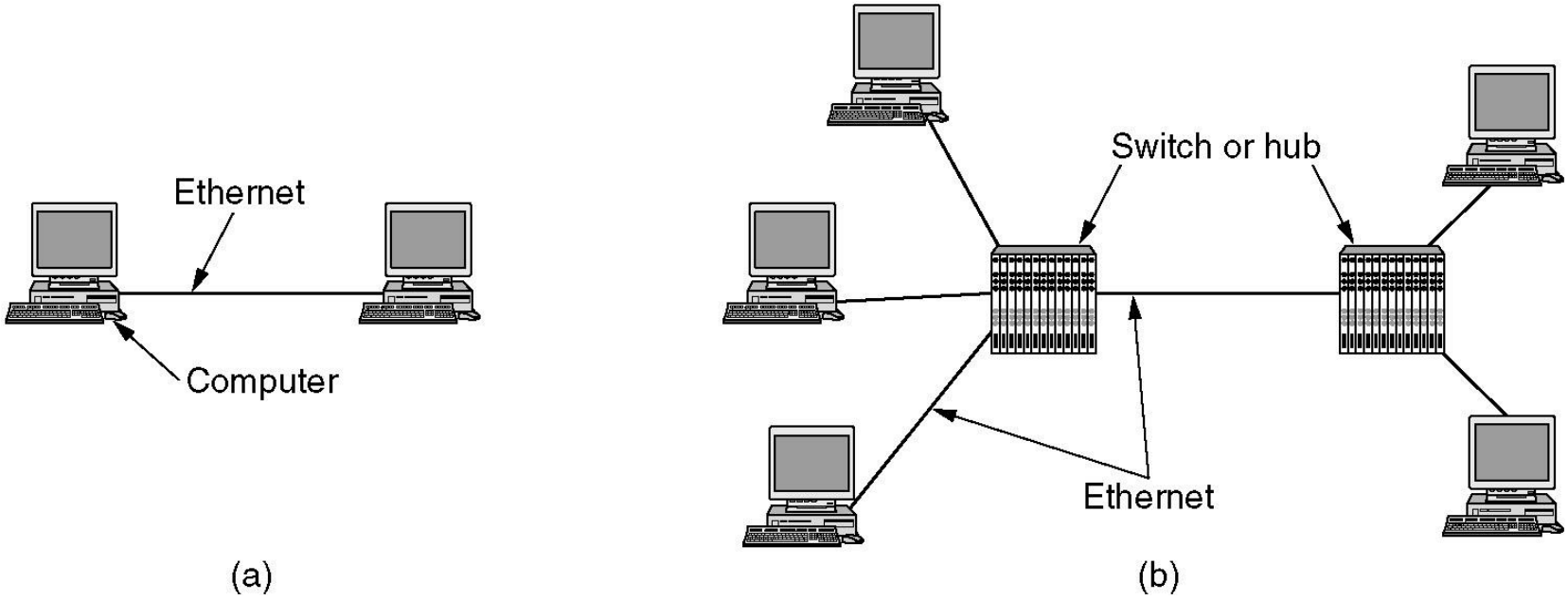
A simple example of switched Ethernet.

Fast Ethernet

Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

The original fast Ethernet cabling.

Gigabit Ethernet



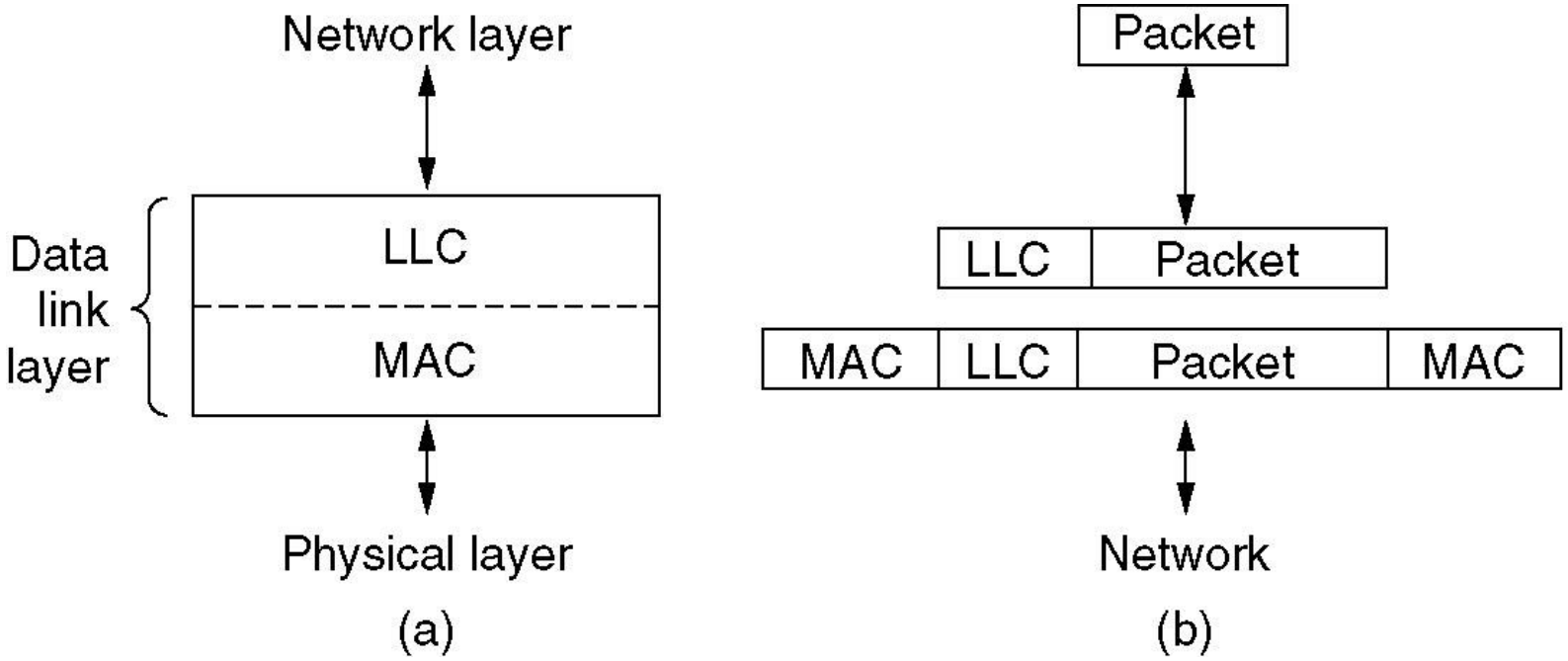
(a) A two-station Ethernet. (b) A multistation Ethernet.

Gigabit Ethernet (2)

Name	Cable	Max. segment	Advantages
1000Base-SX	Fiber optics	550 m	Multimode fiber (50, 62.5 microns)
1000Base-LX	Fiber optics	5000 m	Single (10 μ) or multimode (50, 62.5 μ)
1000Base-CX	2 Pairs of STP	25 m	Shielded twisted pair
1000Base-T	4 Pairs of UTP	100 m	Standard category 5 UTP

Gigabit Ethernet cabling.

IEEE 802.2: Logical Link Control

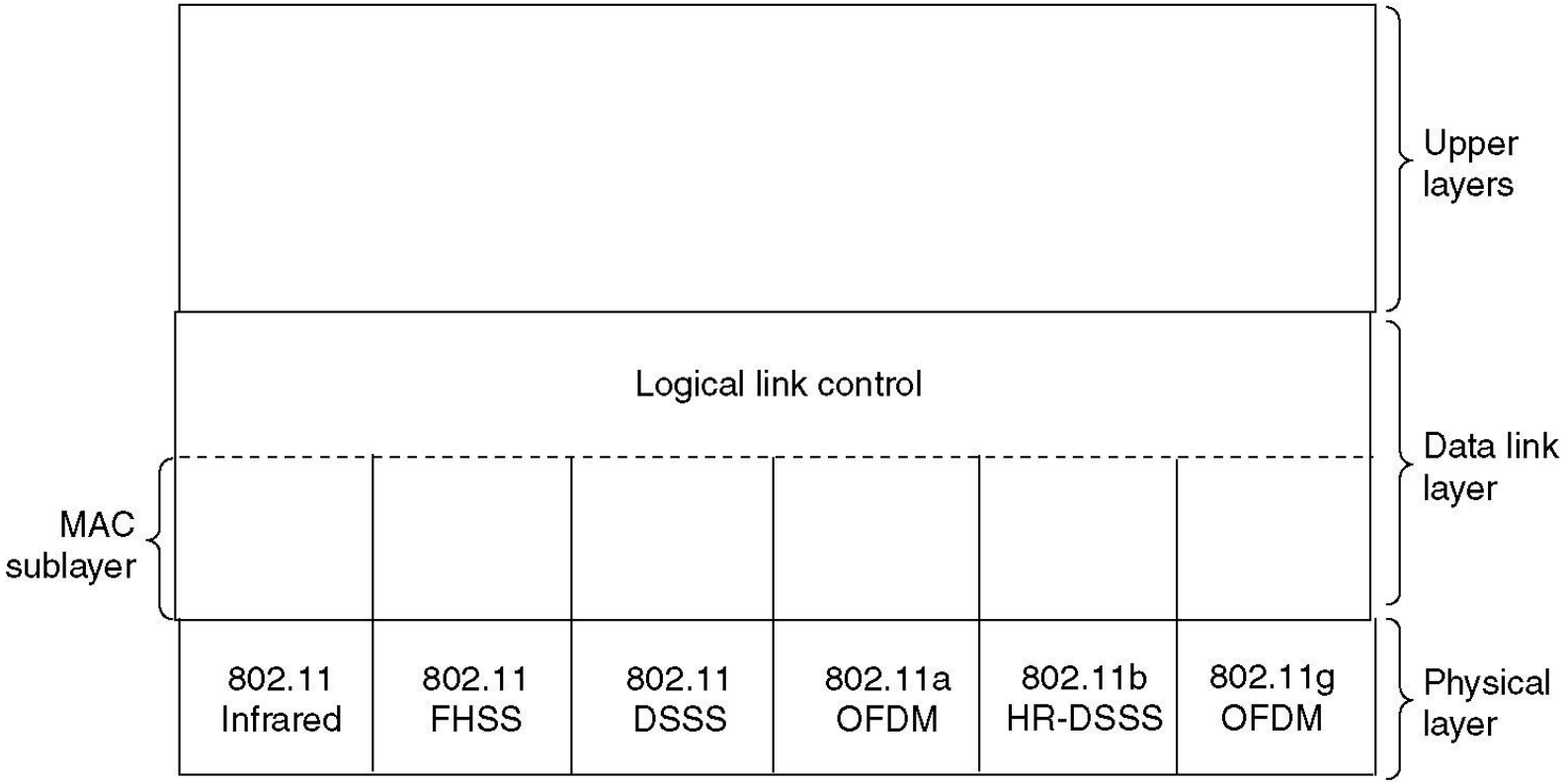


(a) Position of LLC. (b) Protocol formats.

Wireless LANs

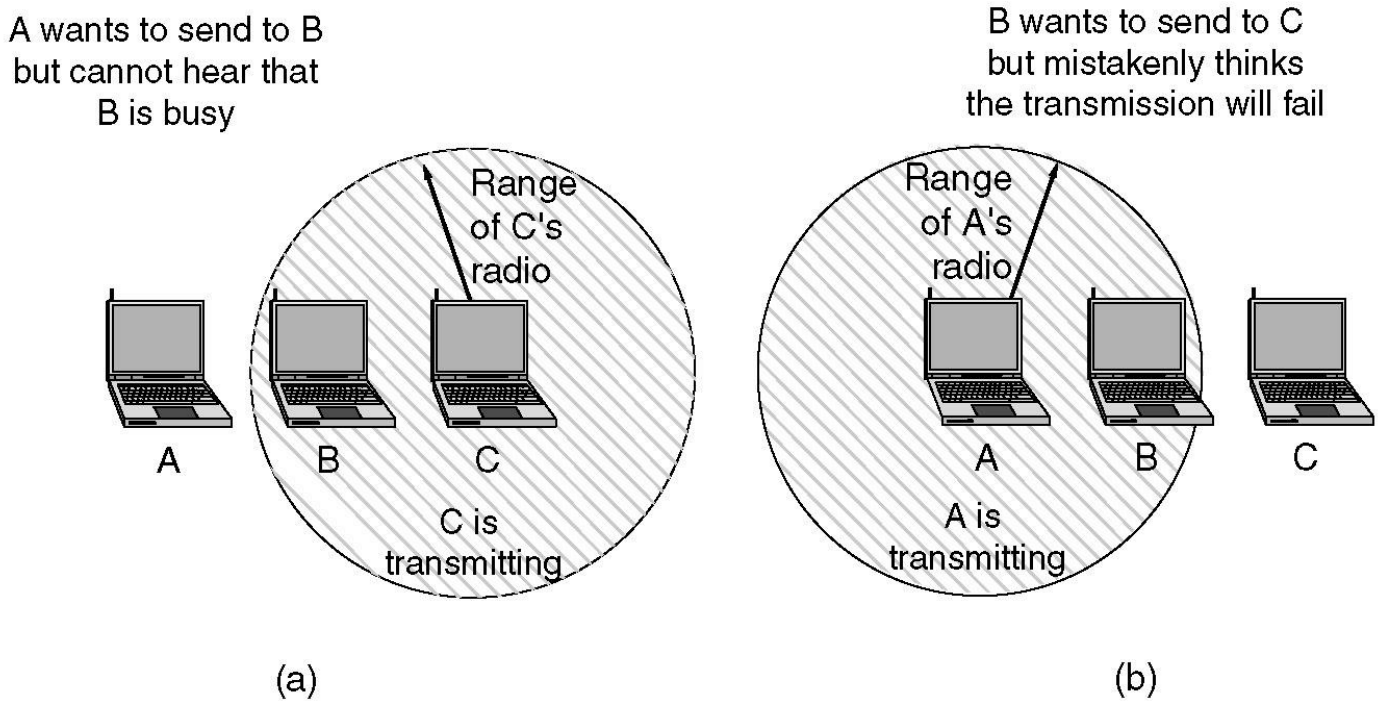
- The 802.11 Protocol Stack
- The 802.11 Physical Layer
- The 802.11 MAC Sublayer Protocol
- The 802.11 Frame Structure
- Services

The 802.11 Protocol Stack



Part of the 802.11 protocol stack.

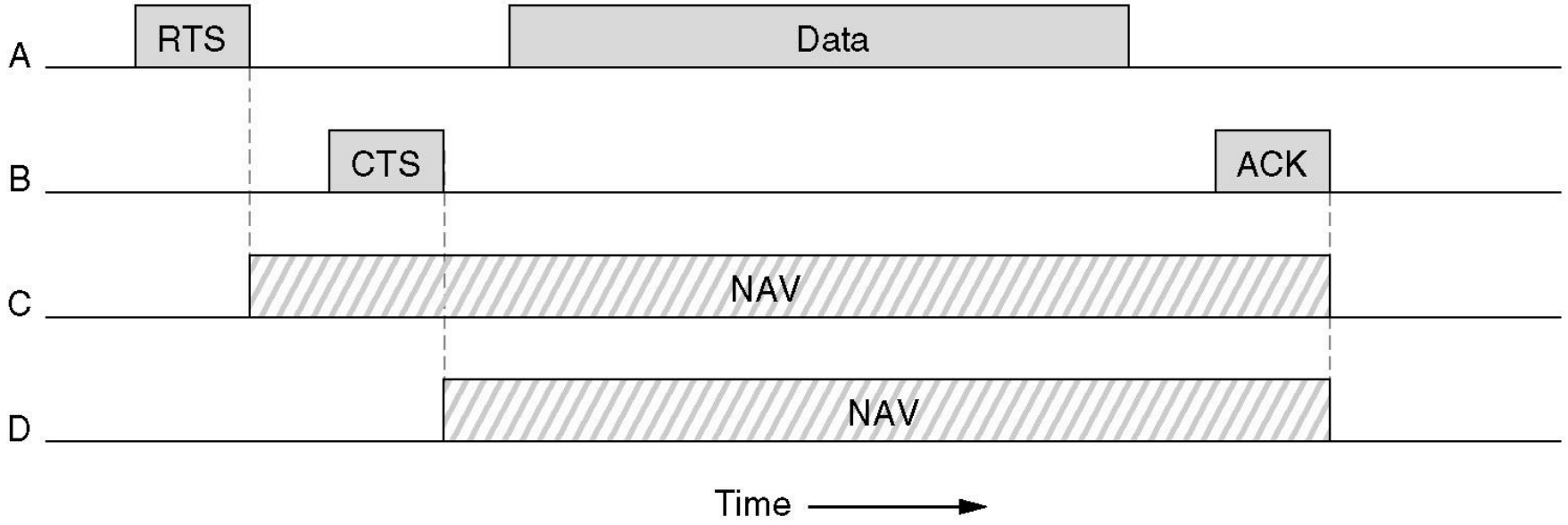
The 802.11 MAC Sublayer Protocol



(a) The hidden station problem.

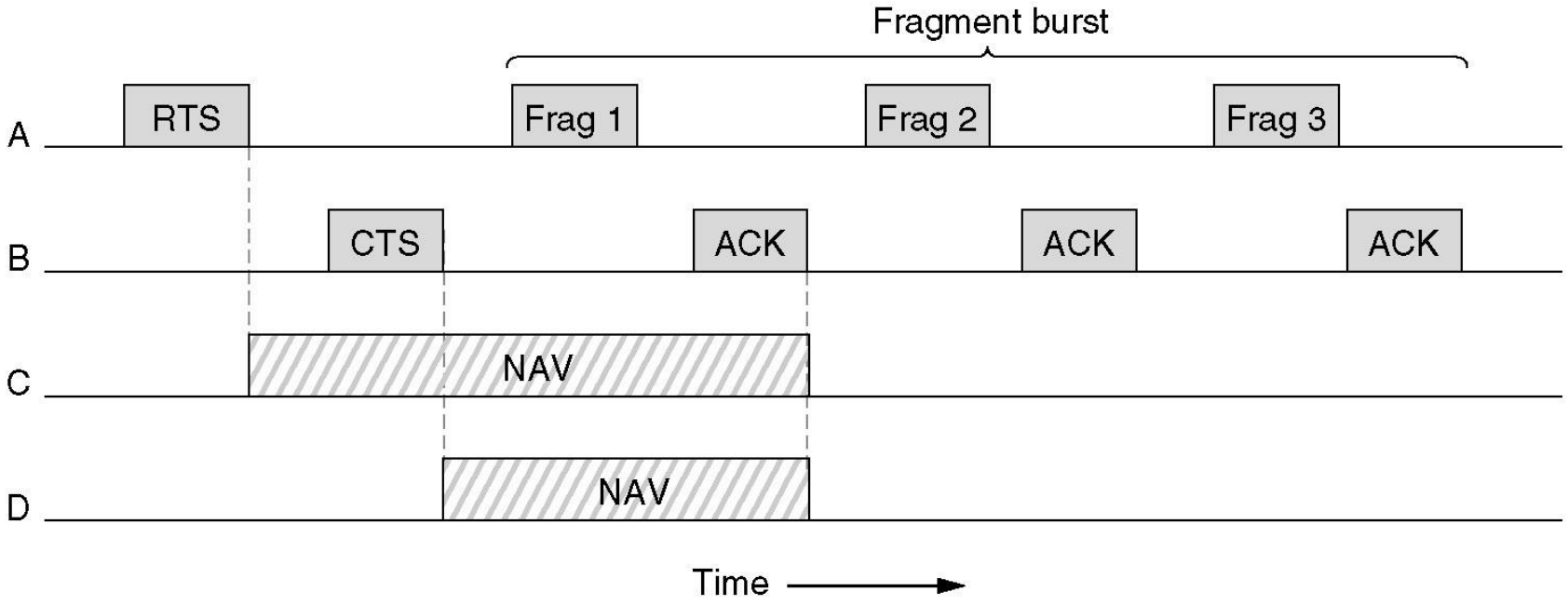
(b) The exposed station problem.

The 802.11 MAC Sublayer Protocol (2)



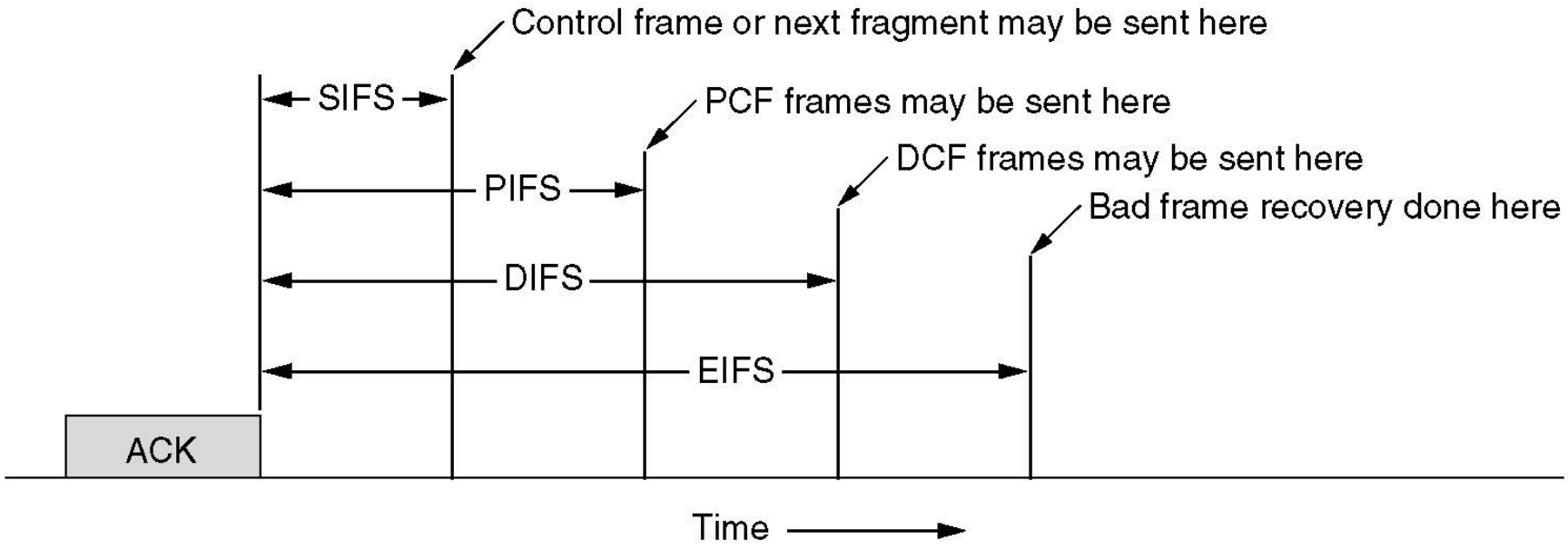
The use of virtual channel sensing using CSMA/CA.

The 802.11 MAC Sublayer Protocol (3)



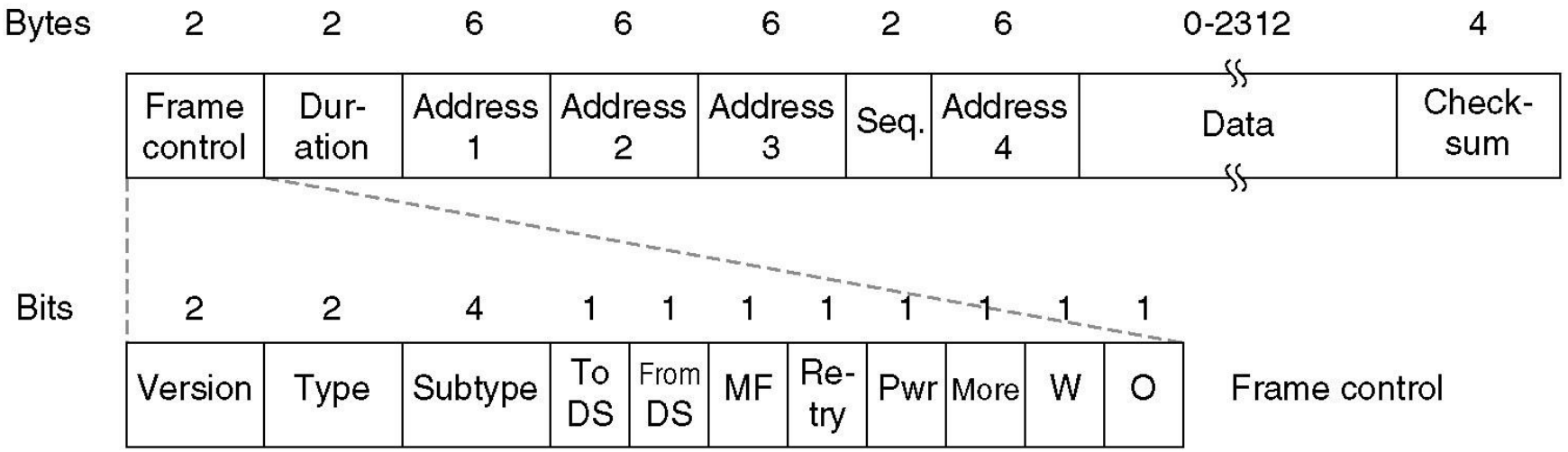
A fragment burst.

The 802.11 MAC Sublayer Protocol (4)



Interframe spacing in 802.11.

The 802.11 Frame Structure



The 802.11 data frame.

802.11 Services

Distribution Services

- Association
- Disassociation
- Reassociation
- Distribution
- Integration

802.11 Services

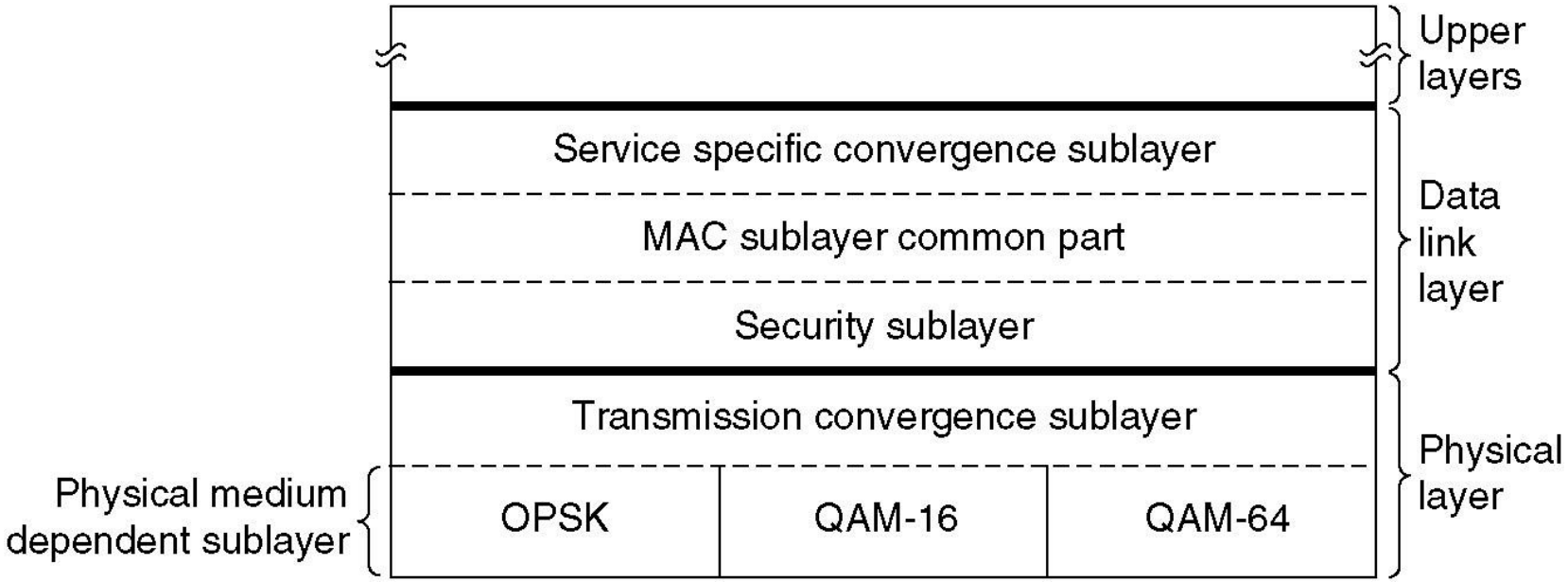
Intracell Services

- Authentication
- Deauthentication
- Privacy
- Data Delivery

Broadband Wireless

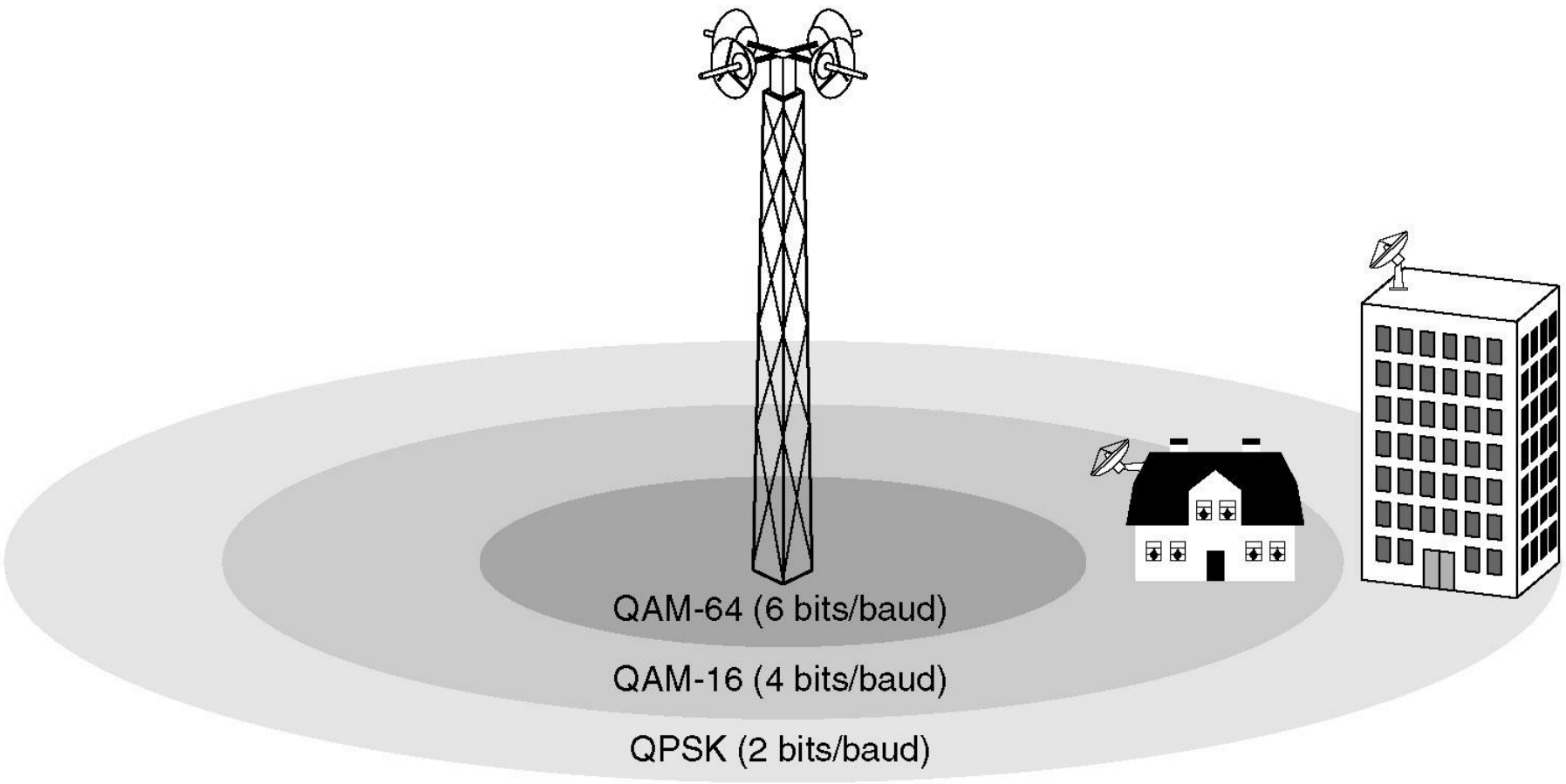
- Comparison of 802.11 and 802.16
- The 802.16 Protocol Stack
- The 802.16 Physical Layer
- The 802.16 MAC Sublayer Protocol
- The 802.16 Frame Structure

The 802.16 Protocol Stack



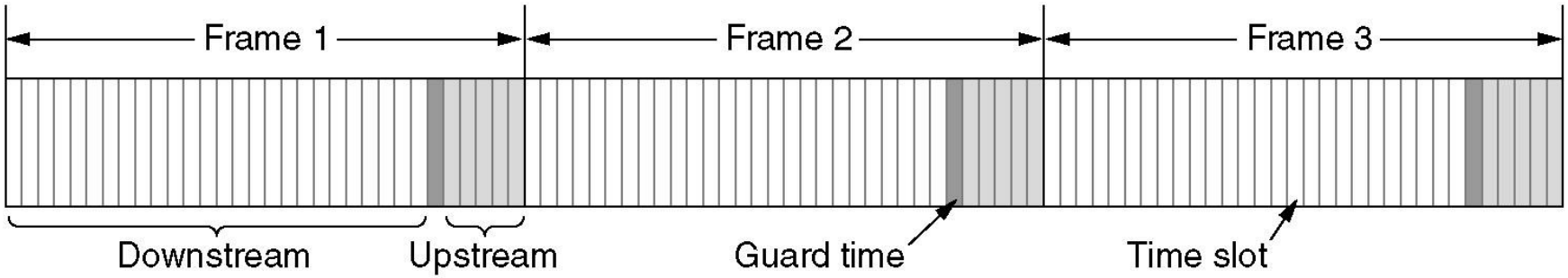
The 802.16 Protocol Stack.

The 802.16 Physical Layer



The 802.16 transmission environment.

The 802.16 Physical Layer (2)



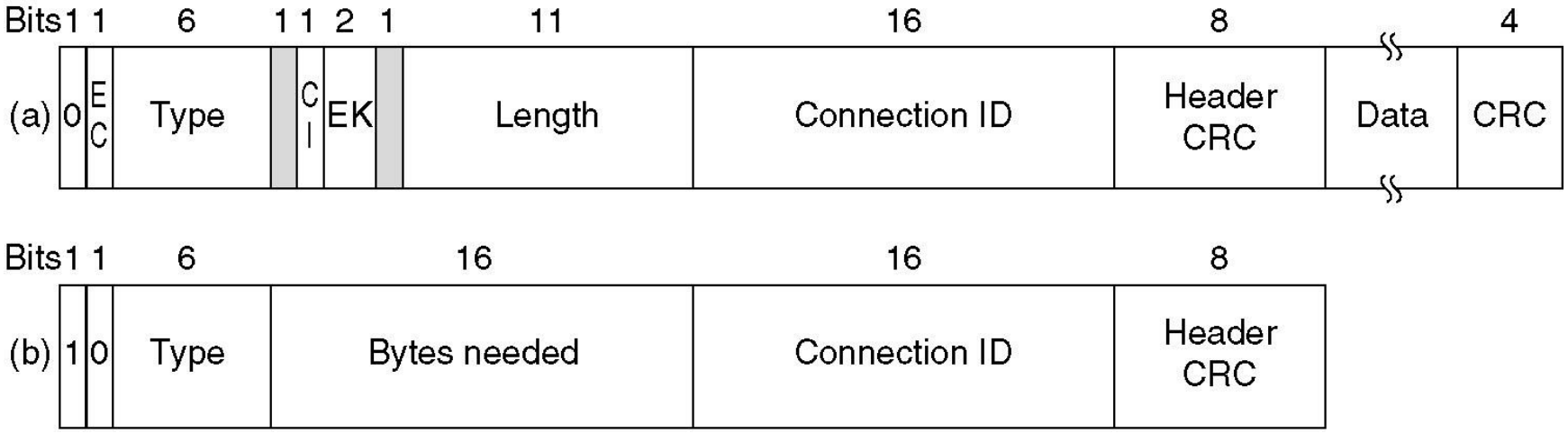
Frames and time slots for time division duplexing.

The 802.16 MAC Sublayer Protocol

Service Classes

- Constant bit rate service
- Real-time variable bit rate service
- Non-real-time variable bit rate service
- Best efforts service

The 802.16 Frame Structure

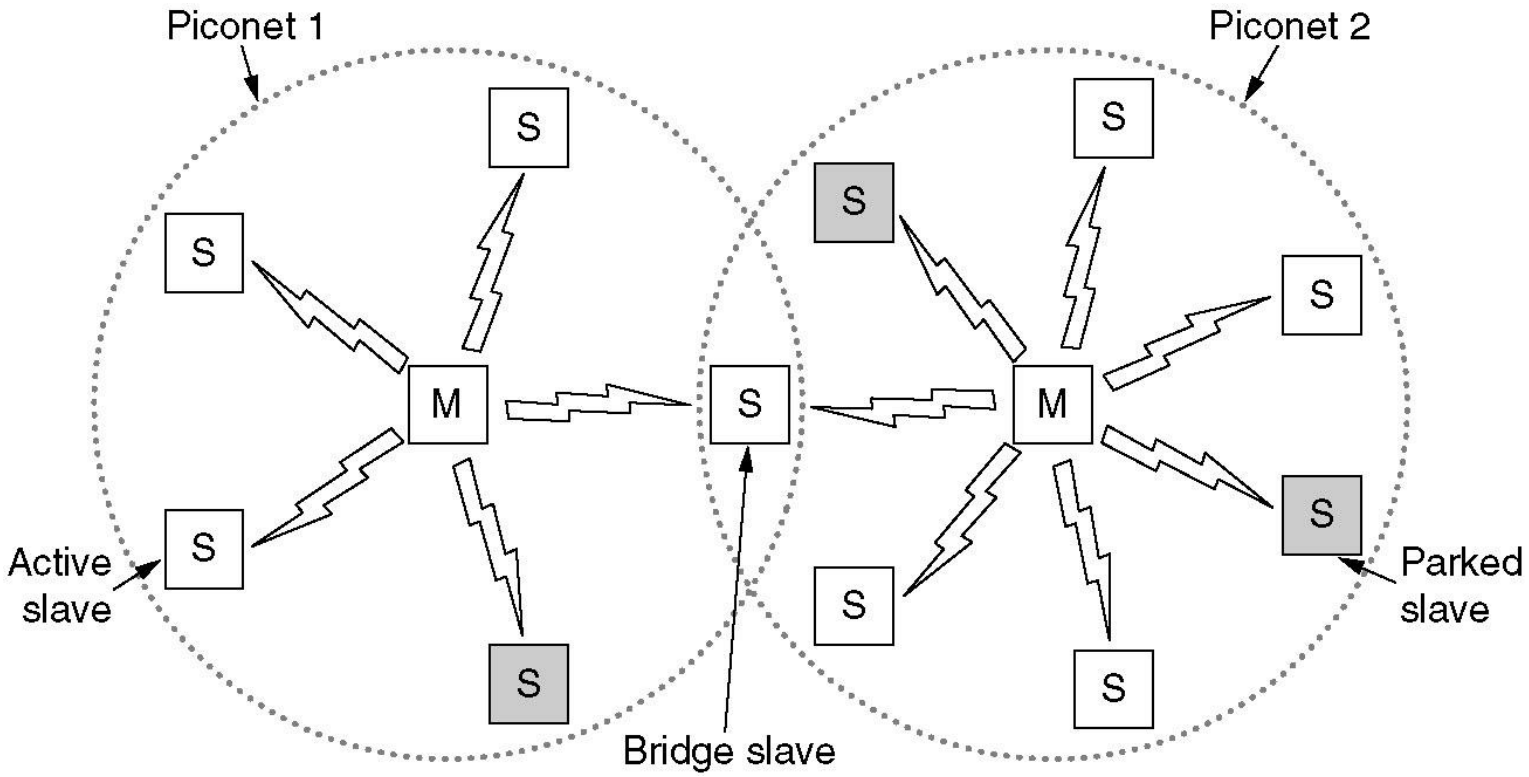


(a) A generic frame. (b) A bandwidth request frame.

Bluetooth

- Bluetooth Architecture
- Bluetooth Applications
- The Bluetooth Protocol Stack
- The Bluetooth Radio Layer
- The Bluetooth Baseband Layer
- The Bluetooth L2CAP Layer
- The Bluetooth Frame Structure

Bluetooth Architecture



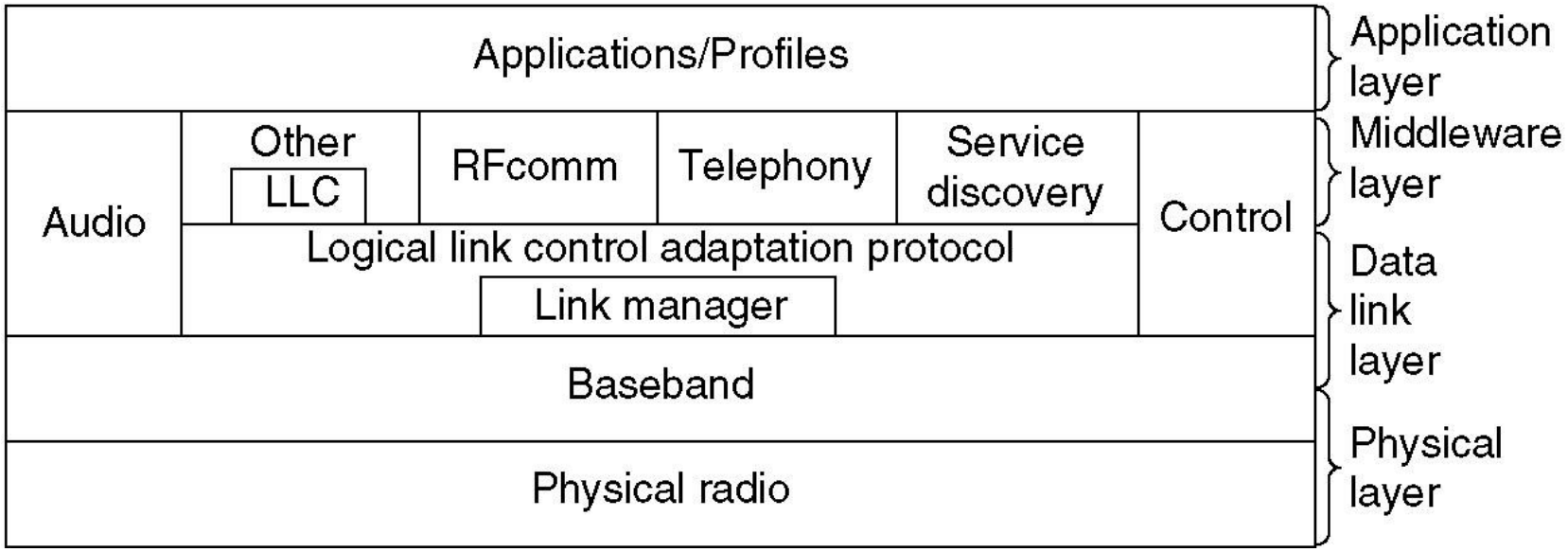
Two piconets can be connected to form a scatternet.

Bluetooth Applications

Name	Description
Generic access	Procedures for link management
Service discovery	Protocol for discovering offered services
Serial port	Replacement for a serial port cable
Generic object exchange	Defines client-server relationship for object movement
LAN access	Protocol between a mobile computer and a fixed LAN
Dial-up networking	Allows a notebook computer to call via a mobile phone
Fax	Allows a mobile fax machine to talk to a mobile phone
Cordless telephony	Connects a handset and its local base station
Intercom	Digital walkie-talkie
Headset	Intended for hands-free voice communication
Object push	Provides a way to exchange simple objects
File transfer	Provides a more general file transfer facility
Synchronization	Permits a PDA to synchronize with another computer

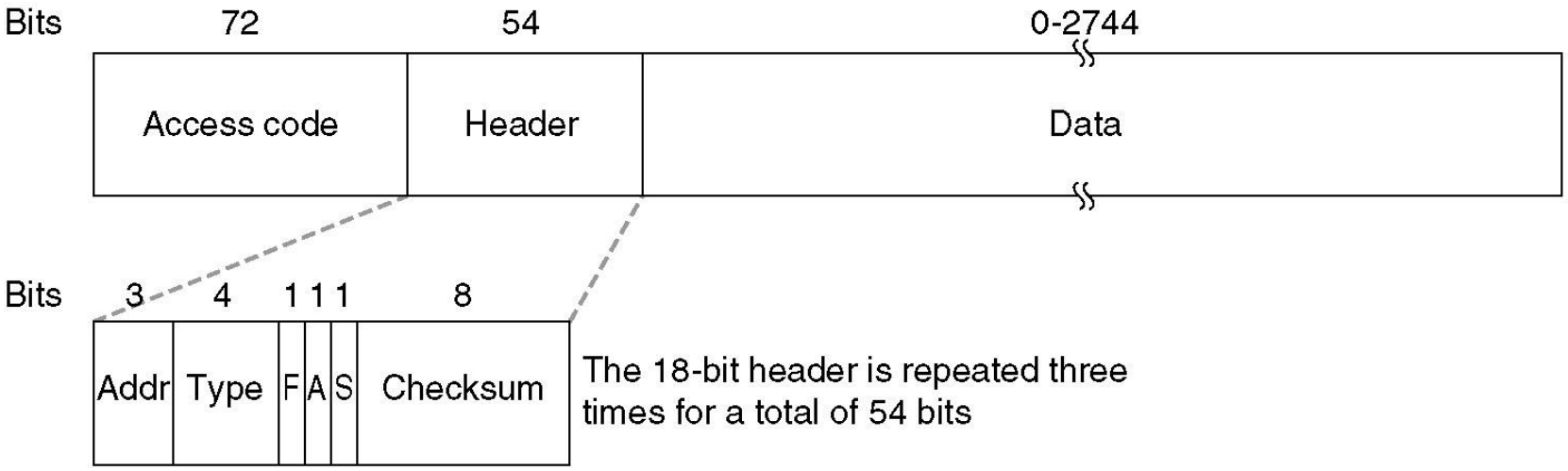
The Bluetooth profiles.

The Bluetooth Protocol Stack



The 802.15 version of the Bluetooth protocol architecture.

The Bluetooth Frame Structure

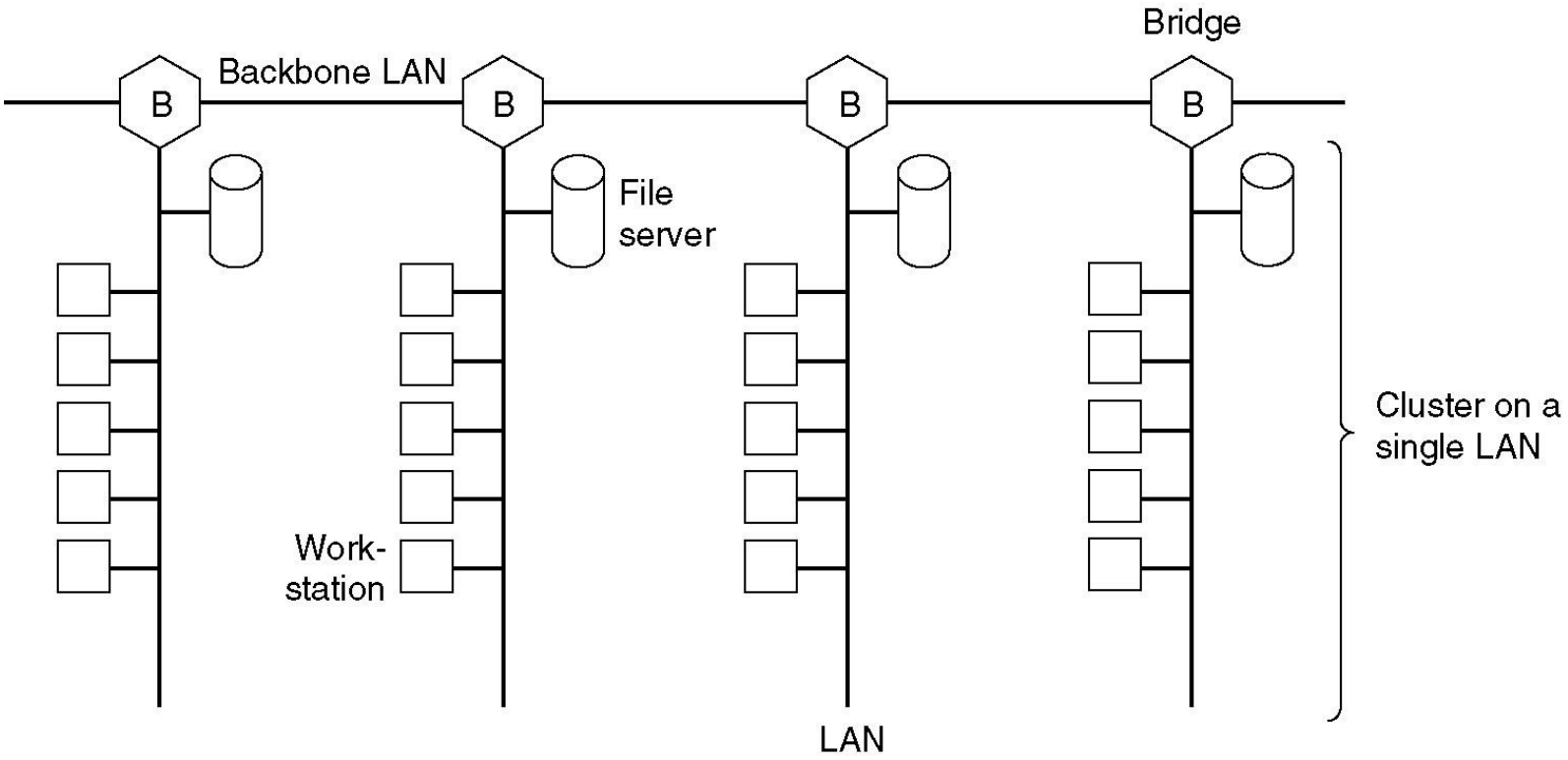


A typical Bluetooth data frame.

Data Link Layer Switching

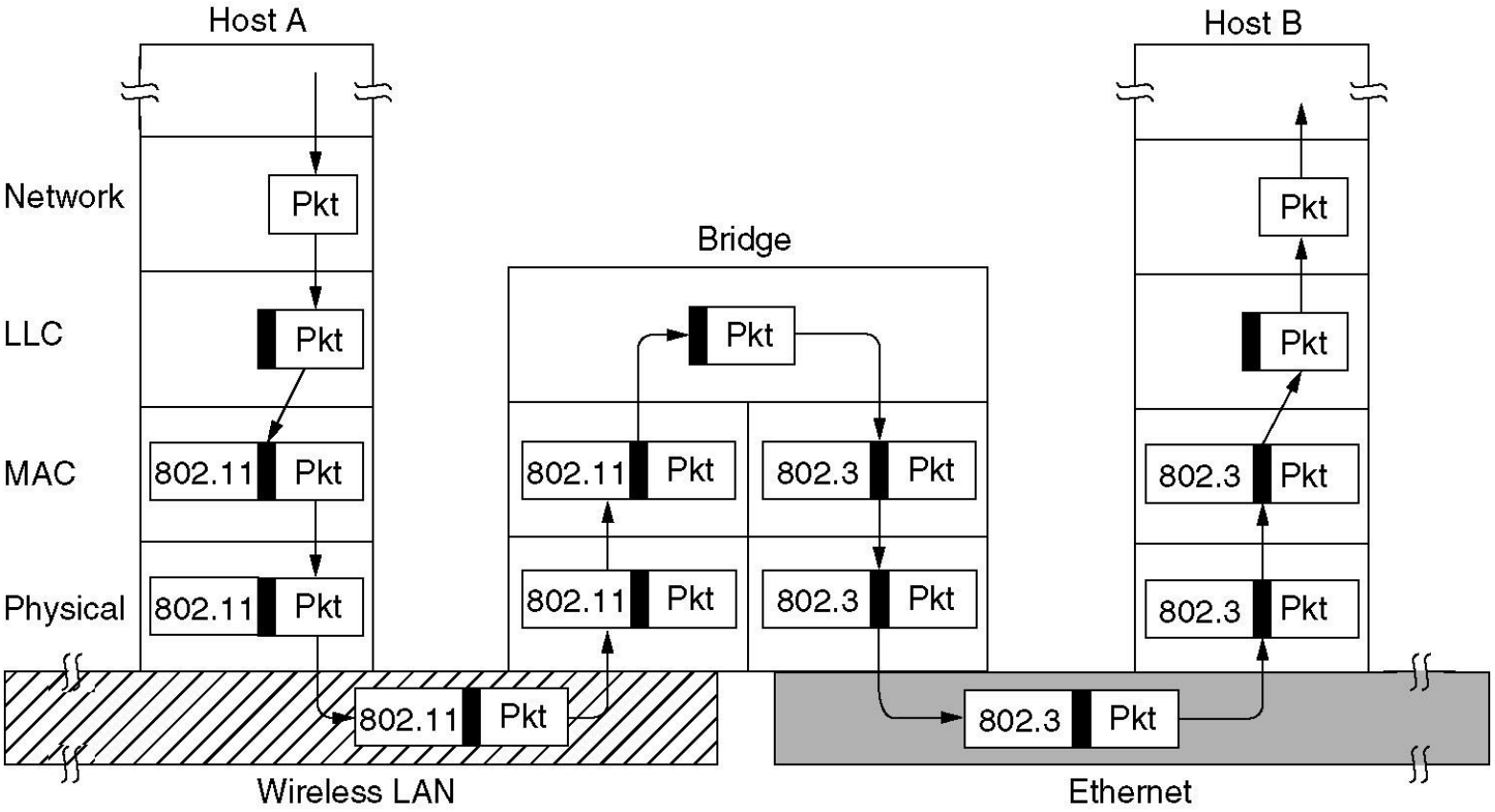
- Bridges from 802.x to 802.y
- Local Internetworking
- Spanning Tree Bridges
- Remote Bridges
- Repeaters, Hubs, Bridges, Switches, Routers, Gateways
- Virtual LANs

Data Link Layer Switching



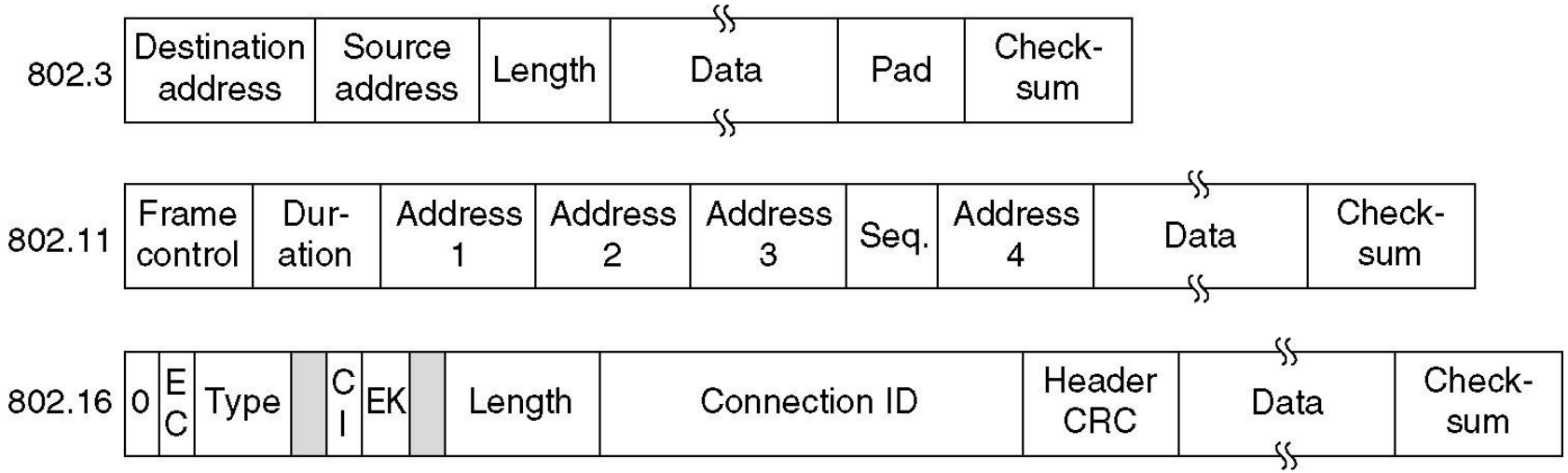
Multiple LANs connected by a backbone to handle a total load higher than the capacity of a single LAN.

Bridges from 802.x to 802.y



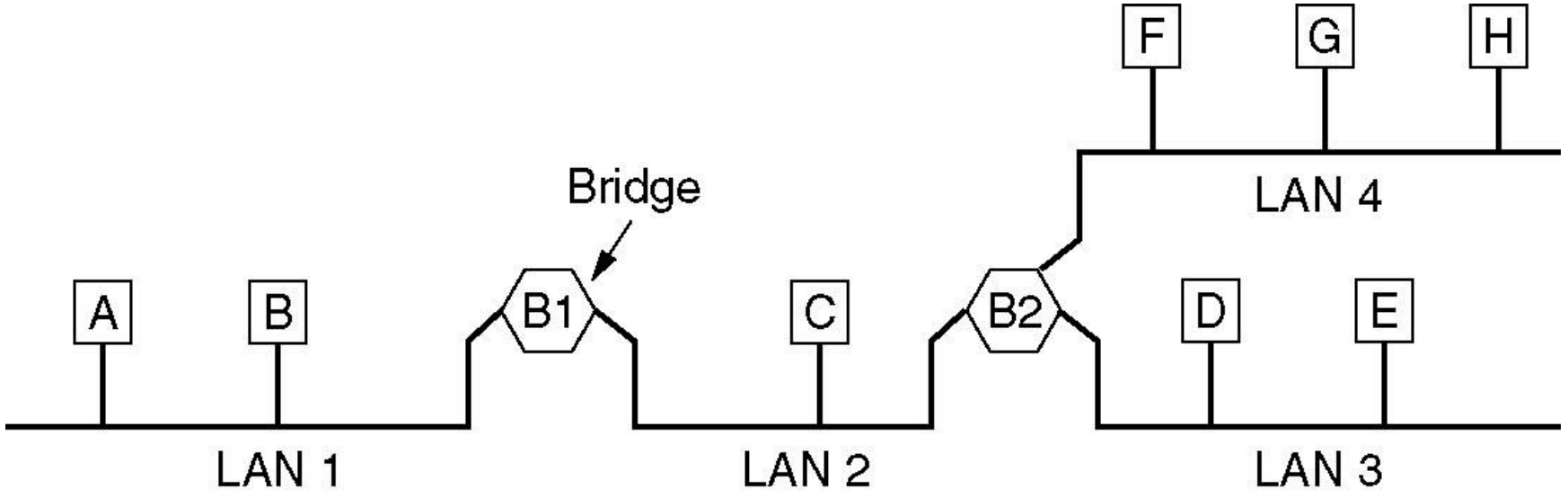
Operation of a LAN bridge from 802.11 to 802.3.

Bridges from 802.x to 802.y (2)



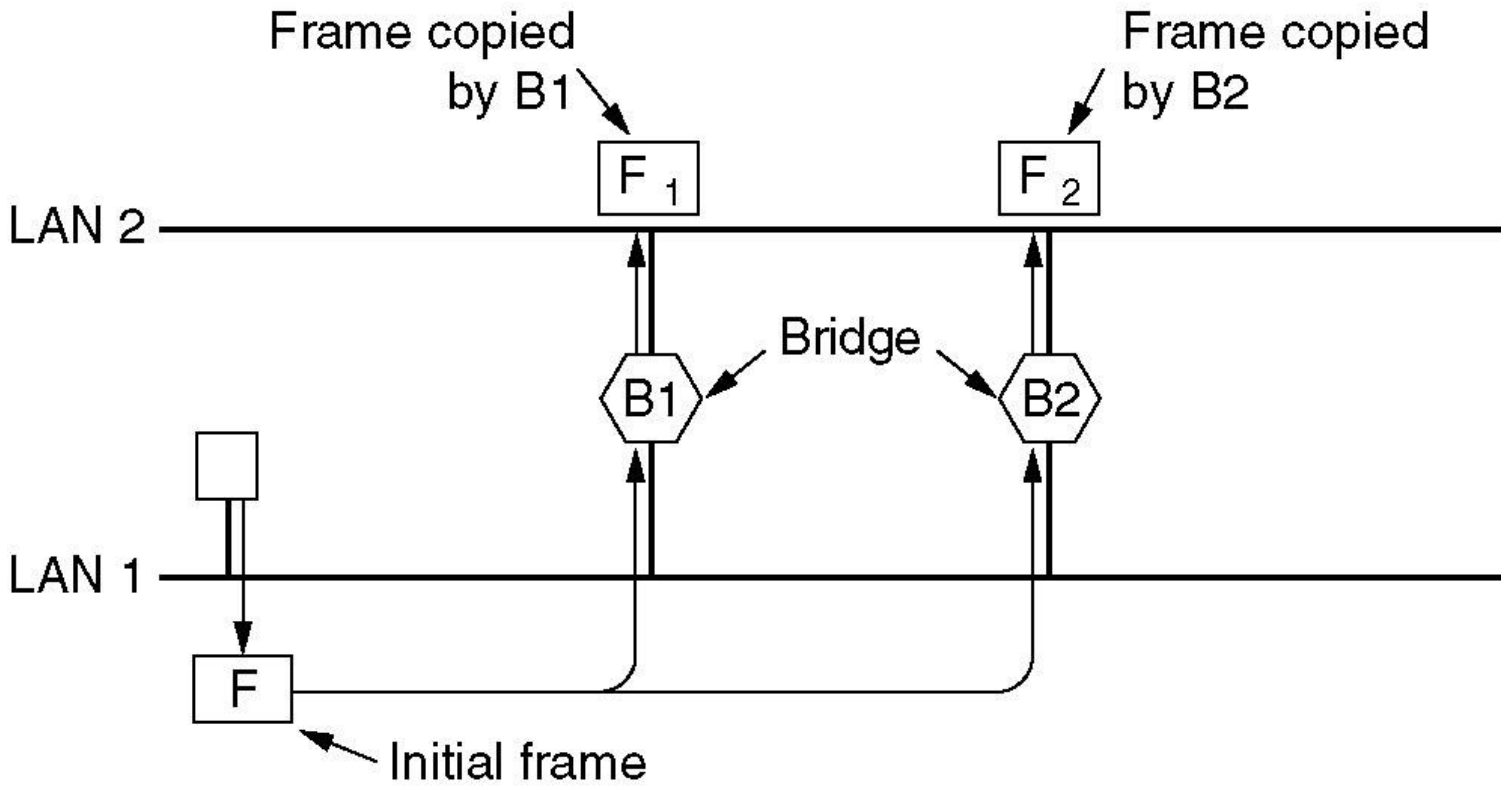
The IEEE 802 frame formats. The drawing is not to scale.

Local Internetworking



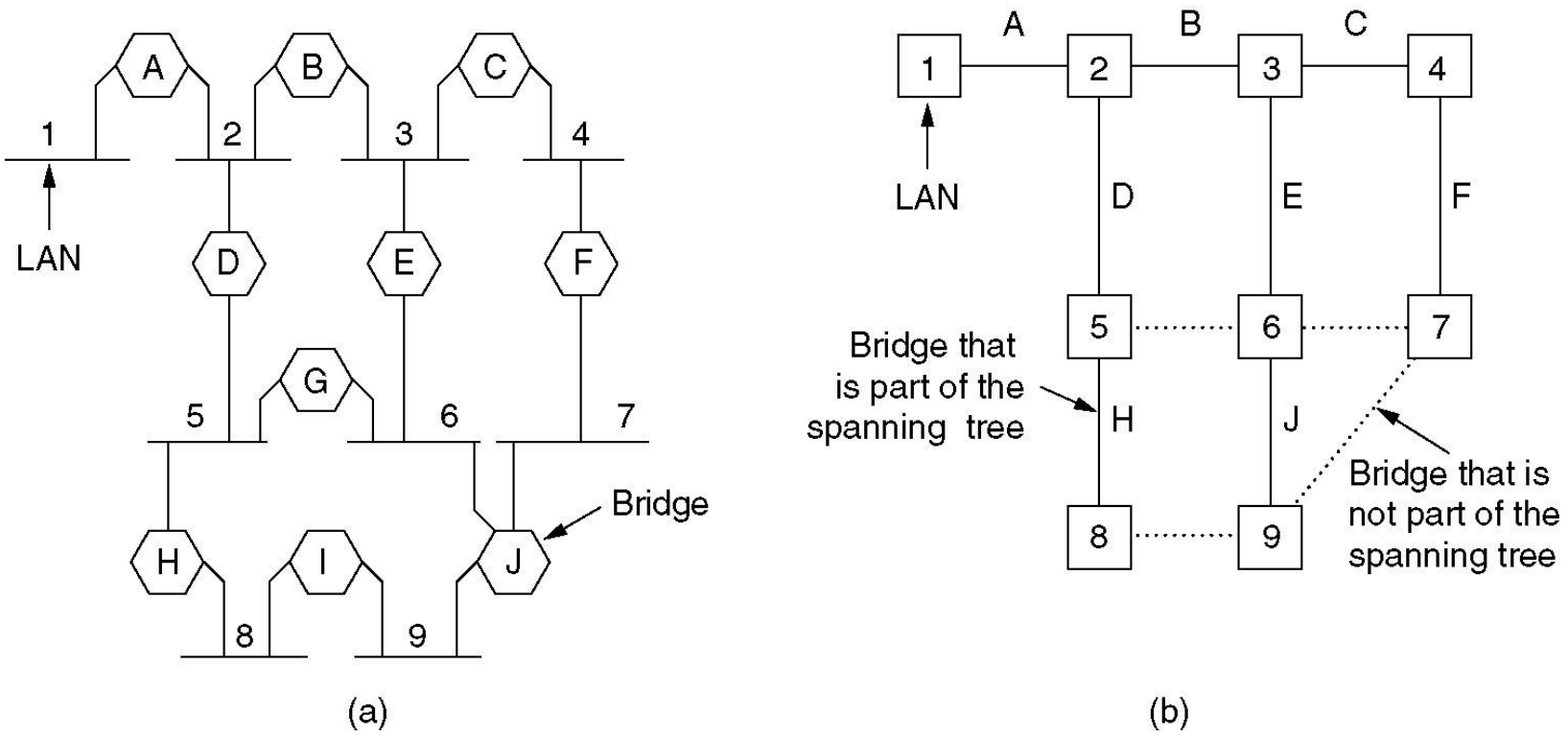
A configuration with four LANs and two bridges.

Spanning Tree Bridges



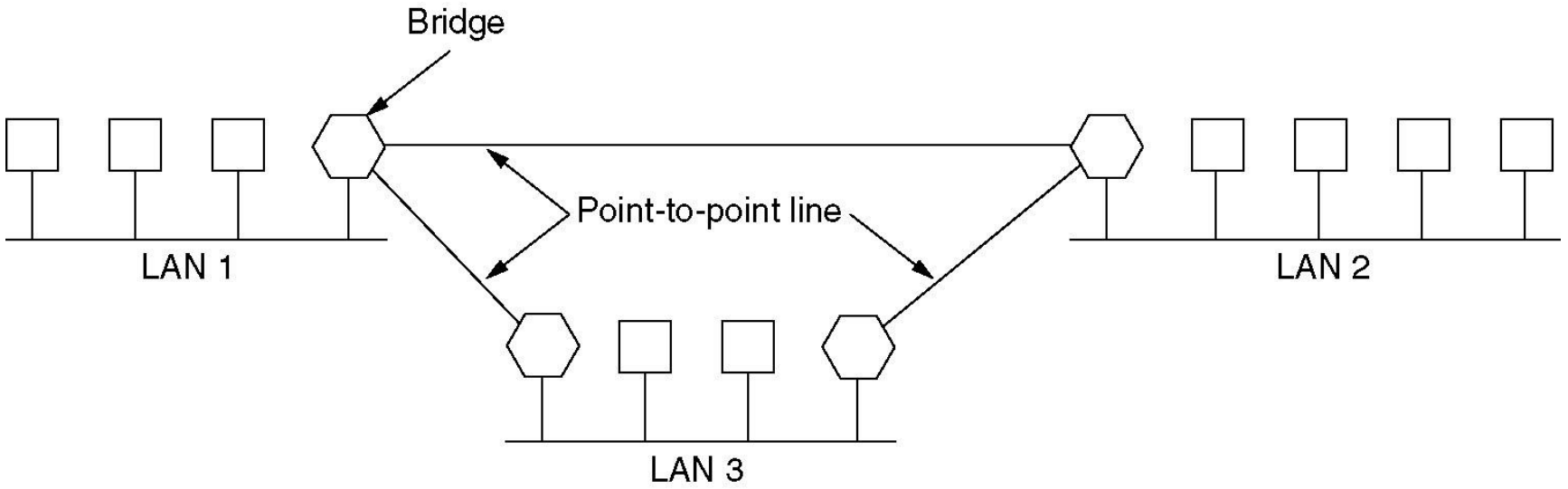
Two parallel transparent bridges.

Spanning Tree Bridges (2)



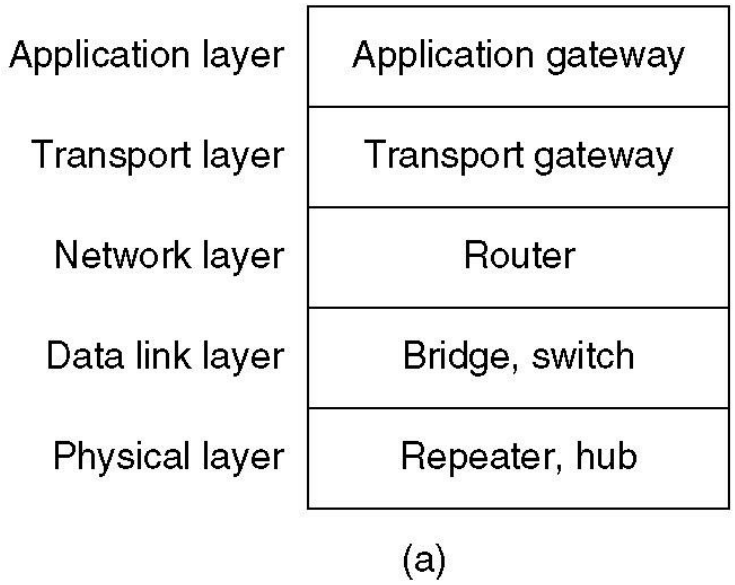
(a) Interconnected LANs. (b) A spanning tree covering the LANs. The dotted lines are not part of the spanning tree.

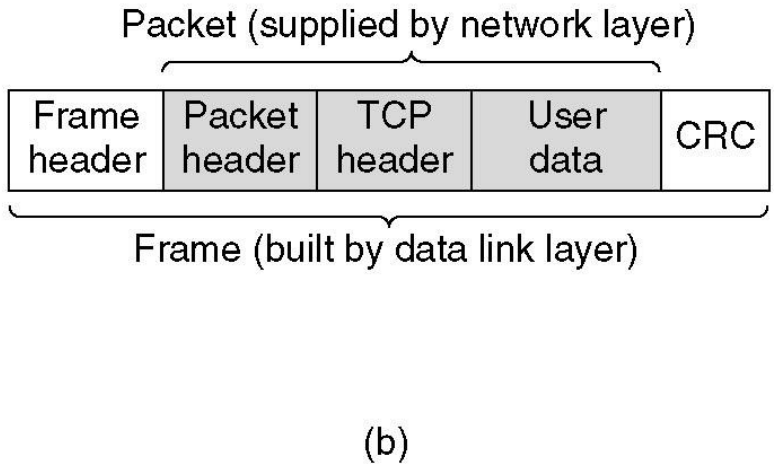
Remote Bridges



Remote bridges can be used to interconnect distant LANs.

Repeaters, Hubs, Bridges, Switches, Routers and Gateways

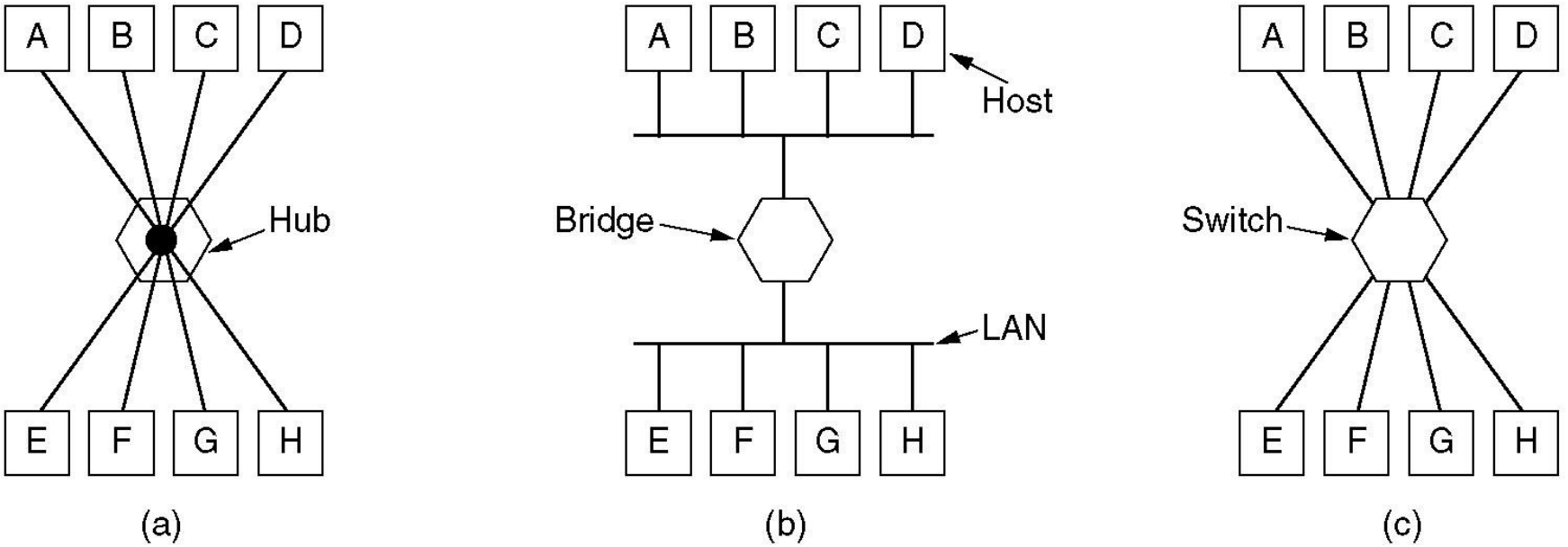




(a) Which device is in which layer.

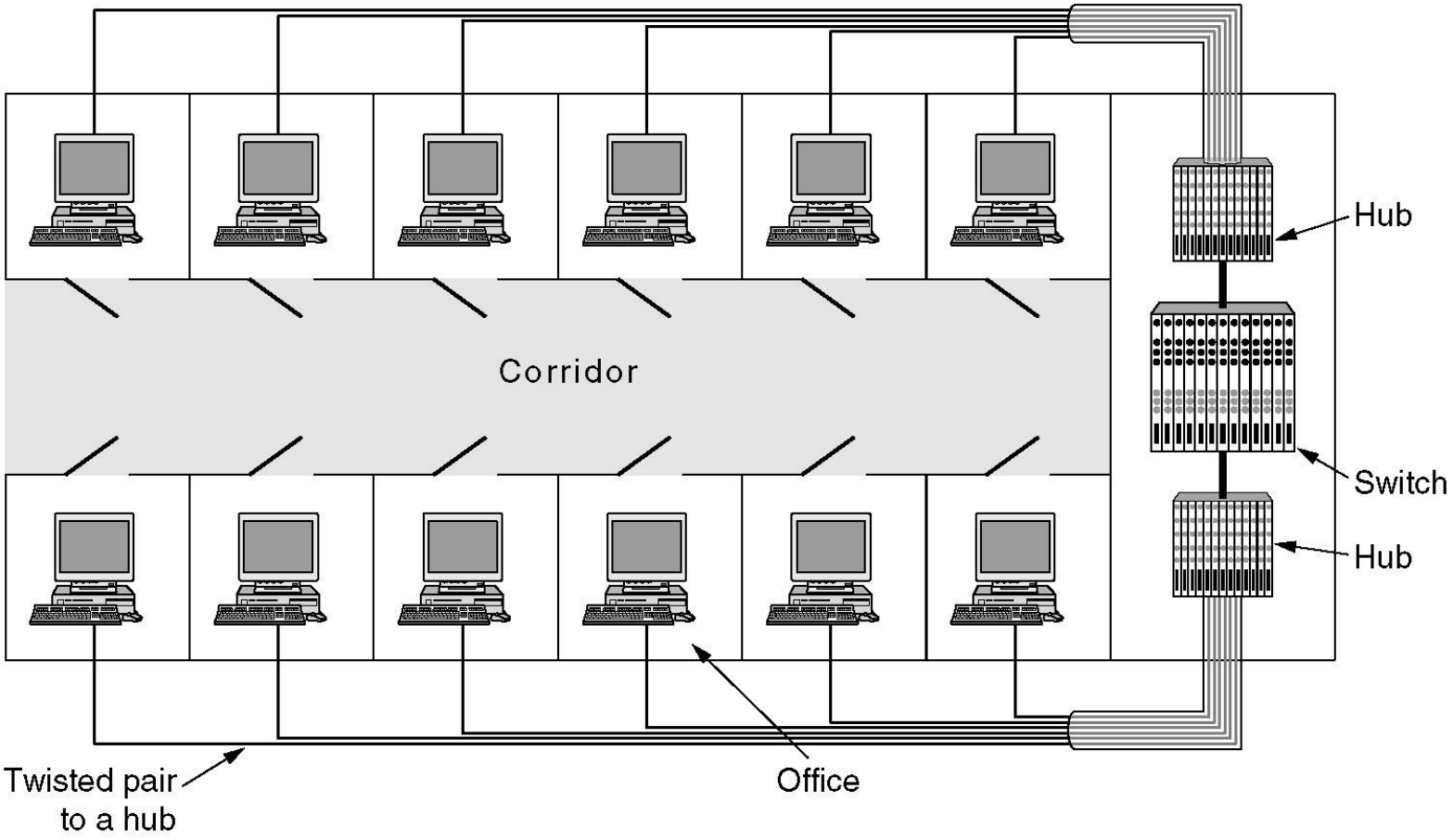
(b) Frames, packets, and headers.

Repeaters, Hubs, Bridges, Switches, Routers and Gateways (2)



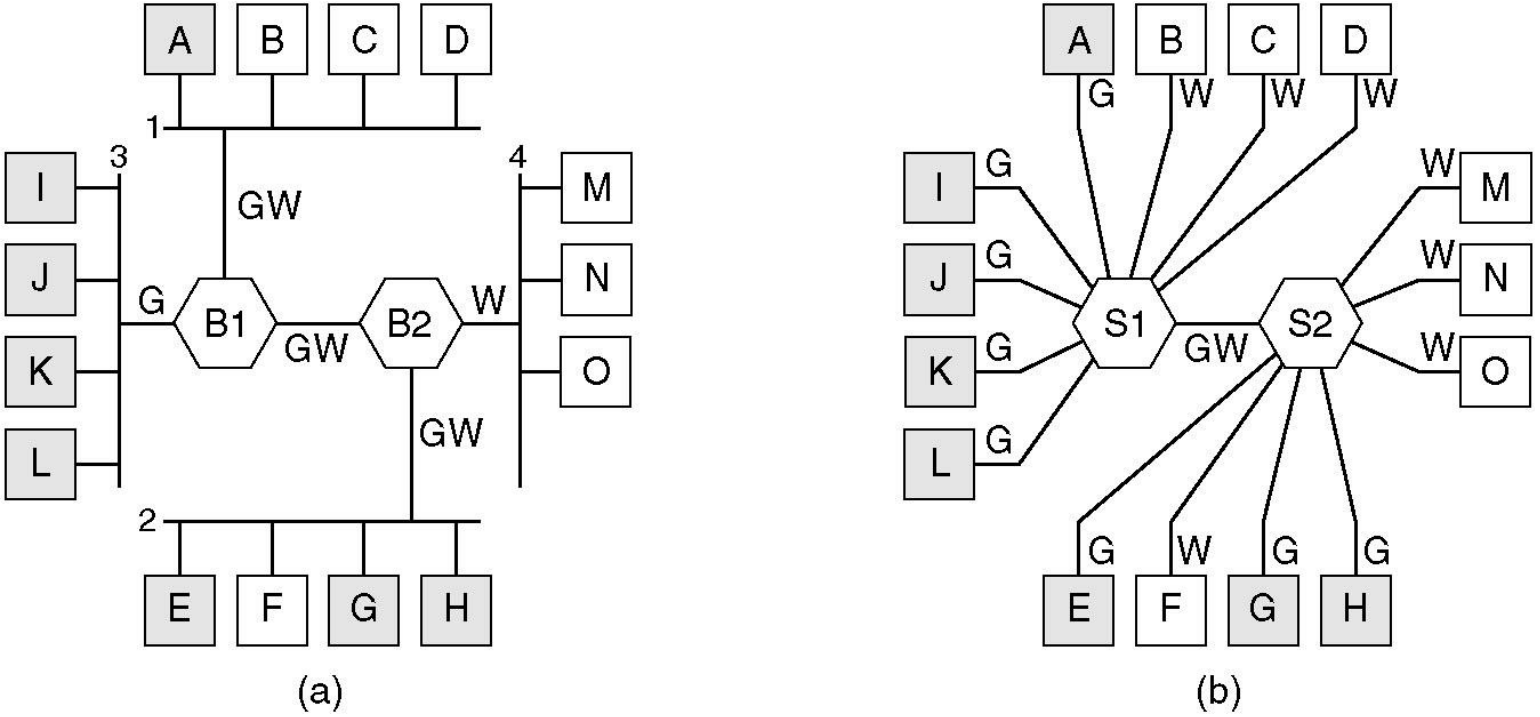
(a) A hub. (b) A bridge. (c) a switch.

Virtual LANs



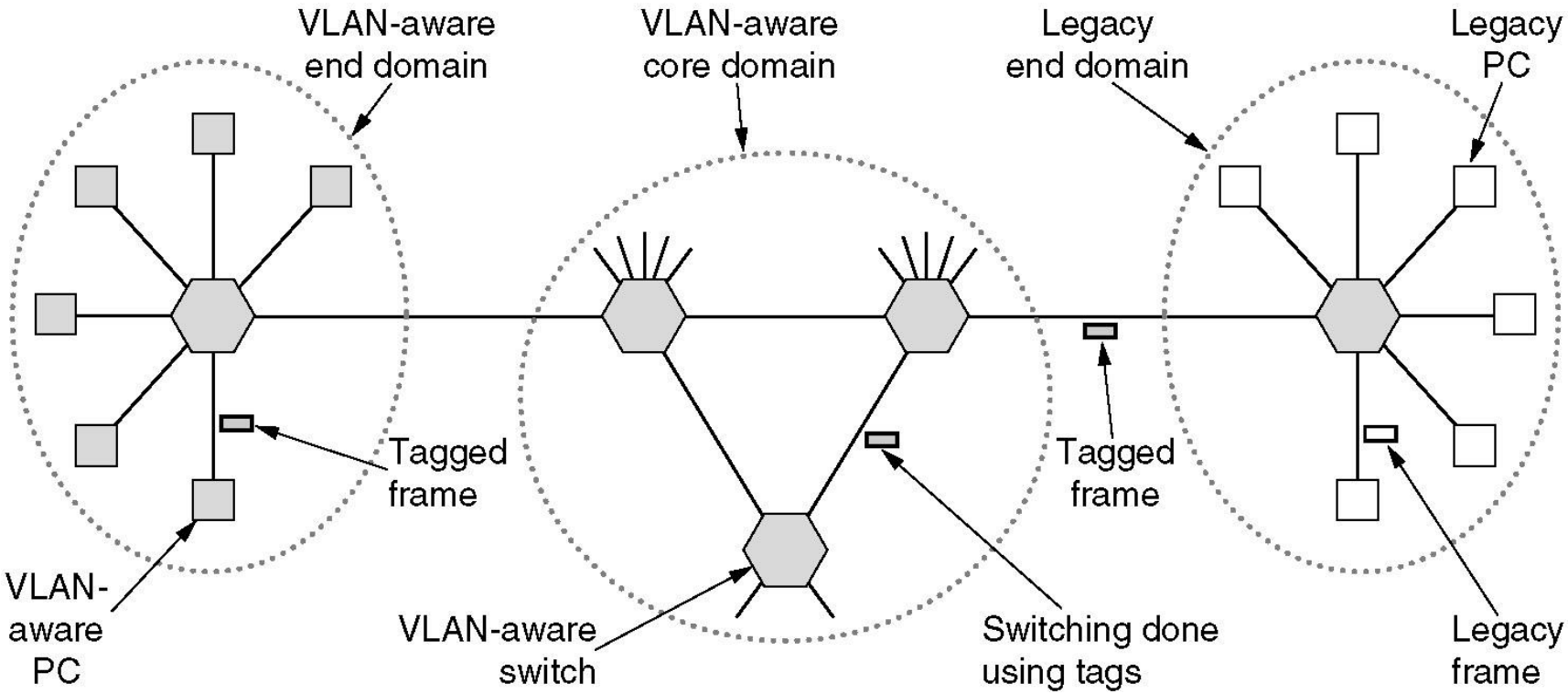
A building with centralized wiring using hubs and a switch.

Virtual LANs (2)



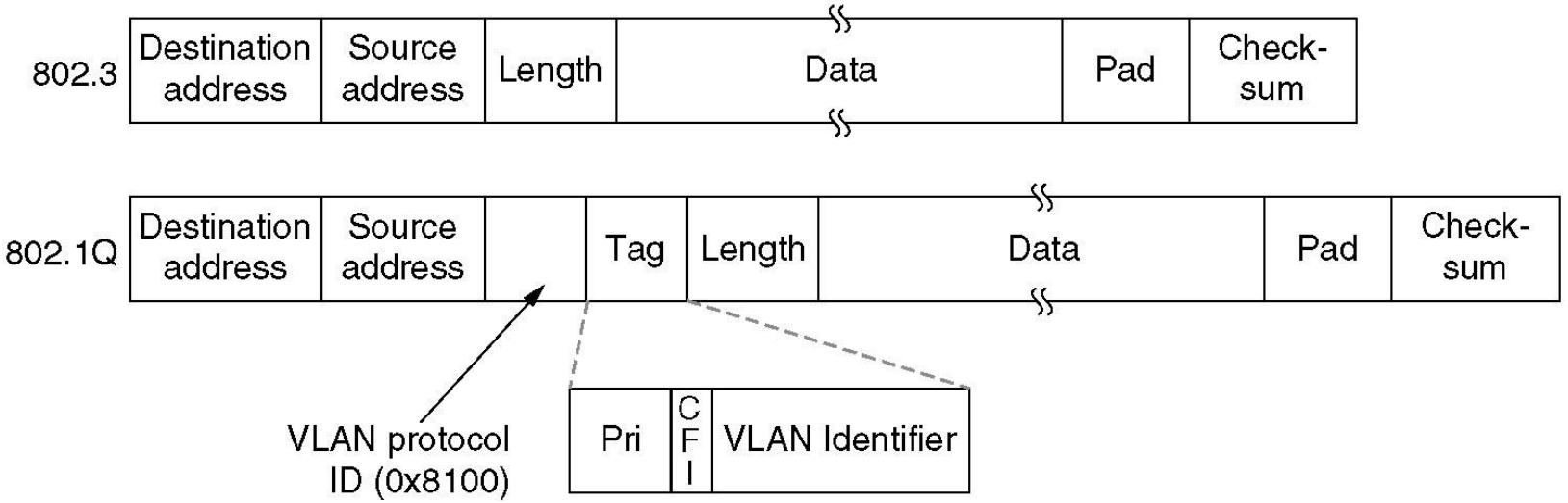
(a) Four physical LANs organized into two VLANs, gray and white, by two bridges. (b) The same 15 machines organized into two VLANs by switches.

The IEEE 802.1Q Standard



Transition from legacy Ethernet to VLAN-aware Ethernet. The shaded symbols are VLAN aware. The empty ones are not.

The IEEE 802.1Q Standard (2)



The 802.3 (legacy) and 802.1Q Ethernet frame formats.

Summary

Method	Description
FDM	Dedicate a frequency band to each station
WDM	A dynamic FDM scheme for fiber
TDM	Dedicate a time slot to each station
Pure ALOHA	Unsynchronized transmission at any instant
Slotted ALOHA	Random transmission in well-defined time slots
1-persistent CSMA	Standard carrier sense multiple access
Nonpersistent CSMA	Random delay when channel is sensed busy
P-persistent CSMA	CSMA, but with a probability of p of persisting
CSMA/CD	CSMA, but abort on detecting a collision
Bit map	Round robin scheduling using a bit map
Binary countdown	Highest numbered ready station goes next
Tree walk	Reduced contention by selective enabling
MACA, MACAW	Wireless LAN protocols
Ethernet	CSMA/CD with binary exponential backoff
FHSS	Frequency hopping spread spectrum
DSSS	Direct sequence spread spectrum
CSMA/CA	Carrier sense multiple access with collision avoidance

Channel allocation methods and systems for a common channel.

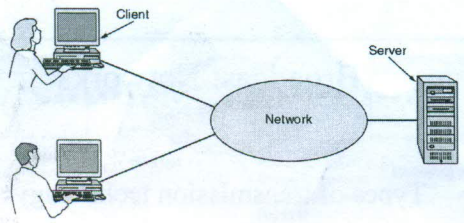
Chapter 1

Introduction

Uses of Computer Networks

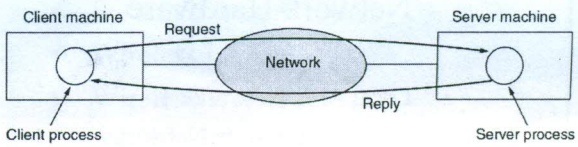
- Business Applications
- Home Applications
- Mobile Users
- Social Issues

Business Applications of Networks



A network with two clients and one server.

Business Applications of Networks (2)

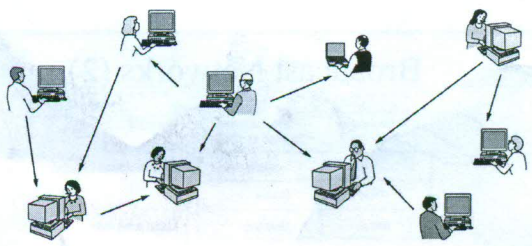


The client-server model involves requests and replies.

Home Network Applications

- Access to remote information
- Person-to-person communication
- Interactive entertainment
- Electronic commerce

Home Network Applications (2)



In peer-to-peer system there are no fixed clients and servers.

Home Network Applications (3)

Tag	Full name	Example
B2C	Business-to-consumer	Ordering books on-line
B2B	Business-to-business	Car manufacturer ordering tires from supplier
G2C	Government-to-consumer	Government distributing tax forms electronically
C2C	Consumer-to-consumer	Auctioning second-hand products on-line
P2P	Peer-to-peer	File sharing

Some forms of e-commerce.

Mobile Network Users

Wireless	Mobile	Applications
No	No	Desktop computers in offices
No	Yes	A notebook computer used in a hotel room
Yes	No	Networks in older, unwired buildings
Yes	Yes	Portable office; PDA for store inventory

Combinations of wireless networks and mobile computing.

Network Hardware

- Local Area Networks
- Metropolitan Area Networks
- Wide Area Networks
- Wireless Networks
- Home Networks
- Internetworks

Broadcast Networks

Types of transmission technology

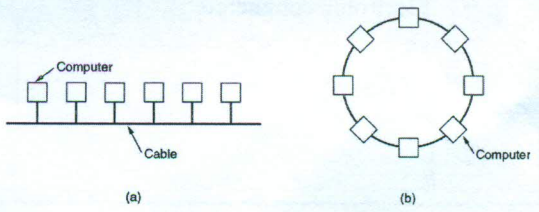
- Broadcast links
- Point-to-point links

Broadcast Networks (2)

Interprocessor distance	Processors located in same	Example
1 m	Square meter	Personal area network
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1000 km	Continent	
10,000 km	Planet	The Internet

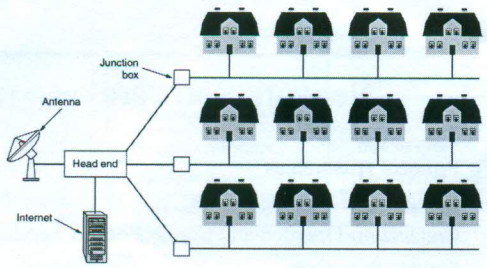
Classification of interconnected processors by scale.

Local Area Networks



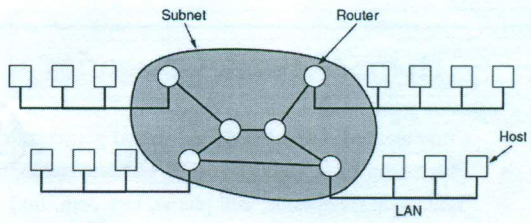
Two broadcast networks
(a) Bus
(b) Ring

Metropolitan Area Networks



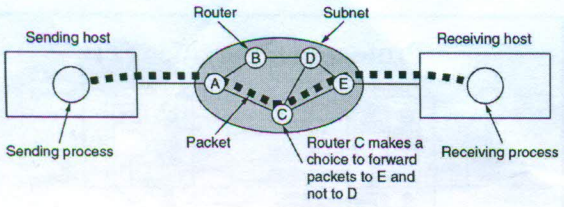
A metropolitan area network based on cable TV.

Wide Area Networks



Relation between hosts on LANs and the subnet.

Wide Area Networks (2)



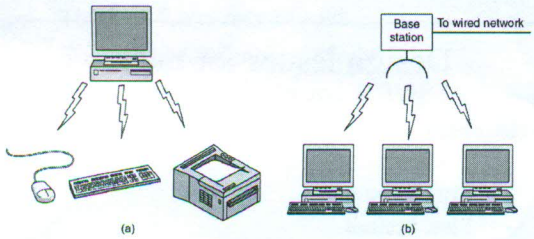
A stream of packets from sender to receiver.

Wireless Networks

Categories of wireless networks:

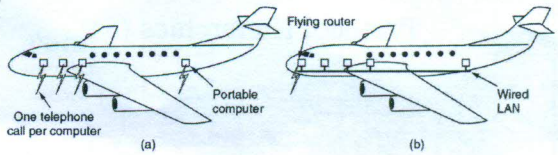
- System interconnection
- Wireless LANs
- Wireless WANs

Wireless Networks (2)



(a) Bluetooth configuration
(b) Wireless LAN

Wireless Networks (3)



(a) Individual mobile computers
(b) A flying LAN

Home Network Categories

- Computers (desktop PC, PDA, shared peripherals)
- Entertainment (TV, DVD, VCR, camera, stereo, MP3)
- Telecomm (telephone, cell phone, intercom, fax)
- Appliances (microwave, fridge, clock, furnace, airco)
- Telemetry (utility meter, burglar alarm, babycam)?

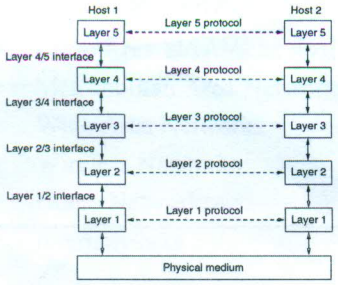
smoke ?

Network Software

- Protocol Hierarchies
- Design Issues for the Layers
- Connection-Oriented and Connectionless Services
- Service Primitives
- The Relationship of Services to Protocols

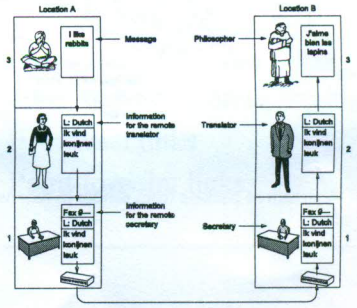
Network Software

Protocol Hierarchies



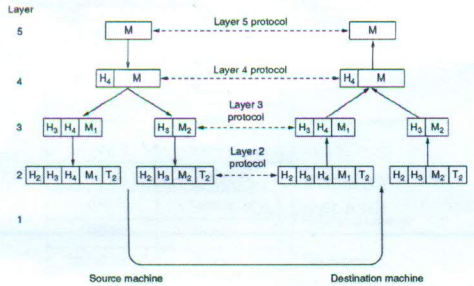
Layers, protocols, and interfaces.

Protocol Hierarchies (2)



The philosopher-translator-secretary architecture.

Protocol Hierarchies (3)



Example information flow supporting virtual communication in layer 5.

Design Issues for the Layers

- Addressing
- Error Control
- Flow Control
- Multiplexing
- Routing

Connection-Oriented and Connectionless Services

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Remote login
	Unreliable connection	Digitized voice
Connection-less	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Registered mail
	Request-reply	Database query

Six different types of service.

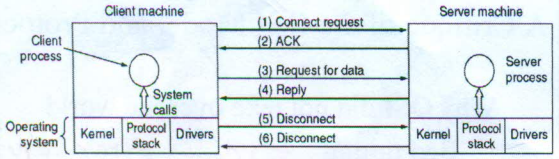
Service Primitives

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

Five service primitives for implementing a simple connection-oriented service.

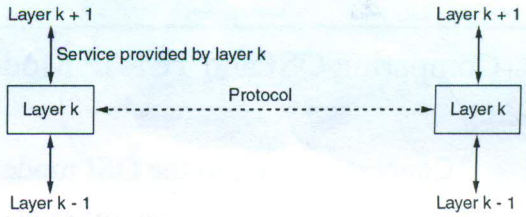
Service ⇒ Set of primitives (operations)
Protocol ⇒ Set of rules

Service Primitives (2)



Packets sent in a simple client-server interaction on a connection-oriented network.

Services to Protocols Relationship

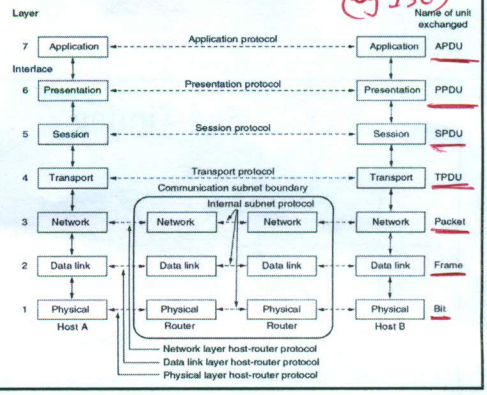


The relationship between a service and a protocol.

Reference Models

- The OSI Reference Model
- The TCP/IP Reference Model
- A Comparison of OSI and TCP/IP
- A Critique of the OSI Model and Protocols
- A Critique of the TCP/IP Reference Model

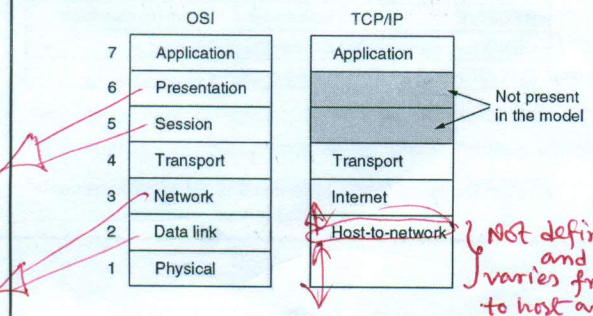
Reference Models



The OSI reference model.

Layer ⇒ perform a well-defined function
interface ⇒ Must be chosen in such way that information flow accross different layers is minimized
# of layers ⇒ should be large enough such that distinct func"s need not be thrown together in the same layer and small enough that the archi does not become unwieldy

Reference Models (2)



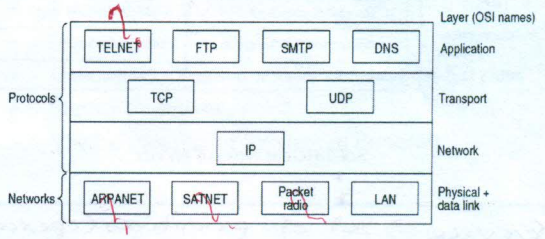
The TCP/IP reference model.

nearly empty

nearly overfull

Not defined and varies from host to host and network to network

Reference Models (3)



Protocols and networks in the TCP/IP model initially.

devised before corresponding protocols were invented.

Comparing OSI and TCP/IP Models

Concepts central to the OSI model

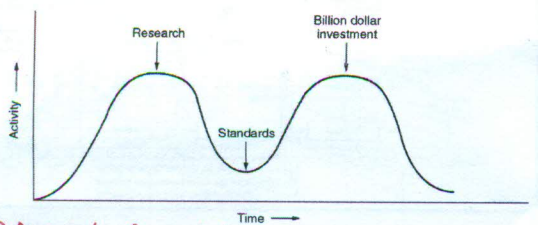
- Services → what the layer does, defines layer's semantics
- Interfaces → (NOT how) How the above layer can access it.
- Protocols → determined by peers

A Critique of the OSI Model and Protocols

Why OSI did not take over the world

- Bad timing ⇒ comes after TCP/IP
- Bad technology ⇒ model & protocols are flawed (extremely complex)
- Bad implementations ⇒ huge, unwieldy, & slow
- Bad politics ⇒ thought to be creature of bureaucrats

Bad Timing



→ Amount of activity surrounding a new subject

The apocalypse of the two elephants.

A Critique of the TCP/IP Reference Model

Problems:

- Service, interface, and protocol not distinguished
- Not a general model
- Host-to-network "layer" not really a layer (Actually an interface)
- No mention of physical and data link layers
- Minor protocols deeply entrenched, hard to replace

→ already deeply used

(however, generally produced by a couple of graduate students)

Hybrid Model

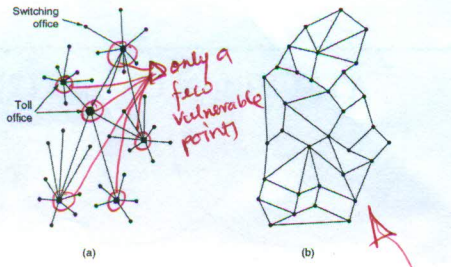
5	Application layer
4	Transport layer
3	Network layer
2	Data link layer
1	Physical layer

The hybrid reference model to be used in this book.

Example Networks

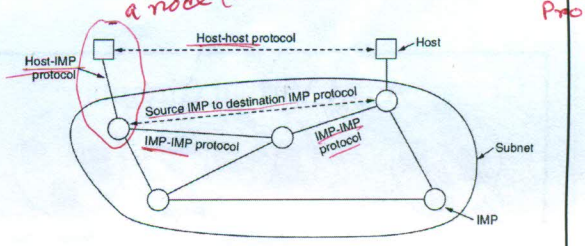
- The Internet
- Connection-Oriented Networks: X.25, Frame Relay, and ATM
- Ethernet
- Wireless LANs: 802:11

The ARPANET



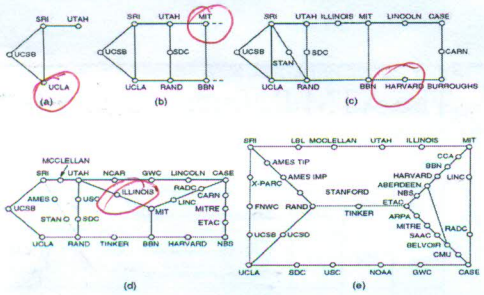
(a) Structure of the telephone system.
(b) Baran's proposed distributed switching system.

The ARPANET (2)



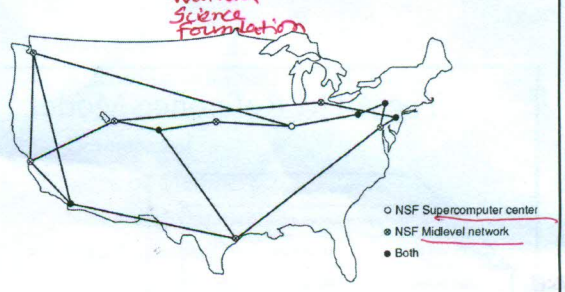
The original ARPANET design.

The ARPANET (3)



Growth of the ARPANET (a) December 1969. (b) July 1970. (c) March 1971. (d) April 1972. (e) September 1972.

NSFNET



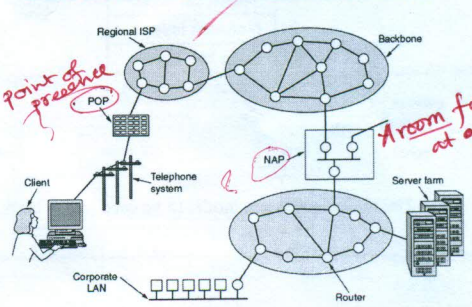
The NSFNET backbone in 1988.

Internet Usage

Traditional applications (1970 – 1990)

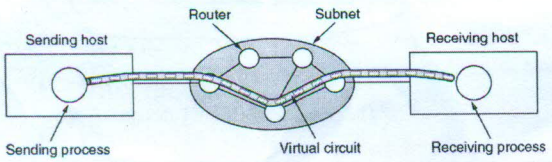
- E-mail
- News
- Remote login
- File transfer

Architecture of the Internet



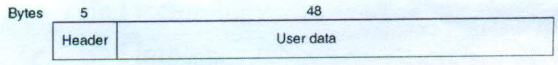
Overview of the Internet.

ATM Virtual Circuits



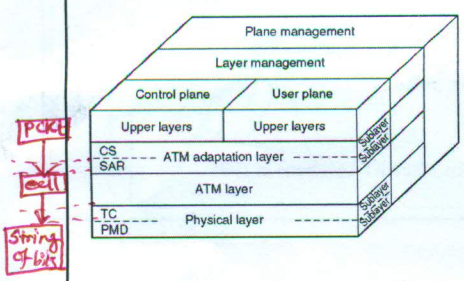
A virtual circuit.

ATM Virtual Circuits (2)



An ATM cell.

The ATM Reference Model



CS: Convergence sublayer
SAR: Segmentation and reassembly sublayer
TC: Transmission convergence sublayer
PMD: Physical medium dependent sublayer

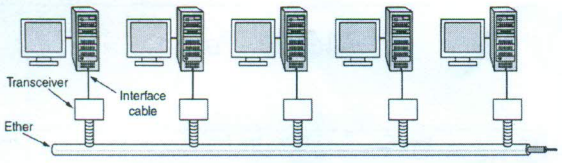
The ATM reference model.

The ATM Reference Model (2)

OSI layer	ATM layer	ATM sublayer	Functionality
3/4	AAL	CS	Providing the standard interface (convergence)
		SAR	Segmentation and reassembly
2/3	ATM		Flow control Cell header generation/extraction Virtual circuit/path management Cell multiplexing/demultiplexing
2	Physical	TC	Cell rate decoupling Header checksum generation and verification Cell generation Packing/unpacking cells from the enclosing envelope Frame generation
1		PMD	Bit timing Physical network access

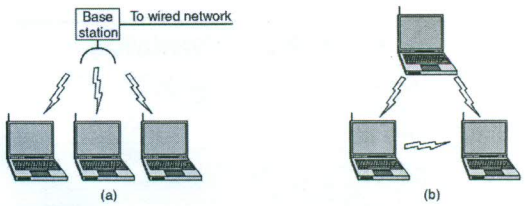
The ATM layers and sublayers and their functions.

Ethernet



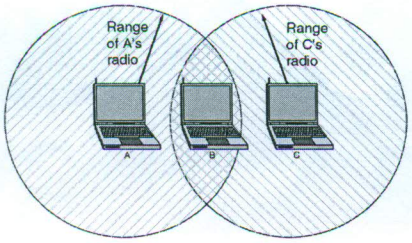
Architecture of the original Ethernet.

Wireless LANs



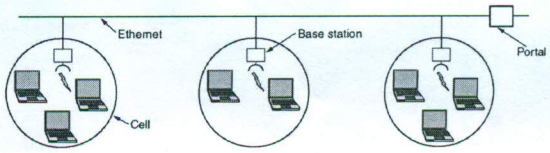
(a) Wireless networking with a base station.
(b) Ad hoc networking.

Wireless LANs (2)



The range of a single radio may not cover the entire system.

Wireless LANs (3)



A multicell 802.11 network.

Network Standardization

- Who's Who in the Telecommunications World
- Who's Who in the International Standards World
- Who's Who in the Internet Standards World

ITU

- Main sectors
  - Radiocommunications
  - Telecommunications Standardization
  - Development
- Classes of Members
  - National governments
  - Sector members
  - Associate members
  - Regulatory agencies

8/26/2015

IEEE 802 Standards

Number	Topic
802.1	Overview and architecture of LANs
802.2 ↓	Logical link control
802.3 *	Ethernet
802.4 ↓	Token bus (was briefly used in manufacturing plants)
802.5	Token ring (IBM's entry into the LAN world)
802.6 ↓	Dual queue dual bus (early metropolitan area network)
802.7 ↓	Technical advisory group on broadband technologies
802.8 †	Technical advisory group on fiber optic technologies
802.9 ↓	Isochronous LANs (for real-time applications)
802.10 ↓	Virtual LANs and security
802.11 *	Wireless LANs
802.12 ↓	Demand priority (Hewlett-Packard's AnyLAN)
802.13	Unlucky number. Nobody wanted it
802.14 ↓	Cable modems (defunct: an industry consortium got there first)
802.15 *	Personal area networks (Bluetooth)
802.16 *	Broadband wireless
802.17	Resilient packet ring

The 802 working groups. The important ones are marked with *. The ones marked with ↓ are hibernating. The one marked with † gave up.

Metric Units

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

The principal metric prefixes.

Chapter 2

The Physical Layer

The Theoretical Basis for Data Communication

- Fourier Analysis
- Bandwidth-Limited Signals
- Maximum Data Rate of a Channel

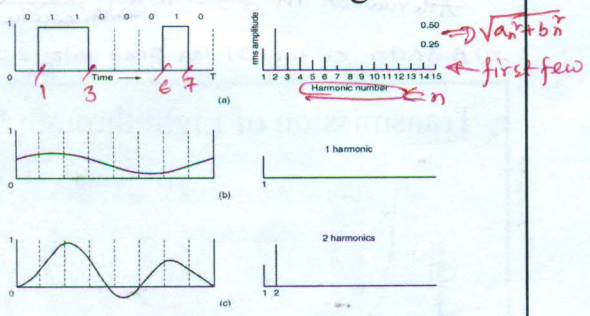
b → 01100010 [8-bit representation]

$a_n = \frac{1}{\pi n}\left[\cos\left(\frac{\pi n}{4}\right) - \cos\left(\frac{3\pi n}{4}\right) + \cos\left(\frac{5\pi n}{4}\right) - \cos\left(\frac{7\pi n}{4}\right)\right]$

$b_n = \frac{1}{\pi n}\left[\sin\left(\frac{3\pi n}{4}\right) - \sin\left(\frac{\pi n}{4}\right) + \sin\left(\frac{7\pi n}{4}\right) - \sin\left(\frac{5\pi n}{4}\right)\right]$

$c = 3/4$

Bandwidth-Limited Signals



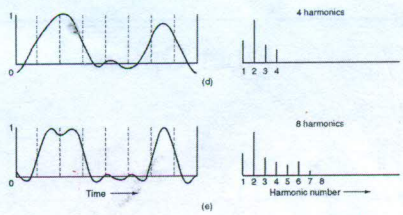
A binary signal and its root-mean-square Fourier amplitudes.
(b) – (c) Successive approximations to the original signal.

$\sqrt{a_n^2 + b_n^2}$ ← first few

# of harmonics ↑ signal accuracy ↑

[Attenuation of different fourier components are different → distortion]

Bandwidth-Limited Signals (2)



Harmonic ↑ freq ↑

(d) – (e) Successive approximations to the original signal.

Bandwidth-Limited Signals (3)

b/8 Hz

Bps	T (msec)	First harmonic (Hz)	# Harmonics sent
300	26.67	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

~~bps ↑ (freq ↑, T ↓) harmonics ↑~~

Relation between data rate and harmonics.

for a cutoff freq of 3000 Hz (given), # of pass freq (harmonics) = $\frac{3000}{(b/8)} = \frac{24000}{b}$

Guided Transmission Data

- Magnetic Media (CD, DVD)
- Twisted Pair
- Coaxial Cable
- Fiber Optics

Fourier series

⇒ Any reasonably behaved periodic function, $g(t)$ with period T can be constructed as sum of a (possibly infinite) number of sines and cosines:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi n f t) + \sum_{n=1}^{\infty} b_n \cos(2\pi n f t)$$

where $f = \frac{1}{T}$ is the fundamental frequency, a_n and b_n are the sine and cosine amplitudes of the nth harmonics (terms), and c is a constant. [a_n, b_n, T, ⇒ unknown]

$a_n = \frac{2}{T}\int_0^T g(t)\sin(2\pi n f t)\,dt$ → through multiplying $\sin(k\pi n f t)$ on both sides and considering $\int_0^T \sin(k\pi k f t)\sin(k\pi n f t)\,dt = \begin{cases} 0 & \text{for } k \neq n \\ \frac{T}{2} & \text{for } k = n \end{cases}$

$b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi n f t)\,dt$

$B_n = \frac{2}{T}\int_0^T g(t)\,dt$

Cat 3	16 MHz
Cat 5	100 MHz
Cat 6	250 MHz
Cat 7	600 MHz

Twisted Pair

Category 3 => two insulated wires gently twisted together

2 such pairs are typically grouped in a plastic sheath

(a)

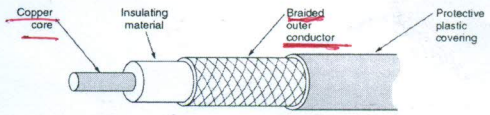
Cat 5 => Similar to Cat 3, but more twists per cm (clean crosstalk better quality signal over long distance)

(b)

(a) Category 3 UTP.
(b) Category 5 UTP.

UTP (Unshielded Twisted Pair)

Coaxial Cable

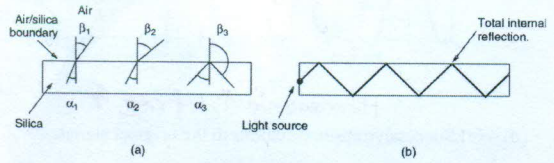


A coaxial cable.

Attenuation in dB = $10 \log_{10} \frac{\text{transmitted power}}{\text{received power}}$

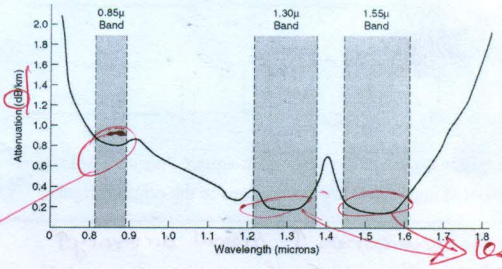
=> A factor of two (2) loss gives $10 \log_{10} 2 = 3$ dB

Fiber Optics



(a) Three examples of a light ray from inside a silica fiber impinging on the air/silica boundary at different angles.
(b) Light trapped by total internal reflection.

Transmission of Light through Fiber



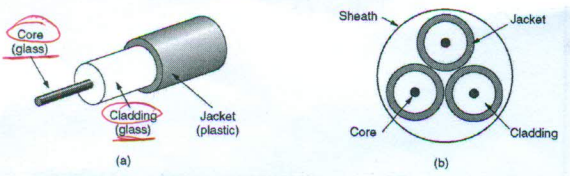
Attenuation of light through fiber in the infrared region.

=> 3 wavelengths are used for optical comm.

less than 5% loss per km

Higher attenuation, however, at this attenuation, laser and electronics can be made from the same materials (GaAs)

Fiber Cables



(a) Side view of a single fiber.
(b) End view of a sheath with three fibers.

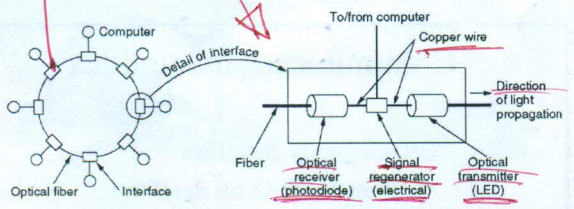
Fiber Cables (2)

Item	LED	Semiconductor laser
Data rate	Low	High
Fiber type	Multimode	Multimode or single mode
Distance	Short	Long
Lifetime	Long life	Short life
Temperature sensitivity	Minor	Substantial
Cost	Low cost	Expensive

Single mode

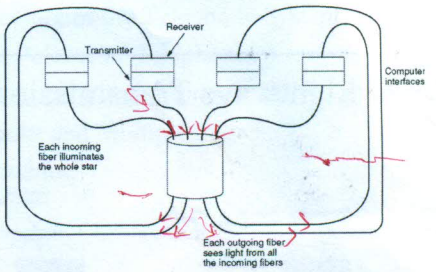
A comparison of semiconductor diodes and LEDs as light sources.

Fiber Optic Networks



A fiber optic ring with active repeaters.

Fiber Optic Networks (2)

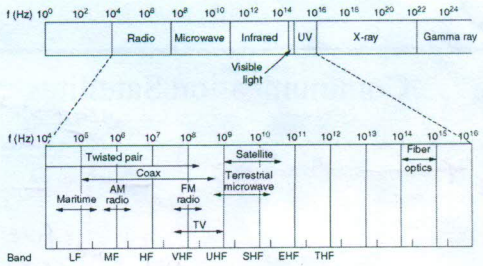


A passive star connection in a fiber optics network.

Wireless Transmission

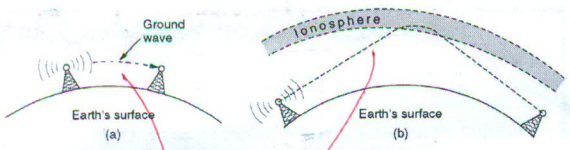
- The Electromagnetic Spectrum
- Radio Transmission
- Microwave Transmission
- Infrared and Millimeter Waves
- Lightwave Transmission

The Electromagnetic Spectrum



The electromagnetic spectrum and its uses for communication.

Radio Transmission

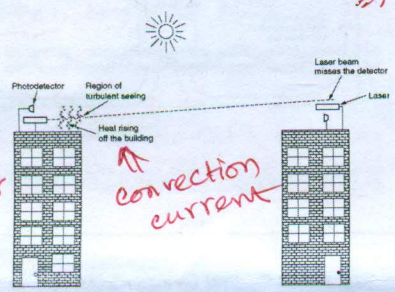


(a) In the VLF, LF, and MF bands, radio waves follow the curvature of the earth.
(b) In the HF band, they bounce off the ionosphere.

Politics of the Electromagnetic Spectrum

The ISM bands in the United States.

Lightwave Transmission

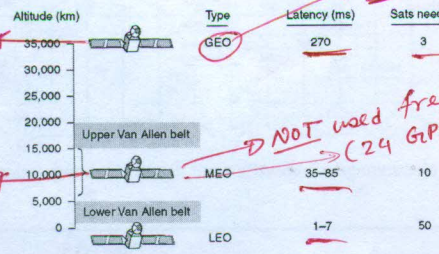


Convection currents can interfere with laser communication systems. A bidirectional system with two lasers is pictured here.

Communication Satellites

- Geostationary Satellites
- Medium-Earth Orbit Satellites
- Low-Earth Orbit Satellites
- Satellites versus Fiber

Communication Satellites



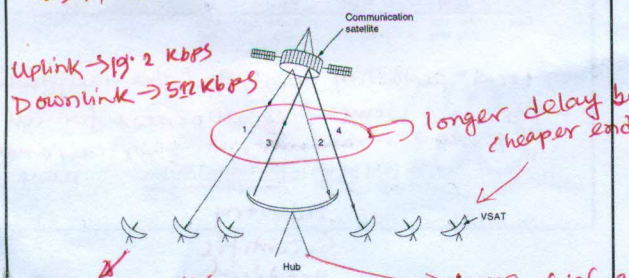
Communication satellites and some of their properties, including altitude above the earth, round-trip delay time and number of satellites needed for global coverage.

Communication Satellites (2)

Band	Downlink	Uplink	Bandwidth	Problems
L	1.5 GHz	1.6 GHz	15 MHz	Low bandwidth; crowded
S	1.9 GHz	2.2 GHz	70 MHz	Low bandwidth; crowded
C	4.0 GHz	6.0 GHz	500 MHz	Terrestrial interference
Ku	11 GHz	14 GHz	500 MHz	Rain
Ka	20 GHz	30 GHz	3500 MHz	Rain, equipment cost

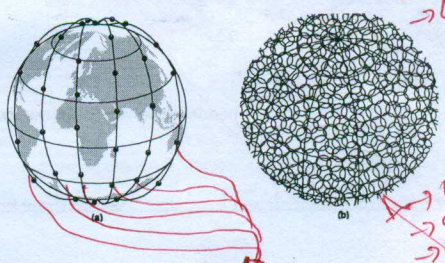
The principal satellite bands.

Communication Satellites (3)



VSATs using a hub.

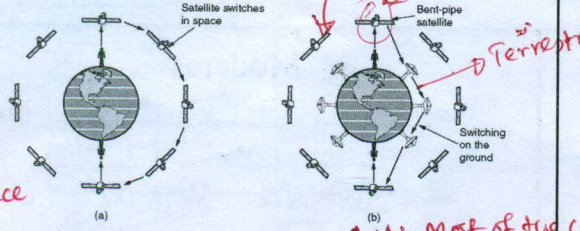
Low-Earth Orbit Satellites



(a) The Iridium satellites from six necklaces around the earth.
(b) 1628 moving cells cover the earth.

	delay	Broadcasting
Satellite	270 msec	good
Terrestrial	3 μs/km	bad (mostly point-to-point)

Globalstar (48 LEO)



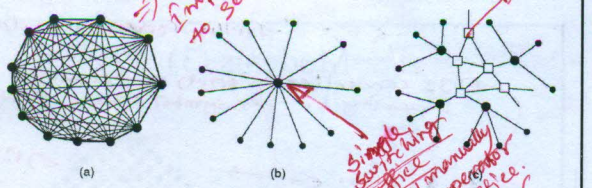
(a) Relaying in space.
(b) Relaying on the ground.

Bent pipe
Terrestrial
Iridium
Comm'n from North pole to south pole
Comm'n in space
Avd: Most of the complexities are on earth; easier to manage

Public Switched Telephone System (PSTN)

- Structure of the Telephone System
- The Politics of Telephones
- The Local Loop: Modems, ADSL and Wireless
- Trunks and Multiplexing
- Switching

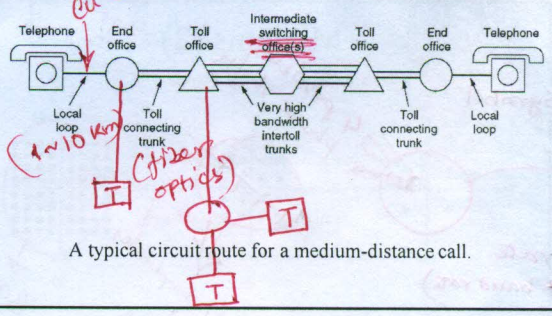
Structure of the Telephone System



(a) Fully-interconnected network.
(b) Centralized switch.
(c) Two-level hierarchy.

Too hard to scale up.
2nd level switching office
Simple switching office (call manually to an operator in the office. The operator manually connects the caller to callee)

Structure of the Telephone System (2)



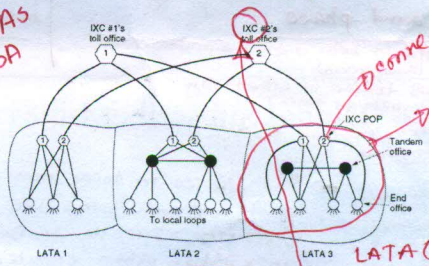
A typical circuit route for a medium-distance call.

Cu cable (twisted pair)
(1 to 10 km)
(fiber optics)

Major Components of the Telephone System

- Local loops
  - Analog twisted pairs going to houses and businesses
- Trunks
  - Digital fiber optics connecting the switching offices
- Switching offices
  - Where calls are moved from one trunk to another

The Politics of Telephones



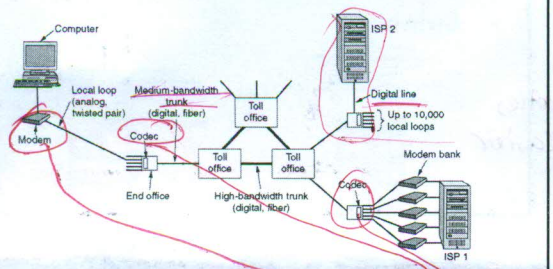
The relationship of LATAs, LECs, and IXCs. All the circles are LEC switching offices. Each hexagon belongs to the IXC whose number is on it.

164 LATAs in USA
connect LATA
One area code
LATA (Local Access and Transport Areas)
Inter Xchange Carriers
local Exchange Carriers (determines one rate, monopoly)
[One LATA can have one or more LECs]

8/26

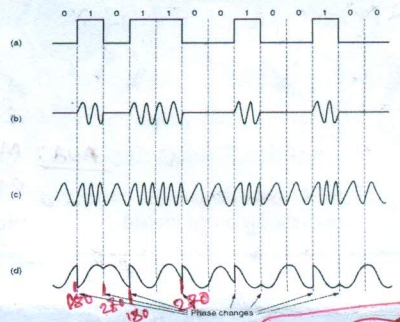
⇒ Square wave ⇒ wide frequency spectrum (used in digital signal) that are subject to strong attenuation and delay distortion
- So, AC signals are used instead of DC

The Local Loop: Modems, ADSL, and Wireless



The use of both analog and digital transmissions for a computer to computer call. Conversion is done by the modems and codecs.

Modems

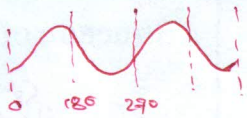


⇒ Sine wave carriers

(a) A binary signal
(b) Amplitude modulation
(c) Frequency modulation (Freq shift keying)
(d) Phase modulation

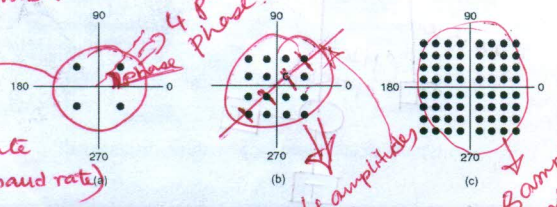
(b) → 2 different amplitudes represent 0 and 1

(c) → 2 diff freq used for 0 and 1



3 Problems encountered by tx lines
① Attenuation: Loss of energy as the signal propagates outward
② Distortion: Due to different propagation speed by different Fourier component over the wire [Differ for different Fourier components]
③ Noise: Unwanted energy from sources other than the transmitter.

Modems (2) =???



baud → Symbol

4 possible phase shifts

1 baud = 2 bit
(So, bit rate = 2 * baud rate)

4 amplitudes and 4 phase

Amplitudes and phases

Quadrature Phase Shift Keying — (a) QPSK → 2 bit/symbol
Quadrature Amplitude Modulation — (b) QAM-16 → 4 bit/symbol
(c) QAM-64 → 6 bit/symbol

4x4

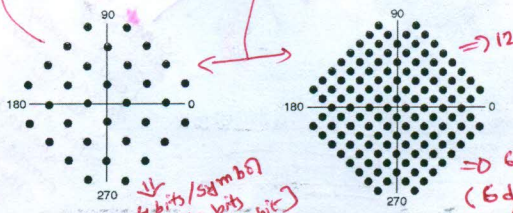
Constellation Diagram
→ Legal combination of amplitude and phase

Full duplex ⇒ Traffic in both directions simultaneously
Half u ⇒ Allows traffic in either way (single railroad track)
Simplex ⇒ Allows only one-way traffic (one-way street)

Modems (3)

Trellis Coded Modulation (TCM)

→ 32 constellation points to transmit 4-bit data and 1 parity bit per symbol (Total 5 bits/symbol) ⇒ 2^5



⇒ 128 constellation points to transmit 6 bits data and 1 bit parity

4 bits/symbol [4 data bits + 1 parity bit]

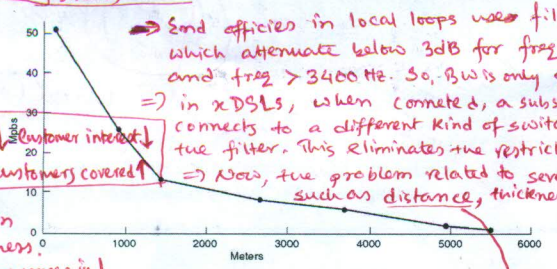
⇒ 6 bits/symbol (6 data bits and 1 parity bit)

over a 2400 baud rate
(a) V.32 for 9600 bps.
(b) V32 bis for 14,400 bps.

	Parity	total bits/sym	data bits	Parity	data rate over 2400 baud rate
QPSK	4	2	2	0	4800 bps
QAM-16	16	4	4	0	9600
QAM-64	64	6	6	0	14,400
V.32	32	5	4	1	9600
V.32 bis	128	7	6	1	14,400
V.34			12		28,800
V.34 bis			14		33,600

⇒ With many points in constellation graphs, small noise in detected amplitude and phase can result in an error, potentially, many bad bits. To reduce this, modems do error corrections.
→ TCM (Trellis Coded Modulation)

Digital Subscriber Lines (DSL)



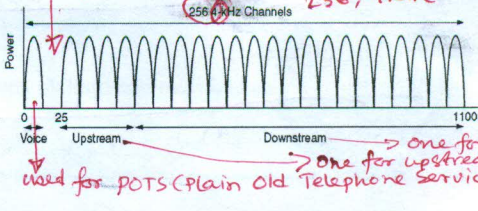
⇒ End offices in local loops use filters which attenuate below 3dB for freq < 300 Hz and freq > 3400 Hz. So, BW is only 3100 Hz
⇒ In xDSLs, when connected, a subscriber connects to a different kind of switch omitting the filter. This eliminates the restriction of 3100 Hz
⇒ Now, the problem related to several factors such as distance, thickness, ...

dist ↑ Speed ↓ (customer interest ↓)
dist ↑ # of customers covered ↑
conflicting in terms of business!
Here, technology comes in!

Bandwidth versus distanced over category 3 UTP for DSL. (assuming other factors remain optimal)

Digital Subscriber Lines (2)

channel 1-5 not used to keep voice and data separated



⇒ 1.1 MHz

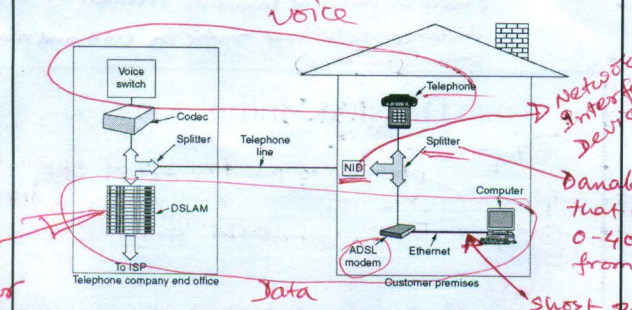
One for downstream control
One for upstream control
Rest for data
used for POTS (Plain Old Telephone Service)

Operation of ADSL using discrete multitone modulation.
Asymmetric
(DMT)

Bandwidth ⇒ Range of freq that passes through a medium with minimum attenuation (property of a medium)
Baud rate/Symbol rate ⇒ # of samples per sec
Bit rate ⇒ baud rate * (bits/symbol)
Determined by modulation technique

For new competetive LECs (CLEC), its toughl to connect, its new customers to its end office (Local Exching carriers)

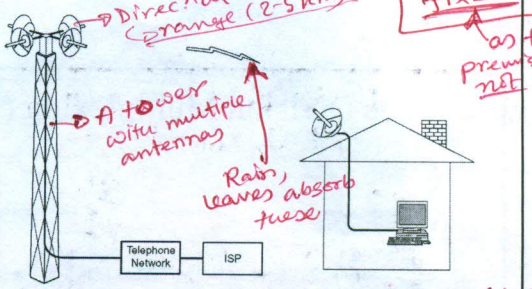
Digital Subscriber Lines (3)



A typical ADSL equipment configuration.

Voice
Network Interface Device
Analog filter that separates 0-4000 Hz voice from data
Short 2-node ethernet
Digital Subscriber Line Access Multiplexer
Data

Wireless Local Loops

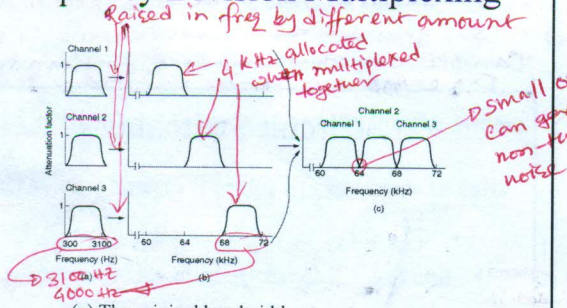


Architecture of an LMDS system.

connect between end office & customer
Directional (range (2-5 km))
A tower with multiple antennas
Rain, leaves absorb these
fixed wireless as the user premise does not move
BW → Asymmetric (Downstream 36 Gbps, Upstream 1 Mbps)
Local Multipoint Distribution Service
⇒ operates in millimeter wave
- 28-31 GHz in USA; 40 GHz in Europe
802.16 ⇒ Wireless MAN

Xmision of many signals over a single trunk ⇒ Multiplexing

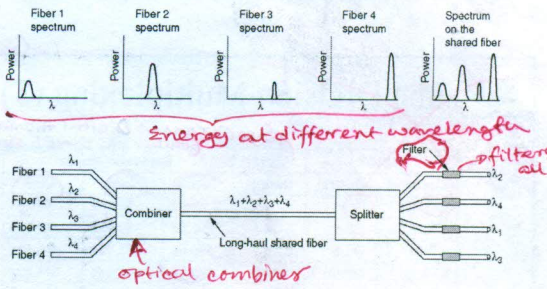
Frequency Division Multiplexing



(a) The original bandwidths.
(b) The bandwidths raised in frequency.
(b) The multiplexed channel.

Raised in freq by different amount
4 kHz allocated when multiplexed together
Small overlap. Can generate non-thermal noise
3100 Hz
4000 Hz
(3100 to 4000 Hz)

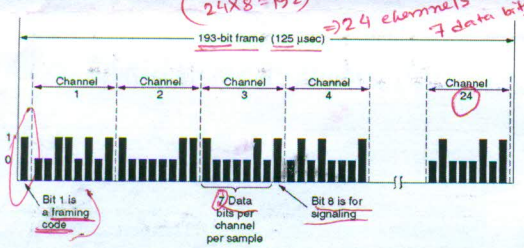
Wavelength Division Multiplexing



Wavelength division multiplexing.

filters out all wavelength but one

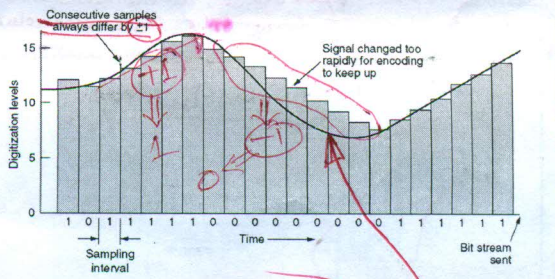
Time Division Multiplexing



The T1 carrier (1.544 Mbps).

(24×8 = 192) + 1 = 193 bit
⇒ 24 channels each having 7 data bits and one control bit
(used is USA and Japan)

Time Division Multiplexing (2)

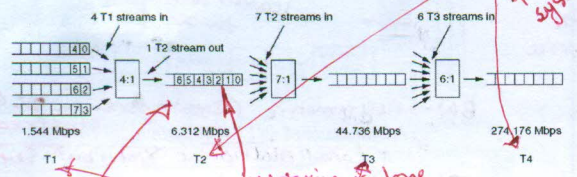


Delta modulation.

⇒ Differential Pulse Code Modulation: Not the digitized amplitude, but the difference between the current value and the previous one.
Variation → Delta modulation
- Require each sample to differ from its predecessor by either +1 or -1.
- A single bit can be transmitted telling whether the new sample is above or below the previous one.
- Get into trouble if signal changes too fast; Here, info gets lost.

SONET → Synchronous Optical NETwork
(used in physical layer of trunks of long distance telephone traffic in USA and many other countries)

Time Division Multiplexing (3)



Multiplexing T1 streams into higher carriers.

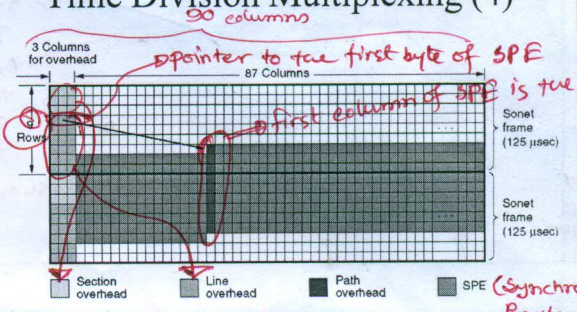
only used by the telephone system itself

4 × 1.544 = 6.176, Additional bits are used for framing and recovery; So, 6.312

Multiplexing is done bit by bit (Not byte by byte)

used by customers

Time Division Multiplexing (4)



pointer to the first byte of SPE

first column of SPE is the path overhead

(Synchronous Payload Envelope) — user data

Two back-to-back SONET frames.

Basis SONET channel called STS-1 (Synchronous Transport Signal-1)

⇒ 90 columns in 9 rows ⇒ 90 × 9 = 810 bytes = 8 × 810 bits = 6480 bits

⇒ 125 μs per frame ⇒ So, 8000 frames/sec (Property of PCM)

⇒ So, gross data rate = 8000 × 6480 = 51.84 Mbps

[an SPE can start anywhere, and can span two frames]

[Need not to wait for the next frame to begin]

Time Division Multiplexing (5)

Optical Carrier

(Synchronous Digital Hierarchy (similar to SONET standard))

SONET		SDH	Data rate (Mbps)		
Electrical	Optical	Optical	Gross	SPE	User
STS-1	OC-1		51.84	50.112	49.536
STS-3	OC-3	STM-1	155.52	150.336	148.608
STS-9	OC-9	STM-3	466.56	451.008	445.824
STS-12	OC-12	STM-4	622.08	601.344	594.432
STS-18	OC-18	STM-6	933.12	902.016	891.648
STS-24	OC-24	STM-8	1244.16	1202.688	1188.864
STS-36	OC-36	STM-12	1866.24	1804.032	1783.296
STS-48	OC-48	STM-16	2488.32	2405.376	2377.728
STS-192	OC-192	STM-64	9953.28	9621.504	9510.912

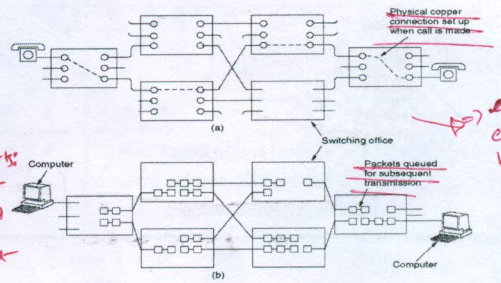
SONET and SDH multiplex rates.

⇒ includes all overhead

⇒ excludes line and section overhead

⇒ excludes all overheads and counts only 86 payload column

Circuit Switching

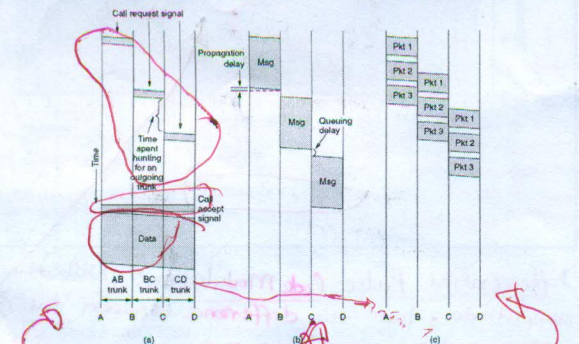


(a) Circuit switching.
(b) Packet switching.

phone systems are divided in two principal parts:
i) Outside plant (local loop and trunks)
ii) Inside plant (switches)

⇒ establish a conceptual conn. betn src & dest.

Message Switching



(a) Circuit switching (b) Message switching (c) Packet switching

Stores user data ~~when for next~~ in switching office and sent ~~by~~ msg by msg

Packet Switching

Item	Circuit-switched	Packet-switched
Call setup	Required	Not needed
Dedicated physical path	Yes	No
Each packet follows the same route	Yes	No
Packets arrive in order	Yes	No
Is a switch crash fatal	Yes	No
Bandwidth available	Fixed	Dynamic
When can congestion occur	At setup time	On every packet
Potentially wasted bandwidth	Yes	No
Store-and-forward transmission	No	Yes
Transparency	Yes	No
Charging	Per minute	Per packet

A comparison of circuit switched and packet-switched networks.

Hand off ⇒ Getting a new channel from the BS which get the strongest signal from the mobile phone.
→ Takes around 300 ms

8/26/2015

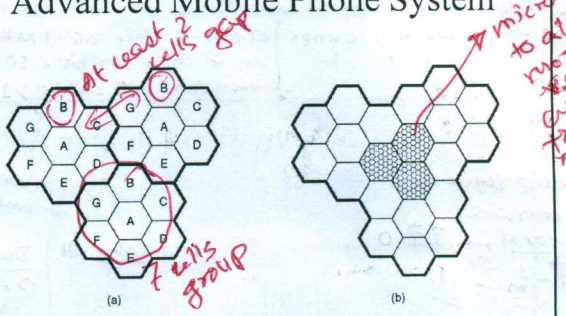
Soft handoff ⇒ Phone is acquired by the new BS before the previous one signs off.
Disadv → Phone needs to tune to two freqs at the same time.
Adv → No loss of continuity

hard handoff ⇒ Old BS drops the phone before the new one acquires

The Mobile Telephone System

- First-Generation Mobile Phones: Analog Voice
- Second-Generation Mobile Phones: Digital Voice
- Third-Generation Mobile Phones: Digital Voice and Data

Advanced Mobile Phone System



(a) Frequencies are not reused in adjacent cells.
(b) To add more users, smaller cells can be used.

AMPS ⇒ 832 full-duplex channels, each consisting of a pair of simplex channels. [FDM is used to separate channels]
Each of 30 KHz → 832 simplex tx channels from 824–849 MHz | 832 × 30 KHz
832 simplex receive channels from 869–894 MHz | ≈ 25 MHz

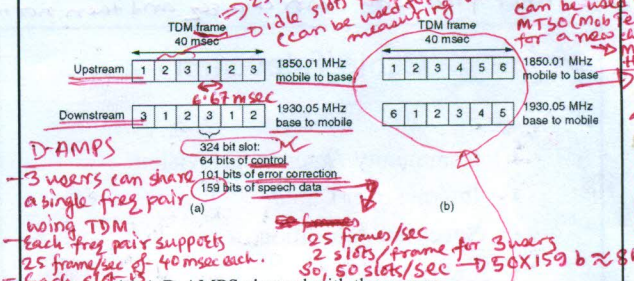
1st generation mobile ⇒ Analog
2nd " " ⇒ Digital → No standard (4 systems)
- D-AMPS
- GSM
- CDMA
- PDC

Channel Categories

The 832 channels are divided into four categories:

- Control (base to mobile) to manage the system
- Paging (base to mobile) to alert users to calls for them
- Access (bidirectional) for call setup and channel assignment
- Data (bidirectional) for voice, fax, or data

D-AMPS
Digital Advanced Mobile Phone System

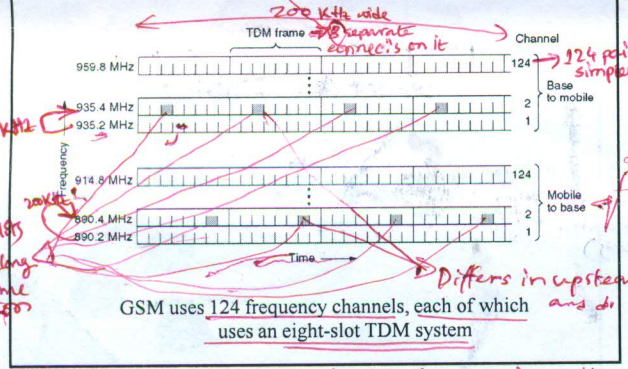


(a) A D-AMPS channel with three users.
(b) A D-AMPS channel with six users.

D-AMPS → USA & Japan
25 frames/sec
Idle slots used for measuring line quality
can be used to inform MTSO (Mob Telp switching Office) for a new channel.
Mobile assisted Handoff (MAHO)
4 kbps
as 25 slots per sec
3 users can share a single freq pair using TDM.
Each freq pair supports 25 frames/sec of 40 msec each.
Each slot is 6.67 msec
25 frames/sec
2 slots/frame for 3 users
So, 50 slots/sec → 50 × 159 b ≈ 8 kbps

D-AMPS ⇒ 30 KHz channel; 3 users;
GSM ⇒ 200 KHz channel; 8 users; higher data rate

GSM
Global System for Mobile Communications

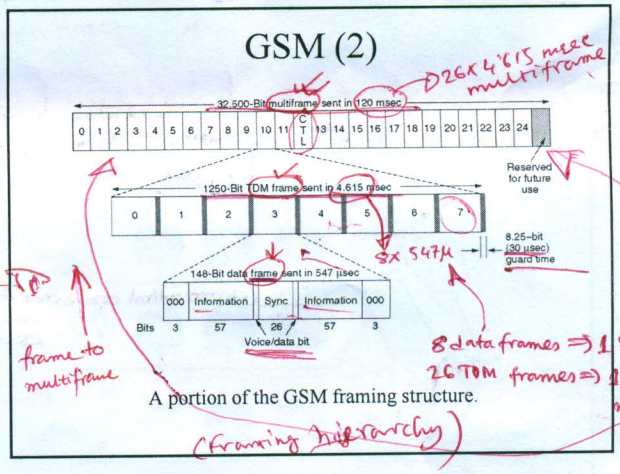


GSM uses 124 frequency channels, each of which uses an eight-slot TDM system

GSM ⇒ Global System for Mobile Comm's
almost everywhere in the world
200 KHz wide
TDM frame ⇒ 8 separate conn's on it
124 pairs of simplex channels
8 shaded slots belong to the same conn
Differs in upstream and do

124 × 8 = 992 channels theoretically. However, in practice, several channels are not available to avoid freq conflicts with neighboring cells.

GSM (2)



A portion of the GSM framing structure.

26 × 4.615 msec multiframe
8 × 547 μ
frame to multiframe
8 data frames ⇒ 1 TDM frame
26 TDM frames ⇒ 120 sec multiframe
(framing hierarchy)

- Each TDMA slot consists of a 148-bit data frame (3+57+1+26+1+57+3 = 148) that occupies the channel for 577 μs (including a 30 μs guard time after each slot)
- Each data frame starts and ends with three 0s, both at beginning and at end.
- 57 bits information each having 1 control bit that says whether the following info field is voice or data.
- 26 bits training for the receiver to synchronize with the sender's boundary

CDMA → Completely different from AMPS, D-AMPS, and GSM, as it allows each station to transmit over the entire frequency spectrum all the time.
- Multiple simultaneous tx are separated using coding theory.
- Relaxes the assumption that colliding frames are totally garbled.

⇒ Each bit time is divided into m short intervals called chips.
- Typical: 64 or 128 chips per bit (In example 8 chips/bit)

⇒ Each station is assigned a unique m-bit code called a chip sequence. To tx 1, it sends the chip seq; to tx 0, it sends one's complement of the chip seq

⇒ Example: 1 MHz band available for 100 users
FDM → 10 KHz, thus 10 Kbps per user
CDMA → chip rate 1 MHz, so, if chips per bit < 100 then data rate will be > 10 Kbps

CDMA – Code Division Multiple Access

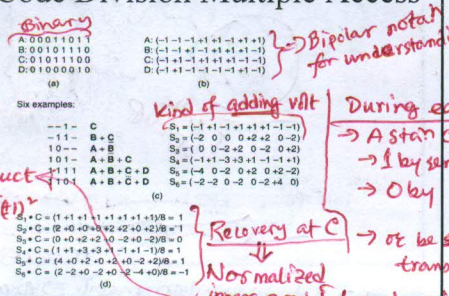
* All chip sequences are pairwise orthogonal, i.e., $\vec{S} \cdot \vec{T} = 0$

$$\vec{S} \cdot \vec{T} = \frac{1}{m} \sum_{i=1}^{m} S_i T_i = 0$$

Normalized inner product

$$\vec{S} \cdot \vec{S} = \frac{1}{m} \sum S_i S_i = \frac{1}{m} \sum (S_i)^2 = \frac{1}{m} \sum (\pm 1)^2 = 1$$

$$\vec{S} \cdot \vec{\bar{S}} = -1$$



(a) Binary chip sequences for four stations
(b) Bipolar chip sequences
(c) Six examples of transmissions
(d) Recovery of station C's signal

Bipolar notation for understanding

Kind of adding volt

During each bit time
→ A station can xmit
→ 1 by sending chip seq
→ 0 by a negative of chip seq
→ or be silent by transmitting nothing

Recovery at C → Normalized inner prod [Synchronization assumed]

General of orthogonal chip seq ⇒ Walsh codes
→ While doing so, inner prod of all other station's chip will give 0 as for being orthogonal. So, only part remains is to have inner prod with own chip seq and then normalization.

Third-Generation Mobile Phones: Digital Voice and Data

Basic services an IMT-2000 network should provide
(IMT-2000 → year freq of op.n (2000 MHz), Bw of service (2000 KHz?), International Mobile Telecomm.n)

- High-quality voice transmission
- Messaging (replace e-mail, fax, SMS, chat, etc.)
- Multimedia (music, videos, films, TV, etc.)
- Internet access (web surfing, w/multimedia.)

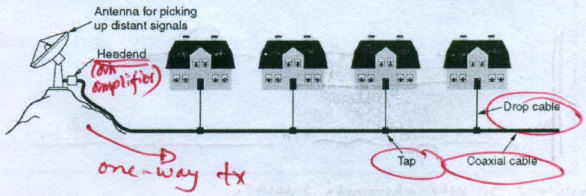
Noticing such actually happened in year 2000.

Cable Television

- Community Antenna Television
- Internet over Cable
- Spectrum Allocation
- Cable Modems
- ADSL versus Cable

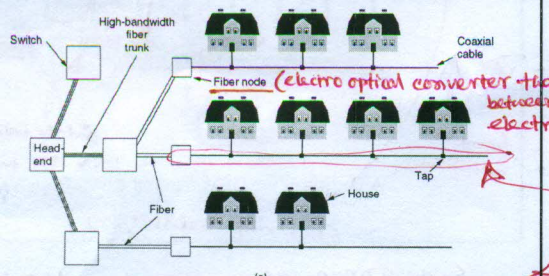
Community Antenna Television

⇒ For people living in rural or mountainous area



An early cable television system.

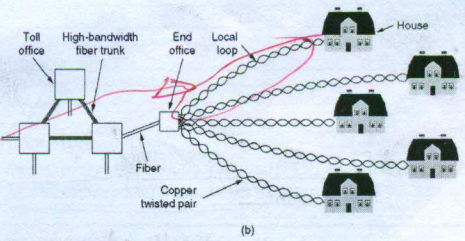
Internet over Cable



Cable television

Hybrid fiber Coax (HFC)

Internet over Cable (2)



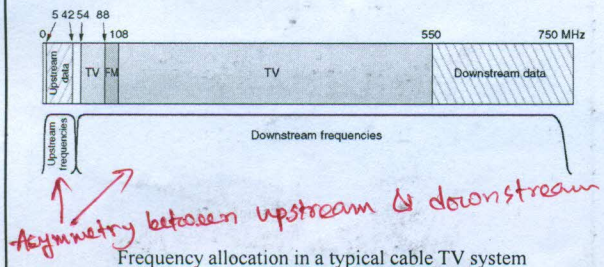
The fixed telephone system.

Technical challenges
① one-way amplifiers are needed to be replaced by two-way amplifiers
② Down in the neighborhoods, a single cable is shared by many houses, whereas in the telephone system, every house has its own local loop.
Sol.n: limit the # of subscribers on each cable through splitting up long cables and connect each one directly to a fiber node.
(# of hoses per cable → 500 ~ 2000 typical)

⇒ Cable operators wanted to get into the Internet access business and often the telephony business as well. However, there are some technical challenges

Spectrum Allocation

for Cable TV and Internet coexist simultaneously



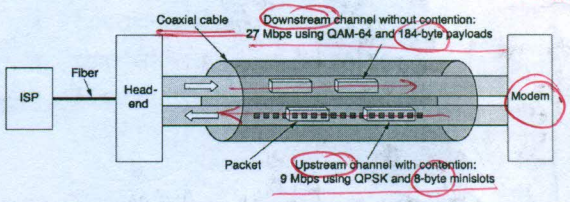
Asymmetry between upstream & downstream

Frequency allocation in a typical cable TV system used for Internet access

Cable Modems

⇒ for Internet access

Two interfaces
- One to computer
- One to cable network



Typical details of the upstream and downstream channels in North America.

	ADSL	Cable
Backbone	Fiber	Fiber
Edge	Twisted pair (less BW, but, less wastage)	Coax (more BW, but, more wastage)
Specific statement on BW	Yes	No (as ~~#~~ of users may vary)
Impact of increasing # of users	less	More
Availability	Everyone has a telephone but not all users are close to their end office to get ADSL	Not everyone has cable. But, if you have cable and the company provides Internet access, you can get it
Security	More (as has own cable)	Less (as shared cable)
Reliability	More (has backup power to continue work even during a power outage)	Less (if any amplifier down the chain fails, all downstream users are cut off instantaneously.)

comparable service and comparable price

Chapter 3

The Data Link Layer

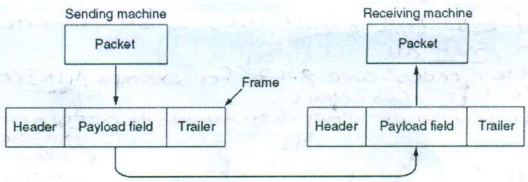
Data Link Layer Design Issues

- Services Provided to the Network Layer
- Framing
- Error Control
- Flow Control

Functions of the Data Link Layer

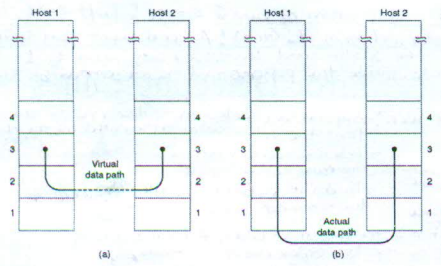
- Provide service interface to the network layer
- Dealing with transmission errors
- Regulating data flow
  - Slow receivers not swamped by fast senders

Functions of the Data Link Layer (2)



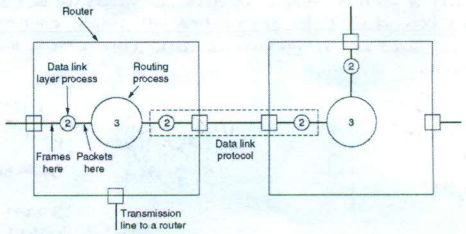
Relationship between packets and frames.

Services Provided to Network Layer



(a) Virtual communication.
(b) Actual communication.

Services Provided to Network Layer (2)



Placement of the data link protocol.

Framing ⇒ Breaking up bit streams into frames

↳ intuitive way like our ordinary text writing ⇒ insert time gaps between frames

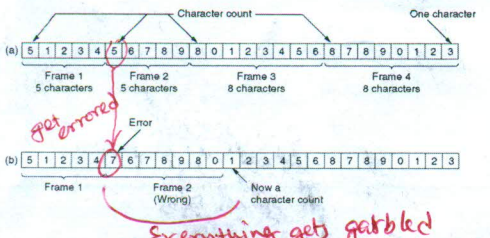
⇒ Problem: Networks rarely make any guarantees about timing, so it is possible these gaps might be squeezed out or other gaps might be inserted during transmission.

Remedy ⇒ four methods of framing

1. Character count
2. Flag bytes with byte stuffing
3. Starting and ending flags, with bit stuffing
4. Physical layer coding violations ⇒ If encoding in physical medium contains some redundancy. For example, in some network 1 data bit is encoded by 2 physical bits. Here, high-low transition refers to 1 and low-high refers to 0, high-high and low-low are unused ⇒ can be a delimiter

8/26

Framing



A character stream. (a) Without errors. (b) With one error.

got errored

Everything gets garbled (Synchronization gets lost but? sender & receiver)

⇒ Possible sol^n: Checksum → The receiver will only know that the frame is bad, but, no way to know where the next frame starts

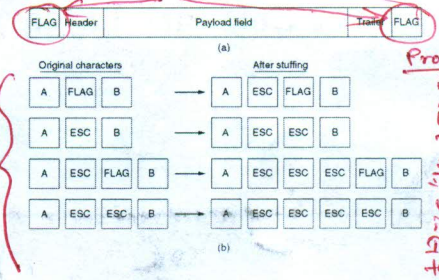
⇒ Another possible sol^n: Sending a frame back to the src asking for a retx → still no way to know how many bits to skip

Framing (2)

Byte stuffing or character stuffing ⇒ omits the synchronization problem

In past ⇒ Starting and ending bytes were different

Now ⇒ Both are same



(a) A frame delimited by flag bytes.

(b) Four examples of byte sequences before and after stuffing.

Problem: what happens if a flag appears in original data (may happen in binary files)

Sol^n: Insert a special "escape" character just before the flag when it appears in data. In case of destuffing, the extra added escape characters are removed.

New problem: what happens if the "escape" character itself appears in the data??

Sol^n: Again, it is stuffed and destuffed in the same way with an additional "escape" character.

Different situations can occur here.

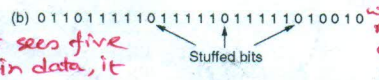
Framing (3)

Problem of byte stuffing ⇒ It uses 8-bit characters. However, not all character codes are 8-bit. For example, UNICODE uses 16-bit characters

Sol^n: Bit stuffing that allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per char.

Flag: 01111110

Stuffing: If sender sees five consecutive 1's in data, it automatically stuffs a 0 into the ongoing stream



Bit stuffing
(a) The original data.
(b) The data as they appear on the line.
(c) The data as they are stored in receiver's memory after destuffing.

Destuffing: If the recvr sees five consecutive 1's followed by a 0, it automatically destuffs or deletes the 0

01111110 → Stuffing (if in data) → 01111101010 → Destuffing → 01111110

Error Detection and Correction

- Error-Correcting Codes → For mostly highly noise medium (wireless)
- Error-Detecting Codes (⇒ Retx) → For mostly low-noise medium (copper, fiber).

Telephone system ⇒ i) switches, ii) Interoffice trunks, iii) local loops

Digital → Analog

⇒ Errors are more common in analog than digital. We may experience this long, as replacing local loops are highly costly. Also, wireless case is far more error prone. So, we need to know how to deal with them.

⇒ Errors may come in bursts or singly.

⇒ Burst: Gives errors in less # of packets

Ex: if error rate is 0.001 per bit and block size is 1000 bits, then most of the blocks will be errored for singly error. For burst errors of 100 bits, then 1 or 2 out of 100 packets will be in error

⇒ Problem of burst error: Difficult to correct

⇒ Error correction: To add redundant or check bits for m bits data. Total n bits. ($n = m+r$) → length of a code word.

⇒ Hamming distance: # of bit positions at which two code words differ.

⇒ To detect d errors, Hamming distance needs to be $(d+1)$

⇒ To correct d errors, Hamming distance needs to be $(2d+1)$

⇒ Calculation of lower bound on r for correcting all single errors

Each of 2^m legal messages has n illegal code words (can get it be inverting each bit pos^n) at a distance 1 from it.

So, each legal message (total 2^m) has $(n+1)$ bit patterns dedicated to it.

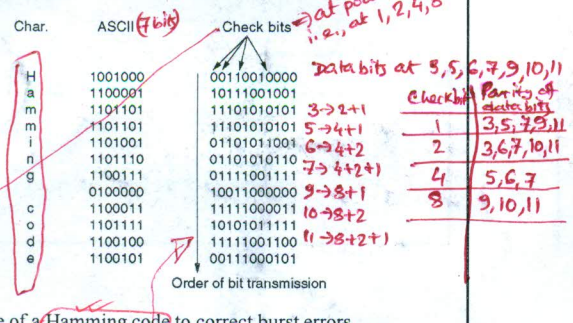
So, $(n+1)2^m \le 2^n$

so, $(m+r+1)2^m \le 2^{m+r}$

$\therefore (m+r+1) \le 2^r$

gives the lower limit on r.

Error-Correcting Codes



Check bit	Parity of data bits
1	3, 5, 7, 9, 11
2	3, 6, 7, 10, 11
4	5, 6, 7
8	9, 10, 11

Use of a Hamming code to correct burst errors.

* check bits ⇒ Give even parity: # of 1's needs to (including itself) be even

⇒ when a code word arrives at a receiver, it initialized a counter to zero. It then add the pos^n k of an errored check bit to the counter.

If counter = 0, then no error

If counter > 0, then the counter gives pos^n of the single error.

⇒ To handle burst error of at most k length, k consecutive code words are arranged as a matrix, one code word per row. Data is transmitted one column at a time. So, in case of k-bits error (in burst), * each column will be affected by only one bit. At the receiver, the matrix is reconstructed, one column at a time. ⇒ Needs (kr) check bits to make blocks of km data bits immune to a single burst error of length k or less.

Error-Detecting Codes

Basic idea: Append a checksum to the end of the frame in such a way that the polynomial represented by the checksummed frame is divisible by G(x)

– when a receiver gets the checksummed frame, it (receiver) divides it by G(x) and check the remainder whether it is 0 or not

(checksummed frame)

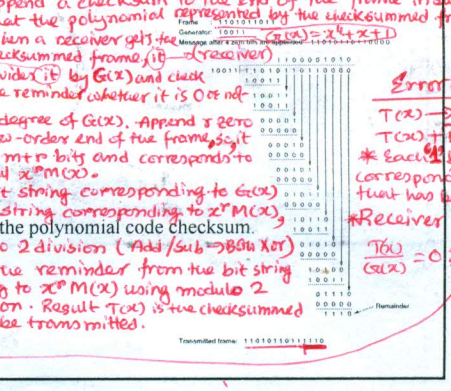
Algo:

1. Let r be the degree of G(x). Append r zero bits to the low-order end of the frame, so it now contains m+r bits and corresponds to the polynomial $x^r M(x)$.
2. Divide the bit string corresponding to G(x) into the bit string corresponding to $x^r M(x)$.

Calculation of the polynomial code checksum.

3. Subtract the reminder from the bit string corresponding to $x^r M(x)$ using modulo 2 subtraction. Result T(x) is the checksummed frame to be transmitted.

Using modulo 2 division (Add/Sub ⇒ Bitwise XOR)



Error detection

T(x) → Sent → no error

T(x) + E(x) → Received

* Each 1 bit in E(x) corresponds to a bit that has been inverted

* Receiver: $\frac{T(x)+E(x)}{G(x)}$

$\frac{T(x)}{G(x)} = 0$; $\frac{E(x)}{G(x)}$ reports error

Cond^n 1: Single-bit error: $E(x) = x^i$; Always detected if G(x) contains 2 or more terms.

Cond^n 2: Two isolated single bit errors: $E(x) = x^i + x^j$, where $i > j$

$E(x) = x^j (x^{(i-j)} + 1)$, ⇒ cond^n: G(x) does not divide $x^k + 1$ for any k up to the maximum value of i−j (i.e., up to the max frame length)

Ex: $x^{15} + x^{14} + 1$ does not divide $x^k + 1$ for $k < 32,768$. (as max frame length)

Cond^n 3: Odd # of bit errors: E(x) contains an odd # of terms.

⇒ No polynomial having an odd # of terms was $(x+1)$ as a factor in the modulo 2 system. So, by making $(x+1)$ as factor of G(x) we can catch all errors consisting an odd # of inverted bits

⇒ proof of no polynomial with an odd # of terms has $(x+1)$ as its factor in modulo 2 system.
→ Let the polynomial can have it. So, $E(x) = (x+1)\,Q(x)$; Now, if $x=1$, then $E(1) = (1+1)\,Q(1) = 0\,Q(1) = 0$
On the other hand, if $E(x)$ contains an odd # of terms putting $x=1$ will give $E(1) = 1$ ← contradiction!

Cond.4 : A polynomial code with r check bits detects all burst (consecutive) errors of length ≤ r.
→ A burst error of length k can be represented as $x^i(x^{k-1} + \dots + 1)$, where i determines how far from the right hand end of the received frame the burst is located. Now, if $G(x)$ contains an x^0 term, then it will not have x^i as a factor.
Besides, if the degree of the polynomial in parenthesized expression is less than the degree of $G(x)$, then the reminder can never be zero.
So, if the degree of the parenthesized expression is less than the degree of $G(x)$, i.e., r, then the the reminder can never be zero. So, it will never divide the error by $G(x)$ with degree ≤ r.

Elementary Data Link Protocols

- An Unrestricted Simplex Protocol
- A Simplex Stop-and-Wait Protocol
- A Simplex Protocol for a Noisy Channel

Protocol Definitions

```
#define MAX_PKT 1024                        /* determines packet size in bytes */

typedef enum {false, true} boolean;         /* boolean type */
typedef unsigned int seq_nr;                /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT];} packet;/*  packet definition */
typedef enum {data, ack, nak} frame_kind;   /* frame_kind definition */

typedef struct {                            /* frames are transported in this layer */
  frame_kind kind;                          /* what kind of a frame is it? */
  seq_nr seq;                               /* sequence number */
  seq_nr ack;                               /* acknowledgement number */
  packet info;                              /* the network layer packet */
} frame;
```

control fields (frame header)

Continued →

Some definitions needed in the protocols to follow.
These are located in the file protocol.h.

Protocol Definitions (ctd.)

```
/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);

/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
void enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void disable_network_layer(void);

/* Macro inc is expanded in-line: Increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

Some definitions needed in the protocols to follow. These are located in the file protocol.h.

seq #

Unrestricted Simplex Protocol

```
/* Protocol 1 (utopia) provides for data transmission in one direction only, from
   sender to receiver. The communication channel is assumed to be error free
   and the receiver is assumed to be able to process all the input infinitely quickly.
   Consequently, the sender just sits in a loop pumping data out onto the line as
   fast as it can. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                              /* buffer for an outbound frame */
  packet buffer;                        /* buffer for an outbound packet */

  while (true) {
      from_network_layer(&buffer);      /* go get something to send */
      s.info = buffer;                  /* copy it into s for transmission */
      to_physical_layer(&s);            /* send it on its way */
                                        * Tomorrow, and tomorrow, and tomorrow,
                                          Creeps in this petty pace from day to day
                                          To the last syllable of recorded time
                                            - Macbeth, V, v */
  }
}

void receiver1(void)
{
  frame r;
  event_type event;                     /* filled in by wait, but not used here */

  while (true) {
      wait_for_event(&event);           /* only possibility is frame_arrival */
      from_physical_layer(&r);          /* go get the inbound frame */
      to_network_layer(&r.info);        /* pass the data to the network layer */
  }
}
```

NL → PL

PL → NL

No flow control, No error control

Simplex Stop-and-Wait Protocol

```
/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be error
   free, as in protocol 1. However, this time, the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                              /* buffer for an outbound frame */
  packet buffer;                        /* buffer for an outbound packet */
  event_type event;                     /* frame_arrival is the only possibility */

  while (true) {
      from_network_layer(&buffer);      /* go get something to send */
      s.info = buffer;                  /* copy it into s for transmission */
      to_physical_layer(&s);            /* bye bye little frame */
      wait_for_event(&event);           /* do not proceed until given the go ahead */
  }
}

void receiver2(void)
{
  frame r, s;                           /* buffers for frames */
  event_type event;                     /* frame_arrival is the only possibility */
  while (true) {
      wait_for_event(&event);           /* only possibility is frame_arrival */
      from_physical_layer(&r);          /* go get the inbound frame */
      to_network_layer(&r.info);        /* pass the data to the network layer */
      to_physical_layer(&s);            /* send a dummy frame to awaken sender */
  }
}
```

flow control

like ack

A Simplex Protocol for a Noisy Channel

```
/* Protocol 3 (par) allows unidirectional data flow over an unreliable channel. */
#define MAX_SEQ 1                       /* must be 1 for protocol 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;            /* seq number of next outgoing frame */
  frame s;                              /* scratch variable */
  packet buffer;                        /* buffer for an outbound packet */
  event_type event;

  next_frame_to_send = 0;               /* initialize outbound sequence numbers */
  from_network_layer(&buffer);          /* fetch first packet */
  while (true) {
      s.info = buffer;                  /* construct a frame for transmission */
      s.seq = next_frame_to_send;       /* insert sequence number in frame */
      to_physical_layer(&s);            /* send it on its way */
      start_timer(s.seq);               /* if answer takes too long, time out */
      wait_for_event(&event);           /* frame_arrival, cksum_err, timeout */
      if (event == frame_arrival) {
          from_physical_layer(&s);      /* get the acknowledgement */
          if (s.ack == next_frame_to_send) {
              stop_timer(s.ack);        /* turn the timer off */
              from_network_layer(&buffer); /* get the next one to send */
              inc(next_frame_to_send);  /* invert next_frame_to_send */
          }
      }
  }
}
```

Continued →

A positive acknowledgement with retransmission protocol.

ACK and its processing

Retransmission

PAR (Positive Acknowledgement with Retransmission)
or ARQ (Automatic Repeat Request)

A Simplex Protocol for a Noisy Channel (ctd.)

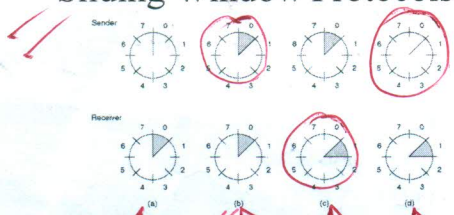
```
void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true) {
    wait_for_event(&event);          /* possibilities: frame_arrival, cksum_err */
    if (event == frame_arrival) {    /* a valid frame has arrived. */
      from_physical_layer(&r);       /* go get the newly arrived frame */
      if (r.seq == frame_expected) { /* this is what we have been waiting for. */
        to_network_layer(&r.info);   /* pass the data to the network layer */
        inc(frame_expected);         /* next time expect the other sequence nr */
      }
      s.ack = 1 - frame_expected;    /* tell which frame is being acked */
      to_physical_layer(&s);         /* send acknowledgement */
    }
  }
}
```

A positive acknowledgement with retransmission protocol.

Sliding Window Protocols

- A One-Bit Sliding Window Protocol
- A Protocol Using Go Back N
- A Protocol Using Selective Repeat

Sliding Window Protocols (2)



A sliding window of size 1, with a 3-bit sequence number.
(a) Initially
(b) After the first frame has been sent
(c) After the first frame has been received
(d) After the first acknowledgement has been received.

A One-Bit Sliding Window Protocol

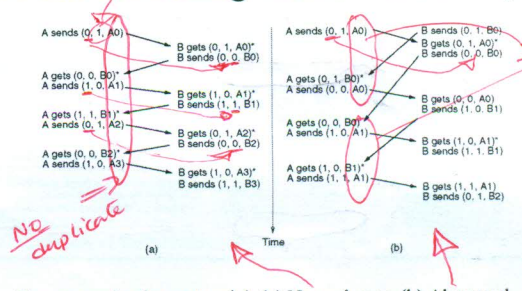
```
/* Protocol 4 (sliding window) is bidirectional. */
#define MAX_SEQ 1                    /* must be 1 for protocol 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
{
  seq_nr next_frame_to_send;         /* 0 or 1 only */
  seq_nr frame_expected;             /* 0 or 1 only */
  frame r, s;                        /* scratch variables */
  packet buffer;                     /* current packet being sent */
  event_type event;

  next_frame_to_send = 0;            /* next frame on the outbound stream */
  frame_expected = 0;                /* frame expected next */
  from_network_layer(&buffer);       /* fetch a packet from the network layer */
  s.info = buffer;                   /* prepare to send the initial frame */
  s.seq = next_frame_to_send;        /* insert sequence number into frame */
  s.ack = 1 - frame_expected;        /* piggybacked ack */
  to_physical_layer(&s);             /* transmit the frame */
  start_timer(s.seq);                /* start the timer running */
```

Continued →

A One-Bit Sliding Window Protocol (ctd.)

```
while (true) {
  wait_for_event(&event);            /* frame_arrival, cksum_err, or timeout */
  if (event == frame_arrival) {      /* a frame has arrived undamaged. */
    from_physical_layer(&r);         /* go get it */

    if (r.seq == frame_expected) {   /* handle inbound frame stream. */
      to_network_layer(&r.info);     /* pass packet to network layer */
      inc(frame_expected);           /* invert seq number expected next */
    }

    if (r.ack == next_frame_to_send) {  /* handle outbound frame stream. */
      stop_timer(r.ack);             /* turn the timer off */
      from_network_layer(&buffer);   /* fetch new pkt from network layer */
      inc(next_frame_to_send);       /* invert sender's sequence number */
    }
  }
  s.info = buffer;                   /* construct outbound frame */
  s.seq = next_frame_to_send;        /* insert sequence number into it */
  s.ack = 1 - frame_expected;        /* seq number of last received frame */
  to_physical_layer(&s);             /* transmit a frame */
  start_timer(s.seq);                /* start the timer running */
}
```

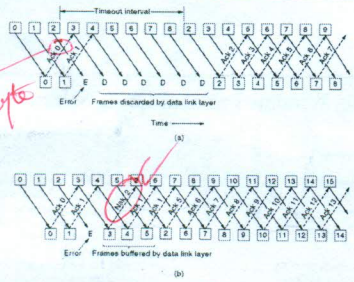
A One-Bit Sliding Window Protocol (2)



Two scenarios for protocol 4. (a) Normal case. (b) Abnormal case. The notation is (seq, ack, packet number). An asterisk indicates where a network layer accepts a packet.

In sessional

A Protocol Using Go Back N



last received byte

Pipelining and error recovery. Effect on an error when
(a) Receiver's window size is 1.
(b) Receiver's window size is large.

Sliding Window Protocol Using Go Back N

```
/* Protocol 5 (pipelining) allows multiple outstanding frames. The sender may transmit up
   to MAX_SEQ frames without waiting for an ack. In addition, unlike the previous protocols,
   the network layer is not assumed to have a new packet all the time. Instead, the
   network layer causes a network_layer_ready event when there is a packet to send. */

#define MAX_SEQ 7                       /* should be 2^n - 1 */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Return true if a <=b < c circularly; false otherwise. */
 if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
     return(true);
 else
     return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/* Construct and send a data frame. */
 frame s;                               /* scratch variable */

 s.info = buffer[frame_nr];             /* insert packet into frame */
 s.seq = frame_nr;                      /* insert sequence number into frame */
 s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);/* piggyback ack */
 to_physical_layer(&s);                 /* transmit the frame */
 start_timer(frame_nr);                 /* start the timer running */
}
```

Continued →

Sliding Window Protocol Using Go Back N

```
void protocol5(void)
{
 seq_nr next_frame_to_send;         /* MAX_SEQ > 1; used for outbound stream */
 seq_nr ack_expected;               /* oldest frame as yet unacknowledged */
 seq_nr frame_expected;             /* next frame expected on inbound stream */
 frame r;                           /* scratch variable */
 packet buffer[MAX_SEQ + 1];        /* buffers for the outbound stream */
 seq_nr nbuffered;                  /* # output buffers currently in use */
 seq_nr i;                          /* used to index into the buffer array */
 event_type event;

 enable_network_layer();            /* allow network_layer_ready events */
 ack_expected = 0;                  /* next ack expected inbound */
 next_frame_to_send = 0;            /* next frame going out */
 frame_expected = 0;                /* number of frame expected inbound */
 nbuffered = 0;                     /* initially no packets are buffered */
```

Continued →

Sliding Window Protocol Using Go Back N

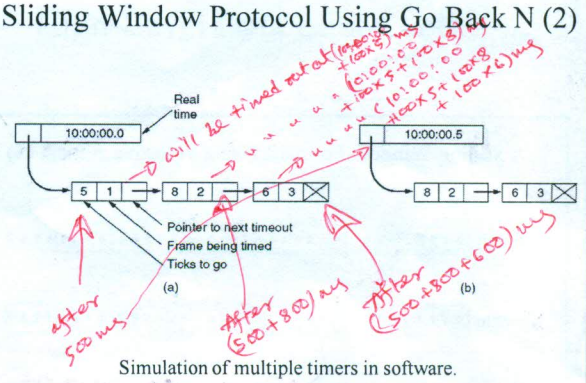
```
while (true) {
 wait_for_event(&event);            /* four possibilities: see event_type above */

 switch(event) {
  case network_layer_ready:         /* the network layer has a packet to send */
       /* Accept, save, and transmit a new frame. */
       from_network_layer(&buffer[next_frame_to_send]); /* fetch new packet */
       nbuffered = nbuffered + 1;   /* expand the sender's window */
       send_data(next_frame_to_send, frame_expected, buffer);/* transmit the frame */
       inc(next_frame_to_send);     /* advance sender's upper window edge */
       break;

  case frame_arrival:               /* a data or control frame has arrived */
       from_physical_layer(&r);     /* get incoming frame from physical layer */

       if (r.seq == frame_expected) {
            /* Frames are accepted only in order. */
            to_network_layer(&r.info);  /* pass packet to network layer */
            inc(frame_expected);  /* advance lower edge of receiver's window */
       }
```

Continued →

Sliding Window Protocol Using Go Back N

```
            /* Ack n implies n - 1, n - 2, etc.  Check for this. */
            while (between(ack_expected, r.ack, next_frame_to_send)) {
                 /* Handle piggybacked ack. */
                 nbuffered = nbuffered - 1;  /* one frame fewer buffered */
                 stop_timer(ack_expected); /* frame arrived intact; stop timer */
                 inc(ack_expected);     /* contract sender's window */
            }
            break;

       case cksum_err: break;           /* just ignore bad frames */

       case timeout:                    /* trouble; retransmit all outstanding frames */
            next_frame_to_send = ack_expected;   /* start retransmitting here */
            for (i = 1; i <= nbuffered; i++) {
                 send_data(next_frame_to_send, frame_expected, buffer);/* resend 1 frame */
                 inc(next_frame_to_send);  /* prepare to send the next one */
            }

       }

       if (nbuffered < MAX_SEQ)
            enable_network_layer();
       else
            disable_network_layer();
  }
 }
```

Sliding Window Protocol Using Go Back N (2)



Simulation of multiple timers in software.

⇒ Because Protocol 5 has multiple outstanding frames, it logically needs multiple timers, one per outstanding frame. However, all the timers can be simulated in s/w using a single h/w clock that causes interrupt periodically. Pending timeouts form a linked list telling the number of clock ticks (at every 500 ms in the example) until the timer expires, the frame being timed, and a pointer to the next node.

Sessional

A Sliding Window Protocol Using Selective Repeat

```
/* Protocol 6 (nonsequential receive) accepts frames out of order, but passes packets to the
   network layer in order. Associated with each outstanding frame is a timer. When the timer
   expires, only that frame is retransmitted, not all the outstanding frames, as in protocol 5. */

#define MAX_SEQ 7                       /* should be 2^n – 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;
#include "protocol.h"
boolean no_nak = true;                  /* no nak has been sent yet */
seq_nr oldest_frame = MAX- SEQ + 1;     /* initial value is only for the simulator */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Same as between in protocol5, but shorter and more obscure. */
  return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/* Construct and send a data, ack, or nak frame. */
  frame s;                              /* scratch variable */

  s.kind = fk;                          /* kind == data, ack, or nak */
  if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
  s.seq = frame_nr;                     /* only meaningful for data frames */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
  if (fk == nak) no_nak = false;        /* one nak per frame, please */
  to_physical_layer(&s);                /* transmit the frame */
  if (fk == data) start_timer(frame_nr % NR_BUFS);
  stop_ack_timer();                     /* no need for separate ack frame */
}
```

Continued →

A Sliding Window Protocol Using Selective Repeat (2)

```
void protocol6(void)
{
  seq_nr ack_expected;                  /* lower edge of sender's window */
  seq_nr next_frame_to_send;            /* upper edge of sender's window + 1 */
  seq_nr frame_expected;                /* lower edge of receiver's window */
  seq_nr too_far;                       /* upper edge of receiver's window + 1 */
  int i;                                /* index into buffer pool */
  frame r;                              /* scratch variable */
  packet out_buf[NR_BUFS];              /* buffers for the outbound stream */
  packet in_buf[NR_BUFS];               /* buffers for the inbound stream */
  boolean arrived[NR_BUFS];             /* inbound bit map */
  seq_nr nbuffered;                     /* how many output buffers currently used */
  event_type event;

  enable_network_layer();               /* initialize */
  ack_expected = 0;                     /* next ack expected on the inbound stream */
  next_frame_to_send = 0;               /* number of next outgoing frame */
  frame_expected = 0;
  too_far = NR_BUFS;
  nbuffered = 0;                        /* initially no packets are buffered */
  for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
```

Continued →

A Sliding Window Protocol Using Selective Repeat (3)

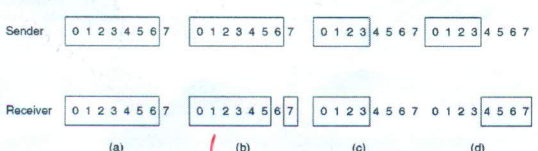
```
while (true) {
  wait_for_event(&event);               /* five possibilities: see event_type above */
  switch(event) {
    case network_layer_ready:           /* accept, save, and transmit a new frame */
         nbuffered = nbuffered + 1;     /* expand the window */
         from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* fetch new packet */
         send_frame(data, next_frame_to_send, frame_expected, out_buf);/* transmit the frame */
         inc(next_frame_to_send);       /* advance upper window edge */
         break;

    case frame_arrival:                 /* a data or control frame has arrived */
         from_physical_layer(&r);       /* fetch incoming frame from physical layer */
         if (r.kind == data) {
              /* An undamaged frame has arrived. */
              if ((r.seq != frame_expected) && no_nak)
                   send_frame(nak, 0, frame_expected, out_buf); else start_ack_timer();
              if (between(frame_expected, r.seq, too_far) && (arrived[r.seq%NR_BUFS] == false)) {
                   /* Frames may be accepted in any order. */
                   arrived[r.seq % NR_BUFS] = true;     /* mark buffer as full */
                   in_buf[r.seq % NR_BUFS] = r.info;    /* insert data into buffer */
                   while (arrived[frame_expected % NR_BUFS]) {
                        /* Pass frames and advance window. */
                        to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                        no_nak = true;
                        arrived[frame_expected % NR_BUFS] = false;
                        inc(frame_expected);    /* advance lower edge of receiver's window */
                        inc(too_far);           /* advance upper edge of receiver's window */
                        start_ack_timer();      /* to see if a separate ack is needed */
                   }
              }
```

Continued →

A Sliding Window Protocol Using Selective Repeat (4)

```
         if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1),next_frame_to_send))
              send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);

         while (between(ack_expected, r.ack, next_frame_to_send)) {
              nbuffered = nbuffered - 1;        /* handle piggybacked ack */
              stop_timer(ack_expected % NR_BUFS);    /* frame arrived intact */
              inc(ack_expected);                /* advance lower edge of sender's window */
         }
         break;
    case cksum_err:
         if (no_nak) send_frame(nak, 0, frame_expected, out_buf);/* damaged frame */
         break;
    case timeout:
         send_frame(data, oldest_frame, frame_expected, out_buf);/* we timed out */
         break;
    case ack_timeout:
         send_frame(ack,0,frame_expected, out_buf);     /* ack timer expired; send ack */
    }
    if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
  }
}
```

A Sliding Window Protocol Using Selective Repeat (5)



If the ack sent by the receiver gets lost here, then and then the sender sends pckt 0 after timeout, then there might arise a problem. Here the receiver will send an ack 6 thinking that 0 to 6 have already been sent

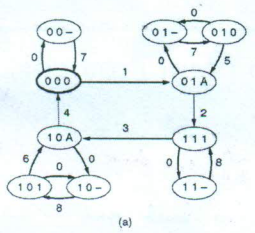
(a) Initial situation with a window size seven.
(b) After seven frames sent and received, but not acknowledged.
(c) Initial situation with a window size of four.
(d) After four frames sent and received, but not acknowledged.

(This happens as the prev acks got lost and the sender has no way to understand which pckt 0 (either new or old) has been acked). This problem arises as there is an overlap between earlier window and new window when an window advances. Its remedy is to make sure that there is no such overlap. To do so, the window size needs to be $\frac{MAX\_SEQ + 1}{2}$.

Protocol Verification

- Finite State Machined Models
- Petri Net Models

Finite State Machined Models

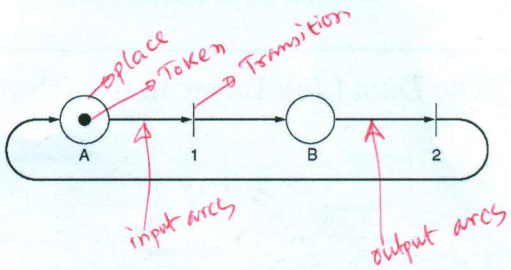


Transition	Who runs?	Frame accepted	Frame emitted	To network layer
		(frame lost)		
0	–			–
1	R	0	A	Yes
2	S	A	1	–
3	R	1	A	Yes
4	S	A	0	–
5	R	0	A	No
6	R	1	A	No
7	S	(timeout)	0	–
8	S	(timeout)	1	–

(b)

(a) State diagram for protocol 3. (b) Transmissions.

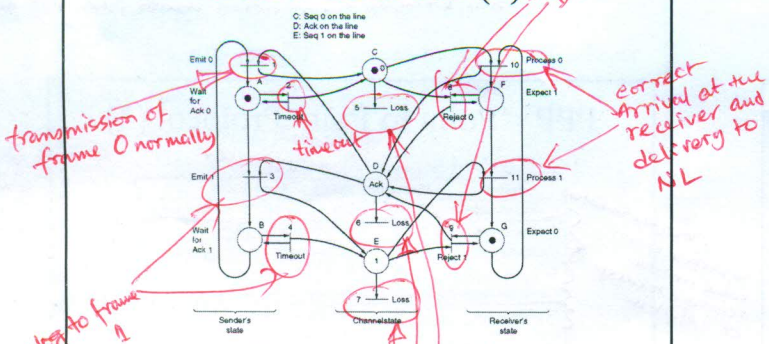
Petri Net Models



A Petri net with two places and two transitions.

place ⇒ represents a state, which the system may be in
Token ⇒ indicates where the system currently in
Transition ⇒ has o/more input arcs, and o/more output arcs
⇒ A transition is enabled if there is at least one input token in each of its input places. Any enabled transition may fire at will, removing one token from each input place and depositing a token in each output place. If # of input and output arcs differs, tokens will not be conserved.

Petri Net Models (2)



A Petri net model for protocol 3.

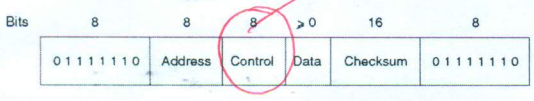
Data frame 0 with wrong seq #
Data ⇒
transmission of frame 0 normally
timeout
correct arrival at the receiver and delivery to NL
Analogy to frame 1
loss of frame 0
loss of ack
loss of frame 1

Example Data Link Protocols

- HDLC – High-Level Data Link Control
- The Data Link Layer in the Internet

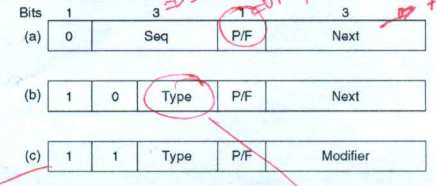
⇒ Both ⇒ bit-oriented
⇒ use bit stuffing

High-Level Data Link Control



Frame format for bit-oriented protocols.

High-Level Data Link Control (2)



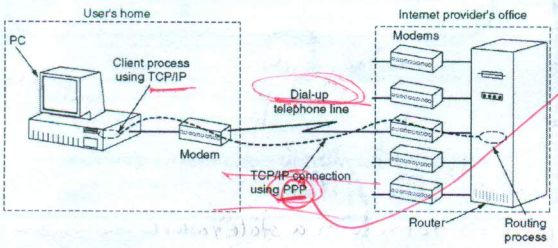
Control field of
(a) An information frame.
(b) A supervisory frame.
(c) An unnumbered frame.

3 kinds of frames

⇒ 3 bits seq #
⇒ P/F ⇒ Poll/final
⇒ polls a group of terminals when set. Remains set to P except for the last (or final) frame, which is set to F.
Piggybacked ACK (next frame expected)
Dissimilar to the protocols, which say the last frame received correctly
0 → Ack or acknowledgment
1 → Reject (NAK) (similar to Protocol 5)
2 → Receive Not Ready
3 → Selective Reject (similar to Protocol 6)

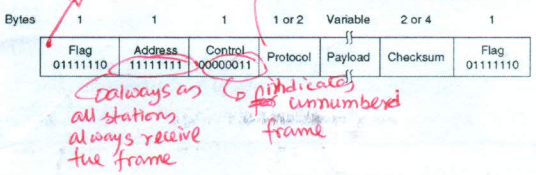
⇒ can be used for both carrying data for unreliable connection and for control purpose.

The Data Link Layer in the Internet



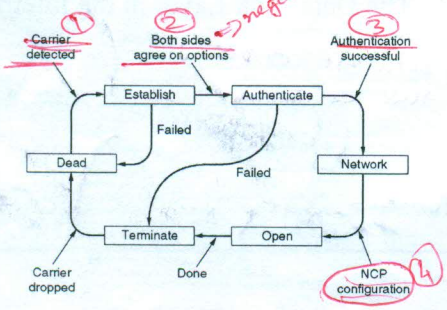
A home personal computer acting as an internet host.

PPP – Point to Point Protocol



The PPP full frame format for unnumbered mode operation.

PPP – Point to Point Protocol (2)



A simplified phase diagram for bring a line up and down.

PPP – Point to Point Protocol (3)

Name	Direction	Description
Configure-request	I → R	List of proposed options and values
Configure-ack	I ← R	All options are accepted
Configure-nak	I ← R	Some options are not accepted
Configure-reject	I ← R	Some options are not negotiable
Terminate-request	I → R	Request to shut the line down
Terminate-ack	I ← R	OK, line shut down
Code-reject	I ← R	Unknown request received
Protocol-reject	I ← R	Unknown protocol requested
Echo-request	I → R	Please send this frame back
Echo-reply	I ← R	Here is the frame back
Discard-request	I → R	Just discard this frame (for testing)

The LCP frame types. (11 frames)

Chapter 4

The Medium Access Control Sublayer

LLC / MAC } DLL

MAC → Sublayer of DLL

The Channel Allocation Problem

- Static Channel Allocation in LANs and MANs
- Dynamic Channel Allocation in LANs and MANs

Dedicated resource (channel for FDM, time for TDM) for a user

No dedicated resource. Here, all frames can be assumed to be somehow magically arranged orderly in a big central queue.

Let, mean time delay → T, channel capacity C bps, arrival rate λ frames/s, frame length follows an exponential probability density function with mean $1/\mu$ bits/frame. Now, from queuing theory, it can be shown for Poisson arrival and service time, $T = \frac{1}{\mu C - \lambda}$

$1/\mu$ bits → 1 frame
1 bit → μ u
C u → $C\mu$ u (service rate)

Ex: C = 100 Mbps, $1/\mu$ = 10,000 bits, λ = 5000 frames/s
Wrong ⇒ $T = \frac{1}{\mu C}$ = 100 μsec ← holds only if no contention
Right ⇒ $T = \frac{1}{\mu C - \lambda}$ = 200 μsec

FDM

Let, N independent channels, so, capacity = C/N, mean input rate = λ/N

So, $T_{static} = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = N T_{dynamic}$

So, delay using FDM is N times worse than ~~static~~ dynamic case
[similar logic holds for TDM]

Dynamic Channel Allocation in LANs and MANs

1. Station Model. → N independent stations/terminals
2. Single Channel Assumption. → A single channel is available
3. Collision Assumption. → Simultaneous tx of two frames makes a collission
4. (a) Continuous Time. → Frame tx can begin at any time
   (b) Slotted Time. → Time is divided into discrete intervals (slots). Frame tx begins always at the start of a slot.
5. (a) Carrier Sense. → Stn. assesses the channel for being idle before using it
   (b) No Carrier Sense. → No sensing

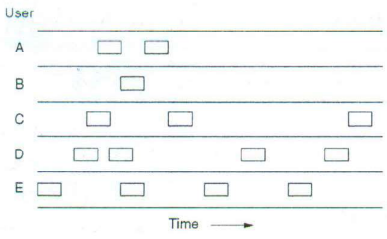
Multiple Access Protocols

- ALOHA < Pure, Slotted
- Carrier Sense Multiple Access Protocols < Persistent, Non persistent, CSMA with collision detection
- Collision-Free Protocols < Bitmap, Binary countdown
- Limited-Contention Protocols – Adaptive treewalk
- Wavelength Division Multiple Access Protocols
- Wireless LAN Protocols < MACA, MACAW

Norman Abramson (1970) © University of Hawaii

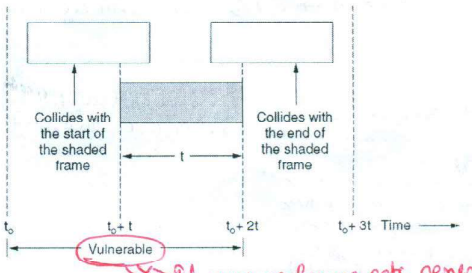
Pure ALOHA

⇒ let users transmit whenever they have data to be sent



In pure ALOHA, frames are transmitted at completely arbitrary times.

Pure ALOHA (2)



Vulnerable period for the shaded frame.

⇒ If any frame gets generated within this time (2 frame time), then there will be a collission.

G → mean # of attempts } per frame time [G ≥ N]
N → mean # of generations

N > 1 : Almost always collission; | Low load: N ≈ 0; so, G ≈ N
0 < N < 1 : Reasonable throughput. | High load: G > N

* Throughput = offered load * prob of a tx succeeding = G * P_0
* Prob that k frames are generated during a given frame time is given by the Poisson distribution: $Pr[k] = \frac{G^k e^{-G}}{k!}$

⇒ So, prob of 0 frame = $P_0[0] = e^{-G}$
Now, vulnerable time period is two frame long, so, here mean # of frames generated is 2G.
* So, prob of no frame during the vulnerable period is $P_0 = e^{-2G}$

So, throughput, $S = G \cdot P_0 = G \cdot e^{-2G}$
for max S, G = 0.5; $S = 0.5 * e^{-2 \cdot 0.5} = 0.184$
So, best channel utilization = 18.4%.

For max S, $\frac{dS}{dG} = 0$; So, $e^{-2G} + G(-2)e^{-2G} = 0$
So, $1 - 2G = 0$; So, $G = 1/2 = 0.5$

Slotted ALOHA ⇒ Time is divided into some discrete intervals. A station is not permitted to send whenever a frame is generated, rather it is required to wait for the beginning of the next slot. So, now the vulnerable period becomes one frame slot.

So, $Pr(0) = \frac{G^0 \cdot e^{-G}}{0!} = e^{-G}$; So, $S = G \cdot P_0 = G \cdot e^{-G}$

For max S, $\frac{dS}{dG} = 0$; So, $e^{-G} + (-1)Ge^{-G} = 0$; So, $1 - G = 0$; So, $G = 1$

So, $S_{max} = 1 \cdot e^{-1} = 0.368$; ⇒ So, for slotted ALOHA, the best we can hope for is

37% slots empty, 37% success, and 26% collisions. ($P_0 = e^{-G}$; $S = G \cdot e^{-G}$; $100 - 37 - 37 = 26$)

Now, prob of avoiding collision $= e^{-G}$; So, prob. of collision $= 1 - e^{-G}$

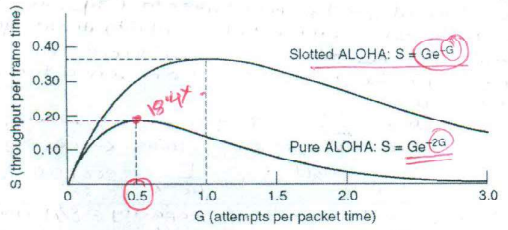
Therefore, prob of a tx requires exactly k attempts $= e^{-G}(1 - e^{-G})^{k-1}$

So, expected # of tx attempts ⇒ $E = \sum_{k=1}^{\infty} k P_k = \sum_{k=1}^{\infty} k e^{-G}(1 - e^{-G})^k = e^{G}$

$\frac{1}{1-(1-e^{-G})} = \frac{1}{e^{-G}} = e^{G}$

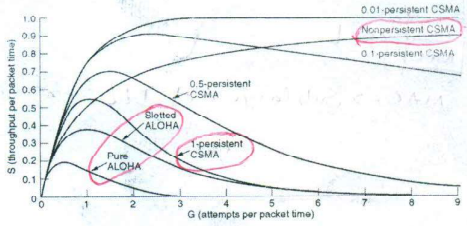
The exponential dependence indicates that small increase in channel load (G) drastically reduces its performance.

Pure ALOHA (3)



Throughput versus offered traffic for ALOHA systems.

Persistent and Nonpersistent CSMA



Comparison of the channel utilization versus load for various random access protocols.

Carrier sense protocols: Stations listen for a carrier and act accordingly.

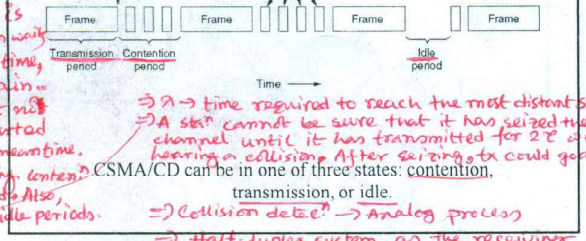
1-persistent: Senses carrier to be found it to be idle. If it is found to be idle, then a stn transmits its frame (prob of tx sensing is 1). If a second stn senses a channel to be idle when the first station's tx is yet to be received, then there can be a chance of experiencing collision (if the second one transmits). If a stn experiences a collision after sending its packet, then it waits for a random amount of time, and then starts all over again.

If the propagn delay is larger, then the chance of collision is higher. Even if the propn delay is zero, collision may happen in case of attempts at the beginning of a slot.

Non-persistent ⇒ In case of sensing a busy channel, waits for a random time. If the channel is idle, then it starts transmitting.

p-persistent ⇒ Uses slotted channels. If a stn becomes ready to send, it senses the channel. If the channel is found to be idle, then the stn txs with prob p or defers until beginning of the next slot by prob $q = 1 - p$. If next slot is idle, then it either txs or defers by prob p or q..... If the channel is found busy, it waits until the next slot and applies the above algo.

CSMA with Collision Detection

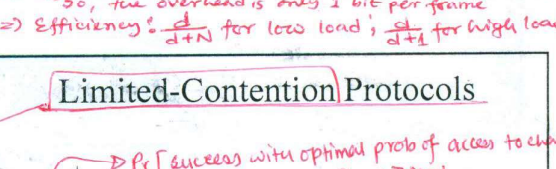


CSMA/CD can be in one of three states: contention, transmission, or idle.

CSMA/CD

- tx stops as soon as a collision is detected. Then wait for a random time, and try again, assuming that no other stn started tx in the meantime.
- ⇒ Alternating contention and tx periods. Also, there can be idle periods.
- ⇒ A → time required to reach the most distant stn
- ⇒ A stn cannot be sure that it has seized the channel until it has transmitted for 2τ without hearing a collision. After seizing, tx could go at any rate.
- ⇒ Collision detectn → Analog process
- ⇒ Half-duplex system, as the receiving line is busy for idle listening to detect collision.
- Bad performance for large 2τ; This can happen for long cable, small frame.
- ⇒ Problem of CSMA/CD → collision can still occur in contention period.

Collision-Free Protocols



The basic bit-map protocol.

⇒ Reservn protocol: Broadcasts desire to tx before actual tx

⇒ Each contention period consists of exactly N slots, 1 for each stn. A stn inserts a 1 to its allocated slot, if it has a frame to tx

⇒ After the contenn slots get over, each stn has a complete knowledge of which stns wish to tx. They begin tx in numeric order

⇒ low load: high #ed stns need to wait for 0.5N slots on an avg

(low #ed stns ⇒ 1.5N)

Considering that a stn gets ready in the middle of contenn period, so, stns need to wait for N slots on an avg ← overhead

⇒ High load: All stns mostly have something to send all the time; So, the overhead is only 1 bit per frame

⇒ Efficiency: $\frac{d}{d+N}$ for low load; $\frac{d}{d+1}$ for high load [d → # of data bit]

Collision-Free Protocols (2)

⇒ High priority for higher #ed stns

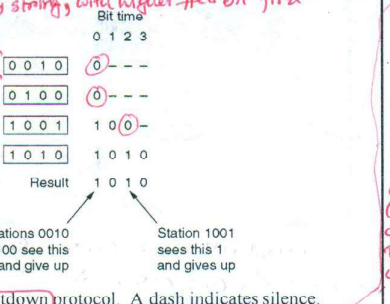
⇒ Broadcasts address as a binary string, with higher-#ed bit first

⇒ Equal length for all addresses

⇒ A stn giving up broadcasting its intent through broadcasting its binary bit string address, once it sees a higher-order bit posn set by someone else

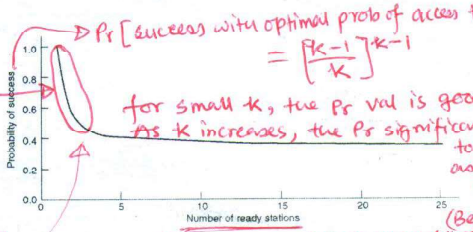
⇒ If the first field of a frame could be made address, then efficiency will be 100%.

⇒ Parallel interface: Successful stns get circularly permuted in order to give higher priorities to stns that have been silent unusually long.



The binary countdown protocol. A dash indicates silence.

Limited-Contention Protocols



Acquisition probability for a symmetric contention channel

Pr [success with optimal prob of access to channel] $= \left[\frac{k-1}{k}\right]^{k-1}$

for small k, the Pr val is good. As k increases, the Pr significantly decreases towards an asymptotic value of $\frac{1}{e}$ (Best case for slotted ALOHA)

Tries to operate here

To do so, stns are divided into (not necessarily disjoint) groups. Then, groups follow collision free protocol among them. On the other hand, stns within a group follow collision free protocol.

⇒ Performance measure: ① delay for low-load, ② Efficiency for high load

→ low-load: Pure / Slotted ALOHA

→ high-load: Collision-free protocol (as efficiency higher)

⇒ P: Prob of channel acquire/access; K: # of stns contending for channel

Pr [some stn successfully acquire the channel] $= {}^{k}C_{1} P(1-P)^{k-1}$

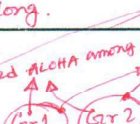
For best performance or optimal p, $\frac{dPr}{dP} = 0$; So, $K(1-P)^{K-1} - K(K-1)P(1-P)^{K-2} = 0$

So, $K(1-P) - K(K-1)P = 0$; $\therefore K - KP - K^2P + KP = 0$; $\therefore P = 1/K$

So, Pr [success with optimal P] $= K \cdot \frac{1}{K}\left(1 - \frac{1}{K}\right)^{K-1} = \left(\frac{K-1}{K}\right)^{K-1}$

Slotted ALOHA among stns ⇒ contention protocol

Bitmap or Binary countdown ⇒ contention-free protocol



Extreme cases

① 1 stn per group: Binary countdown/bitmap

② 1 gr with all stns: Slotted ALOHA