Overview of Computer Networking

What is Computer Networking?

Logical separation of tasks in digital systems

Computation: Local operations (ALU, load, store, branch, OS, …)

Communication: Data exchange between computation units





```
Local computation
  Request information  --communication-->  Accept request
                                            Process request
                                            Local computation
  Receive information  <--communication--   Send response
Local computation
```


What is Computer Networking?

Logical separation of tasks in a digital system

Computation: Local operations (ALU, load, store, branch, OS, …)

Communication: Data exchange between computation units



Making this work

Rules — lots of rules!

Special hardware

Special software



```
Local computation
  Request information  --communication-->  Accept request
                                            Process request
                                            Local computation
  Receive information  <--communication--   Send response
Local computation
```


Approaches to Networking

What's required

Understanding how people and machines communicate

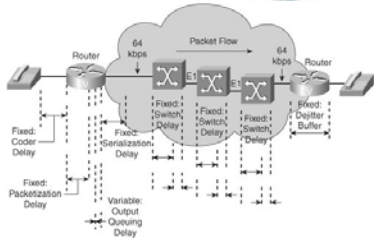


What's technically possible

Network topology (graph theory)

Message encoding (information theory)

Speed and delay (performance theory)



Historical engineering solutions

Division of labor

Hierarchy (top-down)

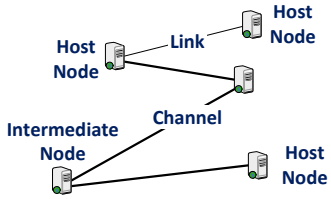
Security





Topology

Computer network as directed or undirected graph



Node

- Host node
  - Network edge — user systems
  - Computer, workstation, …
- Intermediate node
  - Hardware/software systems for data communication
  - Modem, hub, switch, concentrator, multiplexor, router, …

Link

- Transmission path between neighboring nodes

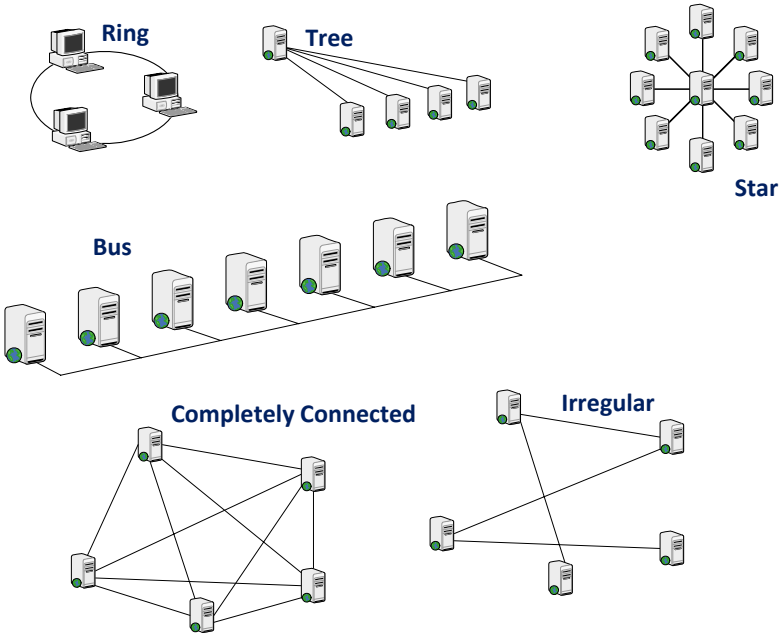
Hop

- Data transfer between neighboring nodes over one link

Channel

- Transmission path between nodes
- May include intermediate nodes

Network Topologies



How People (and Machines) Communicate



Requirements

- Language
- Medium
- Names
- Rules of conversation (protocols)

Preferences

- Keep it simple
- Work with minimum details necessary for specific task
- Obtain details dynamically as needed

Models

- Define roles in computation process
- Define roles in communication process
- Define rules of behavior for each role

Models

Typical roles in computation

- Application program
- Calling function / called function
- OS service
- Class or object

Typical roles in communication

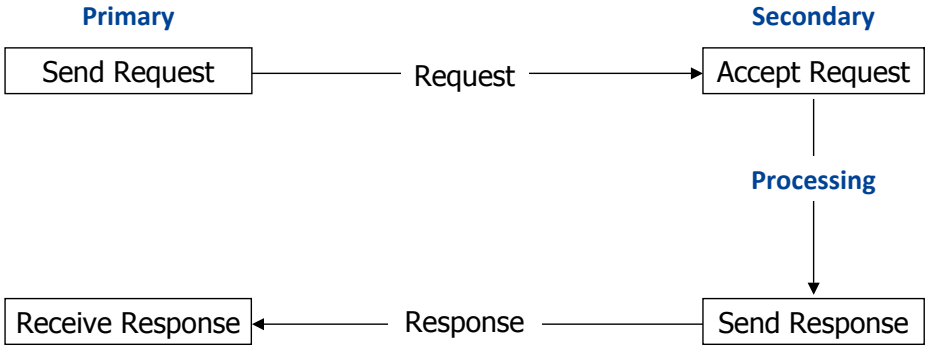
Primary	initiates request and accepts response
Secondary	responds to request
Symmetric	swap roles Primary ⟷ Secondary
Balanced	both roles Primary and Secondary

Example — client/server model

Client and Server	Concurrent application programs / threads
Client	Initiates request to server (Primary)
Server	Responds to client request (Secondary)

Transaction Model

Transaction → request + response



General model with many cases

Familiar examples

Client / Server transaction	Browser requests page from website
Procedural transaction	**`main()`** calls **`function(x)`**

Layered Systems

System divided into logical layers

Within layer

Subsystems interact tightly

Example

```
// subsystems: i, a[i],  b[i], c[i]
for ( i = 0 ; i < 1024 ; i++){
       a[i] = b[i] + c[i] ;
}
```

Between layers

Subsystems interact through programming interface

Example

`main()`

Calling function, Primary

`f(x)`

Called function, Secondary

```
// subsystems: main(), f(x)
main(){
       y = f(x) ;
}
f(x){
       return y;
}
```


Standard Agent Relationships

Agent

Software or hardware entity

Peer relationship

Two+ independent agents at same layer in layered model

Examples

- Independent user application layer programs
  - Microsoft Word + PowerPoint
  - Web Client (browser) + Web Server (website)
- Independent OS layer programs
  - USB driver
  - WiFi driver

Service relationship

`main()` calls **`function(x)`**

Microsoft Word calls printer driver

Application program opens socket (OS call)

Peer-to-Peer Transaction

Peer-to-Peer (P2P)

Transaction between agents of equal level or status

Usually **CLIENT / SERVER** model (not necessarily)

Example

Web service

Browser and web server — application programs (equal status)

Request

Browser (web client) sends page request to web server

Response

Web server sends page content to browser



Primary — web client **Secondary — web server**

Protocol Examples

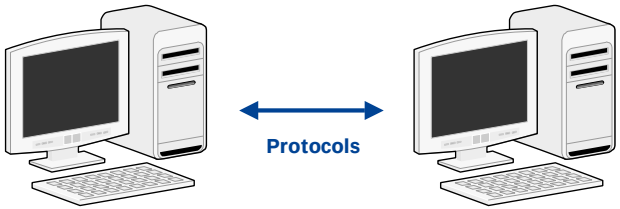
Transaction protocols

Hypertext Transfer Protocol (**HTTP**)

- Browser requests web page from web server
- Web server provides page as response

Post Office Protocol version 3 (**POP3**)

- Client system requests email messages from email server
- Email server provides messages as a response



Service Transaction

Service

Transaction between agents of unequal level or status

Example

User program makes OS call to open file

- User program is application running above OS
- OS performs performs low-level services for applications

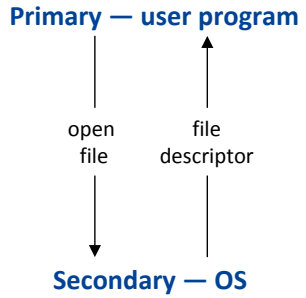
Request

- Application program issues OS call

Response

- OS opens file and returns file descriptor





Service Transaction Example

Calling function → Called function

Request

- Caller invokes called function with parameter

Response

- Called function returns with result

```
user(){
        local work
        response = provider(parameters)
        local work
}
provider(parameters){
        local work
        return response
}
```

Service transaction
Service request
+
Service response

General Layered Service Model

Task divided into layers

Layer n

Provider to layer n + 1

User to layer n – 1

Interface

Boundary between layers

Simple example

Two service transactions

- Layer 3 calls layer 2
- Layer 2 calls layer 1

Layer 2

- Provider to layer 3
- User to layer 1

```
layer_3(){
        local work
        response-2 = layer_2(p3-2)
        local work
}
layer_2(p3-2){
        local work
        response-1 = layer_1(p2-1)
        local work
        return response-2
}
layer_1(p2-1){
        local work
        return response-1
}
```

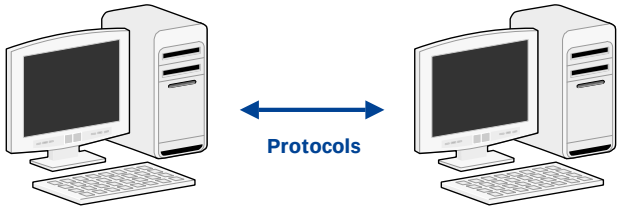
Protocol

Protocol

Rules for transaction between peers

Examples

- Syntax
- Semantics
- Synchronization
- Procedures
- Algorithms
- Naming



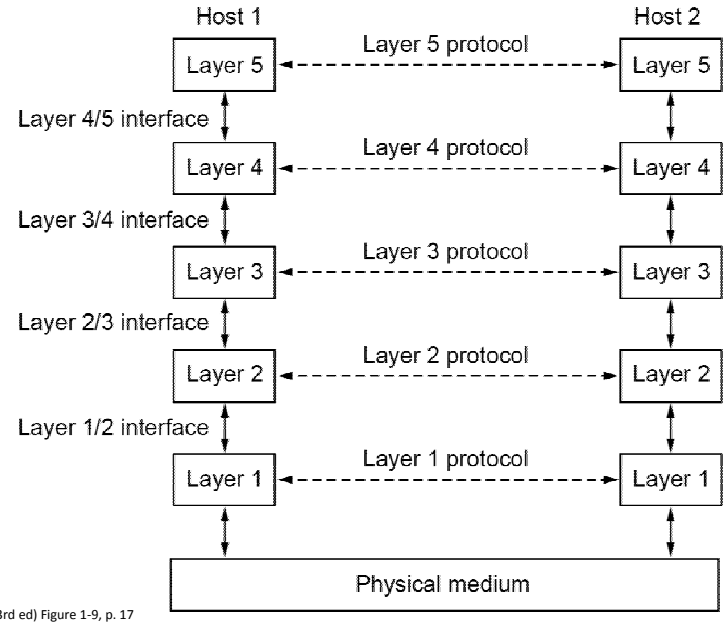
Layered communication

Communication task divided into layers

Protocol stack

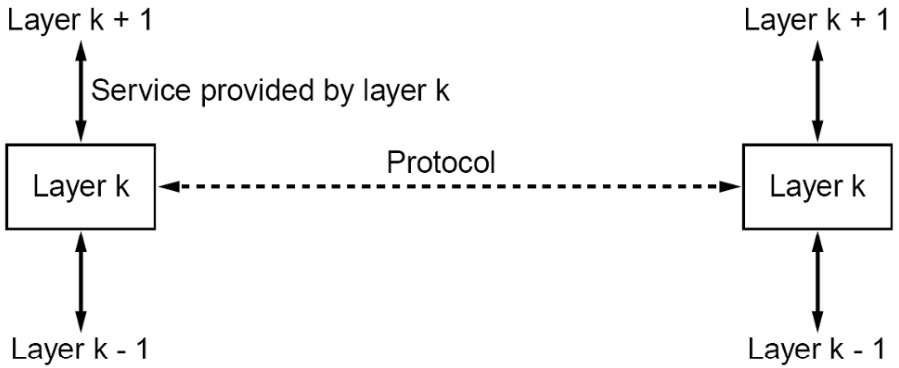
Specific peer-to-peer protocol defined at each layer

Protocol Stack



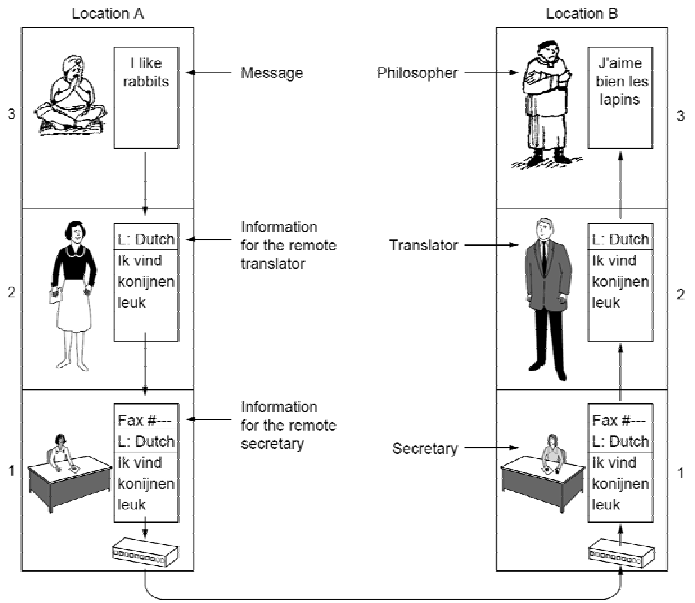
Tanenbaum (3rd ed) Figure 1-9, p. 17

Services and Protocols



The relationship between a service and a protocol

Protocol Stack Example



Tanenbaum (3rd ed) Figure 1-10, p. 19

The philosopher-translator-secretary architecture

Layered Protocol Model

Layer n protocol

Performs **VIRTUAL COMMUNICATION** between layer n peers

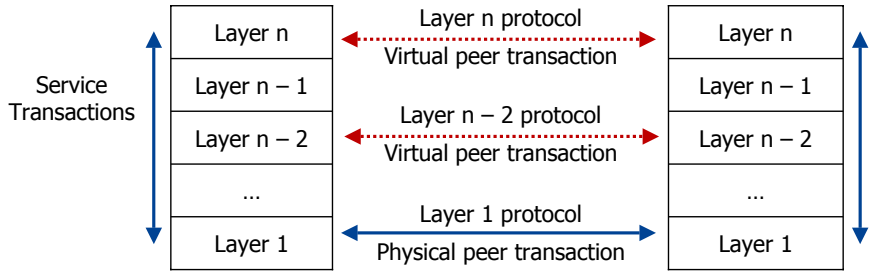
Exchanges layer n information with layer n peer

Layer n service

Receives request from layer n + 1

Passes request to layer n – 1 for communication service

Receives response from layer n – 1



Encapsulation — Protocol Headers

Layer n – 1 protocol

Receives service request from layer n

Request = message to layer n peer agent

Adds layer n – 1 **HEADER**

Header = message to layer n – 1 peer agent

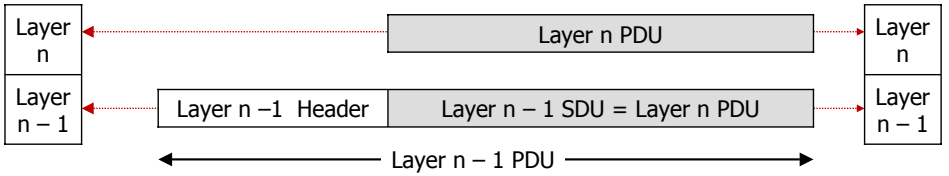
Protocol Data Unit (PDU) at layer n – 1

Message output from layer n – 1 protocol

Layer n PDU + layer n – 1 header

Service Data Unit (SDU) at layer n – 1

Layer n PDU = random data for layer n – 1



Functional Analysis of Communication

Open System Interconnection Model (OSI)

Layer	Function	Description
7	Application	Exchange of data between user applications
6	Presentation	Syntax and semantics of exchanged data
5	Session	Identification, separation, and continuity of multiple ongoing data transactions between software agents
4	Transport	Reliable end-to-end data exchange between host nodes Prevents data loss, errors, repetitions, ordering errors
3	Network	End-to-end data routing between host nodes via multiple hops
2	Data Link	Control of data transmission between neighboring hardware agents (one hop)
1	Physical	Data transmission between neighboring hardware agents on physical channels (electrical, optical, radio, …)

Example of OSI Functional Layers

Hypothetical OSI web browser

Layer	Example Functions
Application	Browser provides GUI — requests web pages by URL
Presentation	Encoding standard for Hebrew (Windows, UTF, ISO, …)
Session	Web page includes multiple graphic files Each file requested and received as separate conversation
Transport	Each request/response checked for errors and completeness Each requested file provided to session layer without errors
Network	Find route to web server by network address File requests/data exchanged with server by network address
Data Link	Data bytes exchanged between host computer and next-hop data communication hardware
Physical	Data bits exchanged with next-hop data communication hardware on physical channels

Internet Functional Model

OSI Layer	OSI Function	Internet Layer	Comment
7	Application	Application	Application provides presentation service and some session service (transactions)
6	Presentation		
5	Session	Transport	Internet session management can be: Reliable — with transport service Unreliable — without transport service
4	Transport		
3	Network	Network	End-to-end data routing as in OSI
2	Data Link	Infrastructure	Internet protocols do not discuss physical data transmission
1	Physical		

Ref: http://tools.ietf.org/html/rfc4949

Example of Internet Functional Layers

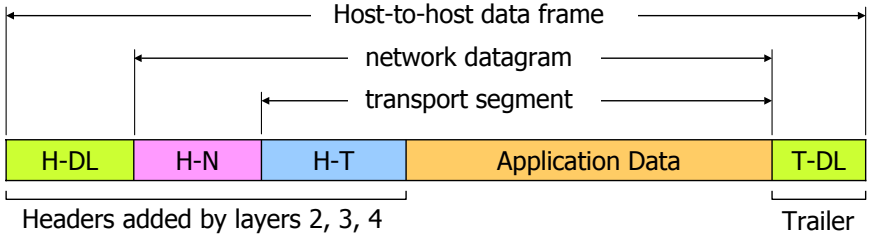
Typical web browser

Layer	Example Functions
Application	Browser provides GUI — requests web pages by URL Translate (DNS) URL into network address (IP) for web server Encoding standard for Hebrew (Windows, UTF, ISO, …) Web page includes graphic files Each file requested/received as separate conversation (HTTP)
Transport	Each file request conversation identified for error control (TCP) Each requested file provided to session layer without errors
Network	File requests/data exchanged with server by network routing (RIP, OSPF, IGRP, BGP) Transfer data across network by network address (IP)
Infrastructure	Network layer messages sent to Internet data communication equipment

Internet PDUs

Protocol Data Unit (PDU)

Layer	Message	PDU
Application	Data	Message
Transport	Header	Segment
Network	Header	Datagram
Data Link	Header + Trailer	Frame
Physical	Bits	Signal



Internet Endpoints

Network Endpoint

Address of **SOFTWARE AGENT** running in **HARDWARE AGENT**

Network Address + Port

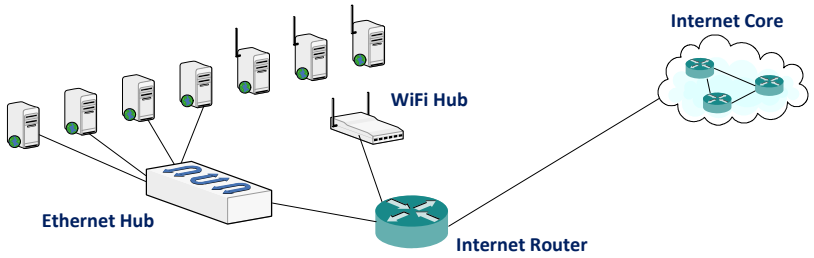
System Level	Layer	Communication ID	
User	Application	Socket	Associates file descriptor with network endpoint
Operating System	Transport	Port	Software address identifies program exchanging data
	Network	Network (IP) Address	Identifies computing node in global network
Hardware	Data Link	Hardware Address	Identifies hardware device (node) in local network
	Physical	Attachment	Physical connection

Well-known ports

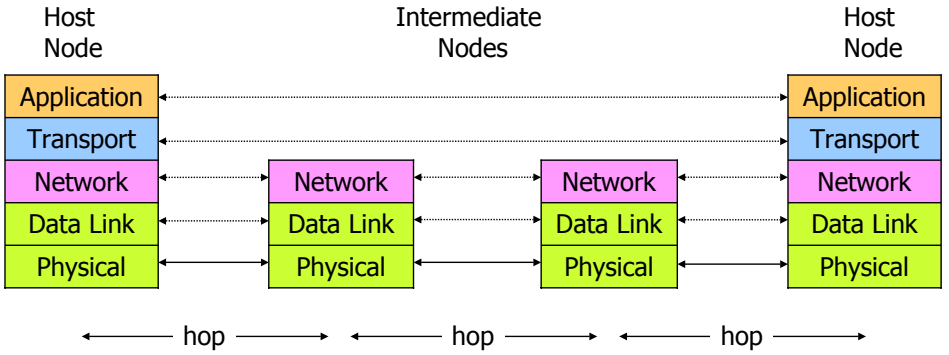
Standard services defined on ports 0 – 1023

Data Communication Equipment (DCE)

Layer	DCE	Function
Network	Router	Receives Network Datagrams in Data Link Frames Sends Datagrams in Data Link Frames to next hop on path to destination
Data Link	Switch (Hub)	Manages physical transmission layer Exchanges Frames among neighboring hardware agents
Physical	Network Interface Card	Modulator/demodulator (modem) Transmits and receives digital bits over physical medium



Internet Hops



Host nodes

- Application data (message) sent to Transport for reliable exchange
- Transport segment sent to Network for addressing and routing

Intermediate nodes

- Examine Network datagrams for addressing and routing
- Treat Transport segment as meaningless data

Network Zoo

Many network types with specific protocol stacks

Wide Area Networks (WAN)

- Public Switched Telephone Network (PSTN)
  - Local loop, backbone, PDH/SDH, ESS, ISDN
  - Public Switched Data Network (PSDN) — X.25
- Broadband Integrated Network
  - ATM, B-ISDN, Frame Relay
- Cellular
  - 2.5G (GPRS/EDGE), 3G (UMTS, CDMA2000), 4G (WCDMA)

Local Area Networks (LAN < 2 km)

Ethernet, WiFi, VLAN, token ring, token bus, FDDI, …

Personal Area Network (PAN < 20 m)

Bluetooth, ZigBee, IrDA, …

Commercial network protocol stacks

SNA, DECnet, Windows Networking, AppleNet, Netware, …

So, what is 'The Internet'?

Internet = Inter-Networking

Protocols for connecting heterogeneous networks

Autonomous System (AS)

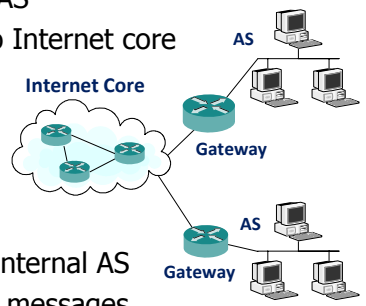
Any network running its own protocol stack

Internet Gateway

- Runs network-specific protocol stack on AS
- Runs Internet protocols on connection to Internet core

Internet core

- Backbone network of Internet routers
- Connected by dedicated links



Typical implementation

- Hosts run network-specific protocols on internal AS
- Hosts use Internet protocols for external messages
- No difference at infrastructure level

Intranet?

Intranet

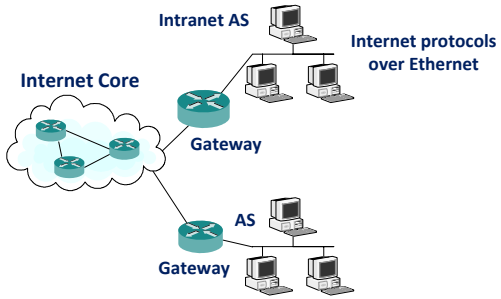
- Using internet protocols in AS

Pure intranet

- Internet protocols above Ethernet/WiFi LAN

Windows network

- Uses Internet protocols for transport and addressing
- Uses Microsoft protocols for message syntax, node location, …

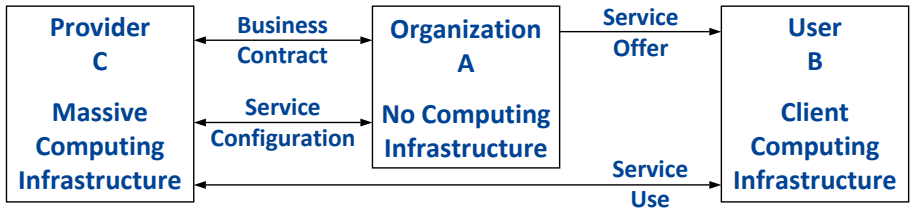


Hey! Hey! You! You! Get Off of My Cloud

words and music: Mick Jagger and Keith Richards

Cloud ≠ Internet ≠ Network

- Network
  - Collection of agents with single defined protocol stack
- Internet
  - Collection of agents using inter-networking protocols at layers 3 & 4
- Cloud
  - Business model
  - Organization A rents computing service from provider C
  - Organization A offers service to user B via provider C network



Why Cloud Computing?

Outsourcing service model

- User gets computing services from service provider
  - Service Level Agreement (SLA) guarantees customer service
  - Provider handles operations+administration+maintenance (OAM)
- Business advantages to organization
  - Economies of scale — large provider can do it cheaper
  - Cuts labor/capital costs from balance sheet → happy investors

Based on standard technologies

- Cloud service organized from conventional resources
  - Hardware + software + network
  - Provider offers menu of services
- Not a fundamentally different computing technology

Unique technological issues

- Service reliability — provider committed to SLA
- Optimization of provider-side resource configuration
- Optimization of user-side resource configuration

Service Configuration in Cloud Computing

Infrastructure as a service (IaaS)

- Organization sees virtual hardware environment
  - Real hardware or hypervisor / system virtual machine
- Organization installs OS → installs software → user runs jobs

Platform as a service (PaaS)

- Organization sees virtual OS environment
  - OS on single hardware platform or virtual OS
- Organization installs software → user runs jobs

Software as a service (SaaS)

- Organization sees virtual application software environment
  - Applications running on private OS or "sandboxed" on shared OS
  - Sandbox — private execution environment per application instance
- User runs jobs

Storage as a service (STaaS)

- User sees virtual mounted storage device

Centralize → Decentralize → Centralize → ?

1950s — 60s

- Centralized mainframe computer + multiple OS instances over hypervisor
- Timesharing OS serves multiple users
- User sees OS environment via dumb terminal (thin client)

1970s

- User applications offloaded to minicomputers + timesharing services
- User sees timeshared OS environment via dumb terminal

1980s

- User applications offloaded to personal workstations (PC)
- User sees single-user OS environment running locally

1990s

- Network single user workstations
- User sees single-user OS environment running locally

2000s

- Centralized control of local OS environment by IT departments

2010s

- Cloud + netbook / tablet / smart phone — dumb terminal with high-res GUI

Issues in Cloud Computing

Cost

- Provider issues
  - Economies of scale ⇒ lower cost per compute job
- Organization issues
  - Capital + OAM costs → operating costs
  - Lower start-up costs ⇒ operating debt

Reliability

- Provider issues
  - Redundant infrastructure → continuity + disaster recovery
  - Centralized management of OAM, security, performance
  - Virtualization → serve multiple users on physical server
  - Multitenancy → provide multiple sandboxed application instances on OS
- User sees guaranteed service

Agility

- Organization / provider reconfigure service as needed
- Growth, load balancing, time-zone serving

Cloud Ownership

Public cloud

- Service provider as public utility — sells / rents computing service
  - Initial providers leverage large existing infrastructure
  - Amazon, Microsoft, Google, IBM
- Menu of services at fixed prices

Private cloud

- Cloud infrastructure for private organization
- Managed internally or outsourced
- Isolates service developers from implementation issues
  - Standard development platform
- Requirements for economic justification
  - Large organization
  - Technology-based services
  - Frequent new service
  - Example — internet content provider

Programming in the Cloud

Depends on environment

- IaaS — Organization sees virtual hardware environment
- PaaS — Organization sees virtual OS environment
- SaaS — Organization sees virtual application software environment

IBM Bluemix

- SaaS from IBM
- Free accounts for students using **name@hadassah.ac.il** address
- Bluemix DevOps Services
  - Develop, track, plan, and deploy software on IBM cloud service
  - Collaboration tools — Git, Jazz SCM, GitHub
  - Build application → deploy to IBM cloud
  - Supports
    - Arduino, C, C#, C++, CSHTML, Embedded, JavaScript (ejs) Erlang, Go, HTML, abstraction markup language (Haml) Jade, Java, JSON, Lua Objective-C PHP, Python, Ruby, Swift, Virtual, Basic (vb) VMHTML, XHTML, XML, Xquery, yaml, Launch, file Dockerfile, gitignore, git config, cfignore
  - "You can go from source code to a running app in minutes."

Some Internet Protocols

Application layer transactions

Hypertext Transfer Protocol (HTTP)

Transport layer

Transport Control Protocol (TCP)

Reliable transport service

User Datagram Protocol (UDP)

Unreliable transport service

Network layer

Internet Protocol (IP)

Node addressing

Internet Control Message Protocol (ICMP)

Messages about messaging

Routing protocols (RIP, OSPF, IGRP, BGP)

Learn network topology for message forwarding

RFC — Internet standard

Protocol	RFC
HTTP	2616
TCP	793
UDP	768
IP	791
ICMP	792

What Internet Protocols Do
Some examples

Hypertext Transfer Protocol (HTTP)

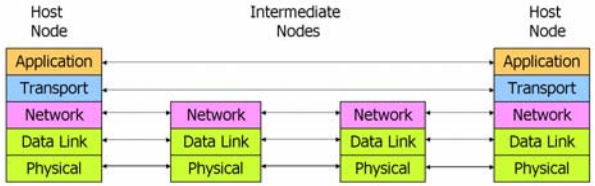
Application layer transactions

Requests

Get	Retrieve file by name
Post	Replace file by name
Delete	Delete file by name

Responses

Data	Contents of requested file
Status	Status of transaction



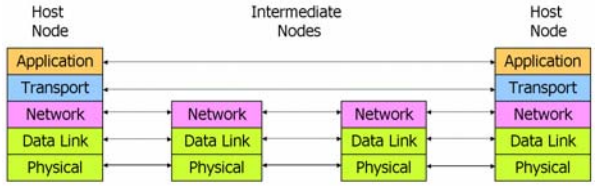
What Internet Protocols Do
Some examples

Domain Name Service (DNS)

Translates node name to Internet address (and vice versa)

Example

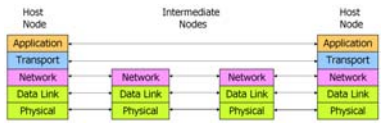
```
c:\> nslookup www.hadassah.ac.il
Name:    www.hadassah.ac.il
Address: 212.179.79.228
```



What Internet Protocols Do
Some examples

Transport Control Protocol (TCP)

Reliable transport service



Sender

Label source and destination software by port number

Number outgoing segments

Wait for **ACK** (acknowledgment) for outgoing segments

Retransmit segments if no **ACK** before timeout

Negotiate segment size (for error and congestion control)

Receiver

Check completeness and order of incoming segments

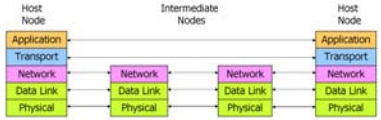
Check incoming segments for errors

Send **ACK** for good segments

Provide good incoming segment to destination software

What Internet Protocols Do
Some examples

Internet Protocol (IP)



- Best effort network service
- No guarantee of delivery

IP version 4 address

- Four octets 0.0.0.0 to 255.255.255.255 (many reserved addresses)

Sender

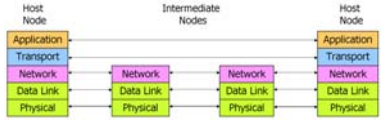
- Attach source and destination network addresses to segment
- Route IP datagram to next hop along route

Receiver

- Intermediate node — route IP datagram to next hop along route
- Host node — provide segment to transport layer

Network Infrastructure
Layers 1 + 2 — bits, bytes, signals, cables, electronics

Scale



- Wide Area Network (WAN < earth)
- Local Area Network (LAN < 2 km)
- Personal Area Network (PAN < 30 m)

Medium

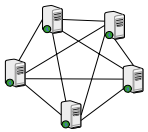
Copper wire and cable Electrical signals	Requires legal right to install cables
Optical fiber Light wave signals	
Open space Radio wave signals	Requires legal right to transmit radio

Traffic statistics

- Constant Bit Rate (CBR) — peak data rate = average data rate
- Variable Bit Rate (VBR) — peak data rate > average data rate

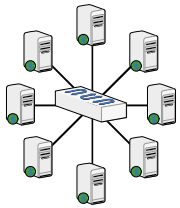
Connectivity = Medium + Topology

Point-to-point



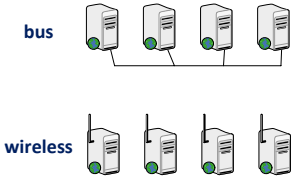
- Dedicated link from node to node
- Fastest and most complex

Switch



- Dedicated link from node to switch
- Switch connects nodes on request
  - Non-blocking provides $n \times (n - 1)$ connectivity
  - Blocking provides $n \times m$ connectivity ($m < n - 1$)

Shared medium



- Nodes share medium access
- Contention
  - Nodes compete for access
- Polling
  - Central controller polls nodes

Physical Transmission

Serial data rate at physical layer

- Bits per second = bps = b/s
- Bytes per second = B/s
- 1 B/s = 8 b/s

Capacity (bandwidth)

- Maximum data rate on medium
- Fixed by transmitter / medium / receiver
- Limits
  - Speed of circuits
  - Signal to noise ratio (SNR)



Physical Transmission

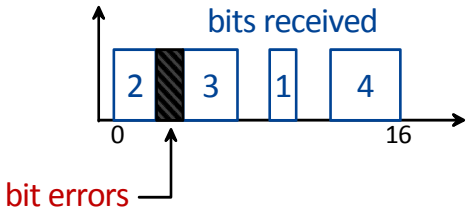
Throughput

$$\text{throughput} = \frac{\text{error-free data received per second}}{\text{capacity}}$$

Takes account of

- Utilization = % time transmitter sending
- Errors ⇒ re-transmission ⇒ more data on same capacity
- Delays ⇒ less data received on same capacity

utilization = 10 / 16 = 62.5%



Baud Rate
Symbols per second

Symbol

Physical signal that encodes bits

Symbol rate (Baud rate)

Symbols transmitted per second

Bit transmission rate

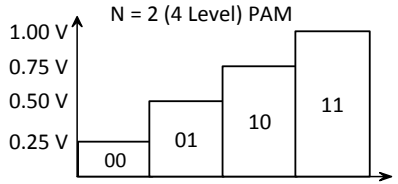
Bits transmitted per second = (symbols / second) × (bits / symbol)

Example

Pulse amplitude modulation (PAM)

Define 2^N electrical levels from 0 to 11…1

Each symbol (level) transmits N data bits



Baud Rate
Symbols per second

33 kbps dial-up modem

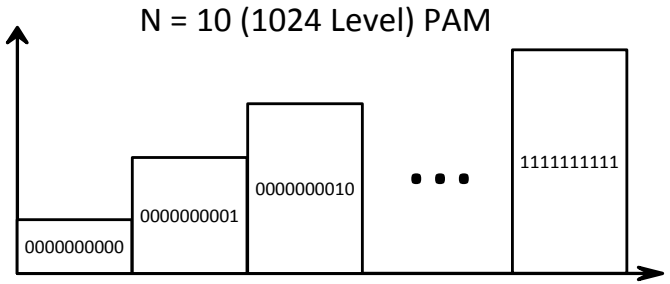
Define 2^{10} = 1024 electrical symbols (max for SNR on phone line)

Baud rate = 3300 symbols / second

Bits transmitted per second

Data rate = (3300 symbols / second) × (10 bits / symbol)

= 33,000 bps



Data Concentration

High capacity link

No single node can utilize link capacity

Example

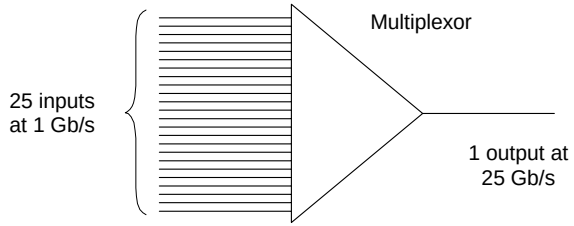
- Optical fiber cable with 4 fibers at 25 Gbps = 100 Gbps

Multiplexing

Combine multiple nodes onto one link

Example

- Optical fiber with 25 Gbps data rate
- Combine 25 nodes transmitting at 1 Gbps



Multiplexing Methods

Frequency Division Multiplexing (FDM)

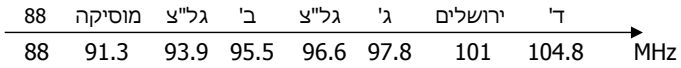
Divide available frequencies (bandwidth) among nodes

Nodes transmit simultaneously on different frequencies

Example

FM radio uses 88 MHz to 108 MHz = 20 MHz bandwidth

Divide 20 MHz into 100 channels = 200 kHz per FM channel



Multiplexing Methods

Time Division Multiplexing (TDM)

Divide capacity into time slots

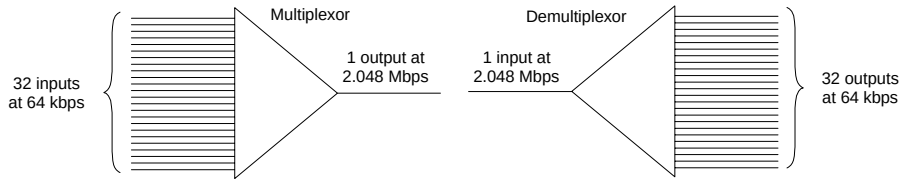
Node transmits in assigned time slot

Example

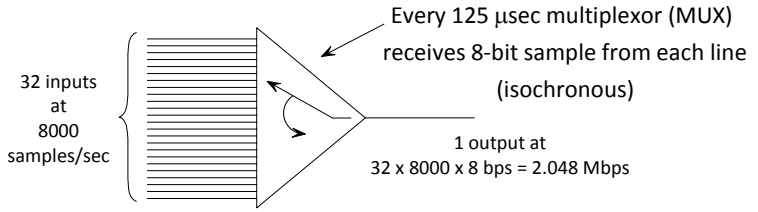
E1 digital line transmits at 2048 kbps

Divide 2048 kbps line into 32 time slots = 64 kbps per node

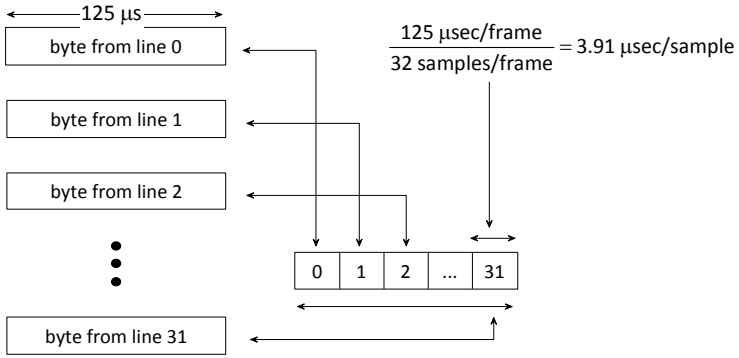
32 x 64 kbps = 2048 kbps = 2.048 Mbps



E1 Multiplex



$$\frac{1}{8000 \text{ samples/second}} = 125 \text{ µs/sample}$$



$$\frac{125 \text{ µsec/frame}}{32 \text{ samples/frame}} = 3.91 \text{ µsec/sample}$$

Mixed Multiplexing
GSM Cellular

Time Division Multiple Access (TDMA)

Used on GSM / UMTS phones — 2G and 3G

Combines FDM and TDM

Frequency Division Multiplexing (FDM)

GSM bands = 25 MHz

Divide 25 MHz into 125 channels = 200 kHz per channel

Transmit 270 kbps over 200 kHz channel

Time Division Multiplexing (TDM)

Divide 270 kbps into 8 times slots = 33 kbps per user

33 kbps = 23 kbps for voice + 10 kbps control

Data Statistics — CBR

Constant Bit Rate (CBR)



- Isochronous data
  - Equal time interval between bits
  - Bits per second = constant

Average data rate

- Average data rate = peak data rate = minimum data rate

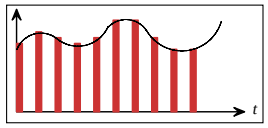
Example

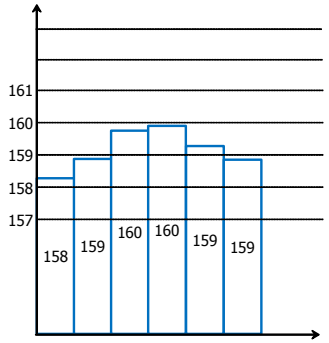
- Uncompressed digital audio
- Sample analog signal every T seconds
- Round-off sample to N-bit number from 0 to $2^N - 1$
- Digital audio stream at N / T bps

Digital Voice on Telco Telephone

Sample analog voice signal every 0.125 ms

- 0.125 ms per voice sample ⇒ 8000 voice samples / second





Round-off sample to 8-bit data

- Data ∈ {0, 1, 2, ... , 255}
- Sample = {158.276, 158.879, 159.724, 159.821, 159.312, 158.791}
- Data = {158, 159, 160, 160, 159, 159}

DS-0 stream

- (8000 samples / second) × (8 bits / sample) = 64 kbps
- 64 kbps digitized voice (no compression)

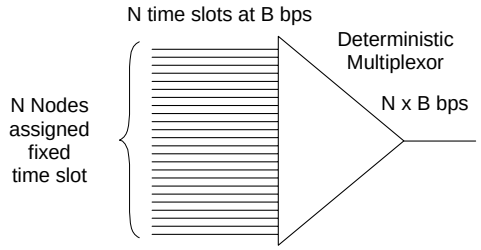
Multiplexing Statistics

Deterministic multiplexing (CBR)

- N Nodes = N time slots
- Node reserves fixed time slot
  - Guaranteed transmission capacity
  - Node transmits in assigned time slot

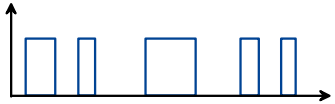
Example

- E1 multiplex for wired telephone — 32 x 64 kbps = 2048 kbps
- E2 multiplex — 4 x 2048 kbps = 8192 kbps



Data Statistics — VBR

Variable Bit Rate (VBR)



- Bursty data
  - Peak data rate B > average data rate λ
  - Assume packets are independent (Poisson statistics)

$P(k,T,\lambda)$ = probability of k bits arriving in T seconds when average rate = λ

$$P(k,T,\lambda) = \frac{(\lambda T)^k}{k!} e^{-\lambda T}$$

Example

- Data sent by time-of-day client
  - Request time-of-day (1000 bits) once every hour (3600 seconds)
  - Average data rate = 1000 bits / 3600 seconds = 0.28 bps
- Peak data rate = 55 Mbps on 802.11g WiFi
  - Peak data rate 55 Mbps > average data rate = 0.28 bps

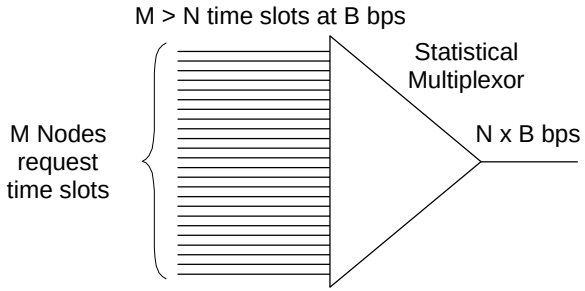
Multiplexing Statistics

Statistical multiplexing (VBR)

- M nodes > N time slots
- Bursty data
  - Average data rate λ < peak data rate B
- Average traffic rate = M x λ < capacity rate = N x B
  - Actual traffic < capacity ⇒ OK
  - Actual traffic > capacity ⇒ data delayed or lost

Example

Internet routers



Overflow in VBR

Overflow

- Actual traffic > capacity
  - Short time (a few time slots) ⇒ data delayed
  - Long time (many time slots) ⇒ buffer overflow ⇒ data lost

Overflow probability

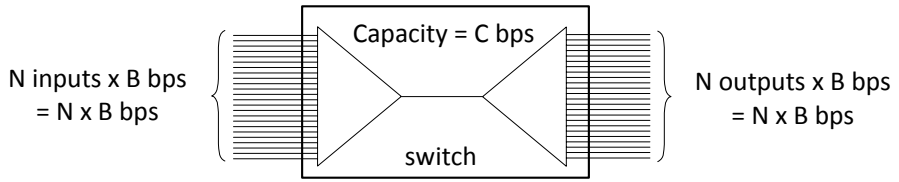
- Average traffic rate = M x λ
  - Average data arriving in time T = M x λ x T
- Capacity rate = N x B
  - Data capacity in time T = N x λ x T
- Overflow in time T
  - Actual data arriving in time T > N x λ x T
    - N x λ x T + 1 or N x λ x T +2 or N x λ x T +3 or ...
    - Independent outcomes

$$P(\text{overflow}) = P(N\lambda T+1 \; or \; N\lambda T+2 \; or \; ...) = \sum_{k=N\lambda T+1}^{\infty} \frac{(M\lambda T)^k}{k!} e^{-(M\lambda T)}$$

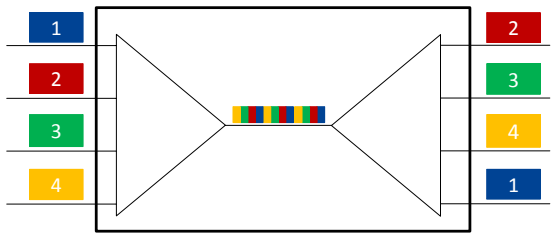
Switching

Switch

- Multiplexor + Demultiplexor
- Data at **`input_port`$_i$ → `output port`$_j$**
  `i,j = 0, 1, 2, ... , N - 1`



Example



Circuit Switching

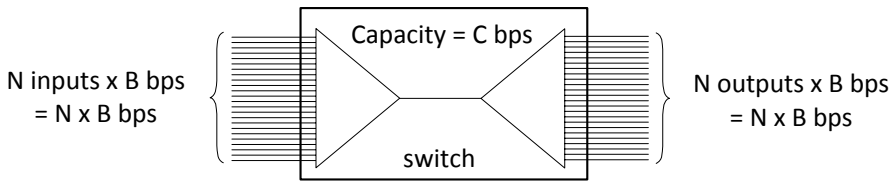
Deterministic multiplexing

- Capacity C = N × B
- Dedicated (reserved) link
  `input_port`$_i$ → `output port`$_j$
  No competition
  Guaranteed capacity B — if used or not

Example

Bezeq phone call

64 kbps from telephone to telephone (even if no one speaks)



Packet Switching

Statistical multiplexing

Capacity C = M × B < N × B

Dynamical time slot assignment (on request)

input_port$_i$ → output port$_j$

Competition

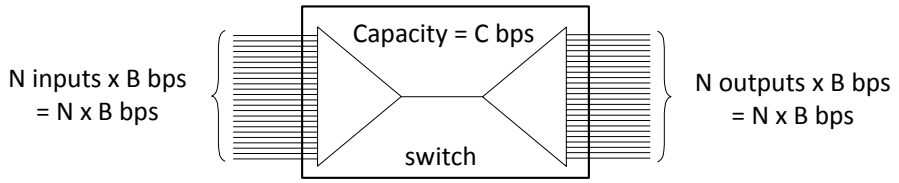
More ports than capacity

Demand > capacity ⇒ delay

Example

Internet router

Packet queue — first come first served



Connection Types

Connection

State machine associated with data exchange

Connection-oriented

First set-up data channel

Multiple data transactions associated with connection state

Monitor channel state during data exchange

Close channel after data exchange

Example — **phone call**

Enter number → answer call → extended conversation → disconnect

Connectionless

Transmit data with no prior channel set-up

No channel state defined by nodes

Each message independent

Example — **email message**

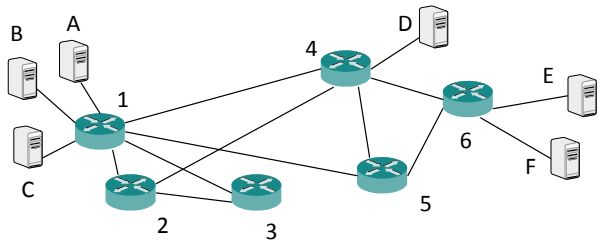
Send email → hope message arrives → hope message is found / read

Datagram Service

Network of routers and links

Packet switching

Connectionless



Each datagram

Has source and destination address in header

Data Link header or Network header

Routed individually through network

Datagrams may follow separate routes

Example

src = B dest = F	data

B → 1 → 4 → 6 → F

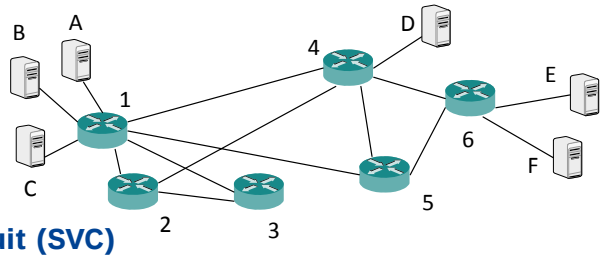
B → 1 → 5 → 6 → F

Switched Virtual Circuit (SVC)

Network of switches and links

Circuit switching or packet switching

Connection-oriented



Switched Virtual Circuit (SVC)

Set-up / close messages carry source and destination addresses

Example

Set-up VC – 1: B → 1 → 4 → 6 → F

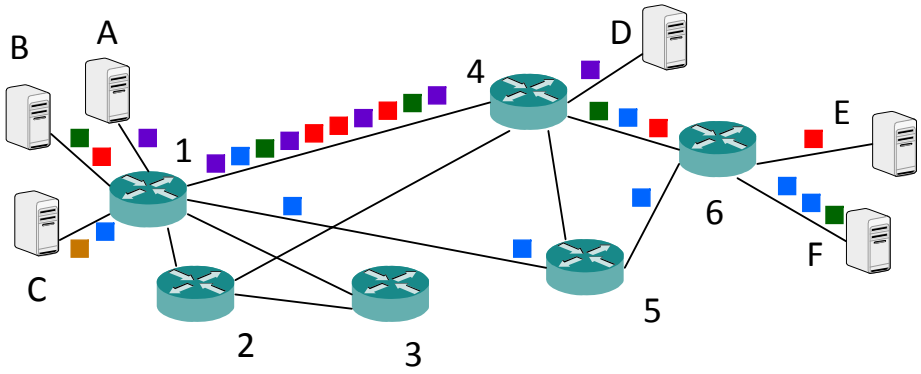
Packet routing by VC ID in header (layer 2 or layer 3)

Every packet follows same VC route

Example

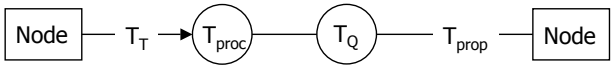
VC – 1	data

Switching Example



A to D — circuit mode (deterministic SVC)
B to E — packet mode (statistical SVC)
B to F — packet mode (statistical SVC)
C to F — packet mode (datagram service)

Transmission Delay



Transmission delay T_T

T_T = Time to inject bits into line = (bits in packet) / (bits per second)

Example: 1000 Mb / 100 Mbps = 10 sec

Processing delay T_{proc}

Packet process time in intermediate node

SVC with fixed route ⇒ shorter delay than datagram routing

Propagation delay T_{prop}

T_{prop} = (length of cable) / (signal speed)

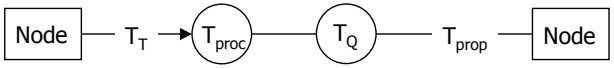
Example: 4 km / (2×10^8 km/s) = 2×10^{-8} sec << 10 sec

Queuing delay T_Q

Time packet waits in buffer for previous packets (congestion)

T_Q = (service time per packet) × (packets waiting in buffer)

Example of Queuing Delay



Queuing delay T_Q

T_Q = (service time per packet) × (packets waiting in buffer)

Packets waiting in buffer = 1 / (1 – utilization)

Queuing delay example

Service time per packet = 10 ms / packet

Service rate = 1 / (10 ms / packet) = 100 packets / second

Average traffic = S = 85 packets / second

Utilization = (85 packets / second) / (100 packets / second) = 0.85

Buffer level = 1 / (1 – 0.85) = 6.67

T_Q = (10 ms / packet) × 6.67 packets = 67 ms

C = switch capacity = service rate = 100 packets / second

Demand > 100 buffer ⇒ overflow ⇒ excess delay

$$P(\text{demand} > C) = \sum_{k=C+1}^{\infty} P(\text{demand} = k) = \sum_{k=C+1}^{\infty} \frac{S^k}{k!} e^{-S} = \sum_{k=101}^{\infty} \frac{85^k}{k!} e^{-85} = 0.05$$

Error Control

Bit error

Data 1 received as 0 or data 0 received as 1

$$\text{Bit Error Rate (BER)} = \frac{\text{bit errors in received data}}{\text{bits in received data}}$$

Packet Loss

Congestion or buffer overflow → packet discarded

$$\text{Packet loss rate} = \frac{\text{packets lost}}{\text{packets transmitted}}$$

Error detection

Error correction code / redundancy code / checksum

Checksum transmitted with data in header / trailer

Receiver compares independent hash with transmitted code

Error control

Required

Discard corrupt packet

Optional

Retransmit discarded / missing packets

Network Scale

Private network

- Small Office / Home Office (SOHO)
  - Small number of computers in a few rooms
  - Simple Ethernet / WiFi LAN
- Enterprise
  - Many nodes in large building / campus
  - Complex Intranet

Access network

- Provide user connection to Internet core
- Infrastructure provider manages layers 1 and 2
- Internet Service Provider (ISP) manages layers 3 and 4

Internet core

- Network of routers and links at layer 3
- Infrastructure provider manages links at layers 1 and 2
- Links are typically built over complex network systems

Private Networks

Simple Ethernet / WiFi LAN

- Ethernet switching hub
  - 4 to 16 nodes
  - Full connectivity (non-blocking)
  - 10 / 100/ 1000 Mbps
- WiFi hub
  - More nodes lowers performance
  - Nodes compete to transmit to hub
  - 11 / 54 / 100+ Mbps

Complex Intranet

- Multiple LAN hubs
- Hubs connected
  - Directly (bridging)
  - Indirectly (routing)

Non-Private Networks
Access + core

Service infrastructure

- Routing + accounting nodes in office buildings

Link infrastructure

- Cables + radio channels on public / private property
  - Legal and licensing issues
- Controlled by companies in cable businesses
  - Telephone companies (Telco)
  - Cable TV companies
  - Electric companies
  - Railroads companies

Choices for small business Intranet at 3 locations

- Pay service provider monthly
- Or
  - Purchase LAN hubs and routers
  - Lease cables from Telco

Telephone Network
It's everywhere

Local loop

- Wired connection to most buildings
- Can carry 1 Mbps (up to 4 km) to 25 Mbps (up to 300 m)
- Voice network
  - Analog voice channel from 300 to 3300 Hz
  - Digitized voice at 64 kbps

Local presence (central office) in every neighborhood

- Local loop attached to non-blocking switches

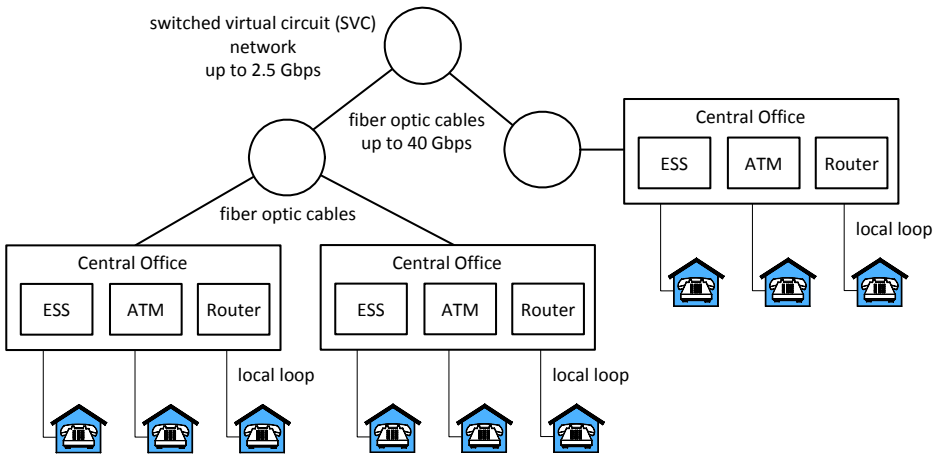
Tree network of switches

- Central offices connect to regional offices on fiber optic backbone

Global broadband switched virtual circuit (SVC) network

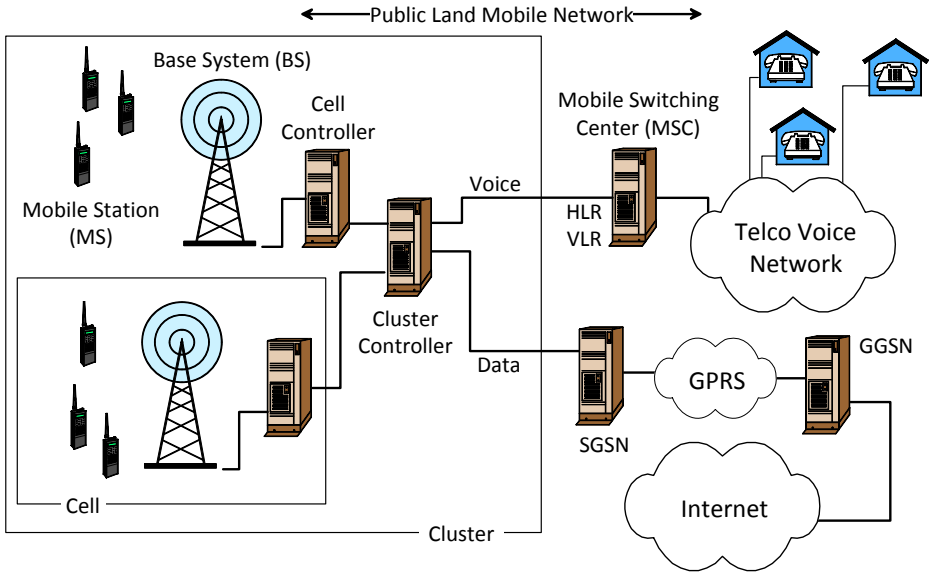
- Circuit mode switches (ESS7) for 64 kbps voice
- Circuit / Packet mode layer 2 switches (ATM) up to 2.5 Gbps
- Private routers throughout network for Internet traffic

Telephone Network



Cellular Network

Wireless to base station — uses Telco network for WAN service



SOHO Access Networks

Dial-up modem (modulator / demodulator)

- Converts digital bits from computer to analog signals for phone line
- User modem connects to ISP modem by phone call
- 56 kbps downstream / 33 kbps upstream

Digital Subscriber Line (DSL)

- FDM on local loop
- Voice channel connected to telephone voice network
- Data channel — 15 Mbps downstream / 750 kbps upstream
  - ATM link between DSL modem and Telco central office
  - Datagrams routed to ISP on Telco router network

Cable modem

- FDM on TV cable
- TV channels connected to TV
- Data channel — 30 Mbps downstream / 2 Mbps upstream (shared)
  - Ethernet link between cable modem and cable head office
  - Datagrams routed to ISP on Telco router network

Enterprise Access Networks

Leased line

- Telco line to DCE on customer premises
- 2.048 Mbps to 40 Gbps
- Carrier Ethernet — Ethernet extensions for metropolitan networks

Asynchronous Transfer Mode (ATM)

- Telco system for broadband switched virtual circuits (SVC)
- Optimized for multimedia transmission
- Layer 2 ATM switch on customer premises
- Telco line up to 2.5 Gbps

Frame Relay (FR)

- Telco system for broadband permanent virtual circuits (PVC)
- Layer 2 FR switch on customer premises
- Telco line up to 45 Mbps

WiMax

- Wireless metropolitan network
- Applies cellular technology for 40 Mbps data

Internet Core

Internet backbone

Collection of core routers and fast links

Core router

Fast router with very high I/O capacity

Up-to-date routing protocols

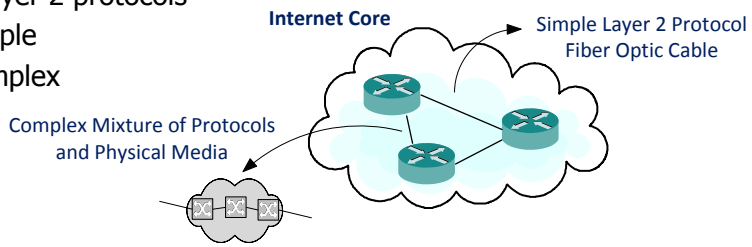
Handle multiple layer 1 and layer 2 protocols

Fast links

Various layer 2 protocols

Some simple

Some complex



Documentation

Standards

Formal documentation of systems, algorithms, protocols

Adopted by international committees

Record technical background and implementation requirements

Standards organizations

ISO	International Standards Organization Organization of governmental standards organizations
ITU-T	International Telecommunications Union - Telecommunications Sector United Nations standards organization (formerly CCITT)
ANSI	American National Standards Institute US government standards organization
IEEE	Institute of Electrical and Electronics Engineers
ACM	Association of Computing Machinery
IETF	Internet Engineering Task Force The Internet Society inherited Internet from US government in 1989 Internet standards called RFC (request for comment) Available at http://www.ietf.org/rfc.html

Application Layer

Application Layer

Application layer

"The user runs an application program. The program selects the data transport service it needs — either a sequence of data messages or a continuous stream of data — and hands application data to the Transport Layer for delivery."

RFC 4949 — Internet Security Glossary

Application layer issues

What data transport service does application need?

Quality of service (QoS)

- Measurable network parameters
- QoS requirements determine transport requirements

How does application hand data to transport layer for delivery?

Socket interface

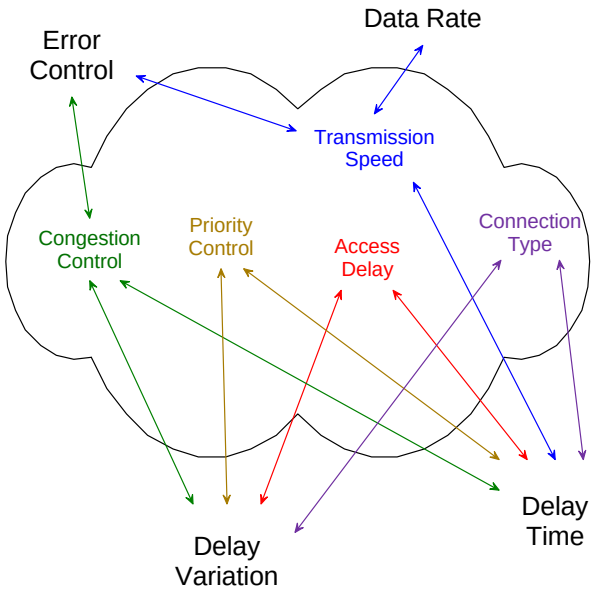
- OS call opens Internet connection to program on remote node
- Socket API — application sees remote program as open file

Quality of Service (QoS)

Network parameters

Data rate	Physical transmission speed in bps
Error control	Bit error rate (BER) $< 10^{-9}$ on fiber optic cable $< 10^{-3}$ on wireless Packet loss rate Depends on congestion control policy
Delay	End-to-end transmission time for one bit High data rate ⇒ lower transmission delay T_{trans} Congestion / priority ⇒ longer queuing delay T_Q Datagram service ⇒ longer processing delay T_{proc}
Delay variation (jitter)	Dynamic variations in Data rate / propagation delay ΔT_{trans} Queuing delay ΔT_Q Processing delay ΔT_{proc}

QoS and Network Policies



QoS Requirements for Various Services

Service	Speed	Error Control	Delay	Delay Variation
e-mail	—	good	—	—
file transfer	reasonable	maximum	reasonable	reasonable
NFS				
database access				
voice	fast	good	very small	very small
video	very fast	good	very small	very small
real time control				

QoS and Transport Trade-Off

	Reliable Transport	Unreliable Transport
Protocol	TCP	UDP
Error control	Error-free	No error correction
Connection	Connection-oriented	Connectionless
Socket type	Stream	Datagram
Processing overhead	Error correction, packet ordering, congestion control, session state management	None

Reliable Transport

Generally preferred when possible

Unreliable Transport

Used when

- Some data loss tolerable
- Delay or jitter intolerable
- Example — video delivery

Application / Transport Interface

Application requests service from transport layer

Application Layer	Prepare Transport service requirements Data for transport Local endpoint — node address + service port Remote endpoint — node address + service port
Transport Layer	Receives Application layer PDU — data for transport Interface Control Information (ICI) Service requirements Local + remote endpoints

Berkeley Sockets

Networking API

De facto standard on Unix / Linux / Apple systems

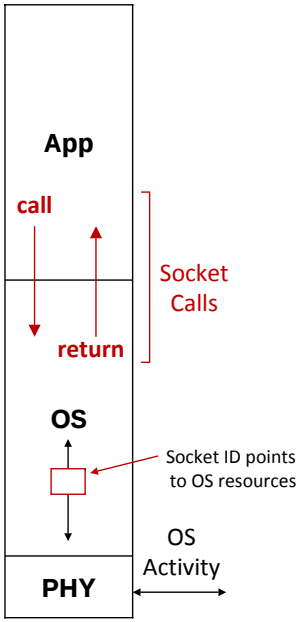
Application performs communication like file I/O

Address	Internet address of local / remote node
Port	ID number of client / server software
Source	Local node
Destination	Remote (foreign) node
Endpoint	Address + port of software process (local / remote)
Association	Description of connection between client / server processes Protocol + source endpoint + destination endpoint
Socket	OS resources associated with protocol + endpoint Server side — socket ID points to service offered to remote nodes Client side — socket ID points to connection to remote service

Socket System Calls

Client and Server	
Socket	Create new socket associated with protocol at local address
Bind	Associate port with socket
Send	Local process sends data to socket (OS forwards to remote node)
Receive	Local process reads data from socket (OS buffer from remote node)
Close	Terminate socket
Client Side	
Connect	Open connection to remote service and associate with socket
Server Side	
Listen	Prepare to accept remote connections to local service socket
Accept	Accept remote connection to local service socket

Socket Calls — Between App and OS



Socket, Bind
- OS creates resources for network connection
- OS returns to App socket descriptor (socket ID)

Listen
- Server OS makes service available

Connect
- Client OS attempts connection to service

Accept
- Server OS creates new connection socket
- Server OS returns new descriptor to App
- Listen socket continues to listen

Send
- App sends data to OS, pointed at socket
- OS transmits data on associated connection

Receive
- App requests data from OS socket buffer
- OS returns data to App

Connection

Client **Server**

```
s_ID = socket(domain, type, protocol)
                address type  service type
bind(s_ID, s_endpoint, s_endpoint_len)
listen(s_ID, backlog)
```

connection set-up

```
c_ID = socket(domain, type, protocol)
bind(c_ID, c_endpoint, c_endpoint_len)
connect(c_ID, service_endpoint)
```

If client skips **bind()** OS sets a default port

```
s = accept(s_ID, c_endpoint, endpoint_len)
```

data

```
send(s, data, len, flags)
receive(c_ID, buffer, len, flags)
```


Node Addresses

IPv4 — Internet Protocol version 4

- 32-bit address
- 4 octets (bytes) written in decimal — A.B.C.D
- $0 \le A, B, C, D \le 255$
- Disconnected intranet addresses 10.x.x.x and 192.168.0.x
  - Usable in any local intranet
  - Not used in global Internet (between intranets)
  - Network Address Translation (NAT) at gateway router
    - Permits sharing one external IP address for entire Intranet
- 127.0.0.1
  - Loopback address used for testing
  - Address alias for "this node"

IPv6 — Internet Protocol version 6

- 128-bit address
- Repairs service flaws in IPv4

Ports

Client (source) port

Client can assign any unused port number 1024 ≤ port ≤ 65,535

Well-known port (destination port) — standard services

Assigned by Internet Assigned Numbers Authority (IANA)

Examples

Service	Port	Comment
ftp-data	20	File Transfer [Default Data]
ftp	21	File Transfer [Control]
ssh	22	The Secure Shell (SSH) Protocol
telnet	23	Telnet
smtp	25	Simple Mail Transfer
dns	53	Domain Name Service
http	80	World Wide Web HTTP
pop3	110	Post Office Protocol - Version 3
imap	143	Internet Message Access Protocol
https	443	http protocol over TLS/SSL

Telnet

Telnet Protocol — RFC 854

Text-oriented reliable communication over TCP

Telnet client forwards user text to server

Telnet server provides remote login

Command line user interface (CLI) on remote node

Telnet program

Implements telnet client protocol

Connect to telnet server (port 23) on server node

```
$ telnet node
```

Connect to port `port` on server node

```
$ telnet node port
```

Example

Connect to Google web server on HTTP (port 80)

```
$ telnet www.google.com 80
```


HTTP — Hypertext Transfer Protocol

Stateless transaction protocol

Client request formulated in ASCII (printable) text

Server response formulated in MIME structures

No stored data on prior transactions

MIME — Multipurpose Internet Mail Extensions

Text in non-ASCII character sets

Non-text attachments

Multipart message bodies

HTTP/1.0

Non-persistent connection

One transaction per TCP connection

HTTP/1.1

Persistent connection

Multiple transactions on same TCP socket

Pipelined transactions

Multiple requests without waiting for response

HTTP Requests

Simple requests

Defined in HTTP1.1 for HTTP/1.0 compatibility

Request: single line — GET, POST, HEAD

Response: raw page without MIME headers

Full requests

Multi-line request format

Line	Incidence
Request-Line	One
Message-header CRLF	Zero or more
CRLF	Blank line
Message-body	Optional

Method	Function
GET	get page
HEAD	get page header
PUT	store page
POST	append to resource
DELETE	delete page
TRACE	echo back message
CONNECT	tunneling via proxy

Request-Line format

Method	Sp	Request-URI	Sp	HTTP-Version	CRLF

Full Request Example

```
GET /index.html HTTP/1.1\r\n
```

Request line

```
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
```

Header lines

```
\r\n
```

CRLF

HTTP Responses

General structure

Line	Incidence
Status Line	One
message-header CRLF	Zero or more
Data	Zero or more

Status line structure

HTTP version	Sp	status code	Sp	description	CRLF

Example

```
HTTP/1.1 200 OK\r\n
```

Status line

```
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
```

Headers

```
data data data data data ...
```

Data

HTTP Response Codes

100	Continue
101	Switching Protocols
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
204	No Content
205	Reset Content
206	Partial Content
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden

404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Time-out
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request-URI Too Large
415	Unsupported Media Type
416	Requested range not satisfiable
417	Expectation Failed
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Time-out
505	HTTP Version not supported

Telnet into Web Server

```
$ telnet www.hadassah.ac.il 80
```

open connection to Hadassah web server

```
Trying 212.179.79.228...
Connected to hathi.hadassah.ac.il.
Escape character is '^]'.
GET / HTTP/1.0
```

Simple request for index page CRLF

Server response

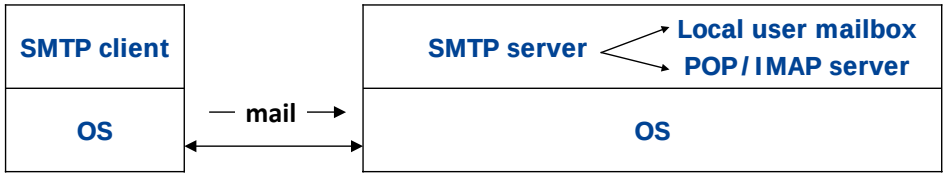
```
HTTP/1.1 301 Moved Permanently
Content-Length: 158
Content-Type: text/html
Location: http://www.hadassah.ac.il/main.html
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Date: Tue, 05 Jul 2011 09:01:35 GMT
Connection: close

<head><title>Document Moved</title></head>
<body><h1>Object Moved</h1>This document may be found <a
    HREF="http://www.hadassah.ac.il/main.html">here</a></bo
    dy>Connection closed by foreign host.
```

email

Simple Mail Transfer Protocol (SMTP)

Defined in RFC 5321



Post Office Protocol (POP3)

Defined in RFC 1939 + 2449 + 1734
Server access with user authentication
Download / delete messages

Internet Message Access Protocol (IMAP)

Defined in RFC 2060
Server access with user authentication
Read / delete / organize messages

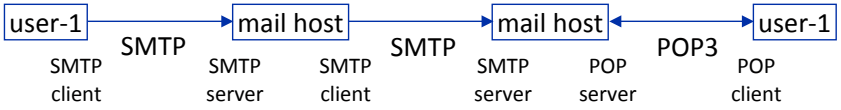
email Use Cases

Unix / Linux mail

Mailbox assigned to each user account
SMTP client sends message → SMTP server → local user mailbox

Outlook / Thunderbird mail

User mailboxes on mail server (host)
Outlook/Thunderbird runs SMTP client + POP3/IMAP clients
SMTP client sends message → SMTP server on user-1 mail host
SMTP client sends message → SMTP server on user-2 mail host
POP3/IMAP client gets message ← POP3/IMAP server on mail host



Webmail

User composes mail on web service
Web service sends mail from SMTP client → user-2 SMTP server
User read mail on web service IMAP client

Telnet to SMTP Server

```
$ telnet mail.bezeqint.net 25                          open connection to Bezeqint SMTP server
Trying 192.115.106.15...
Connected to mail.bezeqint.net.
Escape character is '^]'.
220 sa14.bezeqint.net ESMTP Bezeq International SMTP out Mail Server
helo martin@multinet.net.il                            introduce user
250 sa14.bezeqint.net
mail from:<martin@multinet.net.il>                     begin new message
250 2.1.0 Ok
rcpt to:<martin@multinet.net.il>                       recipient
250 2.1.5 Ok
data                                                   begin message body
354 End data with <CR><LF>.<CR><LF>
this is a line of a message                            message body
.
250 2.7.1 Ok, discarded, id=13560-10 - SPAM            busted!
500 5.5.2 Error: bad syntax
quit                                                   give up
221 2.0.0 Bye
Connection closed by foreign host.
```


Telnet to POP3 Server — 1

```
$ telnet mail.bezeqint.net 110                         open connection to Bezeqint POP3 server
Trying 192.115.106.15...
Connected to mail.bezeqint.net.
Escape character is '^]'.
+OK Bezeqint POP3PROXY Ready 00020c4f
user **********
+OK USER ********** set, mate
pass **********
+OK Maildrop locked and ready
list
+OK scan listing follows
1 500780
2 4432
.
```

POP3 commands	
USER	user id
PASS	password
STAT	number of messages
LIST	size of messages
RETR msg	download message
TOP msg lines	download header + lines
DELE msg	delete
RSET	undelete
QUIT	end session

Telnet to POP3 Server — 2

```
top 1 0                                          request header of message 1 with 0 lines of message body
+OK 500780 octets
Return-Path: <simcha@hadassah.ac.il>
Received: from m5.bezeqint.net (m5.bezeqint.net [192.115.104.85])
        by mas27.bezeqint.net (MOS 3.8.6-GA)
        with ESMTP id HRK58842;
        Mon, 4 Jul 2011 14:19:51 +0300 (IDT)
Received: from baloo-new.hadassah-col.ac.il (EHLO bagheera.hadassah.ac.il)
      ([212.179.79.229])
        by m5.bezeqint.net (MOS 4.1.9-GA FastPath queued)
        with ESMTP id KHN24377;
        Mon, 04 Jul 2011 14:19:50 +0300 (IDT)
Content-class: urn:content-classes:message
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="----_=_NextPart_001_01CC3A3C.603293DC"
Subject: =?windows-1255?B?5Obu8OQ=?=
Date: Mon, 4 Jul 2011 14:18:25 +0300
Message-ID: <27B903C4A7521E47B76F1C36B1CE5ED903044469@arathorn.hadassah-col.ac.il>
From: "Simcha Rozen" <simcha@hadassah.ac.il>
To: <solange@cs.huji.ac.il>, <alhanaty@bezeqint.net>, <martin@multinet.net.il>,
        <yoram.yekutieli@gmail.com>
.
quit                                             end session
+OK
Connection closed by foreign host.
```


File Transfer

File Transfer Protocol (FTP)

- Transfer files between client and server
- Separate control (port 21) and data (port 20) connections

FTP server

- Accesses file system on server node
- Provides Unix-like directory services to client
- Opens, reads, copies file contents to service socket
- Creates, writes file contents from client socket

FTP client

- Requests directory services
- Sends / receives files

Active mode

- Client sends client IP address and port number to server
- Server initiates TCP data connection

Passive mode

- Server sends IP address and port number to client
- Client initiates TCP data connection

FTP Commands

About 80 commands

```
ftp> help
Commands may be abbreviated.  Commands are:
!               dir             mdelete         qc              site
$               disconnect      mdir            sendport        size
account         exit            mget            put             status
append          form            mkdir           pwd             struct
ascii           get             mls             quit            system
bell            glob            mode            quote           sunique
binary          hash            modtime         recv            tenex
bye             help            mput            reget           tick
case            idle            newer           rstatus         trace
cd              image           nmap            rhelp           type
cdup            ipany           nlist           rename          user
chmod           ipv4            ntrans          reset           umask
close           ipv6            open            restart         verbose
cr              lcd             prompt          rmdir           ?
delete          ls              passive         runique
debug           macdef          proxy           send
```

Some common commands

```
!                Send command to local system
open             Connect to remote ftp
close            Terminate ftp session
get              Transfer remote file to local system
put              Transfer local file to remote system
mget / mput      Transfer multiple files using wildcards
lcd              Change local directory
```


Domain Name System (DNS)

Address resolution

Converts between node name and node IP address

```
$ nslookup www.hadassah.ac.il
Server:         10.0.0.180
Address:        10.0.0.180#53

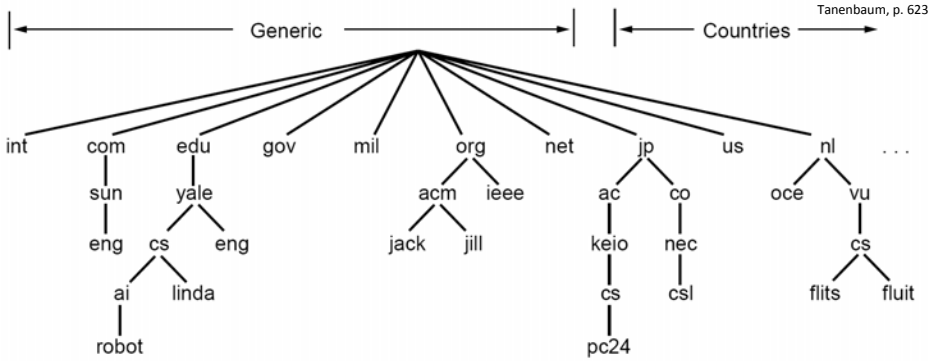
Non-authoritative answer:
Name:   hathi.hadassah.ac.il
Address: 212.179.79.228

$ nslookup 212.179.79.228
Server:         10.0.0.180
Address:        10.0.0.180#53

Non-authoritative answer:
228.79.179.212.in-addr.arpa
name = bzq-179-79-228.static.bezeqint.net
```

Domain Name Space

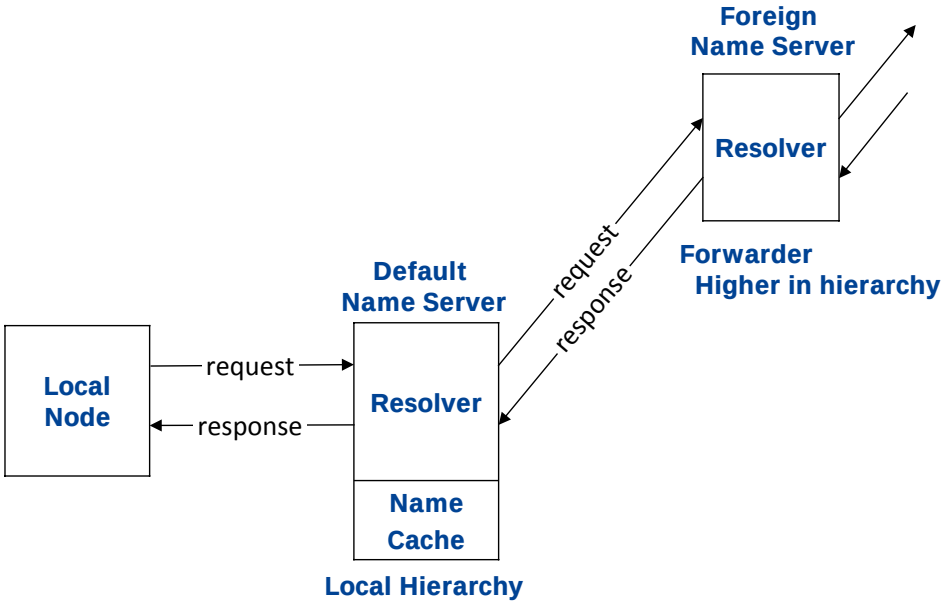
Hierarchal tree



`hathi.hadassah.ac.il`

`hathi`	server	Named server in Hadassah College domain
`hadassah`	organization	Hadassah College domain
`ac`	academic	מרכז החישובים הבינאוניברסיטאי (מחב"א)
`il`	country	איגוד האינטרנט הישראלי (ISOC-IL)

Address Resolution Hierarchy



Hierarchical Name Resolution

Host OS

Resolves addresses using default name server

Windows XP

```
c:\>ipconfig /all
Ethernet adapter Ethernet:
          ...
        IP Address. . . . . . . . . : 10.0.0.5
        Default Gateway . . . . . . : 10.0.0.138
        DNS Servers . . . . . . . . : 10.0.0.11

c:\>nslookup www.hadassah.ac.il
Server:  dns.home
Address:  10.0.0.11
Non-authoritative answer:
Address:  212.179.79.228
Name:  www.hadassah.ac.il
```

Linux

```
$ cat /etc/resolv.conf
# Generated by NetworkManager
search home
nameserver 10.0.0.180
nameserver 10.0.0.11

$ nslookup www.hadassah.ac.il
Server:         10.0.0.11
Address:        10.0.0.11#53

Non-authoritative answer:
www.hadassah.ac.il
Name: www.hadassah.ac.il
Address: 212.179.79.228
```

DNS server

Resolves refers unknown address up the tree

```
$ cat /etc/bind/named.conf.options
forwarders {    192.115.106.10;
                62.219.186.7;
};
```


Resources Records (RR)

RR Field	Description	Length (octets)
NAME	Node name	Length of node name
TYPE	RR type (see table)	2
CLASS	Class code (class = Internet)	2
TTL	Time to live	4
RDLENGTH	Length of RDATA field	2
RDATA	RR-specific data	RDLENGTH

Type Value	Field Name	Description	Function
1	A	address	32-bit IPv4 address
2	NS	name server	Authoritative DNS name servers for domain
5	CNAME	name	Canonical node name
6	SOA	start of authority	Authoritative information about DNS zone: primary name server, email of domain, ...
12	PTR	pointer	Pointer to canonical name (for reverse DNS)
15	MX	mail exchange	List of email agents for that domain
16	TXT	text	Additional information

DNS Request / Response Packets

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Message ID					supplied by questioner and echoed unchanged by responder										
QR	OPCODE				AA	TC	RD	RA	res1	res2	res3	RCODE			
QDCOUNT					number of entries in question section										
ANCOUNT					number of resource records in answer section										
NSCOUNT					number of name server resource records in authority section										
ARCOUNT					number of resource records in additional records section										

QR	Query = 0 / Response = 1
OPCODE	Request operation type 0 = standard query 1 = inverse query 2 = status request
AA	1 = Authoritative Answer
TC	1 = Message truncated (too long for channel)
RD	1 = Recursion Desired
RA	1 = Recursion Available
RCODE	Response type 0 No error condition 1 Format error 2 Server failure 3 Name Error 4 Not Implemented 5 Refused

Client Socket in Perl

Perl IO::Socket package

Provides simplified implementation of socket API

Open new socket

```
use IO::Socket;
$sock = IO::Socket::INET->new(
            Proto    => "Protocol",
            PeerAddr => "Destination Address",
            PeerPort => "Port Number",
            )
      or die "cannot connect to port at localhost";
```

Read data from server

```
$data = <$sock>;
print $data;
```

Send data to server

```
print $sock "hello server!\n";
```


Perl Client Socket Example

Read Hadassah College homepage

```
#!/usr/bin/perl
use IO::Socket;
```

include perl Socket package
open connection to Hadassah web server
OS returns socket descriptor to variable **$sock**

```
$sock = IO::Socket::INET->new(
                Proto    => "tcp",
                PeerAddr => "www.hadassah.ac.il",
                PeerPort => "80",
                )
      or die "Couldn't connect!";
```

send server request for home page

```
print $sock "GET /main.html HTTP/1.0\n\n";

print <$sock>;
```

display on screen data returned from server

```
close($sock);
```

close connection

Perl Client Socket Example

Read Hadassah College homepage

```
#!/usr/bin/perl
use IO::Socket;
$sock = IO::Socket::INET->new(
                Proto    => "tcp",
                PeerAddr => "www.hadassah.ac.il",
                PeerPort => "80",
                )
      or die "Couldn't connect!";
print $sock "GET /main.html HTTP/1.0\n\n";
print <$sock>;
close($sock);
```

Save file as hadassah.pl

At command line

```
$ perl hadassah.pl
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=utf-8
Location: http://www.hadassah.ac.il/en/error-page/
X-AspNet-Version: 4.0.30319
Date: Wed, 19 Nov 2014 17:47:18 GMT
Connection: close

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a
href="http://www.hadassah.ac.il/en/error-page/">here</a>.</h2>
</body></html>
```

Server Socket in Perl

```
#!/usr/bin/perl
use IO::Socket;

$sock = IO::Socket::INET->new(
   LocalHost => 'IP_address',

   LocalPort => port_number,

   Listen => 1,

   Reuse => 1,

   Proto => 'tcp') || die "Error\n";

$client = $sock->accept();
```

open listening socket on server

assign server address

assign service port number

listen system call

reuse port number when process stops

protocol = TCP

accept system call

Perl Server Socket Example

```
#!/usr/bin/perl
$flag = "1";
use IO::Socket;
$sock = IO::Socket::INET->new(
        LocalHost => '127.0.0.1',
        LocalPort => 1234,
        Listen => 1,
        Reuse => 1,
        Proto => 'tcp') || die "Error creating socket\n";
$client = $sock->accept();
while($flag == "1") {
  $line = <$client>;
  print $line;
  print $client "Received\n";
  if ($line =~ /bye/) {$flag = "0";}
}
close($sock);
```

In CLI run perl script

```
$perl servlet.pl
```

In second CLI enter:

```
$telnet 127.0.0.1 1234
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

Server

- Prints data from client Echoes
- `Received`
- Closes on data `bye`

LOOPBACK ADDRESS (for testing)

`127.0.0.1`

Local calls on this machine

Sockets in C

Overview

Header library

Constants, data type, data structures, system calls

Constants

Protocol parameters / sizes, address parameters /sizes, …

Data structures

`struct` templates for addresses, ports, protocols, …

`addrinfo`

`struct` template for connection information

Programmer provides partial endpoint information to instance `hints`

System provides remaining information in instance `res`

System calls

Byte order for network standard

Endpoint parameters based on OS operations

Standard socket calls

Socket, bind, listen, accept, send, receive

Sockets in C

Headers

`<sys/socket.h>`	Socket functions and data structures
`<netinet/in.h>`	`PF_INET` / `PF_INET6` protocol families `AF_INET` / `AF_INET6` address families Theory — multiple AF addresses per protocol family Practice — AF = FP
`<sys/un.h>`	`PF_UNIX` / `PF_LOCAL` address family For communication between programs on same computer
`<arpa/inet.h >`	Functions for manipulating numeric IP addresses
`<netdb.h>`	Functions for translating protocol and host names into numeric addresses

Sockets in C
Some data types and constants

`size_t`	Data type for length of data message
`socklen_t`	Data type for size of endpoint structure
`SOCK_STREAM`	Stream (connection oriented) service
`SOCK_DGRAM`	Datagram (connectionless) service
`AI_PASSIVE`	Allow OS to set local IP address
`INET_ADDRSTRLEN`	Length of IP address string on OS
`PF_INET = AF_INET`	IP protocol family
`PF_INET6 = AF_INET6`	IP version 6
`addrinfo`	**`struct`** template for description of connection
`sockaddr`	**`struct`** template for endpoint (address+port)
`sockaddr_in`	**`struct`** template for building **`sockaddr`**

Sockets in C
Predefined address information (`ai`) template

```
struct addrinfo {
    int     ai_flags;

    int     ai_family;
    int     ai_socktype;
    int     ai_protocol;
    socklen_t  ai_addrlen;
    struct  sockaddr *ai_addr;
    char    *ai_canonname;
    struct  addrinfo *ai_next;
};
```

`AI_PASSIVE` — OS sets address
`AI_CANONNAME` — use hostname
`AF_INET` for IPv4
`SOCK_STREAM` / `SOCK_DGRAM`
protocol / 0
size of `ai_addr` in bytes
pointer to endpoint `struct`
canonical hostname
pointer to linked AI `struct`

Sockets in C
Predefined endpoint information templates

```
struct sockaddr {
    unsigned short      sa_family;

    char                sa_data[14];



};

struct sockaddr_in {
    short int           sin_family;
    unsigned short int  sin_port;
    struct in_addr      sin_addr;
    unsigned char       sin_zero[8];
};
```

Socket address family
`AF_INET` (for IPv4)
protocol address
Port = 2 bytes
IP address = 4 bytes (IPv4)
Padding = 8 bytes (zeros)

Access `sockaddr` fields
`AF_INET`
Port number
IP address as `struct`
zeros (padding)

Sockets in C
`getaddrinfo()`

```
int getaddrinfo(const char *node,
                const char *service,
                const struct addrinfo *hints,
                struct addrinfo **res);
```

System call

Input instance (**`hints`**) ⟶ output instance (**`res`**) using OS information

`node`	Remote node address NULL (OS provides local node address)
`service`	Port number (client's requested service or server's offered service) NULL leaves service uninitialized
`hints`	Input instance of `addrinfo` holding info set by App
`res`	Output instance of `addrinfo` holding info set by OS resolution
`Return`	Status integer — check with function `gai_strerror()`

Sockets in C
Who uses address information (`ai`) templates?

`addrinfo`

Client

Requires **`addrinfo`** instance to describe remote service

Writes remote server address string + remote service to **`hints`**

`getaddrinfo` converts **`hints`** ⟶ **`res`** using OS information

Server

Requires **`addrinfo`** instance to describe local service

Writes NULL address + local service to **`hints`**

`getaddrinfo` converts **`hints`** ⟶ **`res`** using OS information

`sockaddr`

`getaddrinfo` writes **`addrinfo.sockaddr`** as HEX endpoint

`sockaddr_in`

User copies **`addrinfo.sockaddr`** ⟶ **`sockaddr_in`**

Reads port and IP address as separate members

Sockets in C
`getaddrinfo()` example — setting endpoint for LOCAL HTTP server

```
int status;
struct addrinfo hints;
struct addrinfo *local_res;
```

hints = input struct

local_res = output struct

fill **hints** with partial information

```
memset(&hints, 0, sizeof hints);
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_flags = AI_PASSIVE;
```

initialize hints to zero

IPv4

TCP stream

OS provides IP address

system fills **local_res** with local service endpoint = **`local_IP:http`** using **hints** and OS data

```
status = getaddrinfo(NULL, "http", &hints, &local_res)
```

nonzero status — check error

```
fprintf(stderr, "getaddrinfo error: %s\n",
        gai_strerror(status));
```


Sockets in C
`getaddrinfo()` example — setting endpoint for REMOTE server

```
int status;
struct addrinfo hints;
struct addrinfo *remote_res;
```

hints = input struct

remote_res = output struct

fill **hints** with partial information

```
memset(&hints, 0, sizeof hints);
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_flags = AI_PASSIVE;
```

initialize hints to zero

IPv4

TCP stream

OS provides IP address

system fills **remote_res** with local service endpoint = **`remote_IP:80`** using **hints** and OS data

```
status = getaddrinfo(remote_IP, 80, &hints, & remote _res)
```

nonzero status — check error

```
fprintf(stderr, "getaddrinfo error: %s\n",
        gai_strerror(status));
```


Sockets in C
Address conversion

Convert text form to struct (presentation to network)

`inet_pton()`

Example

```
struct sockaddr_in sa;
```

define instance sa of struct sockaddr_in

convert address to struct and write into sa

```
inet_pton(AF_INET, "10.0.0.1", &(sa.sin_addr));
```

Convert struct to text form (network to presentation)

`inet_ntop()`

Example

```
char ip4[INET_ADDRSTRLEN];
struct sockaddr_in sa;
```

INET_ADDRSTRLEN preset by system

convert address and write into ip4

```
inet_ntop(AF_INET, &(sa.sin_addr), ip4,
   INET_ADDRSTRLEN);
```

Sockets in C
Byte order conversion

Big-endian

Integer stored in memory with MSB in lowest address

Standard for network operations (Network Order)

Little-endian

Integer stored in memory with LSB in lowest address

Standard on Intel processors

Conversion functions

Converts Host Order to Network Order

No change if Host Order = big-endian

`htons()`	host to network short (short = 2 bytes = 16 bits)
`htonl()`	host to network long (long = 4 bytes = 32 bits)
`ntohs()`	network to host short
`ntohl()`	network to host long

Sockets in C
Example for setting remote endpoint — part 1

```
/* showip.c show IP addresses for remote host */
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>

int main(int argc, char *argv[])
{
    struct addrinfo hints, *remote_res, *p;
    int status;
    char ipstr[INET_ADDRSTRLEN];
    if (argc != 2) {
        fprintf(stderr,"usage: showip hostname\n");
        return 1;
    }
```

3 instances

character array to hold IP addresses

Sockets in C
Example for setting remote endpoint — part 2

initialize hints to zero

```
memset(&hints, 0, sizeof hints);
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
```

set known hints
IPv4
TCP

set **remote_res** to system-provided info
argv[1] = hostname
port = 0 (not set)

```
if ((status = getaddrinfo(argv[1], NULL, &hints,
                          &remote_res)) != 0) {
    fprintf(stderr, "getaddrinfo: %s\n",
        gai_strerror(status));
    return 2;
}
    printf("IP addresses for %s:\n\n", argv[1]);
```

print error message on fail

print heading for list of addresses

Sockets in C
Example for setting remote endpoint — part 3

loop over linked list

```
for (p = remote_res; p != NULL; p = p->ai_next) {
    void *addr;
    char *ipver;

    struct sockaddr_in *ipv4 =
                    (struct sockaddr_in *)p->ai_addr;

    addr = &(ipv4->sin_addr);
    ipver = "IPv4";

    inet_ntop(p->ai_family, addr, ipstr, sizeof ipstr);
    printf("  %s: %s\n", ipver, ipstr);
}
    freeaddrinfo(res);
    return 0;
}
```

instance **ipv4** ← instance **p.ai_addr**
ai_addr points to **sockaddr** = endpoint = IP:port

addr ← IP address in **p.ai_addr**

convert **addr** = IP address to **ipstr** = character string

print IPv4: **ipstr**

free linked list

Sockets in C
Running **showip**

```
$ showip www.google.co.il
IP addresses for www.google.co.il:

  IPv4: 74.125.230.148
  IPv4: 74.125.230.144
  IPv4: 74.125.230.145
  IPv4: 74.125.230.146
  IPv4: 74.125.230.147
```


Sockets in C
socket() with hard coded **endpoint type**

```
int socket(int domain, int type, int protocol);
```

domain	Address family
type	Service type
protocol	Specify protocol or 0 default — TCP for STREAM or UDP for DGRAM

```
int s;
s = socket(PF_INET, SOCK_STREAM, 0);
```

socket descriptor

PF_INET = AF_INET = IPv4
SOCK_STREAM = reliable transport
0 = use default service protocol — TCP for STREAM or UDP for DGRAM

No local or remote endpoint required to open socket

Sockets in C
socket() using fields in **struct res**

```
int socket(int domain, int type, int protocol);
```

domain	Address family
type	Service type
protocol	Specify protocol or 0 default — TCP for STREAM or UDP for DGRAM

```
int s;
struct addrinfo hints, *res;
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;

getaddrinfo("www.hadassah.ac.il", "http", &hints, &res);

// getaddrinfo(NULL, "http", &hints, &res);

s = socket(res->ai_family, res->ai_socktype,
           res->ai_protocol);
```

socket descriptor

set up **hints**

client side for remote server

server side for local server

socket call using **res**

Sockets in C
bind()

```
int bind(int sockfd, const struct sockaddr *my_addr,
         socklen_t addrlen);
```

prepare **hints**

```
int s;
struct addrinfo hints, *local_res;
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_flags = AI_PASSIVE;
```

OS sets node IP address

resolve local address and port to **local_res** , open socket as server

```
getaddrinfo(NULL, "http", &hints, &local_res);
s = socket(local_res->ai_family, local_res->ai_socktype,
     local_res->ai_protocol);
```

bind socket to local service using **ai_address** endpoint (instance of **sockaddr**)

```
bind(s, local_res->ai_addr, local_res->ai_addrlen);
```

Sockets in C
listen() and accept()

```
int listen(int sockfd, int backlog);
```

```
listen(s, backlog);
```

listen on socket **s**
allow connections ≤ **backlog**

```
struct sockaddr_storage cliaddr;
socklen_t addr_size;
addr_size = sizeof cliaddr;
```

sockaddr_storage
struct for connected endpoint
designed for IPv4 and IPv6
will be cast as **sockaddr** in **accept()**

```
int accept(int sockfd, struct sockaddr *cliaddr,
           socklen_t *addrlen);
```

```
int c_s;
```

create copy **c_s** of socket **s** for connection
accept connection on socket **c_s**
cast **cliaddr** as **sockaddr**
write client endpoint information to **cliaddr**

```
c_s = accept(s, (struct sockaddr *)&cliaddr,&addr_size);
```


Sockets in C
connect()

```
int connect(int sockfd,
            const struct sockaddr *serv_addr,
            socklen_t addrlen);
```

```
int s;
```

prepare **hints**, resolve remote **addrinfo**

```
struct addrinfo hints, *remote_res;
hints.ai_family = AF_INET;
hints.ai_socktype = SOCK_STREAM;
```

resolve remote address and port to **remote_res**

```
getaddrinfo("www.google.com, "http", &hints, &remote_res);
```

open socket **s** for connecting to remote service using **ai_address** endpoint

```
s = socket(remote_res->ai_family, remote_res->ai_socktype,
     remote_res->ai_protocol);

connect(s, remote_res->ai_addr, remote_res->ai_addrlen)
```


Sockets in C
send()

```
int send(int sockfd, const void *msg, size_t len,
         int flags);
```

send string on socket **c_s**
returns number of bytes actually sent

```
char *msg = "Hello, world!";
size_t s_len,
int bytes_sent;
s_len = strlen(msg);
bytes_sent = send(c_s, msg, s_len, 0);
```


Sockets in C
recv()

```
int recv(int sockfd, void *buf, size_t len, int flags);
```

receive string on socket **c_ s**
returns number of bytes actually received

choose desired receive buffer size

```
#define BUFFER_SIZE 100
char buffer[BUFFER_SIZE];
size_t r_len = BUFFER_SIZE;
int bytes_received;
bytes_received = recv(c_s, buffer, r_len, 0);
```

Sockets in C

Connectionless data transfer — sendto() and recvfrom()

```
int send(int sockfd, const void *msg, size_t len,
         int flags);
```

No connect() for local sockfd
Define destination endpoint on the fly

```
int sendto(int sockfd, const void *msg, size_t len,
           int flags, const struct sockaddr *to,
           socklen_t tolen);
```

```
int recv(int sockfd, void *buf, size_t len, int flags);
```

No accept() for local sockfd
Define source endpoint on the fly

```
int recvfrom(int sockfd, void *buf, size_t len,
             int flags, struct sockaddr *from,
             size_t *fromlen);
```

Transport Layer

Application / Transport Interface

Application requests service from transport layer

Application Layer	Prepare Transport service requirements Data for transport Local endpoint — node address + service port Remote endpoint — node address + service port
Transport Layer	Receives Application layer PDU — data for transport Interface Control Information (ICI) Service requirements Local + remote endpoints

Transport Layer Services

Connection management

- Maintain separate sessions between various user applications
- Label sessions with client / server port numbers
- Provide reliability services according to connection type

Multiplexing

- Divide outgoing data stream into segments
- Combine segments from multiple sessions into single output

Demultiplexing

- Separate incoming segments by sessions
- Combine incoming segments for each session into user data stream

Transport Layer Connections

Reliable transport (TCP)

- Connection-oriented
  - TCP connection established before data transfer
- Error-free delivery
  - Data delivered
    - In original order
    - No errors, duplications, omissions
- Flow control
  - Control sender rate to prevent buffer overflow in receiver
- Congestion control
  - Control sender rate to prevent buffer overflow in network

Unreliable transport (UDP)

- Connectionless
- Lower overhead ⇒ faster but no guarantees
- Segments with errors discarded with no warning to application

Source / Destination Ports

Client

Opens socket to send requests

Client / OS binds port number to socket

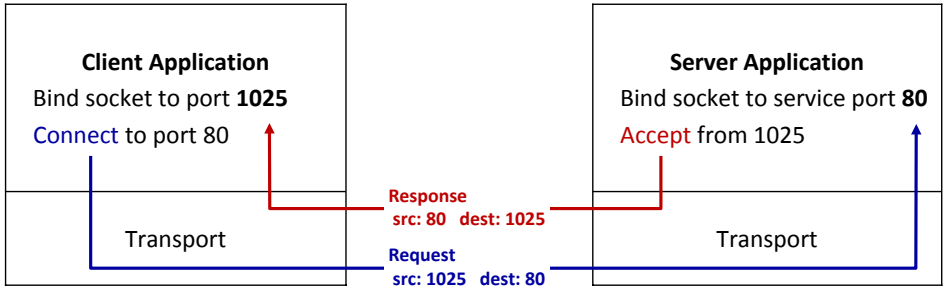
1024 ≤ client port ≤ 65,535 identifies client application

Server

Opens listen socket mapped to accept sockets for requests

Binds well-known port to service socket

0 ≤ well-known port ≤ 1023 identifies service application



Multiplexing / Demultiplexing

Applications send / receive data on sockets

Multiple sockets ⇒ multiple conversations

Transport layer segment

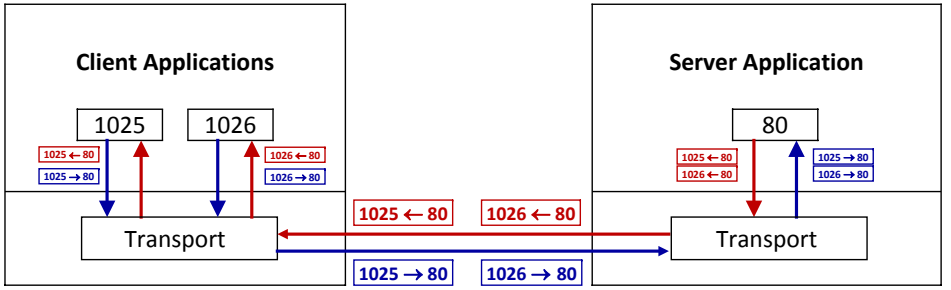
Transport header + application data (PDU)

TCP / UDP headers carry source + destination ports

Multiplexing / demultiplexing

Segments transmitted on same infrastructure

Sorted by destination port at destination



User Datagram Protocol (UDP)

Internet unreliable transport protocol

Defined in RFC 768

Used when low delay / jitter more important than error control

Streaming multimedia, multiplayer games, …

UDP segment

header	application data

UDP header

32 bits

source port	destination port
length	checksum

Length

Number of bytes in datagram < 2^{16} = 65,536

Maximum length = 64 KB – lengths of all headers

UDP / TCP Checksum

Calculation at source

Break UDP segment into sequence of 16-bit words

Add IP pseudo-header (IP src/dest addresses, protocol, length)

Pad with zeros if necessary

Add all 16 bit words (not counting checksum field)

Add carry-out to 0-order bit (rotate)

Perform 1's complement

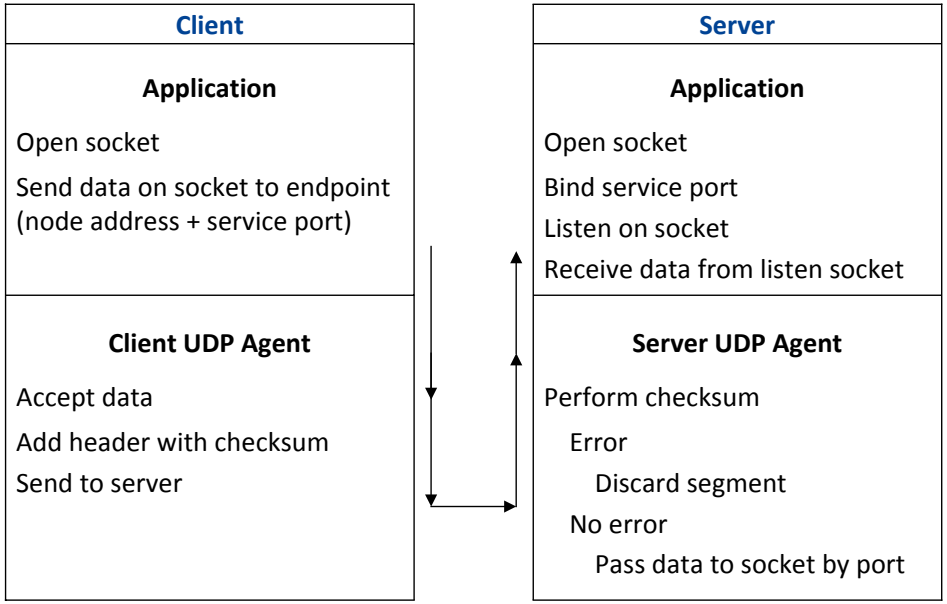
Calculation at destination

Repeat calculation and compare

Example

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
rotate	(1)	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
																1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Using UDP



If required — add reliability features at client / server application level

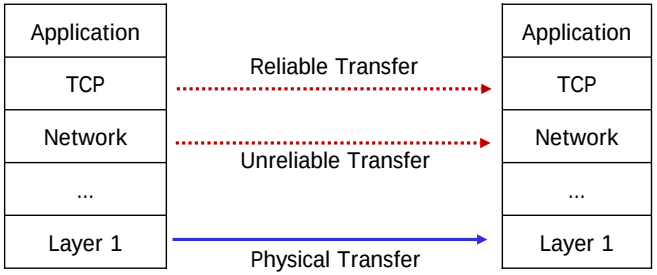
Transport Control Protocol (TCP)

Reliable transport over unreliable layers

Network + infrastructure layers can introduce errors

- Bit errors (1 ↔ 0)
- Lost packets
- Duplicate packets
- Out-of-order packets

TCP detects and corrects errors



Basic Theory of Reliable Data Transfer (RDT)

Bottom-up approach

Version 1.0 for error-free channel

No error checking necessary

Version 2.0

Handles bit errors

Version 2.1

Bug fix of version 2.0

Version 2.2

Simplification of version 2.1

Version 3.0

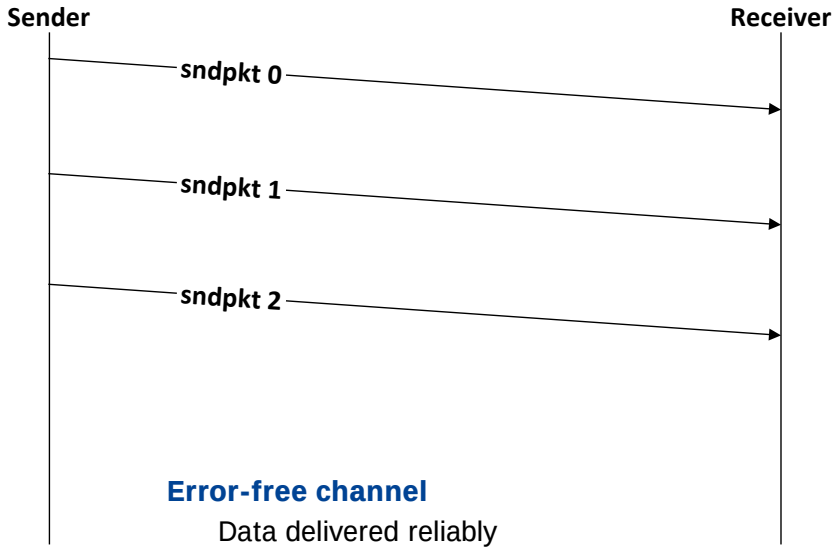
Handles missing packets

Version 4.0

Improved performance

Theory of Reliable Data Transfer (RDT)

Version 1.0 on error-free channel



Error-free channel

Data delivered reliably

Handling Bit Errors

Error detection

Sender adds checksum to packet header

Receiver

- Re-computes checksum
- Compares with checksum in header

Corrupt packets (packets with errors) discarded

Error control

Receiver provides sender with feedback about received data

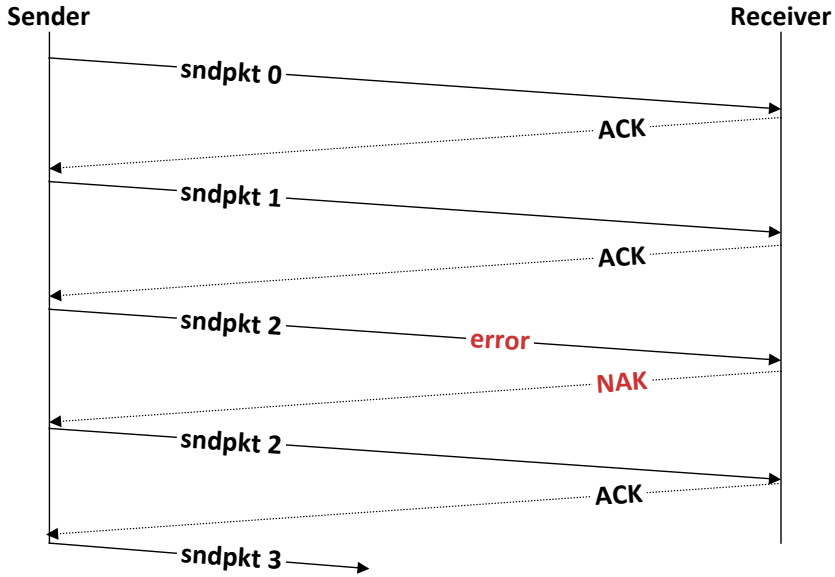
ACK	Acknowledgement	Data received without bit errors
NAK	Negative acknowledgement (reject)	Data received with bit errors

Automatic Repeat Request (ARQ)

Sender re-transmits on NAK

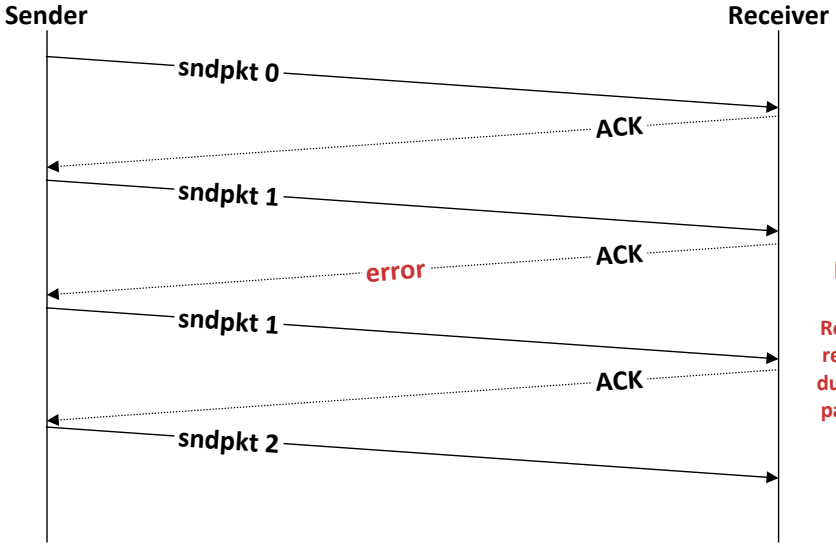
Theory of Reliable Data Transfer (RDT)

Version 2.0 on channel with bit errors



Theory of Reliable Data Transfer (RDT)

Version 2.0 on channel with bit errors



Bug in Version 2.0

Corrupt ACK / NAK packet

Option 1

- Sender interprets corrupt ACK / NAK = ACK
- Receiver misses data packet

Option 2

- Sender interprets corrupt ACK / NAK = NAK
- Sender retransmits packet
- Receiver may receive duplicate packet

Bug fix

Label packet with sequence number (SEQ)

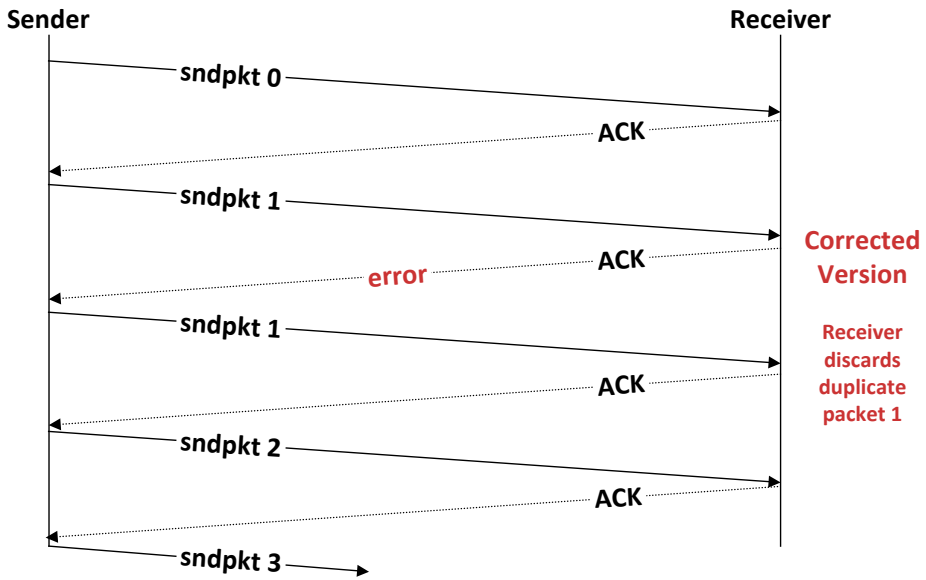
- SEQ = packet_number mod 2 = 0 or 1 (sufficient for stop and wait)

Sender follows option 2

Receiver gets duplicate packet

- Re-transmits ACK
- Discards duplicate packet

Theory of Reliable Data Transfer (RDT)
Version 2.1 with SEQ



Complication in Version 2.1

Version 2.1 — SEQ + ACK + NAK

Sender
- ACK or NAK = response to last packet
- Corrupt ACK or NAK = NAK
- NAK ⇒ re-transmit packet

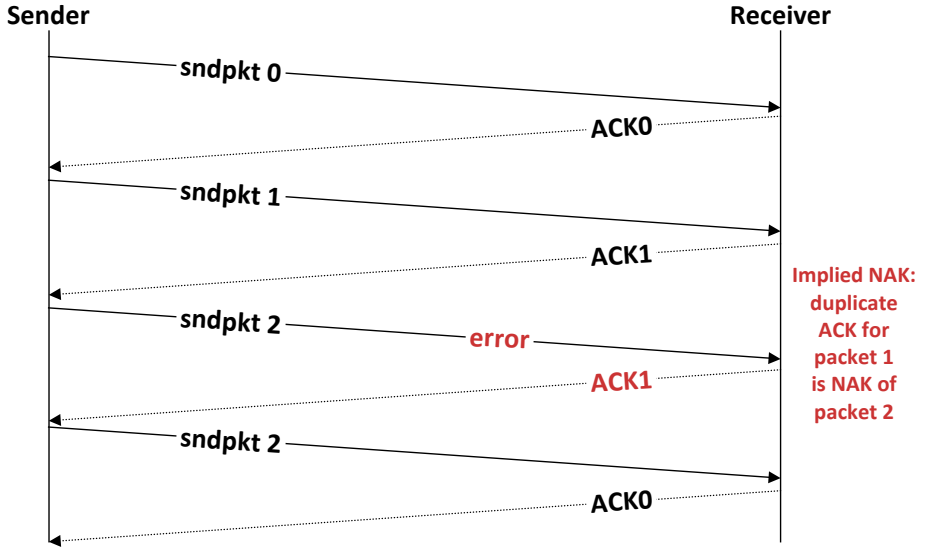
Receiver
- ACK or NAK for each packet
- Corrupt packet ⇒ re-transmit packet
- Duplicate SEQ ⇒ re-transmit ACK but discard packet

Version 2.2 — SEQ + ACK without transmitted NAK

Sender
- ACK + SEQ = response to packet SEQ
- Corrupt ACK || ACK to previous SEQ = implied NAK
- Implied NAK ⇒ re-transmit packet

Receiver
- ACK + SEQ for each packet
- Corrupt packet ⇒ re-transmit ACK for **previous SEQ**
- Duplicate SEQ ⇒ re-transmit ACK but discard packet

Theory of Reliable Data Transfer (RDT)
Version 2.2 with SEQ + ACK without NAK



Lost Packets

Packet loss

Lost data packet or ACK

Discarded by intermediate system or lower network layer

Buffer overflow

At receiver — no error or indication of lost packet

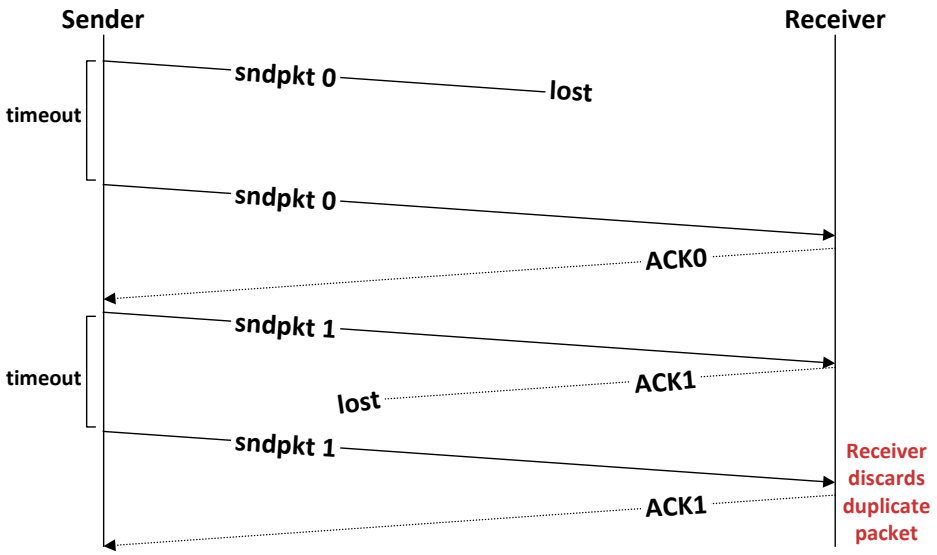
Handling lost packets

Sender sets timeout counter

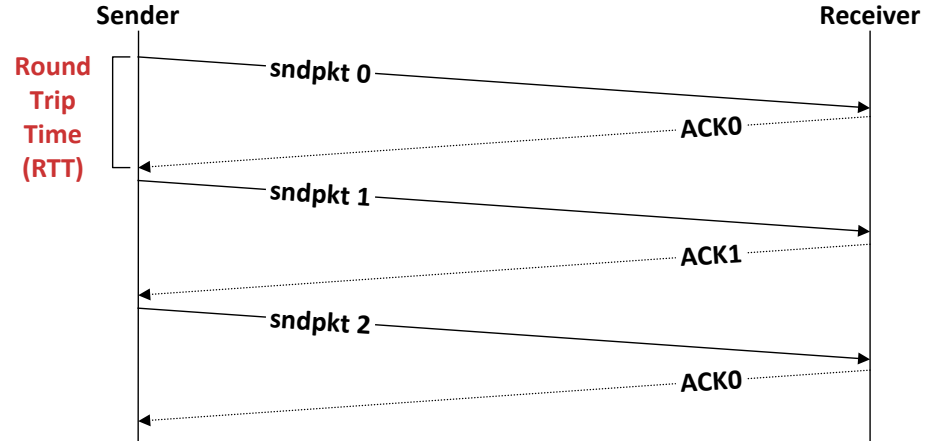
Timeout || corrupt ACK || ACK to previous SEQ = implied NAK

Duplicates handled by SEQ

Theory of Reliable Data Transfer (RDT)
Version 3.0 with SEQ + ACK + Timeout



Performance Problem with Stop-and-Wait
Wait too long



$$T_T = \text{transmission delay} \approx \frac{64\ \text{KB} \times 8\ \text{bits/B}}{10\ \text{Mbps}} = 5\ \text{ms}$$

Typical RTT > 30 ms

$$\text{utilization} = \frac{T_T}{T_T + \text{network latency}} = \frac{T_T}{T_T + \text{RTT}} \approx \frac{5\ \text{ms}}{5\ \text{ms} + 30\ \text{ms}} = 0.14$$

Theory of Reliable Data Transfer (RDT)
Improving performance

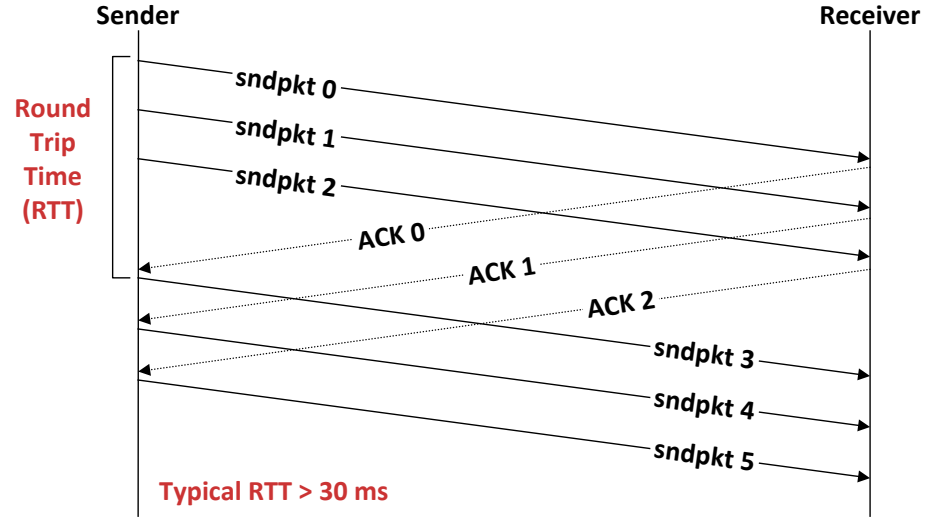
Sender transaction
- Send packet
- Receive ACK

Stop-and-wait protocol
- Send packet → wait → receive ACK
- Finish transaction n → begin transaction n + 1
- Transmitter idle while waiting for ACK

Pipelined protocol
- Begin new transaction before previous finishes
- Reduce transmitter idle time

Source window
- N = window size
- Transmit N packets before stop-and-wait
- N packets in process ("in flight" or "in pipeline") at any time

Theory of Reliable Data Transfer (RDT)
Pipelining with window size = 3



Typical RTT > 30 ms

$$\text{utilization} = \frac{N \times T_T}{T_T + \text{RTT}} \approx \frac{3 \times 5\ \text{ms}}{5\ \text{ms} + 30\ \text{ms}} = 0.43$$

Theory of Reliable Data Transfer (RDT)
Pipelined protocol — error handling method 1

Selective repeat

Sender

- Buffers packet until ACK
- Timeout counter for each packet
- Timeout ⇒ sender re-transmits "unACKed" packet

Receiver

- ACKs received packets
- Received in order — passed to application
- Received out-of-order — held in buffer

Window size = N

"unACKed" packets < N

Send N packets without ACK ⇒ stop sending

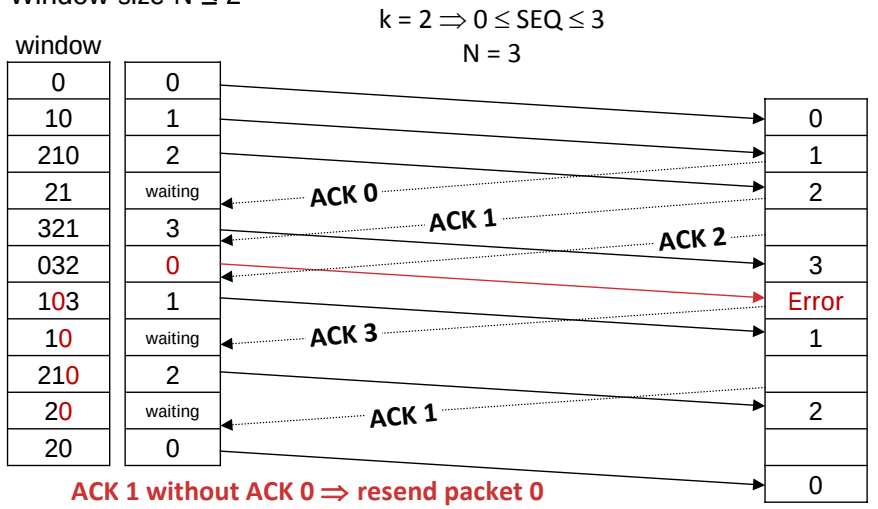
N < receiver buffer size

Theory of Reliable Data Transfer (RDT)
Selective repeat

Sequence number (SEQ)

k bit SEQ ⇒ $0 \le SEQ \le 2^k - 1$ (modulo 2^k)

Window size $N \le 2^k$

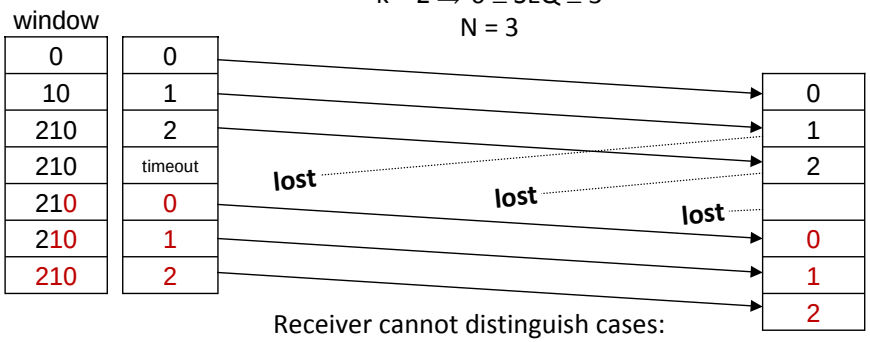


Theory of Reliable Data Transfer (RDT)
Bug in selective repeat

Sequence number (SEQ)

k bits ⇒ $0 \le SEQ \le 2^k - 1$ (modulo 2^k)

Window size $N \le 2^k$



Receiver cannot distinguish cases:

1. Re-transmission of duplicates
2. New packets

Solution — long SEQ (≥ 32 bits)

Theory of Reliable Data Transfer (RDT)
Pipelined protocol — error handling method 2

Go Back N (GBN)

Sender

- Buffers packet until ACK
- Timeout counter for oldest "unACKed" packet
- Timeout ⇒ sender re-transmits all "unACKed" packets from buffer

Receiver sends **CUMULATIVE ACK**

- ACK for last packet
  - Implies ACK for all previous packets
- No buffer for out-of-order packets
- Error or missing packet ⇒ no ACK packet or any subsequent packet

Window size = N

"unACKed" packets < N

Send N packets without ACK ⇒ stop sending

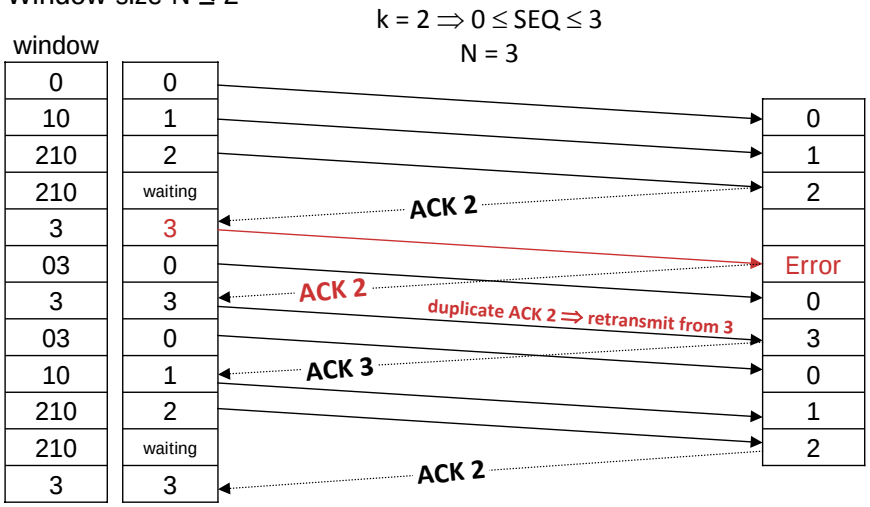
N < transmit buffer size

Theory of Reliable Data Transfer (RDT)
Version 4.0 — Go Back N

Sequence number (SEQ)

k bits ⇒ $0 \le SEQ \le 2^k - 1$ (modulo 2^k)

Window size $N \le 2^k$



TCP as Reliable Transport Protocol

Connection-oriented

Set up connection before data transfer

Maximum Segment Size (MSS)

64 KB (including IP header)

Error detection

Checksum as in UDP

Error control

ARQ with ACK + SEQ + timeout

No corrupt, missing, duplicate, or out-of-order data

Piggybacking — send ACK within data segment

Pipelining

Variable window size at sender and receiver

GBN cumulative ACK with optional selective repeat

Flow /congestion control

Dynamic window size ⇒ control of sender utilization

TCP Header

32 bits

source port			destination port
sequence number (SEQ)			
acknowledgement number (ACK)			
HLEN	not used	flags	window size
checksum			urgent pointer
Options			

HLEN (data offset)	4 bits	Length of TCP header in 32-bit words
Reserved	3 bits	Not used
Flags	9 bits	Control bits
Window size	16 bits	Number of bytes receiver can receive now
Urgent pointer	16 bits	Offset from SEQ points to last urgent data byte
Options	10 – 320 bits	Options fields + padding for multiple of 32 bits

TCP Header
Flags

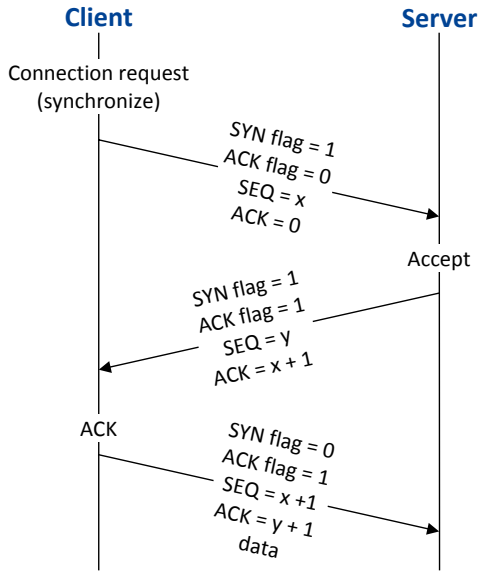
NS	ECN-nonce concealment protection
CWR	Congestion Window Reduced (CWR) flag Sender indicates receiving segment with ECE flag set
ECE	ECN-Echo If SYN = 1 peer is ECN capable If SYN = 0 packet with Congestion Experienced flag in IP header received during normal transmission
URG	Urgent pointer field valid
ACK	**Acknowledgment field valid**
PSH	Push buffered data to receiving application
RST	**Reset connection**
SYN	**Synchronize sequence numbers**
FIN	**No more data from sender**

TCP Connection Set-up
Three-way handshake

Client SYN segment
- SYN flag = 1
- SEQ = random number x
- No data

Server SYN-ACK segment
- SYN flag = ACK flag = 1
- SEQ = random number y
- ACK = x + 1
- No data

Client ACK segment
- SYN flag = 0
- ACK flag = 1
- SEQ = random number x + 1
- ACK = y + 1
- May contain data

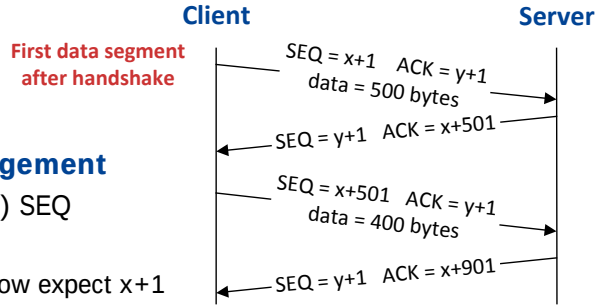


TCP SEQ + ACK

Initial sequence number (ISN)
- Random SEQ in 3-way handshake
  - Prevents counterfeit segments
- At end of handshake SEQ = ISN + 1

SEQ — byte sequencing
SEQ = previous SEQ + length(data bytes in previous segment)
= ISN + 1 + data bytes sent in all **previous** segments

ACK — byte acknowledgement
ACK = next (expected) SEQ
ACK = x + 1
Acknowledge x — now expect x+1



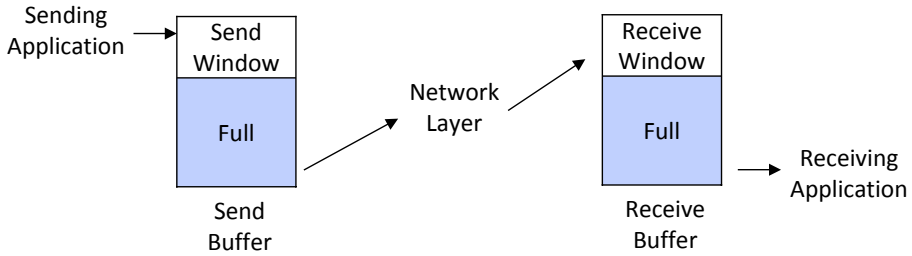
Send and Receive Windows

Send buffer
- Holds sent segments until ACKed
- On timeout resend segments from send buffer

Send window
- Available space in send buffer

Receive buffer
- Holds received segments until requested by application

Send window
- Available space in send buffer



Basic TCP Sender Behavior

Start
SEQ = SendBase = ISN + 1

Receive data from application
- Prepare segment with SEQ
- Send segment when unACKed bytes < SendWindow
- SEQ = SEQ + length(segment data)
- Start timer

Receive ACK = y
- y > SendBase ⇒ SendBase = y
- unACKed bytes > 0 ⇒ restart timer

Timeout
- Resend segments starting from SendBase
- Restart timer

Typical Control Program

```
default_inits ;
while ( true ) {                /* main loop */
    if ( condition_1 ) {
      handle_condition_1 ;
    }

    if ( condition_2 ) {
      handle_condition_1 ;
    }

    if ( condition_3 ) {
      handle_condition_1 ;
    }

    default_updates ;
}
```


Simplified TCP Sender — 1

```
// initialize
    SEQ = ISN + 1
    SendBase = ISN + 1              // last byte ACKed
    InFlight = 0                    // unACKed bytes sent
    RTO                             // timeout interval
    SendWindow, RecvWindow          // send & receive windows

// main loop
    if (new data from application)
        Prepare data segment
            sequence number = SEQ
            SEQ = SEQ + length(data)
        if InFlight < min{SendWindow,RecvWindow}
            Pass segment to IP
            InFlight = InFlight + length(data)
            if !(timer running) timer = RTO
```


Simplified TCP Sender — 2

```
    if (receive ACK = y)
        stop timer
        if (y > SendBase)
            newACKs = y – SendBase        // bytes ACKed
            SendBase = y
            InFlight = InFlight – newACKs
            if (InFlight > 0) timer = RTO

    if (timeout)
            SEQ = SendBase = min{unACKed SEQ} and resend
            timer = RTO
```


Basic TCP Receiver Behavior

Start

Expected = ISN + 1 // sender ISN

Receive error-free segment with SEQ < Expected

Resend ACK = Expected

Ignore packet

Receive segment with error || SEQ > Expected

Resend ACK = Expected (implied NACK)

Ignore packet

Receive error-free segment with SEQ = Expected

Expected = Expected + length(segment data)

Buffer ACK

Send ACK = Expected if

- Delayed ACK timer > maximum
- Buffered ACKs > maximum
- ACK following implied NACK

Simplified TCP Receiver — 1

```
// initialize
    Set RecvWindow = receiver buffer size
    expected = Sender ISN + 1
    ack_buffer = 0          // received unACKed segments
    ack_max                 // delayed ACK trigger
    ack_delay = 250 msec    // local policy: < 500 msec
    Start ACK delay timer = ack_delay

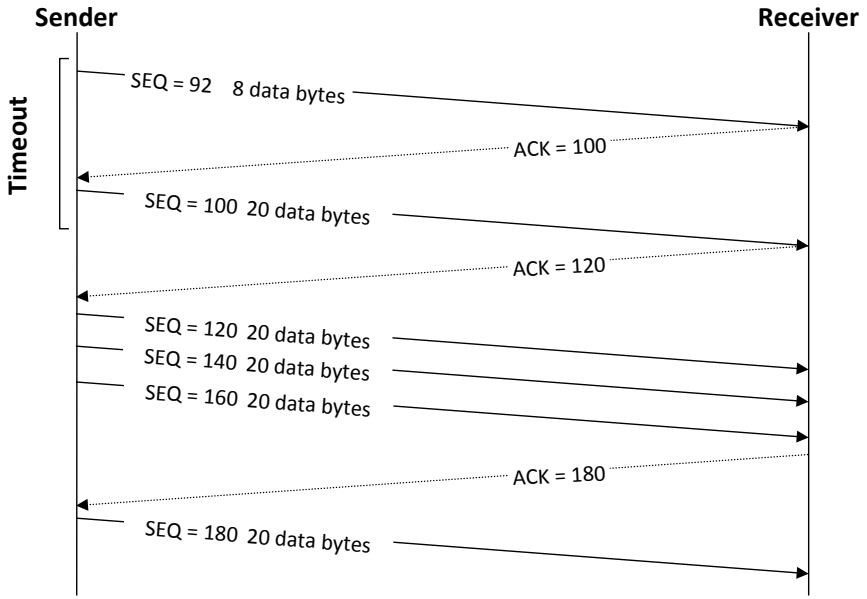
// main loop
if (ACK delay timer = 0 && ack_buffer > 0)
    Send ACK = expected with updated RecvWindow
    ACK delay timer = ack_delay
    ack_buffer = 0
```


Simplified TCP Receiver — 2

```
if (receive SEQ = x)                    // new packet arrives
    if (x = expected && error-free)
        expected = expected + length(data)
        if (NACK = 1)
            Send ACK = expected with updated RecvWindow
            ACK delay timer = ack_delay
            ack_buffer = 0
            NACK = 0
        else if (ack_buffer < ack_max)
            nextACK = expected
            ack_buffer++
        else if (ack_buffer = ack_max)
            Send ACK = expected with updated RecvWindow
            ACK delay timer = ack_delay
            ack_buffer = 0
    else
        Send ACK = expected with updated RecvWindow
        ACK delay timer = ack_delay
        NACK = 1
```

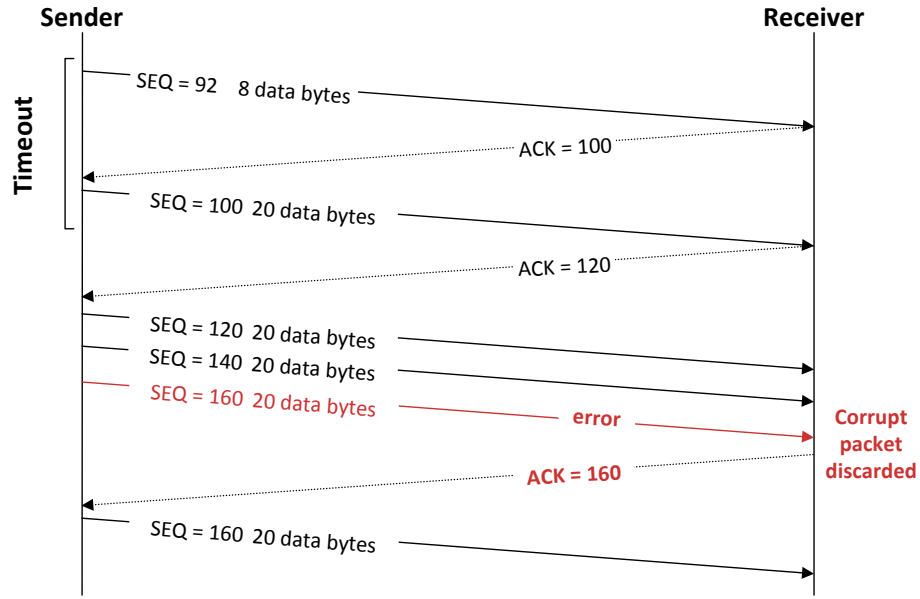

TCP SEQ + ACK

No errors

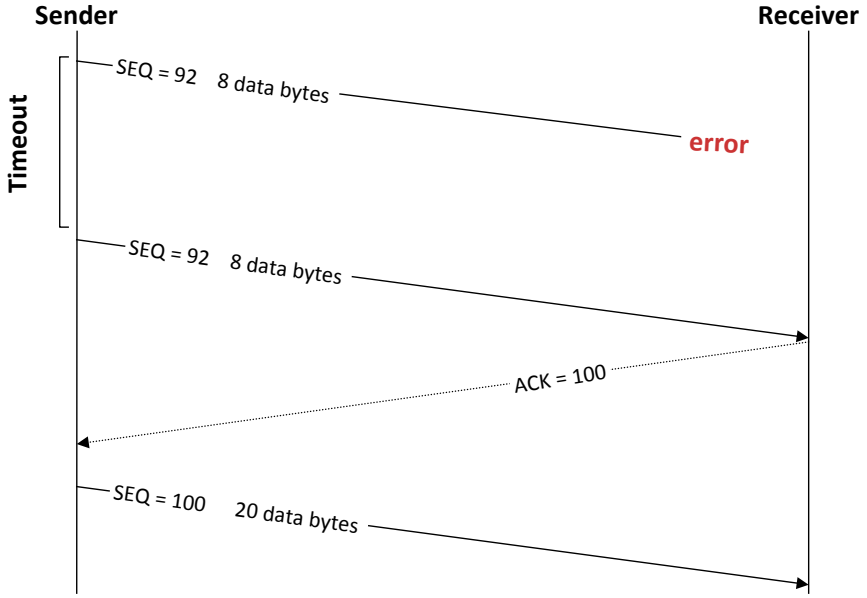


TCP SEQ + ACK

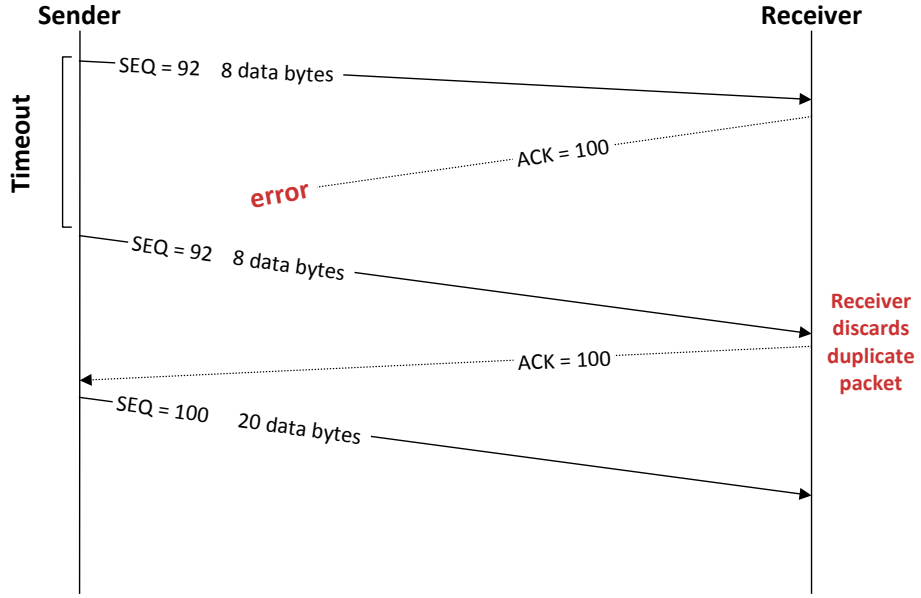
Bit errors



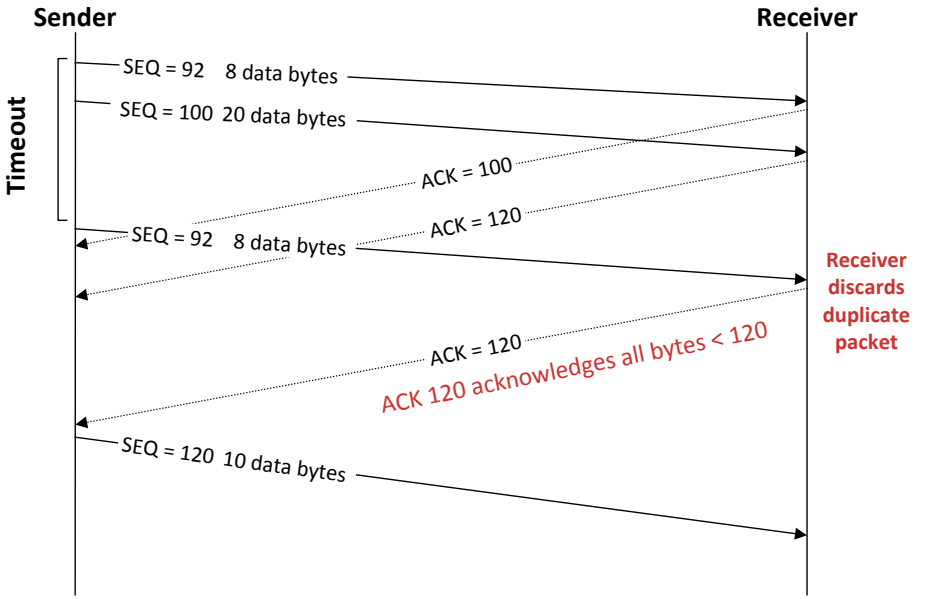
TCP SEQ + ACK
Timeout



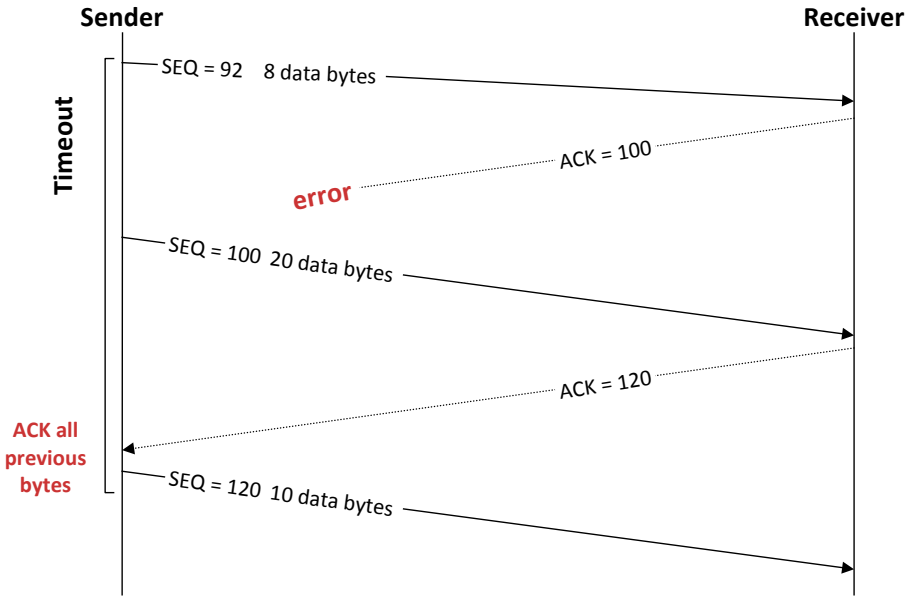
TCP SEQ + ACK
Lost ACK



TCP SEQ + ACK
Missed ACK



TCP SEQ + ACK
Cumulative ACK



Selective Acknowledgment Option

Selective ACK (SACK)

Permits ACK for segments with gaps

Option negotiated between hosts

Defined in RFC 2018

Example

Last ACK = 5000

Send 8 segments × 500 data bytes / segment

Case 1

First 4 segments received and last 4 dropped

Receiver returns normal ACK = 5000 + 4 * 500 = 7000

No SACK option field

Case 2

First segment lost and 7 segments received

For each segment receiver returns segment with

ACK = 5000

SACK option field with start + end ACK

Data	ACK	Option Field	
		Start	End
5000	—	—	—
5500	5000	5500	6000
6000	5000	5500	6500
6500	5000	5500	7000
7000	5000	5500	7500
7500	5000	5500	8000
8000	5000	5500	8500
8500	5000	5500	9000

Setting Retransmission Timeout (RTO) Value

RTO > RTT

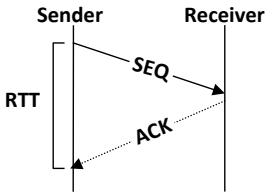
RTT = round trip time = minimum time to receive ACK

Timeout too short

Too many missed ACKs

Timeout too long

Waste too much time before re-transmission



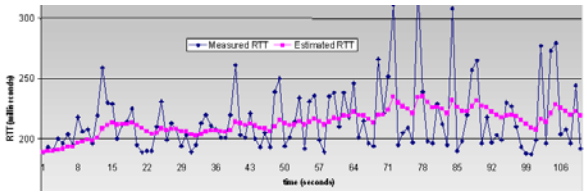
Method

Measure **RTT**

On each new ACK update

RTO ← max{(1 - α)* RTO + α * measured_RTT, 1 sec}

Typical: **α = 0.125**



TCP Connection Close

Symmetric

Client or server may close connection

FIN segment

FIN flag = 1

SEQ = cumulative SEQ number

ACK segment

ACK flag = 1

ACK = SEQ + 1

FIN segment

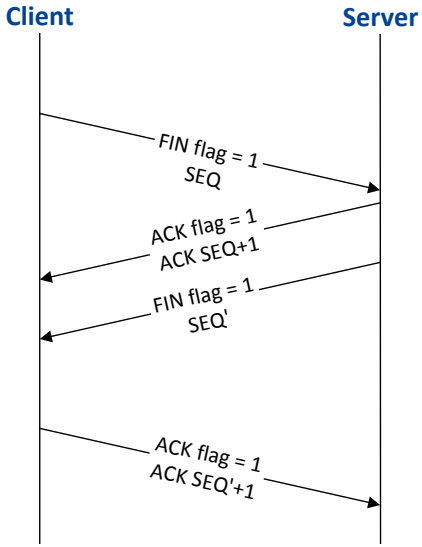
FIN flag = 1

SEQ' = cumulative SEQ number

ACK segment

ACK flag = 1

ACK = SEQ' + 1



Flow Control and Congestion Control

Flow control

Sender avoids overflow of receiver buffer

Congestion control

All sender avoid overflow of intermediate network buffers

Buffer fill rate

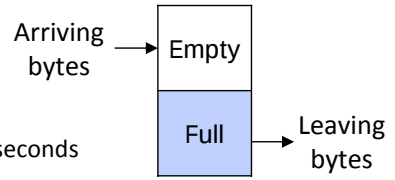
Bytes / second arriving from network

Buffer empty rate

Bytes / second leaving to network or application layer

Buffer file time

$$T_{overflow} = \frac{\text{buffer size}}{\text{buffer fill rate} - \text{buffer empty rate}}$$



Example

$$T_{overflow} = \frac{64\ \text{KB}}{8\ \text{KB/sec} - 4\ \text{KB/sec}} = \frac{64\ \text{KB}}{4\ \text{KB/sec}} = 16\ \text{seconds}$$

Flow Control

Source window

Initial source window = maximum number of "unACKed" bytes

Determined by congestion + flow control

Destination window

Number of bytes receiver can accept

Determined by available space in receiver buffer

Buffer level = Previous level + arriving bytes – bytes read by App

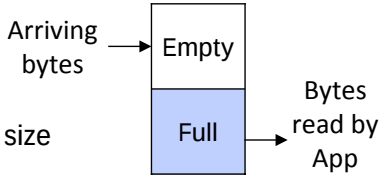
Application reads too slowly ⇒ decrease destination window

Sliding window

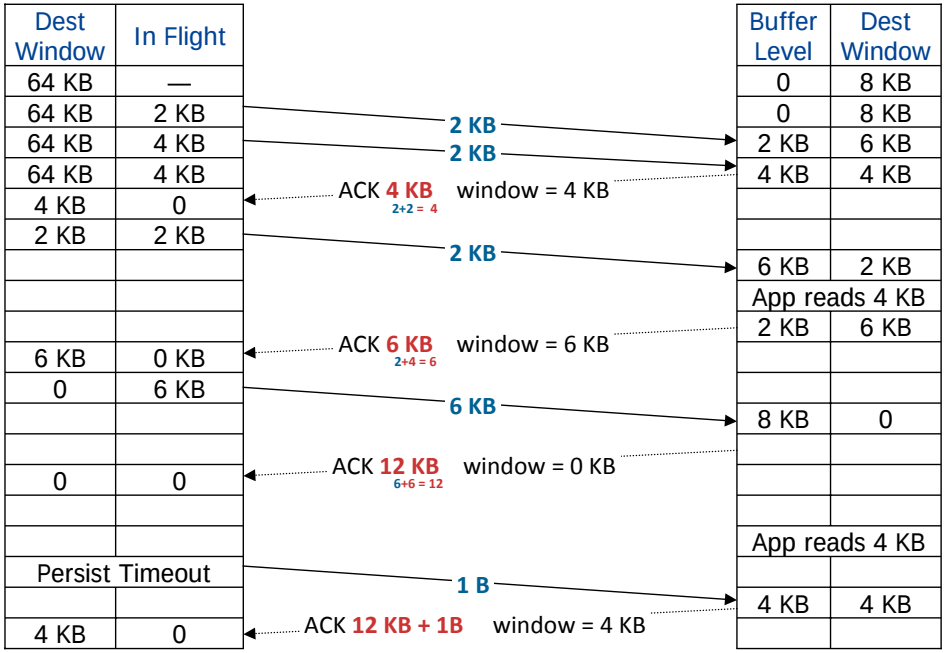
Windows field in TCP header

Number of bytes receiver will accept

Receiver discards bytes above window size



Flow Control Example



Receive Window Bugs — 1

Bug — deadlock

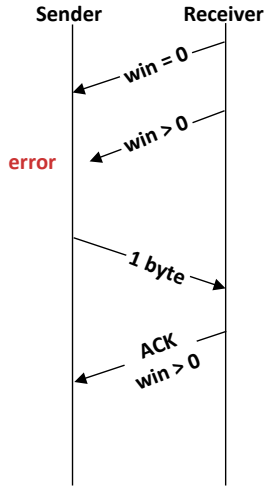
Receiver advertises window = 0

Window update with window > 0 is lost → deadlock

Fix — persist timeout

Sender attempts small segment

ACK contains new window size



Receive Window Bugs — 2

Silly Window Problem

Application reads received data slowly

Receiver advertises small window

Data bytes ~ header bytes

More segments / file transfer ⇒ larger total traffic (data + headers)

Nagle Algorithm — bug fix for Silly Window

Sender accumulates application data — sends large segments

Works badly with Telnet (requires small segments)

Receiver side bug fix

Receiver keeps 0 window size until it can advertise large window

Congestion Control

Queuing theory

Assumptions

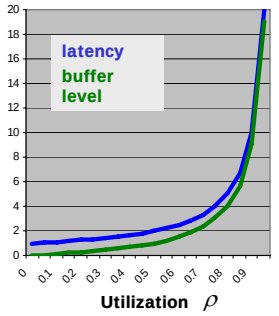
- Segments arrive independently (Poisson statistics)
  - Random length (bytes)
  - Average arrival rate in steady state
- Segments leave independently (Poisson statistics)
  - Average emptying rate in steady state

Results

$$\rho = \text{Utilization} = \frac{\text{arrival rate}}{\text{empty rate}}$$

$$\text{Latency} = \frac{1}{\text{empty rate} - \text{arrival rate}} = \frac{1}{\text{empty rate}}\left(\frac{1}{1-\rho}\right)$$

$$\text{Buffer Level} = \text{Latency} \times \text{arrival rate} = \frac{\rho}{1-\rho}$$



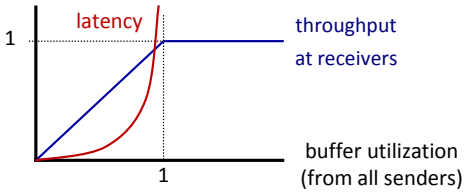
Congestion Control

Buffer throughput

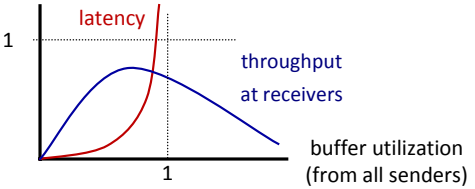
(Over)-simplified throughput model

$$\text{throughtput} = \frac{\text{receive rate}}{\text{maximum receive rate}}$$

$$\text{buffer utilization} = \frac{\text{arrival rate}}{\text{empty rate}}$$



Realistic throughput behavior



- High arrival rate at buffer
- Longer latency + overflow
- Sender timeouts
- Re-transmit ⇒ more segments ⇒ higher arrival rate at buffer

TCP Congestion Control

End-to-end congestion control

- Based on host estimates
- No feedback from intermediate network nodes

Slow-start

- Begin session with low transmission rate
- Increase rate until timeouts begin

Fast retransmit

- Do not wait for timeout
- Re-transmit after duplicate ACKs (dupACKs)

Congestion avoidance

- Limit transmission rate after duplicate ACKs
- Transmission rate → initial slow-start rate

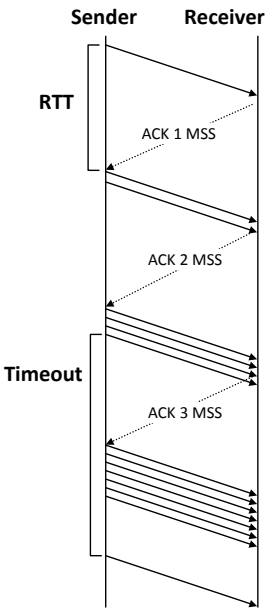
Fast recovery

- Congestion avoidance with larger transmission rate

Slow-Start

Congestion window (cwnd)

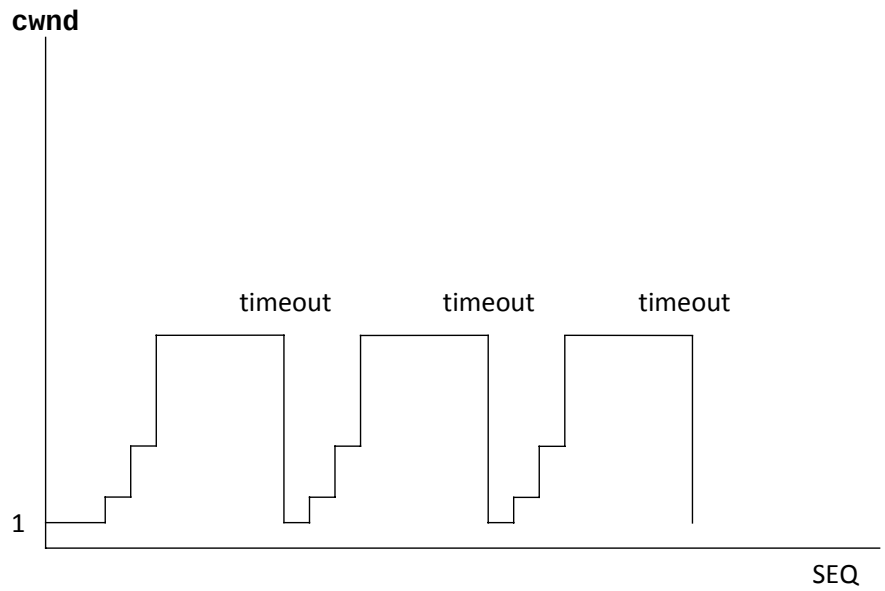
- Source window
  - Maximum number of "unACKed" bytes
- Initial **cwnd** = 1 MSS (maximum segment size)
- Data rate = 1 MSS / RTT
- Maximum **cwnd** = destination window



Exponential growth

```
On (ACK)
    cwnd ← cwnd + size of data ACKed
    if (cwnd > maximum cwnd)
        cwnd ← max cwnd
On (ACK timeout)
    cwnd ← initial cwnd = 1 MSS
```

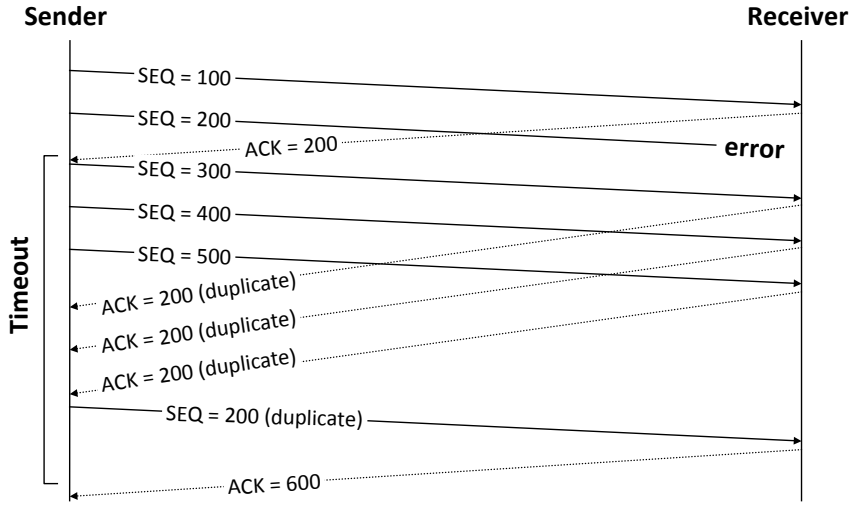
Behavior of **cwnd** in Slow Start



Fast Retransmit

Better performance with RTO >> RTT

3 duplicate ACKs (dupACKs) for segment ⇒ re-send segment



Congestion Avoidance

Tahoe protocol

Slow start threshold

ssthresh ← large initial value (possibly maximum **cwnd**)

Slow start phase

```
On (ACK && cwnd < ssthresh)
   cwnd ← cwnd + size of data ACKed
```

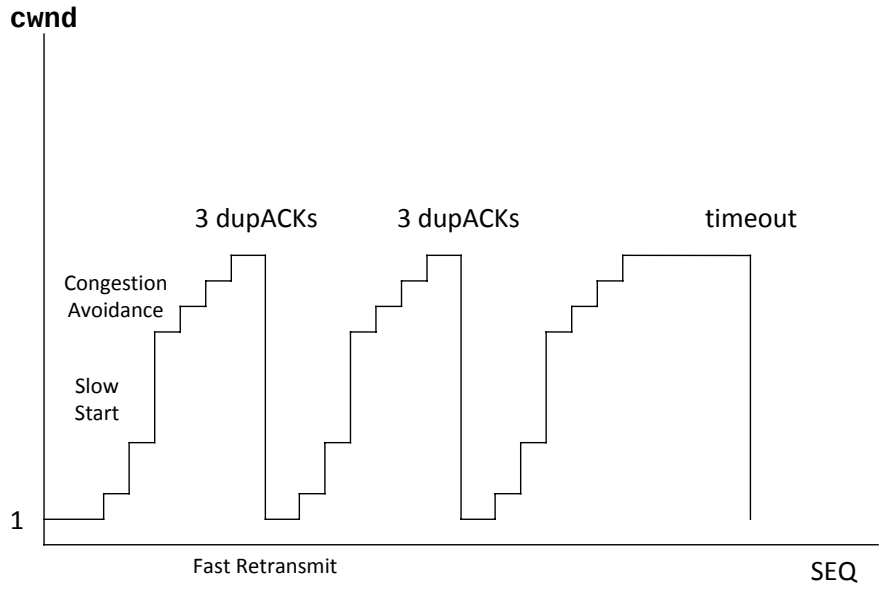
Congestion avoidance phase

```
On (ACK && cwnd > ssthresh)
   cwnd ← cwnd + 1 MSS
```
(exponential → linear growth)

Fast retransmit

```
On (ACK timeout || 3 dupACKs)
   ssthresh ← cwnd
```
(pre-timeout value)
```
   cwnd ← initial cwnd = 1 MSS
```


Behavior of **cwnd** in Tahoe



Congestion Avoidance

Reno protocol

Slow start phase

```
On (ACK && cwnd < ssthresh)
    cwnd ← cwnd + size of data ACKed
On (ACK timeout)
    ssthresh ← cwnd
    cwnd ← initial cwnd = 1 MSS
    RTO ← 2 * RTO
```

Congestion avoidance phase

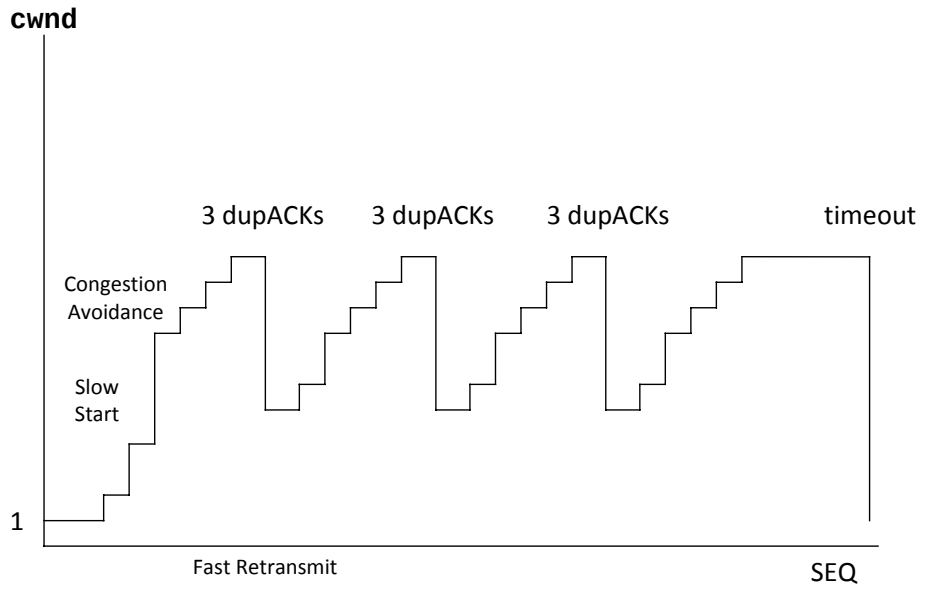
```
On (ACK && cwnd > ssthresh)
    cwnd ← cwnd + 1 MSS
```

Fast retransmit with fast recovery

```
On (3 dupACKs)
    ssthresh ← cwnd
    cwnd ← cwnd / 2
```

Retransmit lost packet
Wait 1 RTT → continue sending
For > 3 dupACKs
cwnd++ on each new dupACK

Behavior of **cwnd** in Reno



TCP Sender with Reno — 1

```
// initialize
    SEQ = ISN + 1
    SendBase = ISN + 1
    InFlight = 0
    cwnd = 1 MSS
    Set ssthreshold large (local policy)
    RTO = timeout

// main loop
on (new data from application)
    Prepare data segment:sequence number = SEQ
    if InFlight < min{cwnd,SendWindow,RecvWindow)
        Pass segment to IP
        SEQ = SEQ + length(data)
        InFlight = InFlight + length(data)
        if !(timer running) timer = RTO
```


TCP Sender with Reno — 2

```
if (receive ACK = y)
    stop timer
    if (y > SendBase)
        dupACK = 0
        newACKs = y – SendBase            // bytes ACKed
        SendBase = y
        InFlight = InFlight – newACKs
        if (cwnd < ssthresh) cwnd = cwnd + newACKs
        else cwnd = cwnd + 1 MSS
        if (InFlight > 0) timer = RTO
```

TCP Sender with Reno — 3

```
    // if (y > SendBase)
    else
        dupACK++
        if (dupACK = 3)
            SEQ = SendBase = min{unACKed SEQ} and resend
            timer = RTO
            ssthresh = cwnd
            cwnd = cwnd / 2
            wait 1 RTT   // wait for ACK of resent packet
        if (dupACK > 3)
            cwnd = cwnd + 1 MSS
if (timeout)
        SEQ = SendBase = min{unACKed SEQ} and resend
        ssthresh = cwnd
        cwnd = initial cwnd = 1 MSS
        RTO = 2 * RTO
        timer = RTO
```


TCP Receiver with Reno — 1

```
// initialize
    Set RecvWindow = receiver buffer size
    expected = Sender ISN + 1
    ack_buffer = 0
    ack_max (local policy: delayed ACK trigger)
    ack_delay = 250 msec (local policy: < 500 msec)
    Start ACK delay timer = ack_delay

// main loop
if (ACK delay timer = 0 && ack_buffer > 0)
    Send ACK = expected with updated RecvWindow
    ACK delay timer = ack_delay
    ack_buffer = 0
```


TCP Receiver with Reno — 2

```
if (receive SEQ = x)
    if (x = expected && error-free)
        expected = expected + length(data)
        if (NACK = 1)
            Send ACK = expected with updated RecvWindow
            ACK delay timer = ack_delay
            ack_buffer = 0
            NACK = 0
        else if (ack_buffer < ack_max)
            nextACK = expected
            ack_buffer++
        else if (ack_buffer = ack_max)
            Send ACK = expected with updated RecvWindow
            ACK delay timer = ack_delay
            ack_buffer = 0
    else
        Send ACK = expected with updated RecvWindow
        ACK delay timer = ack_delay
        NACK = 1
```

Network Layer

Transport / Network Interface

Transport layer requests service from network layer

Transport Layer	Prepare Network service requirements Transport segment for transfer across network Application data + transport header Local node address Remote node address
Network Layer	Receives Transport layer PDU — data + transport header Interface Control Information (ICI) Service requirements Local + remote node addresses

Connection Types

Connection

State machine associated with data exchange

Connection-oriented

Set-up end-to-end network channel before data any exchange
Monitor channel state during data exchange
Multiple transactions associated with connection state
Close channel after data exchange
Example — **phone call**
Enter number → answer call → extended conversation → disconnect

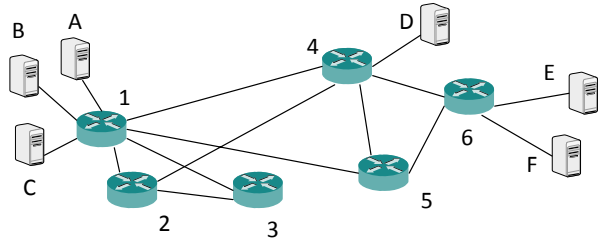
Connectionless

Data transmitted with no prior channel set-up
No channel state defined by nodes
Each message independent
Example — **email message**
Send email → hope message arrives → hope message is found / read

Datagram Service

Network of routers and links

Packet switching
Connectionless



Every datagram

Has source and destination address in header
Data Link header or Network header
Routed individually through network
Datagrams may follow separate routes
Best effort delivery — unreliable service at layer 3
Example

src = B dest = F	data

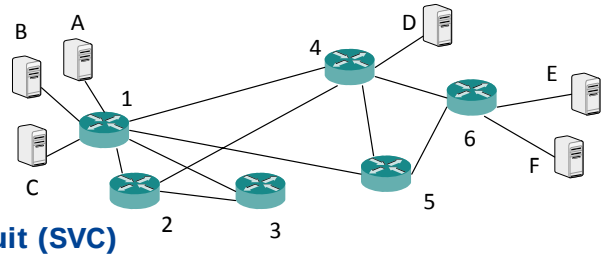
B → 1 → 4 → 6 → F
B → 1 → 5 → 6 → F

Switched Virtual Circuit (SVC)

Network of switches and links

Circuit switching or packet switching

Connection-oriented



Switched Virtual Circuit (SVC)

Set-up / close messages carry source and destination addresses

Example

Set-up VC–1: B → 1 → 4 → 6 → F

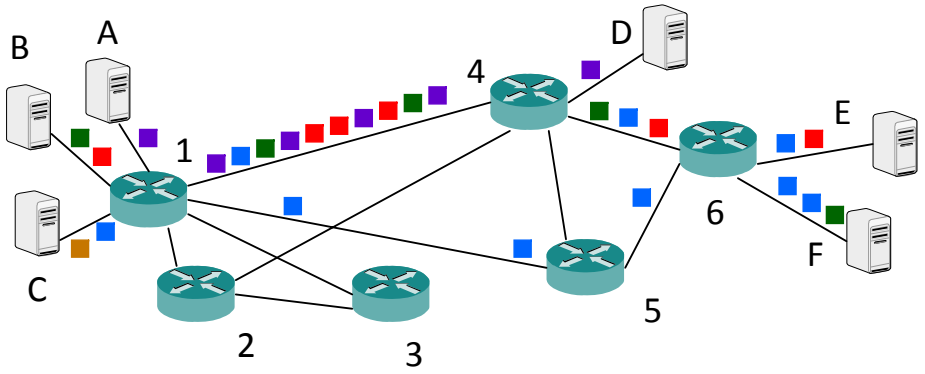
Packet routing by VC ID in header (layer 2 or layer 3)

Every packet follows same VC route

Example

VC – 1	data

Switching Example



A to D — circuit mode (deterministic SVC)

B to E — packet mode (statistical SVC)

B to F — packet mode (statistical SVC)

C to F — packet mode (datagram service)

Virtual Circuit Labeling

Routing

VC number assigned at each switch

Source-to-destination label on one hop

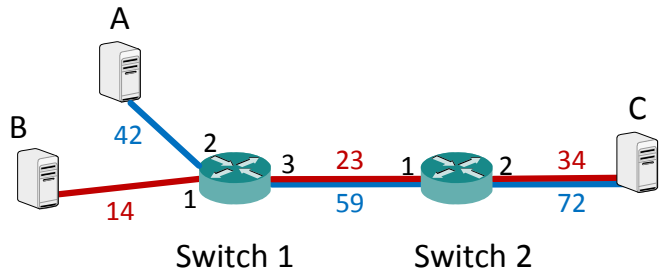
Can change at each switch

VC in packet header identifies path

Example

X.25 network packet

Format	VC	Type	Data



Virtual Circuit Routing Table

Routing pair

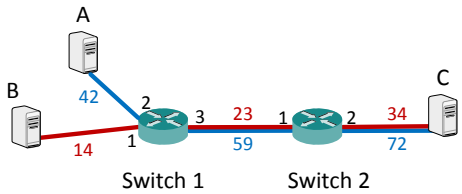
Interface number

VC

Routing table for VC

VC set-up = write table entries

Map input pair → output pair



Switch 1 Routing Table			
Input		Output	
Interface	VC	Interface	VC
1	14	3	23
2	42	3	59
3	23	1	14
3	59	2	42

Switch 2 Routing Table			
Input		Output	
Interface	VC	Interface	VC
1	23	2	34
1	59	2	72
2	34	1	23
2	72	1	59

Datagram Forwarding

Routing

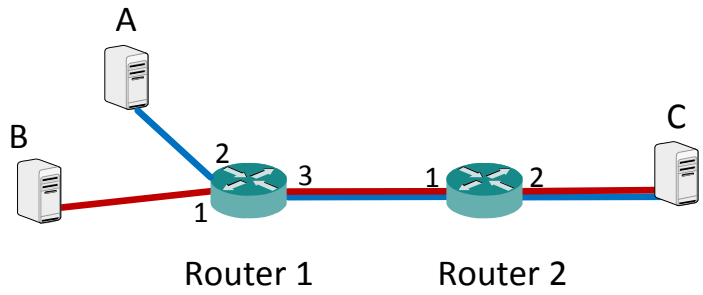
Source address + destination address in IP header

Destination pair identifies next hop

Example

IP

Src	Dest	Other Fields	Data

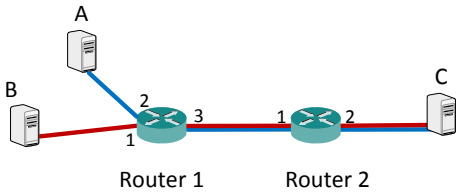


Datagram Forwarding

Routing table for datagram

Routing algorithm → table entries

Map destination address → output interface



Router 1 Table	
Destination Address Range	**Interface**
A	2
B	1
Other	3

Router 2 Table	
Destination Address Range	**Interface**
C	2
Other	1

Routing algorithm

1 tells 2 "I have A + B"

2 tells 1 "I have C"

Connection-Oriented / Connectionless Networks

Connection-oriented network services

ESS#7	64 kbps pubic switched telephone network (PSTN)
X.25	64 kbps pubic switched data network (PSDN)
Frame relay	2.048 – 45 Mbps replacement for X.25
ATM	64 kbps – 625 Mbps replacement for ESS#7 + Frame Relay
GSM / UMTS	Cellular voice network
GPRS / EDGE	384 kbps – 1.9 Mbps cellular data network

Connectionless network service

Internet Protocol (IP)

Internet Protocol version 4 (IPv4)

IP datagram format

4 bits	4 bits	8 bits	16 bits
Version	Hlen	Service Type	Total Length (header + data in bytes)
Identification		Flags	Fragment Offset (13 bits)
Time to Live		Protocol	Header Checksum
Source IP Address			
Destination IP Address			
Options			
Data			

Hlen	Header length in 32-bit fields
Service type	Differentiated Services Code Point (DSCP) Explicit Congestion Notification (ECN)
Identification	Provides unique datagram ID
Flags	**0** **DF** (Don't Fragment) **MF** (More Fragments — all frags but last)
Fragment Offset	Offset in 8-byte units from start of original datagram (fragmented)
Time To Live (TTL)	Routers perform: `{if (--TTL == 0) delete datagram}`
Protocol	Protocol of data carried by datagram

Maximum Transmission Unit (MTU)

Largest datagram accepted by Layer 2 links

Fragmentation

Divide datagram longer than MTU into < 2^{13} = 8192 datagrams

If DF = 1 datagram discarded

Reassembly

Original datagram reassembled at final destination

Example

MTU = 1500 bytes = 20 (header) + 1480 (data)

Total Length = 4000 bytes = header + 3980

3980 – 2 * 1480 = 1020

3 fragments with original header except

1 — Length = 1500 bytes MF = 1 offset = 0

2 — Length = 1500 bytes MF = 1 offset = 1480 / 8 = 185

3 — Length = 1040 bytes MF = 0 offset = 2*1480 / 8 = 370

Internet address

32-bit address

4 octets (8-bit decimal numbers) separated by dots

Example

www.hadassah.ac.il ↔ 212.179.79.228

Network / Host

Host address = 212.179.79.228

Host address = Network.Host

Network number = 212.179.79.0

Host number = 228

Network / Subnet / Host

Divide hosts into sub-networks

$$\text{Host number} = \begin{cases} 0, \ldots, A-1 & \text{(subnet 0)} \\ A, \ldots, 2A-1 & \text{(subnet 1)} \\ 2A, \ldots, 3A-1 & \text{(subnet 2)} \\ \ldots & \end{cases}$$

Example

Subnet 0 = 212.179.79.1 – 212.179.79.127

Host address range = 212.179.79.X

X = 0xxxxxxx (binary)

Subnet 1 = 212.179.79.128 – 212.179.79.254

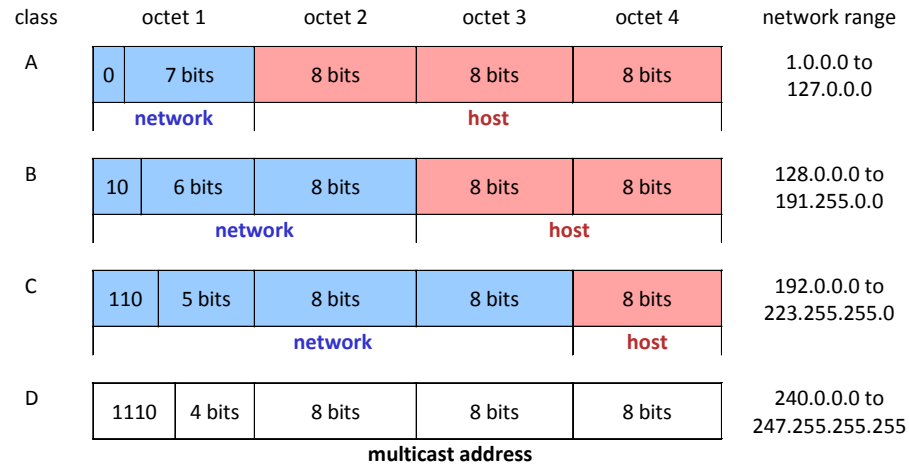
Host address range = 212.179.79.X

X = 1xxxxxxx (binary)

3 main address classes

A — small number of large networks (up to 2^{24} = 16 Mhosts)

C — large number of small networks (up to 2^{8} = 256 hosts)



Internet Protocol version 4 (IPv4)

Classless Inter-Domain Routing (CIDR)

Address format

`Octet1.Octet2.Octet3.Octet4/bits_in_network_number`

`bits_in_network_number` = 1, 2, 3, ... , 31 (unrestricted)

Examples

Class A address — **`10.0.1.5/8`**

8-bit network number = **`10.0.0.0`**

Class C address — **`192.168.0.37/24`**

24-bit network number = **`192.168.0.0`**

General node address — **`192.168.0.33/27`**

192	168	0	33	
11000000	10101000	00000000	001	00001
27-bit network address				host

Network address = **`192.168.0.32`** Host number = **`0 – 31`**

Host addresses = **`192.168.0.32 – 192.168.0.63`**

Internet Protocol version 4 (IPv4)

Subnet mask

Forming subnet mask

`1` in all bits of network number

`0` in all bits of host number

Encodes same information as number of bits in network number

Example

General node address — **`192.168.0.33/27`**

27-bit network number

Mask

`11111111.11111111.11111111.11100000`

`255.255.255.224`

Using mask

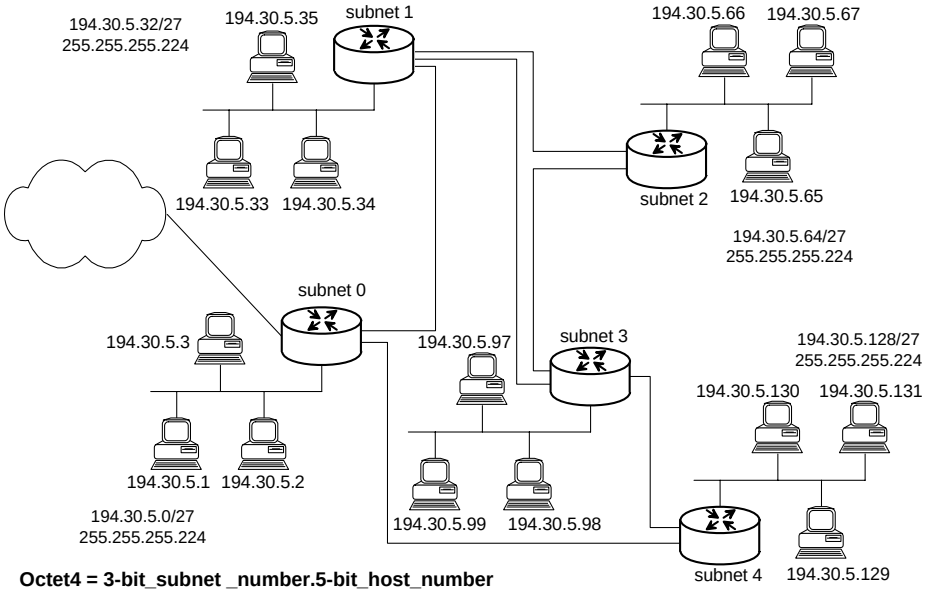
Mask AND IP address = network number

Example

`255.255.255.224 AND 192.168.0.33 = 192.168.0.32`

Internet Protocol version 4 (IPv4)

Subnet example



Octet4 = 3-bit_subnet _number.5-bit_host_number

Internet Protocol version 4 (IPv4)

Reserved addresses

CIDR address block	Description
0.0.0.0/8	Current network (source address)
10.0.0.0/8	Private network
127.0.0.0/8	Loopback (destination = this node)
172.16.0.0/12	Private network
192.168.0.0/16	Private network
224.0.0.0/4	Multicast (Class D)
240.0.0.0/4	Reserved

Broadcast on CIDR network

IP address = **`octet1.octet2.octet3.octet4/nw_bits`**

MASK	11...1	00...0
	nw_bits	32 – nw_bits

MASK'	00...0	11...1
	nw_bits	32 – nw_bits

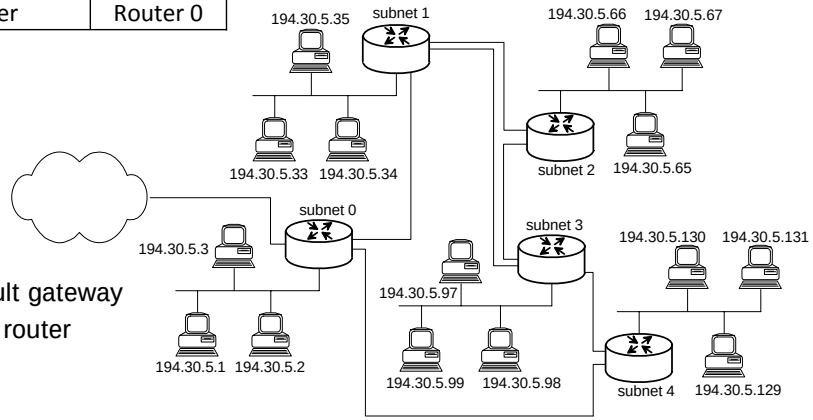
Network number = MASK AND IP

Broadcast = MASK' OR IP

CIDR Routing Table

Router 1 Table	
Destination	**Interface**
194.30.5.32/27	LAN
194.30.5.64/27	Router 2
194.30.5.96/27	Router 3
194.30.5.128/27	Router 3
Other	Router 0

Router 4 Table	
Destination	**Interface**
194.30.5.128/27	LAN
194.30.5.64/27	Router 3
194.30.5.96/27	Router 3
Other	Router 0



Router 0

Default gateway

Edge router

Routing Algorithms

Application of graph theory

Model network as undirected graph G(N,E)

- Nodes N = switches / routers
- Edges E = one-hop links between nodes
- Weight = metric (cost, distance) > 0
  - Weighted average of path parameters
  - Path length, queuing and transmission delay, cost, etc

Shortest path routing by standard algorithms

Static routing

Apply Dijkstra algorithm

Link State Routing

Requires knowledge of all weights

Dynamic routing

Solve Bellman-Ford equation locally

Nearest neighbor nodes exchange weight parameters

Distance Vector Routing

Router Graph

Nodes

N = {u, v, w, x, y, z}

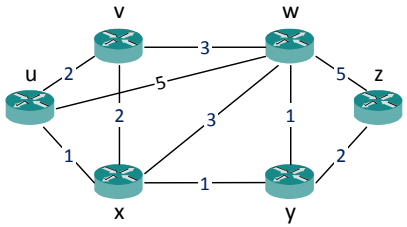
Weights

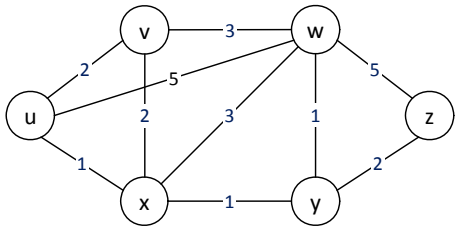
c(i,j) = link metric from i to j

Optimal path

Path from x to y with minimum cost

$d_x(y)$ = metric of optimal path from x to y

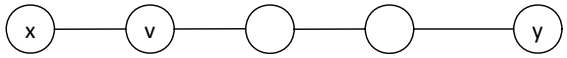




Bellman-Ford Equation

Principle of optimality

Node v on optimal path from x to y ⇒ xv and vy = optimal paths



Corollary

$$xvy \text{ optimal} \Rightarrow d_x(y) = c(x,v) + d_v(y) \Rightarrow c(x,v) = d_x(v)$$

Bellman-Ford equation

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

Basis for static and dynamic routing algorithms

Dijkstra Algorithm

Initialize

$$N' = \{u\} \qquad D(v) = \begin{cases} c(u,v), & uv \text{ is a link} \\ \infty, & \text{otherwise} \end{cases} \qquad p(v) = \begin{cases} u, & uv \text{ is a link} \\ 0, & \text{otherwise} \end{cases}$$

While $(N \backslash N' \neq \phi)$

find $w \in N \backslash N'$ with minimum $D(w)$

move w from N to N'

for $(v \in N \backslash N')$ {

$L = D(w) + c(v,w)$

if $(L < D(v))$ {

$D(v) = L$

$p(v) = w$

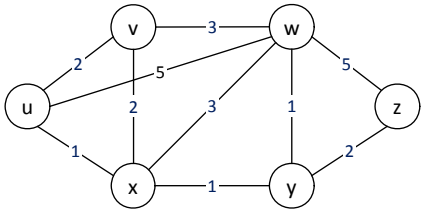
}

}

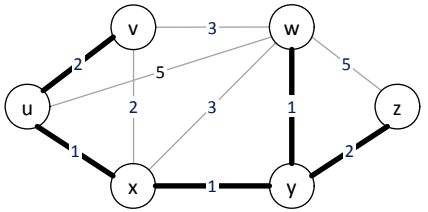
Dijsktra Example

Source u

N = {u, v, w, x, y, z}



N	N'	Minimum D	D{v, w, x, y, z}	p{v, w, x, y, z}
{v, w, x, y, z}	{u}		{2, 5, 1, ∞, ∞}	{u, u, u, 0, 0}
{v, w, y, z}	{u, x}	D(x) = 1	{2, 4, 1, 2, ∞}	{u, x, u, x, 0}
{w, y, z}	{u, v, x}	D(v) = 2	{2, 4, 1, 2, ∞}	{u, x, u, x, 0}
{w, z}	{u, v, x, y}	D(y) = 2	{2, 3, 1, 2, 4}	{u, y, u, x, y}
{z}	{w, u, v, x, y}	D(w) = 3	{2, 3, 1, 2, 4}	{u, y, u, x, y}



Distance Vector Routing

Routing table in each node

Index = known nodes

Entry = metric to node

Information exchange

Nodes pass routing tables to one-hop neighbors

Table entries

One-hop node

Metric = measured cost parameters

Multi-hop nodes

Metric = min{measured cost to neighbor + neighbor entry to node}

Bellman-Ford routing

Assume route information from node v to node y is $d_v(y)$

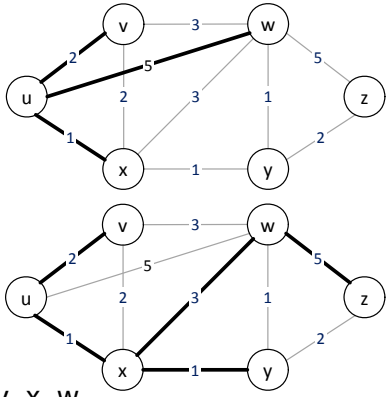
At node x choose route to y via v where

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

DV Example — Exchange 1

Initial tables

	source					
dest	u	v	w	x	y	z
u	—	2	5	1	—	—
v	2	—	3	2	—	—
w	5	3	—	3	1	5
x	1	2	3	—	1	—
y	—	—	1	1	—	2
z	—	—	5	—	2	—



Exchange 1

Node u gets routing information from v, x, w

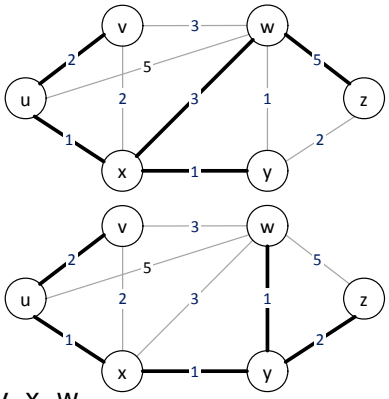
	Metric	Next hop
u	—	—
v	2	v
w	**4**	x
x	1	x
y	**2**	x
z	**9**	w

v	w	x
2	5	1
—	3	2
3	—	3
2	3	—
—	1	1
—	5	—

DV Example — Exchange 2

After Exchange 1 at v, x, w

	v	w	x
u	2,v	4,x	1,u
v	—	3,v	2,v
w	3,w	—	2,y
x	2,x	2,y	—
y	**3,x**	1,y	1
z	**8,w**	3,y	3,y



Exchange 2

Node u gets routing information from v, x, w

	Metric	Next hop
u	—	—
v	2	v
w	**3**	x
x	1	x
y	2	x
z	**4**	x

v	w	x
2	**4**	1
—	3	2
3	—	**2**
2	**2**	—
3	1	1
8	**3**	**3**

Spanning Tree

Spanning tree

Subgraph of G(N,E') $\subseteq$ G(N,E)

Includes every node

Unique path from root node u to any node

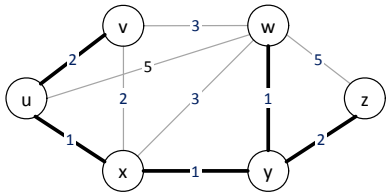
No cycles (loops)

Dijkstra and DV routing

Converge to spanning tree for simple graphs

Minimal spanning tree

Minimal cost from root to any node



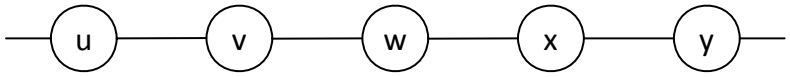
Distance Vector Metric Updates

Good news travels fast

Metric

Hops to node u

Link uv recovers from failure



	Routing Table Entries for Node u			
exchange	v	w	x	y
0	**1,u**	∞	∞	∞
1	1,u	**2,v**	∞	∞
2	1,u	2,v	**3,w**	∞
3	1,u	2,v	3,w	**4,x**

Distance Vector Metric Updates

Bad news travels slowly

Link uv fails

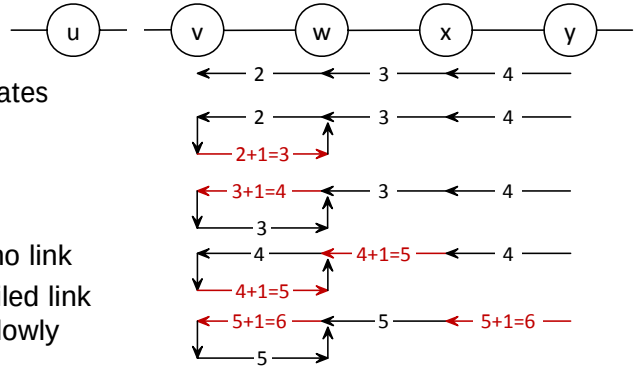
Circular table updates

Count to infinity

N_{max} hops

Metric > N_{max} ⇒ no link

Information on failed link propagates slowly



	Routing Table Entries for Node u			
exchange	v	w	x	y
0	1,u	2,v	3,w	4,x
1	**3,w**	2,v	3,w	4,x
2	3,w	**4,v**	3,w	4,x
3	**5,w**	4,v	**5,w**	4,x
4	5,w	**6,v**	5,w	**6,x**

Internet Routing

Autonomous System (AS)

Nodes managed by one organization

Hierarchical routing

Interior Gateway Protocols (IGP)

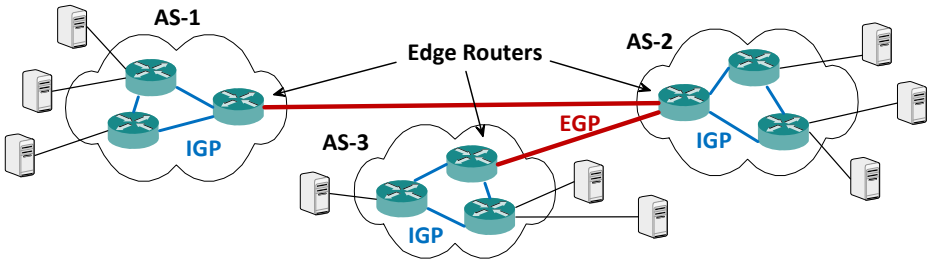
Routing protocols within one AS (Intra-AS)

Exterior Gateway Protocols (EGP)

Routing protocols between ASs (Inter-AS)

Edge router (gateway router)

Router within AS linked to router in different AS



Internal Gateway Protocols (IGP)

Intra-AS tasks

Optimize internal routing

Exchange routing tables with IGP nodes

Link to primary gateway (edge router)

IGP protocols

Routing Information Protocol (RIP)

RFC 1058, RFC 2453

Open Shortest Path First (OSPF)

RFC 2328

Interior Gateway Routing Protocol (IGRP)

Cisco proprietary

Routing Information Protocol (RIP)

Older protocol

Introduced in BSD-UNIX (1982)

Distance Vector (DV) algorithm

Routing information sent as data in UDP segments

DV advertisements

Routing information exchanged with 1-hop neighbor nodes

Exchanged every 30 seconds

No routing update in 180 seconds ⇒ node = unreachable

Metric = hops

Table entry = 1 hop to neighbor + neighbor table entry

N_{max} = 15

$N > N_{max}$ ⇒ node = unreachable

Open Shortest Path First (OSPF)

Newer protocol (1998)

Link State algorithm

Independent network layer protocol (not data over UDP)

LS advertisements (LSAs)

Start

Routing metrics exchanged among 1-hop neighbors

Continue

Routing tables synchronized among all routers in AS

Shortest Path First (SPF)

Dijkstra algorithm among routers in AS

Sub-optimal next hop permitted for load balancing

Metrics

Hops, speed, delays, connections from destination

Multiple service-dependent metrics permitted

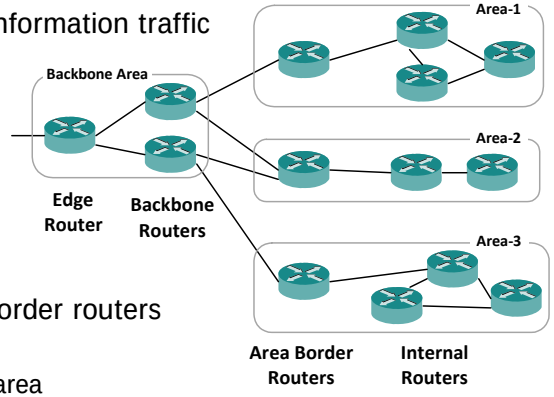
Hierarchical Open Shortest Path First (OSPF)

Divide AS into hierarchy of areas

- Routers in area perform OSPF among themselves
- Reduces overall routing information traffic

Area hierarchy

- Edge router
- Backbone
- Area border routers
- Internal area routers



LS advertisements (LSAs)

- Internal routers + area border routers
  - Perform full OSPF
  - Optimal routing within area
- Area border routers + backbone
  - Exchange summary metrics
  - Basic topology
- Backbone + edge perform OSPF as an area

Exterior Gateway Protocols (EGP)

Inter-AS tasks

- Exchange routing information among EGP nodes
- Determine routes for inter-AS traffic
- Manage economic, political, security considerations
- Avoid global instabilities (routing cycles)

EGP protocols

- Exterior Gateway Protocol (EGP)
  - Obsolete
  - RFC 827, RFC 904
- Border Gateway Protocol (BGP)
  - RFC 4271

Border Gateway Protocol (BGP)

Routing hierarchy

eBGP

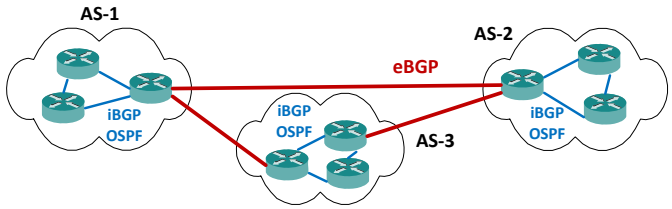
- Exchange routing information between AS edge routers
- Determine routes for inter-AS traffic

iBGP

- Exchange routing information within AS
- Determine preferred gateway for inter-AS traffic

OSPF (or other IGP)

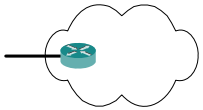
- Exchange routing information within AS
- Determine routes for intra-AS traffic



Border Gateway Protocol (BGP)

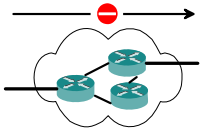
AS classification

Stub network



- 1 edge router
- Cannot carry transit traffic
  - All incoming / outgoing traffic has dest / src in AS

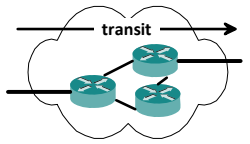
Multi-homed network



- More than 1 edge router
- Refuses transit traffic
  - All incoming / outgoing traffic has dest / src in AS
- Typically large corporate network or commercial ISP
  - Unwilling to accept traffic burden

Transit network

- More than 1 edge router
- Accepts transit traffic
- Internet backbone



AS reachable via

- One hop link
- Transit network

Border Gateway Protocol (BGP)

Routing information exchange

BGP speaker	Router supporting BGP
BGP peers	Two BGP speakers exchanging routing information
BGP session	Persistent TCP session between two BGP peers Used to exchange routing information
AS number (ASN)	16-bit number to identify AS
Network Layer Reachability Information (NLRI)	CIDR description of reachable Internet destination AS
AS route	NLRI with network attributes AS path List of ASNs describing path to NLRI Next-Hop IP address of next hop edge router

Border Gateway Protocol (BGP)

Path selection

No routing cycles

Router eliminates path that includes itself as transit network

Local preference attribute

Administrator policy sets preference value for paths

Lower preference paths eliminated

Shortest AS path

Path with least transit networks

Paths with longer AS path eliminated

Closest Next-Hop router

Possibly multiple routes to next AS

Determine minimum cost link to router in next AS

Routes with higher cost route to next hop router eliminated

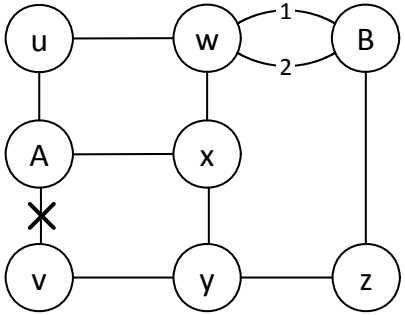
Border Gateway Protocol (BGP)

No routing cycles

Path from A to B

A receives paths from peers

Peer	Peer Recommends
u	uwB
x	xwB
v	vAuwB



No loops

A eliminates path AvAuwB

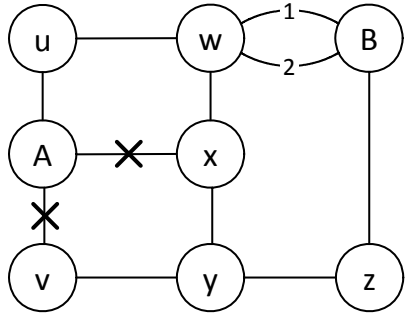
Border Gateway Protocol (BGP)

Local preference attribute

Path from A to B

A receives paths from peers

Peer	Peer Recommends	Local Preference
u	uwB	1
x	xwB	2
v	vAuwB	2



No loops

A eliminates path AvAuwB

Local administrator sets preferences

AS u preferred to AS x as transit network

Example — data security questions

A eliminates path AxwB

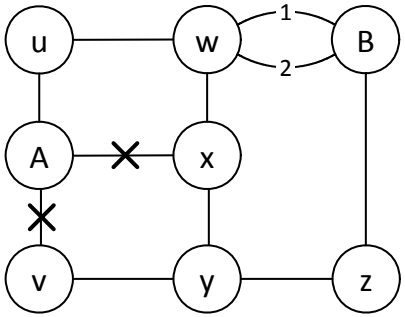
Border Gateway Protocol (BGP)

Shortest AS path

Path from A to B

A receives update from v

Peer	Peer Recommends	Local Preference
u	uwB	1
x	xwB	2
v	vyzB	2



Local administrator sets preferences

AS u preferred to AS x as transit network

Example — data security questions

A eliminates path AxwB

Shortest AS path

AvyzB longer than AuwB

A eliminates AvyzB

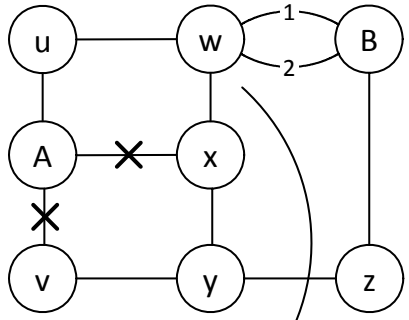
Border Gateway Protocol (BGP)

Closest Next-Hop router

Path from A to B

A receives update from u

Peer	Peer Recommends	Local Preference
u	uwB	1
x	xwB	2
v	vyzB	2



Local administrator sets preferences

AS u preferred to AS x as transit network

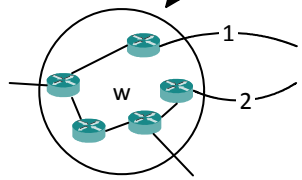
Example — data security questions

A eliminates path AxwB

Shortest AS path

AvyzB longer than AuwB

A eliminates AvyzB



Path 1 closer than path 2

Border Gateway Protocol (BGP)

Refusing transit traffic

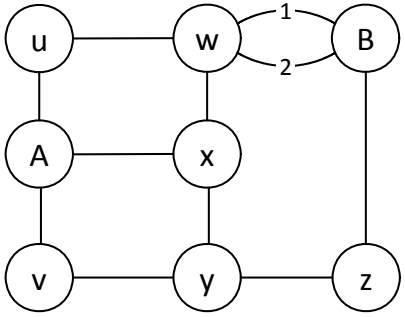
Transit policy

Commercial ISP A

- Revenue from attached customers
- Customers require forwarding

Transit traffic

- Adds load to ISP network
- Produces no revenue



A does not advertise

Connection Ax to u or v

Connection Av to u or x

Connection Au to x or v

Broadcast / Multicast Services

Broadcast

Forward datagrams to all nodes

Multicast

Forward datagrams to group of nodes

Local broadcast / multicast

Among nodes attached to one router

Server advertisements

- DHCP
- DNS
- Router services

Handled as layer 3 / layer 2 transactions

Global broadcast / multicast

Among multiple routers

File sharing

Content distribution

Media streaming

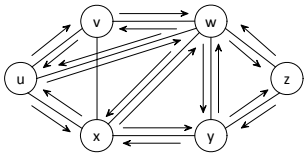
Broadcast Issues

Node-level broadcast

- Multiple unicast
- Requires list of all nodes
- Not practical

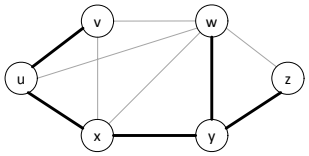
Uncontrolled flooding

- Each router forwards all packets to all routers
- Broadcast storm
  - Traffic multiplies exponentially
  - Network paralyzed



Controlled flooding methods

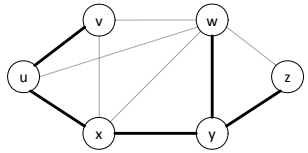
- Tag packets
  - Transmit each packet one time
- Time to Live (TTL)
  - Delete packets after N_max hops
- Broadcast only on spanning tree
  - No cycles in graph



Spanning Tree Broadcast

Reverse Path Forwarding (RPF)

- Only forward packet arriving on default node-to-source path
- Node uses default routing table
  - Does not know full spanning tree
- Example
  - Node broadcast u arrives at node x
    - Arrives on u-x
      - Node x forwards to y on x-y
    - Arrives on y-x, w-x, v-x
      - Node x deletes



Node x Routing Table	
u	u
v	
w	y
y	
z	

Center-based forwarding

- Designated center node
- Nodes send join messages to center node by conventional routing
- Center node broadcasts to group members

Multicast

Multicast group

- Subset of all routers / hosts receiving multicast datagrams

Multicast router

- Maintains group membership information for multicast distribution

Multicast address

- Class D address refers to multicast group
- Multicast datagrams addressed to multicast address
- Multicast routers distribute to group members

Internet Group Management Protocol (RFC 3376)

- Nodes report multicast group memberships to multicast routers

Protocol Independent Multicast (PIM)

- Dense mode — multicast routers closely distributed
  - RPF spanning tree construction
- Sparse mode — multicast routers widely distributed
  - Center-based forwarding

Network Address Translation (NAT)

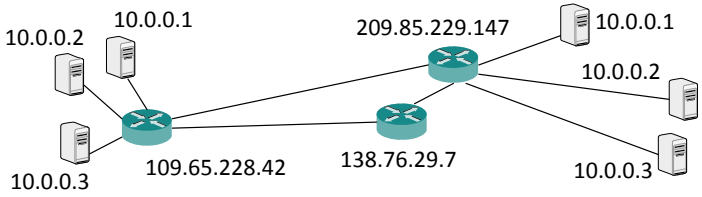
Router

- Receives IP datagram
- Exchanges IP source / destination address in header
- Forwards datagram
- Maintains list of translations

Autonomous systems

- Allocate private network addresses internally
  - 10.0.0.0 and 192.168.0.0
- Only gateway and backbone devices require unique IP addresses
  - Local nodes on AS re-use private addresses

Example



Network Address Translation (NAT)

Translation details

No translation required on traffic internal to AS

- Local addresses = real IP addresses

Local node to external node

- Outgoing packet
  - Source endpoint = local IP address + application source port number
  - Dest endpoint = remote IP address + well-known port number
- Gateway router NAT
  - Replaces
    - Local IP address with gateway IP address
    - Local application source port with unique unused NAT port
  - Records mapping
    - NAT source port → local IP address + source port number
  - Forwards packet

External node to local node

- Gateway router replaces NAT port → local IP + app source port

Network Address Translation (NAT)

Example

Host A HTTP request to Server S

- Source port = **1025** Source IP = **10.0.0.2**
- Destination port = 80 Destination IP = 209.85.229.147

Outgoing NAT at Gateway router B

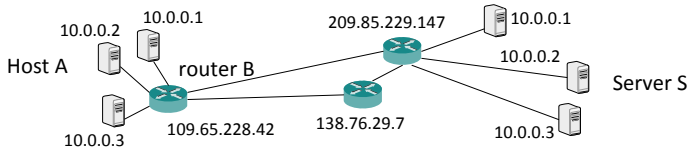
- Source port = **3745** Source IP = **109.65.228.42**
- Destination port = 80 Destination IP = 209.85.229.147

Server S HTTP response to Host A

- Source port = 80 Source IP = 209.85.229.147
- Destination port = **3745** Destination IP = **109.65.228.42**

Incoming NAT at Gateway router B

- Source port = 80 Source IP = 209.85.229.147
- Destination port = **1025** Destination IP = **10.0.0.2**



Network Address Translation (NAT)

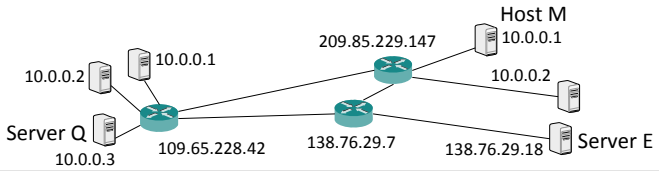
Traversal problem

Problem

- How to operate service behind NAT

Solutions

- Static router referral
  - Define service port on Server Q
  - Refer all incoming traffic at gateway for service port to Server Q
- Relay server
  - Server Q connects to Server E with public address (no NAT)
  - Host M requests service from Server Q via Server E
  - Server Q replies via Server E
- Universal Plug and Play (UPnP)
  - Specialized protocols for NAT traversal



Assistive Protocols used with IP Layer

Domain Name System (DNS)

- Convert between node name and network address

Address Resolution Protocol (ARP)

- Convert between network and hardware addresses
- Discover local subnet topology

Dynamic Host Configuration Protocol (DHCP)

- IP address allocation on request from DHCP server
  - Server typically in gateway router
- Server allocates IP address from pool of available addresses
- DHCP message types
  - DHCP server discovery (host broadcast)
  - DHCP offer (server response)
  - DHCP request (host request)
  - DHCP ACK (server provides address)

IP Configuration Example

```
c:\>ipconfig /all
Windows IP Configuration
        Host Name . . . . . . . . . . . . : ******
        Primary Dns Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Unknown
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : ******
Ethernet adapter Local Area Connection:
        Connection-specific DNS Suffix  . : ******
        Description . . . . . . . . . . . : ******
        Physical Address. . . . . . . . . : ******
        Dhcp Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 10.0.0.5
        Subnet Mask . . . . . . . . . . . : 255.0.0.0
        Default Gateway . . . . . . . . . : 10.0.0.138
        DNS Servers . . . . . . . . . . . : 10.0.0.11
                                            10.0.0.138
c:\>arp -a
Interface: 10.0.0.5 --- 0x2
  Internet Address      Physical Address      Type
  10.0.0.7              70-1a-04-95-a4-4f     dynamic
  10.0.0.11             00-50-fc-b1-6f-ff     dynamic
  10.0.0.138            5c-d9-98-06-26-36     dynamic
```


Internet Control Message Protocol (ICMP)

Control protocol

- Network management information
- Error reporting
  - Unreachable host / network / port / protocol
- Echo request / reply
- Packets carried in IP datagrams

Packet structure

Offset	0 – 7	8 – 15	16 – 23	24 – 31
0	Type	Code (subtype)	Checksum	
32	Rest of Header			
64	Padding			

- Rest of Header — message specific
- Padding — data field

Selected Message Types		
Type	Code	description
0	0	echo reply
3	0	dest network unreachable
	1	dest host unreachable
	2	dest protocol unreachable
	3	dest port unreachable
	6	dest network unknown
	7	dest host unknown
8	0	echo request
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Ping

Client application program

- Client sends echo request packets to destination IP address
  - Type = 8 Code = 0 16-bit ID number 16-bit sequence number
- OS dependent
  - Linux ping
    - Default = continuous packets with 56 padding bytes
  - Windows ping.exe
    - Default = 5 packets with 32 padding bytes

Ping server (OS service)

- Responds to each echo request packet with echo rely packet
  - Type = 0 Code = 0 16-bit ID number 16-bit sequence number

Ping client

- Measures round trip time (RTT)
- Reports packet losses, RTTs and average RTT

UDP ping

- Uses UDP instead of ICMP

Ping Example

```
c:\>ping www.hadassah.ac.il

Pinging hathi.hadassah.ac.il [212.179.79.228] with 32 bytes of data:

Reply from 212.179.79.228: bytes=32 time=32ms TTL=119
Reply from 212.179.79.228: bytes=32 time=70ms TTL=119
Reply from 212.179.79.228: bytes=32 time=135ms TTL=119
Reply from 212.179.79.228: bytes=32 time=83ms TTL=119

Ping statistics for 212.179.79.228:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 32ms, Maximum = 135ms, Average = 80ms
```

Traceroute

Client application program

- Multiple echo request packets to destination IP address
  - First packet TTL = 1
  - Each subsequent packet has TTL++
- OS dependent
  - Linux traceroute / tracepath
    - UDP to unlikely port number (port typically not defined)
  - Windows tracert.exe
    - ICMP echo request packets

Intermediate routers

`if (--TTL == 0){delete ; ICMP type 11 to client}`

- ICMP packet carries router name + IP address
- ICMP sequence = list of intermediate routers (hops = 1, 2, ...)

Destination node

- UDP segment — ICMP port undefined (type 3 code 3) message
- ICMP packet — ICMP echo reply message

Traceroute Example

```
c:\>tracert www.hadassah.ac.il

Tracing route to hathi.hadassah.ac.il [212.179.79.228]
over a maximum of 30 hops:

  1     2 ms     1 ms     2 ms  10.0.0.138
  2    18 ms    19 ms    17 ms  bzq-179-37-1.static.bezeqint.net [212.179.37.1]
  3    18 ms    17 ms    17 ms  bzq-179-48-201.static.bezeqint.net [212.179.48.201]
  4    17 ms    17 ms    17 ms  bzq-179-80-241.static.bezeqint.net [212.179.80.241]
  5    17 ms    18 ms    17 ms  bzq-179-124-201.static.bezeqint.net [212.179.124.201]
  6    17 ms    17 ms    17 ms  bzq-179-124-138.static.bezeqint.net [212.179.124.138]
  7    19 ms    18 ms    18 ms  bzq-179-59-1.static.bezeqint.net [212.179.59.1]
  8    32 ms    36 ms    41 ms  10.20.110.18
  9   105 ms    84 ms    71 ms  bzq-179-79-228.static.bezeqint.net [212.179.79.228]
 10    73 ms    88 ms   150 ms  bzq-179-79-228.static.bezeqint.net [212.179.79.228]

Trace complete.

c:\>
```

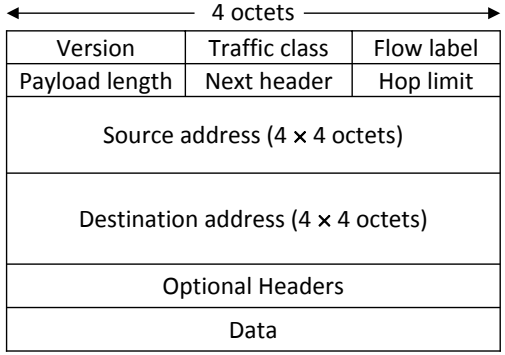

IPv6 (RFC 2460)

Motivation

- IPv6 address = 128 bits (16 octets)
- Service-specific fields
  - Quality of Service
  - Multicasting
  - Security
  - Mobility

Header

- Fixed 40-byte header
  - 10 × 4 bytes
- Optional header fields
  - Service-specific
- Next Header field
  - Identifies optional header or protocol of carried data
- Header daisy chain
  - Each optional IPv6 header contains Next Header field



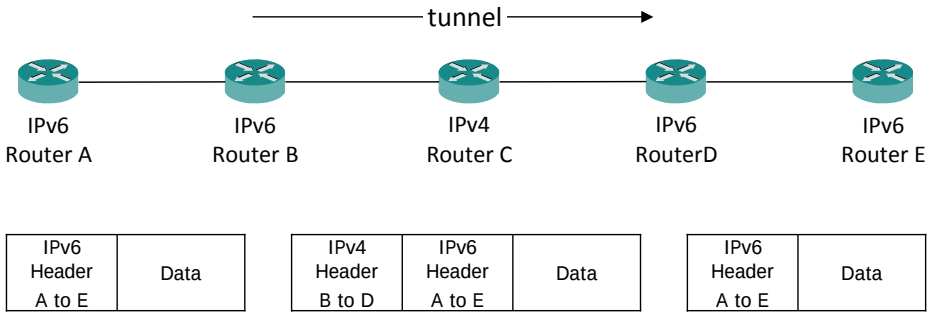
IPv4 To IPv6 Transition

IPv4 routers

- Replaced gradually
- Work with IPv6 routers

IP tunneling

- Encapsulate IPv6 datagram as data in IPv4 datagram



Infrastructure Layers

Infrastructure
Internet perspective

Change in point of view

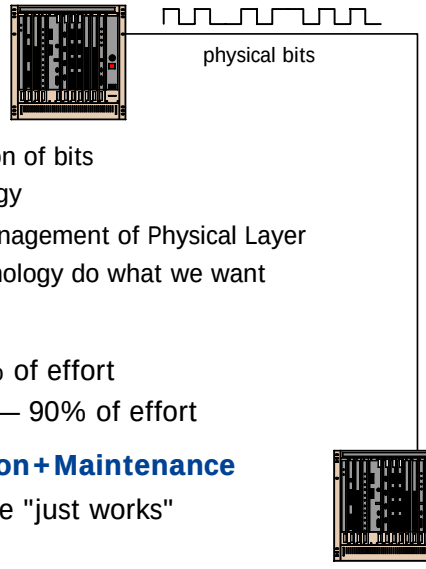
Internet standards do not discuss Data Link + Physical Layers

Hardware developers define standards

Internet Layer Model		
Application	Internet application Expects Internet services from OS	Internet Aware
Transport	Local + remote ports Service requirements	
Network	End-to-end IP routing + forwarding	
Infrastructure	Data Link Layer — hardware management Physical Layer — hardware	Not Internet Aware

Infrastructure
Engineering perspective

Infrastructure layers



Bottom-up design

Physical layer (**PHY**)

Defines physical transmission of bits

Exploits a physical technology

Data Link layer (**DL**) defines management of Physical Layer

How to make physical technology do what we want

Infrastructure management

Delivering data messages — 10% of effort

Making hardware work correctly — 90% of effort

OAM = Operations+Administration+Maintenance

Application assumes infrastructure "just works"

"Just works" ⇒

Reliability, availability, stability, serviceability, growth

Infrastructure
Economic perspective

Enormous investment in existing equipment

Global network of hardware nodes + transmission lines

Developed to provide many services

Internet (IP-based unreliable connectionless) just one service

Most developed before Internet

Telegraph — 1794

Telephone — 1876

Teletype modem — 1943

Digital telephone — 1962

Internet opened to public — 1992

Hardware updates

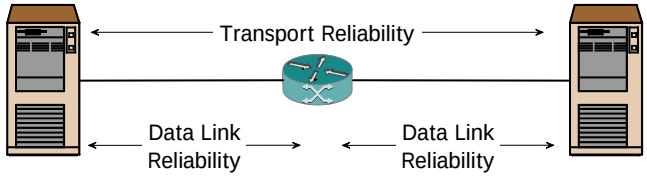
Replacement of manufactured hardware

Slower than software updates

More expensive than software updates

Data Link Functions

Similar to transport layer functions



Framing

- Assemble network PDUs into hardware packets
- Attach header + trailer for Data Link and Physical layers

Medium access + flow control + congestion control

- When / how transmitter sends data onto link
- Transmitter avoids overflow of receiver buffer
- Transmitters avoid interfering with other transmitters

Error control

- Detect / correct transmission bit errors

Local addressing

- Convert network addresses to hardware local addresses

Data Link Sublayers

Logical Link Control (LLC) sublayer

- Multiplexing of data sources / destinations
- Packet type identification
- Error correction
- Flow control

Medium Access (MAC) sublayer

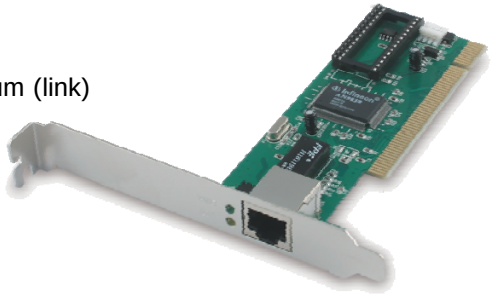
- Network topology
- Medium access management
  - Sharing medium among nodes
  - Permission to transmit
- Data frame structure
- Hardware (MAC) addressing
- Error detection

LLC Sublayer	Data Link Layer	2
MAC Sublayer		
Physical Layer		1

Interface to Data Link Layer

Infrastructure layers

- Typically implemented in hardware
  - PHY — physical circuits for transmitter / link / receiver
  - DL — embedded program in firmware (ROM) + controller
- Host node
  - Network interface Card (NIC)
    - Connection port to medium (link)
    - Controller
    - Transceiver



- Switching node
  - Switching fabric
  - Controller + multiple transceivers + connection ports to medium (link)

DL layer interface

- Interface to OS level hardware driver
- Network PDU ↔ OS driver ↔ NIC / switch controller ↔ transceiver

Framing

Data frame format

Header	Data	Trailer

Data link protocol management

- Header / Trailer format
  - Similar to headers at network and transport layers
- Addressing, error control, flow control, ...

Physical layer hardware management

- Transmission parameters
  - Bit rate, Baud rate, modulation method, ...
- Transmitter / receiver synchronization
  - Clock training bits
  - 1010101010 ... allows receiver clock to sync
- Frame marking
  - Start / Stop Fields
  - Start field / byte count
  - Lose sync ⇒ drop frame + wait for new Start Field

Frame Marking Methods

STX / ETX

ASCII control codes

STX	Start of Text	0x02
ETX	End of Text	0x03
DLE	Data Link Escape	0x10

DLE STX	Rest of Header	Data	Rest of Trailer	DLE ETX

DLE in data stream

Byte stuffing

Transmitter sends **DLE** as **DLE DLE**

Receiver removes extra **DLE**

Frame Marking Methods

Flags

Start = Stop = 01111110 = 0x7E

01111110	Rest of Header	Data	Rest of Trailer	01111110

7E in data stream

Byte / bit stuffing

Byte stuffing

Send **7E** as **7E 7E** — receiver removes extra **7E**

Bit stuffing

Send **11111** as **111110** — receiver removes extra **0**

Error Control

Check sequence

Transmitter

Calculates hash of data

Includes sequence in transmitted header / trailer

Receiver

Calculates hash of data

Compares received sequence with calculated sequence

Error Control

Parity (even parity)

Parity bit = XOR of data bits

Data + parity = even number of 1 bits

Cyclic Redundancy Code (CRC)

D = Data field

G = Generator

Predetermined pattern of r+1 bits

R = Remainder of $(D \times 2^r) / G$ (modulo 2 division) = CRC field

Generally $(D \times 2^r + R) / G = 0$

D = data	$R = (D \times 2^r) / G$

Modulo 2 Polynomial Arithmetic

Represent data bits as coefficients of polynomial

$$D = a_{n-1}a_{n-2}\ldots a_0$$

$$D(x) = a_{n-1}\times x^{n-1} + a_{n-2}\times x^{n-2} + \ldots a_0 \times x^0$$

Arithmetic modulo 2 in each order (XOR)

$$0+0=1+1=0 \qquad 0+1=1+0=1$$

Polynomial addition = subtraction

$$A(x)\pm B(x) = a_{n-1}\times x^{n-1} + \ldots + a_0\times x^0 \pm \left(b_{n-1}\times x^{n-1} + \ldots + b_0 \times x^0\right)$$
$$= \left(a_{n-1}\oplus b_{n-1}\right)\times x^{n-1} + \ldots + \left(a_0 \oplus b_0\right)\times x^0$$

Polynomial multiplication / division

Multiply / divide as usual

Modulo 2 arithmetic in each order

Examples $\left(x^3+x\right)\left(x^2+1\right) = x^5 + x^3 + x^3 + x = x^5 + (1+1)x^3 + x = x^5 + x$

$\left(x^5+x\right)\div\left(x^2+1\right) = x^3 + x$

Modulo 2 Long Division

$$\frac{x^5+x}{x^2+1}$$

$$x^2+1 \;\overline{)\; x^5 + x \;}\; x^3 + x$$

$$x^5 + x^3$$

$$x^3 + x$$

$$x^3 + x$$

$$0$$

Cyclic Redundancy Code (CRC)

Why it works

Data

D = data

Shift left D r bits ⇒ D → D × 2^r

D = data	0 … 0

Divide by G ⇒ D × 2^r → D × 2^r / G = Q + **R** / G

Quotient Q

Remainder R

Transmit T = D × 2^r + **R**

D = data	R

Receiver calculates T / G

T / G = D × 2^r / G + **R** / G = (Q + **R** / G) + **R** / G = Q + (**R** + **R**) / G

= Q + **0** / G = Q

CRC Example

Data stream

D = 1101011011

Generator

G = 10011

```
                1 1 0 0 0 0 1 0 1 0
          ------------------------
1 0 0 1 1 | 1 1 0 1 0 1 1 0 1 1 0 0 0 0
            1 0 0 1 1
          ---------
              1 0 0 1 1
              1 0 0 1 1
          ---------
                  0 1 0 1 1 0
                  1 0 0 1 1
                ---------
                  0 0 1 0 1 0 0
                    1 0 0 1 1
                  -----------
                      1 1 1 0
```

Remainder

R = 1110

Transmit

T = 1101011011 1110

CRC check at receiver

```
                1 1 0 0 0 0 1 0 1 0
          ------------------------
1 0 0 1 1 | 1 1 0 1 0 1 1 0 1 1 1 1 1 0
            1 0 0 1 1
          ---------
              1 0 0 1 1
              1 0 0 1 1
          ---------
                  0 1 0 1 1 1
                  1 0 0 1 1
                ---------
                  0 0 1 0 0 1 1
                    1 0 0 1 1
                  -----------
                      0 0
```

Zero remainder ⇒ no error

CRC Standards

CRC-12

$$G(x) = x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$$

CRC-16

$$G(x) = x^{16} + x^{15} + x^2 + 1$$

CRC-CCITT

$$G(x) = x^{16} + x^{15} + x^5 + 1$$

CRC-32

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Flow Control

Go Back N (GBN)

- Transmit N frames
- Wait for ACK
- Piggybacking — transmit ACK signals in data frame

Sliding Window

- N-bit SEQ number
- Window size — number of unACKed frames before stopping
- Source window
  - SEQ numbers of unACKed frames
  - Frames buffered at transmitter until ACKed
- Destination window
  - SEQ numbers of frames to be accepted
  - Frames passed to network layer in SEQ order
    - Out-of-order frames buffered
    - Frames too far ahead of window rejected
- Timeout — retransmit if no ACK after fixed time

High-Level Data Link Control (HDLC)

Family of data link protocols

Based on IBM SDLC

- Layer 2 protocol in mainframe SNA
- Originally for communication between CPUs and peripherals

Link Access Protocol (LAP)

Versions of HDLC used in public network architectures

X.25	**LAPB**
Frame Relay	**LAPF**
ISDN	**LAPD**
Ethernet Logical Link Control (LLC)	**IEEE 802.2**
Internet point-to-point	**SLIP, PPP**

High-Level Data Link Control (HDLC)

Data link attributes in HDLC

Framing	0x7E flag with byte / bit stuffing
Addressing	hardware level addressing possible
Error control	16-bit CRC-CCITT or 32-bit CRC-32
Flow control	3-bit SEQ number
Frame types	Information, Supervisory, Unnumbered

HDLC Frame Structure

General HDLC frame

01111110	Address	Control	data	CRC	01111110
8	8	8	≥ 0	16 / 32	8

Address

8 bit address ⇒ 256 hardware addresses

Control field

Specifies frame type / control

Information (data)

7	6 5 4	3	2 1 0
0	SEQ N(S)	p/f	NEXT N(R)

Supervisory (flow control)

7	6	5 4	3	2 1 0
1	0	type	p/f	NEXT N(R)

Unnumbered (management / connectionless)

7	6	5 4	3	2 1 0
1	1	type	p/f	subtype

HDLC Control Fields

Flow control

SEQ — sequence number of data frame

NEXT — next expected SEQ (ACK all previous frames)

Type

00 — ACK + Receiver Ready (RR)

01 — Reject (REJ): retransmit all frames from N(R)

10 — Receiver Not Ready (RNR): ACK N(R) but stop sending

11 — Selective Reject (SREJ): retransmit N(R)

Information (data)

7	6 5 4	3	2 1 0
0	SEQ N(S)	p/f	NEXT N(R)

Supervisory (flow control)

7	6	5 4	3	2 1 0
1	0	type	p/f	NEXT N(R)

Unnumbered (management / connectionless)

7	6	5 4	3	2 1 0
1	1	type	p/f	subtype

HDLC Polling

Polling

Primary host initiates communication

Secondary host responds

Poll / Final (p/f) bit

Invitation — primary to secondary with p = 1

Response

Secondary sends I-frames to primary with f = 0

Secondary sets f = 1 on last response frame

Information (data)

7	6 5 4	3	2 1 0
0	SEQ N(S)	p/f	NEXT N(R)

Supervisory (flow control)

7	6	5 4	3	2 1 0
1	0	type	p/f	NEXT N(R)

Unnumbered (management / connectionless)

7	6	5 4	3	2 1 0
1	1	type	p/f	subtype

HDLC Internet Dial-Up Protocols

Serial Line Internet Protocol (SLIP)

RFC 1055

0xC0	IP datagram with byte stuffing (C0 → DB DC, DB→ DB DB)	0xC0

Point-to-Point Protocol (PPP)

Layer 2 protocol used between

Internet routers

Host and Internet service provider (ISP)

01111110	11111111	11000000	Protocol	Data	CRC	01111110
8	8	8	8 or 16	≥ 0	16 / 32	8

Address = 11111111 = broadcast

HDLC control = 11000000 = Unnumbered (connectionless data)

Protocol

Protocol in data field

Network protocol or link negotiation protocol (upper layer 2 sublayer)

PPP Protocol Options

01111110	11111111	11000000	Protocol	Data	CRC	01111110
8	8	8	8 or 16	≥ 0	16 / 32	8

Standard network protocols

- IP, IPX, AppleTalk, ...
- Datagram in data field

Control Protocols

- Link Control Protocol (LCP)
  - PPP options
    - Header compression (remove control / address fields)
    - Size of protocol / CRC fields and data
  - Test
  - Terminate
- Network Control Protocol (NCP)
  - Network layer options
    - Protocol
    - Address
    - Header compression (encode header fields)
  - Authentication (ISP user / password exchange)

Shared Medium Networks

Physical layer

- Multiple nodes transmit on single medium
  - Time division
  - Frequency division
  - Code division
- Shared physical medium ⇒ local area network (LAN)

Data link layer

- Medium access (MAC) sublayer
  - Allocates medium capacity among nodes
  - Error detection
  - Network topology
- Logical link control (LLC) sublayer
  - Frame types
  - Flow control
  - Error correction
  - Protocol negotiation

LLC Sublayer	Data Link Layer	2
MAC Sublayer		
Physical Layer		1

Medium Access Sharing

Time division

- Each host granted full bandwidth in allocated time slot
  - Time slot allocated statistically or deterministically
- Examples
  - Deterministic — telephone switching
  - Statistical — Ethernet, WiFi, ...

Frequency division

- Each host granted partial bandwidth in all time slots
- Examples
  - Commercial radio / TV
  - Bluetooth

Code division

- Each host granted full bandwidth in all time slots
- Each host transmits using different coding scheme
- Example
  - Cellular CDMA

Common Shared Medium Networks

Ethernet	IEEE 802.3 local area network Wired time-division access
WiFi	IEEE 802.11 local area network Wireless time-division access
Bluetooth	IEEE 802.15 personal area network Wireless frequency-division access
WiMAX	IEEE 802.16 metropolitan area network Wireless time/frequency-division access
GSM / UTMS	ITU 2G / 3G cellular network Wireless time/frequency-division access
CDMA / CDMA2000	ITU 2G / 3G cellular network Wireless code-division access

What are IEEE 802 and ITU?

Institute of Electrical and Electronics Engineers (IEEE)

- Professional organization
- Coordinates technical standards for electronic equipment

IEEE 802 Committee

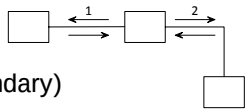
- Standards committee for Data Link and Physical Layer
- OEMs (original equipment manufacturers)
  - Develop hardware / software systems at infrastructure layers
  - Request standardization (recognition) from 802 committee
- Other 802 standards
  - 802.1 — bridging (interconnecting different 802 LANs)
  - 802.2 — LLC sublayer for 802 LANs
  - 802.4 — Token Bus (LAN for manufacturing environments)
  - 802.5 — Token Ring (ring topology LAN)

International Telecommunication Union (ITU)

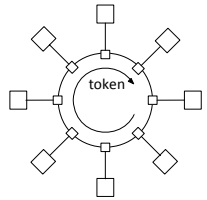
- UN standards committee
- Sets telephone and (non-Internet) WAN standards

Time Division Medium Access

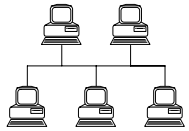
Deterministic



- Polling
  - Primary node initiates session (sends data to secondary)
  - Secondary node responds (sends data to primary)
  - Optional mode in WiFi
- Token passing
  - Token message passes from host to host
  - Host with token may transmit
  - Token Ring, Token Bus, FDDI



Statistical



- Aloha
  - Hosts transmit at random / hope to avoid collisions
- Carrier sense
  - Hosts listen for other transmissions / try to avoid collisions
- Arbitration
  - Deterministic procedure chooses among random group of hosts

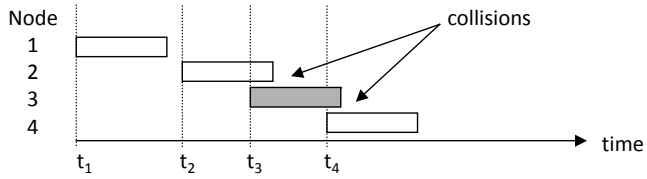
ALOHAnet

Background

- First wireless packet data network (1971)
- Low data volume
- Connected University of Hawaii campuses (separate islands)

Protocol

- Host transmits when ready
- Two frames overlap in time
  - Collision ⇒ both frames corrupted
  - Retransmit after random wait



Utilization and Throughput

Capacity

τ = time (seconds) to transmit 1 packet

R = maximum packets/second on medium = $\frac{1}{\tau}$

Traffic

λ = actual packets/second transmitted by hosts

Utilization

$$G = \frac{\lambda}{R} = \lambda\tau$$

Collisions

P_0 = probability success (packet transmitted without collision)

At receiver

$\lambda' = \lambda P_0$ = uncorrupted packets/second

Throughput

$$S = \frac{\lambda'}{R} = \frac{\lambda P_0}{R} = GP_0$$

ALOHA Throughput

Success = no collisions

Previous packet starts at $t_2 < t_3 - \tau$

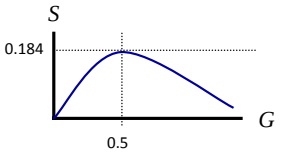
Next packet starts at $t_4 > t_3 + \tau$

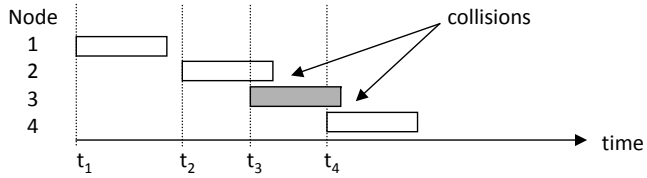
Interval of no transmissions = $t_4 - t_2 > 2\tau$

Packets obey Poisson statistics

$$P(k \text{ packets in } T \text{ seconds}) = \frac{(\lambda T)^k}{k!} e^{-\lambda T} \Rightarrow P_0 = \frac{(2\tau\lambda)^0}{0!} e^{-2\tau\lambda} = e^{-2\tau\lambda} = e^{-2G}$$

$$\boxed{S = Ge^{-2G}}$$





Slotted ALOHA

Central clock

Synchronize packet transmissions

Transmit new packet constructed between t_k and $t_k + \tau$

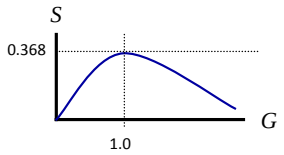
Collision

Two nodes construct packets in same interval

Probability of success (of my packet)

No other packets constructed during interval τ

$$P_0 = \frac{(\tau\lambda)^0}{0!} e^{-\tau\lambda} = e^{-\tau\lambda} = e^{-G} \Rightarrow \boxed{S = Ge^{-G}}$$



Carrier Sense

Carrier Sense Multiple Access (CSMA)

Nodes listen for transmissions before transmitting

No transmission — node can transmit

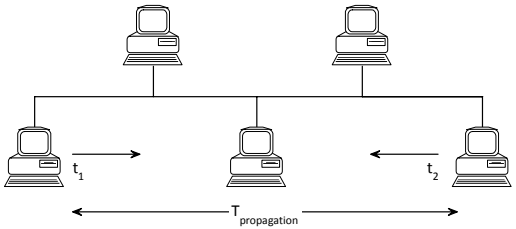
Transmission — node waits until end of transmission

Collision

Multiple nodes transmit "at same time"

"Same time"

$|t_1 - t_2| < T_{propagation}$



Collision Detection (CD)

Nodes listen for collision

Corrupted data

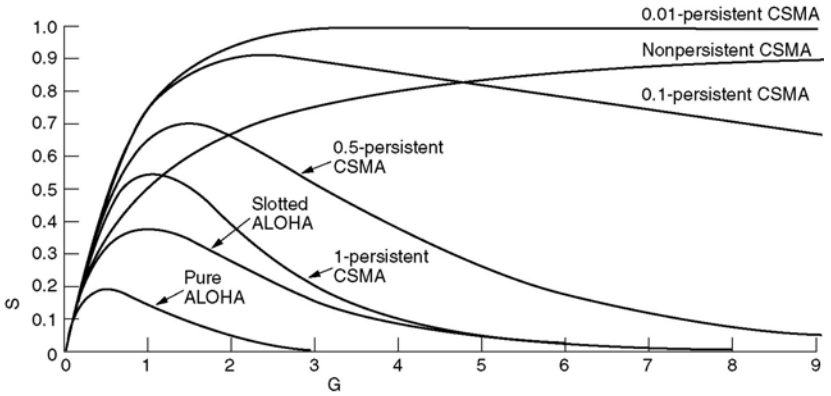
On collision

All nodes stop transmitting

Nodes jam transmission

Nodes waits random backoff before retransmitting

CSMA/CD Throughput



Tanenbaum, Fig. 4-4

Persistent CSMA

No carrier detected ⇒ node with data transmits

q-persistence

Node transmits with probability $0 < q < 1$

$q < 1$ ⇒ fewer collisions but longer latency

Arbitration

Deterministic medium access

- Random group of nodes request access
- One node chosen by deterministic algorithm
- No collisions
- Efficient throughput

Used within computer

- Peripheral Component Interconnect (PCI)
  - Multiple CPUs and peripherals compete for access to memory
  - PCI bridge allocates memory access efficiently
- Intel Multibus II
  - Multiple nodes request bus access using pseudo-randomized ID
  - Highest ID proceeds

Binary countdown switch

- Multiple hosts begin transmitting onto bus
  - Bus output = logical OR of all inputs
- Host sends 0 but sees 1 on bus ⇒ host stops

Ethernet

Family of wired LAN systems

- Defined at physical and data link layers
- Dominant / generic LAN technology

Background

- Developed 1974 at Xerox PARC
- Commercialized by Xerox / Intel / Digital in 1980
- Standardized as IEEE 802.3 in 1982
  - 10 Mb/s baseband transmission
  - Bus topology — single coaxial cable < 2.5 km
  - CSMA/CD
    - Shared bus topology → CSMA
    - Long propagation delay on coaxial cable → CD

Developments

- Bit rates: 10 Mbps → 100 Mbps → 1 Gbps → 10 Gbps → 100 Gbps
- Media: coaxial cable → hub (virtual bus on star) → switch
  - Ethernet switch — non-blocking N × N switch with no collisions

Ethernet Topologies

Coaxial cable

Distributed CSMA/CD

- 1980 – 1990
- Original Ethernet design at 10 Mbps

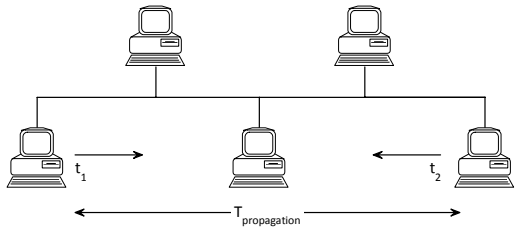
Shared physical bus

- Coaxial cable < 2.5 km

$$\text{End-to-end propagation delay} = \frac{2.5 \text{ km}}{3\times10^{5} \text{ km/sec}} = 8\times10^{-6} \text{ sec}$$

$$\text{Bits transmitted before carrier detect} = (10 \text{ Mb/s})\times(8\times10^{-6} \text{ sec}) = 80 \text{ bits}$$



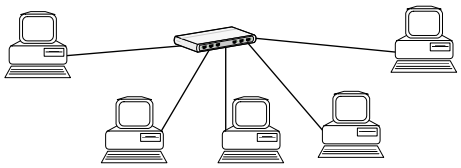


Ethernet Topologies

Passive hub

Centralized CSMA/CD

- 1990 – present
- Fast Ethernet
- 100 Mbps



Logical bus on physical star

- Central hub
- Multiple cables < 100 m
- Each station (STA = node) receives logical OR of all inputs
- Multiple frames ⇒ collision

$$\text{End-to-end propagation delay} = \frac{200 \text{ m}}{3\times10^{8} \text{ m/sec}} \approx 7\times10^{-7} \text{ sec}$$

$$\text{Bits transmitted before carrier detect} = (100 \text{ Mb/s})\times(7\times10^{-7} \text{ sec}) = 70 \text{ bits}$$

Ethernet Topologies

Active hub

Ethernet switch

1995 – present

Physical star

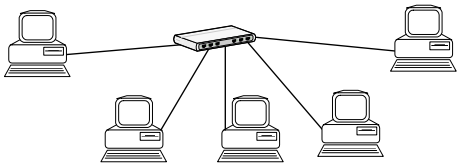
100 Mbps → 1 Gbps → 10 Gbps → 100 Gbps

Fast N × N non-blocking switch

- Hub learns MAC addresses at each switch port
- Each frame directed to port by destination address in frame
- Large output buffer at each port

All stations can send at same time

No collisions



802.3 Ethernet Standards

10 Base 5	10 Mb/s on thick coaxial cable
10 Base 2	10 Mb/s on thin coaxial cable
10 Base T	10 Mb/s on 1 twisted pair
100 Base T	100 Mb/s on 1 twisted pair
100 Base TX	100 Mb/s full duplex on 2 twisted pairs
100 Base FX 100 Base SX 100 Base BX 100 Base LX	100 Mb/s full duplex on optical fibers
1000 Base TX	1 Gb/s full duplex on 2 twisted pairs
1000 Base SX 1000 Base LX 1000 Base BX 1000 Base ZX	1 Gb/s full duplex on 2 optical fibers

Ethernet Frame

Preamble	Start	Dest Address	Src Address	Type or Length	Data	CRC
7 bytes	1 byte	6 bytes	6 bytes	2 bytes	46 – 1500 bytes	4 bytes

Preamble	7 bytes of 10101010 for sync of receivers	
Start	10101011	
Address	Hardware (MAC) address of node 48-bit MAC addresses assigned by OEM and fixed in hardware Broadcast address FF:FF:FF:FF:FF:FF (frame read by all STAs)	
Length	Length of data field (<1500)	
Type	Code identifying protocol in data field Used in most Ethernet systems Type codes > 1536 =0x600	IP = 0x0800 AppleTalk = 0x809B ARP = 0x0806
CRC	CRC-32	

IEEE 802.2

LLC sublayer for 802.3 Ethernet

Based on HDLC

Permits connection oriented services at data link layer

802.2 I/S

DSAP	SSAP	control	data + pad
1 byte	1 byte	2 bytes	42 - 1496 bytes

802.2 U

DSAP	SSAP	control	data + pad
1 byte	1 byte	1 bytes	43 - 1497 bytes

DSAP	Destination service access point (protocol / service at destination)
SSAP	Source service access point (protocol / service at source)
Control	Frame type + SEQ + ACK (I / S frames) Frame type (U frame)

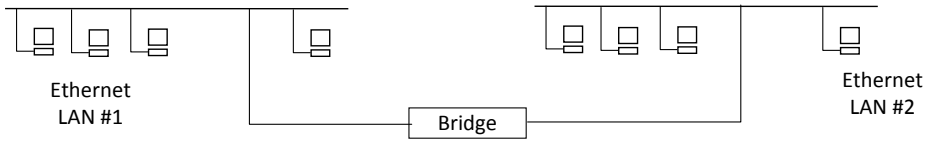
Segmentation

Ethernet bridge

2-port switch

Connects 2 Ethernet segments

Reduces traffic in each segment



Initialization — promiscuous mode

Bridge passes every Ethernet frame

- Listens as destination STA
- Repeats Ethernet frame as source STA

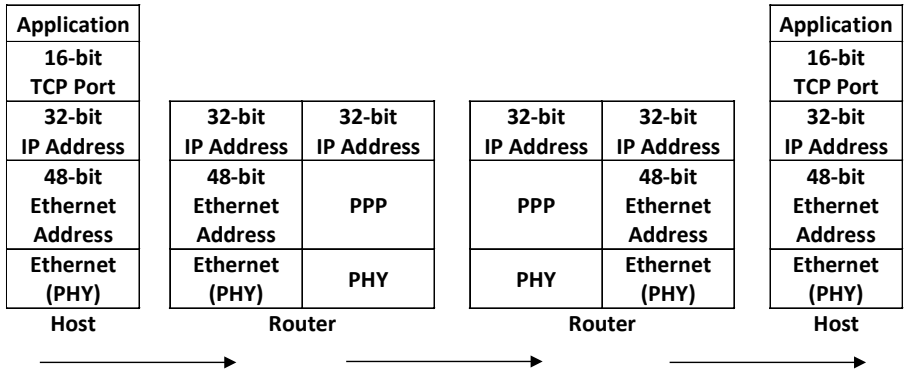
Bridge learns network topology

Builds table of source MAC addresses

Forwards only inter-segment frames

Relationship of Protocol Layers

Typical network



Locate router by IP address (uses default gateway)

Send to router by MAC address

Point-to-point

Locate host by IP address

Send to host by MAC address

Ethernet always uses source / destination Ethernet addresses — not IP addresses

How does host find MAC address for router?

Address Resolution Protocol (ARP)

Look-up MAC address by IP address (RFC 826)

Q: Who has IP = a.b.c.d ? (MAC layer broadcast)

A: I am IP = a.b.c.d with MAC = u:v:w:x:y:z

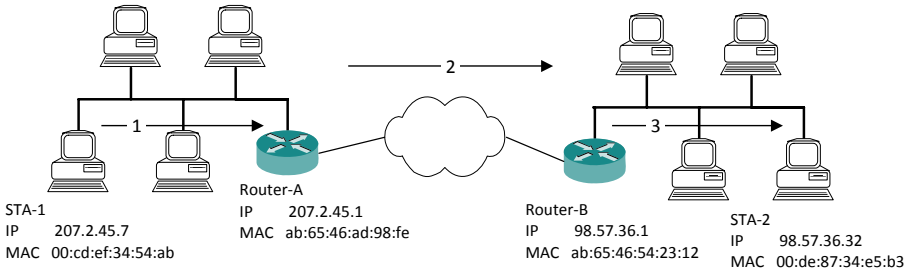
STAs store mappings in arp table

Windows / Linux **`arp -a`** prints arp table

ARP packet fields

HTYPE	Hardware type — MAC protocol
PTYPE	Protocol type — network protocol
HLEN	Hardware length — length in octets of MAC address
PLEN	Protocol length — length in octets of network address
Operation	1= request / 2 = reply
SHA	Sender hardware address
SPA	Sender protocol address
THA	Target hardware address (ignored in requests)
TPA	Target protocol address

Example



Frames between STA-1 and STA-2

	MAC Header	IP Header	IP data	MAC Trailer
1	SRC: 00:cd:ef:34:54:ab DST: ab:65:46:ad:98:fe	SRC: 207.2.45.7 DST: 98.75.36.32	TCP segment	CRC
2	PPP	SRC: 207.2.45.7 DST: 98.75.36.32	TCP segment	CRC
3	SRC: ab:65:46:54:23:12 DST: 00:de:87:34:e5:b3	SRC: 207.2.45.7 DST: 98.75.36.32	TCP segment	CRC

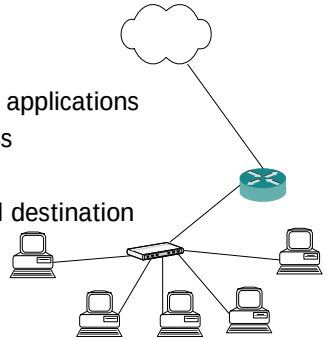
Advanced Switch / Router Interactions

Standard model

Switch organizes STAs into LAN

Intra-LAN traffic

- STAs use IP addresses as names for TCP/IP applications
- STAs use ARP to translate IP to MAC address
- STAs send frames on LAN by MAC address
- Packets contain MAC and IP address of local destination



Router organizes LAN into Internet AS

Inter-LAN traffic

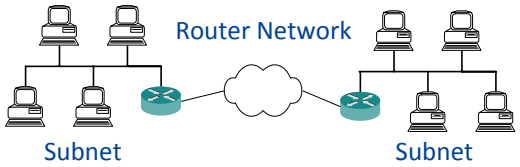
- STAs use IP addresses as names for TCP/IP applications
  - Local MAC addresses not available for remote STAs
- STAs send frames via router
- Packets contain
  - MAC address of router
  - IP address of remote destination

Advanced Switch / Router Interactions

Virtual LAN

Standard subnet

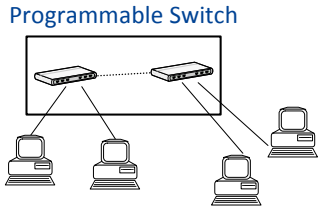
subnet-1 and subnet-2 are LAN broadcast domains



Virtual LAN (VLAN)

LAN switch configured to partition nodes into subnets

No router needed for subnetting



Advanced Switch / Router Interactions

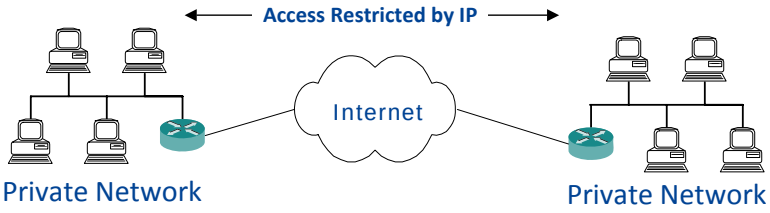
Virtual Private Network

Virtual Private Network (VPN)

Private network implemented on public infrastructure

Access to private networks restricted by IP

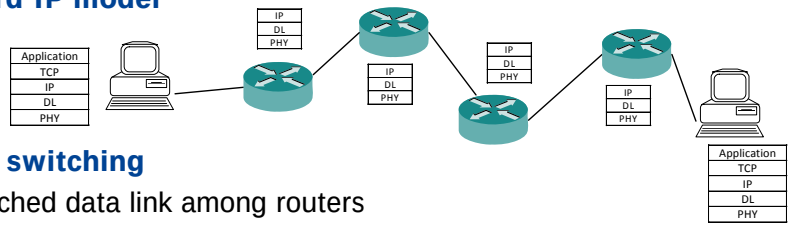
Possible encryption of data over public infrastructure



Advanced Switch / Router Interactions

Layer 3 Switching

Standard IP model



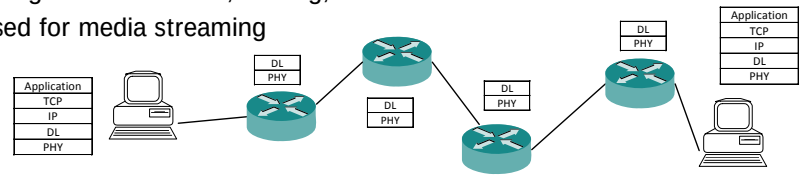
Layer 3 switching

Switched data link among routers

- Connection-oriented virtual circuit network
- Frame Relay, ATM, label switching, ...

Traffic crosses router network at layer 2

- Saves time of layer 3 processing
- Datagram read / write, routing, TTL
- Used for media streaming



Advanced Switch / Router Interactions

Label switching

Multiprotocol Label Switching (MPLS)

MAC Header	MPLS	IP Header	TCP Header	Data	MAC trailer

Header fields

One or more headers per frame — "stack" of labels

Label	20-bit ID
Traffic Class	3-bit QoS (quality of service) field
Stack flag	1-bit — if set, current label is last of "stack" of labels for frame
TTL	8-bit time to live field

Connection-oriented

Set (reserve) router path before data traffic begins

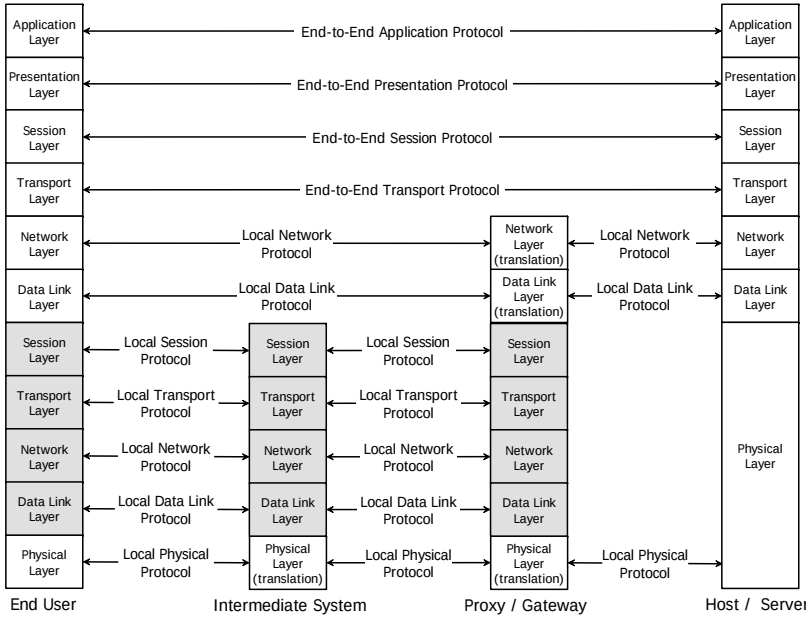
Label Distribution Protocol (LDP)

RSVP-TE — extension of Resource Reservation Protocol (RSVP)

MPLS-aware routers

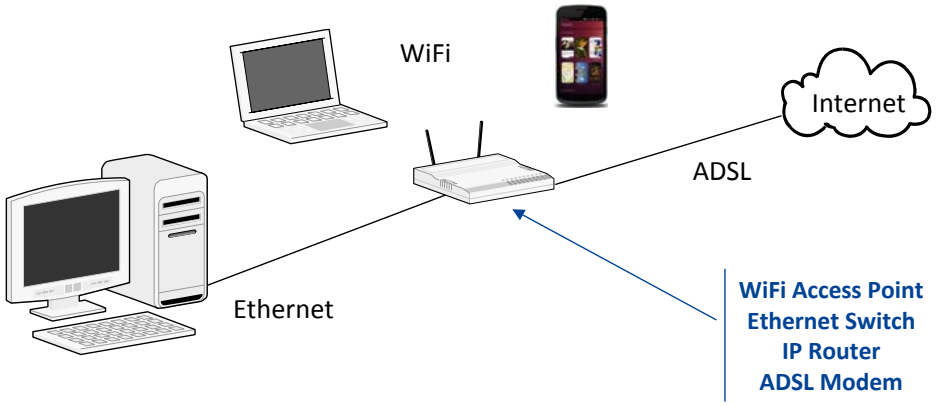
Forward frames on preset route by label ID

Tunneling in the OSI Model



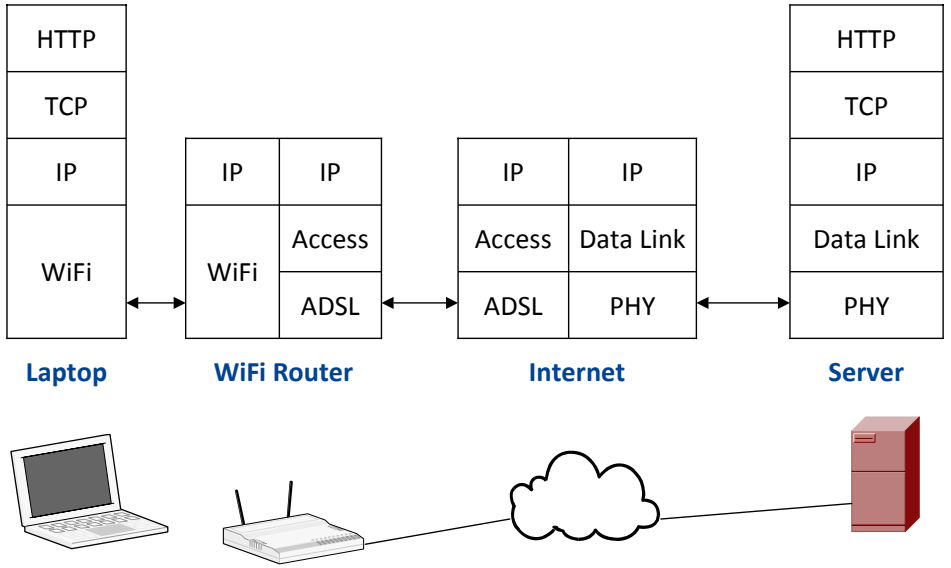
Small Office / Home Office (SOHO)

LAN (Local Area Network) to WAN (Wide Area Network)



Ethernet	802.3	Cable-based **LAN** protocol defined at **DATA LINK** and **PHY** layers
WiFi	802.11	Wireless **LAN** protocol defined at **DATA LINK** and **PHY** layers
ADSL	G.992.5	Cable-based transmission protocol defined at **PHY** layer

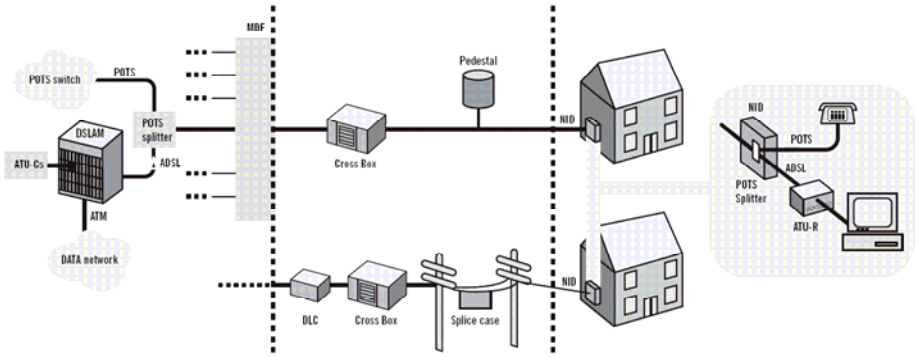
Laptop Browser to Web Server — Simplified View



ADSL — Asymmetric Digital Subscriber Line

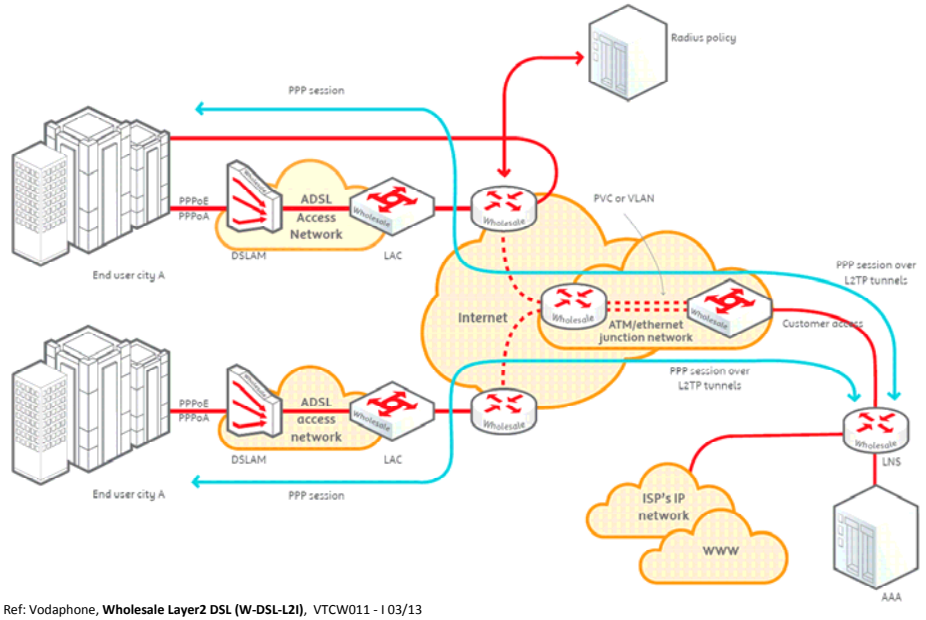
High speed transmission on standard voice line

- POTS — plain old telephone service
- 24 Mbps downstream
- 3.3 Mbps upstream



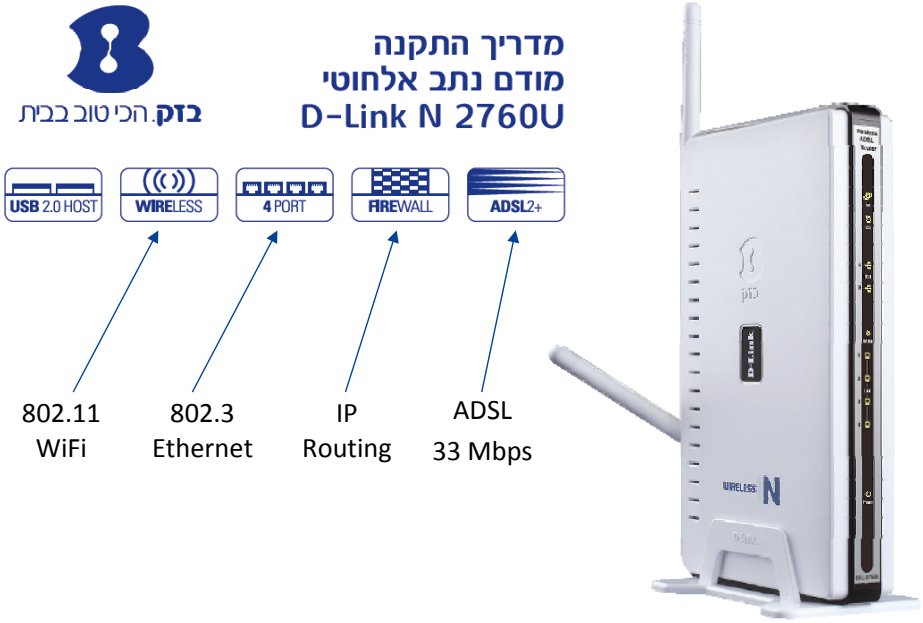
Ref: JDSU, **ADSL Technology**, JDS Uniphase Corporation, 2005

ADSL Access Network

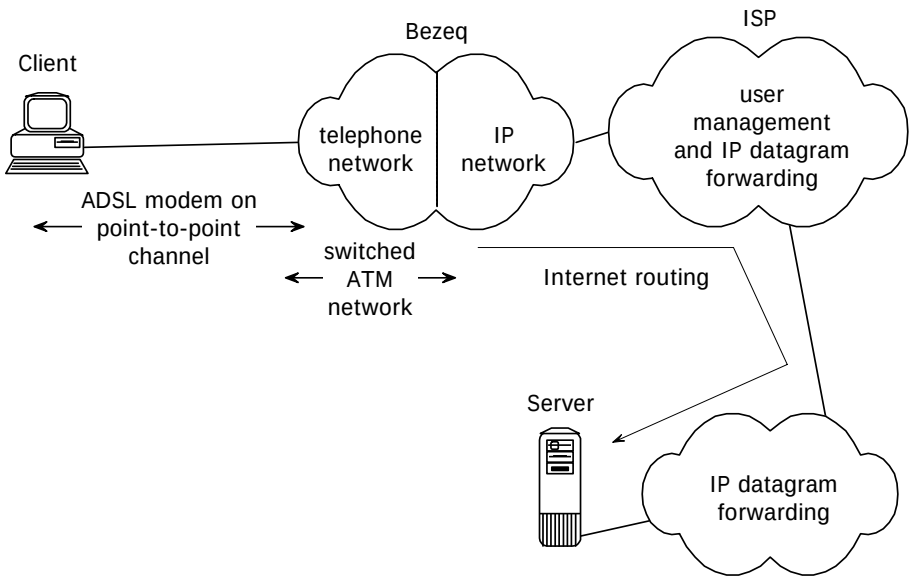


Ref: Vodaphone, **Wholesale Layer2 DSL (W-DSL-L2I)**, VTCW011 - I 03/13

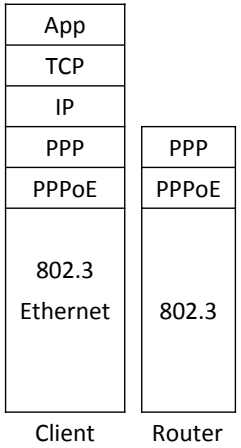
Typical Bezeq ATU-R



Fast Internet Access



Fast Internet Protocols — Typical Campus Case



PPP

Point to Point Protocol

Logon + connection management

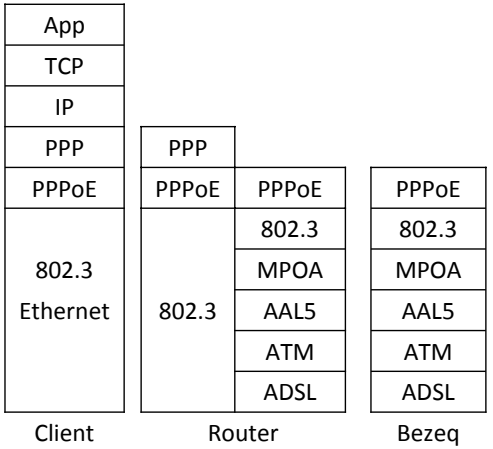
PPPoE

PPP over Ethernet

Virtual point-to-point connection over shared LAN

Client opens private session with ISP

Fast Internet Protocols — Typical Campus Case



ATM

Asynchronous Transfer Mode

Data Link protocol for broadband telephone services

Permits real time QoS

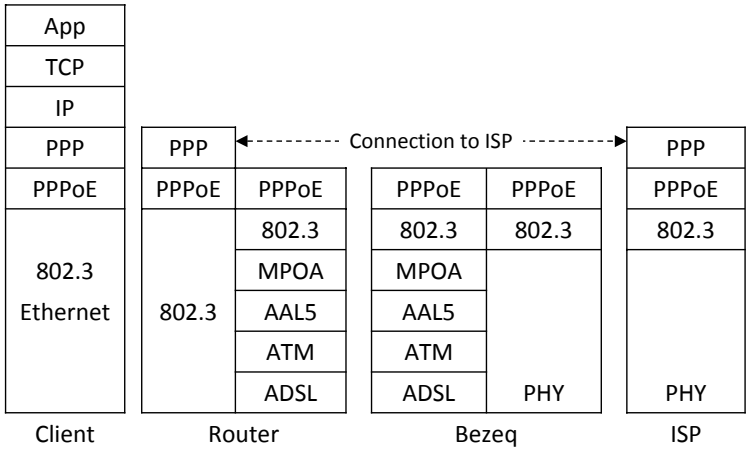
MPOA + AAL5

Adaptation protocols for ATM

ADSL

Physical bit transmission

Fast Internet Protocols — Typical Campus Case



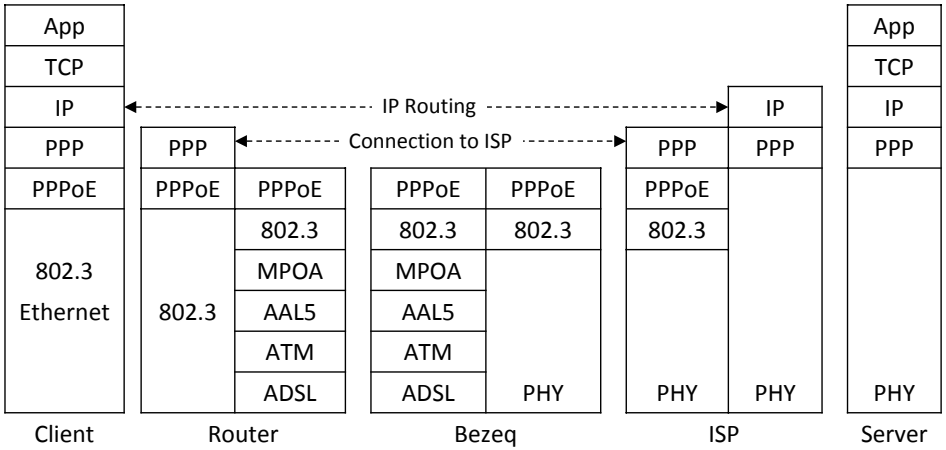
Connection to ISP

Client runs Network Control Protocol (NCP) over PPP

CHAP (challenge handshake authentication protocol) — User Name + Password

ISP authorizes user and engages IP forwarding

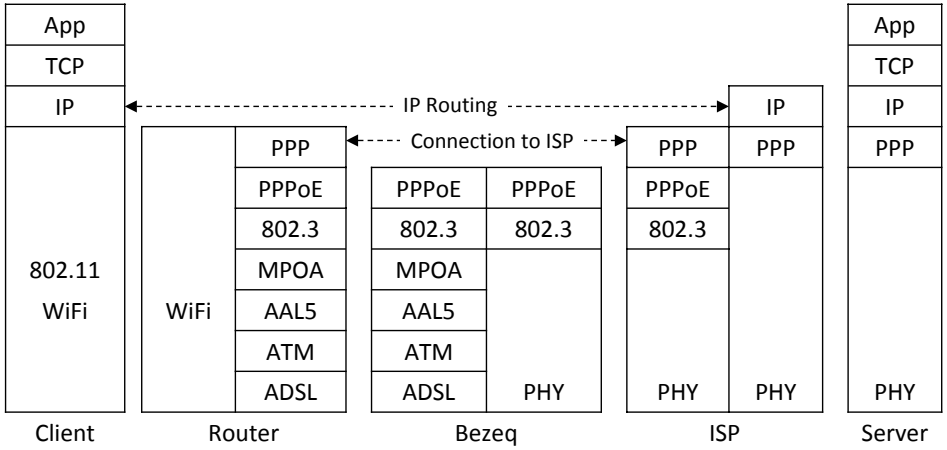
Fast Internet Protocols — Typical Campus Case



IP forwarding

ISP forwards IP datagrams to server via Internet backbone

Fast Internet Protocols — Typical SOHO Case



Router/modem initiates connection to ISP

- Runs NCP over PPP over PPPoE over Ethernet
- Router provides always-on Internet access over WiFi + Ethernet

Network Security and Management

Network Security

Confidentiality

Sender encrypts message
Receiver decrypts message
Only sender + receiver understand contents

Authentication

Sender confirms identity to receiver

Message integrity

Message not changed between sender and receiver

Service availability

Prevent Denial of Service (DoS) attacks

Friends and Enemies

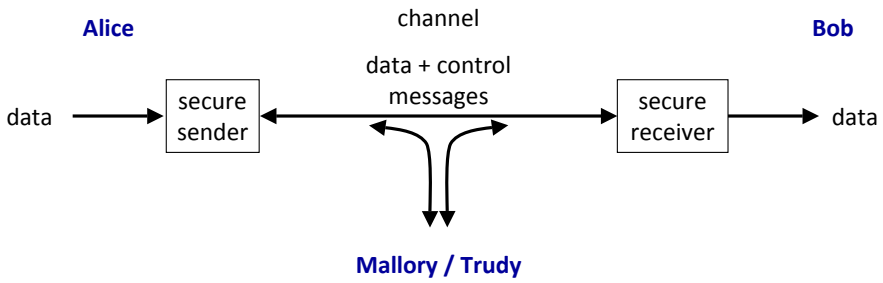
Bob + Alice

Legitimate system users
Communicate securely

Mallory + Trudy

Intruders
Intercept / delete / add / change messages

Philology of cryptology
Alice — party **A**
Bob — party **B**
Mallory — **mal**icious
Trudy — in**trud**er



Malicious Network Activity

Eavesdrop

Intercept legitimate messages for unauthorized reading

Message injection

Insertion of unauthorized messages into legitimate channel

Impersonation

Fake (spoof) source address in packet

Hijacking

Replacing legitimate sender or receiver in existing connection

Denial of Service (DoS)

Prevent access to service
Overloading server resources
Replacing legitimate server software / content

Cryptography

Plaintext

Original message m

Ciphertext

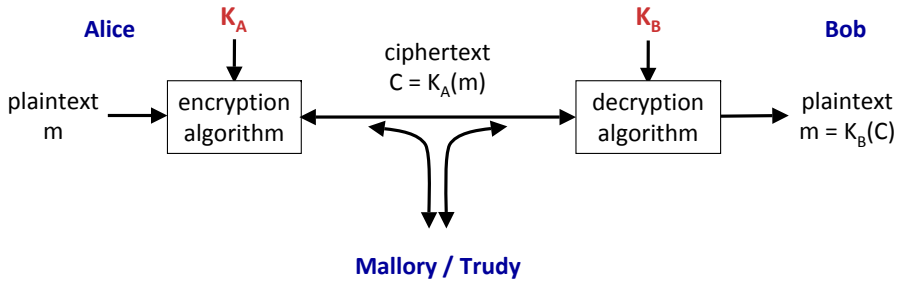
Message m encrypted with key K_A

Ciphertext C = $K_A(m)$

Decipher

Recover plaintext by decryption with key K_B

Plaintext m = $K_B(C) = K_B(K_A(m))$



Symmetric Key Cryptography

One secret key K

Encryption C = K(m)

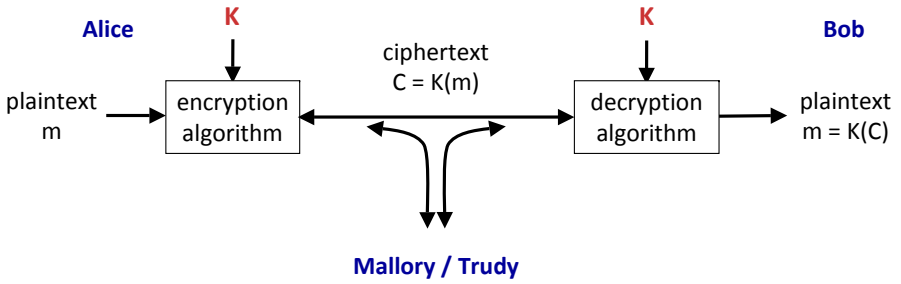
Decryption m = K(C) = K(K(m))

Alice + Bob share secret key K

Typical algorithm

Ciphertext = m ⊕ K

Ciphertext ⊕ K = (m ⊕ K) ⊕ K = m ⊕ (K ⊕ K) = m ⊕ 0 = m



Asymmetric Key Cryptography

Public key cryptography

Public encryption key K^+

$C = K^+(m)$

Private decryption key $K^- \neq K^+$

$m = K^-(C) = K^-(K^+(m))$

Rivest, Shamir and Adleman (RSA) algorithm

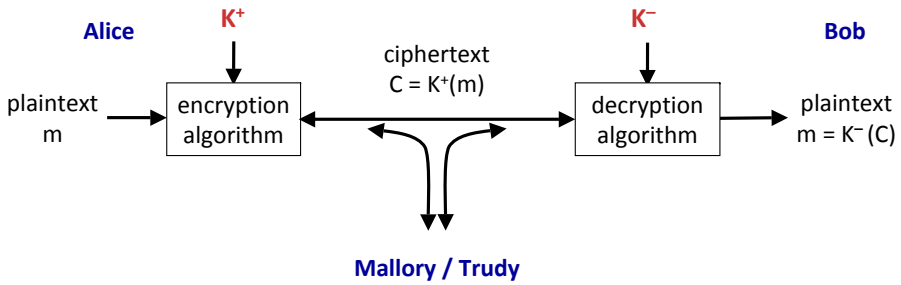
$m = K^-(K^+(m)) = K^+(K^-(m))$

Cannot obtain K^- from K^+ or K^+ from K^-

No shared key

Alice encrypts with Bob's published K^+

Bob decrypts with private K^-



Message Integrity

Message content not altered

Not corrupted / maliciously changed

Message digest

Public hash function h = H(m)

$m \neq m' \Rightarrow$ small probability of H(m) = H(m')

MD5 (RFC 1321) / SHA-1 (US NIST standard)

Alice

Computes h = H(m)

Sends (m,h)

Bob

Computes h = H(m) and compares with received h

H(m) = h ⇒ high probability of message integrity



Authentication

AAA Protocols

- Authentication
  - Check that user identity is authentic (as claimed)
  - Login, password, ...
- Authorization
  - Granting access / service rights
- Accounting
  - Tracking resource consumption by users
  - Logins, disk usage, ...

Common AAA systems

- Terminal Access Controller Access-Control System (TACACS)
- Kerberos
- Remote Authentication Dial In User Service (RADIUS)

Challenge / Response Authentication

Unencrypted

Challenge	Response	System Action
`Login:`	`Alice`	Check user name in database
`Password:`	`********`	Compare password with database

Encrypted

Challenge	Response	System Action
`Login:`	`Alice`	Check user name in database Set key K = password in database Set nonce R (one-time value) Send encrypted K(R)
`Password:`	`********` Set key K = `********` Decrypt R = K(K(R)) Send R + 1 unencrypted	 Correct value ⇒ correct password

Message Authentication Code (MAC)

Authentication key

Alice and Bob share secret key s

Authenticated message

- Alice
  - Computes h = H(m, s)
  - Sends (m, h)
- Bob
  - Computes h = H(m, s) and compares with received h
  - H(m,s) = h ⇒ high probability of message integrity + authenticity



Playback Attack

Authenticated message

- Alice sends Bob legitimate message with MAC
- Bob
  - Receives message
  - Confirms secret shared key
- Mallory
  - Captures Alice's message
  - Resends Alice's message to Bob
- Bob
  - Receives message
  - Confirms secret shared key
- Example
  - Electronic funds transfer (EFT)

Defense

Nonce (one-time use number)

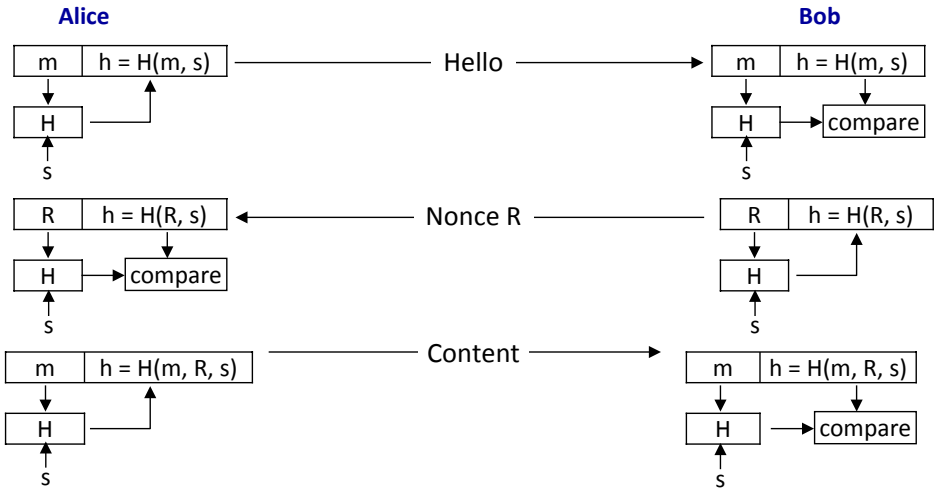
MAC with Nonce

Protected against replay

Nonce ⇒ correct 3-step handshake

Hello → R → Content

No 3-step handshake ⇒ ignore content message



Digital Signature

MAC with public key cryptography

Exploits RSA symmetry

$m = K^- (K^+ (m)) = K^+ (K^- (m))$

Signed message

Bob

Computes hash h = H(m)

Encrypts $ch = K^- (h)$ using private key K^- (only Bob can encrypt)

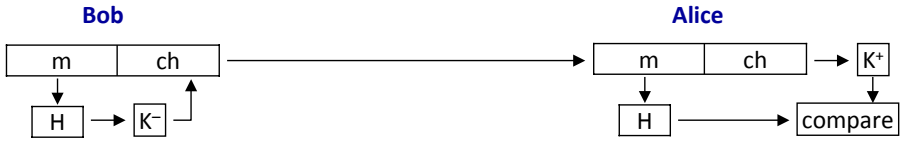
Sends (m, ch)

Alice

Decrypts $h = K^+ (ch)$ using Bob's public key K^+

Computes H(m) and compares with h

h = H(m) ⇒ message hash was signed using Bob's private key



Public Key Certification

Vulnerability of digital signature

Alice

Receives (m, ch) from Trudy calling herself "Bob"

Requests public key from "Bob"

Confirms m sent by "Bob"

Question

Is "Bob" really Bob

Certification Authority (CA)

Trusted organization

Certifies real-world identity of user

Shifts question to

Reliability of CA

Deviousness of intruder

Issues certificate to user

Certificate = real-world user information + user public key K^+

Digitally signed with CA private key K^-_{CA} (only CA can encrypt)

Certified Message

Bob

Calculates hash h = H(m)

Encrypts $ch = K^- (h)$ using private key K^- (only Bob can encrypt)

Sends (m, ch, certificate)

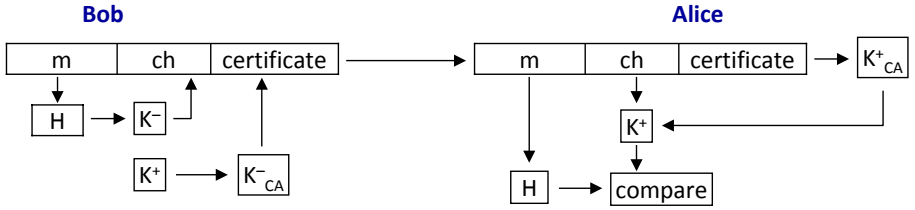
Alice

Decrypts $K^+ = K^+_{CA}$(certificate) using CA's public key K^+_{CA}

Decrypts $h = K^+ (ch)$ using Bob's public key K^+

Computes H(m) and compares with h

h = H(m) ⇒ message hash was encrypted using Bob's private key



Secure Sockets

Security layer between application and transport

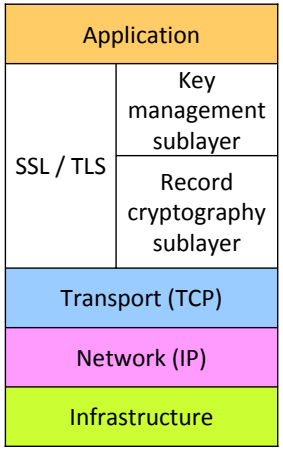
- Secure Sockets Layer (SSL)
  - Netscape browser (1993)
- Transport Layer Security (TLS)
  - RFC 2246 (1999 – 2011)

Services to application layer

- Confidentiality (RSA encryption)
- Authentication (certificates)
- Integrity (MAC hash protection)

Data fragmentation

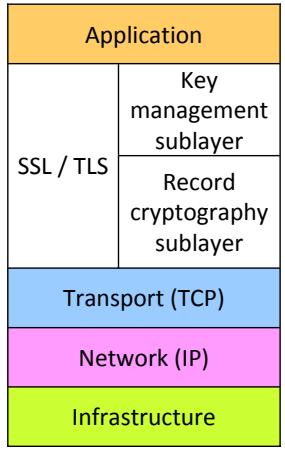
- Record = header + encrypted data fragment



Secure Sockets

Programmer perspective

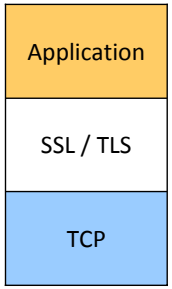
- Socket call invokes SSL / TLS instead of TCP
- SSL / TLS passes encrypted / protected data to TCP
- FTPS — FTP client / server over SSL / TLS
- HTTPS — client / server over SSL / TLS



SSL / TLS Overview

Session initialization

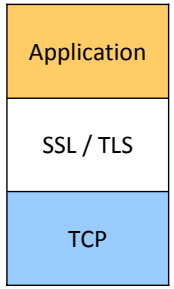
- Application
  - Client opens secure socket for channel to server
- SSL / TLS
  - Opens TCP session between client / server
  - Client / server negotiate certificates + keys + options
  - Establish secure channel over TCP
    - Confidentiality + authentication + integrity
  - Return socket descriptor to application



SSL / TLS Overview

Data exchange

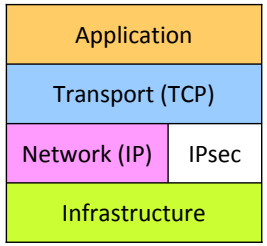
- Application
  - Sends data to open socket
- SSL / TLS
  - Client fragments app data
  - Encrypts data fragments + MAC
  - Adds record header
  - Forwards records to TCP
  - Server receives records from TCP
  - Checks / removes record header
  - Decrypts data fragments + MAC
  - Reassembles fragments to app data



Network Layer Security

Security layer at network layer

IPsec — RFC 4301 (2005)



Service options for IP datagrams

	Authentication Header (AH)	Encapsulating Security Payload (ESP)
Confidentiality		√
Integrity	√	√
Authentication	√	√
Replay protection	√	√

Modes of operation

Transport mode

- IP data encrypted / authenticated
- IP header not modified / encrypted

Tunnel mode

- Encrypted / authenticated datagram = data field of new datagram

Virtual Private Network (VPN)

Private network

Dedicated physically separate infrastructure

- Switches / routers / cables

Most secure at highest cost

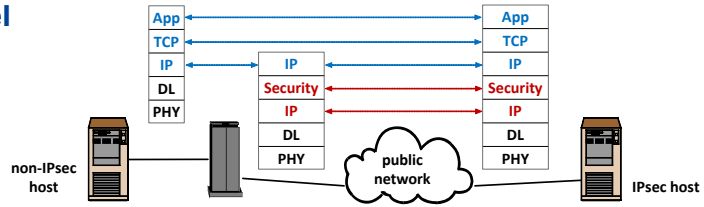
Virtual Private Network

Secure network implemented in shared public infrastructure

Security options

- Proprietary encryption software
- IPsec in tunnel mode
- TLS
- Traffic segregation in shared physical hardware

VPN tunnel



Security Association (SA)

Managing AH / ESP connections

Internet Security Association & Key Management Protocol (ISAKMP)

Security Association (SA)

Establish SA from sending to receiving entity before sending data

Simplex SA

- Each direction sets up separate SA

Connection-oriented stateful mechanism over IP

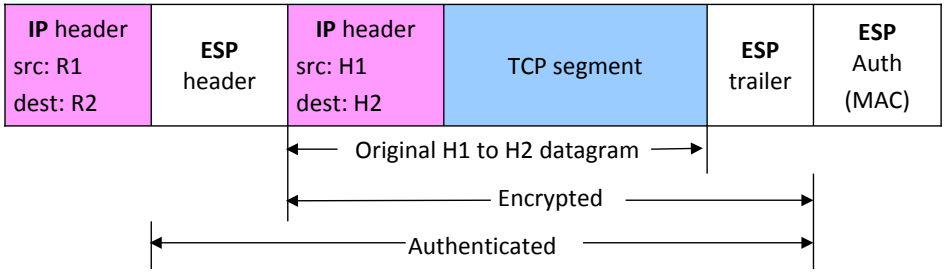
Example — SA for R1 to R2

R1 state table entries

- 32-bit SA identifier: Security Parameter Index (SPI)
- Source / destination interfaces — 200.168.1.105 / 193.68.2.23
- Encryption method / key
- Integrity check method / authentication key



IP Encapsulation



ESP header

Security Parameter Index (SPI)

SEQ

ESP trailer

Padding (for block cypher) + length of padding

Next header — protocol in original datagram



Firewall

Software / hardware based filter

- Blocks packets based on protocol criteria
- Intended to prevent
  - Unauthorized access to internal network from WAN
  - Unauthorized access to WAN by internal software

Stateless packet filtering

- Block packets based on
  - Source / destination IP address
  - TCP/UDP source / destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits (prevent any incoming TCP connections)

Stateful packet filtering

- Track status of TCP connections from SYN to FIN
  - Block packets inconsistent with TCP state machine
  - Block inactive connections

Intrusion Detection Systems (IDS)

Deep packet inspection

- Compare packet contents with database of virus / attack signatures

Correlations among multiple packets

- Port scanning
  - TCP SYN sequence of ports — map open ports
- Network mapping
  - Ping sequence IP addresses — map responding hosts
- Denial of Service (DoS) attack
  - Replayed packet to overload server

Statistical anomaly

- Compare traffic with statistical norms
  - Bandwidth demand
  - Protocols and service ports
  - User access
  - File updates

Data Management Zone (DMZ)

Term of military origin

- DMZ — Demilitarized Zone
- Border area between military powers
- Military activity not permitted

Network DMZ

- Border zone of AS (perimeter network)
- Unrestricted access from WAN

DMZ services

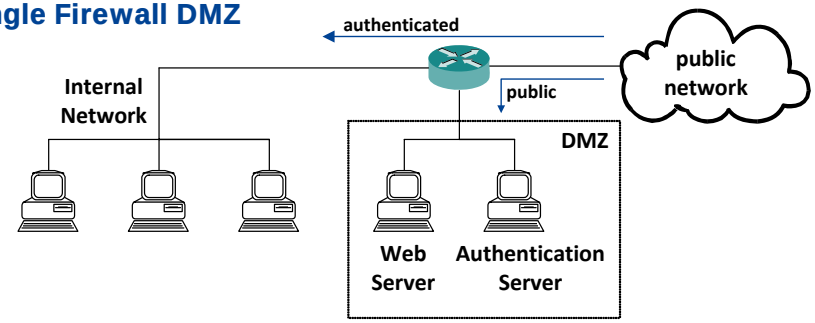
- Public web server
- Other unsecured services
  - Incoming mail server
  - Anonymous FTP server
- Secure authentication server to internal network

Reported intrusions to Pentagon network (1995)

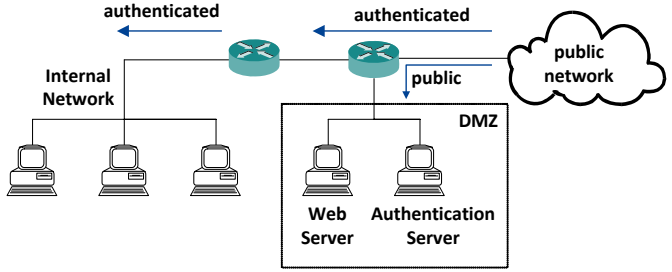
- DMZ — 250,000
- Internal — 0

DMZ Configurations

Single Firewall DMZ



Double Firewall DMZ



Network Management

OAM

- Operation — service availability and statistics monitoring
- Administration — Network resource accounting / assignment
- Maintenance — Repairs, upgrades

Provisioning

- Resource configuration to support services

Typical management tasks

- Resource
  - Controlling, planning, allocation, deployment, coordination, monitoring
- Traffic routing, load balancing, performance management
- Security management
- Fault management
- Account management

Management tools

- Monitoring agents installed on infrastructure
- Activity logs
- Sniffers

Simple Network Management Protocol (SNMP)

Management agents

- Hardware / software feature of network infrastructure
- Permits remote management of device
  - Monitor device activity
  - Configure device parameters

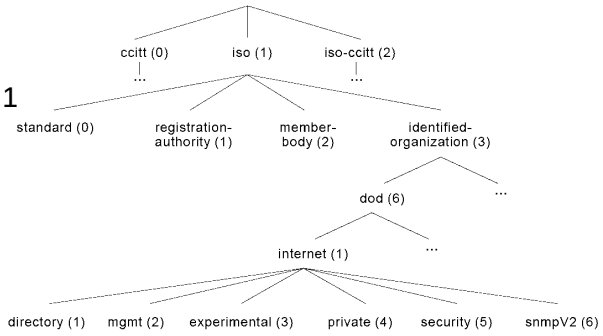
Role of SNMP

- Management Information Base (MIB)
  - Object-orient database record describing network entity
- Structure of Management Information (SMI)
  - Data definition language for MIB objects (subset of ASN.1)
- SNMP protocol
  - Manage entity by read (monitor) / write (control) of MIB information
  - Transaction mode
    - Manager request + agent response
  - Trap mode
    - Agent message on specified condition

Management Information Base (MIB)

MIB classification

- Tree hierarchy
- Internet MIB = 1.3.6.1



MIB

- Set of all MIB objects for managed device

MIB module

- Set of related MIB objects

Managed object

- Specific feature of managed device
- Uniquely identified by object ID
- Leaf level

Example — UDP MIB module

Object ID	Name	Object
1.3.6.1.2.1.7.1	UDPInDatagrams	Datagrams delivered at node
1.3.6.1.2.1.7.2	UDPNoPorts	Undeliverable datagrams — no server at port
1.3.6.1.2.1.7.3	UDInErrors	Undeliverable datagrams — other reasons
1.3.6.1.2.1.7.4	UDPOutDatagrams	Datagrams sent
1.3.6.1.2.1.7.5	udpTable	Entry for each port in use by app
1.3.6.1.2.1.7.5.1	udpEntry	
1.3.6.1.2.1.7.5.1.1	udpLocalAddress	udpTable entry — IP address
1.3.6.1.2.1.7.5.1.2	udpLocalPort	udpTable entry — Port

SNMP Message Types

Manager to agent

- GetRequest
  - Retrieve value of variable or list
- SetRequest
  - Change value of variable or list
- GetNextRequest
  - Discover available variables and their values
- GetBulkRequest
  - Multiple iterations of GetNextRequest

Agent to manager

- Response
  - Returns variable / acknowledgement for
    - GetRequest, SetRequest, GetNextRequest, GetBulkRequest
- Trap
  - Asynchronous notification of trapped event

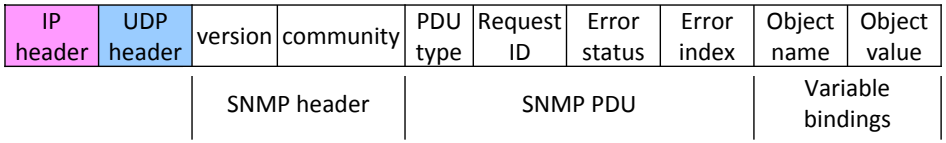
Manager to manager

- InformRequest
  - Asynchronous notification of trapped event

SNMP Packets

SNMP an application layer protocol

Messages send over UDP



Fields

- Version — SNMP version
- Community — environment for group of management systems
- PDU type — message type
- Request ID — associate requests / responses
- Error Status — indicates error type in response operation
- Error Index — associates error object instance
- Object name — name of object to set / monitor
- Object value — value of object

Wireless Networks

Some Basic Observations

Wireless

- Free-space electromagnetic transmission
- Radio, optical, IR

Differs from wired at infrastructure layers

- Physical transmission / reception
- Medium access issues

Application programmer usually ignores infrastructure

- Generally sees OS-provided network API (sockets)
- Special case — telephone / PDA applications

Special issues in wireless infrastructures

- Mobility management
- Broadcast infrastructure
- Channel reliability

Wireless Personal Area Network (wPAN)

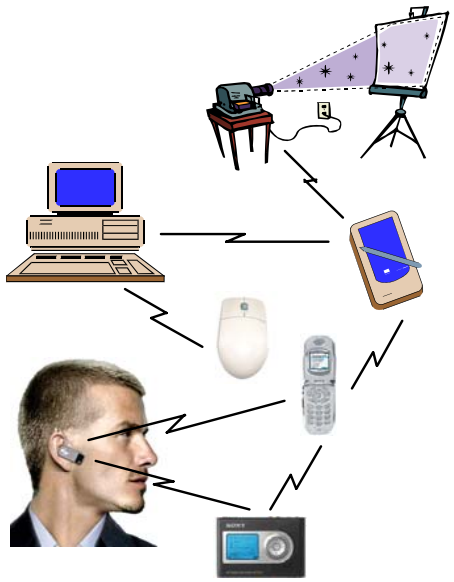
Short range broadcast transmission

Standard technologies

- Bluetooth
- Infrared Data Association (IrDA)
- Wireless USB

Applications

- Wireless computer peripherals
- Bluetooth earpiece
- Transfer interface for laptops, PDAs, cellphones
- Remote control



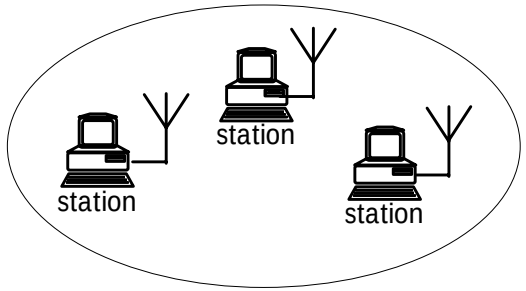
Wireless Local Area Networks

Wireless equivalent to local Ethernet

- Wireless network card
- Defines user authentication and encryption
- No external connection

Standard technologies

- IEEE 802.11 (WiFi)
- Bluetooth
- IrDA



Basic Wireless LAN

Wireless LAN with WAN Infrastructure

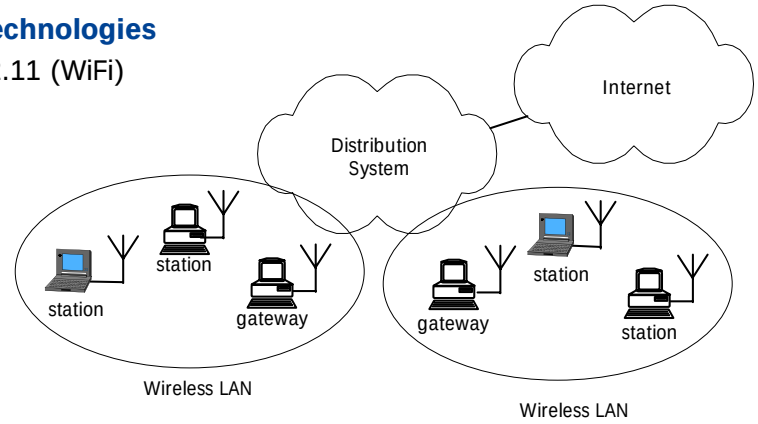
Extension of wireless LAN

Allows mobile access to external networks

Allows roaming between wLAN groups

Standard technologies

IEEE 802.11 (WiFi)



Wireless LAN Access to WAN

Cellular Telephony

Medium range broadcast with private channel assignment

Standard technologies

AMPS / TACS (1G)

GSM / d-AMPS (2G)

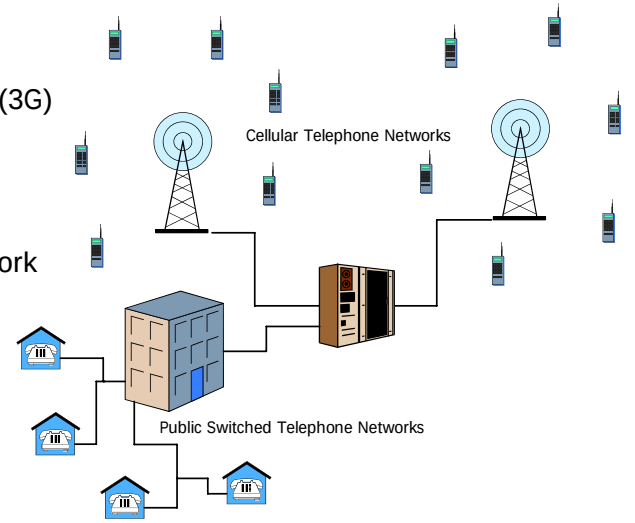
CDMA (2G)

UMTS / CDMA2000 (3G)

WCDMA (4G)

Application

Wireless voice network



Cellular Data Networks and Wireless IP

Wireless wide area data network (wWAN)

Data WAN over cellular telephone network

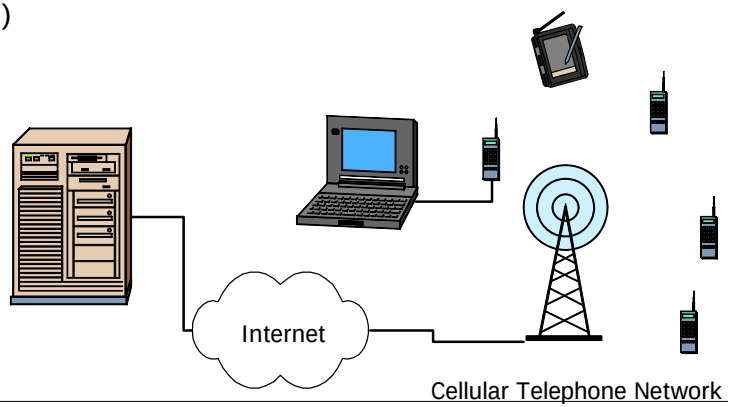
Standard technologies

CDPD (1.5G)

GPRS (2G)

EDGE (2.5G)

UMTS (3G)



Cellular Telephone Network

Wireless Application Protocol (WAP)

Protocol stack for mobile web interface

Adapts web for

- Phone screens
- PDA keypad

WML interactive scripting language



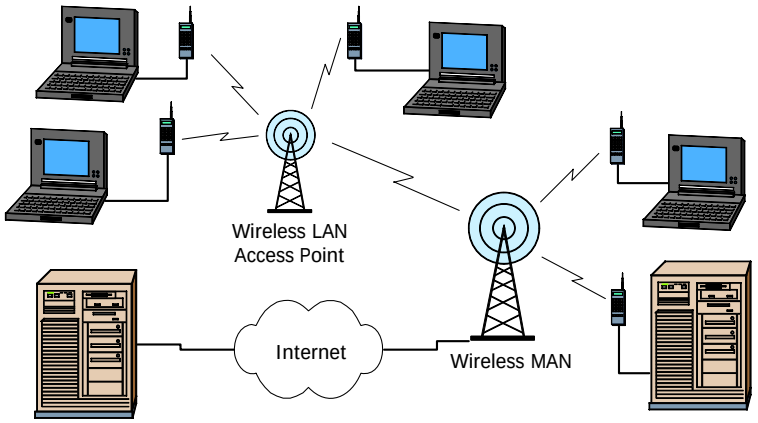
Wireless Metropolitan Area Network (wMAN)

Cellular broadband data access

WAN access via wireless network

Standard technologies

IEEE 802.16 (WiMAX)



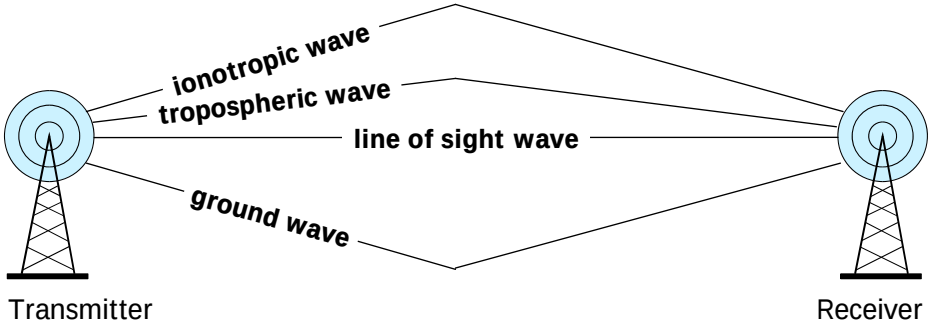
Radio Wave Propagation

Transmitter generates radio waves

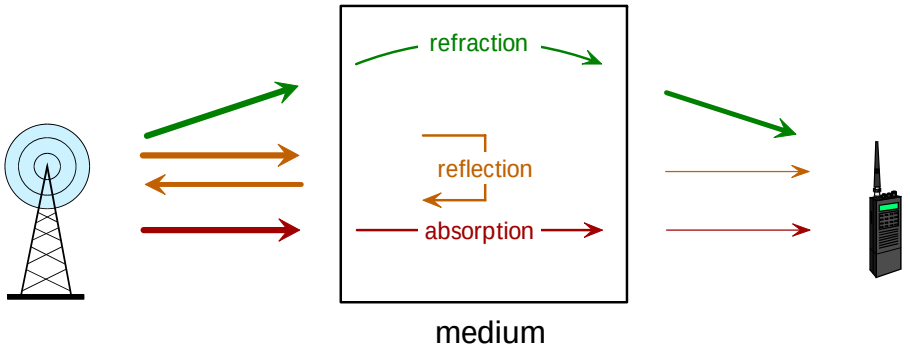
Waves propagate (spread out) through space

Part of radiated power may be obstructed

Part of radiated power is detected by receiver



Interference with Radio Signals



Multipath Fading

Obstacles reflect radio waves

Receiver gets signals from multiple paths

Time-to-arrive depends on path taken by signal

Receiver gets signals transmitted at different times

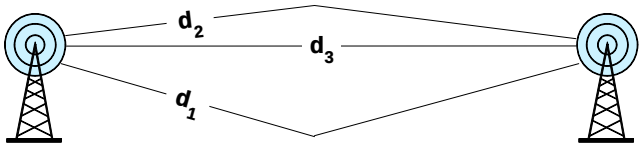
Example

Three signals sent at times $t_1 < t_2 < t_3$

Antenna receives all three signals at time t

- Signal 1 — sent first and followed longest path d_1
- Signal 2 — sent second and followed second longest path $d_2 < d_1$
- Signal 3 — sent last and followed shortest path $d_3 < d_2$

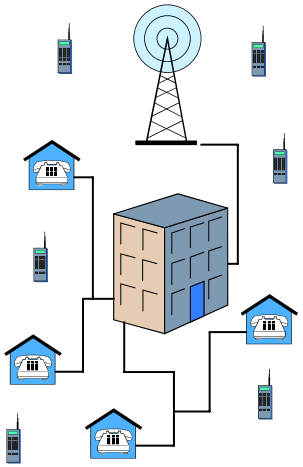
Sum of waves can cancel out signals



0G (1970) Mobile Phone System (MPS)

One central transceiver (transmitter/receiver)

- Mobile telephones communicate via central transceiver
- Transmit at high power for maximum distance
- System covers 65 to 80 km



Modulation is standard analog FM

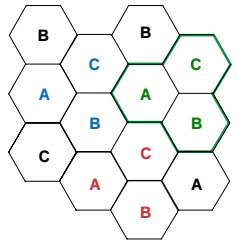
- Supports 12 simultaneous mobile phone calls
- If 12 channels busy, other calls are blocked

Requires 24 carrier frequencies

- 2 frequencies per phone:
  - Dedicated transmit frequency
  - Dedicated receive frequency

Cellular Concept

Divide coverage area into cells



In each cell

- Central cell transceiver serves all clients in cell
- Mobile Stations communicate via cell transceiver

Transmit at low power (just enough to cover a cell)

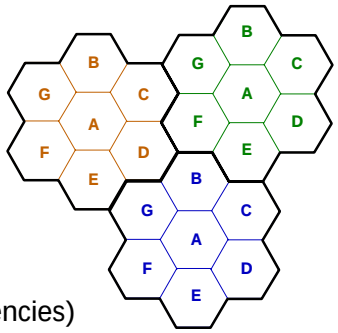
- Use same frequencies in many cells
- No interference between cells

Handoff

- Telephone can move from cell to cell during a call
- Requires cell-to-cell infrastructure and coordination

Cell Implementation

Divide region into clusters



Divide cluster into seven cells

- A, B, … , G

In each cell

- One central transceiver
- Many mobile stations (telephones)
- A frequency group (set of dedicated frequencies)

Each telephone has a private link with central transceiver

- Dedicated transmit frequency
- Dedicated receive frequency

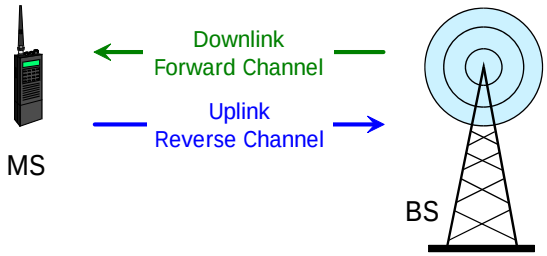
7 cell reuse

- Frequency group A assigned to every A cell
- Frequency group B to every B cell, ...
- At least two cells separate every pair of A cells, etc.

Transmission Directions

Downlink

- Base Station (BS) transmit frequency
- Mobile Station receive frequency
- Forward Channel



Uplink

- Mobile Station (MS) transmit frequency
- Base Station receive frequency
- Reverse Channel

Handoff

User moves between cells

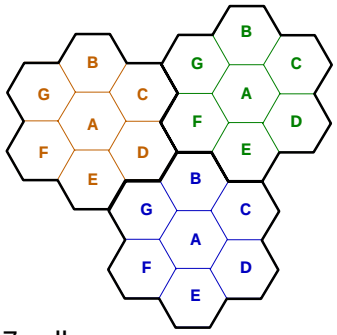
Hard Handoff

- Old cell transfers control to new cell
- Break-Before-Make sequence
  - Transceiver in old cell stops transmitting to user
  - Transceiver in new cell begins transmitting to user
- New BS assigns user frequency pair from its frequency group

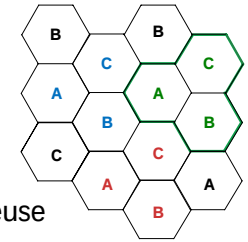
Soft Handoff

- Central transceiver coordinates with nearest cells
- Determines which transmitter is receiving strongest signal from user
- Make-Before-Break sequence
  - Transceiver in old cell transmitting to user
  - Transceiver in new cell begins transmitting to user
  - Transceiver in old cell stops transmitting to user

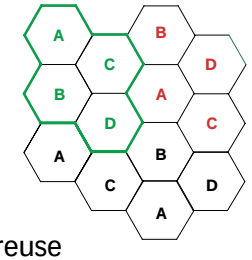
Reuse Patterns



7 cell reuse



3 cell reuse

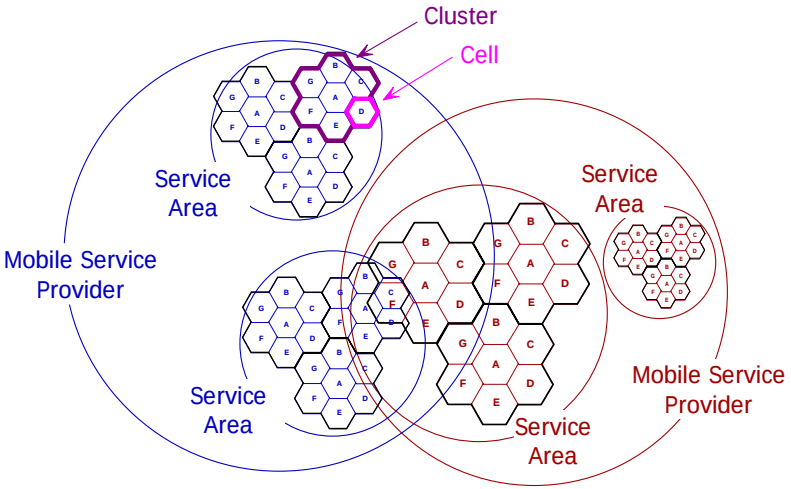


4 cell reuse

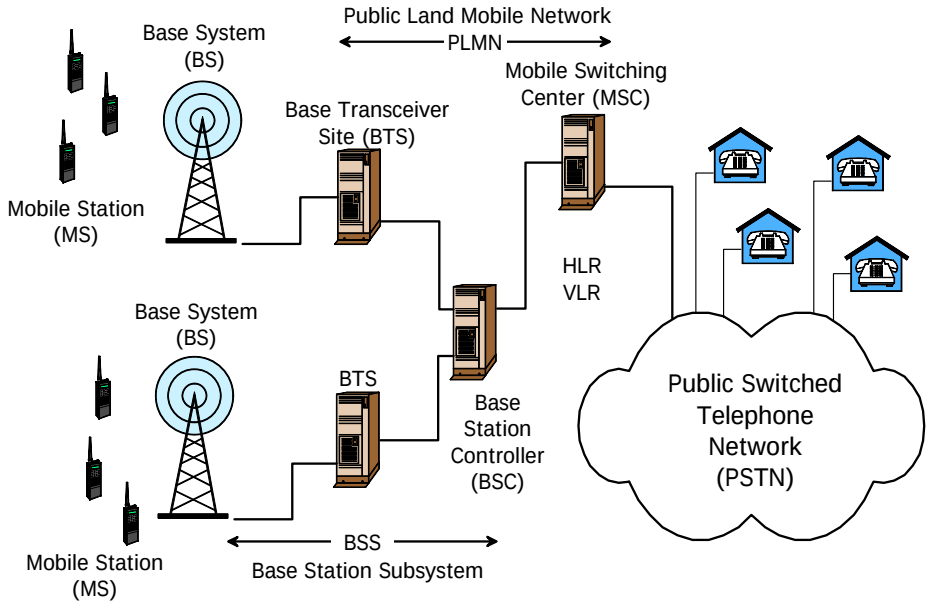
Mobile Network Switching Hierarchy

Mobile Service Provider

- Service Areas or Registration Areas
  - Clusters
    - Cells



The Cellular and Wired Telephone Network



Elements of GSM Mobile Network Hierarchy

Mobile Station (MS)

The telephone/terminal

Base Transceiver Site (BTS)

Fixed radio transmitter/receiver

Manages channels for with MSs in one cell

Base Station Controller (BSC)

Coordinates cluster of cells

Base Station Subsystem (BSS)

One BCS and all BTSs it controls

Mobile Switching Center (MSC)

Telephone Central Office for one Service Area

Handles local calls and Routes calls out of Service Area

Public Land Mobile Network (PLMN)

The wired portion of one Service Area (BTSs, BCSs, and MCS)

Mobility Service

Home Service Area

Service Area in which MS subscribes to cellular service

Home Subscriber

MS operating in its Home Service Area

Roamer

MS operating outside its Home Service Area

Handoff

Call control transfer when MS moves between cells in Service Area

Roaming

Call control transfer when MS moves between Service Areas

Problems of Mobility

MS must locate service provider access point

User must authenticate to service provider

Service provider must locate the MS

Provider must verify user's access rights

Home Location Register (HLR)

Located in MSC of Home Service Area

Maintains user's account information

Maintains location information for active MSs

Visitor Location Register (VLR)

Located in MSC for each Service Area

Cache of HLR data on active roamers

Registration Process

MS enters Service Area

Establishes low bit-rate control channel with service provider

MS requests service

BS allocates a frequency pair

MS reports to Mobile Switching Center (MSC)

Location, Status, and Identity

- Dedicated hardware ID code in phone
- Subscriber Identity Module (SIM) card identifies customer in GSM
- Mobile Station generates access code to network

Transmits code by public key encryption (PKE) algorithm

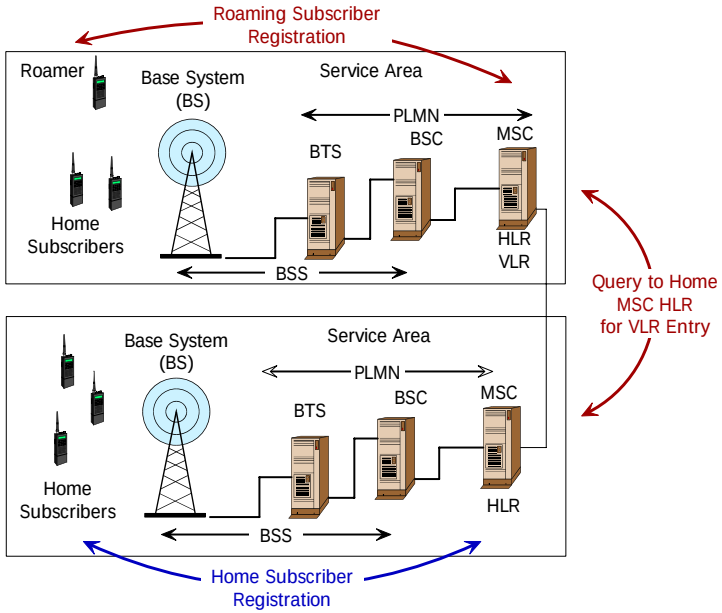
Mobile Switching Center (MSC)

Authenticates customer identity with HLR

For roaming subscriber, creates VLR entry

Updates Home Location Register (HLR) and billing database

Mobility Elements in the Cellular Network



1G — Advance Mobile Phone Systems (AMPS)

North American first generation analog system — IS-553

25 MHz transmission band per direction

Mobile Station (uplink): 825 - 849 MHz

Base Station (downlink): 870 - 895 MHz

Frequency Division Multiple Access (FDMA)

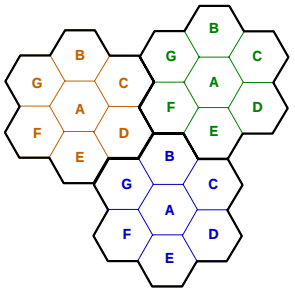
Divide band into 30 kHz RF voice channels

$$\frac{\text{25 MHz per cluster}}{\text{30 kHz per channel}} = 832 \text{ channels per cluster}$$

7 cell frequency reuse pattern (A, B, ..., G)

832 channels / 7 cells < 118 channels per cell

Typically 90 useful channels per cell



Second Generation Systems

2G Analog systems

Triple number of channels per cell

Motorola proprietary products

- Narrowband Advance Mobile Phone Systems (N-AMPS)
- Motorola Integrated Radio System (MIRS)

Time Division Multiple Access (TDMA)

Divide FDMA radio channel into time slots

MS transmits digitized voice in one time slot on one frequency

North American d-AMPS

European GSM

Code Division Multiplex Access (CDMA)

Create orthogonal binary digital transmission codes

MS transmits in one code on one frequency

GSM

Global System for Mobile Communications

European Union 2G digital cellular

Channelization

Divide band into 200 kHz RF channels

25 MHz per cluster / 200 kHz per channel = 125 channels per cluster

Digital transmission

Transmit 270.883 kbps in each 200 kHz radio channel

Voice and control modulation

- Gaussian minimum-shift keying (GMSK) — optimized FSK

Time Division Multiple Access (TDMA)

Divide each channel into 8 time slots

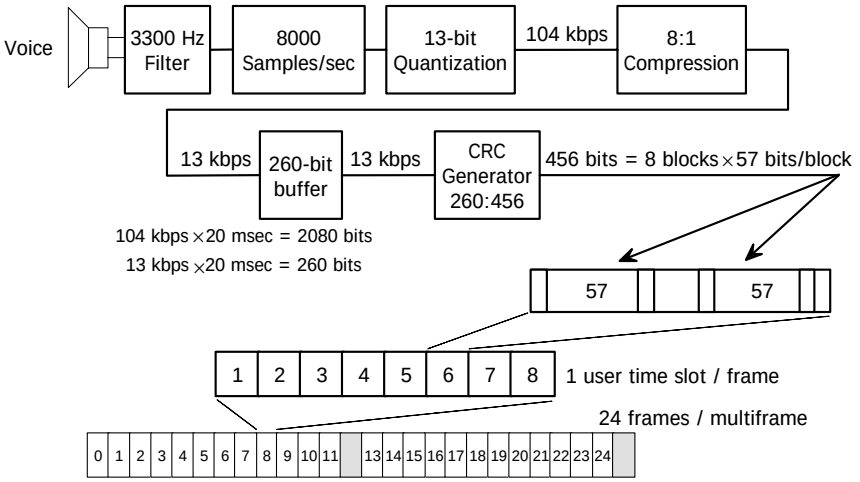
Allocate 1 time slot per user

- 270.883 kbps per channel / 8 users per channel = 33,086 bps per user

Standards

European Telecommunications Standards Institute (ETSI)

GSM Voice Transmission Summary



57 user bits per field × 2 fields per frame × 24 frames per multiframe = 2736 user bits per multiframe

2736 bits per multiframe / 120 ms per multiframe = 22.8 kbps

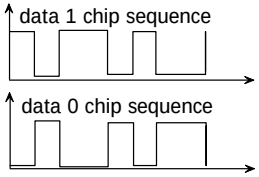
22.8 kbps / (456/260) = 13 kbps

Direct Sequence Spread Spectrum (DSSS)

Transmit data bit as chip sequence

- Chip
  - Shortest binary pulse on transmission channel
  - n-chip sequence is symbol for one data bit

Multiplies transmission rate



- User generates data at m bits per second
- Transmit n-chip sequence for every user bit
- Example
  - 1-sequence for data 1 = 10110100
  - 0-sequence for data 0 = 01001011
- Chip rate = m bps × n chips per bit = n × m chips per second (cps)

Receiver easily distinguishes 1-sequence from 0-sequence

- Bit error requires > n / 2 chip errors
- Works well in noisy environment

CDMA

Code Division Multiple Access

- Commercial system developed by Qualcomm
- Operates on AMPS frequencies

Channelization

- 25 MHz radio band per direction
- Divide band into 1.25 MHz RF channels
- 25 MHz per cluster / 1.25 MHz per channel = 20 channels per cluster

DSSS digital transmission

- Transmit 1.2288 Mcps in 1.25 MHz radio channel
- Voice and control modulation — QPSK

Code division

- Users transmit simultaneously using independent chip sequences
  - Orthogonal (Walsh) Codes / Pseudorandom noise (PN) codes
- Receiver separates channels by decoding chip sequences

Standards

- IS-95 — now called CDMAone

Orthogonal CDMA Codes

m-dimensional vector space with inner product

$$\mathbf{U} \cdot \mathbf{V} = \frac{1}{m} \sum_{i=1}^{m} U_i \times V_i$$

m orthonormal basis vectors

$S_i, i = 1, \ldots, m$

$\mathbf{T} = \sum_{i=1}^{m} t_i \times S_i$, with coefficient t_i for any vector $\mathbf{T}$

$$S_i \cdot S_j = m \times \delta_{ij} = \begin{cases} 0, i \neq j \\ m, i = j \end{cases}$$

$$t_i = S_i \cdot \mathbf{T} = S_i \cdot \frac{1}{m} \sum_{j=1}^{m} t_j \times S_j = \frac{1}{m} \sum_{j=1}^{m} t_j \times \left(S_i \cdot S_j \right) = \frac{1}{m} \sum_{j=1}^{m} t_j \times m\delta_{ij} = t_i$$

Code scheme

Basis vector S_i is code assigned to station i

Station i transmits $t_i \times S_i$ with coefficient

$$t_i = \begin{cases} -1, \text{data 0} \\ 0, \text{no transmission} \\ +1, \text{data 1} \end{cases}$$

Total transmission from all stations

$$\mathbf{T} = \sum_{i=1}^{m} t_i \times S_i$$

Example — 4-Chip CDMA

Code vectors for m = 4 stations

$$S_1 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \quad S_2 = \begin{bmatrix} -1 \\ +1 \\ +1 \\ -1 \end{bmatrix} \quad S_3 = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} \quad S_4 = \begin{bmatrix} -1 \\ +1 \\ -1 \\ +1 \end{bmatrix}$$

4-bit transmission levels (chips)

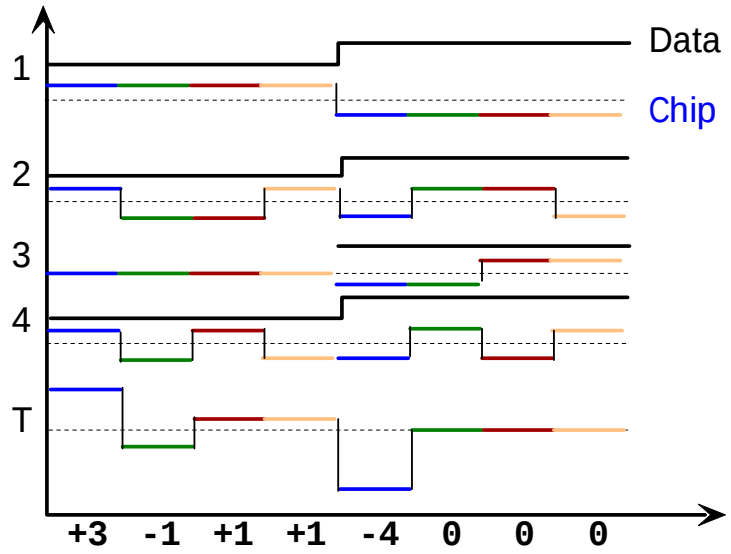
	Binary 1	Binary 0
Station 1	−1 −1 −1 −1	+1 +1 +1 +1
Station 2	−1 +1 +1 −1	+1 −1 −1 +1
Station 3	−1 −1 +1 +1	+1 +1 −1 −1
Station 4	−1 +1 −1 +1	+1 −1 +1 −1

Radio signal amplitudes added together

Example — 2-bit Transmission

Station 1	Data	0	1
	Signal	+1 +1 +1 +1	−1 −1 −1 −1
Station 2	Data	0	1
	Signal	+1 −1 −1 +1	−1 +1 +1 −1
Station 3	Data	no data	1
	Signal	0 0 0 0	−1 −1 +1 +1
Station 4	Data	0	1
	Signal	+1 −1 +1 −1	−1 +1 −1 +1
Total Transmission	Signal	+3 −1 +1 +1	−4 0 0 0

Example — 2-bit Transmission



Example — Decoding

Inner Product

$$\mathbf{U} \cdot \mathbf{V} = \frac{1}{4} \sum_{i=1}^{4} U_i V_i \qquad t_j = \mathrm{T} \cdot \mathrm{S}_j$$

First bit **T** = (+3, -1,+1,+1)	$t_1 = \frac{1}{4}(3,-1,+1,+1)\cdot(-1,-1,-1,-1) = \frac{1}{4}[-3+1-1-1] = -1 \Rightarrow 0$ $t_2 = \frac{1}{4}(3,-1,+1,+1)\cdot(-1,+1,+1,-1) = \frac{1}{4}[-3-1+1-1] = -1 \Rightarrow 0$ $t_3 = \frac{1}{4}(3,-1,+1,+1)\cdot(-1,-1,+1,+1) = \frac{1}{4}[-3+1+1+1] = 0 \Rightarrow$ no data $t_4 = \frac{1}{4}(3,-1,+1,+1)\cdot(-1,+1,-1,+1) = \frac{1}{4}[-3-1-1+1] = -1 \Rightarrow 0$
Second bit **T** = (-4,0,0,0)	$t_1 = \frac{1}{4}(-4,0,0,0)\cdot(-1,-1,-1,-1) = \frac{1}{4}[4] = +1 \Rightarrow 1$ $t_2 = \frac{1}{4}(-4,0,0,0)\cdot(-1,+1,+1,-1) = \frac{1}{4}[4] = +1 \Rightarrow 1$ $t_3 = \frac{1}{4}(-4,0,0,0)\cdot(-1,-1,+1,+1) = \frac{1}{4}[4] = +1 \Rightarrow 1$ $t_4 = \frac{1}{4}(-4,0,0,0)\cdot(-1,+1,-1,+1) = \frac{1}{4}[4] = +1 \Rightarrow 1$

Orthogonal Walsh Codes

Walsh 0

$W_0 = \boxed{1}$ $\quad W_0' = \boxed{-1}$

Walsh 1

$$W_1 = \begin{array}{|c|c|}\hline W_0 & W_0 \\ \hline W_0 & W_0' \\ \hline \end{array} = \begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & -1 \\ \hline \end{array}$$

Walsh 2

$$W_2 = \begin{array}{|c|c|}\hline W_1 & W_1 \\ \hline W_1 & W_1' \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 1 \\ \hline 1 & -1 & 1 & -1 \\ \hline 1 & 1 & -1 & -1 \\ \hline 1 & -1 & -1 & 1 \\ \hline \end{array} = \begin{array}{|c|}\hline S_1 \\ \hline S_4 \\ \hline S_3 \\ \hline S_2 \\ \hline \end{array}$$

Walsh 3

$$W_3 = \begin{array}{|c|c|}\hline W_2 & W_2 \\ \hline W_2 & W_2' \\ \hline \end{array}$$

Walsh N

$$W_N = \begin{array}{|c|c|}\hline W_{N-1} & W_{N-1} \\ \hline W_{N-1} & W_{N-1}' \\ \hline \end{array}$$

Walsh N is $2^N \times 2^N$ matrix

Pseudo-Noise (PN) Coding

Pseudorandom Bernoulli sequence of 1 or –1

Equivalent to sequence of m coin tosses

Nearly equal number of 1 and –1 in each code

By central limit theorem

$$P_{-1} = P(-1) = \frac{1}{2} + \delta \qquad P_{+1} = P(+1) = \frac{1}{2} - \delta \qquad |\delta| < \frac{1}{\sqrt{m}}$$

Codes are "nearly orthogonal"

For codes A and B with chip patterns $C_i^{(A)}$ and $C_i^{(B)}$

$$A = B \Rightarrow \frac{1}{m}\sum_{i=1}^{m} C_i^{(A)} \times C_i^{(B)} = \frac{1}{m}\sum_{i=1}^{m} [\pm 1]^2 = 1$$

$$A \neq B \Rightarrow \frac{1}{m}\sum_{i=1}^{m} C_i^{(A)} \times C_i^{(B)}$$

$$= \frac{1}{m}\sum_{i=1}^{m} \left[P_{+1} \times P_{+1} - P_{+1} \times P_{-1} - P_{-1} \times P_{+1} + P_{-1} \times P_{-1}\right] = 4\delta^2 < \frac{4}{m}$$

Channel Coding

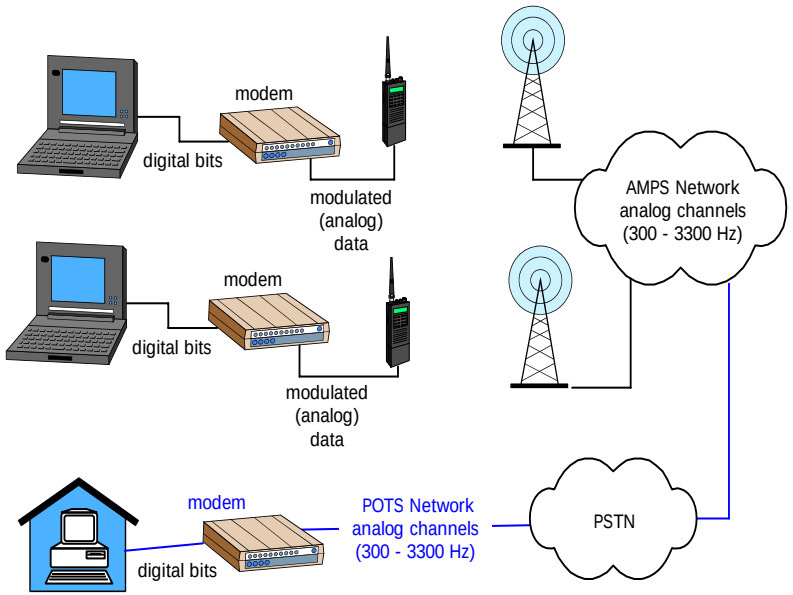
Forward channels

- 64 orthogonal Walsh codes to 64 users
  - Theoretically perfect separation between users
- All signals in same cell scrambled using PN sequence
  - Reduces interference between same Walsh code in neighboring cells
  - Short PN sequence uses cell ID as seed
  - Paging and traffic scrambled with long PN sequence before Walsh

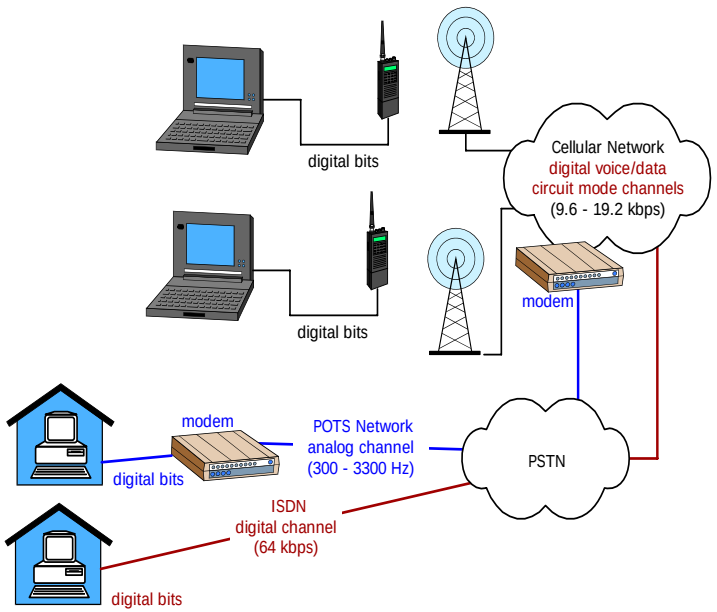
Reverse channels

- Orthogonal codes not applicable in uplink
  - Orthogonality requires time synchronization
  - MSs transmit asynchronously
- Long PN sequence
- Stream is scrambled using short PN sequence
- Carries cell ID

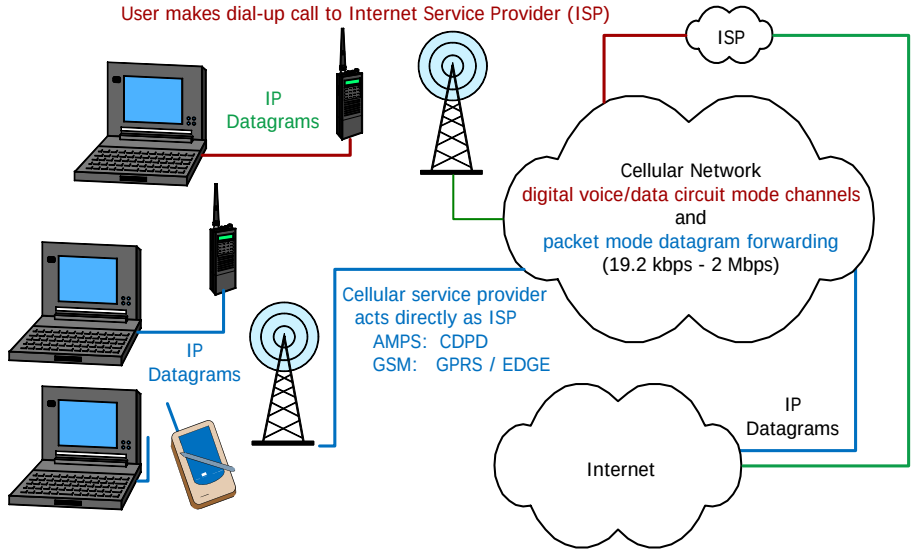
Data over AMPS



Cellular Circuit Mode Data Services



Cellular Packet Mode Data Services



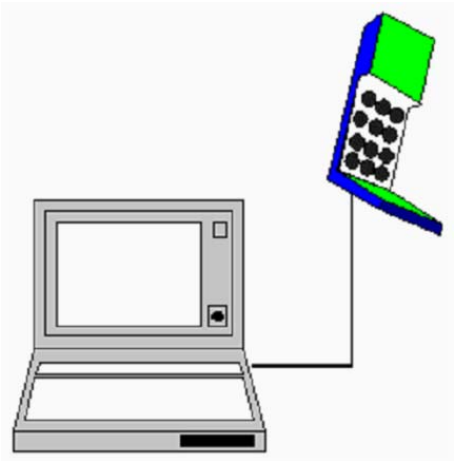
Cellular Data Terminals

Laptop with integrated cellular modem





Smartphone
with integrated cellular
modem



Laptop using cellular phone
as modem

General Packet Radio Service (GPRS)

Provides packet mode data access for GSM

- IP-based architecture
- Considered 2.5G enhancement

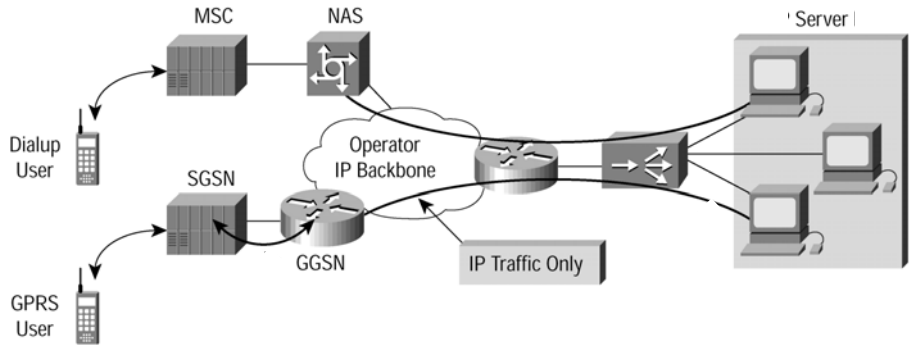
IP datagrams separated from circuit mode traffic at cluster

- Packet Control Unit (PCU)
  - Packet mode function in BSC to handle IP datagrams
- Circuit mode voice/data routed to MSC
  - Forwarded to other MSC or PSTN
- Packet mode data is routed to Serving GPRS Support Node (SGSN)
  - Forwarded to Internet or X.25 PSDN
  - PCU to SGSN runs IP over Frame Relay

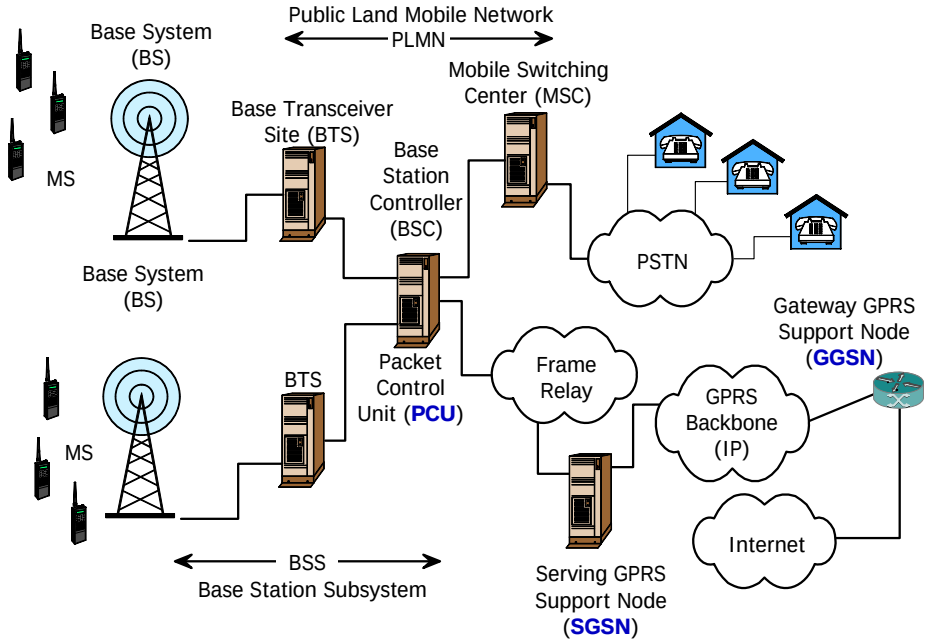
Mobility management

- Circuit mode traffic uses PSTN / PLMN routing
- Packet mode traffic uses IP routing

GSM Circuit Mode and GPRS Packet Mode Data



GPRS System Architecture



Short Message Service (SMS)

Transmission of short text messages

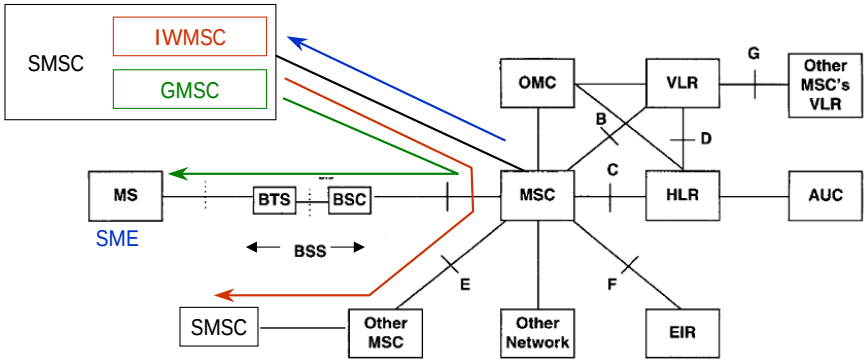
- Up to 160 bytes of alphanumeric data
  - 160 English ASCII characters
  - 16-bit unicode for non-English alphabets ⇒ 80 characters
- GSM permits message concatenation

Supported by GSM, d-AMPS, and CDMA

Out-of-band signaling

- SMS sent over signaling channel — not traffic channel
- SMS can be sent during a voice call
- Short messages do not create a heavy traffic load

SMS in GSM Architecture



SME: Short Messaging Entity

SMSC: Short Message Service Center

GMSC: Gateway Message Service Center

IWMSC: Interworking Message Service Center

SMS Nodes

Short Messaging Entity (SME)

- Any entity that can receive or send short messages
  - Fixed network element
  - Mobile Station
  - Another service center

Short Message Service Center (SMSC)

- Store and forwarding of SMS between SME and MS

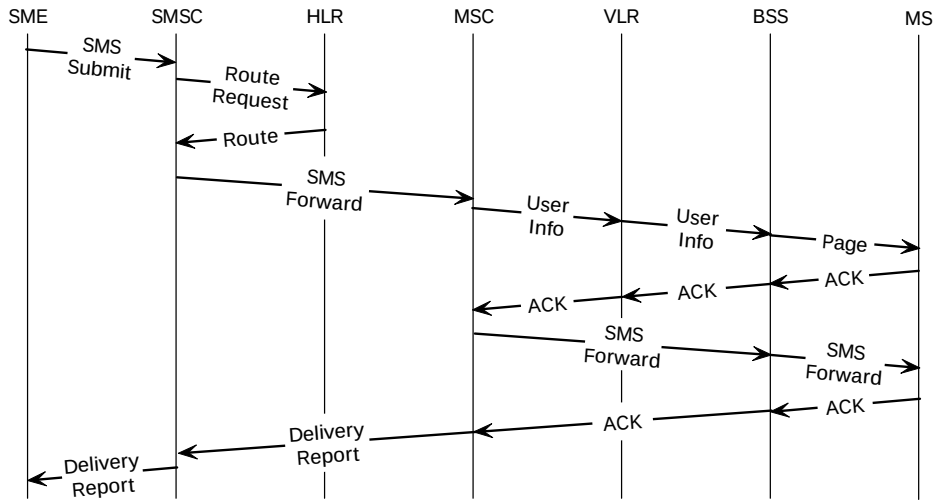
Gateway Message Service Center (GMSC)

- Receives SMS from SMSC
- Interrogates HLR for routing information
- Delivers SMS to MSC for destination SME

Interworking Message Service Center (IWMSC)

- Receives SMS from MSC
- Delivers SMS to appropriate SMSC for forwarding

SMS Delivery to MS



High Speed Circuit Switched Data (HSCSD)

Circuit Switched Data (CSD)

- 14.4 kbps circuit mode data connection in 2G GSM
- User data replaces digitized voice in 1 time slot

High Speed Circuit Switched Data (HSCSD)

- 2.5G enhancement
- Up to 8 slots (full user frame) allocated to one data channel
- Up to 115.2 kbps

Transparent data transmission

- User data stream can contain signaling to network
  - Allows dynamic reconfiguration of data connection (data rate, QoS)
- HSCSD data frames carry data sub-stream numbers
  - Maintains order of transmission over GSM

Non-transparent data transmission

- Only user data in data stream
  - No signaling or reconfiguration
- LLC functions performed by GSM protocols

Telecommunication Market Evolution — 1

Late 20th century

- Voice traffic >> data traffic
- Data traffic over analog / digital voice infrastructure
  - Access
    - V.35 / ADSL modem over telephone local loop
  - Backbone
    - Routers / switches on leased telco trunk lines
- Separate PSTN and cellular networks
  - Cellular backhaul
    - PLMN infrastructure on leased telco trunk lines
- Most profitable market sectors
  - PSTN
    - Long distance voice calls
  - Cellular
    - Air time

Telecommunication Market Evolution — 2

Early 21st century

Voice traffic < data traffic

Integrated networks — voice / data + fixed / mobile

IP over voice infrastructure → Voice over IP (VoIP)

Most profitable market sectors

PSTN

Leasing lines for data infrastructure

Cellular

Messaging, ring tones, multimedia services

Implications for Third Generation

System goals

Global mobility

Wide range of services

Voice telephony

Messaging + paging

Internet (WWW + email) access

Broadband data transport

Gateways among incompatible radio systems

More flexible PLMN routing infrastructure

Migration paths

TDMA d-AMPS → retirement

GSM → UMTS

More efficient radio spectrum utilization (CDMA replaces TDMA)

CDMA → cdma2000

More efficient radio spectrum utilization (higher capacity CDMA)

3G Standardization

Internal Mobile Telecommunications (IMT-2000)

International Telecommunications Union (ITU) standards for 3G

Defines multiple competing (incompatible) systems

Universal Mobile Telecommunications System (UMTS)

GSM/GPRS replacement using CDMA radio interface

Third Generation Partnership Project (3GPP)

Consortium of manufacturers (www.3gpp.com)

CDMA 2000

CDMA replacement using cdma2000 radio interface

Third Generation Partnership Project 2 (3GPP2)

Consortium of manufacturers (www.3gpp2.org)

WiMAX

Broadband wireless data access using cellular technology

WiMAX Forum

Consortium of manufacturers (www.wimaxforum.org)

UMTS

Physical layer

User access: GSM TDMA → W-CDMA or TD-CDMA

Similar to cdmaOne and cdma2000 but not compatible

Different frequency bands

Different pseudorandom noise (PN) coding scheme

Circuit mode data rates up to 1.92 Mbps

144 kbps and 384 kbps on high-utilization systems

New PLMN node definitions

BSS (base station subsystem) → RNS (radio network system)

BSC (base station controller) → RNC (radio network controller)

BTS (base transceiver system) → Node B

Protocols

New internal network operations

Frame Relay in backbone infrastructure → ATM

High Speed Downlink Packet Access (HSDPA)

Higher data rates for packet data

- Downlink speeds of 1.8, 3.6, 7.2, 14.0 Mbps

HS-DSCH simplified for fast packet data

- Power control and variable chip rate eliminated

Hybrid automatic repeat-request (HARQ)

- LLC layer added between PHY and MAC (not in RLC)
- Incremental redundancy
  - Corrupted packets not discarded
  - Retransmitted packets combined until error-free packet assembled
  - Faster than waiting for uncorrupted retransmitted packet

Fast packet scheduling

- 2 ms scheduling granularity (instead of 10 ms)
- Transmission scheduled to UEs reporting highest power levels

Adaptive Modulation and Coding (AMC)

- Modulation scheme and code rate depend on channel quality

cdma2000

Replacement for IS-95 CDMA (now called cdmaOne)

- Same radio frequencies
- Non-compatible pseudorandom noise (PN) coding scheme
- Higher data rates using improved modulation techniques
- Packet mode data — Mobile IP on voice network (like CDPD)

Evolutionary change from cmdaOne

- Multiple upgrade paths
- Operates in same radio frequencies
  - No new licensing costs for additional radio spectrum
- Backward compatible with cmdaOne
- Minimum risk to existing operators

Third Generation Partnership Project 2 (3GPP2)

- Consortium of manufacturers (www.3gpp2.org)

Standard

- IS-2000

IS-2000 Spreading Rates

1xRTT

- Same 1.25 MHz radio channel as IS-95
- Double IS-95 chip rate → 128 chips per bit
- Double users → 128 users per channel
- RF compatible with IS-95 in same cell
  - Uses codes orthogonal to IS-95 codes

1xEV-DO (data only)

- Physical layer different from 1xRTT
- Higher data rates (3.1 Mbps forward / 1.8 Mbps reverse)
- No increase in voice capacity

3x (3xRTT)

- Uses 3.75-MHz radio channels
- Direct Spread (DS) — one 3.75-MHz RF carrier
- Multicarrier (MC) — spreads data among 3 IS-95 1.25 MHz channels

Next Generation Networks (NGN)

ITU initiative for long-term network planning

- Standardizes current view of technology convergence

NGN definition

- Packet-based network
- Provide telecommunication services
- Use multiple broadband QoS-enabled transport technologies
- Service functions independent of transport technology
- Enables unfettered user choice of access to
  - Networks
  - Competing service providers and/or services
- Supports generalized **MOBILITY**
- Allow consistent and ubiquitous provision of services to users

From ITU-T Recommendation Y.2001 (12/2004)

NGN in the Marketplace

Mobility

- Basic feature of contemporary workflow
- Important source of profit for telecommunications industry

Convergence

- Workflow ⇒ universal access to services through any networks
- Multiple incompatible networks ⇒ market share + profits

Where do technologies converge?

- Most systems can interface service to infrastructure with TCP/IP
- Inherently digital services → internet
- Inherently analog services → A/D + compression → internet

NGN generally means all-IP network

- All services defined to work over IP
- All infrastructures defined to work below IP
- Problem — QoS, reliability, mobility not natural in IP

NGN Visions

Migration of all existing voice networks

- Most voice infrastructure is still hierarchical
  - DS-0 circuit switching
  - High speed trunk lines organized in tree topology among ESSs
  - Isochronous circuit mode operation natural for voice traffic
- NGN requires transforming voice networks to VoIP

Migration of local access from voice to DSL

- Single fast digital interface to doorstep
- Fiber to the door an expensive dream

Migration to flexible metropolitan area networks (MAN)

- "Carrier Ethernet" and cellular broadband (WiMAX) in urban areas

Improvement of QoS in IP networks

- Multiprotocol Label Switching (MPLS)
- Session Initiation Protocol (SIP)

4G Cellular

Initial planning for 4th generation cellular systems

- ITU working group planning IMT-2000 → IMT-Advanced
- Conceived as network supporting mobility — not telephones + data
- Convergence with NGN

4G objectives

- Higher network capacity than 3G
- Spectral efficiency (high bps / Hz and bps / Hz /site)
- 100 Mbps for moving client and 1 Gbps for stationary client
- 100 Mbps between any two points in world
- Smooth handoff across heterogeneous networks
- Global roaming across multiple networks
- QoS for multimedia support — audio, HDTV, etc
- Interoperability with existing wireless standards
- All IPv6 packet switched network — eliminate circuit mode entirely

Long Term Evolution (LTE)

3G standard

- Upgrade of 3G UMTS
  - Improved radio interface
  - Downlink < 300 Mbps
  - Uplink < 75 Mbit/s
- Marketed as 4G
  - Does not conform to 4G standards
  - Upgrade path while waiting for 4G

Flat IP-based network

- Evolved Packet Core (EPC) replaces GPRS
- Voice calls handled Voice over LTE (VoLTE)
  - Form of Voice over IP (VoIP)
  - Routed over EPC packet switched network
  - No separate circuit switched network for voice

IEEE 802.11

Specified by IEEE 802 Committee for LAN/MAN

Standards for Infrastructure Layers (OSI 1 and 2)

Extends Ethernet for wireless physical layer

Data rates

802.11 (1997) specified 1 or 2 Mbps (legacy)
802.11a (1999) specifies 6 to 54 Mbps
802.11b (1999) 5.5 Mbps and 11 Mbps (WiFi)
802.11g (2003) 54 Mbps (WiFi)
802.11n (2009) specifies up to 300 Mbps

Wireless Issues in LANs

Mobility

Addressable unit is a mobile station (STA)
Dynamic topologies
Medium boundaries are neither absolute nor visible
Lack full connectivity — STAs may be "hidden"

Reliability

Medium less reliable than wired PHY
Time-varying and asymmetric propagation

Power management

IEEE 802.11 wLAN Architectures

Ad Hoc Mode

Simple Peer-To-Peer Mode (STA-to-STA)
Limited to local communication
- No WAN access or hand-off

Authentication and Registration
- Permitted but not required

Infrastructure Mode

Basic topology
- Permits forwarding to wired LANs and WANs
- All communication via central Access Point (AP)
- Permits Authentication
- Requires Registration

Extended topology
- Permits hand-off among WLAN segments

Ad Hoc Mode (Peer-To-Peer Mode)

Independent Basic Service Set (IBSS)

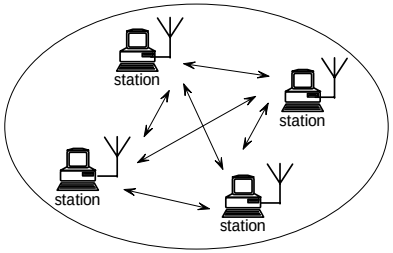
Any set of 802.11 STAs (wireless stations)
No connection to a wired network

Simple unmediated communication

STAs communicate directly with one another
Useful for quick set up
Authentication or Registration not required

Multiple IBSSs are independent

No bridging
No hand-off



Independent Basic Service Set

Infrastructure Mode

Basic Service Set (BSS)

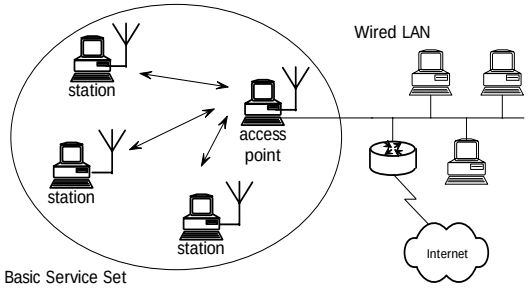
A set of wireless end stations (STA)

An Access Point (AP)

- Connected to the wired network infrastructure
- Acts as base station for the wireless network
- All traffic flows through AP by Contention or Polling (CFP)

Stations must Associate with AP

- Authentication
- Registration



Infrastructure Mode

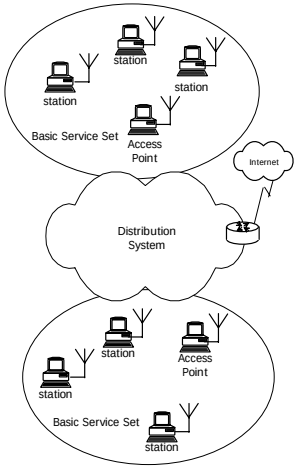
Extended Service Set (ESS)

Two or more BSSs

- Form single subnetwork (broadcast domain)
- Looks like one large BSS to LLC layer
- One Access Point (AP) in each BSS

BSSs connected via Distribution System (DS)

- DS is backbone network
- DS performs MAC-level transport of MAC SDUs
- DS implementation not specified in 802.11



Portal

Software gateway function in AP

Bridges BSS to any non-802.11 DS protocol

DS services permit handoff

Station moving from one BSS to another

Requires coordination between APs

802.11 Protocol Layers

PHY Dependent Sublayer

Transmission type

Modulation scheme

Data transmission rates

Physical Layer Convergence Sublayer

PHY medium dependent

Specifies header for PHY Dependent Sublayer

MAC layer

Medium access

Addressing

Procedures

Data Link Layer	LLC	802.2	LLC frame for SEQ/ACK/Control
	Bridging		Exchange of 802.2 PDUs
	MAC	802.11	CSMA/CA, MACA, CFP
Physical Layer	Convergence		PHY-Dependent Convergence Sublayer
	PHY		FHSS, DSSS, IR, Data rates

MAC Layer Issues

Channel Allocation Method

Contention (distributed control)

Round Robin (deterministic)

Polling (centralized control)

Collision Detection and Error Detection

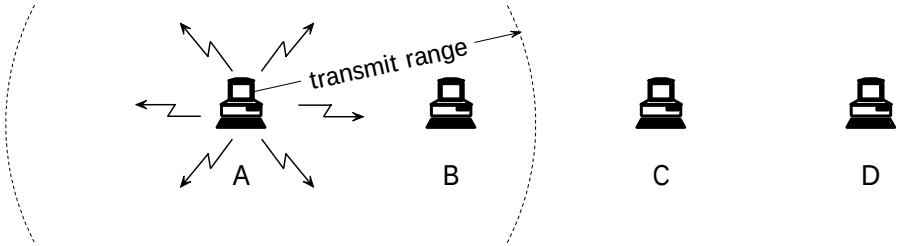
Fragmentation

Addressing

Control and Management Frames

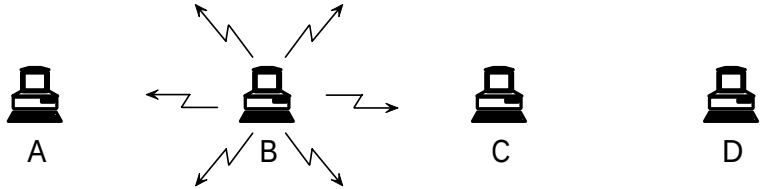
Hidden Node Problem

A transmits to B
C cannot receive from A — out of range
C is may interfere with A's transmission



Exposed Node Problem

B transmits to A
C receives B's transmission and is not free to start
C delays its transmission to D unnecessarily



CSMA with Collision Avoidance (CSMA/CA)

Carrier Sense Multiple Access (CSMA)

- Stations listen for transmissions
- Do not transmit if carrier is detected
- Collision detection not possible
  - Hidden node problem
  - Antenna cannot receive while transmitter active

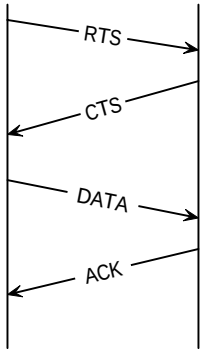
Collision Avoidance (CA)

- Non-persistent access
- Random backoff

Multiple Access with Collision Avoidance (MACA)

Channel set-up before data transmission

- RTS — Request To Send
- CTS — Clear To Send
- ACK — Acknowledgment of error-free transmission



Net Allocation Vector (NAV)

- Transmitted in RTS
- Predicted data transmission time

Improves behavior of Hidden Nodes and Exposed Nodes

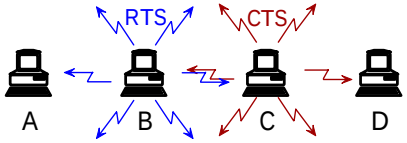
Multiple Access with Collision Avoidance (MACA)

B sends 30-byte RTS (request to send) packet to C

- Includes a NAV for the data to be sent
- All stations in B's range hear RTS

C responds with CTS (clear to send) packet to B

- Echoes NAV
- All stations in C's range hear CTS



B in range of A but not D

- A receives RTS but not CTS
- A can transmit without interfering with B's destination

C in range of B but not A

- D receives CTS but not RTS
- D waits data transmit time before transmitting

Station Services (SS) — 1

Privacy in wired LAN

- Design assumes physical closure
- Illegal access requires physical connection

Privacy in wLAN

- Any 802.11 receiver in range can receive all frames
- Wired Equivalent Privacy (WEP) algorithm
  - Shared key encryption
  - Not secure
  - No worse than wire

Station Services (SS) — 2

Authentication

- Station provides proof of identity to AP or STA
- Method not specified in 802.11
- Required before Association

Deauthentication

- Terminate authentication of another station
- Deauthentication invokes Disassociation

MAC Service Data Unit (MSDU) Delivery

- End-to-end delivery of LLC packets
- LLC packets (PDUs) are the SDUs of the MAC

Distribution System Services (DSS) — 1

Association

- Station associates with one AP
- Association provides STA/AP mapping to the DS
- DS forwards to STA via unique AP association

Reassociation

- Station moves from BSS to New BSS
- Station associates with New AP in New BSS

Disassociation

- New AP informs Old AP of Reassociation
- Old AP terminates old association
- APs may also disassociate all STAs (for maintenance)

Distribution System Services (DSS) — 2

Distribution

Delivery of packets to stations through DS

STA sends to source AP

Logically invokes DSS Distribution Service

DS passes frame to Destination AP

Destination AP passes frame to Destination STA

Integration

Portal services provided by DS

Source AP sends frame to Portal

Portal forwards to foreign (not 802.11) network

MAC Layer Address Fields

4 Address Fields

5 possible MAC entities:

BSS Identification Number (BSSID)

Source Address (SA)

Station which initiated the message

Destination Address (DA)

Final destination for the message

Transmitting Station Address (TA)

Station sending the message on this hop

Receiving Station Address (RA)

Destination for the message on this hop

Address Field Definitions

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	DA	SA	BSSID	—
0	1	DA	BSSID	SA	—
1	0	BSSID	SA	DA	—
1	1	RA	TA	DA	SA

Address 1	Immediate destination address
Address 2	Immediate source address
Address 3	Final destination or source when DS performs distribution
Address 4	Source address for DS to DS messages (802.11 is also DS)

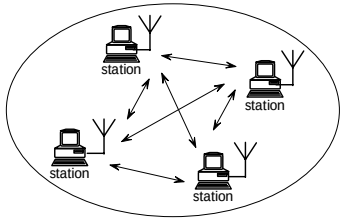
Addressing in an IBSS

To DS	From DS	Address 1	Address 2	Address 3
0	0	DA	SA	BSSID

Independent Basic Service Set (IBSS)

No Access Point (AP) and no DS

Fields To DS and From DS are 0



Independent Basic Service Set

Address 1	Immediate destination address (DA)
Address 2	Immediate source address (SA)
Address 3	BSSID Identifies Ad Hoc network Prevents message from reaching outside IBSS

Data Addressing in a BSS

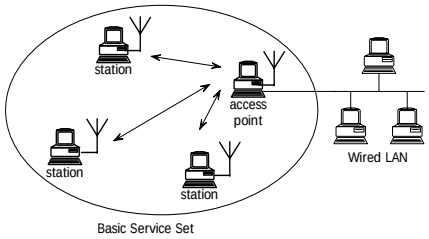
To DS	From DS	Address 1	Address 2	Address 3
0	1	DA	BSSID	SA
1	0	BSSID	SA	DA

Basic Service Set (BSS)

All transmissions are sent To/From Access Point

To/From DS actually means To/From AP

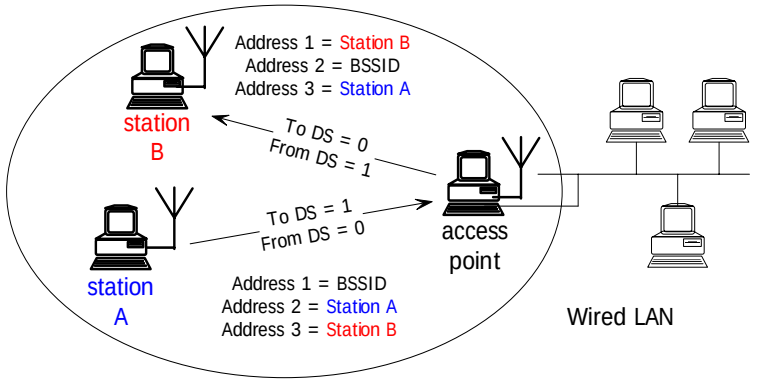
Address 1	Immediate destination address (DA)
Address 2	Immediate source address (SA)
Address 3	Final Destination or Source



BSS Addressing Example

To DS	From DS	Address 1	Address 2	Address 3
0	1	DA	BSSID	SA
1	0	BSSID	SA	DA

Station A sends message to Station B via AP (BSSID)



Control and Management Addressing in a BSS

To DS	From DS	Address 1	Address 2	Address 3
0	0	DA	SA	BSSID

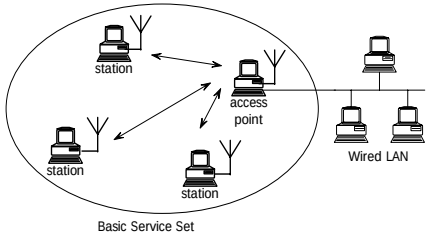
Control and Management messages in a BSS:

Only involve stations in the BSS and the AP

Are sent with To DS = From DS = 0

Either the Source or the Destination will be the AP (BSSID)

Address 3 in included as an error check



Addressing in an ESS

To DS	From DS	Address 1	Address 2	Address 3
0	1	DA	BSSID	SA
1	0	BSSID	SA	DA

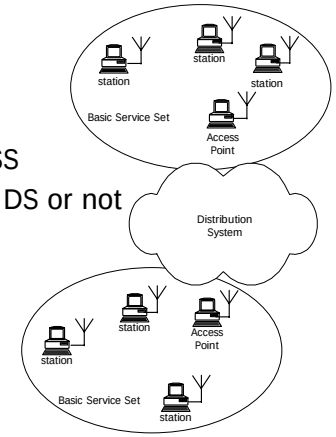
Extended Service Set (ESS)

All transmissions are sent via an AP

To the stations, entire ESS looks like one BSS

Stations do not know if message passes via DS or not

Address 1	Immediate destination address (DA)
Address 2	Immediate source address (SA)
Address 3	Final Destination or Source



ESS Addressing Example

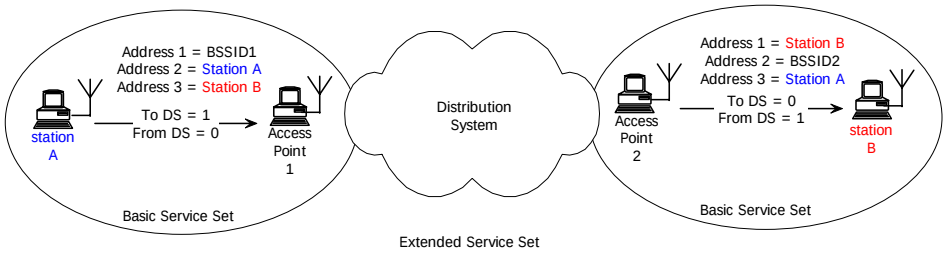
To DS	From DS	Address 1	Address 2	Address 3
0	1	DA	BSSID	SA
1	0	BSSID	SA	DA

Station A sends message to Station B via

AP1 (BSSID1) → DS → AP2 (BSSID2)

DS must forward Data, Sequence, SA, and DA

By some legal means



WEP Encryption/Decryption Procedure

Plaintext

MAC Layer PDU (MPDU)

CRC-32 Frame Check Sequence (FCS) on MPDU

Key Sequence

Generated from Secret Key and Initialization Vector (IV)

Key length is MPDU length + 4

Transmission

Encrypted Plaintext

Unencrypted Initialization Vector (IV)

Receiver

Generates Key Sequence from Secret Key and IV

Deciphers Plaintext and checks FCS for errors

WEP Encryption Algorithm

Secret Key distributed by some background process

Initialization Vector (IV) 24-bit suffix generated by transmitter

IV may be changed as frequently as every MPDU

IV transmitted unencrypted with message to receiver

Receiver needs IV to decrypt

IV provides no information about secret key

Seed

64-bit concatenation: Secret Key ## IV

Seed input to Pseudo-Random Number Generator (PRNG)

Key Sequence k

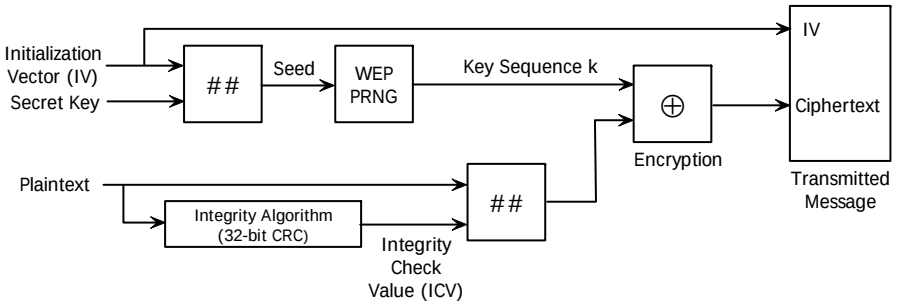
Pseudo-Random Number generated by PRNG using seed

Integrity Check Value (ICV)

32-bit CRC on MPDU

Plaintext (MPDU ## ICV) encrypted with Key Sequence

WEP Encryption Algorithm



WEP Decryption Algorithm

Key Sequence generated from IV and Secret Key

Decryption

- Key Sequence applied to Ciphertext
- Plaintext includes MPDU and ICV

Integrity check performed on Plaintext

- On error in received MPDU
  - Error indication is sent to MAC management
  - Data not passed to LLC

Problems with WEP Algorithm

XOR encryption is not very strong

Secret Key is too easy to deduce

- Part of MPDU may be easy to guess
  - Example: IP header fields
  - Can find k from P and C

Encryption strength

- Depends on lifetime of Initialization Vector (IV)
- Best privacy when IV is changed for every MPDU

More Problems with WEP

AP beacons

- Announce service availability
- Can be found by unauthorized listeners

WEP not always implemented

Weak encryption

- 40-bit secret key
- Simple XOR of key with plaintext

Weak authentication

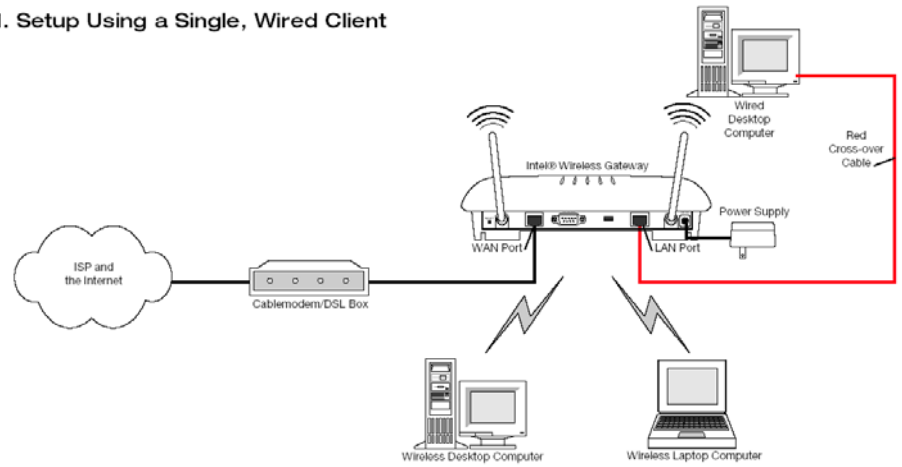
- STA requests service
- AP sends random number
- STA returns number encrypted with key (password)

Authentication password is used as encryption key

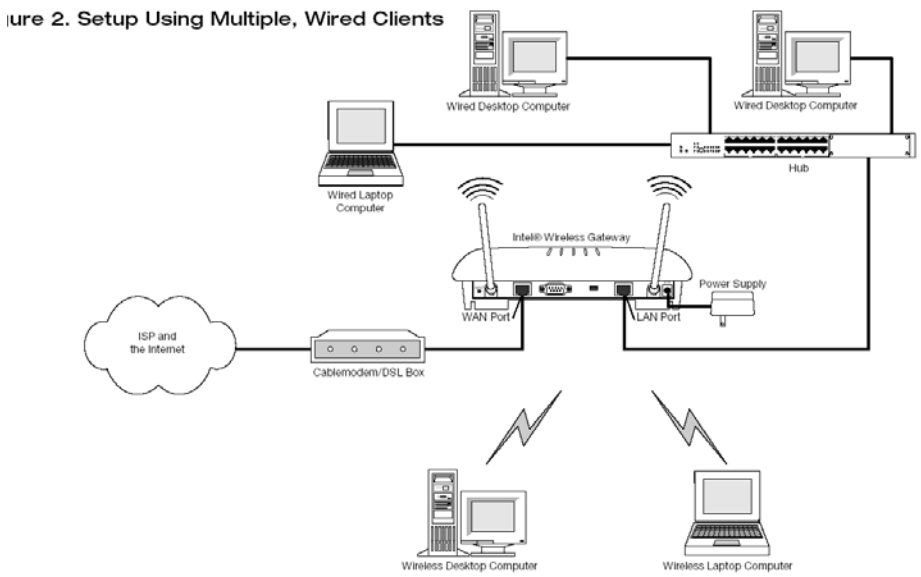
- Eavesdropper can learn key from plaintext and encrypted number

Infrastructure Network Configurations — 1

1. Setup Using a Single, Wired Client



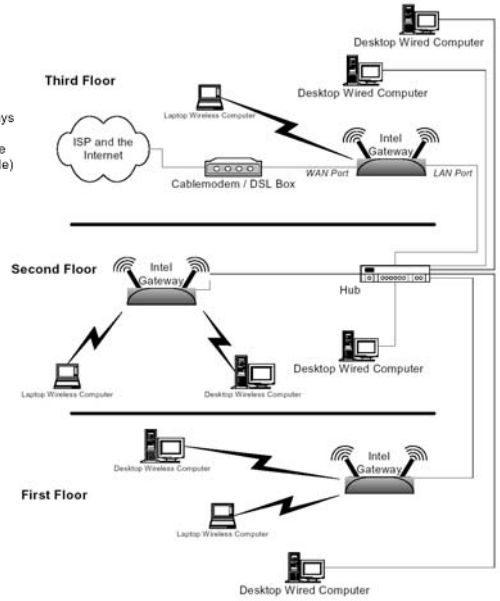
Infrastructure Network Configurations — 2



Infrastructure Network Configurations — 3

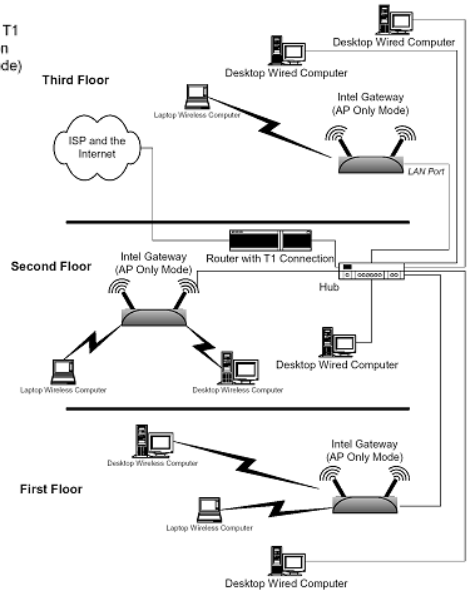
Installation example with multiple gateways

(one Intel Gateway in gateway/AP mode and other Intel Gateways in AP only mode)



Infrastructure Network Configurations — 4

Installation example with multiple Intel Gateways and T1 or other Internet connection (all Gateway's in AP only mode)



The Bluetooth Vision

Universal wireless connectivity

- Replace existing cables with radio
- Connect systems that have been separate

Ubiquitous computing environment

- Intelligent devices performing distributed services
- Redesign hardware as object-oriented

Unconscious connectivity paradigm

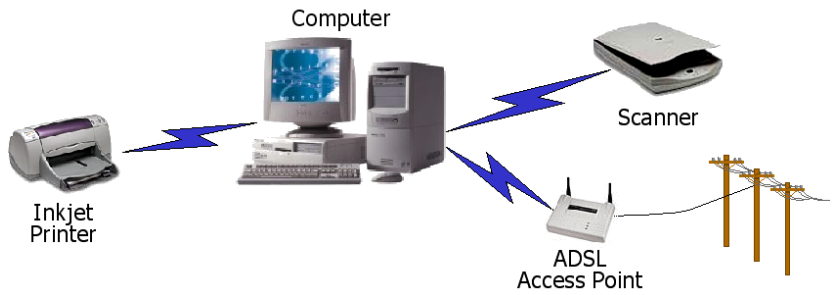
- Devices interconnect automatically
- Minimal user intervention

Wireless Personal Area Network (wPAN)

- Small networks formed dynamically
- Wireless internetworking among wPANs

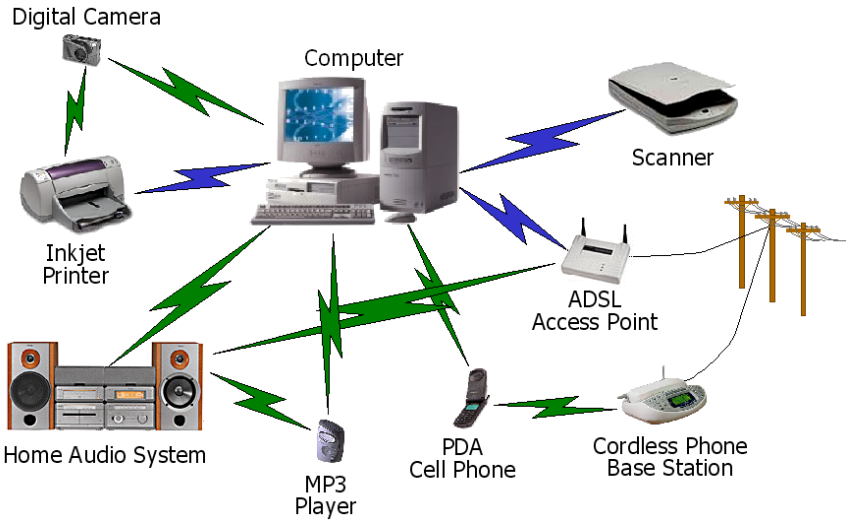
Universal Wireless Connectivity

Replace existing cables with radio



Universal Wireless Connectivity

Connect systems that have been separate



Ubiquitous Computing Environment

Intelligence is local and communication is universal

Bluetooth devices

- Search for other compatible devices
- Share information about services they provide
- Exchange commonly defined data objects

Service provision is distributed over wPAN

Integrated automation of

- Central servers
- Information repositories
- Sensors
- Actuators

Unconscious Connectivity Paradigm

Connectivity is a problem for the user

- Inconvenient to establish connections manually
- Available devices change frequently
- Users may not remember how to connect

Devices connect automatically and dynamically

- Devices discover one another
- Devices determine when and why to connect
- Users do not need to remember how to connect

Example of The Vision

User

Enters hotel lobby

PDA in user's pocket

Connects to hotel reservations system for check in

Receives key code for door

Displays room number

Alerts laptop in suitcase to log onto hotel email server

User's Laptop

Downloads messages while user waits for elevator

User's PDA

Unlocks door of hotel room

User's laptop

Uploads music to audio system

User's PDA

Orders room service from menu user prepared on airplane

Example of a Real Product

Three-in-One Telephone

Automatic network selection by environment:

Intercom at home or in office

PSTN phone when a PSTN access point is available

Cellular mobile phone otherwise

How is Bluetooth Different?

In cellular and wLAN systems:

Base Stations and Mobile Stations are clearly distinct

Base Stations handle services

Channel access

Channel allocation

Traffic control

Interference problems

Mobile Stations are relatively simple clients

In Ad Hoc Bluetooth networks:

Communication is peer to peer

No central controller

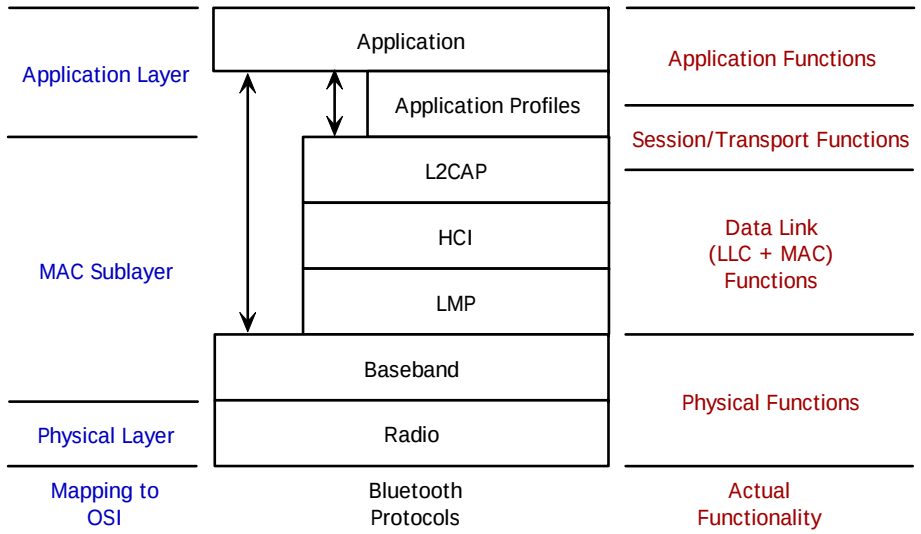
Devices in area self-organize in a shared channel

May be many Bluetooth devices in region

Only a few need to communicate

Mutual coordination is complex

Protocol Layers



Protocol Overview

Application Layer	User application programs
Application Profiles	User application support protocols: FTP, TCP, WAP, PPP, telephony, USB, Serial Port, etc
Logical Link Control and Adaptation Protocol (L2CAP)	Channel management (socket-type interface), Segmentation and Reassembly, QoS (speed, reliability, delay)
Host Controller Interface (HCI)	Supports standard I/O hardware standards (when Bluetooth device is external to PC)
Link Manager Protocol (LMP)	Manages Piconet membership and link activity
Baseband Layer	Manages point-to-point links, handles security, and interfaces user data to the radio links
Radio Layer	Physical data transmission (FHSS in ISM band, at 10 or 100 meter broadcast range)

Frequency Hopping

Bluetooth transmits using Frequency Hopping (FHSS)

Group of RF frequencies = 2401 + k MHz,
for k = 0, 1, ... , 78

Specific Hop Sequence depends on

Bluetooth Service

Bluetooth Clock

Bluetooth Device

Data transmission

Pseudorandom hop sequence

Connection control

Deterministic hop sequences

Frequency Hop Sequence

Train = sequence of integers $\{k_0, k_1, k_2, ..., k_N\}$

$0 \le k_i \le 78$, for i = 0, 1, ..., N

N = 16 or 32

Time Slots

Bluetooth Clock is a 28-bit counter

Upper 27 bits define Bluetooth Time Slot

2 Clock Cycles per Time Slot

Counter creates 2^{27} = 134,271,728 numbered Time Slots

Counts from 0 to $2^{27} - 1$ (then returns to 0)

Each Time Slot is 625 µs in length (1600 slots/second)

Time slot number returns to 0 every 23.3 hours

Frequency Hopping

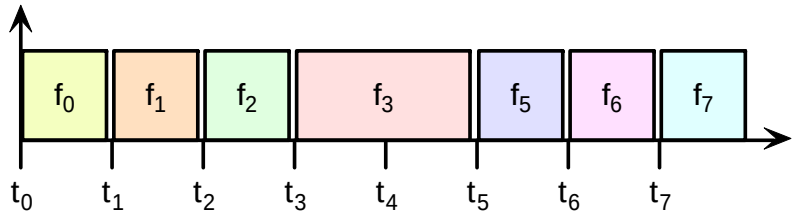
Packet transmission begins on a Time Slot boundary

Packets may be up to 5 Time Slots in length

Frequency hop on each Time Slot

Unless packet is longer than 1 Slot

No frequency hop during a multi-slot packet



Piconet Topologies

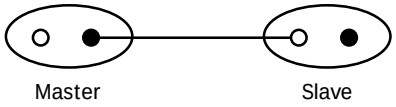
Piconets (from pico = 10^{-12})

Physical Channel

Specific Frequency Hop Sequence

Point-to-Point Piconet

Two devices on a common Physical Channel
FHS is unique to a given Piconet
Master device acts as client
Slave device acts as server



Synchronous Connection Oriented (SCO) Links

Point-to-Point link between Master and Slave

Circuit-mode connection based on reserved slots

Symmetric transmission rate
Supports isochronous information like voice

Master can support

1 to 3 SCO links to one or more Slaves

Slave can support

1 to 3 SCO links with one Master
1 or 2 SCO links from different Masters

Asynchronous Connectionless Link (ACL)

Point-to-Multipoint link

Connects Master and all active Slaves in Piconet

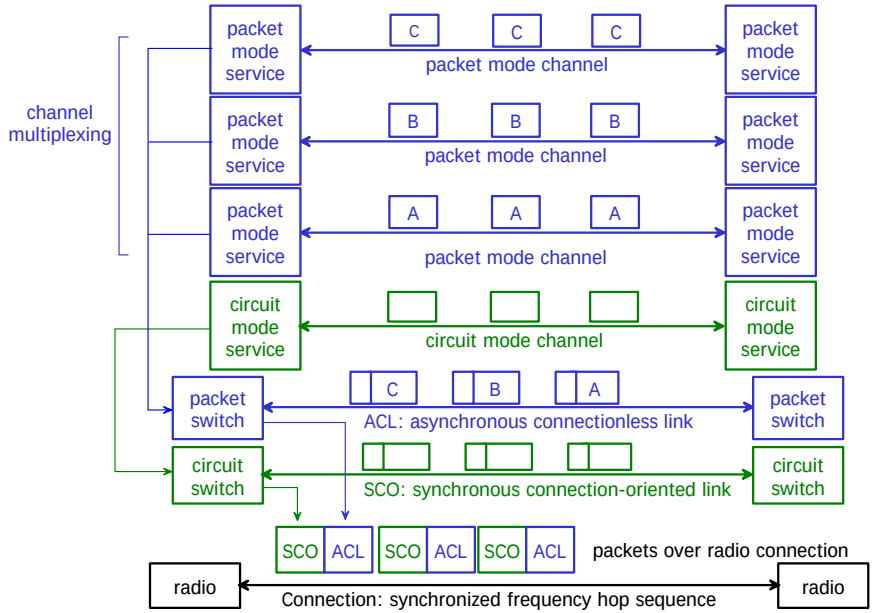
Packet-mode connection

Based on statistical multiplexing
Uses available slots not reserved for SCO links

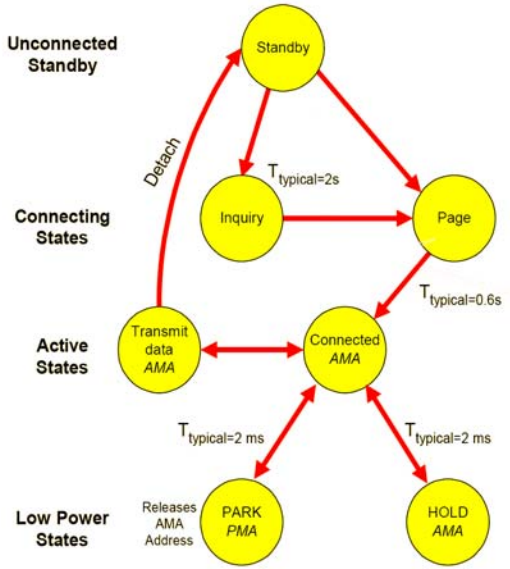
Asynchronous and Isochronous services supported

Only one ACL link between a Master and a Slave

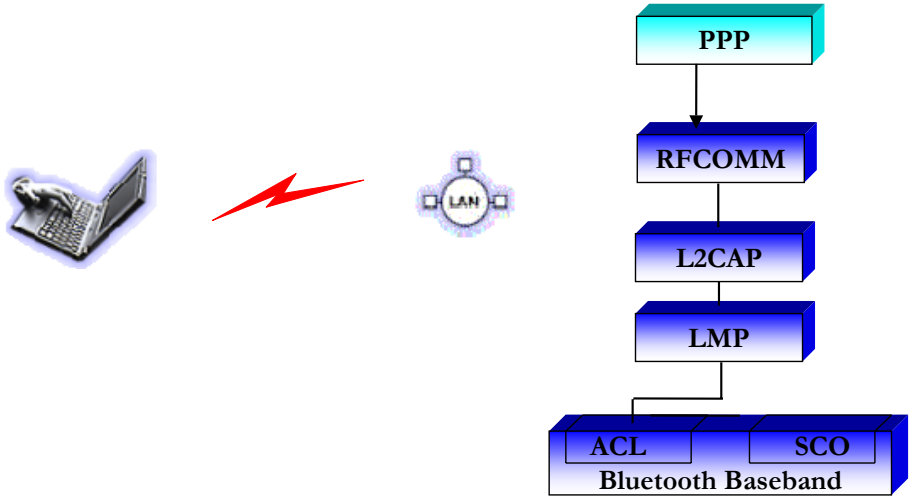
Bluetooth Connection Layers



State Relationships



LAN Access Point Profile



Bluetooth Earpiece

Philips Semiconductor VWS26003

3 Integrated Circuits

Baseband processor (VWS26002)
Ceramic Multi-chip RF module (PBA 31301)
External Flash memory

NiMh or Lithium ion battery

Talk time ~4 hours

Size weight 75g, 15cc



Philips Semiconductor VWS26003

VWS26002 Baseband processor

ARM7 TDMI 32-bit embedded RISC processor
- 72 kbytes internal SRAM
- 4 kbytes internal ROM
- 4 kbytes internal SRAM instruction cache
- Timers and watchdog.
- 8 general purpose PIO pins.
- Voice Codec

PBA 31301 Radio Frequency Module

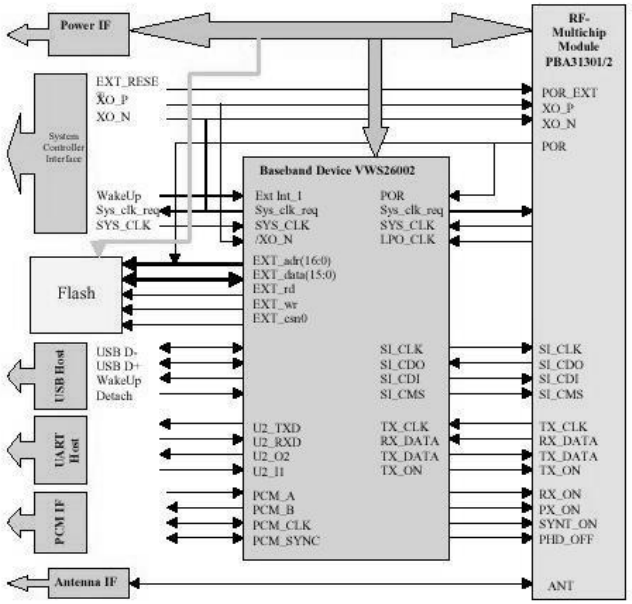
Software

Point to Point Protocol stack

Systems

or NiMh or Li Ion battery

Philips Semiconductor VWS26003



Single Chip Bluetooth Device Controller

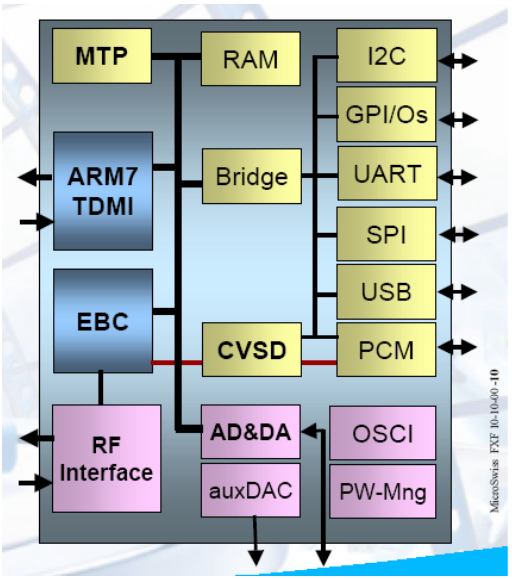
Philips PCD87750E

MTP = Multiple Time Programmable ROM

EBC = Ericsson Bluetooth Core

CVSD = Continuously Variable Slope Delta modulation

SPI = Security Parameter Index



Typical Earpiece Organization

