Engineering 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
```


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

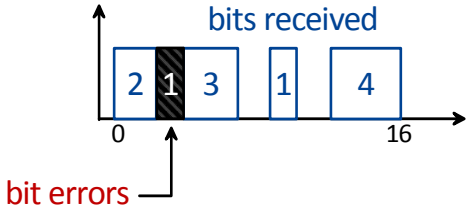
$$\text{throughput} = \frac{\text{error-free data received per second}}{\text{capacity}}$$

Takes account of

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

utilization = 11 / 16 = 68.75%

througput = 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

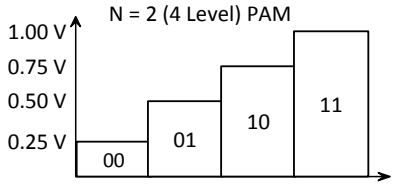
Bits transmitted per second = (symbols / second) × (bits / symbol)

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

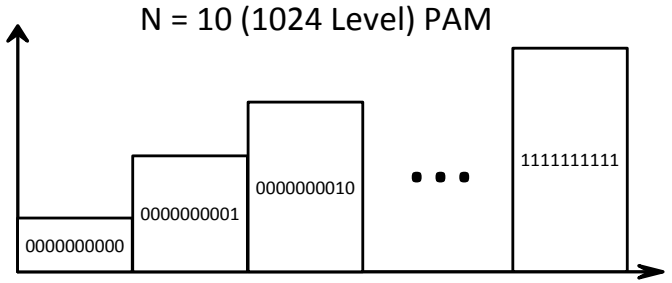
Define 2^{10} = 1024 electrical symbols (max for SNR on phone line)

Baud rate = 3300 symbols / second

Bits transmitted per second

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

= 33,000 bps



Modeling Information

Information

Set of possible answers (outcomes) to questions (tests)

- Finite set (yes/no, day of week, 256 pixel colors, etc)
- Infinite set ("Guess what happened today!")

Communication — transmission of symbol to receiver

Before transmission receiver has limited knowledge of symbol

- Permitted range of symbols (universe of outcomes)
- Statistical distribution of symbols within range

After transmission receiver has better knowledge of outcome

- Receiver tests message to decides on most likely symbol (outcome)
- Decision accuracy limited by noise

Noise

Interference, rounding-off errors, resolution of detector, etc.

Communication does not determine unique outcome

Modeling Noise

Receiver detects

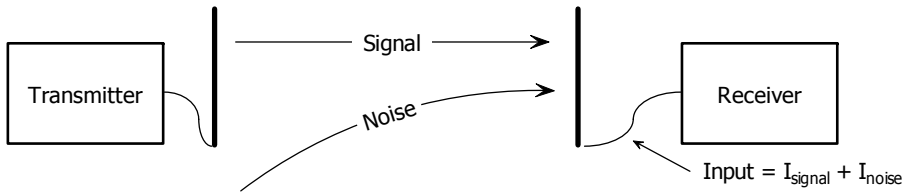
Signal from transmitter

Noise sources

- Other transmitters
- Resolution errors
- Electrical cables and devices
- Lightening

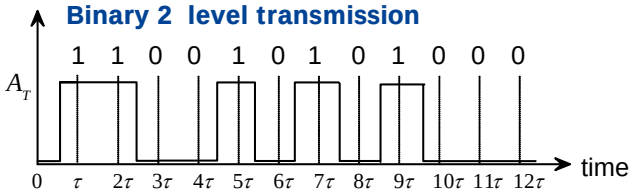
Input

- Electrical current or voltage
- Sum of Signal and Noise

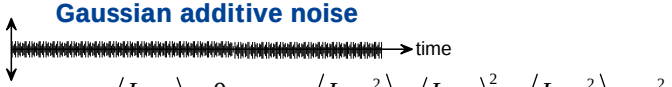


Signal and Noise Inputs in 2-Level Transmission

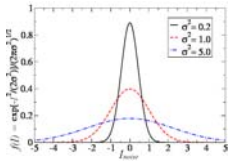
Binary 2 level transmission



Gaussian additive noise



$$\langle I_{noise} \rangle = 0 \qquad \langle I_{noise}^2 \rangle - \langle I_{noise} \rangle^2 = \langle I_{noise}^2 \rangle = \sigma^2$$



Received signal = faded transmission + added noise



$$\langle I_{signal} \rangle = \tfrac{1}{2} A \qquad \langle I_{signal}^2 \rangle = \tfrac{1}{2} A^2$$

Decision

Signal < $A/2$ ⇒ binary 0

Signal > $A/2$ ⇒ binary 1

Signal to Noise Ratio

$$SNR = \frac{\langle I_{signal}^2 \rangle}{\langle I_{noise}^2 \rangle} = \frac{A^2}{2\sigma^2}$$

Probability of Error in 2 Level Transmission

$$P(\text{error}) = P(\text{error}\,|\,0)\cdot P(0) + P(\text{error}\,|\,1)\cdot P(1)$$

$$= P(\text{error}\,|\,0)\cdot\frac{1}{2} + P(\text{error}\,|\,1)\cdot\frac{1}{2}$$

$$= \frac{1}{2}\left[P(\text{error}\,|\,0) + P(\text{error}\,|\,1)\right]$$

$$= \frac{1}{2}\left[P\left(I_{noise} > \frac{A}{2}\right) + P\left(I_{noise} < -\frac{A}{2}\right)\right]$$

$$= \frac{1}{2}\left[\frac{1}{\sqrt{2\pi\sigma^2}}\int_{A/2}^{\infty} e^{-I^2/2\sigma^2}\,dI + \frac{1}{\sqrt{2\pi\sigma^2}}\int_{-\infty}^{-A/2} e^{-I^2/2\sigma^2}\,dI\right]$$

$$= \frac{1}{2}\left[1 - \text{erf}\left(\frac{A}{2\sqrt{2}\sigma}\right)\right] = \frac{1}{2}\left[1 - \text{erf}\left(\frac{1}{2}\sqrt{\frac{A^2/2}{\sigma^2}}\right)\right] = \frac{1}{2}\left[1 - \text{erf}\left(\frac{1}{2}\sqrt{\frac{\langle I_{signal}^2 \rangle}{\langle I_{noise}^2 \rangle}}\right)\right]$$

$$= \frac{1}{2}\left[1 - \text{erf}\left(\frac{1}{2}\sqrt{SNR}\right)\right], \qquad \text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-y^2}\,dy$$

$SNR = 1 \rightarrow P(\text{error}) = 0.24$

$SNR = 25 \rightarrow P(\text{error}) = 2.1\times10^{-4}$

$SNR = 1023 \rightarrow P(\text{error}) = 1.2\times10^{-113}$

Quantifying Information

Set of possible outcomes (received symbols)

$K = 2^k$ different symbols

Symbol label = k-bit binary integers

Communication content

One symbol of K possible symbols

One label: $k = \log_2 K$ bits

Information Rates

System transmits one symbol in τ seconds

Baud rate W = $(1/\tau)$ symbols / second

Bit rate b = (W symbols / second) × (k bits / symbol)

= $W \times k$ bits / second

= $W \log_2 K$ bps

Shannon's theorem

$K_{\text{maximum}} = 1 + SNR$ ⇒ Maximum Capacity $= W \times \log_2[1 + SNR]$

Shannon's Theorem — Examples

Average signal power = average noise power

$$\text{Capacity} = W \times \log_2[1 + SNR] = W \times \log_2[1+1] = W$$

V.34 modem

Capacity (Bit Rate) = 33,000 bps

Symbol rate W = 3,300 Hz

$$33{,}000 = 3{,}300 \times \log_2[1 + SNR]$$

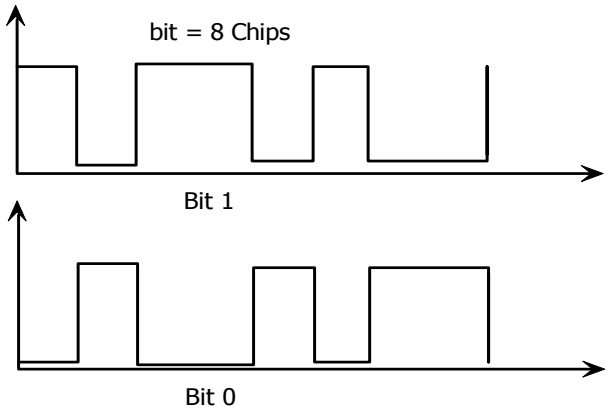
$\log_2[1 + SNR] = 10$ ⇒ Required $SNR = 2^{10} - 1 = 1023$

Probability of Error in DSSS

Direct Sequence Spread Spectrum (DSSS)

Encode 1 data bit as an m-bit "chip" sequence

1 data bit error $\Rightarrow$ $m/2$ bit errors in one chip



DSSS Lowers Probability of Error

$$\begin{aligned} P(\text{bit error}) &= P(\text{at least } m/2 \text{ chip errors}) \\ &= P(m/2 \text{ chip errors}) + P(m/2+1 \text{ chip errors}) + \dots \\ &= \sum_{k=\frac{m}{2}}^{m} \binom{m}{k} \left[P(\text{chip error})\right]^k \left[1 - P(\text{chip error})\right]^{m-k} \\ &= \binom{m}{m/2} P(\text{chip error})^{\frac{m}{2}} + \cdots \end{aligned}$$

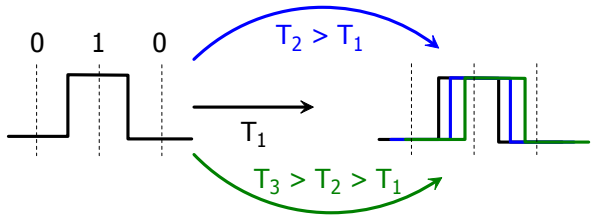
Example

$m = 8$ bits per chip

$P(\text{chip error}) = 10^{-2}$

$$P(\text{bit error}) = \binom{8}{4}\left(10^{-2}\right)^4\left(1-10^{-2}\right)^4 + \dots \approx 10^{-6}$$

Inter-Symbol Interference



Transmitted signal undergoes multipath delay

Received signal is sum of delayed contributions

Inter-Symbol Interference (ISI)

Interference caused by overlap between sequential bits

Causes bit errors

Jitter

Delay varies from bit to bit

Difficult to determine proper sampling clock

Data Concentration

High capacity link

No single node can utilize link capacity

Example

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

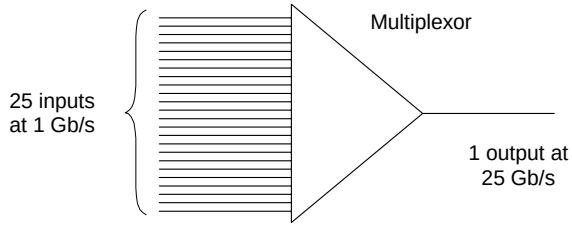
Multiplexing

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)

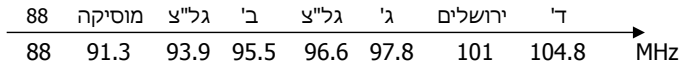
Divide available frequencies (bandwidth) among nodes

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

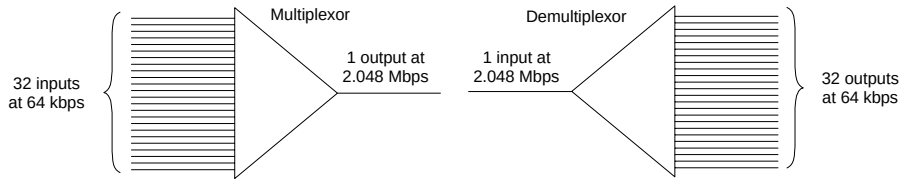
Node transmits in assigned time slot

Example

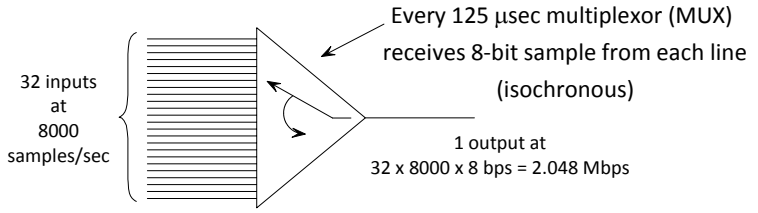
E1 digital line transmits at 2048 kbps

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

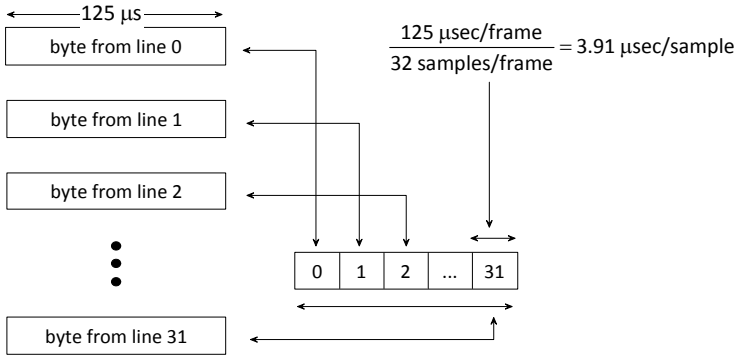
32 x 64 kbps = 2048 kbps = 2.048 Mbps



E1 Multiplex



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



Mixed Multiplexing
Cellular telephone

Time Division Multiple Access (TDMA)

Used on GSM (2G) and UMTS (3G) phones

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

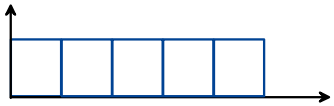
Time Division Multiplexing (TDM)

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

Digital audio stream at n / T bps

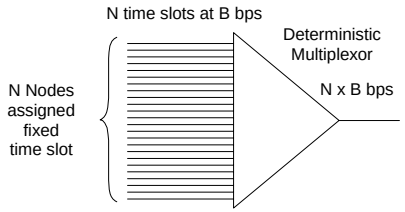
Multiplexing Statistics

Deterministic multiplexing (CBR)

N Nodes = N time slots

Node reserves fixed time slot

- Guaranteed transmission capacity
- Node transmits in assigned time slot



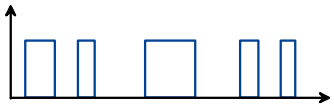
Example

PDH		
E1	32 x DS0	2.048 Mbps
E2	4 x E1	8.448 Mbps
E3	4 x E2	34.368 Mbps
E4	4 x E3	139.264 Mbps

SDH		
STM-0	21 x E1	51.84 Mbps
STM-1	E4	155.52 Mbps
STM-4	4 x E4	622.08 Mbps
STM-16	16 x E4	2.48832 Gbps
STM-64	64 x E4	9.95328 Gbps
STM-256	256 x E4	39.81312 Gbps

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 (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 B = 55 Mbps > average data rate λ = 0.28 bps

$$P(1\text{ bit}, 1\text{ second}, \lambda = 0.28) = \frac{(0.28)^1}{1!} e^{-0.28} = 0.21$$

Multiplexing Statistics

Statistical multiplexing (VBR)

M nodes > N time slots

Bursty data

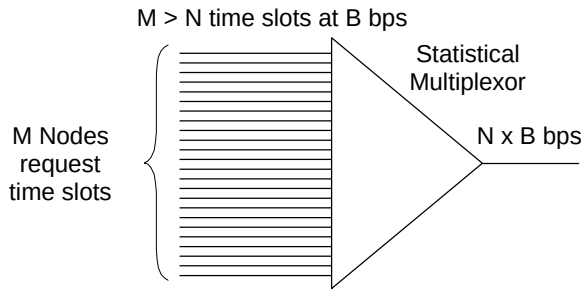
- Average data rate λ < peak data rate B

Average traffic = M x λ < capacity = N x B

- Actual traffic < capacity ⇒ OK
- Actual traffic > capacity ⇒ data delayed or lost

Example

Internet routers



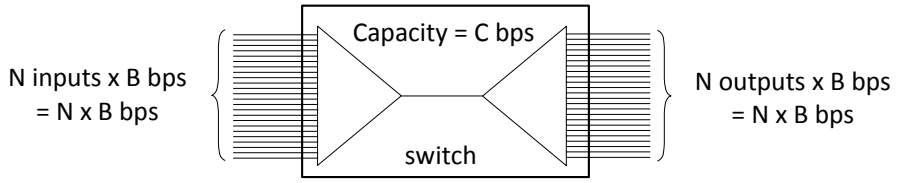
Switching

Switch

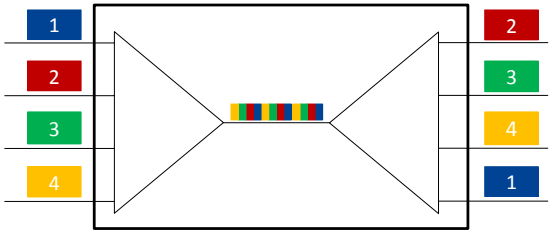
Multiplexor + Demultiplexor

Data at **input_port$_i$ → output port$_j$**

i,j = 0, 1, 2, ... , N - 1



Example



Circuit Switching — Circuit Mode Connection

Deterministic multiplexing

Capacity C = N × B

Dedicated (reserved) link

input_port$_i$ → output port$_j$

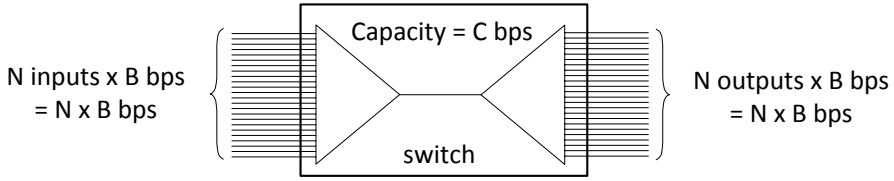
No competition (M nodes = N time slots)

Guaranteed capacity B — if used or not

Example

Bezeq phone call

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



Packet Switching — Packet Mode Connection

Statistical multiplexing

Capacity C = N × B < total possible demand = M × B

Dynamical time slot assignment (on request)

input_port$_i$ → output port$_j$

Competition

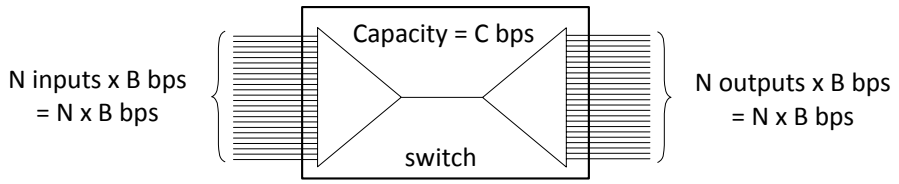
More ports than capacity (M > N)

Demand > capacity ⇒ delay

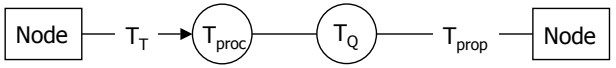
Example

Internet router

Packet queue — first come first served



Message 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 processing time in intermediate node

Propagation delay T_{prop}

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

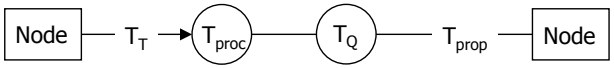
Example: 4 km / (2×10^8 km/s) = 2×10^{-8} sec << 10 sec

Queuing delay T_Q

Time packet waits in buffer for previous packets (congestion)

T_Q = (service time per packet) × (packets waiting in buffer + 1)

Example of Queuing Delay



Queuing delay T_Q

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

T_Q = (service time per packet) / (1 – utilization)

Queuing delay example

Service time per packet = 10 ms / packet

Service rate = 100 packets / second

Average traffic = S = 85 packets / second

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

Buffer level = 0.85 / (1 – 0.85) = 5.7

T_Q = (10 ms / packet) / (1 – 0.85) = 67 ms

Switch capacity C = 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

Flow Control and Congestion Control

Flow control

Sender avoids overflow of receiver buffer

Congestion control

All senders avoid overflow of intermediate network buffers

Buffer arrival rate

Bytes / second arriving from network

Buffer empty rate

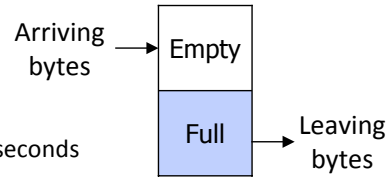
Bytes / second leaving to network or application layer

Buffer file time

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

Example

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



Congestion Control

Queuing theory

Assumptions

Data packets arrive independently (Poisson statistics)

Random length (bytes)

Average arrival rate in steady state

Data packets leave independently (Poisson statistics)

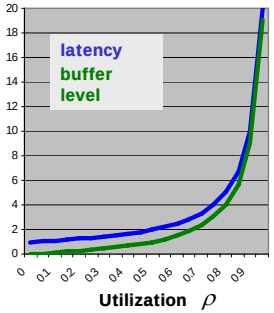
Average emptying rate in steady state

Results

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

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

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

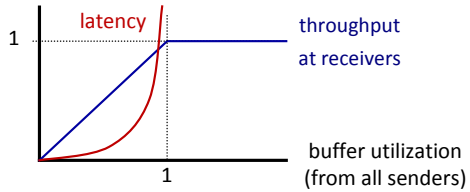


Congestion Control
Buffer throughput

(Over)-simplified throughput model

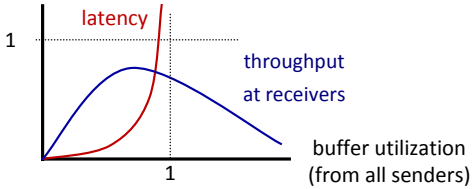
$$\text{throughtput} = \frac{\text{receive rate}}{\text{maximum receive rate}}$$

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



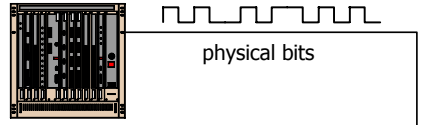
Realistic throughput behavior

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



Data Link

Infrastructure layers



Physical layer (**PHY**)
- Physical transmission of bits

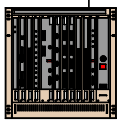
Data Link layer (**DL**)
- Management of **PHY**
- Make physical technology do what we want

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

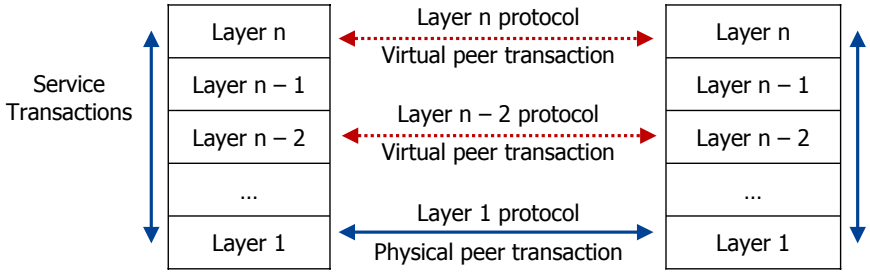


Layered Protocol Model

Layered communication
Communication task divided into layers

Protocol stack
Specific peer-to-peer protocol defined at each layer

Layer n protocol
Performs **VIRTUAL COMMUNICATION** between layer n peers
Processes only layer n information
Passes request to layer n – 1 for communication service
Receives response from layer n – 1



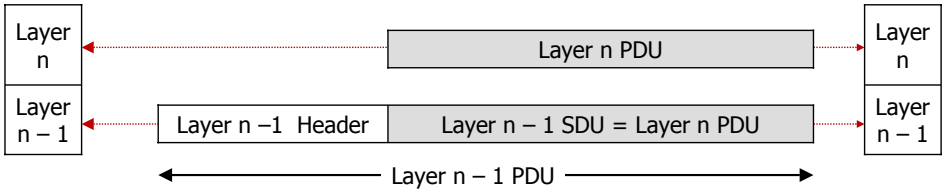
Encapsulation — Protocol Headers

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

Service Data Unit (SDU) at layer n – 1
Message received from layer n
Treated as meaningless data by layer n – 1

Protocol Data Unit (PDU) at layer n – 1
Message sent by layer n – 1 protocol
Includes layer n – 1 SDU = layer n – 1 header + layer n PDU



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, …)

Internet Functional Model

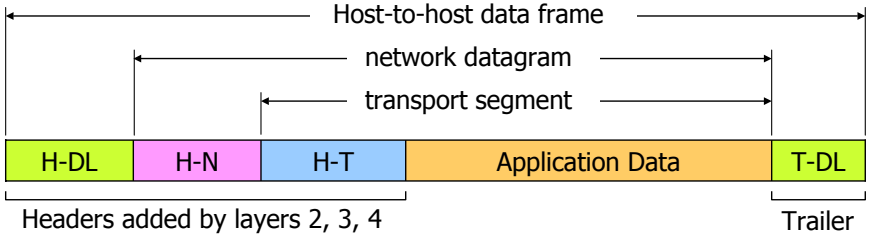
OSI Layer	OSI Function	Internet Layer	Comment
7	Application	Application	Application provides presentation service and some session service (transactions)
6	Presentation		
5	Session	Transport	Internet session management can be: Reliable — with transport service Unreliable — without transport service
4	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

Internet PDUs

Protocol Data Unit (PDU)

Layer	Message	PDU	ID
Application	Data	Message	Socket
Transport	Header	Segment	Port
Network	Header	Datagram	Network (IP) Address
Data Link	Header + Trailer	Frame	Hardware Address
Physical	Bits	Signal	Attachment



Endpoint

Network Address + Port

Data Link Sublayers

Logical Link Control (LLC) sublayer

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

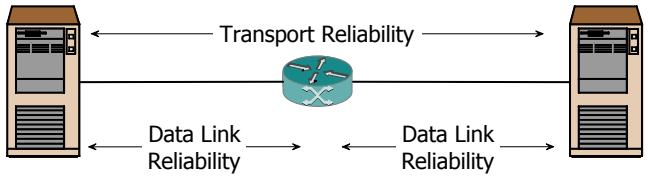
Medium Access (MAC) sublayer

- 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

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

- Transmit locally using hardware MAC addresses

Medium Access Sharing

Time division

- Each host granted full bandwidth in allocated time slot
  - Time slot allocated deterministically or statistically
- 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

Connection Types

Connection

- State machine associated with data exchange
  - Set-up channel before data exchange
  - Monitor channel state during data exchange
  - Multiple transactions associated with connection state
  - Close channel after data exchange

Connection-oriented

- Example — **phone call**
  - Connection at application layer
  - Enter number → answer call → extended conversation → disconnect

Connectionless

- Example — **email message**
  - No connection at application layer
  - Send email → hope message arrives → hope message is found / read

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

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 Internet Protocols

Serial Line Internet Protocol (SLIP)

RFC 1055 (old and rarely used)

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 (layer 2 control 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 sublayers

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)

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	

Ethernet Switch (Hub)

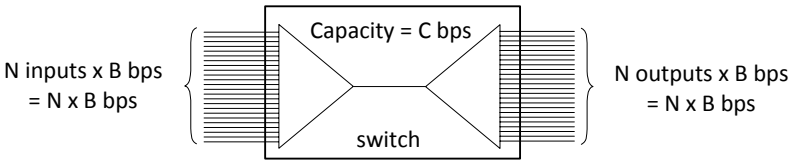
Implements Ethernet medium access (MAC) protocol

Simplest layer 1 + 2 packet switch architecture

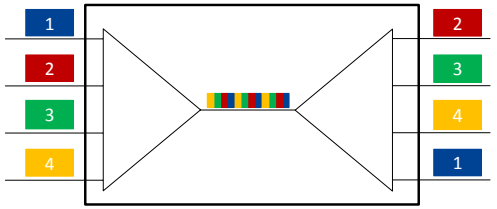
Frame at input port → output port by DEST address

Learning mode

SRC address in frame → associates MAC address with port number



Switch Fabric: typically C = N x B ⇒ no blocking or buffering



Asynchronous Transfer Mode (ATM)

Complex layer 1 + 2 packet switch infrastructure

Path + Circuit Switching

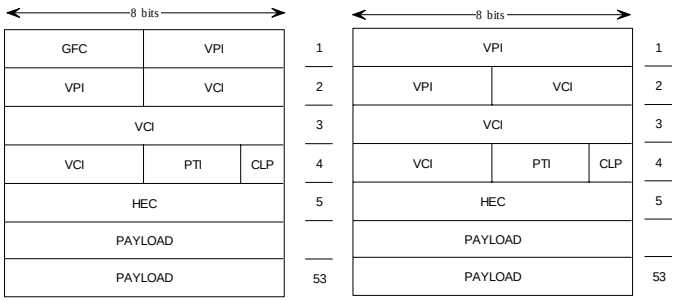
- VPI — virtual path identifier
- VCI — virtual circuit identifier

Capacity

- 2.5 Gbps (STM-16)

Small frames (cells) ⇒ fast priority switching + low latency (delay)

- 53 byte cell = 5 byte header + 48 byte data

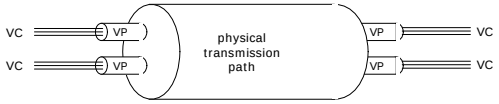


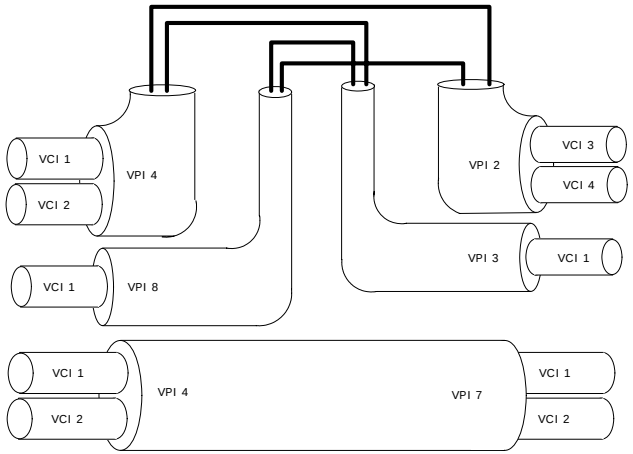
at User-to-Network Interface (UNI) at Network-to-Network Interface (NNI)

GFC
Generic Flow Control
PTI
Payload Type Indicator
CLP
Cell Loss Priority
HEC
header error check

Virtual Path:Virtual Circuit

Path = set of circuits





Quality of Service (QoS) Parameters

Service Level Agreement (SLA)
- Defines parameters for each service flow
- Service implemented in switching operations

Peak Cell Rate (PCR)
- Maximum instantaneous transmission rate

Sustained Cell Rate (SCR)
- Average transmission rate measured over time

Minimum Cell Rate (MCR)
- Minimum required cell rate

Cell Loss Ratio (CLR)
- Percentage of cells lost to error / congestion

Cell Transfer Delay (CTD)
- Total in system delay due to propagation + queuing + service delay

Cell Delay Variation (CDV)
- Variance of CTD (jitter)

Burst Tolerance (BT)
- Maximum burst size (cells) permitted at peak rate

Quality of Service (QoS)

Traffic categories

Constant Bit Rate (CBR)
- Emulates isochronous circuit mode

Variable Bit Rate (VBR)
- Statistical multiplexing
- Specified QoS parameters

Available Bit Rate (ABR)
- Cheaper statistical multiplexing
- No guaranteed minimum loss or delay

Unspecified Bit Rate (UBR)
- Cheapest statistical multiplexing
- No guaranteed QoS

Protocol Layers

ATM Adaptation Layer (AAL)

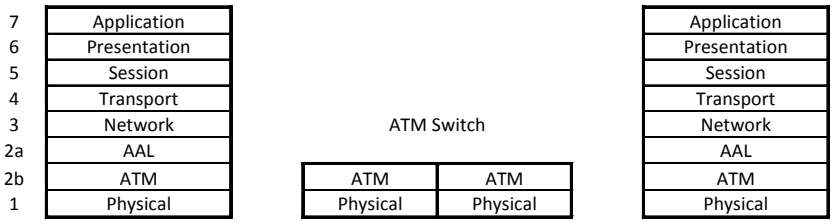
Convergence sublayer
- SAP to higher layers
- Provides service specific functions

Segmentation And Reassembly (SAR) sublayer
- Package higher layer data into / from ATM cell
- Cell loss detection

ATM Layer

Addressing / Switching

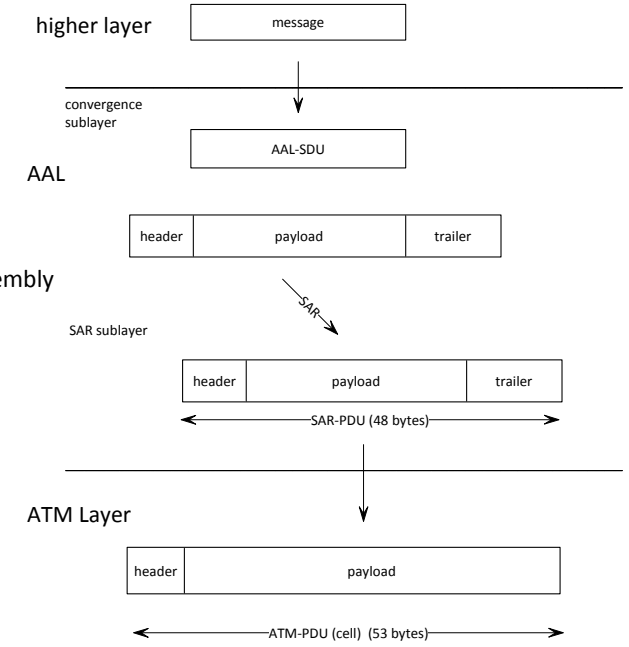
QoS



ATM Adaptation Layer

SAR

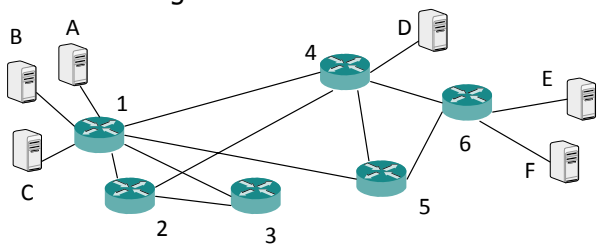
Segmentation and Reassembly



Connection-Oriented Routing

Network of switches and links

Circuit switching or packet switching



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

Virtual Circuit Labeling

Switch topology

Mapping of node to interface number

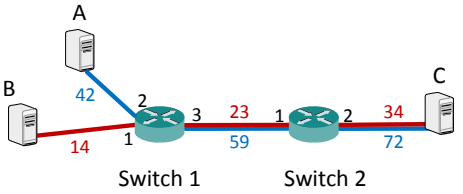
Example

At Switch 1

Interface 1 = Node B

Interface 2 = Node A

Interface 3 = Switch 2



VC number

Labels src-to-dest path

Assigned at routers

Can change at each switch

Example

ATM network packet

VPI	VCI	Type	Priority	CRC	Data

Switch 1 Routing Table			
In		Out	
Intfc	VC	Intfc	VC
1	14	3	23
2	42	3	59
3	23	1	14
3	59	2	42

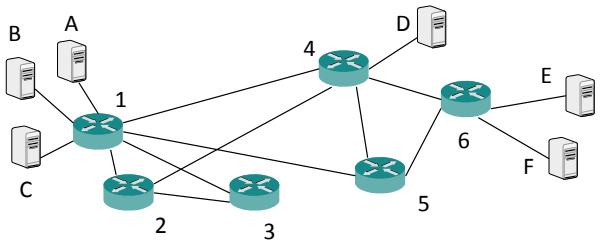
Routing table

Mapping between interface / VC pairs

Connectionless Routing

Network of routers and links

Packet switching



Each datagram

Routed individually through network

Has source and destination address in header

Data Link header or Network header

Datagrams may follow separate routes

Example

src = B dest = F	data

B → 1 → 4 → 6 → F

B → 1 → 5 → 6 → F

Datagram Forwarding

Router topology

Mapping of node to interface number

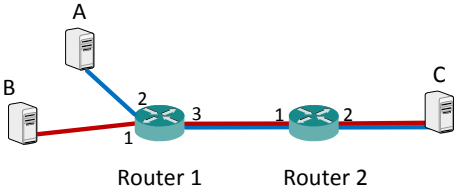
Example

At Router 1

Interface 1 = Node B

Interface 2 = Node A

Interface 3 = Router 2



Node address

Labels node

Router handles next hop

Src + dest address in IP header

src	dest	other fields	data

Routing table

Mapping between dest addr and interface

Switch 1 Routing Table	
In	Out
Dest Addr Range	Intfc
A	2
B	1
other	3

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 multiples
Service type (see chapter 5)	Differentiated Services Code Point (DSCP) Explicit Congestion Notification (ECN)
Identification	Provides a unique ID to each datagram
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 (usually TCP or UDP)

Internet Protocol version 4 (IPv4)

IPv4 Addressing

Internet address

32-bit address

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

Example

`www.hadassah.ac.il = 212.179.79.228`

Networks / Subnets / Hosts

`Network.Host`

`Network number = 212.179.79.0`

`Host number = 228`

`Network.Subnet.Host`

`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)`

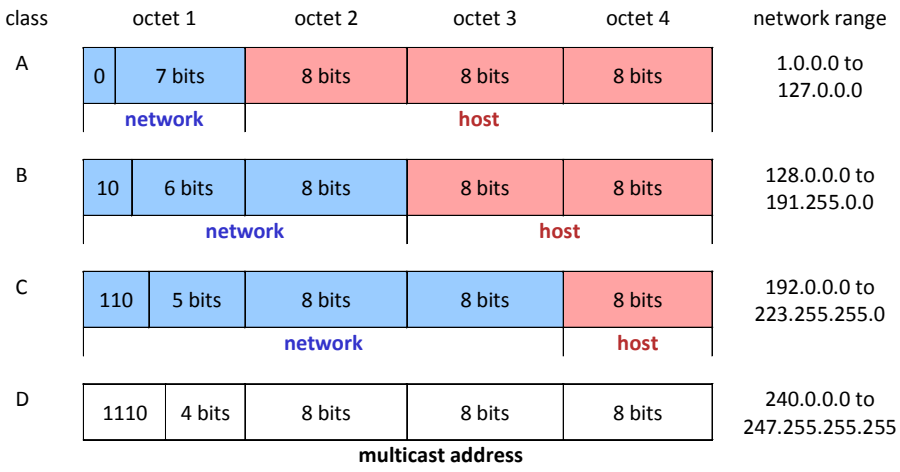
Internet Protocol version 4 (IPv4)

Address classes

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)

Length of network number = 1, 2, 3, … , 31

Not restricted to 7, 16, 24

Address format

`Octet1.Octet2.Octet3.Octet4/bits_in_network_number`

Examples

Class A address — **`10.0.1.5/8`**

8-bit network number = **`10.0.1.0`**

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 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

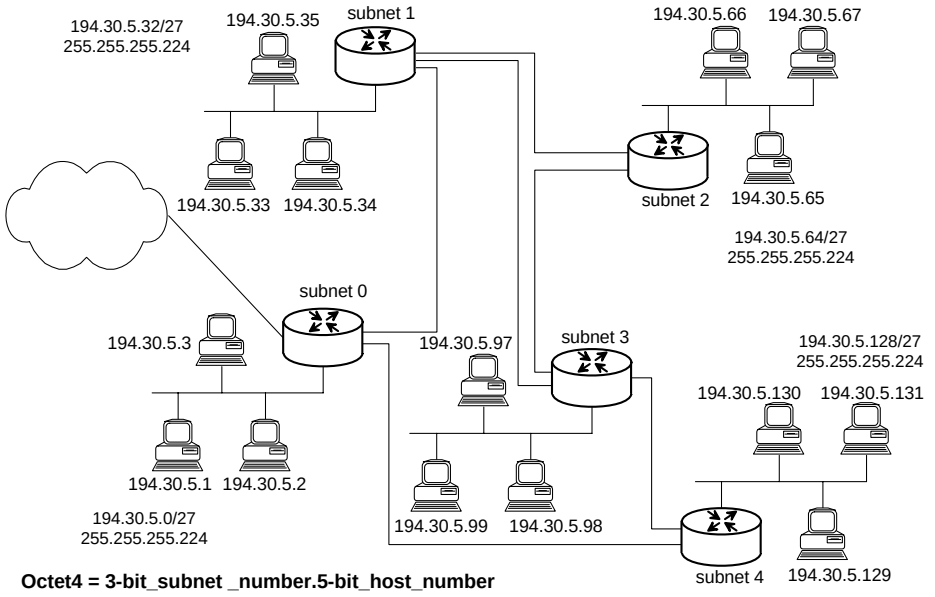
Mask AND IP address = network number

Example

255.255.255.224 AND 192.168.0.33 = 192.168.0.32

Internet Protocol version 4 (IPv4)

Subnet example



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

Internet Routing

Autonomous System (AS)

Nodes managed by one organization

Hierarchical routing

Interior Gateway Protocols (IGP)

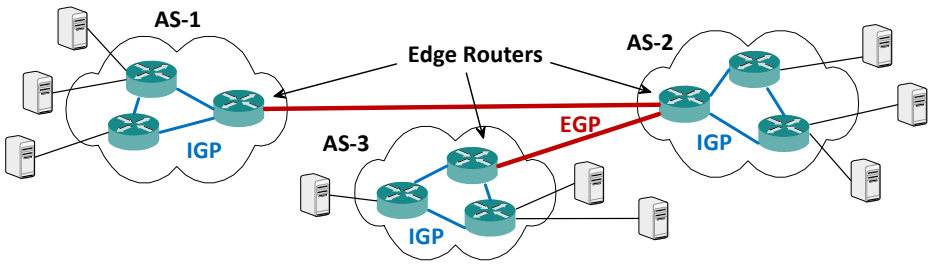
Routing protocols within one AS (Intra-AS)

Exterior Gateway Protocols (EGP)

Routing protocols between ASs (Inter-AS)

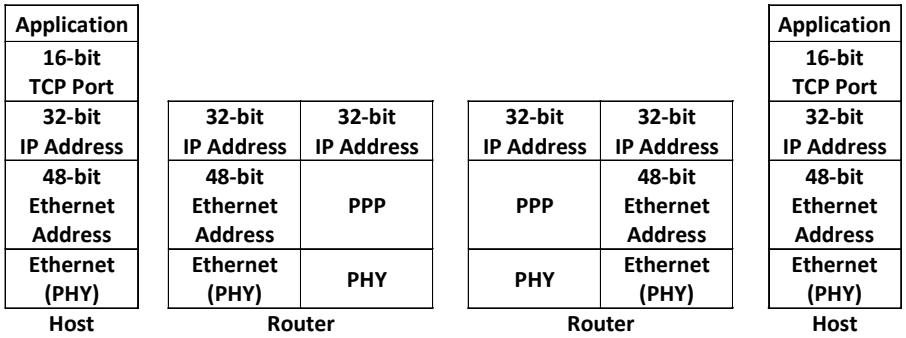
Edge router (gateway router)

Router within AS linked to router in different AS



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

Host finds MAC address for router using an **address resolution protocol (ARP)**

Network Address Translation (NAT)

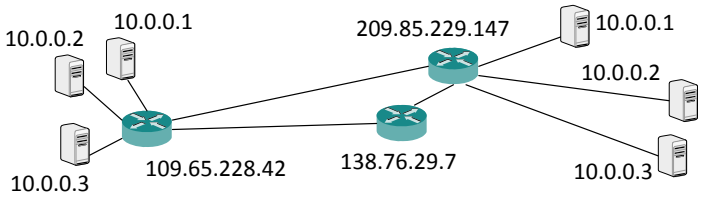
Router

- 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

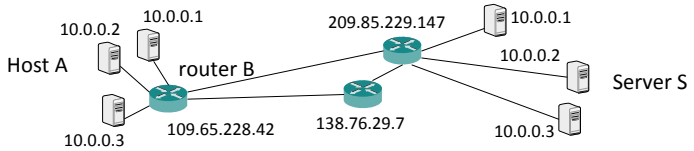
Source port = **3745** Source IP = **109.65.228.42**
Destination port = 80 Destination IP = 209.85.229.147

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

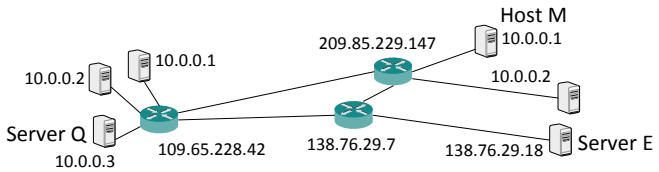
- Define service port on Server Q
- Refer all incoming traffic at gateway for service port to Server Q

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

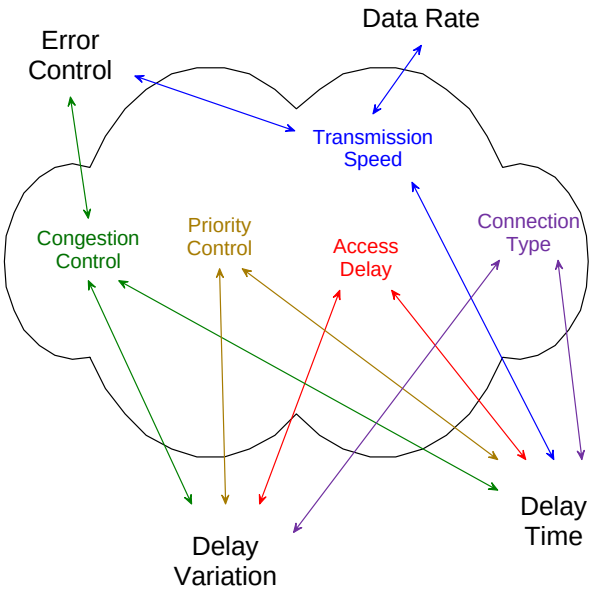


Quality of Service (QoS)

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	Dynamic variations in Data rate / propagation delay (jitter) 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				

Internet 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

Transport Layer — QoS 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

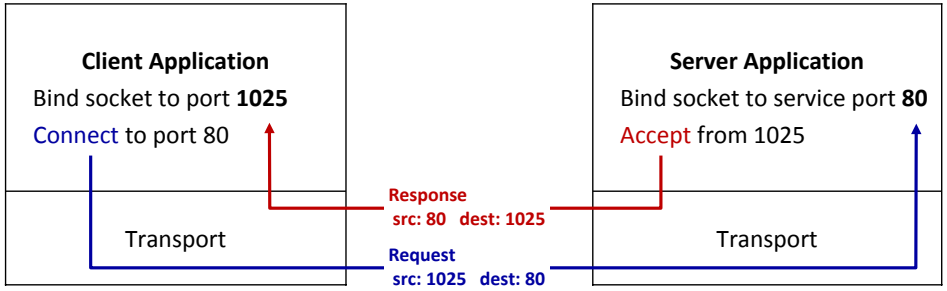
Source / Destination Ports

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

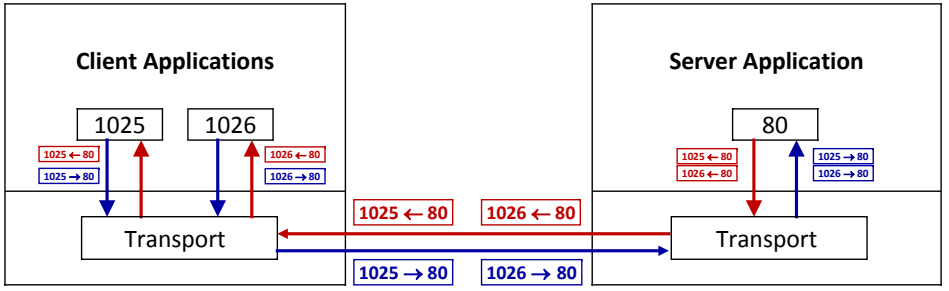
- Multiple sockets ⇒ multiple conversations

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 UDP segment < 2^{16} = 65,536

Maximum length of data = 64 KB – lengths of all headers

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

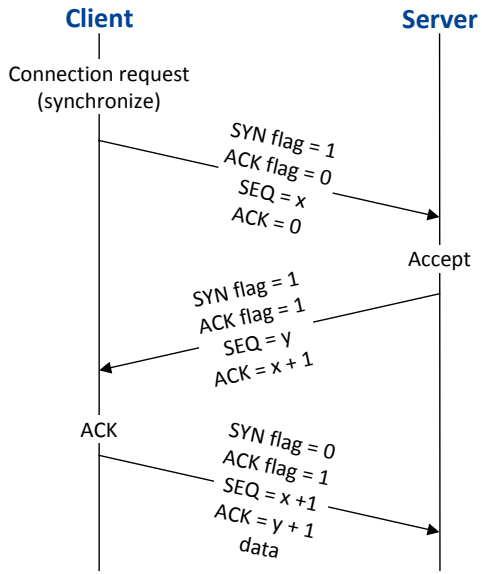
SYN flag = 0

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)

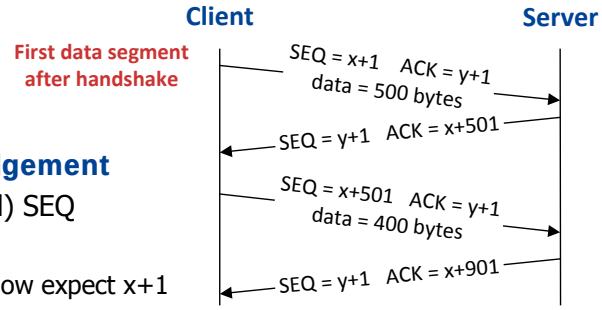
= ISN + 1 + data bytes sent in all **previous** segments

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

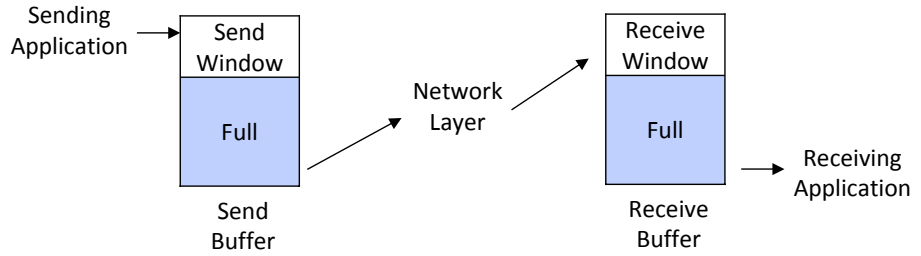
Available space in send buffer

Receive buffer

Holds received segments until requested by application

Receive window

Available space in receive buffer



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 for segment = 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}
            resend
            timer = RTO
```

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

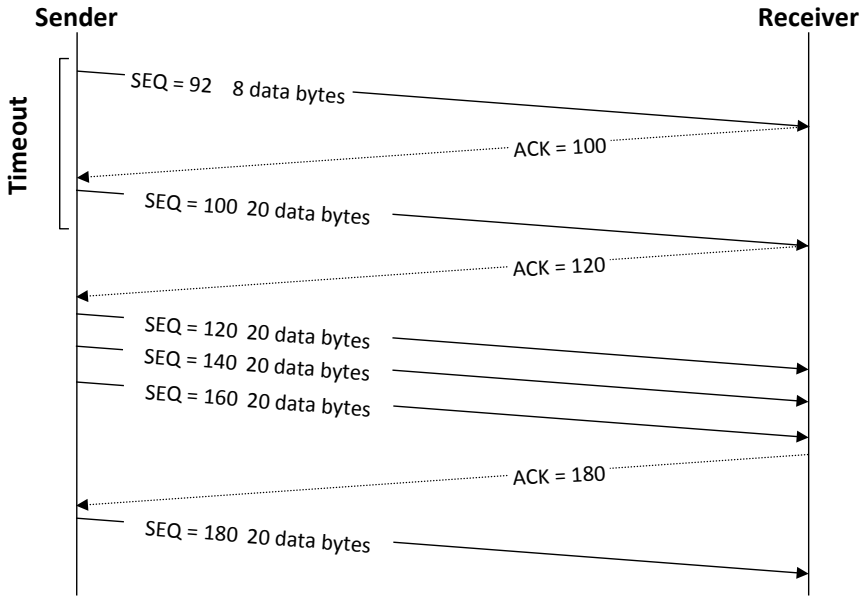
if (ACK delay timer = 0 && ack_buffer > 0)
    Send ACK = expected with updated RecvWindow
    ACK delay timer = ack_delay
    ack_buffer = 0
```


Simplified TCP Receiver — 2

```
if (receive SEQ = x)
    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
```

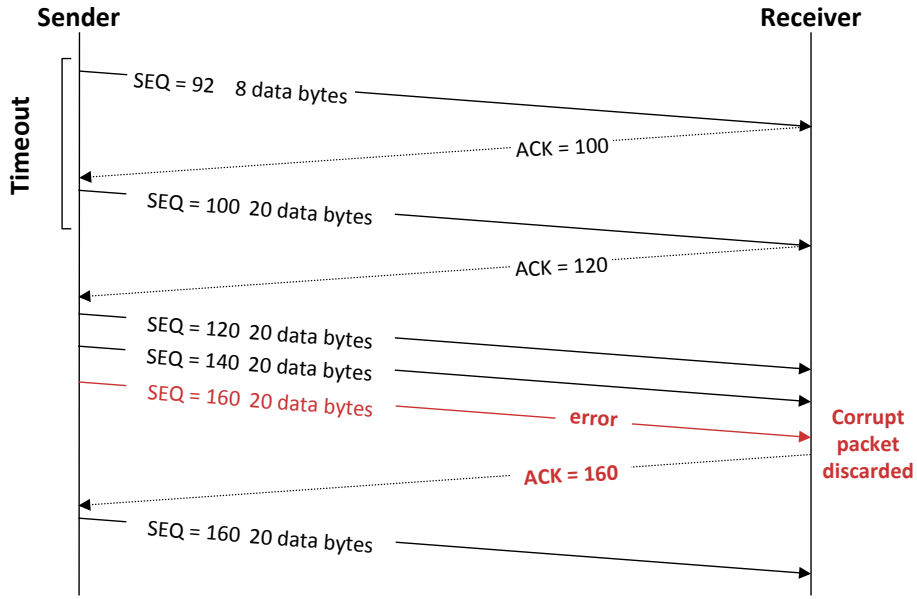

TCP SEQ + ACK

No errors

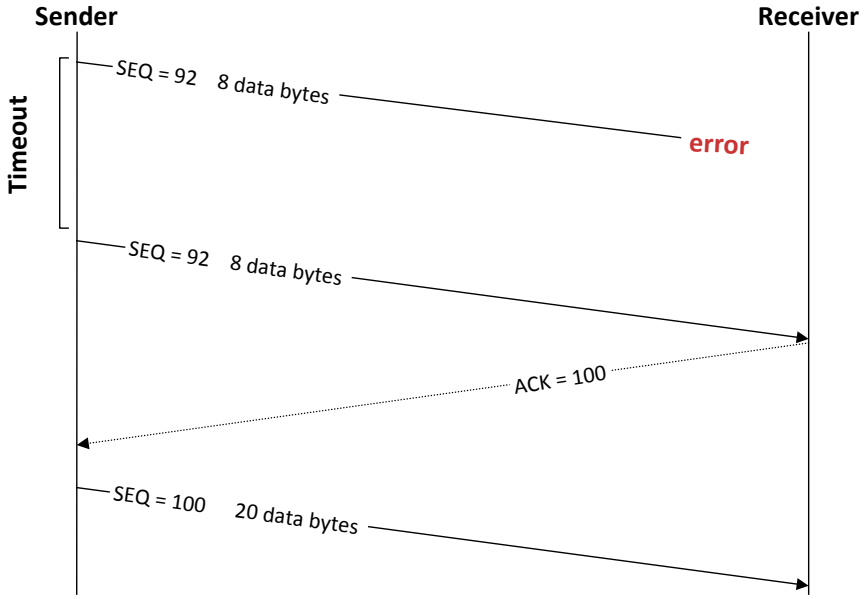


TCP SEQ + ACK

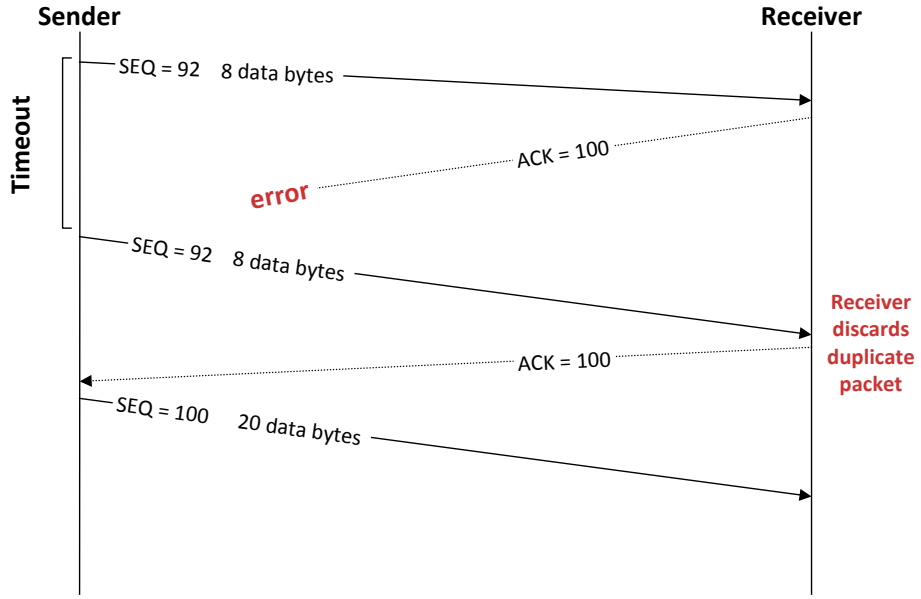
Bit errors



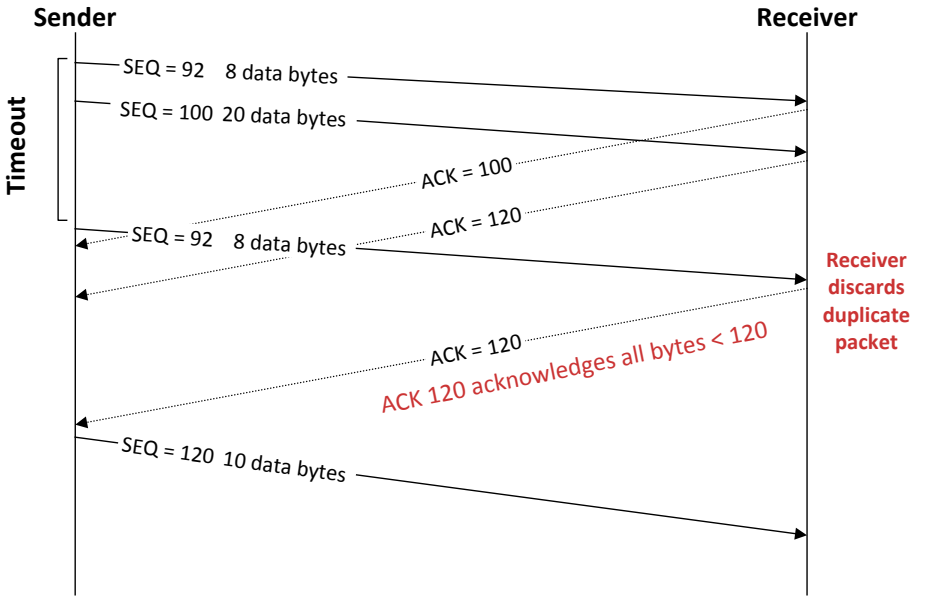
TCP SEQ + ACK
Timeout



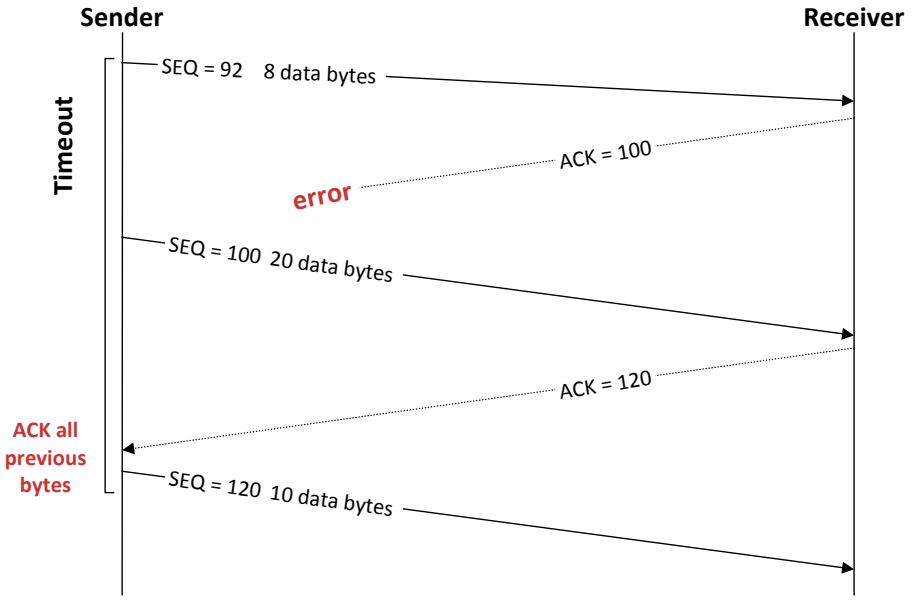
TCP SEQ + ACK
Lost ACK



TCP SEQ + ACK
Missed ACK



TCP SEQ + ACK
Cumulative ACK



TCP Connection Close

Symmetric

Client or server may close connection

FIN segment

SYN flag = 1

SEQ = cumulative SEQ number

ACK segment

ACK flag = 1

ACK = SEQ + 1

FIN segment

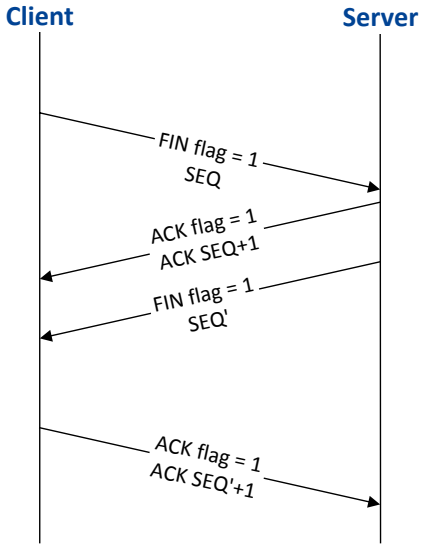
FIN flag = 1

SEQ' = cumulative SEQ number

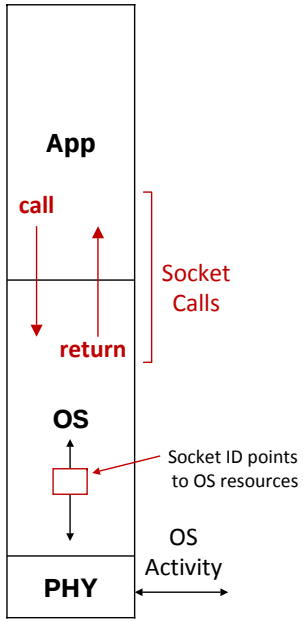
ACK segment

ACK flag = 1

ACK = SEQ' + 1



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

Using UDP

Client	Server
Application Open socket Send data on socket to endpoint (node address + service port)	**Application** Open socket Bind service port Listen on socket Receive data from listen socket
Client UDP Agent Accept data Add header with checksum Send to server	**Server UDP Agent** Perform checksum Error Discard segment No error Pass data to socket by port

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

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)
```

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

Assistive Application 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

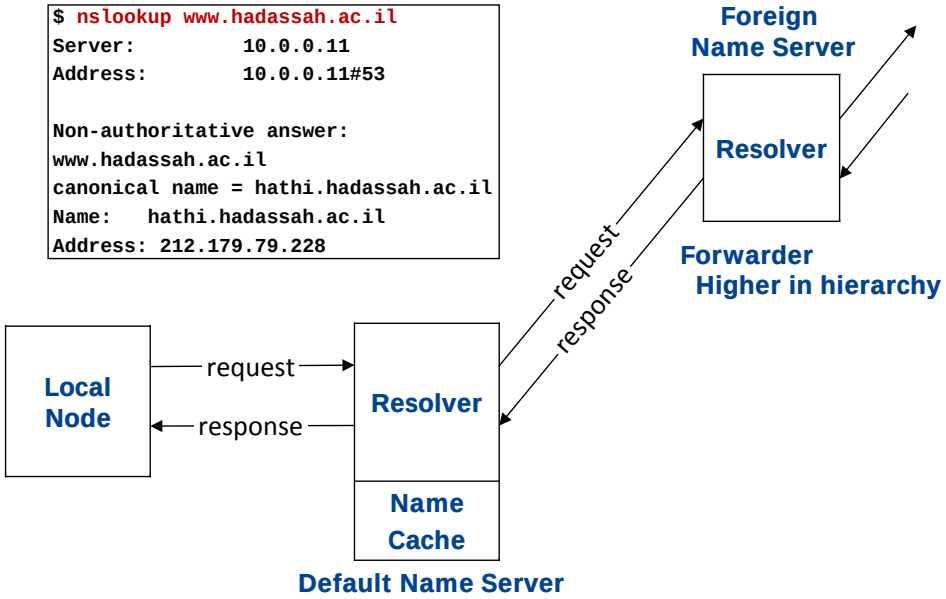
Server allocates IP address from pool of available addresses
DHCP message types
- DHCP server discovery (host broadcast)
- DHCP offer (server response)
- DHCP request (host request)
- DHCP ACK (server provides address)

Domain Name System (DNS)

Address Resolution Hierarchy

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

Non-authoritative answer:
www.hadassah.ac.il
canonical name = hathi.hadassah.ac.il
Name:   hathi.hadassah.ac.il
Address: 212.179.79.228
```



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

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

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 = 4 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:\>
```