Föreläsning 1

* Millions/Billions of connected computing devices ← PC, laptops, Smart phones etc
  hosts = End Systems running network applications

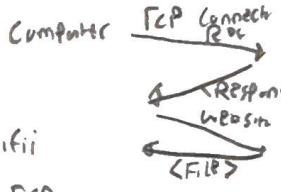
* Communication links ← wireless links & wired links
  Fiber, Copper, Radio and Satellite transmission rate: Bandwidth

* Packet Switches: Forward packets (chunks of data) ← Router
  Router and Switches

Internet "network of network" → interconnected Internet Service Providers
Protocols control sending and receiving of messages eg TCP, IP, HTTP, Skype, 802.11
Internet Standards RFC: Request for comments IETF: Internet Engineering Task Force

* Infrastructure that provides services to applications & provides programming interface to applications

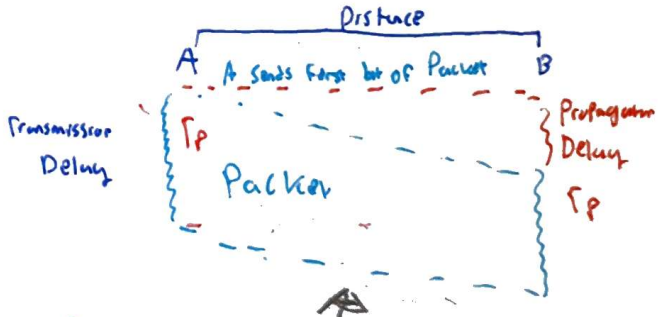
Protocols define format and order of messages sent and received among network entities, and actions taken on message transmission and reception. Machines rather than humans and all communication activity on the internet is governed by protocols.



Network Edge: Hosts: Clients and Servers. Servers often in data centers ← wifi
Network Core: Interconnected routers and network of networks ← The ISP wifi is connected to
Access networks, Physical media: Wired or wireless communication links

How Hosts Sends Packets of data

1. Takes application message
2. Breaks into smaller chunks, known as packets of length L bits
3. Transmit packet on link into access network
4. Links are serial → Packets transmitted one bit at a time, at a certain rate R
   ↳ link transmission rate, link capacity or link bandwidth



Packet transmission Delay	=	Time needed to transmit a Packet with L bits into a link	=	L (bits) / R (bits/sec)

$$T = T_t + T_p = \frac{L}{r} + \frac{d}{s}$$

L = Packet size
R = Link rate
S = Propagation speed
d = Distance

Föreläsning 2 - ZOOM.

The network core

Mesh of interconnected routers. Packet switching: host breaks application-layer messages into packets

↳ Forward packets from one router to the next, across links on path from source to destination. Each pack is transmitted at full link

Two key network-core function

routing: - Determines source destination route taken by packets.

Forwarding: Moves packets from routers' input to appropriate output link

Queing delay and Packet loss

Only one packet can be sent at a time on a link. Othe packets have to wait in queue in the router. If the queue is full then the packet is dropped. Packet loss is it called. Not the Routers fault

Internet Structure: Network of Networks

End systems connect to the internet via access ISPs. Access ISPs in turn must be interconnected. → Resulting network of network is very complex.

Option: Connect each access ISP to a global ISP for transit. Customer and Provider Isp have economic agreement

Peering link: Cooperative link. Share the data traffic between ISPs.

Internet Exchange Point: En neutral point som funkar som Peering link.

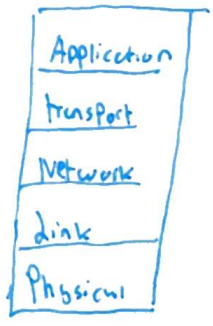
Regional net: Samla alla lokala access network sedan skicka till Globala ISP

Content Provider Network: Gör så att all information är tillgängligt överallt. Distribuerar info snabbt (google osv)

Protocol layers

Why layering

* Explicit structure allows identification and relationship of the different peices
* Modularization eases maintenance and update of system



Internet Protocol Stack

Applications: Supporting network application ← FTP, SMTP, HTTP

transport: Process-Process data transfer ← TCP, UDP

Network: Routing of datagrams from source to destination ← IP, routing Protocols

Link: Data transfer between neighbouring network elements, ← Ethernet 802.11 wifi etc

Physical: bit on the wire

→ Transport layer: Provides communication between applications. End to End Communication

TCP: Skicka sedan få bekräftelse att man har fått packetet. ↳ Stora datamängder

UDP: Skicka och hoppas på det bästa ↳ Enkla grejer som att skicka en fråga

Network Layer: Deliver individual Packets from sending to receiving host. Network Layer Protocol in every host and router. Router examines header file in all IP datagrams passing thru it.

Link Layer: Transfer datagrams between physical adjacent nodes over a link.

↳ Bluetooth, 3G, 4G, Ethernet osv. "What exactly" are we sending this information to

Physical Layer: How the bits move

Chapter 2 - Föreslasning Slides

Creating Applications

Write programs that: run on (different) end systems, communicate over network eg webserver software communicates with browser software

No need to write software for network-core devices: Network-core devices do not run user applications. Application on end systems allows for rapid application development and propagation

Application architectures:

Client-server achitectures: Server: - Always on hosts - Permanent IP-address - data centers for scaling

Clients: - Communicate with server - May be intermittently connected - May have dynamic IP-adresses - Do not communicate direcly with each other.

P2P achitectures: - No always on server - Arbitrary end-systems directly communicate

↳ peer request service from other peers, provide service in return to other peers.

- Peers are intermittently connected and change IP-adresses → Compler management

self Scalability: new peers bring new service capacity as well as new service demand

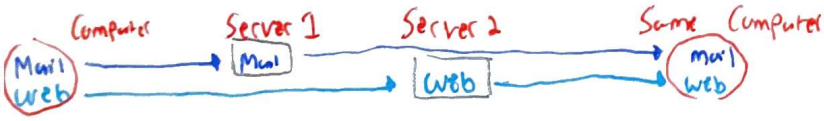
Processes Communicating

Process: Program running within a host. Within same host, two processes communicate using inter-process communication defined by operating system

↳ In different hosts communicate by exchanging messages.

Client, Servers

Client process: Process that iniates communication Server Process: Process that waits to be contacted



Many client processes on same hosts, communicating with different server processes. A server process communicates with many client process.

Adressing Processes

- To receive messages, process must have identifiers
- Each host device has unique IP-adress

Identifiers includes both IP-adresses and port number associated with process on host

HTTP Server: Port 80
Mail Server: Port 25

- Application layer protocol defines:

Types of messages exchanged: request & respons

Messages Syntax: . whats fields in messages . & how fields are delineated

Message Semantics: . Meaning of information in fields

Rules: . For when and how processes send . & respond to messages

Open Protocols: 1) Defined in RFCs 2) Allows for interoperability 3) HTTP SMTP

Proprietary Protocols: Skype

What transport service does an application need?

Data integrity Timing Throughput Security

Internet Transport Protocols Services TCP vs UDP

TCP Service:

Reliable transport: between sending and receiving process

Flow control: Sender won't spam receiver

Congestion Control: Throttle Sender when network overloades

Does Not provide: Timing, minimum throughput guarantee Security

Connection-Oriented: Set-up required between Client and Server process

UDP Service:

Unreliable data transfer: Between Sending and receiving process

Does not provide: Reliability, Flow Control, Congestion Control, Timing, guarantee, security or Connection Set-up

Securing TCP

TCP & UDP: No encryption, Passwords transverse Internet in Cleartext

SSL (Secure Socket Layer) → Provides encrypted TCP Connection, data integrity, end-point authentication.

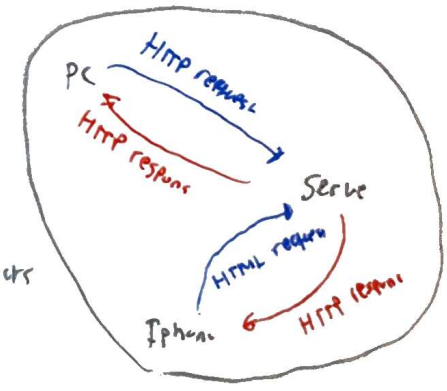
SSL is at application Layer: Applications use SSL libraries which "talk" to TCP.

WEB AND HTTP

A webpage: Consists of objects. Objects can be HTML, JPeg, Java applet etc. Webpages consist of base HTML-file which includes several referenced objects. Each object is addressable by a url

HTTP://www.Facebook.com/RukasProfile

Protocol host name Path name



HTTP overview (HyperText Transfer Protocol)

Web application layer protocol. Client/Server Model

Client: Browser that requests, receives (using HTTP Protocol) and "displays" web objects

Server: Web server (using HTTP) objects in respons to request

HTTP: Respons Status Codes

Status code appear in first line in server-to-client response messages.

- 200 OK → Request succeeded, requested object later in this message
- 301 Moved Permanently → Requested object moved, new location specified later in the messages
- 400 Bad request → Request messages not understood by server
- 404 Not Found → Requested document not found by the server

505 HTTP Not Supported

HTTP is Stateless

Uses TCP, Port 80: Stateless Protocol: Client sends request, Server responds, that's it

Aside
↳ Server maintains no information about past client request

* Protocols that maintain "state" are complex. If client/server crashes, their view of state may be inconsistent, must be reconciled

User-Server State: Cookies Four Components:

1) Cookie header line of HTTP response message
2) Cookie header line in next HTTP request message
3) Cookie file kept on user's host, managed by users browser
4) Back-end database at website

Example

1) Susan accesses internet from PC
2) Visits e-commerce site for the first time
3) When initial HTTP requests arrive at site, site creates
   ↑ unique ID & entry in backend database

Cookies can be used for authorization, shopping carts, recommendations, user session state

Web Caches (Proxy Server)

goal: Satisfy client request without involving origin server

User configures browser: Web accesses via cache. browser sends all HTTP requests to cache.

↳ If object in Cache: Cache returns object. Else: Cache requests object from origin server, then returns object to client.

* Caches acts as both Client & server: Server for original requesting client, Client to origin server

Why Proxy Servers? → Reduce response time for client request
↳ Reduce traffic on an institution's access link

Electronic mail

↳ Three major components: User agent, mail servers, Simple mail transfer protocol: SMTP

User agent

↳ Aka mail reader. Composing, editing, reading mail messages. Outgoing, incoming messages stored on server.

Mail Servers:

Mailbox: Contains incoming messages for user

Messages queue: Of outgoing (to be sent) mail m

SMTP Protocol: Between mail servers to send email messages

note: Client Side: Sending mail server
Server: Receiving mail server.

SMTP:

→ Uses TCP to reliably transfer email messages from client to server, PORT 25

→ Direct transfer: Sending server to receiving server

→ Three phases of transfer ① Handshaking, ② Transfer of messages, ③ Closure

→ Command/Response interaction (like HTTP, FTP) Commands: ASCII text Response: Status code and phase

→ Messages must be in 7-bit ASCII.

Scenario

1. Alice uses UA to compose message to Bob at bob@someschool.edu
2. Alice's UA sends message to her outgoing mail server; message placed in message queue.
3. Client's side of SMTP opens TCP connection with Bob's incoming mail server.
4. SMTP client sends Alice message over the TCP connection.
5. Bob's incoming mail server places the message in Bobs mail box
6. Bobs invokes his user agent to read message.

HELO, MailFrom, RCTP, Quit, Start, Finish

Mail access Protocols

→ SMTP: Delivery/Storage to receiver's server

Mail access Protocol: Retrieve from Server

POP: Post office Protocol [RFC 1939]: Authorisation download

IMAP: Internet Mail Access Protocol [RFC 1730): more feature, including manipulation of stored msgs on server

HTTP: Gmail, Hotmail, Yahoo etc

POP3 (more) and IMAP

→ Previous examples uses POP3 "download and delete" mode → Bob cannot re-read email if he changes client

→ POP3 "download and keep": copies of messages on different clients

→ POP3 is stateless across sessions

→ Imap keeps all messages in one place: at server

→ Imap allows user to organise messages in folders

→ Imap keeps user state across sessions: names of folders and mappings between message ids and folder name

Dns: Domain name System

People: Social Security number or Passport number. | Domain Name System:

Internet: IP-adress → Used for adressing datagrams

Distributed database: Implemented in hierarchy of many name servers

Application-layer Protocol: Hosts name servers communicate to resolve names (adress/name translation)

Internet Domain

↳ Country domains (country code top-level domains, ccTLDs)

→ Generic domains (three letters or more. .com, .net, .org etc.)

DNS: Services and Structure

DNS Services: Hostname to IP-address translation, "resolving", Host aliasing, Canonical, alias names, Mail server aliasing

load distribution, replicated Web servers: many IP-addresses correspond to one name.

Why not centralize DNS (1) Single point of Failure (2) Traffic volume (3) Distant centralized database (4) Maintence

DOESNT SCALE

DNS → a distributed, Hierarchical Database

Client wants IP-address for www.Amazon.com

(1) Client queries root server to find "com" DNS server

(2) Client queries "com" DNS server to get "Amazon.com" DNS server

(3) Client queries "Amazon.com" DNS server to get IP-address for "www.Amazon.com"

DNS: root name servers

→ Contacted by local name server when it cannot resolve name

Root name server: → Maintains database of TLD (Top-Level Domain) servers

↳ When contacted, returns list of DNS servers for TLD in question

TLD, authoritative Servers

Top level domain (TLD) servers

- Responsible for com, org, net, jobs and all top-level country domains eg uk, fr osv
- When contacted, returns a list of authoritative DNS servers for organisation in question

Authoritative DNS Server

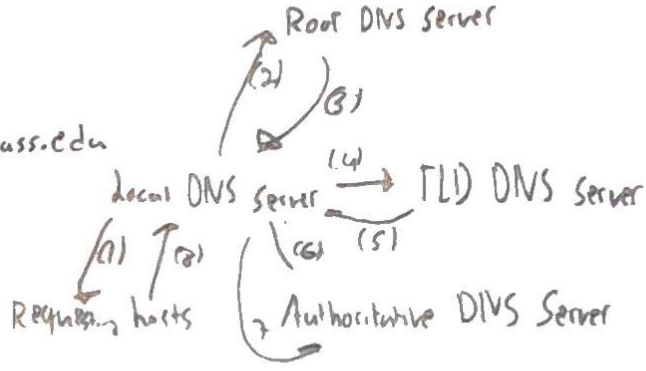
- Organization's own DNS servers, Providing authoritative hostname to IP-mappings for organization's named hosts.
- Can be maintained by organization or service provider
- When contacted, returns host's IP-address(es) → OR possibly, refers to other servers in the organisation

Local DNS server

- Does not strictly belong to hierachy, each ISP (residental ISP, company, uni) has one
- When host makes DNS query, query is sent to its local DNS server
  - Has local cache of recent names-to-address translation pairs, acts as proxy, forwards query into hierachy

DNS name resolution example

1) Host at cis.poly.edu wants IP address for gaia.cs.umass.edu



iterated query

- Contacted server replies with name of server to contact
- "I don't this name but ask this server"
- This is what root, TLD and authoritative DNS server do.

Recursive query

- Contacted server replies with IP-address. → Resolver name
- This is what local server does

DNS: Caching, updating records

- Once any name server learns mapping, it caches mapping
  - Cache entries timeout (disappear) after some time (TTL, time to live)
  - TLD servers typically cached in local name servers thus root name servers not often visited

DNS Records

DNS: Distributed database storing resource records (RR)

RR format: (name, value, type, ttl)

type=A name is hostname, value is IP address

type=NS name is domain, value is hostname

type=MX value is name of mail server associated with name

type=CNAME name is alias name for some (canonical) name

Föreläsning 4 - Socket interface

Sockets

→ Process sends/receive messages to/from its Socket.

2 types. TCP: Reliable, byte stream-oriented → reponsible for dividing the stream into packets

UDP: Unreliable, byte block-oriented datagram → Application responsible for dividing data into packets

Socket Programming with Client

First, Client must open a TCP connection to the server

TCP Connection: ordered delivery of data as a stream of bytes → Delivered in the same order they are sent

Reliable: → If the data does not make it to the other side, the application will learn about it.

Bi-directional: → Both sides (Client and Server) can send and receive.

When communication is completed, the connection is closed.

TCP Client

Client creates a Socket and perfoms a number of Socket System calls on it. - functions in the operating system

open connection on the Socket to a port on a host → Socket(...) Connect(...)

Send Data on the connection → Send(...)

Receive(...) Receive data on the connection

Close(...) Close the connection

In Java

In Java, Streams (Inputstream and Outstream) are the basic classes for byte I/O

Void write (byte, [], buffer)

int read (byte, [], buffer)

TCP Server

Single-threaded version

Socket(...)
Bind(...)
Listen(...)
Accept(...)
Receive
Send
Close

↳ Single-threaded version

Socket(...)

bind(...) ↳ Associate port number to the Socket

listen(...) ↳ Welcoming Socket

Accept(...) ↳ Waiting for TCP Connection

recv(...) Receive data on connection Socket

Send(...) Send data on Connection Socket

Close(...)

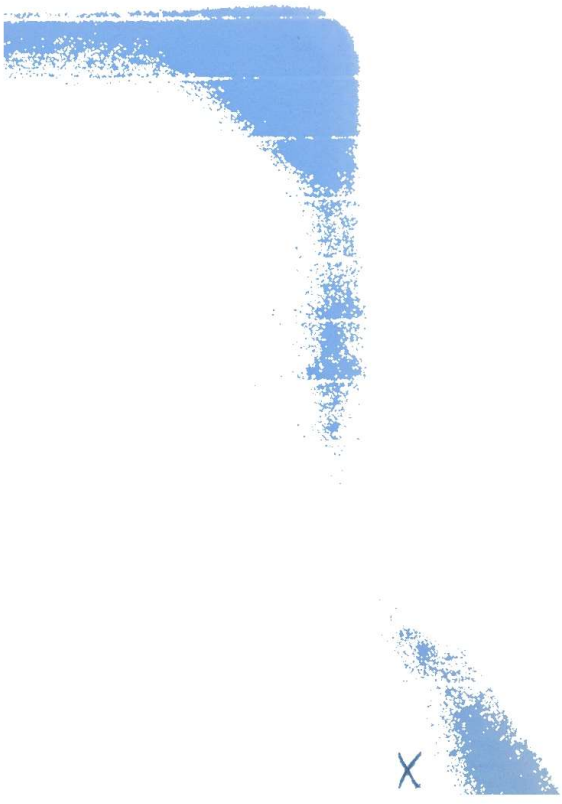
Encoding to Decode

Programming languages have data types. To transfer a string between two programs (processes) over the network, for example, we must decide how to represent the string as a sequence of bytes.

└ in other words, how to convert between bytes and data types.

↳ ASCII, UTF-8, UTF-16, MS-Windows etc

Java has several methods and constructors to convert between Strings and bytes



Föreläsning 5 - UDP

Transport Services and Protocols

- Provide logical communication between app processes running on different hosts.
- Transport protocols run in end systems. *Send side: break app messages into segment, passes to network
  RCV side: Reassemble segments into messages, passes to app layer
- More than one transport protocol available to apps Internet: TCP and UDP.

Transport vs Network layer

Network layer: Logical communication between hosts → Post Service
Transport layer: logical communication between processes. → Individual Postbox

Internet transport-layer Protocol

Reliable, in order delivery (congestion control, flow control, connection set-up) ← TCP
Unreliable, unordered delivery: no-frills extention of "best-effort" IP ← UDP
→ We cannot delay gurarenteer or bandwidth gurarentees

Multiplexing: Handle data from multiple Sockets, add transport header (Later used for demultiplexing)
Demultiplex: Use header info to deliver received segments to correct Sockets.

How demultiplexing works

Host receives IP-datagram: Each datagrams has source IP-address and destination IP
Each datagram carries one transport-layer segment
Each segment has source and destination port number

- Host uses IP-addresses and Port numbers to direct segment to appropriate Sockets.

UDP: User Datagram Protocol [RFC 768]

no frills" "bare bones" Internet transport protocol. "best service" service UDP segments may be lost or delivered out-of-orde app

- Connectionless: No handshaking between UDP sender & receiver. Each UDP segment handled independently of others.

UDP → Streaming multimedia apps, DNS, SNMP

Why UDP? → 1) No connection establishment
2) Simple: no connection state at sender or receiver
3) Small header size
4) No congestion control: UDP can blast away as fast as desired

UDP checksum Goal: Detect "errors" in transmitted segment

Sender: 1) Treat segment contents, including header fields, as a sequence of 16-bits integers

2) Checksum: addition of segment contents

3) Sender puts checksum value into UDP checksum field

Receiver: 1) Compute checksum of received segment

2) Check if computed checksum value equals checksum value — no- error detected; Yes- no error detected but not all cases covered

Reliable data transfer: Getting started

rdt_send(): called from above, Passed data to deliver to receiver upper layer

deliver_data(): Called by rdt to deliver data to upper

udt_send(): ——— unreliable channel ——— rdt_rcv()

*Called by rdt, to transfer packet over unreliable channel to receiver

We will: incrementally develop sender, receiver sides of reliable data transfer Protocol (rdt)

Consider only unidirectional data transfer → but control info will flow on both directions!

Use finite state machine to specify sender & receiver

RDT 1.0: Reliable transfer over a reliable channel

Underlying channel perfectly reliable: 1) No bits errors 2) No loss of Packets

Seperate FSMs for sender and receiver: Sender sends data into underlying channel & receiver reads data from underlying channel.

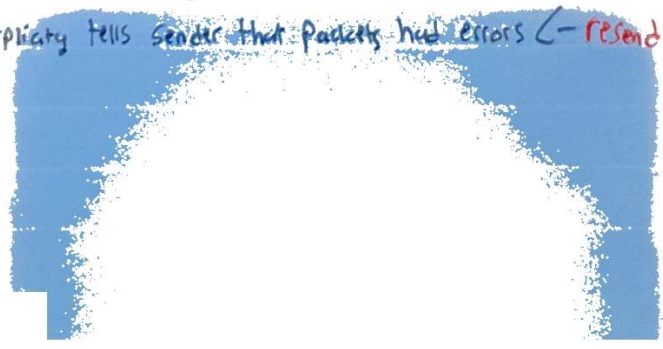
rdt2.0: Channel with bit errors

Underlying channel may flip bits in packets: Checksum to detect bit error

The question: How to recover from errors::

↳ ACKs: receiver tells Sender that PKT received OK

↳ NAKs: Receiver explicitly tells Sender that Packets had errors <- resend Packet



RDT 2.0 has a Fatal Flaw

What happens if ACKs/NAKs corrupted?: Sender doesn't know what happened at receiver. Can't just retransmit possible duplicate.

↳ Handling duplicates: Sender retransmits current packet if ACK/NAK corrupted. Sender adds sequence number to each packet.

↳ Receiver discards duplicate packet.

Stop & wait: Sender sends one packet then waits for receiver response

RDT 2.2 a NAK-free Protocol

↳ Same functionality as rdt2.1 using ACKs only. Instead of NAK, receiver sends ACK for last packet received OK

↳ receiver must explicitly include sequence number of packet being ACKed.

duplicate: ACK at sender results in same action as NAK: retransmit current packet.

RDT 3.0: Channels with error and loss

New assumption: underlying channel can lose packets.

Approach: Implements a countdown timer.

Performance of RDT 3.0

↳ rdt3.0 is correct but performance stinks Eg 1Gbps link, 15ms propagation delay, 8000 bit packet

$$D_{trans} = \frac{L}{R} = \frac{8000}{10^9} = 8 \cdot 10^{-6}$$

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{0,008}{30.008} = 0,00027$$

↳ if RTT = 30msec 1kB pkt every 30msec: 33kB/sec

Pipelined Protocols

↳ Sender allows multiple "in-flight" yet-to-be-acknowledge pkts.

Föreläsning 6

TCP- is a connection-oriented Protocol. A TCP connection is a full duplex connection between exactly two end-point

Broadcast and multicast are not applicable to TCP.

TCP provides a reliable byte stream service. A stream of 8-bit bytes is exchanged across TCP connection.

↳No record markers inserted by TCP. The receiving (reading) end cannot tell what sizes the individual writes were at the sending end. TCP decides how much data to send (not the application); each unit is a Segment.

Lots of applications have been implemented on top of TCP.

How TCP Provides Reliability

- TCP breaks application data into best sized chunks (segments)
- TCP maintains a timer for each segment, waiting for ack
- TCP retransmits segment if ack is not returned before time-out
- When TCP receives data it sends an Ack back to sender.
- Recognises duplicate data segment
- Provides flow control. Finite buffer at each end of connection

→ TCP has 20 bytes header, TCP Header fields

Src and Dst port numbers: identify sending/receiving application

Sequence number: Identifies the byte in the stream that the first byte of this segment represents. Wraps that around $2^{32}-1$

Flags: 6 Flags determining the purposes and contents of the segment

Windows Size: Defines the number of bytes

Checksum: Covers header & data. The calculation & verification is important

TCP Flags

Ack - Acknowledge number valid — note. Checksum för man inte hamnar över

RST - Reset the connection

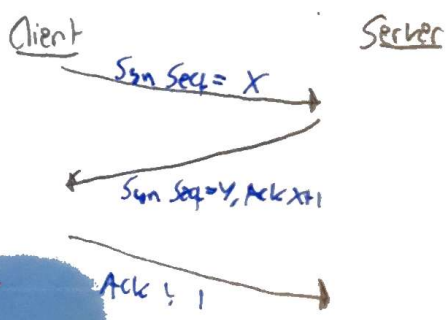
Syn - Synchronize numbers to initiate connection

Fin - Sender is finished sending data

URG - Urgent Pointer Valid. *Set when Sender wants receiver to read data urgently

Psh - Receiver should immediately pass the data to application

TCP Connection Establishment



Tcp uses a three-way handshake to establish a connection. 3-way connection.

- Guarantees both sides ready to transfer data
- Allows both sides to agree on initial sequence numbers
- Initial sequence number must be chosen such that each incarnation of a specific TCP Connection between two-end Points has different ISN

Maximum Segment Size

- MSS is the largest chunk of data TCP will send to the other side.
- MSS can be announced in the options field of the TCP header during connection establishment.
- If MSS is not announced, a default value of 536 is assumed.

Flow Control

- Flow control defines the amount of data a source can send before receiving an ack from receiving
  - Which makes sure it aint slow nor overwhelmed with data
- Uses a Slided window Protocol to accomplish Flow Control.

Sliding Windows

Receiver: Offered window + ack data sent and what it is prepared to receive.

Sender: The receiver sends a window update with a non-zero offered window size.

Silly Windows Syndrome

- Serious problems can arise in the sliding windows operation when
  - Sending application creates data slowly or receiving applications consumes data slowly

Bandwith-Delay Product

Calculate the capacity of the "pipe" as

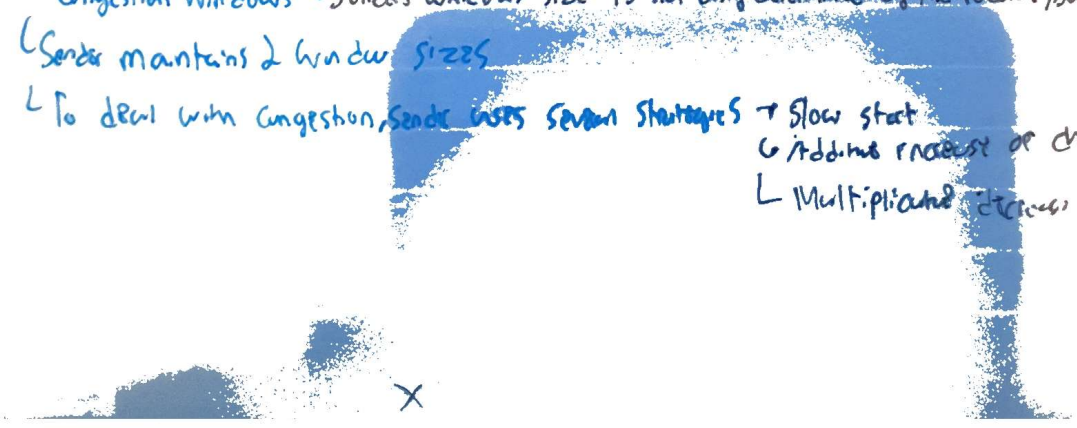
Capacity(bits) = bandwidth(bit/sec) x RTT(sec)

Regarding Flow Control

- The src does not have to send a full window's worth of data. Window size can be increased/decreased by dst. The dst can send an ACK at any time.

Congestion Control

- Each router along the way has buffers, that stores the incoming packets before they are processed and forward.
- If packets are received faster than they can be processed, congestion might occur and packets could be dropped.
  - Congestion Windows → Sender's windows size is not only determined by the receiver, but by the congestion in the network
  - Sender maintains 2 window sizes
  - To deal with congestion, Sender uses several strategies → Slow start
    - Additive increase or decrease
    - Multiplicative decrease of cwnd



Föreläsning 7 - Network layer

Network layer

→ Transport segment from sending to receiving host. On sending side encapsulates segment into datagrams. On receiving side, delivers segments to transport layer. Network layer protocols in every host, router. Router examines header fields in all IP datagrams passing through it.

↳ Two key network-layer function

Forwarding: Move packets from router's input to appropriate output → Move Car

Routing: Determine route taken by packets from source to destination → GPS in the car

Datagram forwarding table: To avoid searching $2^{32}-1$ ip adresses, the router has a table with destination Adress range. The closest to it, gets the data packet. This is called longest prefix match

IP-fragmentation/reassembly

Network links have different max transmission unit, largest possible link-level frame.

IP datagram needs to be divided ("fragmented") if it is larger than link MTU

↳ One datagram becomes several datagrams, reassembled at final destination. IP header bits used to identify fragments.

IP-addressing: Introduction

IP (version 4) adress: 32 bit identifier. Never larger than 255. Dotted "quad" notation

interface: Connection between host/router and physical

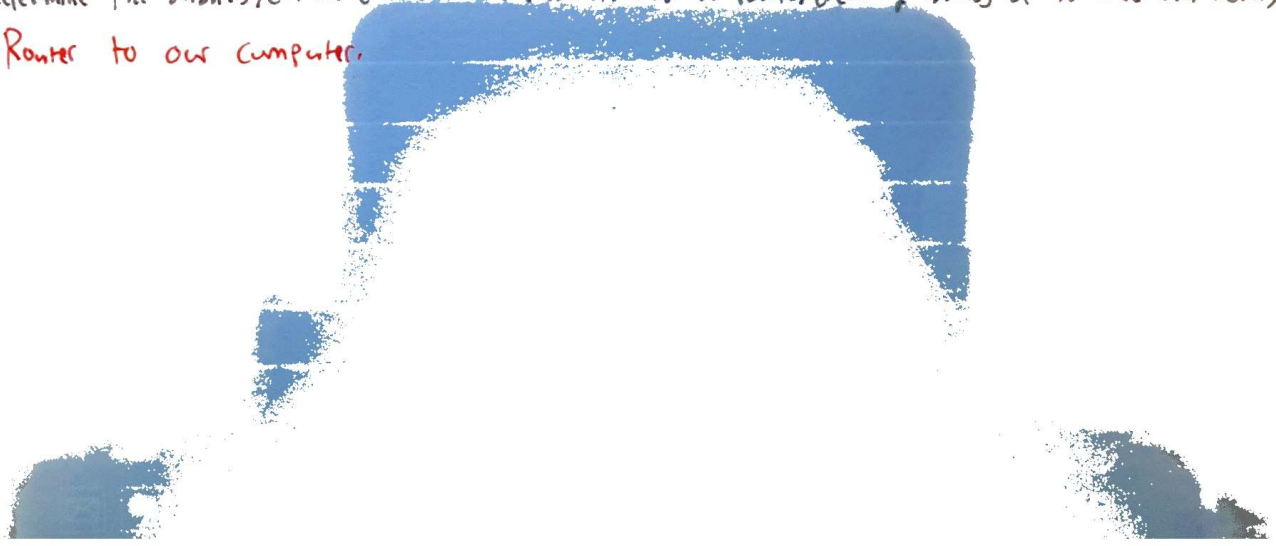
Subnet.

IP-address: Subnet part - high order bits, host part - low order bits

↳ Subnet?: Device interfaces with same Subnet part of IP-address. Can physical reach other without intervening router.

Recipe: To determine the subnets, detach each interface from its host or router, creating islands of isolated networks

↳ Router to our computer.



IP-addressing CIDR

↳ Subnet portion of address of arbitrary length.

↳ Address format: a.b.c.d/x where x is number of bits in subnet portion of address.

IP-addresses: How to get one

↳ Hard-coded by system admin ← You did this manually. Old System

↳ DHCP: ← dynamically get address from server

DHCP

Goal: Allow host to dynamically obtain its IP address from network server when it joins network

↳ Can renew its lease on address in use. Allow reuse of addresses ⊕ Support for mobile networks

1) Host broadcasts "DHCP" discover" msg — Discover
2) DHCP server responds with "DHCP offer" msg — offer
3) Host requests IP-address: "DHCP request" msg — request
4) DHCP server sends address "DHCP ack" msg — Ack

DHCP use UDP

DHCP: more than IP address

↳ address of first-hop router for client

↳ Name and IP-address of local DNS server

↳ Network mask (indicating network versus host portion of address.

NAT: network address translation

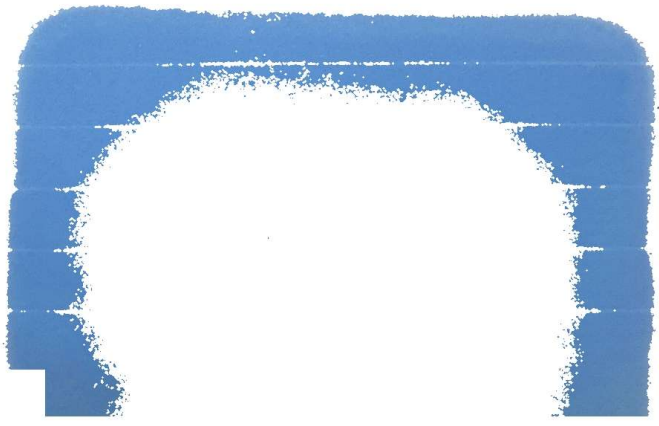
motivation: local network uses just one IP address as far as outside world is concerned:

→ Range of addresses not needed from ISP: just one IP address for all devices

→ Can change address of devices in local network without notifying outside world.

→ Can change ISP without changing addresses of devices in local network

→ Devices inside local net not explicitly addressable, visible by outside world.



implementation: NAT Router must:

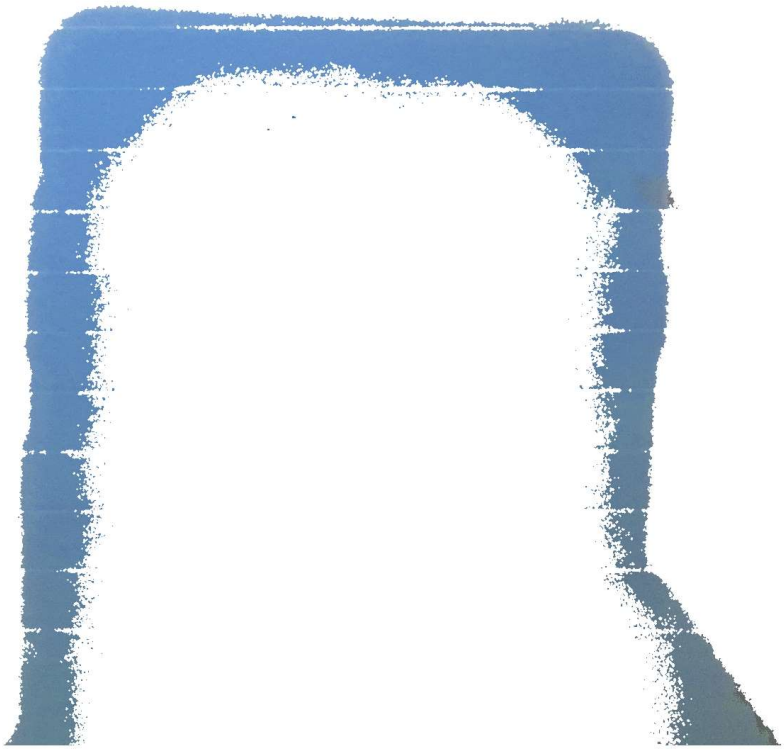
Outgoing datagrams: replace (Source IP address, port number) of every outgoing datagram to (NAT IP address, new port number)
↳ remote clients/server will respond using (NAT IP address, new port number) as destination addr.

Remember (in NAT translation table) every (Source IP-addres, port number) to (NAT IP address, new port number) translation pair

Incoming datagram: replace (NAT IP address, new port number) in dest fields of every incoming datagram with corresponding (Source IP address, Port number) stored in NAT table.

→ 16-bit port-number fields: → 65,535 simultaneous connections with a single LAN-side address!

→ NAT is controversial: → Routers should only process up to layer 3
→ Violates end to end argument → must be taken account by app designer
→ Address shortage should instead be solved by IPV6.



Föreläsning 8 - Network Layer

NAT traversal Problem

How reach a server on the LAN side?

- Solution1: Statically configure NAT to forward incoming connection requests at given port to server.
- Solution2: Universal plug and play. Allows NATed host to learn public IP address and add/remove port mappings
- Solution3: Relaying (Skype). NATed client establishes connection to relay, external client connects to relay
  - relay bridges packets between to connection

Basic Routing

- Basic "manuel" approach — Next-hop routing
- This approach works only for small IP-networks — Logical (IP) address
- We need to support dynamic large networks, — Static Tables

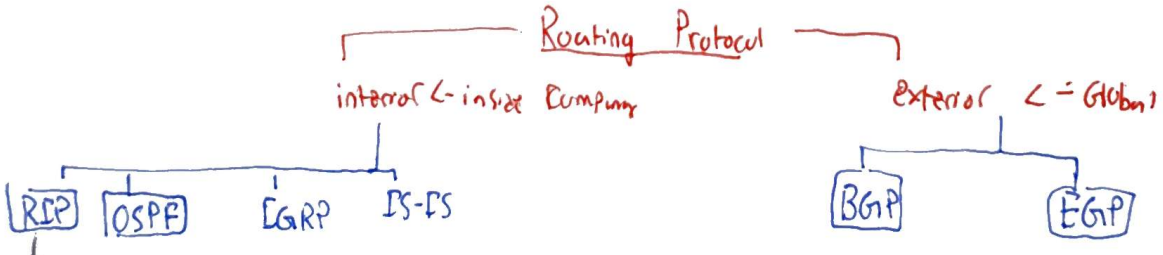
Reachability and Metrics

- The most fundamental functionality in a dynamic routing protocol: Find the "best path" to a destination.
- Two algorithms in use to find best path: Distance-Vector or Dijkstra
  - But what is the best path? - Interior routing: typically number of hops or bandwidth
    Exterior routing: Business relations - peering

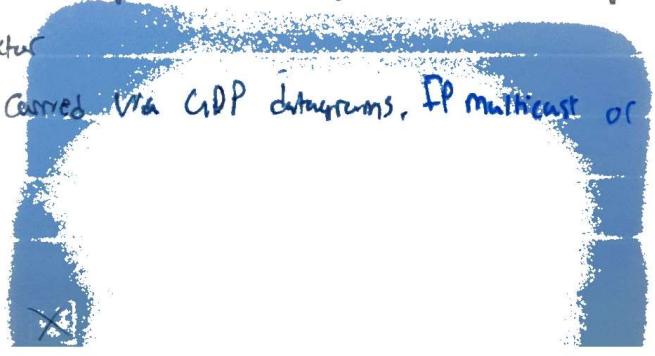
Metrics - Number of hops, Bandwidth, Delay, Cost, Load and policies

Aggregation

- Also called Summarization. The netid part of IPv4 address can be aggregated into shorter prefixes leads to shorter routing tables



- Metric is hop-counts. 1: directly connected. If you are 16 hops away then you're outside
- RIP uses distance vector
- RIP messages are carried via UDP datagrams. IP multicast or Broadcast



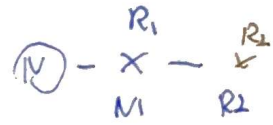
Distance Vector

→ A node advertizes its "distance vector" → A list (vector) of all nodes that the nodes knows about
→ The distance to each of them.

Advertizements are sents to neighbours only

- Each neighbour updates its routing table and sends the new distance-vectors to its neighbours

RIP Problem: Count to infinity



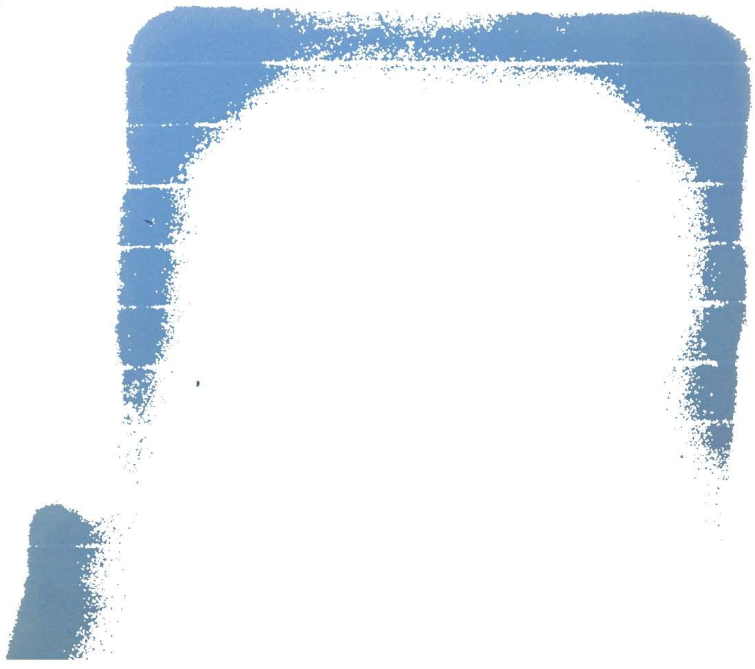
1. Initially R_1 and R_2 both have a route to N with metric 1 and 2 respectively
2. The link between R_1 and N fails
3. Now R_1 removes its route to N, by setting its metric to 16 (infinity)
4. Now two things can happens: Either R_1 reports its route to R_2. Everything is fine
5. The other alternative is that R_2, which still has a route to N, advertises it to R1. Now things start to go wrong:
   Packets to N are looped until their TTL expires!
6. Eventually (~10, 20s), R1 sends an update to R_2. The cost to N increases, but the loop remains
   Yet some time later, R_2 sends an update to R_1
7. Finally the cost reaches infinity at 16, and N is unreachable. The loop is broken

Disadvantage with RIP

- Slow convergence: Changes propagate slowly, each neighbor only speaks every 30 seconds
- Instability: - After a router or link failure RIP takes minutes to stabilize
- Hops count may not be the best indication for which is the best route
- The maximum useful metric value is 15: Network diameter must be less than or equal to 15
- RIP uses a lot of bandwidth: It sends the whole routing table in updates

Why use RIP

- Because RIP is generally available
- Simple to configure

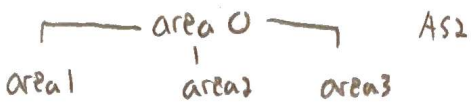


Open Shortest Path First

OSPF is a link-state protocol. - Builds link state Advertisements (LSAs)
- Distributes LSAs to other routers
- Computes delivery tree using the Dijkstra algorithm

OSPF uses IP directly

OSPF networks are partitioned into areas to minimize cross-area communication.



→ Area 0 is the backbone area. All traffic goes via the backbone

Link State Protocols

) Actively tests the status of all neighbour/links
) Build LSA from this information and propagate it to all area within the area
) Uses LSAs from other routers, Compute a shortest delivery tree typically using dijkstra algo

Advantages

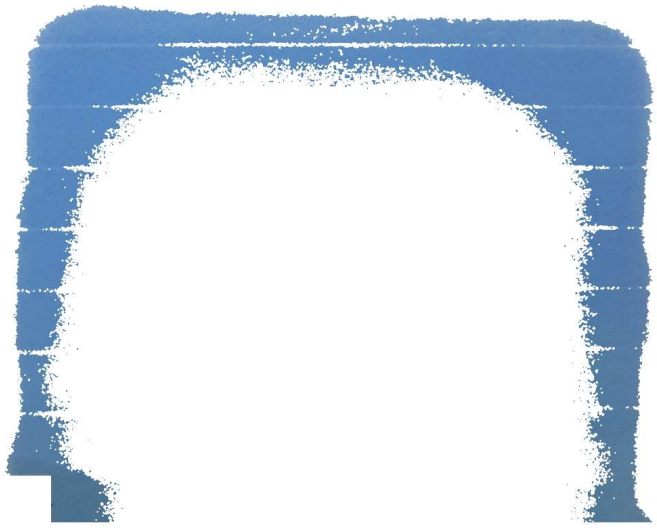
- Full topology knowledge
- Easier to trouble shoot

Disadvantage

- Uses more memory

3 Sub Protocols

) Hello Protocol → Check for neighbours, authentication, designated routers
Exchange Protocols → Exchange LSA, First get LSA headers. Then transfer actual LSAs on request
) Flooding Protocols - When links change/age. Send LSA updates to neighbour and flood recursively



Alternative to OSPF

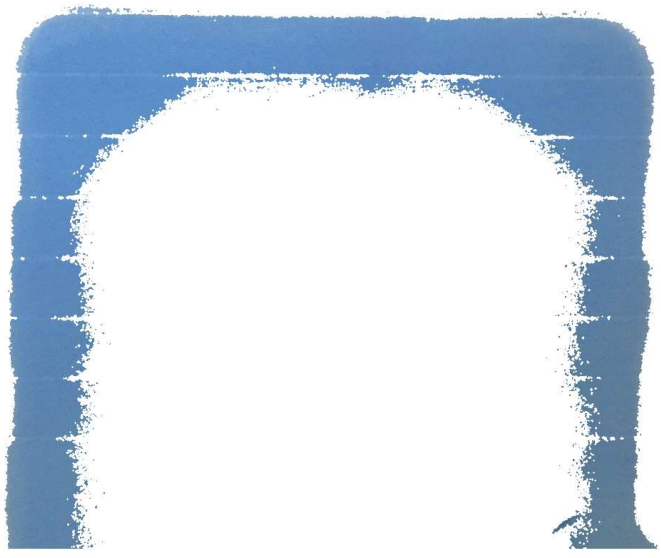
- Link-State Routing
- Originally designed for Decnet and then CLNP
- Has been stable for a long time than OSPF

Border Gate Protocol

- Inter-domain routing.
- Simple cases: uses static routing
- Main purpose: Network reachability between autonomous systems
- BGP version 4 is the exterior routing protocol used in the internet today
- BGP uses TCP; - TCP is reliable: reduces the protocol complexity
- Uses path vector - Enchancement of distance vector
- BGP implements policies - Chosen by the local administrator

Autonomous Systems

- An autonomous system is generally administered by a single entity
- An AS contains an arbitrary complex sub-structure.
- Each AS selects the routing protocol to be used within the AS
- Policies or updates within an AS are not propagated to other ASes
- An As-number is currently a 16-bit unique identifier



Föreläsning 9- Link Layer

Introduction Services Del 1

terminology

- hosts and routers: nodes
- Communication channels that connect adjacent nodes along communication path: link
  - (wired links, wireless links, LANs
- Layer-2 Packets: frame, encapsulates datagram
  - Link Layer has the responsibility of transfering datagram from one node to physically adjacent node over a link

Link layer: Context

Datagram transferred by different link protocols over different links.

↳ Ethernet on first link, frame relay on intermediate links, 802.11 on last link

Each link Protocol provides different Services

↳ e.g may or may not provide reliable data transfer over link

Analogy

Trip from Princeton to Lausanne:

limo: Princeton to JFK

Plane: JFK to Geneva

Train: Geneva to Lausanne

Tourist = Datagram

Transport Segment = Communication Link

Transportation mode = Link Layer Protocol

Travel-agent -. Routing algorithm

Link layer Services

- Framing link access
- Encapsulate datagram into frame, adding header and trailer
- Channel access, if shared medium
- "MAC" addresses used in frame headers to identify source and destination, → Medium Access Control → Different from IP-address.
- Reliable delivery between adjacent nodes (TCP)
  - We learned how to do this already in (chapter 3 → (TCP and UDP)
  - Seldom used on low bit-error link (fiber, some twisted pair)
  - Wireless link: high error rates ← Often used on places with high error rates

Link Layer Services (more)

Flow Control: Pacing between adjacent sending and receiving nodes.

Error detection: → Errors caused by signal attenuation and noise

→ Receiver detects presence of errors: → Signal sender for retransmission or drops frame

Error Correction → Receiver identifies and corrects bit errors without resorting to retransmission

Half & full-duplex → With half duplex, nodes at both ends of link can transmit, but not at the same time.

Error detection Correction Del 2

Error Detection → Receiver cannot tell by just looking at data whether data has been corrupted

D- Data protected by error detection

EC):- Error detection function

EDC:- Error Detection and Correction bits, sents together with data, Calculated as EDC = EC(D)

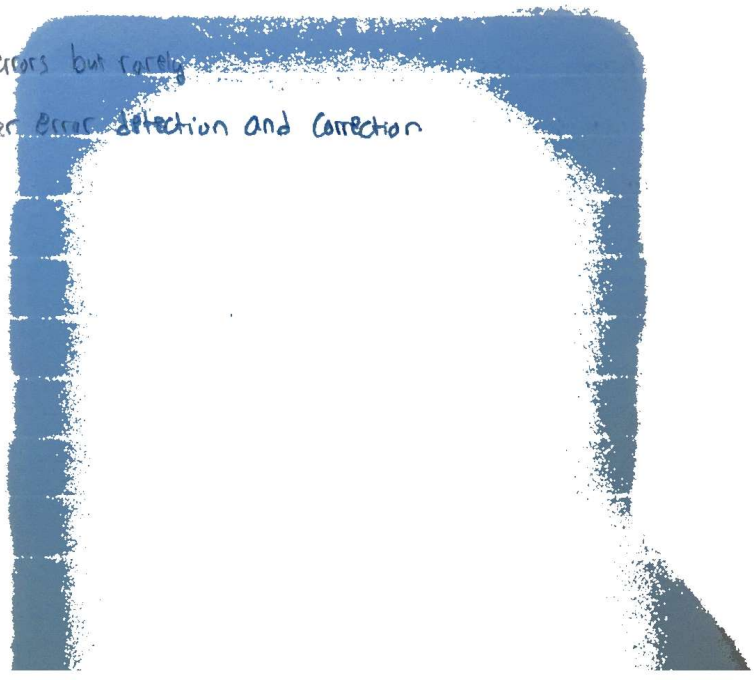
Error detection Functions

1) Even Parity	1 bit - Set so total numbers of one's is even. Detects single bit errors	Memory, Cards Disks	
2) Internet Checksum	16 bits - 16 bit sum (modular arithmetic)	IP, TCP, UDP	← Implemented in software
3) CRC-32	32 bit - Cyclic code (Polynomial division). Detects all burst errors up to 32 bit long.	Ethernet, Wireless LAN	

* Error detection not 100% reliable

→ Protocols may miss somes errors but rarely

→ Larger EDC field yields better error detection and correction



Multiple access Protocols Del 3

Multiple access link → A single link that many share

Broadcast (shared wire or medium) → Old-fashion Ethernet, upstream HFC, 802.11 wireless LAN

→ Only one can send at a time. How coordinates sending? - MAP. MAC - multiple Access Control

An ideal multiple access Protocol

Given: Broadcast Channel of rate R bits

Desiderate: 1. When one node wants to transmit, it can send at rate R

2. When M node want to transmit, each can send at average rate R/M

MAC protocols: three broad classes

Channel Partition → Divide Channel into smaller "pieces" (time slots, Frequency, codes)

→ Allocate piece to node for exclusive use

→ TDMA - (Time, Division, Multiple Access), FDMA (Freq), WDMA (wavelength) CDMA ("Code)

Random Access → Channel not divided → Collision could happen → Ethernet, IEEE 802.11

→ Recover from Collisions

Taking Turns → Nodes takes turn → but nodes with more to send can take longer turns

→ Coordinated access

→ Bluetooth, token ring, IEEE 80211 PFC (Point, Coordinate, Function

Random Access Protocol:

→ When node has packet to send

└ Transmit at full Channel data rate R

└ No a priori cordinate among nodes

→ Two or more transmitting nodes → Collision

→ Random access MAC protocols specifies

└ How to detect collisions

└ How to recover from collisions (i.e via delayed retransmissions)

→ Examples of random access MAC protocols:

└ Aloha, Slotted Aloha, CSMA, CSMA/CD, CSMA/CA

CSMA (Carrier Sense Multiple Access)

CSMA: Sense (listen) before transmit: if channel sensed idle: transmit entire frame
└ if channel sensed busy, defer transmission (wait)

CSMA Collisions

Collisions can still occur: Propagation delay means two nodes may not hear each other's transmission

Collision: Entire packet transmission time wasted + distance and propagation delay play role in determining collision probability

CSMA/CD Collision Detection

+ Collision detected within short time
+ Colliding transmissions aborted, reduced channel wastage
- Collision Detecting: → Easy in wired LANs: measure signal strengths, compared transmitted and received signal
→ Difficult in wireless LANs: Received signal strength overwhelmed by local transmission strength
Human analogy: The polite conversationalist

LANs (4)

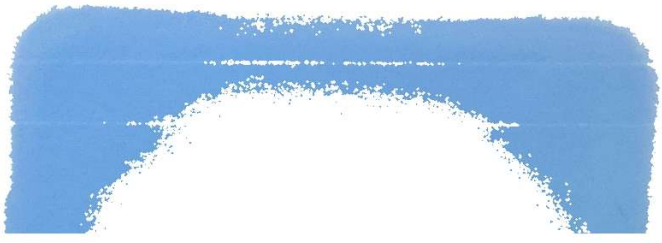
MAC addressing and ARP

+ 32 bit (128-bit) IP address - Used for layer 3 (network layer)
- Network-layer address for interface

- MAC (or LAN or physical or Ethernet) address
- Function: used "locally" to get frame from one interface to another physically-connected interface (same in network, IP-address space)
→ Each adapter on LAN has unique LAN address

LAN addresses

+ MAC address allocation administered by IEEE → Manufacturers buys portion of MAC address space
+ MAC address bound to specific hardware (adapter)
Analogy: Mac address like person's name
IP address like postal address
→ If moving host changes IP address (but same Mac address)



ARP Protocol: Same LAN

1. A wants to send data to B
   - B's MAC address not in A's ARP table
2. A broadcast ARP query, containing B's IP address
   - To broadcast MAC address = FFF... F, all nodes on LAN receive the ARP query
3. B receives ARP query, replies to A with its (B's) MAC address
   - Frame sent to A's MAC address
4. A caches (saves) IP-to-MAC pair to its ARP table until information becomes old (times out)
   - Soft state: information that times out (goes away) unless refreshed
5. ARP is "plug and play"
   - nodes create their ARP table without config or intervention from network administrator

Ethernet Switch

Link-layer device: takes an active role

- Store and forward Ethernet frames
- Examines incoming frame's MAC address, selectively forward to one, or more links
- When frame is to be forwarded on segment, uses CSMA/CD to access segment
  - Transparent: Hosts are unaware of presence of switches
  - Plug and play: Switches configure themselves

Switch: Multiple Simultaneous transmissions

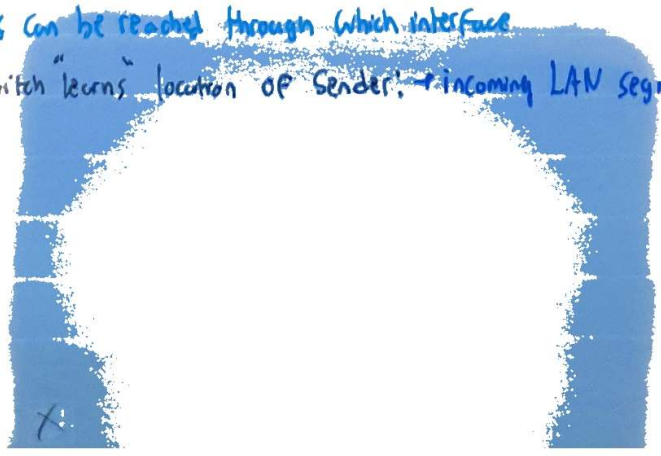
- Each host has dedicated, direct connection to switch
- Switches buffer packets

Ethernet protocol used on each incoming link, but no collisions, full duplex

Switching: A to A and B to B can transmit at the same time out collisions

Switch: Self learning: Plug n Play

- Switch learns which hosts can be reached through which interface
  - When frame received, switch "learns" location of sender: incoming LAN segment



Switches vs Router

Both are store-and-forward

Routers: network-layer devices (examine network-layer headers)

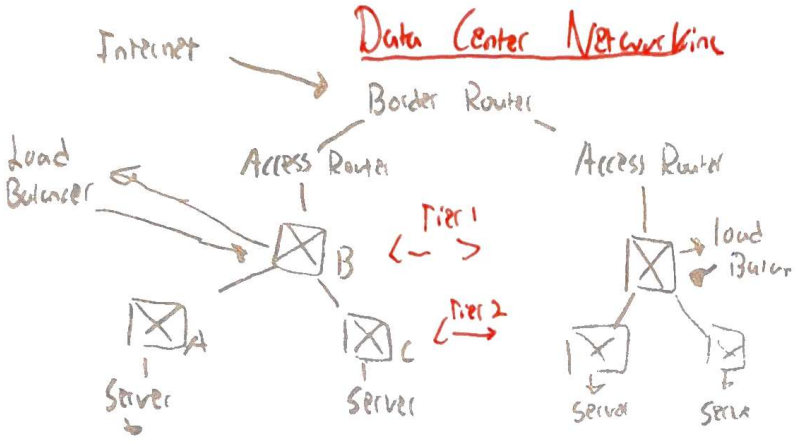
Switches: Link-layer devices (Examines link-layer headers)

Both have forwarding tables

Routers: Compute tables using routing algorithms, IP-addresses

Switches: Learn forwarding table using Flooding, learn MAC addresses

Data Center Networking



Load Balancer: application-layer routing.

- Receives external client requests
- Directs workload within data center
- Returns result to external client

- Rich interconnection among switches Roks.

A day in the life of a web request (gör själv)

Föreläsning 10 - Wireless and Mobile Networks

→ Two important (but different) challenges.

Wireless: Communication over wireless link

Mobility: Handling the mobile user who changes point of attachment to network.

Elements of a wireless link (1)

Wireless host: • Runs applications.

Base Station: • Typically connected to wired network

└ Relay: responsible for sending packets between wired and wireless network in its area

Wireless link: • Typically used to connect mobiles to base station

└ Multiple access protocol coordinate link access

└ Various data rates, transmission distance.

Infrastructure mode: Base station connects mobiles into wired network

Handoff: Mobile changes base station providing connection into wired network.

Ad hoc mode: └ No base station

└ Nodes can only transmit to other nodes within link coverage

└ Nodes organise themselves into a network: route among themselves

Wireless Links, characteristics (2)

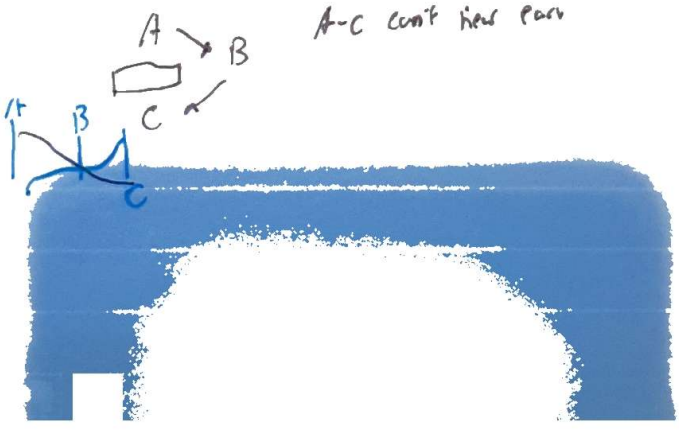
• Decreased signal strength: Radio signal attenuates as it propagates through matter

• Interfere from other sources: Standardized wireless network frequencies shared by other devices interfere as well

• Multipath Propagation: Radio signal reflects off objects, ground, arriving at destination at different times

• Hidden terminal Problem:

A-C can't hear each



• Signal Attenuation

IEEE 802.11 Wireless LANs (3)

802.11b: → 2.4-5 Ghz unlicensed spectrum. Up to 11 Mbits/sec

802.11g: → 2.4-5 Ghz range. Up to 54 Mbits/s

802.11 multiple antennae → 2.4-5 ghz range. Up to 200 Mbps

802.11 Multiple antennae multi-user: → 5 ghz range. Atleast 1000 Mbs multistation

802.11 Architecture LAN

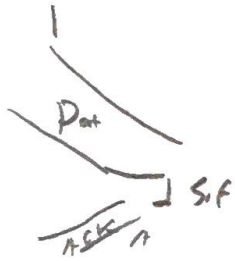
Wireless host communicates with base station. Base station = Access point (AP) internet

Basic Service Set ("aka cell") in infrastructure mode contains BSS1 — hub — BSS2

- Wireless hosts, - Access point (AP): base station

802.11 Channel, association

- 802.11b spectrum divided into 11 channels at different frequencies
  - AP admin chooses channel for AP
  - Interface possible: Channel can be same as that chosen by neighboring AP!
- Host: must associate with an AP
  - Scan channels, listening for beacon frames containing AP's name (SSID) and MAC address
  - Selects AP to associate with
  - May perform authentication
  - Will typically run DHCP to get IP address in AP's subnet

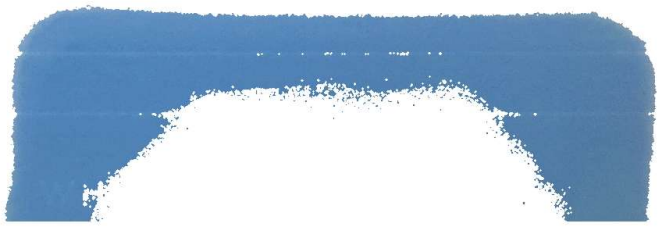


IEEE 80211 MAC Protocol: CSMA/CA

1. If Sense: Channel idle for DIFS then → transmit entire frame (no collision detect)
2. If Sense Channel busy then: → Start random back-off time → Timer counts down while channel idle → transmit when timer reaches zero → If no ACK, increase random back-off interval, repeat 2

} Sender case

If frame received ok: → Return ACK after SIF (ACK needed due to hidden terminal problem) } Receiver



Avoiding Collisions (more)

idea: Allow sender to "reserve" channel rather than random access of data frames: avoid collisions of long data frames

→ Sender first transmits small request-to-send (RTS) packets to BS using CSMA

└ RTS may still collide with each other but they are short

→ BS broadcasts clear-to-send CTS in response to RTS

→ CTS heard on the nodes

└ Sender transmit data frame

└ Other stations defer transmission