Communication Networks

Summary

Andreas Biri, D-ITET 13.07.15

1. Networking Basics

Client-Server computing: simple, dumb terminals connected to "mainframes" which compute

Peer-to-Peer computing (P2P): distributed system
- all participating computers (nodes) are peers
- peers serve both as clients and servers
- centralized node / service handles management

Web 2.0: web no longer dominated by single providers
- everybody offers content (e.g. social networks)

Cloud Computing: computing offloaded to infrastructure located elsewhere ("in the cloud")
- *"Software as a Service" (SaaS):* SW accessed via Internet
- *"Hardware as a Service" (HaaS):* physical infrastructure provided (hardware, storage, networking) as a service
- *"Platform as a Service" (PaaS):* develop SaaS / HaaS

Expectations & requirements
- *user:* access anytime, anywhere, with any device, cheap
- *programmer:* sufficient network service quality, good API
- *network designer:* efficient usability (cost & energy)
- *network provider:* manageable, maintainable, income
- *content provider:* ensure accessibility of content, tracking

Computer Network: basic & ubiquitous communication infrastructure for distributed applications ("connects")
- self-aware, self-configuring, self-monitoring, self-healing

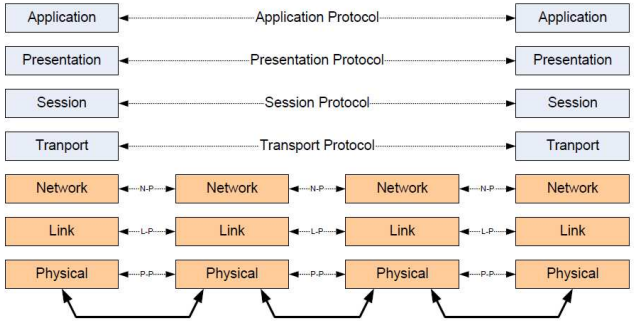
Channel: abstraction of a link
- Point-to-point vs. broadcast channel
- Properties: propagation delay, capacity, transmission errors

Possible network problems
- data transmission on a link
- naming & addressing (identification)
- resource sharing (fairness, flow & congestion control)
- resilience against failure (error handling, redundancy)
- routing
- accounting of service/resource usage, charging

Network: - set of nodes interconnected by links
- set of networks interconnected by nodes

Multiplexing: multiple logical flows over one physical link
- *Space Division Multiplexing (SDM):* distinction in space
- *Time Division Multiplexing (TDM):* distinct time slots
- *Frequency Division Multiplexing (FDM):* distinct frequency bands within the whole frequency range
- *Code Division Multiplexing (CDM):* encoded signals

OSI (Open Systems Interconnection) Model



OSI vs. TCP/IP Layers

ISO/OSI-Layers		Data container
7	Application	(Stream)
6	Presentation	
5	Session	
4	Transport	Segment
3	Network	Datagram (Packet)
2	Data Link	Frame
1	Physical	Bit

TCP/IP Layers

4	Application	
3	TCP	UDP
2	IP	
1	Subnetwork (Ethernet 802.3, WLAN 802.11)	

TCP: Transmission Control Protocol

IP: Internet Protocol

2. Multiple Access Protocols

Senders and receivers share one medium (air, cable)

Challenge to coordinate usage of shared medium
- multiplexing schemes: frequency, time, code
- distributed or centralized coordination
- pre-allocation of medium to each sender
- allocation of medium on demand

Local Area Network (LAN): TDMA, distributed, on-demand, variable frame length, Medium Access Control

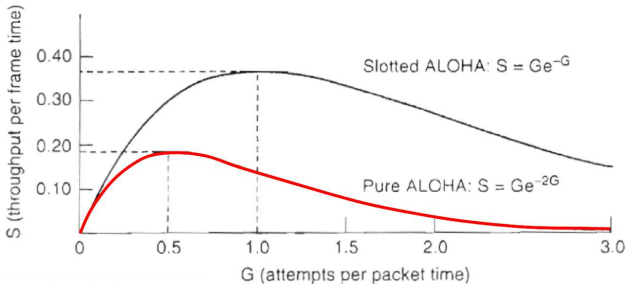
ALOHA

Central hub which *receives packets from all stations on the same frequency* and *returns acknowledgement on another*

Collision: two stations try to send simultaneously
- back off and resend after waiting a random time

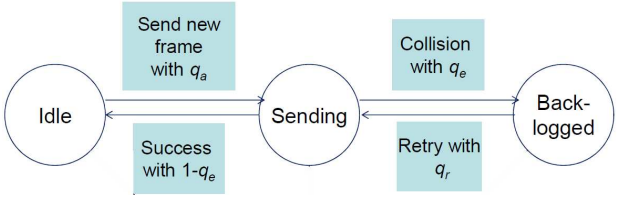
Throughput S: rate of frames without collision

Average number of generated frames $\boldsymbol{G} = g * D$
- g : average channel access rate
- D : transfer time (depends on frame length)



Pure ALOHA: $S = G\,e^{-2G}$, $S_{max} \approx 18.4\,\%\ at\ G = \frac{1}{2}$

Slotted ALOHA: $S = G\,e^{-G}$, $S_{max} \approx 36.8\,\%\ at\ G = 1$



Carrier Sense Multiple Access (CSMA)

"Listen before talk": check if channel is free before sending

Collision Detection (CD): "find out if collision occurred"

Vulnerable period: depends on length of medium

$$\tau = t_p + t_D \quad \rightarrow \quad T = 2 * \tau$$

Ethernet: CSMA / CD
- detection of collision only if frame has *minimum length*
- *jamming signal* to make sure everyone realises collision
- random wait time: *Binary Exponential Backoff* (increases)

Performance tuning of CSMA

Maximal throughput is indirectly proportional to β:

$$\beta = \frac{\tau}{m} = \frac{\tau * C}{L} \quad , \qquad C = \frac{L}{m}$$

τ	propagation delay	[s]
m	frame length	[s]
L	frame length	[bits]
C	transmission rate	[bit/s]

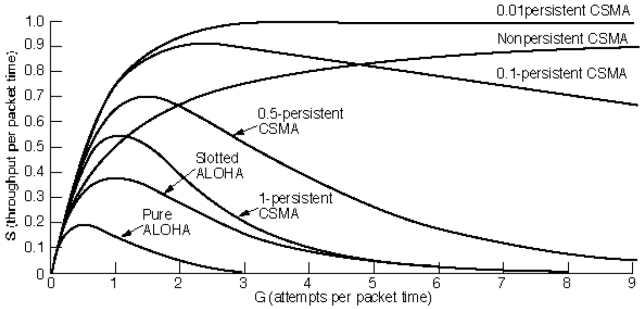
Binary Exponential Backoff: increase backoff time for each further successive collision (suggests congestion)
- c collisions → choose slot number between $0\ and\ 2^c - 1$

Persistency

Persistent CSMA	
Free:	send
Occ.:	Wait until free, then send

p-Persistent CSMA	
Free:	send with probability p
Occ.:	Wait till free, then send with probability p

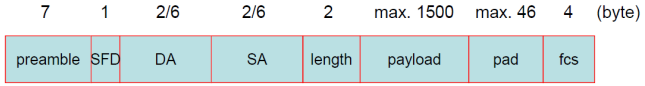
Non-Persistent CSMA	
Free:	send
Occ.:	Wait random time, then retest medium; send if free



Logical Link Control (LLC) sub-layer
- Type 1: connection-less, unreliable, all addressing modes
- Type 2: connection-oriented, reliable, flow control
- Type 3: connection-less, request / response service

Ethernet

CSMA/CD medium access control (MAC) algorithm

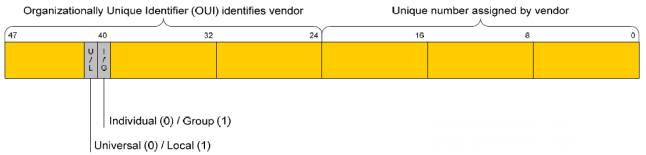


- preamble (Bit synchronization) & SFD (Byte synchronization)
- DA (destination address) & SA (source address)
- pad (to fill up a short frame)
- FCS (Frame Check Sequence; 32-Bit CRC for error detection)

MAC Address

48 bit address, globally unique (*universal*)

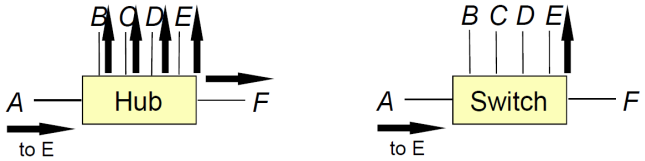
Broadcast: ff-ff-ff-ff-ff-ff (straight ones)



Hardware

Hub: repeats signals (all receive the same)

Switch: forwards frame (only one output)
- requires address examination and forwarding
- reduces CSMA/CD collision domain drastically



Network topologies: 36-37
- shared/switched backbone
- (de-)centralized workgroup segmentation)

3. Wireless LANs

	Bluetooth (802.15.1)	Wi-Fi (802.11)	3G Cellular
Typical link length	10 m	100 m	Tens of kilometers
Typical data rate	2 Mbps (shared) v4: 24 Mbps	54 Mbps (shared) 11n: 600 Mbps	Hundreds of kbps (per connection)
Typical use	Link a peripheral to a computer	Link a computer to a wired base	Link a mobile phone to a wired tower
Wired technology analogy	USB	Ethernet	DSL

Characteristics of Wireless LAN (WLAN)

+ very flexible within the reception area
+ ad-hoc networks without previous planning possible
+ no wiring necessary (buildings, robust against disaster)

- typically lower bandwidth compared to wired networks
- interference, signal attenuation, environmental influence
- many restrictions & incompatibilities around the world

Infrastructure network

dedicated hardware & access points to backbone

Station (STA): terminal with access to access point

Basic Service Set (BSS): AP + served group of STAs

Access Point (AP): interface between WLAN & DS

Portal: bridge to other (wired) networks

Distribution System (DS): interconnection network

Ad-hoc network

multiple equal devices, spontaneous, no dedicated AP
direct communication within a limited range

Station (STA): terminal with access to wireless medium

Independent Basic Service Set (IBSS): group of stations

802.11 – Physical layer

- *Frequency Hopping Spread Spectrum (FHSS)*
  fast hopping: multiple freq. / bit ; *slow hopping:* multiple bits / freq.
- *Direct Sequence Spread Spectrum (DSSS)*: spread frequencies by **chipping** (XOR bit with chipping sequence)
- *Infrared:* 850-950nm, 10m range

Wireless MAC Layer

Problems which prevent CSMA/CD for wireless medium:

Hidden Node Problem: don't see all nodes of network; therefore, collisions can occur which are not noticed (can be solved with Request-to-Send (RTS) packages)

Exposed Node Problem: one access prevents all further communication, even though would be possible

MAC Layer operating modes

Distributed Coordination Function (DCF)
- contention-based medium access control
- collision avoidance: sense channel & evaluate RTS packet
- if free for one Inter-Frame Spacing (IFS), can start sending
- if busy, wait for free duration, then wait back-off timer
- acknowledgment if correct, else automatic retransmission

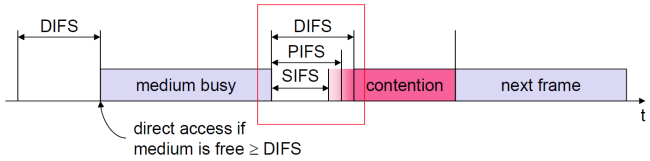
Point Coordination Function (PCF)
- Access point as master, client station as slave
- AP establishes contention-free period & polls clients during this period himself (reserved for him)

Network allocation vector (NAV): specifies earliest point a station can try to access medium again by regarding RTS

Request-to-Send (RTS) & Clear-to-Send (CTS): reservation

Beacon: frames sent at regular intervals, contain management information (timestamp, BSSID etc.)

MAC Layer access priorities



SIFS (Short Inter Frame Spacing): Highest priority (ACK, CTS)

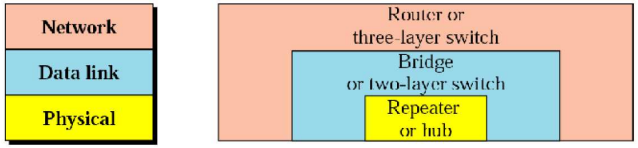
PIFS (PCF Inter Frame Spacing): Medium priority, for PCF

DIFS (DCF Inter Frame Spacing): Low Priority, asynchronous

4. Internetworking

4.1 Extended Local Area Network

Simply extending an ordinary LAN is no option
- declining performance, larger collision domain
- lower reliability and security



Repeater/Hub (layer 1): signal regeneration / repetition
- larger collision domain, all devices use same speed

Bridge/Switch (layer 2): connect LANs, store-and-forward
- increases broadcast domain, decreases collision domain
- forwards frames (needs table), can filter packets

Router (layer 3): routing at the network layer (IP)

Table mapping / forwarding table

Self-Learning / Transparent Switch

- extract *source address* and add port number & SA
- examine *destination address* and check forwarding table
  - if it's not there, broadcast and wait for response

Configuration mode: switch requests addresses of devices

Spanning Tree

Eliminate loop-problem which can occur with self-learning

- each switch gets a unique 48-bit ID
- exchange "configuration bridge protocol data units"
  - calculate costs between two switches
  - select one single switch as **root**
  - elect **designated switch** to forward root traffic
  - select ports to be included in the spanning tree

Spanning Tree Algorithm

1. Search node with smallest ID and select as **root switch**

2. For each switch, search the fastest way to the root (◊) and keep your neighbours up-to-date (if equals, lower ID wins)

3. On each LAN segment, select a **designated bridge** (the one with the lowest costs / if equal smallest ID) and mark the corresponding port as **designated port** (◊◊) (root: all)

4. Only forward frames on marked ports (rest is blocked)

41.13.90 : (root ID).(#hops away).(sender ID / neighbour)

Failure Management

Configuration messages sent periodically, maximal age 20s
- discarded if no new message arrive with max age
- node recalculates algorithm without expired message

Topological changes lead to
- temporary loss of connectivity
- temporary loops when packets multiply (BAD; so switches wait with reconfiguration from blocked to forwarding)

Virtual LAN (VLAN)

Used to divide LAN into different parts
- create virtual workgroups
- keep broadcasts isolated (no "broadcast storm")

Grouping:
- by port number
- by MAC address
- by tagging frames with a VLAN number (also span VLAN over several switches)

For **frame tagging**, tag header is added to Ethernet header
- 12-bit VLAN ID allows up to 4096 VLANs

4.2 Circuit & Packet Switching

Space switch: forward on different parallel paths
Shared Memory Switch: buffer incoming and distribute
Time switch: input written on ring, output ports will read

Circuit Switching

A (physical) circuit is established between two end-points
- resources are dedicated to the connection (exclusively)
- example: Public switched telephone system
- done with TDM or FDM

Characteristics of Circuit Switching
+ Guaranteed bandwidth: predictable performance
+ Simple abstraction: reliable, no lost/out-of-order package
+ Simple forwarding: only based on time slot / frequency

- wasted bandwidth: no mutual usage of the channel
- blocked connections: can only handle so much traffic
- network state: network must save information

Packet switching

Application stream is broken into packages which are operated as separate units and given a unique identity
- method for resource sharing in networks

Virtual Circuits (VC)

Connection-oriented: route determined at set-up and kept
- only path known; packets are still transported separately

Virtual Connection Identifier (VCI): used for forwarding
- carried in packet header for identifying a packet on a link

Datagrams

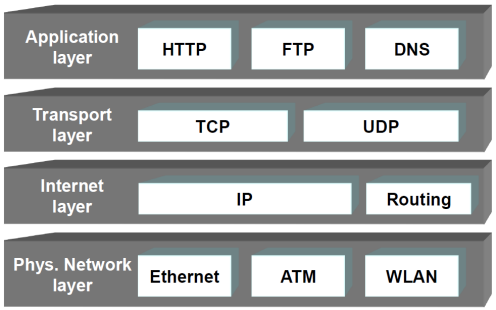
All packets contain both *source & destination address*
- *connection-less service*: forwarded according to DA
- robust against router failures

Incremental forwarding: routers only know "next hop"
Source routing: entire route is explicitly stored in packet

4.3 Internet Protocol (IP)

IP uses packet switching with datagrams, which implies connection-less service with router failure tolerance
- connectionless, unreliable service
- no guarantees for: delivery, reliability, throughput, delay
- **best effort service** ("best effort is enough")

Separation of network and application services: "dumb" core with intelligence on the edge devices (applications)



IPv4 Addresses

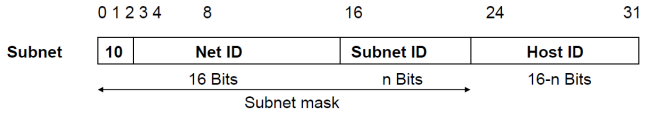
identifies **network interface** (host can have multiple)
- 32 bit addresses which offers 4'294'967'296 addresses
- hierarchically (network number + host number)

Class-based routing

Class A	0	netid				hostid				/8	0.0.0.0/8 - 127.0.0.0/8
Class B	1	0	netid				hostid			/16	128.0.0.0/8 - 191.0.0.0/8
Class C	1	1	0	netid				hostid		/24	192.0.0.0/8 - 223.0.0.0/8
Class D	1	1	1	0	multicast address						224.0.0.0/8 - 239.0.0.0/8
Class E	1	1	1	1	reserved						240.0.0.0/8 - 255.0.0.0/8

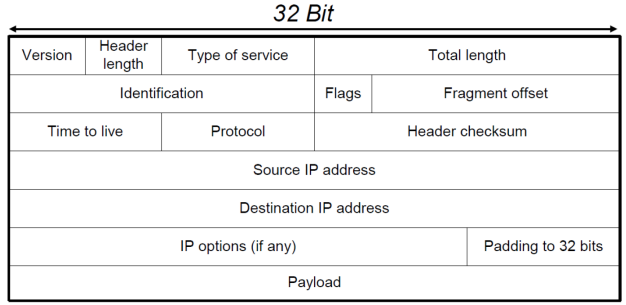
Class-less inter-domain routing (CIDR)
divide class A/B/C addresses into own subnets with *"Subnet ID"* between *netid* and *hostid*

Subnetting: network-internal addressing of subnetworks



Example: Net 129.132.0.0, Mask 255.255.255.192 = 10 Bit Subnet

IPv4 Packet Format



Address Resolution Protocol (ARP)

Request-Response protocol for same-LAN-communication
- maps IP addresses to MAC addresses for direct sending

Request: "Who has 192.168.1.2? Tell 192.168.1.1"
Response: "192.168.1.2 is at *f0:de:f1:13:ab:b6*"

Dynamic Host Configuration Protocol (DHCP)

Clients/hosts receive configuration parameter from server
- DHCP provides an address for some "lease" time
- sets: IP address, router address, subnet mask, DNS, MTU

Maximum Transmission Unit (MTU): maximal packet size
- gives limit for transferring packets through the network
- if too small: large overhead, high chance of loss, difficult

Fragmentation: split packet and adapt to smallest MTU
- re-assembly to original packet is done at destination
- guess minimum MTU and adapt if not successful

Internet Control Message Protocol (ICMP)
- signaling protocol, logically on same level as IP
- used for error signaling (dropped package) or for control

Mobility

- *Nomadic:* user move, stop and stay attached to network
- *Roaming:* user communicate while on the move

Solution: two addresses for mobility management
- *identifying* address ("who is it?" ; from Home Agent)
- *locating* address ("where is it?" ; from Foreign Agent)

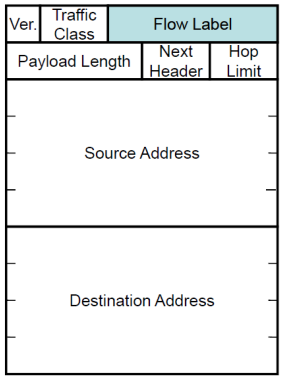
4.4 IP Version 6 (IPv6)

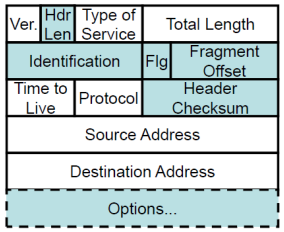
Network Address Translation (NAT)
Internally use $192.168.X.X$ addresses
Outside only sees IP of router + sequence number
- not compatible with IPSEC, as needs to change header
- as IPv6 has enough addresses, NATs are obsolete

Advantages of IPv6

- **MORE addresses** (128 bits instead of 32 bits)
- IPSEC: end-to-end, IP-layer authentication & encryption
- phase out NATs to simplify & improve performance
- elimination of "triangle routing" (FA & HA) for mobile IP

Comparison of IPv6 & IPv4 header





blue fields have no equivalent in the other version

IPv6 header is twice as long (40 bytes) as IPv4 header without options (20 bytes)

Next Header: used to add headers (each 40 bytes)
- Hop by Hop, Destination Options, Routing Header, Fragment Header, Authentication Header (AH), Encapsulating Security Payload (ESP), Upper Layer Header

Changes from IPv4 to IPv6

- Address length from 32 bits → 128 bits
- Time To Live (TTL) → Hop Limit
- Protocol → Next Header
  - many options now in dedicated, own headers
- no more checksums, as already implemented in link layer (routers don't have to adjust TTL anymore all the time)

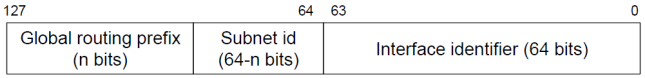
Addressing Types

Unicast: one to one
Multicast: one to many
Anycast: one to any ("any one of you can do it")

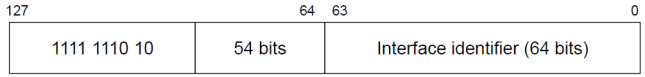
Always multiple addresses per interface with at least one being link-local; can be marked *deprecated* by router

Unicast Addresses

Global Unicast: allocations currently only from $2000::/3$



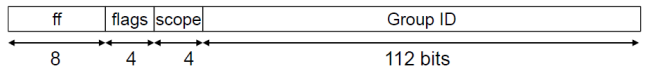
Link-local Unicast ($fe80::/10$): like local addresses in IPv4



Unique Local Unicast ($fd00::/8$)

First 40 bits are random bits; probably unique address

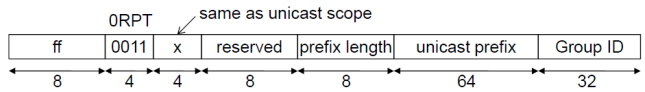
Multicast Addresses ($ff00::/8$)



$ff02::1$: all hosts $ff02::2$: all routers

Unicast-Prefix-based Multicast Prefix

Global unique multicast prefix for each $/48$ … $/64$ prefix



Address acquisition

- manual configuration
- DHCPv6
- derive interface ID from MAC address
- pseudo-random generation of interface ID (client privacy)

Neighbour Discovery

Neighbour Discovery replaces IPv4's ARP by ICMPv6

NDP sends *neighbor solicitation* (cf. ARP "request") and receives *neighbor advertisements* (cf. ARP "response")

Router advertisement: announces prefix periodically
- hosts learn about available routers & prefixes
- includes available configurations & infos such as MTU

IPv4 to IPv6 transition strategies

Dual Stack Operation

Applications can use both IPv4 & IPv6 simultaneously
- DNS contains both addresses

Tunnel Broker

- IPv6 over IPv4 tunnel to a PoP (*point of presence*)

6in4 – Protocol 41

- specifies how to put an IPv6 packet inside IPv4 (acts as if it were a new transport layer)
- cannot cross NATs, as NATs work on transport layer
- tunnels are not authenticated (easily tricked)

6to4

- uses protocol 41 / static IPv6 tunnels
- creates globally unique IPv6 addresses from IPv4

$2002:aabb:ccdd::/48$, $aabb:ccdd\ is\ IPv4$

6rd - IPv6 Rapid Deployment

- similar to *6to4*, but uses ISP specific prefix instead of public which solves some problems of *6to4* and gives the ISP more control over its traffic

AYIYA – Anything in Anything

- tunnel IPv6 inside IPv4/UDP and signs these packets
- solves problem of *6in4* (works with NATs + authenticated)

5. Internet Routing

Routing: "process of selecting a path for transmission"
- control plane (algorithm, preparation)
- individual routers *creating* a forwarding table

Forwarding: directing data packet to an outgoing link
- data plane (process, execution)
- individual routers *using* a forwarding table

Line Cards (Interface Cards, HW)
- Interfacing: Physical Link, Switching fabric
- Packet handling: forwarding, adjusting TTL, buffering

Computing paths between routers requires
- router to use to reach a destination prefix
- outgoing interface to use to reach that router

Computing the Shortest Path / Dijkstra

Given network topology with link costs (hop-count, weight)

Dijkstra's Algorithm
- after *k* iterations, know least-cost to *k* nodes (converges)

S: nodes whose least-cost path definitely known (initially only source node, one added per iteration)

D(v): current cost of path from source to node *v*

```
1  Initialization:
2    S = {u}
3    for all nodes v
4      if (v is adjacent to u)
5          D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9    find w not in S with the smallest D(w)
10   add w to S
11   update D(v) for all v adjacent to w and not in S:
12       D(v) = min{D(v), D(w) + c(w,v)}
13 until all nodes in S
```

Link-state (LS) Routing

each router knows the entire topography of the network

1. each router keeps track of its incident links
2. each router broadcasts the link state
3. each router computes the shortest paths (Dijkstra)

Beaconing: periodic "Hello" messages between neighbours

Flooding: node sends link-state information
- neighbours distribute packages to everyone
- **"routing by propaganda":** tell everyone what you know

Open Shortest Path First (OSPF)

- uses a designated router per broadcast domain to distribute information within the domain (minimal traffic)
- uses reliable flooding by acknowledging flooded message
- supports load-sharing across equal-cost routes

Convergence: changing topology induces wrong information / inconsistency in forwarding tables
- shorter "Hello"-times decrease convergence delay
- black holes & congestion due to wrong routing

Distance Vector (DV) Routing

"routing by rumour" : tell neighbour what you know

Bellman-Ford Algorithm with **distance vector (DV)**
- calculate least-cost path to destination
- update distances based on information about neighbours
- notify neighbours only when its DV changes
- converge to the same vector as link-state routing

Count-to-infinity problem: route through yourself

1. Link fails to another router
2. Route through neighbour, which in term still routes through you as he has not yet adapted his DV
3. In turn adjust distances as route through each other

Poison reverse: tell neighbour you are routing through that you have an infinite distance to route through him
- may solve Count-to-infinity, problems with more than 2

Routing Information Protocol (RIP)

- Distance vector protocol, send DV periodically or changes
- all links have cost 1 ; valid distances from 1 to 15
- infinity: 16 → smaller "count-to-infinity" problem

Split Horizon: never advertise through your own source

Comparison LS vs DV Routing

LS: knows entire system, complexity $O(\#nodes * \#edges)$
DV: only exchange information between neighbours

LS: relatively fast convergence
DV: convergence times vary, may have loops (count-to-infinity)

LS: can advertise incorrect *link* cost, calculates *own* table
DV: can advertise incorrect *path* cost, uses *other* tables

Similarities: - shortest-path routing (metric based)
- commonly used inside an organization (OSPF & RIP *intra*domain protocols)

Interdomain routing

Internet is divided into **Autonomous Systems (AS)**
- distinct region of administrative control
- managed by single institution (provider, company, …)
- AS hierarchy: large provider, regional provider, company
- AS numbers are 32 bits values

ASes don't want to share internal topologies & costs
- no internet-wide notion of a link cost metric
- use path vector routing

Path Vector Routing

Link-State routing (OSPF) is only used inside an AS
- high bandwidth & overhead, need to know ALL of it
- don't want to let others know my private topology

Path vector routing extends DV routing and adds:
- supports flexible routing policies
- avoid count-to-infinity problem (slow convergence)
- **advertise the entire path**

DV: send *distance metric* per destination d
PV: send the *entire path* for each destination d

Advantages of PV routing
- node can easily detect a loop (AS mentioned twice)
- node can easily discard loops & fasten path
- each node can apply local policies
    - through where do I want to go?
    - which paths do I want to advertise?

Border Gateway Protocol (BGP)
- prefix-based path-vector protocol
- policy-based routing based on paths

Incremental Protocol: improve during multiple sessions
- announcement: advertise new option to neighbours
- withdrawal: cannot reach destination anymore

BGP packages are continuously update during a session:
- Destination prefix: 128.112.0.0/16
- AS path: "7018 6730"
- Next-hop IP address: 12.127.0.121

AS Path length != Router hops (which are not included)
- enables policy-based router though

BGP runs over TCP
- only sends updates when changes occur
- TCP doesn't detect lost connectivity on its own
- Keep-alive: 60 seconds ; Hold timer: 180 seconds

BGP converges very slowly, but avoids count-to-infinity

Interior Gateway Protocol (IGP): maps egress point to outgoing link (used to compute paths within the AS)

Business relationships

Customer-Provider
- customer needs to be reachable from everyone
- customer does not want to provide transit service

Peer-to-Peer (between customers)
- AS exports only customer routes to a peer
- AS exports a peer's route only to its customer

Multi-Homing: use multiple providers (reliability, power)

Stub AS: no transit service, connect to upstream providers

Import policies
- Filter unwanted routes from neighbour (not own traffic)
- detect configuration mistakes & attacks
- Manipulate attributes to influence path selection

Export policies
- filter routes you don't want to tell your neighbour (P2P)
- manipulate attributes to control what they see
  (e.g. AS prepending to make path look longer)

6. Software-Defined Network (SDN)

Key to success of Internet: layers & ease of use

Data plane: packet processing & delivery (forward, filter)
Control plane: distributed algorithms, compute routes

SDN is about the design of network control (routing etc.)
- **separates the control-plane from the data-plane**
- provides open API to **directly access the data-plane**

- logically-centralized control on a single device
- enables load balancing, traffic engineering, management
- switch only listens to API; dumb, fast & cheap

OpenFlow

Simple packet-handling rules
- Pattern: match packet header bits
- Actions: drop, forward, modify, send to controller
- Priority: if patterns overlap, decide what to do
- Counters: #bytes and #packages

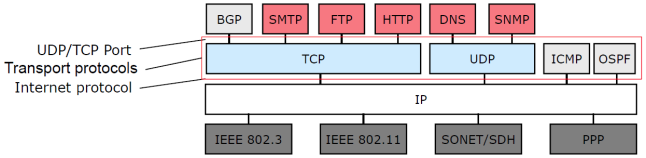
10. src=1.2.*.*, dest=3.4.5.* → drop
05. src = *.*.*.*, dest=3.4.*.* → forward(2)
01. src=10.1.2.3, dest=*.*.*.* → send to controller

Allows switches to emulate different hardware:
- Router: match longest destination & forward
- Switch: match destination MAC address & forward
- Firewall: permit or deny IP addresses & port numbers
- NAT: rewrite address and port of incoming traffic

Controller receives information and programs switches
- first packet goes to controller, rest directly ("fast track")

Dynamic access control: install rule to route traffic
Seamless mobility/migration: just adapt rules if change
Server Load balancing: pre-install load-balancing policy
Network Visualization: combine virtual networks to one

7. Transport Protocols



Transport Layer / Layer 4: are responsible for
- connection handling: connection-oriented/-less
- reliability of the channel (delay, flow control)
- performance: throughput, security

Client/Server: server offers service, client requests service
Peer-to-peer: equal peers, coordination required

Connection-oriented communication: before sending, a connection between endpoints is established (e.g. TCP)
- offers reliable transmission (identification, management)

Connection-less communication: packet delivery without connection establishment (e.g. UDP, IP)

User Datagram Protocol (UDP)

2 Byte	2 Byte
Source Port Number	Destination Port Number
Total Length	Checksum

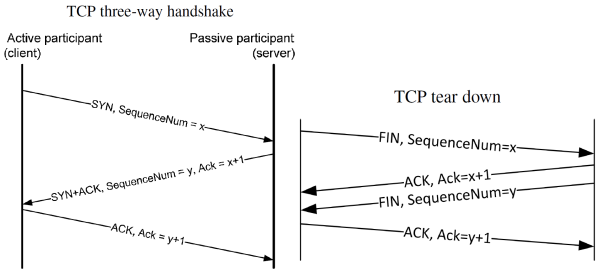
Connectionless datagram service (only addressing, no QoS)
- very little overhead with only 8 bytes for header
- checksum calculated over UDP header, payload & IP

Usage of unreliable data service
- application does flow & error correction itself
- application doesn't need reliable service
  (updates, statistics, measurement & monitoring)
- connection-oriented service too costly (overhead, set-up)
- retransmissions are useless (audio & video, real-time)
- multicasting (TCP doesn't support multicasting)

Transmission Control Protocol (TCP)

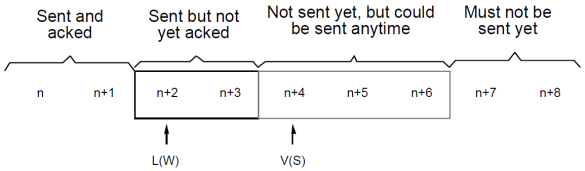
Possible problems with connections
- connection establishment → three-way handshake
- sequence error → sequence numbers
  (for check ordering & detection of duplication)

Features of TCP
- connection-oriented (full-duplex connection)
- stream-oriented (byte sequence numbers)
- sliding window protocol (window size in bytes)
- three-way handshake
- packaging with **MSS** (*maximum segment size*)



Source port address 16 bits								Destination port address 16 bits
Sequence number 32 bits								
Acknowledgment number 32 bits								
HLEN 4 bits	Reserved 6 bits	u r g	a c k	p s h	r s t	s y n	f i n	Window size 16 bits
Checksum 16 bits								Urgent pointer 16 bits
Options and padding								

Sliding window flow control



- SenderWindow = MIN(AdvertisedWindow, CongestionWindow)
- reduce sending rate to prevent receiver overflow

Self-clocking: sending rate reduces to ACK rate (adjust)

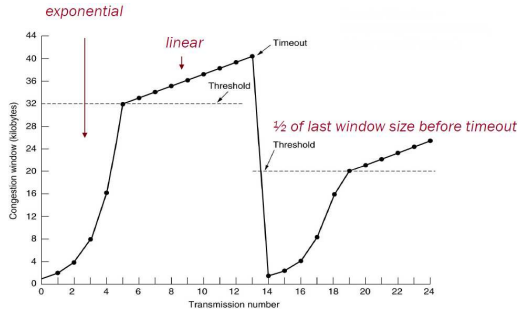
Acknowledgments
- receiver sorts segments in sequence number order
- received segments are acknowledged (ACK #seq.nr.)
  (cumulative: number identifies the next expected byte)
- send the advertised window size in the ACK to sender
- three duplicated ACKS start retransmission of packet
- can implement various protocols:
  - stop-and-go/stop-and-wait
  - go-back-N ARQ (Automatic Repeat Request)
  - selective-repeat ARQ (TCP SACK)
- **Retransmission TimeOut (RTO):** when timer down, retransmit as assume that sequence not received

$$RTO = \beta * avRTT + \gamma * devRTT$$

Flow control: avoid overflow at *receiver* (sliding window)
Congestion control: avoid overflow at *routers* (ACKs)

Congestion Control
- duplicated ACKS / segments may indicate congestion
- reduce load to prevent buffers from overflowing
- duplicated ACKS: data is getting through, but with losses
  - reduce congestion window by half
- Time out (RTO) indicates nothing is getting through
  - reduce congestion window to one segment
- slow start: congestion window starts with size 1
  - for each received ACK, increase by 1
  - exponential increase until threshold / first loss

Additive Increase, Multiplicative Decrease (AIMD)



8. End-to-end protocols

Inter Process Communication (IPC): establishes communication between processes over network

Service Oriented Architecture (SOA): uses transport layer technologies to remotely access/invoke services

Mechanisms for communication
- **Message Passing:** send messages over network
- **Space-based comm.:** write to virtual shared memory
- **Remote Service Invocation:** e.g. HTTP
- **Remote Procedure Call (RPC):** execute a procedure on a remote machine (e.g. Sun RPC, Java RMI)
- **Mobile Code:** migrating code to the remote machine

8.1 Remote Procedure Calls (RPC)

Function call which executes function on remote machine
- remote machine locally represented by a **stub**
- resistant against: server crashes, lost requests & responses, crashing clients, slow connection

Marshalling: format so it can be sent over the network

Location transparency: hide effect of client/server location
Performance transparency: runs like on own machine
Error transparency: hide effect of errors during execution

RPC semantics

"maybe": procedure may be executed or not, try once

"at least once": repeat until I got a reply (minimally once)
- must be idempotent: always same answer to call

"at most once": requests repeated if no reply

"exactly once": needs detection & handling of errors
- requires consistency, atomicity, isolation, durability

Stateless communication: interpretation independent of past, no state is maintained by either partners (HTTP, UDP)
Stateful communication: reaction depends on state (TCP)

8.2 Real-Time Transport Protocol (RTP)

Transportation of multimedia applications over network
- TCP not suitable (retransmission useless, congestion control causes rate fluctuations), but it works!

Quality of Service (QoS): jitter, lag, loss, delay
Quality of Experience (QoE): degraded experience

RTP uses UDP, as is widely used and passes firewalls & NAT
- end-to-end communication
- real-time stream data
- *profiles* describe media stream formats (MP3 etc.)

RTP Control Protocol (RTCP): extra control stream
- feedback on performance, may influence encoding
- used to correlate & synchronize different media streams

8.3 Session Initiation Protocol (SIP)

End-to-end application-layer session signalling protocol
- provides user presence and mobility management
- at which device is the user, and is he available right now?
- session set-up: which ports, which protocol & media type

SIP user agent (UA): logical network end-point (e.g. phone)

SIP servers
- **proxy server:** routes call requests, makes requests for UA
- **registrar:** accept registration, map SIP name to addresses
- **redirect server:** redirects to other server or other domain
- **location server:** database storing location information

SIP proxies: serve as rendez-vous points at which callees are globally reachable (perform relay signalling to user)

SIP reply

1xx: provisional
- E.g. 100 TRYING, 180 RINGING

2xx: success
- 200 OK
- 202 ACCEPTED

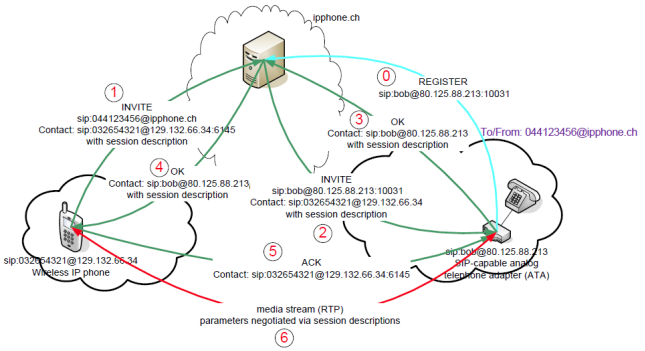
3xx: redirection
- E.g. 302 redirect to other server

4xx: client error
- E.g. 401 UNAUTHORIZED

5xx: server error
- E.g. 504 TIMEOUT

6xx: global failure

SIP protocol



8.4 Application-Layer Protocols

Application: communicating, distributed processes

Application-layer protocols define messages exchanged by applications and actions taken (use lower-level services)

Client: initiates contact, requests service from server
Server: provides requested service to client

Not provided by transport-layer:
- naming infrastructure
- structured data exchanged
- security: confidentiality, integrity, availability
- massively distributed applications

9. Domain Name System (DNS)

DNS provides an online, distributed naming service for hierarchically mapping hostnames to IP addresses (not directory: only one way, cannot search name for IP)
- application-layer protocol: hosts resolve names

HOSTS.TXT: store all names & addresses in one single file

Distributed & hierarchical system
- centralized system does not scale
- no single point of failure
- maintenance individually by each authority

Authoritative name server: stores mappings to values
- only answers requests it is authoritative about
- answers from primary/secondary server are authoritative
- answers from caches are non-authoritative

Resolver (local name server): performs query for hosts
- answer any requests, especially for "local" network
- **stub resolver:** can only send queries to another resolvers

Host sends *recursive* query to its resolver, which then sends *iterative* queries to authoritative name servers

Top Level Domain (TLD): just below root servers

Root Servers: 13 nominal root servers
- hundreds of physical servers by using *anycast*
- fail resistant, DDoS cannot take out internet

Glue records: name server are located in delegated zone

Reverse Lookup: given IP, find corresponding name
- use ARPA top level domain: "in-addr.arpa.", e.g. "1.0.0.127.in-addr.arpa. IN PTR www.heise.de"

DNS types
- A : name is hostname, value is address
- NS: name is domain (e.g. ethz.ch), value is name of ANS
- CNAME: used for aliasing
- AAAA: IPv6 Host Address (128 bits)

10. HTTP & Content Distribution Networks (CDNs)

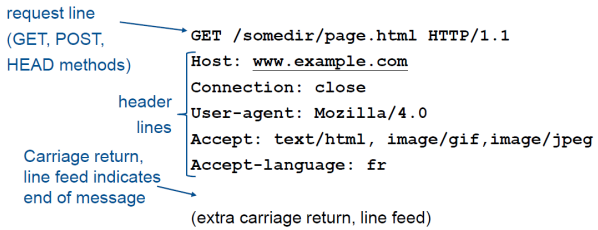
Hypertext Transfer Protocol (HTTP)

World-Wide-Web's application layer protocol
- stateless (every message contains all infos)
- uses TCP as transport service on port 80

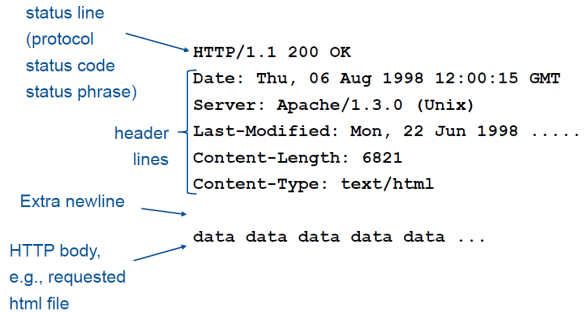
Stateful: information stored in end-points
- Hard: state installed and removed by receiver
- Soft: installed by receiver, removed after timeout

Stateless: scalability advantage, much easier protocol

Request



Response



Request methods

- GET
  - Fetch data from server
- HEAD
  - Fetch meta information from server
- POST
  - Send data to server
- CONNECT
  - Directly connect the client to a server, e.g., for SSL
- PUT
  - Upload a file
- DELETE
  - Remove a file
- OPTIONS
  - Ask the server for capabilities
- TRACE
  - For debugging – server or proxy reflects entire HTTP request

Response status codes

- `200 OK`
  - Request succeeded
- `301 Moved Permanently` and `302 Found`
  - Requested object moved, new location specified in Location header
- `304 Not Modified`
  - For cache control: no newer version of object
- `400 Bad Request`
  - Request message not understood by server
- `404 Not Found`
  - Requested document not found on this server
- `500 Internal Server Error`
- `501 Not Implemented`

Informational	1xx
Successful	2xx
Redirection	3xx
Client error	4xx
Server error	5xx

HTTP 1.1: allows persistent connections
- multiple objects transferred within one TCP connection

Pipelining: directly send multiple requests without waiting for response first

Cookies: state in the application layer; save data locally

Content Delivery Network (CDN)

30 ASes contribute 30% of inter-domain traffic

Advantages of serving clients via CDN
- enormous demand for popular content
- bad performance due to large client-server distance
- single point of failure
- high temporary costs
- sharing physical hardware between multiple hosts

AS resolvers are the ones which actually contact the CDN distribution (DNS) servers, it might happen that wrong conclusions are gained and wrongly redirected
- for content locality, use the local resolver

11. Email

Client delivers to local **Mail Transport Agent (MTA)**

MTA looks up DNS *MX* record for domain

```
$ dig -t mx tik.ee.ethz.ch

; […]
;; QUESTION SECTION:
;tik.ee.ethz.ch.                IN      MX

;; ANSWER SECTION:
tik.ee.ethz.ch.         86400   IN      MX      30 numenor.ee.ethz.ch.
tik.ee.ethz.ch.         86400   IN      MX      10 smtp.ee.ethz.ch.
tik.ee.ethz.ch.         86400   IN      MX      20 tranquillity.ee.ethz.ch.

;; […]
```

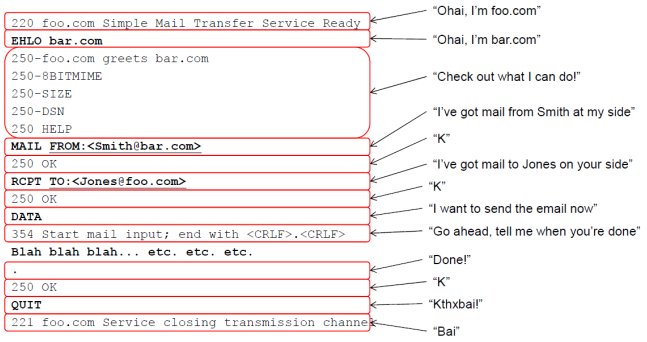
Give me the INternet address of the Mail eXchanger for tik.ee.ethz.ch

TTL = Time To Live (seconds)

Priority (lower number = higher priority)

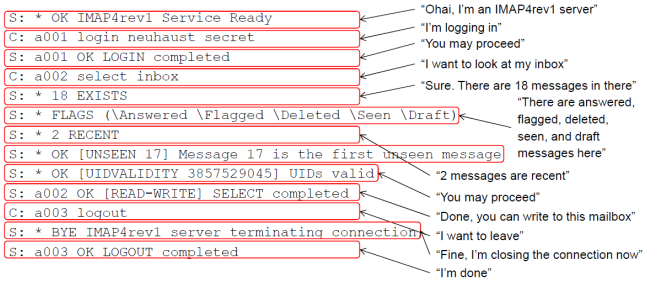
Simple Mail Transfer Protocol (SMTP)

- only server needs to keep state



Internet Message Access Protocol (IMAP)

- lets you access emails, create folders on servers etc.
- server and client need to keep state



12. Network Security

Security consists of the three pillars (CIA)

Confidentiality: ensure access only to authorized subjects

Integrity: ensure data is not modified (maliciously)

Availability: ensure reliable access to information

Assumption: attacker knows entire cryptographic system
- **"security by obscurity"** does not hold for long
- **computationally secure system:** takes ages to break

Symmetric Key Cryptography

depends on a secure (=authentic & confidential) channel

Symmetric Block Cipher: split into blocks & encrypt
- DES: 64-bit block cipher (key 56 bits)
- AES: 128-bit block cipher (key 128,192,256 bits)
- vulnerable to **Frequency Analysis** (based on language)

One-Time Pad: perfect security (not only computationally)
- key same length as plaintext, bits are randomly chosen
  (use **stream ciphers** to create pseudo-random sequence)
- XOR plaintext with key, ciphertext is statistically indep.

Data Encryption Standard (DES)
- 56bit key, generate 16 per-round key out of it
- substitution using controversial S-boxes
- nowadays easily broken due to short key

Advanced Encryption Standard (AES)
- successor to DES in 2000
- initialisation vector & feedback for *cipher block chaining*

Attacks

Replay attack: use packet more than once
Reflect attack: reflect own package back
Man-in-the-Middle attack: fake identity in the middle
(effective attack if channel is not authentic)
Known Plaintext attack: have cipher text, know plaintext

Public Key Cryptography

Do not rely on previously shared secret, but share it openly
- mostly used to share secret for symmetric key crypto

Diffie-Hellman Key Exchange
- public: base g, prime p
- private: x_a , x_b
- common secret: $S_{AB} = y_B^{x_A} = (g^{x_B})^{x_A} = y_A^{x_B} = S_{BA}$

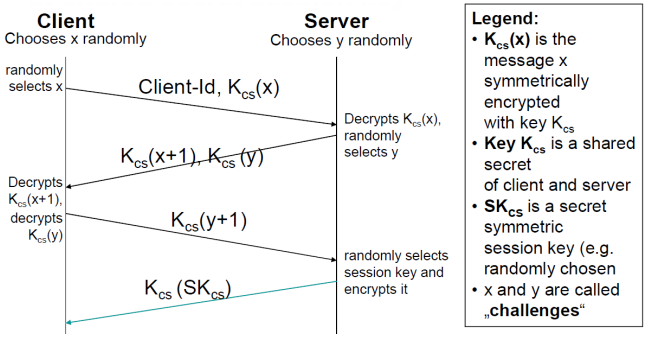
Encryption: sender encrypts with recipients public key, which only the recipient can decrypt with his private key

Signing: sender "signs" message with private key, everyone can see this by decrypting with his public key

Rivest-Shamir-Adleman (RSA)
- two large primes p, q : $m = p * q, f = (p-1)(q-1)$
- choose e such that e and f have no common dividers
- secret key: $d = e^{-1}\ (mod\ f)$, public key: (m, e)

$$c = x^e\ (mod\ m)\ , \qquad x = c^d\ (mod\ m)$$

Mutual authentication & exchange of shared session key



Message Integrity Code (MIC): to check if decryption successful & check integrity and authenticity

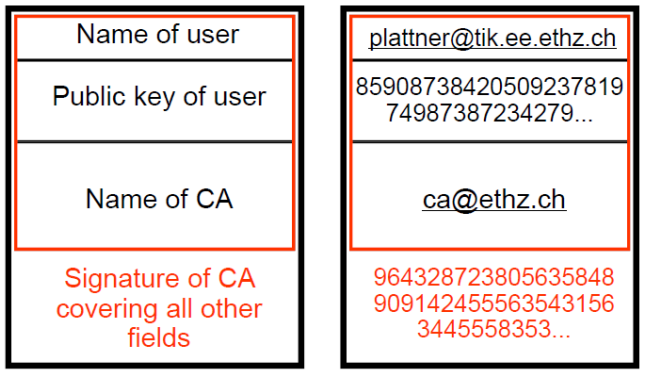
Symmetric Cryptography: secret key
- VERY fast encryption, but long key required

Asymmetric Cryptography: public key
- slow, but practical key; used to create secret key for SC

Certificates & Certificate Authorities (CAs)

- possibility of digital signature: it is really me you are talking to, not someone who wants to listen to us

Assumption: public key of CA can be distributed in an authentic way (mostly built-in in browser)



Hash-function: create short message from message of arbitrary size (compress to certain length)
One-way function: easy in one way, but inverse impossible
One-way hash function: easily create short message from long one, but not possible to forge output

Nonce: *number only used once*

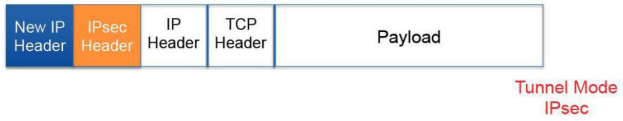
12.1 IPSec – Network Layer Security

IPSEC provides Layer 3 security

Transparent to applications (do not require support)
- transparent to all protocols above IP
- good for VPNs
- cannot cross NATs (as they change the TCP header)

Modes of operation

Tunnel Mode: entire IP packet is encrypted and becomes data component of a new IP packet (large size)



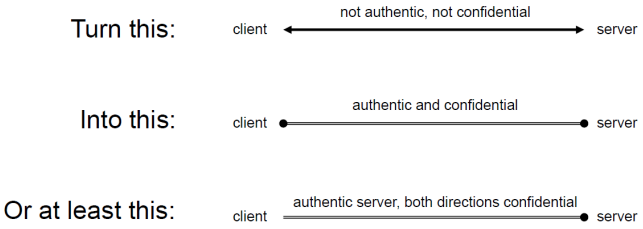
Transport Mode: IPsec header inserted into the IP packet
- may be used for remote-access VPNs and networks where long packets may cause issues

Security Associations (SA)
- binds crypto keys and session state to session
- includes: Security Parameter Index (SPI), IP destination

Authentication Header (AH)
- provides source authentication & data integrity

Encapsulating Security Payload (ESP)
- provides everything of AH + data confidentiality
- uses symmetric key encryption

12.2 Secure Socket Layer (SSL)/ Transport Layer Security (TLS)



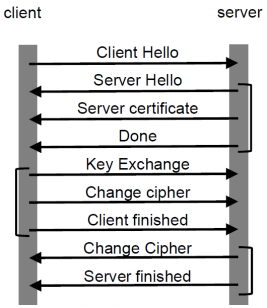
Cipher Suites

Key exchange methods
- RSA: encrypt key with receiver's public key
- Diffie-Hellman

Cipher spec
- Cipher Algorithm (DES, 3DES, AES, RC4) with modes
- MAC algorithm (hash functions)
- stream or block (cipher type)

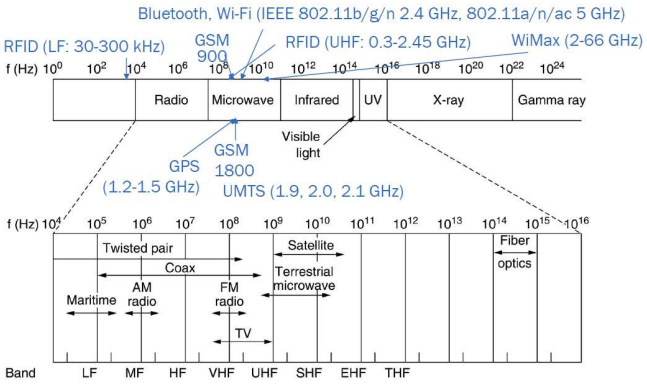
Server adapts to capabilities & preferences of client
- TLS falls back to SSLv3 if fails or if wished

Handshake



13. Various

Wireless spectrum



IP Address Specifications

Identify a network: $129.132.\mathbf{0.0}$

Address all entities (broadcast): $129.132.\mathbf{255.255}$

Loopback address / network: $127.0.0.1$ / $127.X.X.X$

IPv4 Prefix: $129.0.2.0/\mathbf{24}$

Notation for IPv6 addresses

Full form: $2001:0db8:0000:0000:0000:0000:0000:029a$

Compressed: $2001:0db8::29a$

IPv4 embedded in IPv6: $0:0:0:0:0:ffff:192.0.2.42$

Localhost: $::1$

IPv6 Prefix: $2001:db8::/\mathbf{32}$
(no more network masks, only prefixes)

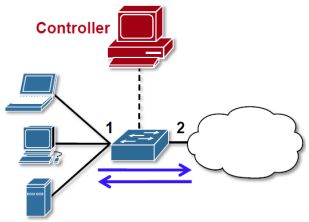
Zone qualifier (%): $fe80::202:2dff:fe2a:3f78\,\%\mathbf{4}$

URL (with port) : $\boldsymbol{http}://[2001:db8::42]:[\mathbf{80}]$

OpenFlow Programming

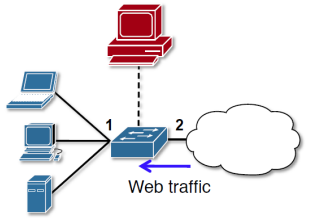
Simple Repeater

```
def switch_join(switch):
  # Repeat Port 1 to Port 2
  p1 = {in_port:1}
  a1 = [forward(2)]
  install(switch, p1, DEFAULT, a1)

  # Repeat Port 2 to Port 1
  p2 = {in_port:2}
  a2 = [forward(1)]
  install(switch, p2, DEFAULT, a2)
```



Monitor "port 80" traffic

```
def switch_join(switch):
  # Web traffic from Internet
  p = {inport:2,tp_src:80}
  install(switch, p, DEFAULT, [])
  query_stats(switch, p)

def stats_in(switch, p, bytes, …)
  print bytes
  sleep(30)
  query_stats(switch, p)
```



Repeater + Monitor

```
def switch_join(switch):
  pat1 = {inport:1}
  pat2 = {inport:2}
  pat2web = {in_port:2, tp_src:80}
  install(switch, pat1, DEFAULT, None, [forward(2)])
  install(switch, pat2web, HIGH, None, [forward(1)])
  install(switch, pat2, DEFAULT, None, [forward(1)])
  query_stats(switch, pat2web)

def stats_in(switch, xid, pattern, packets, bytes):
  print bytes
  sleep(30)
  query_stats(switch, pattern)
```

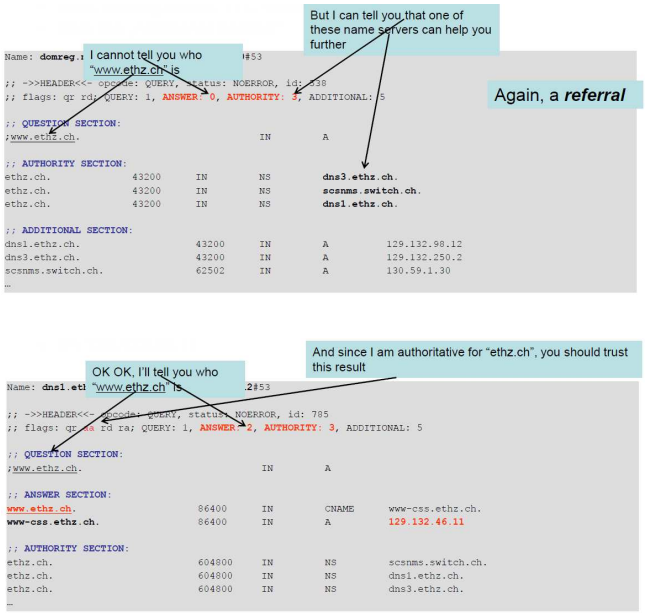
Port numbers

IP address: for addressing *network interface*

Port: for addressing *running process*

Port	Name	Protocol	Description
20	ftp-data	TCP	File Transfer Protocol (data channel)
21	ftp	TCP	File Transfer Protocol (control channel)
23	telnet	TCP	Terminal
25	smtp	TCP	Mail transfer
53	dns	UDP/TCP	Domain name server
67	bootps	UDP/TCP	Bootstrap server
68	bootpc	UDP/TCP	Bootstrap client
69	tftp	UDP	Trivial File Transfer Protocol
80	www	UDP/TCP	WorldWideWeb HTTP
109	pop2	UDP/TCP	Post Office Protocol ver 2
110	pop3	UDP/TCP	Post Office Protocol ver 3
119	nntp	TCP	USENET News Transfer Protocol
123	ntp	UDP/TCP	Network Time Protocol

DNS Lookup



SMTP reply code

SMTP Reply Codes: First Digit

- 1xy — Positive Preliminary reply
  - OK so far, but confirmation is pending
- 2xy — Positive Completion reply
  - Everything successfully completed
- 3xy — Positive Intermediate reply
  - Everything OK so far, pending further information
- 4xy — Transient Negative Completion reply
  - Not accepted, action did not occur. Try again later
- 5xy — Permanent Negative Completion reply
  - Not accepted, action did not occur. Don't try this again.

SMTP Reply Codes: Second Digit

- x0z: Syntax
  - Syntax errors, unimplemented commands
- x1z: Information
  - Replies to requests for information (status, help)
- x2z: Connections
  - Reply concerning the transmission channel
- x3z, x4z: unspecified
- x5z: Mail System
  - Status of receiving mail system w.r.t. current request
  - E.g., 250 = "OK"