Lecture 11: IP routing, IP protocols

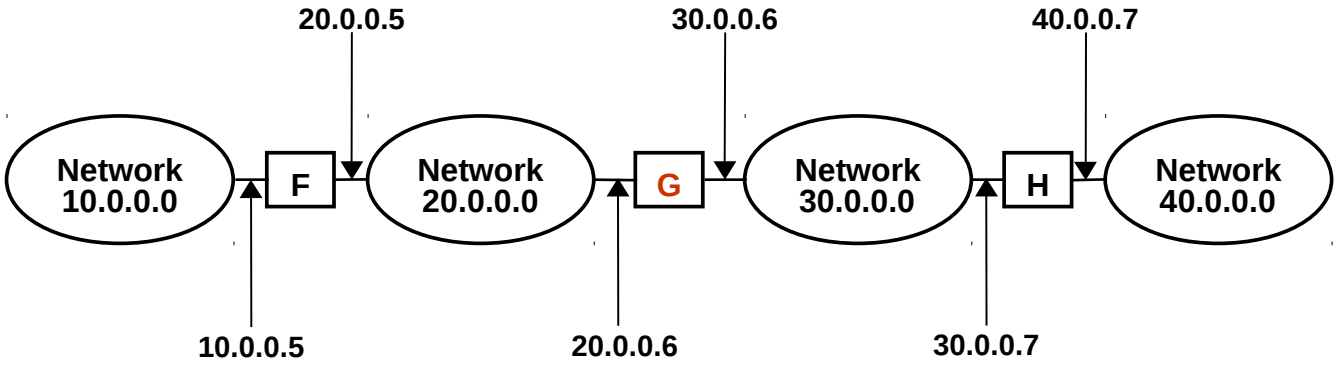
Contents

- Routing principles
- Local datagram delivery
- ICMP Protocol
- UDP Protocol
- TCP/IP Protocol
- Assuring requirements for streaming TPC
- Building and terminating TCP connection
- API for network services (brief info on sockets)

Datagram routing (1)

- Routing is a decision making process on where to forward the datagram (or its fragment).
  - Any device making such decision is called router
  - Routing can be direct or indirect
    - Direct routing occurs when the destination host is a part of the LAN directly connected to the router so that the datagram can be delivered directly to the destination.
    - All other cases represent indirect routing
  - Routers in the Internet form a cooperative interconnected structure. Datagrams pass from router to router until they reach a router capable to deliver directly (locally) to the destination
  - Table driven routing
    - Every router maintains a **routing table** with entries in the form of pairs (*N*, *G*), where *N* is *netid* of the destination network and *G* is the IP address of the "next router" along the path to the destination network *N*. The "next router" must be reachable directly so that the datagram can be forwarded there without further routing

Datagram routing (2)



Destination is on the net	Forward datagram to
20.0.0.0	Deliver directly
30.0.0.0	Deliver directly
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

Routing table of G

- Default routes
  - Often LAN's are connected to the "rest of world" through a single router. Such router is called default gateway, i.e. the address where all host on the LAN send datagrams with the destination outside the LAN
- Host-Specific Routes
  - Sometimes it may be reasonable to assign a special routing information for administrative, security, or technical reasons (e.g., dedicated serial line connection)

Datagram routing (3)

- Routing algorithm
  - Extract the destination IP address *DA* from the datagram and using the netmask determine the *netid* of the destination network

1. If *DA* corresponds to a *host-specific route* send the datagram directly to the destination host
2. If *netid* is equal to the *netid* of a directly connected LAN then deliver directly (locally)
3. If *netid* can be found in the routing table send the datagram to the corresponding “next router”
4. If the *default route* is specified send the datagram to the *default gateway*
5. Otherwise send a ICMP “*Destination unreachable*” message to the datagram sender

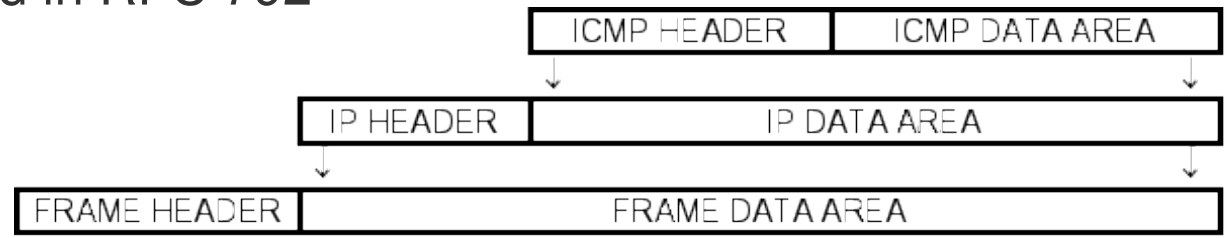
Local datagram delivery

- Direct routing must assure local datagram delivery
  - The same has to be done when forwarding to a router directly connected on the LAN (not through *point-to-point* link)
  - Datagram contains the destination IP address but the delivery must be accomplished using a LAN specific physical address
- Mapping IP addresses to physical addresses
  - *Address Resolution Protocol* (ARP) – dynamic mapping
  - Ethernet case example solution: datagram sent by host *A* to host *B*
    - The sender has IP address I_A and physical address P_A needs to learn the physical address P_B but knows only the IP address I_B.
    - Host *A* sends an ARP ethernet broadcast frame and in its data part will contain both I_A and I_B (*Who-has ARP request*). This frame get all hosts on the LAN.
    - Host which recognizes its IP address I_B responds to this broadcast (*ARP reply*) and tells the requestor its physical address P_B.
    - The ethernet broadcast makes a big load for the LAN host; so the requestor caches P_B for some time (5 minutes as standard)
    - Host *B* has learnt from ARP request also addresses I_A a P_A so it caches it, too (an early datagram from *B* to *A* can be expected).

```
A real network capture:
  12:32:35.363 arp who-has 147.32.85.117 tell 147.32.85.8
  12:32:35.696 arp reply 147.32.85.117 is-at 00:1a:80:d9:27:64
```

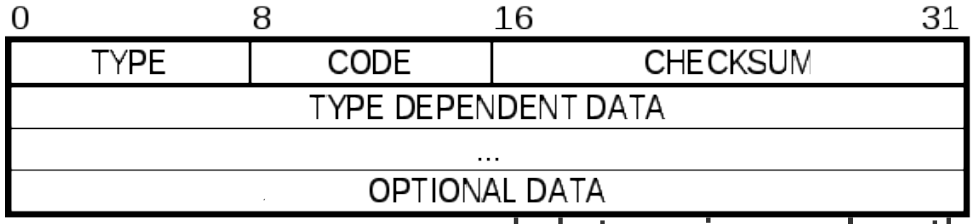
ICMP Protocol

- ICMP (= *Internet Control Message Protocol*), IP protocol number = 1 (cf. IP datagram header)
  - The simplest protocol for network control and error message passing defined in RFC 792



ICMP encapsulated in an IP datagram on a physical network

  - ICMP header does not have a fixed structure (except for the first 4 bytes)



  - Field TYPE shows ICMP message purpose and determines also the format and meaning of other fields – some ICMP types are:

TYPE	Purpose	TYPE	Purpose
0	Echo reply	8	Echo request
3	Destination unreachable	11	Datagram TTL exceeded
5	Redirect (route change)	12	Datagram parameter problem

IPv6

- IPv6 has 128 bits address (IPv4 only 32 bit)
- IPv6 improves some issues from IPv4 but it is still IP layer
- Why 6? Internet Stream Protocol from 1979 uses in IP header byte version 5. Next free number was 6.
- Improvements
  - Larger address space – aprox 3.4×10^{38} of address
  - Multicasting – sending one packet to different computers
  - SLAAC – Stateless address autoconfiguration –automatic configuration using Neighbor Discovery from router with ICMPv6 protocol (can be used DHCPv6 or static settings)
  - Network security is in IPv6 mandatory
  - Mobility – avoid triangular routing for mobile devices
  - Jumbogram – datagram with size $2^{32} - 1$ (in IPv4 max 65535)

IPv6 address

- Notation 8 groups with 16 bits – each group has 4 hexadecimal numbers
  - 2001:0db8:85a3:0000:0000:8a2e:0370:7334
  - Zeros are not necessary, if they has no meaning
  - 2001:db8:85a3:::8a2e:370:7334
- General format unicast and anycast:
  - 48 or more – routing prefix
  - 16 or fewer – subnet-id (subnet-id+routing prefix = 64)
  - 64 – interface identifier (it can be MAC-address of network card, assigned by DHCPv6, generated randomly, assigned manually)
- Format for multicast:
  - 8 bits prefix – started with FF
  - 4 bits flag
  - 4 bits scope
  - 112 bits group ID

IPv6 routing

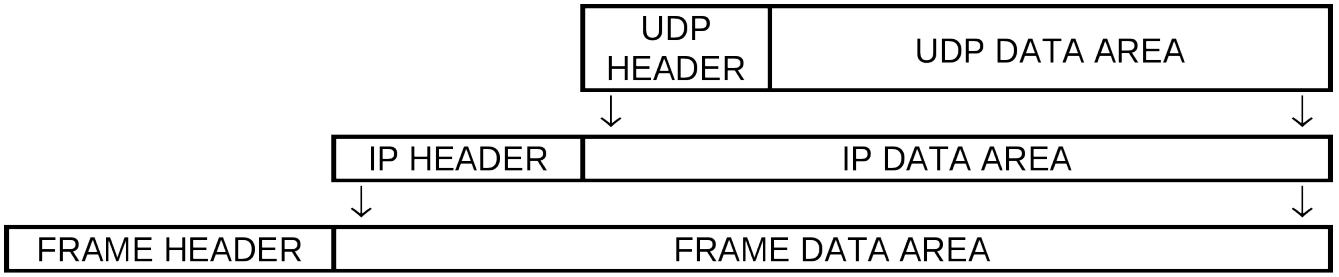
- Simplified processing by routers
- Packet header in IPv6 is more simple
- IPv6 Routers do not perform fragmentation
  - Minimal MTU (Maximal transmission unit) is 1280
  - Perform MTU discovery – it means detect MTU for specific route
- IPv6 has no checksum (it is in transport and link layer)
- TTL – (Time To Live) is replaced by Hop Limit – routers need not to compute time in queues
- Routing prefix specify all routing information
  - RFC 3177 (2001) suggested that all end sites receive /48 address allocation
  - RFC 6177 (2011) reduces this address allocation to /56

From IPv4 to IPv6

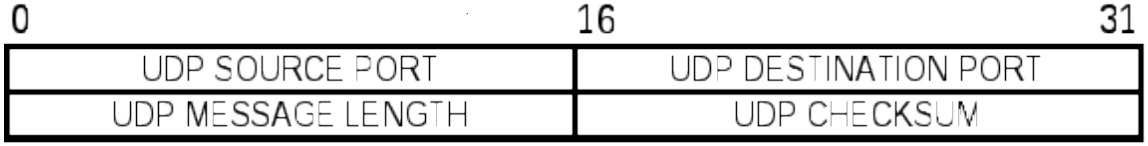
- IPv4 with CIDR routing and NAT slow down change to IPv6
- How to transform IPv4 to IPv6
  - Dual-stack – router can handle IPv4 and IPv6 together
  - Tunneling – encapsulation of IPv6 into IPv4 protocol
  - Proxying and translation for IPv6-only host
  - Backward compatibility, IPv6 host can communicate with IPv4 network
- You can check your computer on:
  - www.test-ipv6.cz
  - www.test-ipv6.com
  - ipv6test.google.com

Transport layer - UDP Protocol

- UDP (= *User Datagram Protocol*), IP protocol number = 17
  - Very simple transport protocol defined in RFC 768
  - Provides connectionless and unreliable transport of user datagrams.
  - If any kind of transmission reliability is needed, it must be implemented in the user's application.
  - Compared to pure IP datagrams, UDP has the ability to address target processes on the destination host using the port field



UDP encapsulated in an IP datagram



UDP Header

  - PORT fields are used to distinguish computing processes awaiting a UDP datagram on the destination host.
    - SOURCE PORT field is optional; must be 0 if unused.
    - Otherwise it contains the port number to which a possible reply should be sent.

UDP Protocol (cont.)

- To ensure that different hosts on Internet will understand each other, IANA (= Internet Assigned Numbers Authority) issues the obligatory list of "*Well Known Port Numbers*"
- Selected UDP port numbers are:

Port	Keyword	Meaning
0	-	Reserved
7	echo	Echo the datagram
9	discard	Discard the datagram
13	daytime	Time of the day
53	dns	Domain Name Service
67	bootps	Bootstrap Protocol Server, DHCP Server
68	bootpc	Bootstrap Protocol Client, DHCP Client
69	tftp	Trivial File Transfer
137	netbios-ns	NETBIOS Name Service
...		

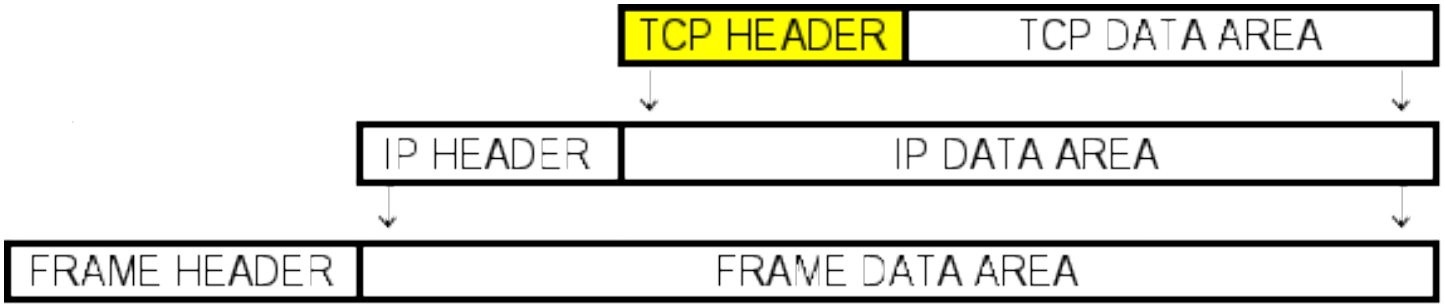
- For the complete list see http://www.iana.org/assignments/port-numbers

TCP Protocol for reliable data streaming

- TCP is the most important general reliable transport service providing a virtual bidirectional communication channel between two hosts
  - TCP/IP is the IP implementation of this service
- TCP properties
  - Data stream
    - Applications communicating through a TCP connection consider the communication channel as a byte stream similarly to a file
  - Virtual connection
    - Before the data transmission can start, the communicating applications have to negotiate the connection by means of the network components of their local operating systems – create and connect the sockets.
    - The protocol software (transport layer) in the operating systems of both hosts make an agreement on the connection using messages passed over the network. The hosts also verify that the connection can be reliably established and that both end-point systems are ready to communicate – open the sockets.
    - Afterwards the end-point applications are informed about the established connection and the data communication can start.
    - If the connection breaks, both communicating applications are informed
    - The term virtual connection is used to create an illusion that applications are interconnected through a dedicated line.
    - The reliability is ensured by full hand-shake communication (everything must be acknowledged by the other party)

TCP/IP – TCP Protocol IP Implementation

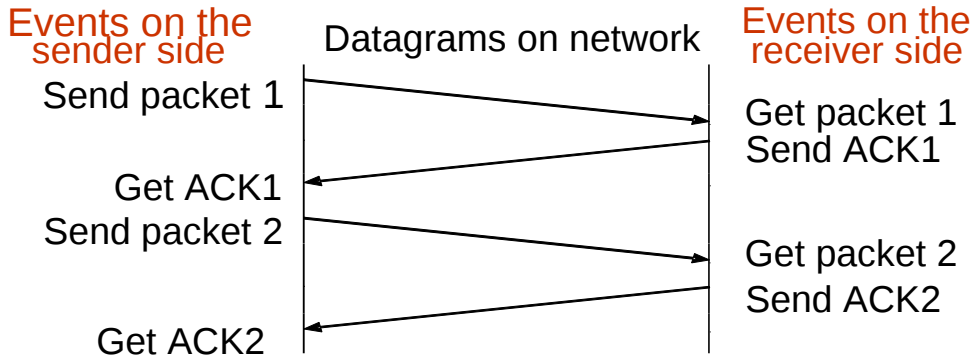
- Besides of the general TCP properties, the TCP/IP implementation provides:
  - Buffered transport (streaming)
    - To improve the efficiency, the TCP/IP module in the OS assembles bytes from the stream into packets (datagrams) of reasonable size. If this is not desirable (e.g., TELNET ctrl-C), TCP/IP is equipped by a mechanism enforcing the priority transfer of a short datagram “out-of-order”.
  - Full duplex connection
    - Application processes can see the TCP/IP link as two independent data streams running in opposite directions without an apparent interaction. The protocol software actually acknowledges data running in one direction in the packets sent together with the data in the opposite direction.



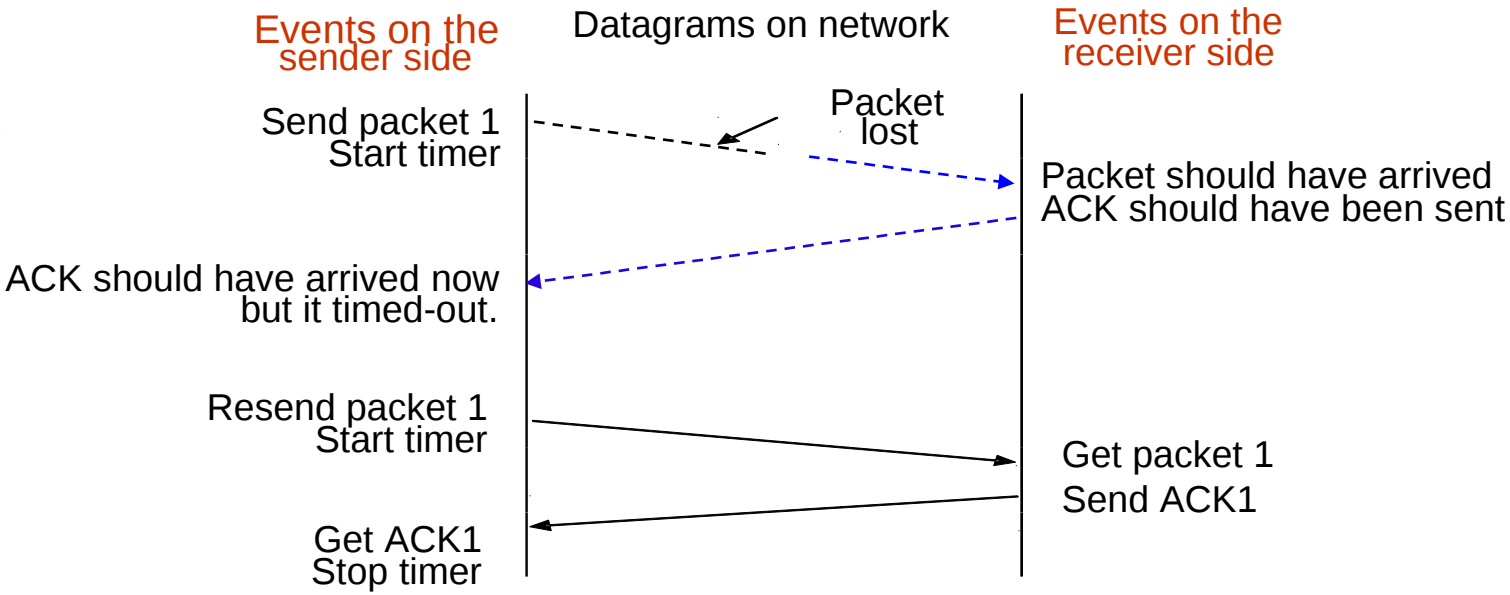
TCP/IP encapsulated in an IP datagram on a physical network

TCP reliability solution

- The reliable transfer is ensured by
  - positive acknowledging of the received data together with repeated packet transfer

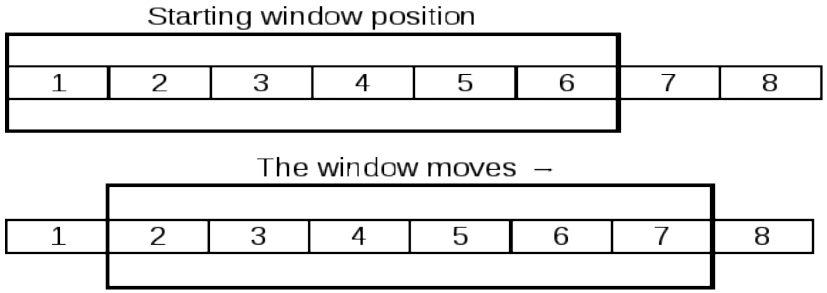


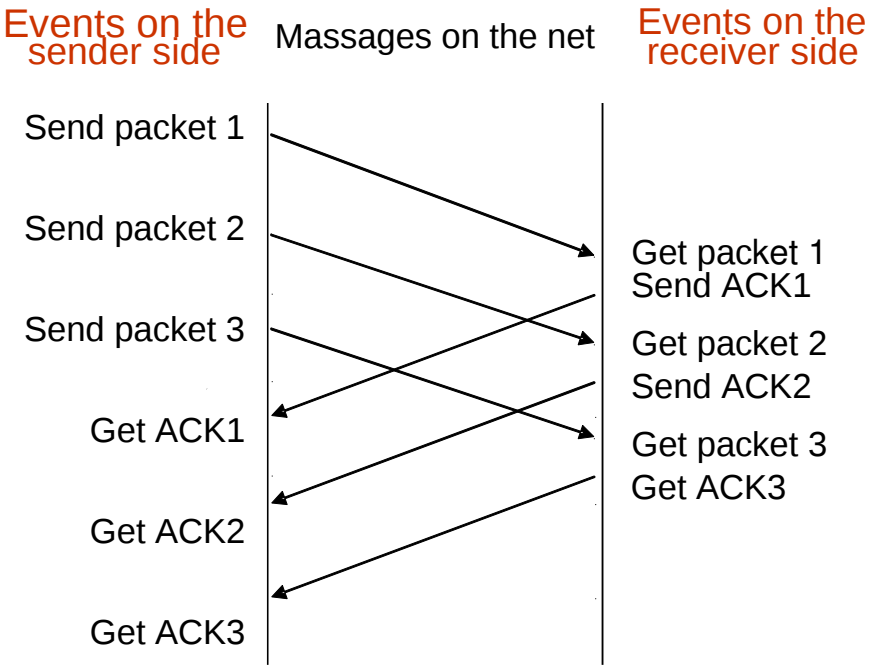
  - Lost packet are repeated based on suitable timeouts



TCP reliability solution (cont.)

- Datagrams may get duplicated during the travel (both data and ACK).
  - This problem is solved by sequential numbering of datagrams
- The problem of positive acknowledgement efficiency
  - The moving window method



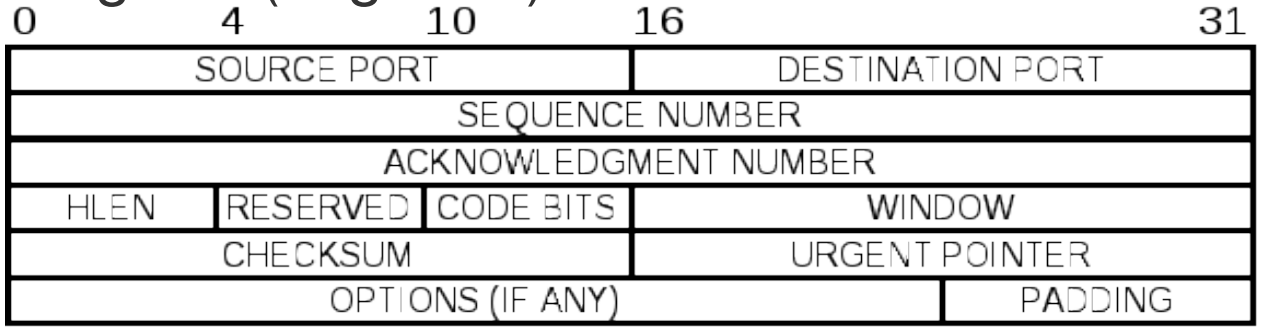


TCP ports, connections and end-points

- TCP/IP also uses ports to distinguish target applications on the connected hosts
  - The port numbers can be the same as in UPD because protocols are separated first
  - To use a service provided by a server by any number of client hosts, TCP/IP is equipped by so called virtual connections (virtual channels):
    - These virtual channels are actually connection between so called end-points, which IP addresses with an appended port number (6 bytes in IPv4), e.g., 147.32.85.34:80
    - TCP/IP virtual connection is then specified by two end-points of the particular connection (12 bytes in IPv4)
- Passive and active end-point opening
  - TCP/IP needs the system between which the connection should be established to negotiate the connection 'parameters'
    - The application at the first end-point must ask its local OS to accomplish passive open on the particular port indicating thus the willingness to accept the incoming requests for connections. This end-point is usually denoted as server. It listens on to particular port for incoming connection requests.
    - If a client application wants to connect to a server, it asks its local OS to make an active open to the server IP address and the applicable port. The local client OS finds a free local port (usually 1024 – 2047). Both hosts establish the connection that is identified by the two end-points and may start communicating each to other.

TCP/IP segments and their format

- TCP data stream is split into segments
  - Segments travel over the Internet as IP datagrams
    - Every byte in the data stream has its 32-bit sequential number within the connection
- TCP datagram (segment) header:



- TCP header fields
  - SOURCE PORT, DESTINATION PORT: Application identifications on both connection end-points
  - SEQUENCE NUMBER: Sequential number of the first byte in the data stream transferred in the datagram.
  - ACKNOWLEDGMENT NUMBER: Sequential number of the first byte in the opposite stream which the sender expects from the receiver as the answer.
    Remark: SEQUENCE NUMBER is related to the transfer direction, in which the segment is sent while ACKNOWLEDGMENT NUMBER relates to the opposite direction.

TCP/IP segments and their format (cont.)

- TCP header fields (cont.)
  - HLEN: Header length in 32-bit words
  - CODE: Structured field containing 1-bit flags:
    - URG: Field URGENT POINTER is valid
    - ACK: Datagram carries an acknowledgement of the opposite running segment
    - PSH: This segment requires a "push", i.e. an immediate delivery to the target application without any receiver side buffering
    - RST: Connection reset
    - SYN: Active request to establish a connection (sequence numbers synchronization)
    - FIN: Connection termination (sender detected the end of data strem)
  - WINDOW: Determines how much data the sender can accept in the opposite running data stream
  - URGENT POINTER: Pointer to an urgent data element in the segment data section (e.g., ctrl-C in an Telnet session) – makes sense only if the URG flag is set
  - OPTIONS: Optional fields used during the connection negotiation (e.g., maximum segment size)

Retransmission timeouts

- Constant value of timeout when to resend the TCP/IP segment is inappropriate
  - Internet is too heterogeneous and is composed of a huge number of LAN's based on different HW technologies
    - Giga-bit ethernet, 33 kbit serial line, intercontinental satellite link, etc.
- TCP/IP adapts to changing timing parameters of the virtual connection
  - A simple adaptive algorithm to adjust the timeout is used
  - The algorithm is based on continuous monitoring of „*round trip time*" (RTT)
    - Time between dispatching the packet and its acknowledgement.
  - The real timeout is then computed as a weighted average of RTT measured in the recent history.
  - This strategy quickly accommodates to the speed and load changes on the intermediate networks and routers

Establishing the TCP connection

- TCP uses a three-stage procedure to establish the virtual connection:
  1. In the first step, the connection initiator (client) sends the other party (server) a segment containing SYN bit = 1, randomly generated SEQUENCE NUMBER = *x* and an empty data section.
  2. The server responds by a segment with SYN and ACK set to 1, random SEQUENCE NUMBER = *y* and ACKNOWLEDGMENT NUMBER = *x*+1.
  3. After receiving this segment, the client acknowledges it by sending a segment with ACK set to 1, SYN bit=0 and the ACKNOWLEDGMENT NUMBER = *y*+1.
  - This way the initial values of SEQUENCE NUMBER and ACKNOWLEDGMENT NUMBER fields are synchronized for the future life time of the virtual connection and are used to number the following TCP/IP segments
  - The sequential numbers are random
    - It enables to detect a failure or restart of hosts on the connection ends that can happen during a longer timeout

TCP connection termination

- Connection is normally terminated on request of one of the connected applications
  - Application tells TCP that there are no more data to be exchanged
    - If server – passive close
    - If client – active close
  - TCP software closes the connection by sending a segment with a set FIN bit
  - For the final termination of the virtual connection, it is necessary also to close the opposite direction. The party that received the segment with FIN bit reacts by sending a segment with FIN bit, too.
  - The TCP connection can be terminated forcibly using the RST bit

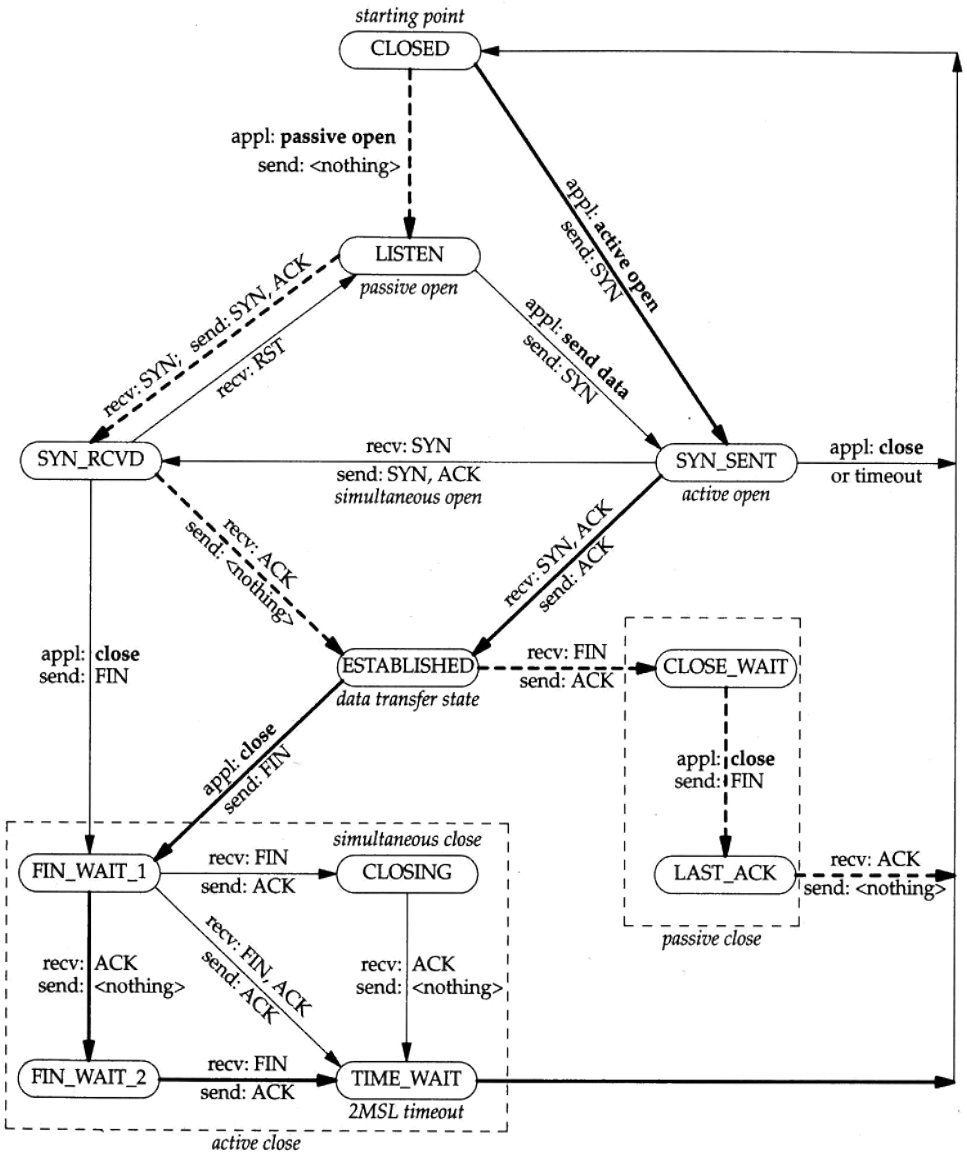
Well known TCP ports

- Examples of some TCP/IP ports

Port	Keyword	Meaning
0	-	Reserved
7	echo	Echo the datagram
13	daytime	Time of the day
20	ftp-data	File Transfer Protocol (data stream)
21	ftp	File Transfer Protocol (controls)
22	ssh	Secure Terminal Connection
23	telnet	Terminal Connection
25	smtp	Simple Mail Transport Protocol
53	domain	Domain Name Server Query
79	finger	Finger service (who is logged on?)
80	http	World-wide web
110	pop3	Post Office Protocol - V3 – get incoming mail from server
443	https	Secured world-wide web

and many others (see http://www.iana.org/assignments/port-numbers)

TCP finite state machine



- Dashed arrows – server side state transitions
- Solid arrows – client side state transitions
- Lower left frame – active connection close
- Lower right frame – passive connection close

- appl: transition caused by system call from application
- recv: transition caused by received segment
- send: shows what is sent in reaction to state change

API for network services

- The basic system API for network communication is a socket (see also - *Berkeley sockets*)
  - Create socket (get a socket "file" descriptor)
    sock_descr = **socket**(af, type, protocol)
    - af – specifies address and protocol family (for IP af = AF_INET)
    - type – determines the way of communication
      - SOCK_STREAM, SOCK_DGRAM, SOCK_RAW, ...
    - protocol – particular protocol type (TCP, UDP, ICMP, ...)
  - Assign socket a local address (passive socket open – server side)
    bind(sock_descr, local_addr, addr_len)
    - sock_descr – socket
    - local_addr – local address, for AF_INET a structure including port
    - addr_len – address length in bytes
  - Listen for incoming connections (server side)
    listen(sock_descr, backlog)
    - sock_descr – socket, backlog – number of allowable connections
  - An incoming connection arrived (server gets the client identification)
    new_sd = **accept**(sock_descr, *client_addr, *client_addr_len)
  - Connect to a remote address (active open – client side)
    connect(sock_descr, remote_addr, addr_len) – parameters like for **bind**()
  - Data transfers using
    write, **send**, **sendmsg**, **read**, **recv**, **recvmsg**
  - Terminating the connection
    close(sock_descr)

Using the network API

- Server side
  1. Create socket calling `socket()`
  2. Assign the local address (and port) using `bind()`
  3. Prepare socket for incoming connection requests calling `listen()`- „listening socket“
  4. Calling `accept()` the server blocks until a request for connection comes. The `accept()` return value is a new socket descriptor open for communication. The original socket continues listening and `accept()` can be called again.
  5. Communication using `send()` and `recv()` or `write()` and `read()`
  6. Possible call to `close()` when the server exits (passive close)

- Client
  1. Create socket calling `socket()`
  2. Call `connect()` to make a connection to server. The local socket descriptor becomes open for succeeding communication (local operating system assigns a free local port)
  3. Communication with the server using `send()` and `recv()` or `write()` and `read()`
  4. Calling `close()` ends the connection to the server (active close)

End of Lecture 11

Questions?