Lecture 13: Application layer

Contents

- Goals of transportation layer
- UDP
- TCP
- Port vs. Socket
- QoS

Goals of transportation layer

- End-to-end communication
- Distinguish different senders and receivers on one host
- Enable connection-oriented data transfer
- Ensure reliability
- Ensure Quality of service
- Control data transfer

- Main features of TCP
  - Data arrive in-order
  - Data hava minimal error – correctness
  - Duplicate data are discarded
  - Lost/discarded data are resent

Transportation Layer

- Transportation layer suppose 3 types of network layer:
  - Category A: no lost packets, no disconnection – local networks
  - Category B: no lost packets, some disconnection – private data networks
  - Category C: lost packets and disconnections – whole internet
- 5 classes of transportation layer:
  - TP0 – simple layer for category A
  - TP1 – layer for category B, solve disconnections
  - TP2 – for category A, enable use one network layer for more transport connections – ports
  - TP3 – for category B, enable use one network layer for more transport connection and handle disconnections
  - TP4 – transport layer for category C, reliable data transfer

Transportation layer

- Network layer and bellow layers are minimalistic, layers focus on speed and robustness, no connections and no reliability
- Transport layer can change character of network layer if the application wants
- Transportation layer can be reliable and connection-oriented
- UDP – simple transport layer with no connection and no reliability
- TCP – connection-oriented and reliable
- Application can select TCP or UDP

Transportation layer

- Network layer uses host as a unit
- Transportation layer distinguishes users on host – different applications from different users
- Transportation layer is making multiplex – gathers data from different users and makes de-multiplex – received data are assigned to different users
- To distinguish users transportation layer cannot use PID, because PIDs are dynamical, the static gates are used
- ISO/OSI defines SAP – Service Access Point, TCP calls this gates as ports
- Application connects to SAP and use the SAP as distinguish descriptor

Socket vs. port

- Port is number that define SAP for connection from different applications
- Socket was created as abstract file for controlling data transfer through SAP
- Sockets were used for networking as API for using ports - SAPs
- Ports are same for different computers
- Sockets depend on specific implementation (WINSOCK for Windows, BSD Socket for unix)

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)

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, or you can use bind function, before connect)
  3. Communication with the server using `send()` and `recv()` or `write()` and `read()`
  4. Calling `close()` ends the connection to the server (active close)

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 computed as a weighted average and dispersion of RTT measured in the recent history.
  - This strategy quickly accommodates to the speed and load changes on the intermediate networks and routers

Flow control

- The receiver sends to sender size of its free part of receiving buffer
- Sender sends packet with data
- Receiver accepts data and computes new size of free part of buffer and sends acknowledgment with this new size
- If there is no free space in buffer, receiver sends 0 and blocks the sender
- After reading data from application, receiver sends again acknowledgment with new size of free part of receiving buffer

QoS

- Different applications have different demands
- Standard functionality of transport layers is best effort – as much as possible
- Solution
  - Brute force – increase network capacity
  - QoS – Quality of Service – different demands
- QoS – different qualities
  - Dropped packets The routers might fail to deliver (*drop*) some packets if their data is corrupted or they arrive when the buffers are already full. The receiving application may ask for this information to be retransmitted, possibly causing severe delays in the overall transmission.
  - Errors Sometimes packets are corrupted due to bit errors caused by noise and interference, especially in wireless communications and long copper wires. The receiver has to detect this and, just as if the packet was dropped, may ask for this information to be retransmitted.

QoS

- QoS types
  - Latency - It might take a long time for each packet to reach its destination, because it gets held up in long queues, or takes a less direct route to avoid congestion. This is different from throughput, as the delay can build up over time, even if the throughput is almost normal. In some cases, excessive latency can render an application such as VoIP or online gaming unusable.
  - Jitter - Packets from the source will reach the destination with different delays. A packet's delay varies with its position in the queues of the routers along the path between source and destination and this position can vary unpredictably. This variation in delay is known as jitter and can seriously affect the quality of streaming audio and/or video.
  - Out-of-order delivery - When a collection of related packets is routed through a network, different packets may take different routes, each resulting in a different delay. The result is that the packets arrive in a different order than they were sent. This problem requires special additional protocols responsible for rearranging out-of-order packets to an synchronous state once they reach their destination. This is especially important for video and VoIP streams where quality is dramatically affected by both latency and lack of sequence.

QoS

- Different applications have different demands:
  - E-mail – minimal error
  - FTP – minimal error, maximal capacity
  - WWW – Minimal error, big capacity, low latency
  - Audio on Demand – can have error, average latency, minimal jitter, big capacity
  - IP telephony – can have error, minimal latency, minimal jitter, good capacity (to understand voice you can lost 20% of information)
  - Videoconference – can have error, minimal latency, minimal jitter, maximal capacity
- Solution
  - Support for QoS on network layer – all routers must support QoS
  - Integrated Services
  - Differentiated Services

Integrated Services

- Flow specification – TSPEC
  - Token bucket algorithm
  - Use tokens to limit data transfer
  - Without token you cannot send data
- Resource Reservation Protocol – RSVP
  - All routers on path must support this function
  - Reserve capacity for specific transmission
  - Routers can accept requested reservation and once they accept reservation, they have to carry traffic
  - Router can reject request for reservation

Differentiated Services

- Smaller influence on network layer
- In December 1998, the IETF published RFC 2474 - *Definition of the Differentiated Services Field (DS field) in the IPv4 and IPv6 Headers*, which replaced the IPv4 TOS field with the DS field - a range of eight values (Class Selectors) is used for backward compatibility with the *IP precedence* specification in the former TOS field
- DiffServ operates on the principle of traffic classification, where each data packet is placed into a limited number of traffic classes, rather than differentiating network traffic based on the requirements of an individual flow. Each router on the network is configured to differentiate traffic based on its class. Each traffic class can be managed differently, ensuring preferential treatment for higher-priority traffic on the network.
- Per-Hop Behaviors:
  - *Default* PHB (Per hop behavior)—which is typically best-effort traffic
  - *Expedited Forwarding* (EF) PHB—dedicated to low-loss, low-latency traffic
  - *Assured Forwarding* (AF) PHB—gives assurance of delivery under prescribed conditions
  - *Class Selector* PHBs—which maintain backward compatibility with the IP Precedence field.

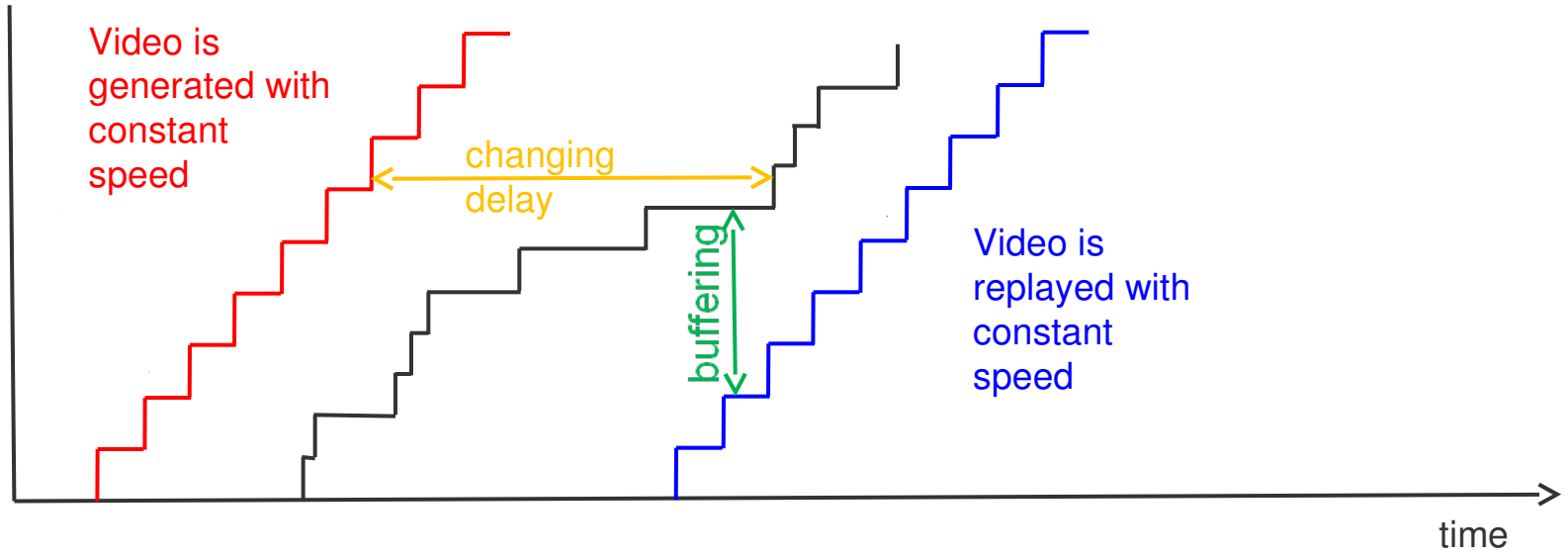
Application solution

Client buffering – QoS on application layer

- oneway no interactive multimedia (for example video playback) can remove "jitter" with buffering
- interactive video can use buffer too, but the overall delay cannot be too big

RTSP – Real Time Streaming Protocol

- Application protocol
- Enable mutual set-up on speed transfer



RTP/RTCP

Support for QoS on transportation layer

- standard for packing multimedia data into transported packets with support according to type of multimedia
- no influence to network layer and type of network transfer – best effort

RTP – Real Time Protocol

- creates it's own multimedia packets and this packet are inserted into UDP packet
- add information
    - multimedia type (PCM 64kbps, GSM 13kbps, Motion JPEG, MPEG2 video)
    - order of packets – each packet has sequentional number, enable to detect lost packets and order
    - timestamp – time when the data were created, improve buffering on client side
    - stream identification – one RTP can be used for more streams
    - support multicast

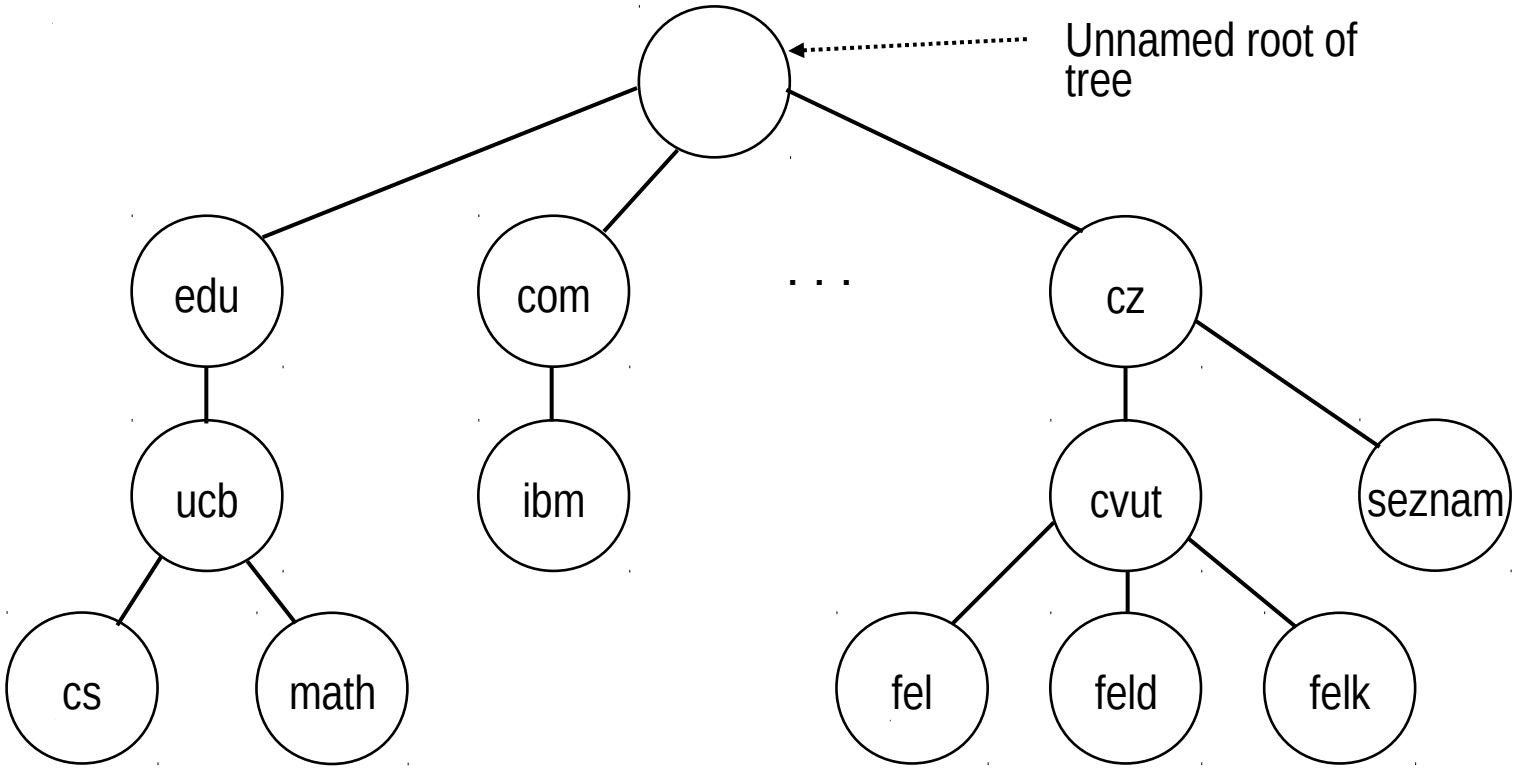
RTCP – Real Time Control Protocol

- exchange information between generator to receiver
- information about reliability of connection – ration of lost packets, average delay, dispersion of delay, ability of receiver
- description of RTP stream

Domain Name System (DNS)

- Users prefer names
  - Mapping ***computer name<=> IP address***
- Internet uses
  - hierarchical namespace
  - Independent on IP addresses
  -
- Flexible hierarchy of names
  - Computer names – *domain name*
  - Domain name syntax
  - Labels separated by '.'
  - example cyber.felk.cvut.cz
    - Last albel ('cz') *top-level domain* = TLD
    - Then subdomains

DNS Hierarchy



- Recursive and caching name servers
  - cvut makes registration by administrator of TLD cz and is responsible for all names in cvut.cz

Example of DNS configuration

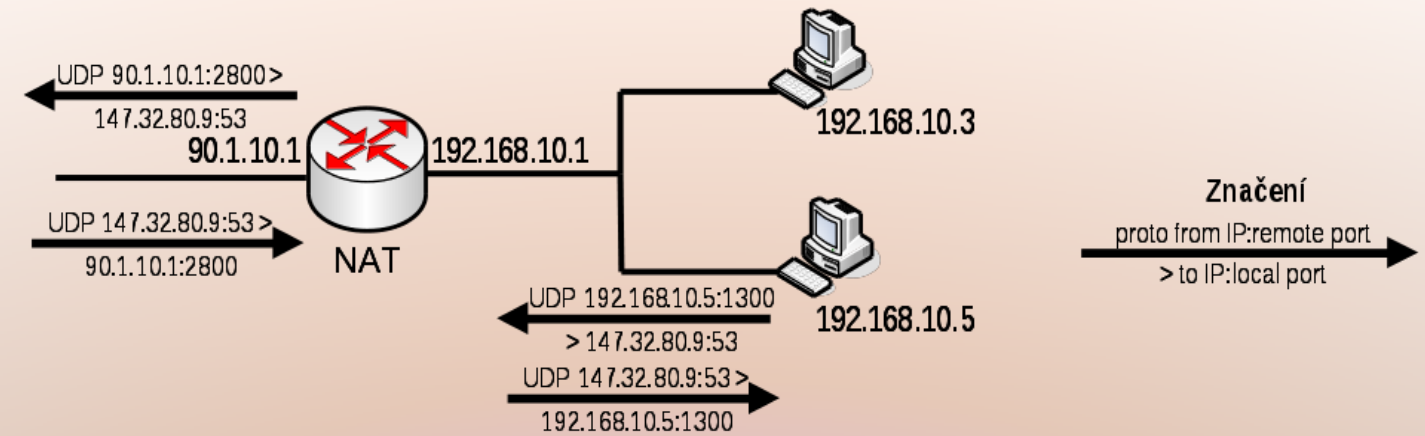
```
$ORIGIN firma.cz.
$TTL    12h
@               SOA  ns.domena.cz. spravce.mail.domena.cz.  (
                  2011090801 ; Serial
                  3h         ; Slave refresh (3 hours)
                  1h         ; Slave retry time in case of a problem (1 h)
                  2h         ; Slave expiration time (2 days)
                  3600       ; Maximum caching time of failed lookups (1 h)
                )
firma.cz.       NS          ns.firma.cz.
firma.cz.       NS          ns2.iol.cz.
firma.cz.       MX            0 mail.firma.cz.
firma.cz.       MX          100 relay.iol.cz.

ns              A     190.18.113.16
mail            A     190.18.113.16
www             A     190.18.113.14
extern-site     A     147.120.198.155
```

Reverz rezolution

- Direct resolution: domain name → IP_address
- Reverz resolutio: IP_address → domain_name
- Local resolver prepare query
  - Target IP is *aaa.bbb.ccc.ddd*
  - New query *ddd.ccc.bbb.aaa.in-addr.arpa*
    - Old top level domain *arpa* ←
  - Domain server receive PTR query with this name and finds answer

NAT – network address translation



- NAT analyze outgoing telegram
  - Remeber address and port of source computer
  - Replace local address with public IP address and new port
  - Send telegram to destination computer
- Answer is translated using table od source computer IP and port
- Easy for TCP
- Some complications with UDP
- Problems for ICMP

End of Lecture 13

Questions?