MINISTERSTVO ŠKOLSTVA, VEDY, VÝSKUMU A ŠPORTU SLOVENSKEJ REPUBLIKY

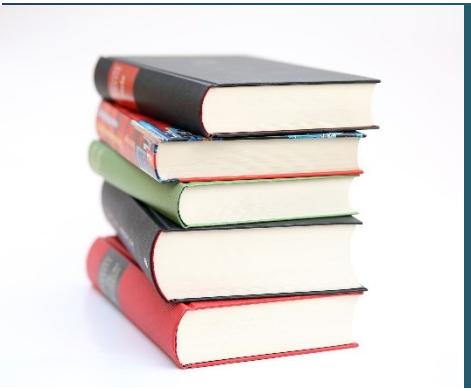
PIKS, prednáška 10

Transportná vrstva

Introduction to Networks v6.0 – Chapter 9

Katedra informačných sietí
Fakulta riadenia a informatiky, UNIZA

KIS FRI UNIZA – Vytvorené v rámci projektu KEGA 011STU - 4/2017.



Obsah prednášky

Časť 9.1: Transportné protokoly

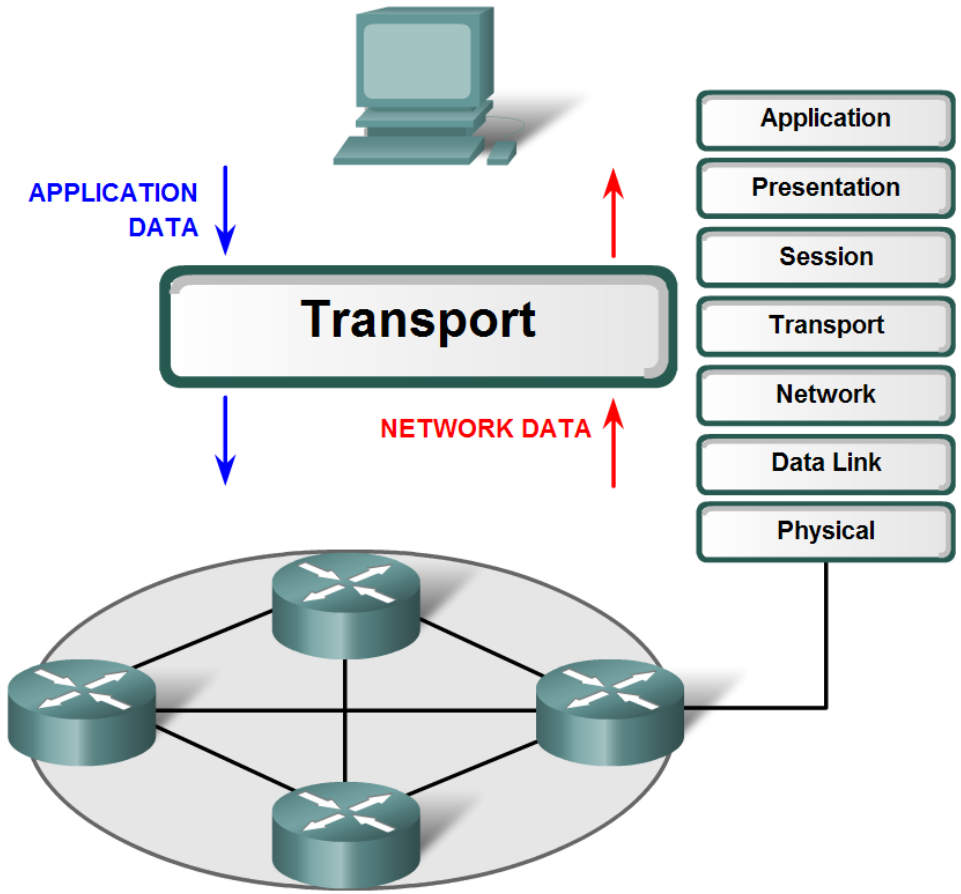
Na konci budeme vedieť:

- Popísať aké funkcie plní transportná vrstva pri manažovaní doručovania dát pri komunikácii medzi koncovými zariadeniami.
- Popísať charakteristiky TCP a UDP protokolov, vrátane čísiel portov a ich využitie.

Téma 9.1.1:
Úlohy transportnej vrstvy

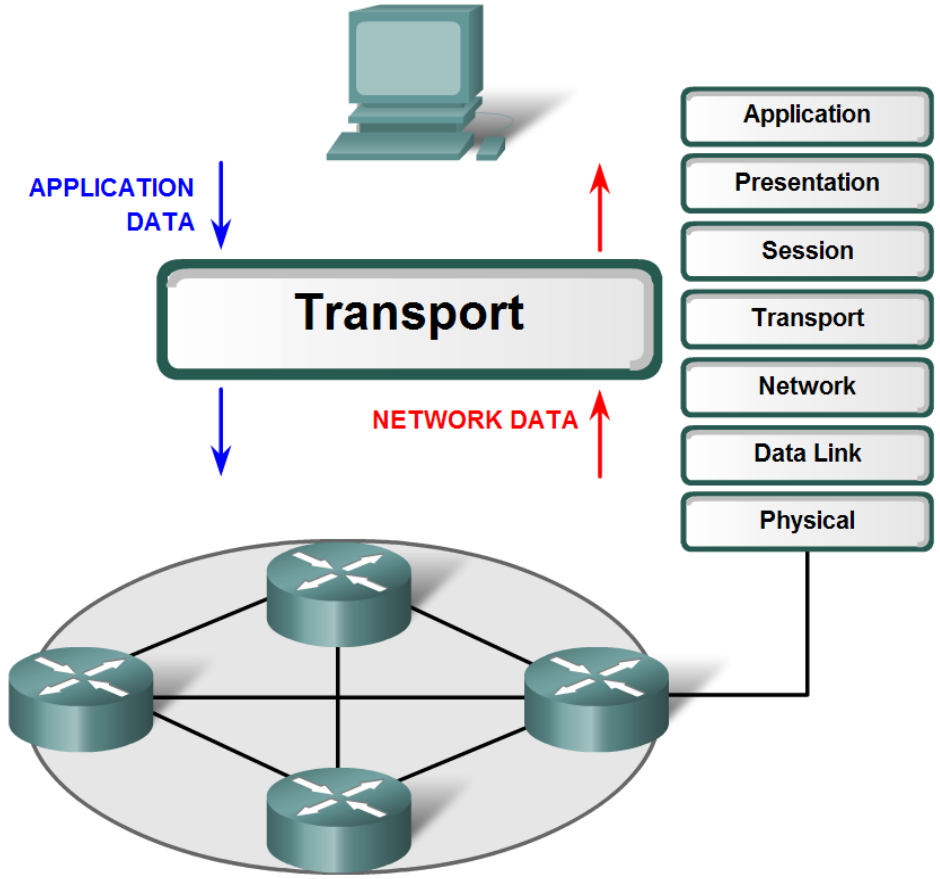
Prečo potrebujeme transportnú vrstvu?

- K transportnej vrstve OSI sa z vyšších vrstiev dostávajú dáta, ktoré
  - ... vytvoril istý aplikačný program alebo služba (OSI Layer7)
  - ... majú správny spoločný formát (OSI Layer6)
  - ... sú určené správnemu procesu a dialógu u príjemcu (OSI Layer5)
- Dáta v tomto tvare však nie sú na prenos sieťou vhodne pripravené
  - Zatiaľ nie sú segmentované na menšie úseky – sú v bloku, v akom ich vytvorila aplikácia, plus informácie z L7-L5 vrstvy
  - Ak sa budú sieťou prenášať ako oddelené segmenty, môže dôjsť k ich preusporiadaniu alebo strate
- Riešiť tieto nedostatky je práve úlohou transportnej vrstvy



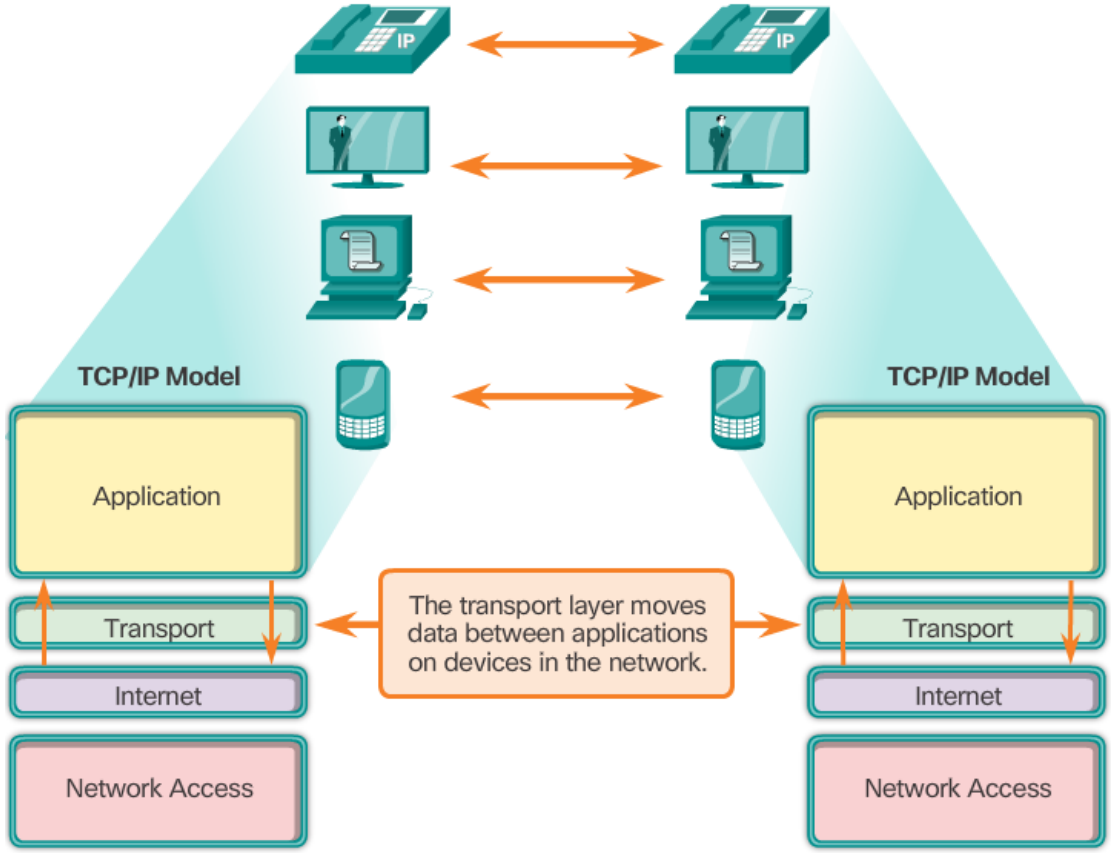
Úlohy transportnej vrstvy

- Transportná vrstva je na rozhraní medzi aplikačnými dátami a sieťovou infraštruktúrou
  - Zabezpečuje prenos dát medzi konkrétnymi komunikujúcimi **aplikáciami** na koncových uzloch
  - Dáta prenáša ako rad úsekov, tzv. **segmentov**
  - Tieto segmenty môžu niesť poradové čísla a iné info pre ich správne doručenie a opätovné poskladanie u príjemcu
- Transportná vrstva zodpovedá za doručenie dát cez sieť v takom tvare, v akom boli odoslané
  - ... ak to aplikácia žiada



Úlohy transportnej vrstvy v TCP/IP

- V protokolovom modeli TCP/IP plní transportná vrstva funkcie transportnej vrstvy RM OSI, no naviac pokrýva aj niektoré činnosti relačnej vrstvy



Úlohy transportnej vrstvy v TCP/IP

- **Oddelenie konverzácií**
  (Track individual conversations)
- **Segmentácia dát**
  (Segment data)
- **Spätná rekonštrukcia** pôvodných dát zo segmentov
  (Reassemble segments)
- **Identifikácia komunikujúcich aplikácií**
  (Identify the applications)

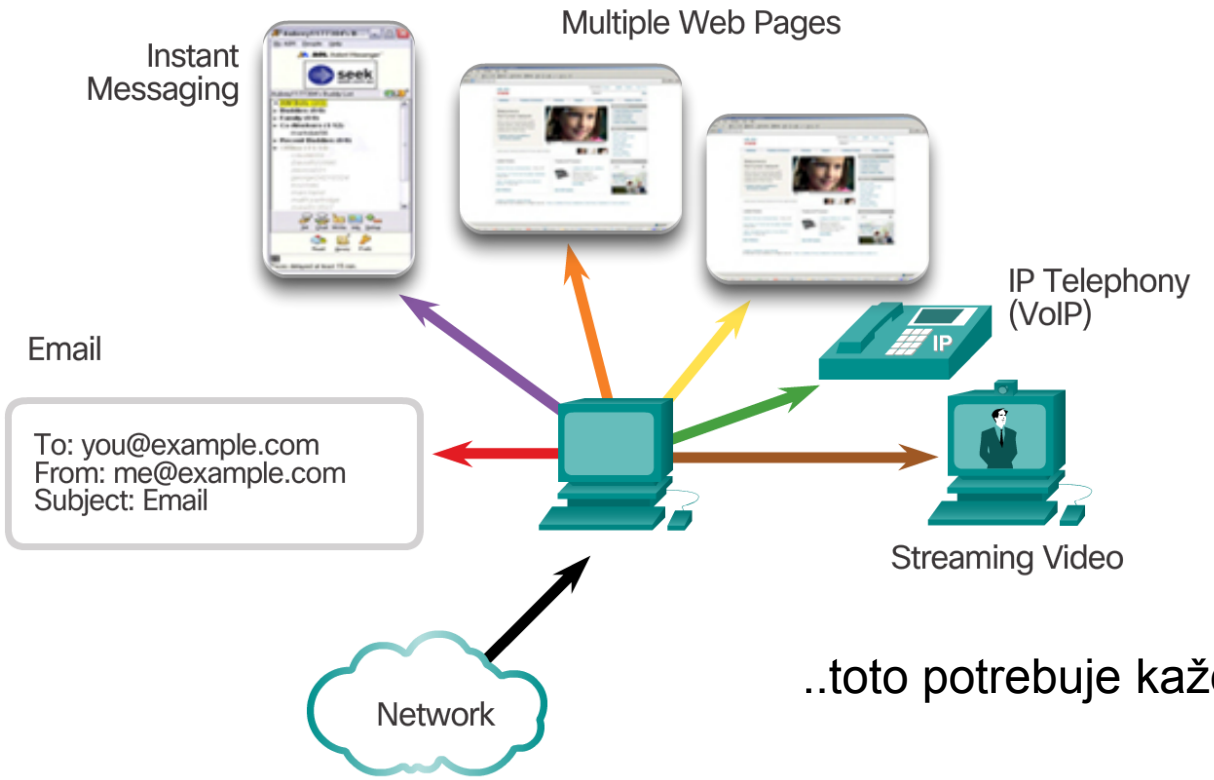
..základné funkcie

- **Spojovanosť**
  (Connection-oriented data stream support)
- **Spoľahlivosť**
  (Reliability)
  - **Usporiadanosť**
    (Delivery ordering)
- **Riadenie toku dát**
  (Flow control)

..rozširujúce funkcie
závisí od aplikácie, či ich potrebuje

Úlohy transportnej vrstvy v TCP/IP

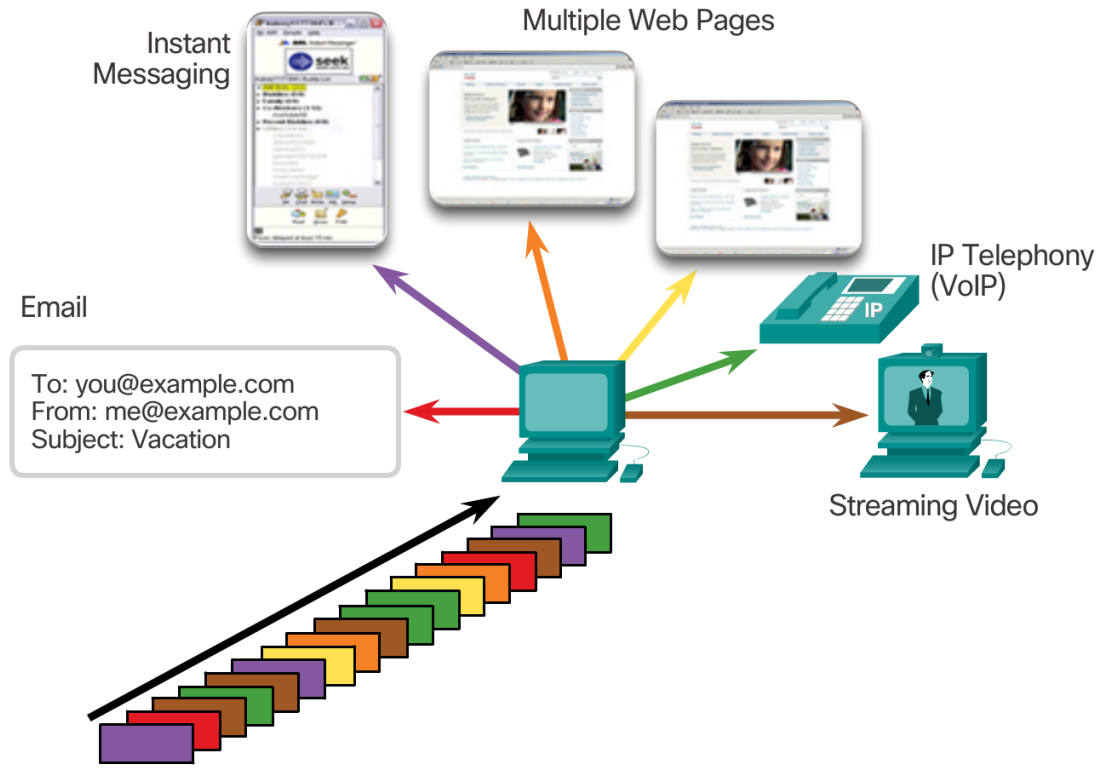
- Oddelenie konverzácií
  - Umožňuje, aby sa komunikujúce aplikácie dokázali navzájom správne adresovať (napr. webový server a klient)
    - Umožňuje oddeliť od seba súbežné konverzácie medzi tým istým klientom a serverom



..toto potrebuje každá aplikácia

Úlohy transportnej vrstvy v TCP/IP

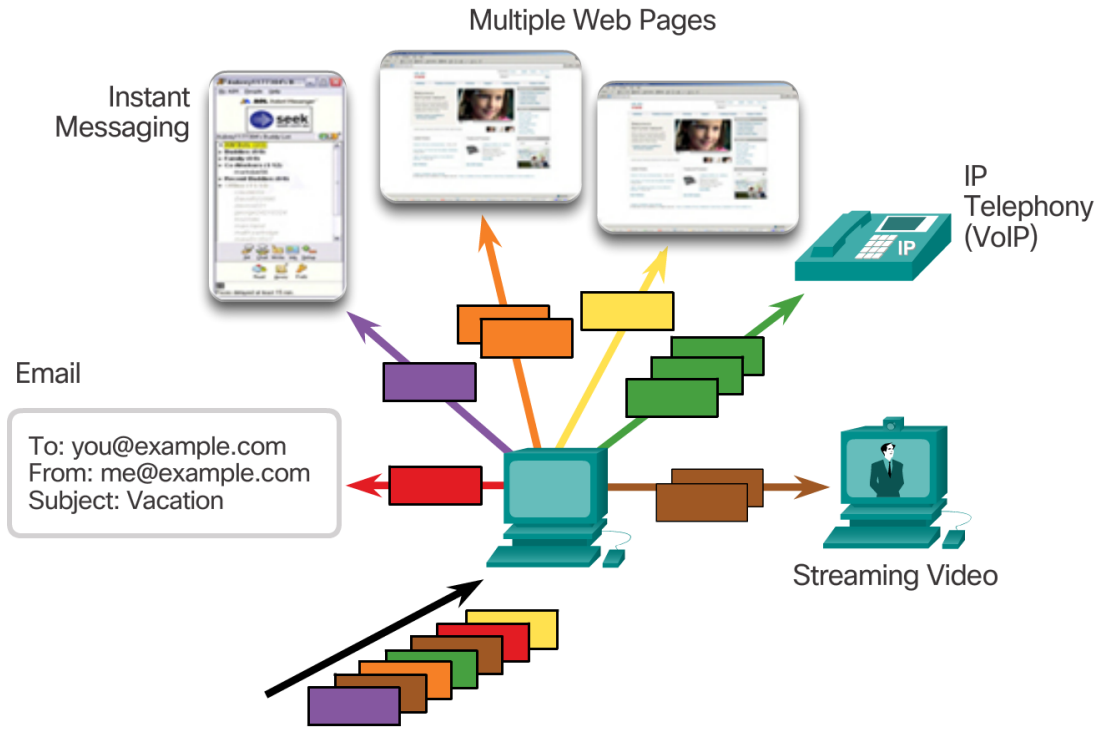
- **Segmentácia dát (Segment data) a spätná rekonštrukcia pôvodných dát zo segmentov (Reassemble segments)**
- Transportná vrstva pridá každému bloku hlavičku, ktorá:
  - identifikuje daný segment
  - umožňuje rôznym transportným protokolom vykonávať rôzne funkcie, ktoré sú potrebné pre manažovanie danej komunikácie pre danú aplikáciu
  - umožňuje multiplexovanie mnohých rôznych aplikácií od veľkého počtu používateľov na jednej sieti
    - tým že sa dáta rozdelia na menšie bloky, môžu viaceré aplikácie súbežne odosielať a prijímať dáta
  - uľahčuje prípadné riešenie problémov s poškodením prenášaných dát
    - checksum pre každý segment



..toto potrebuje každá aplikácia

Úlohy transportnej vrstvy v TCP/IP

- **Identifikácia komunikujúcich aplikácií** (Identify the Applications)
  - Zabezpečí, že aj keď na jednom zariadení beží niekoľko aplikácií, všetky dostanú tie dáta, ktoré sú pre ne určené



..toto potrebuje každá aplikácia

Úlohy transportnej vrstvy v TCP/IP

- Aplikácie majú podľa svojho typu rôzne nároky na spôsob prenosu ich dát sieťou:
- **Spojovanosť** (Connection-oriented data stream support)
  - Proces, ktorým sa klient a server pred výmenou dát navzájom dohodnú o komunikácii

a) **Spojovo orientovaná komunikácia** (connection-oriented)
   - Najprv sa musí zostaviť spojenie, až potom je možné prenášať dáta.
   - Ak spojenie nie je vytvorené, aplikácia musí byť notifikovaná
   - Ak spojenie je prerušené, aplikácia musí byť notifikovaná
   - Po prenose dát je potrebné spojenie ukončiť

b) **Nespojovano orientovaná komunikácia** (connectionless)
   - Dáta je možné odosielať okamžite bez dohody

...závisí od aplikácie, či potrebuje **a)** alebo **b)**

Úlohy transportnej vrstvy v TCP/IP

- Aplikácie majú podľa svojho typu rôzne nároky na spôsob prenosu ich dát sieťou:
  - **Spoľahlivosť** (Reliability)
    - Proces, ktorý má zaistiť, že dáta budú prijaté presne v tom tvare, v akom boli odoslané
    a) **Spoľahlivý prenos** (reliable) garantuje, že prijaté dáta budú všetky
      - každý prenos dát musí byť potvrdený príjemcom, ak dáta neboli potvrdené, odosielateľ opakuje vysielanie
      - kontrolujú sa chyby prenesených dát

    b) **Nespoľahlivý prenos** (unreliable) negarantuje, že prijaté dáta budú všetky
  - **Usporiadanosť** (Delivery ordering)

    a) Rekonštrukcia segmentov **v pôvodnom/správnom poradí** (same-order delivery)

    b) Rekonštrukcia segmentov **v takom poradí ako prídu** (no order data reconstruction)

...závisí od aplikácie, či potrebuje **a)** alebo **b)**

Úlohy transportnej vrstvy v TCP/IP

- **Riadenie toku dát** (Flow control)
  - Schopnosť riadiť rýchlosť odosielania dát, aby sa predišlo zahlteniu prenosovej cesty a následným stratám segmentov

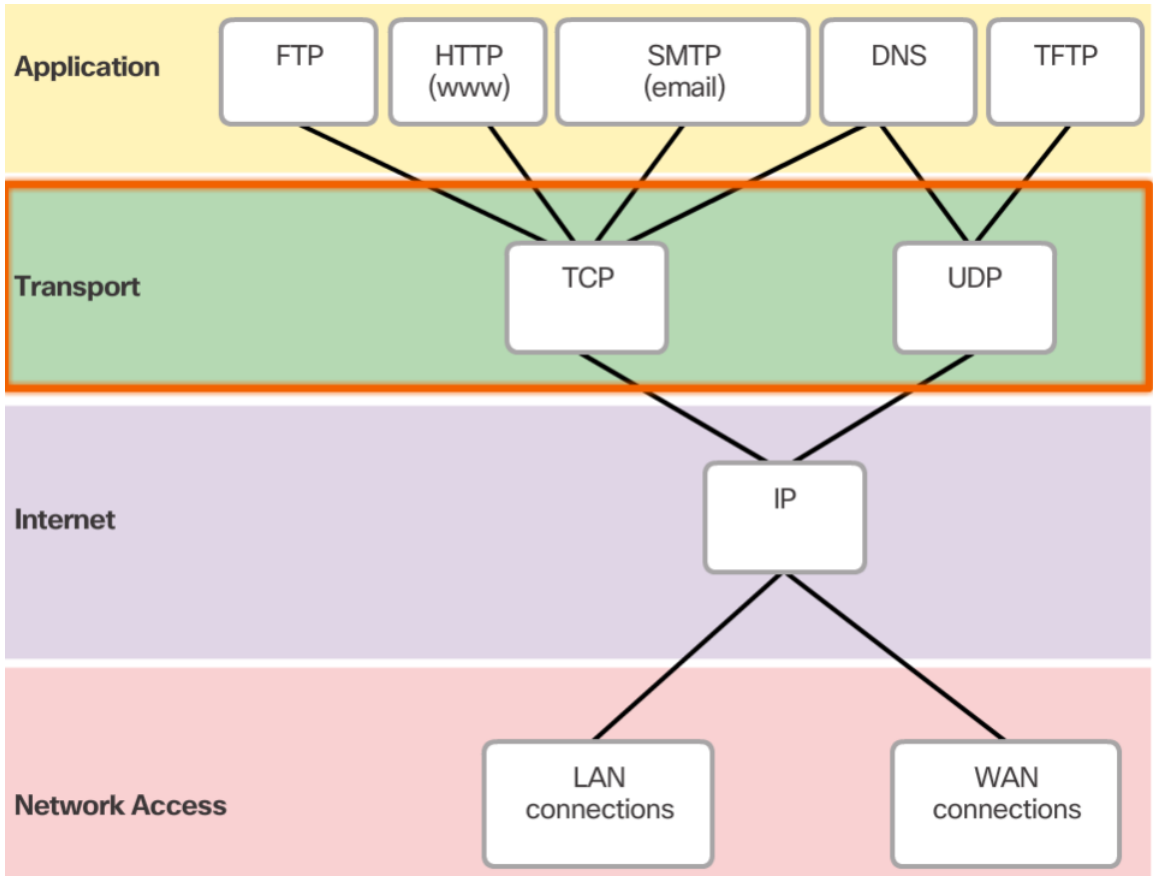
a) **S riadením toku dát** (flow control)

b) **Bez riadenia toku dát** (no flow control)

...závisí od aplikácie, či potrebuje **a)** alebo **b)**

Rozdielne nároky aplikácií na doručovanie dát

- Aplikácie majú podľa svojho typu rôzne nároky na spôsob prenosu ich dát sieťou
  - Spojovo resp. nespojovo orientovaná komunikácia
  - Spoľahlivý resp. nespoľahlivý prenos dát
    - Potvrdzovaný resp. nepotvrdzovaný prenos dát
  - Komunikácia s riadením resp. bez riadenia toku dát
- V TCP/IP architektúre sú najčastejšie používané transportné protokoly:
  - **TCP** (Transmission Control Protocol)
  - **UDP** (User Datagram Protocol)



Rozdiely medzi TCP a UDP

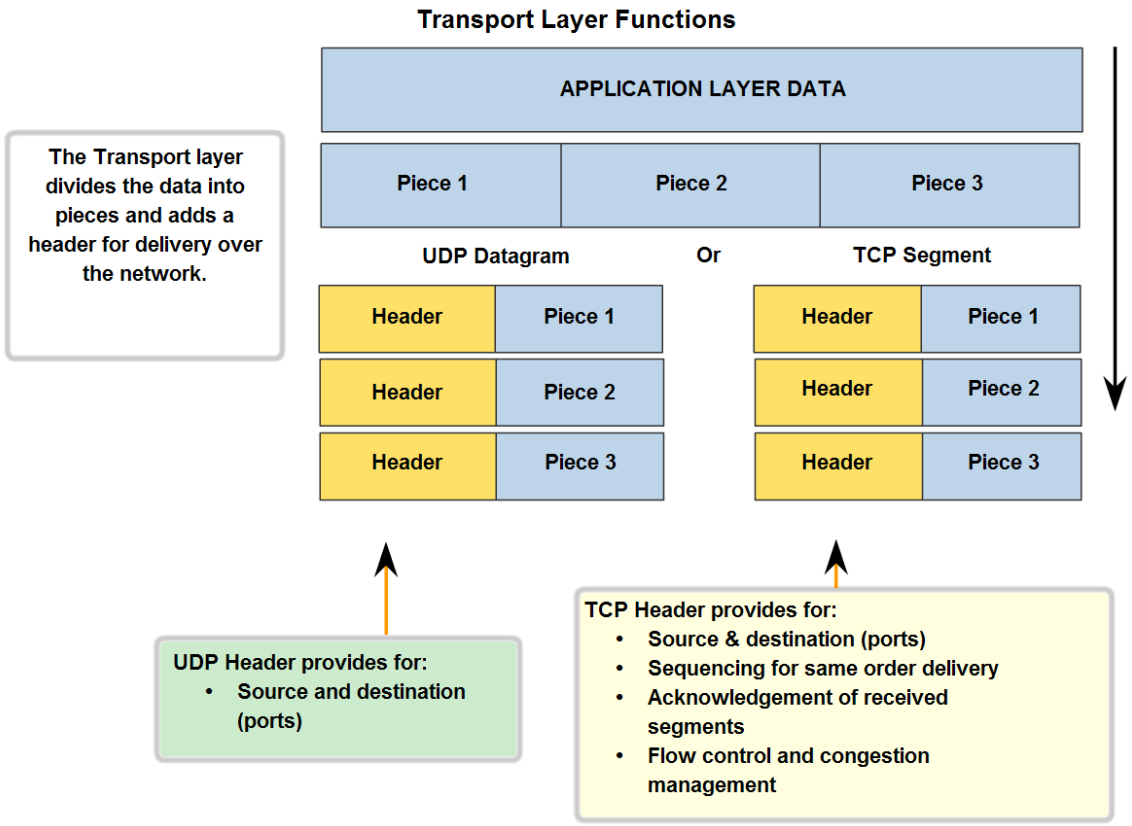
TCP	UDP
• Spojovo orientovaný ○ Vytvára obojsmerné spojenie	• Nespojovaný

• Spoľahlivý
  ○ Potvrdzovaný
	• Nespoľahlivý
	  ○ Nepotvrdzovaný

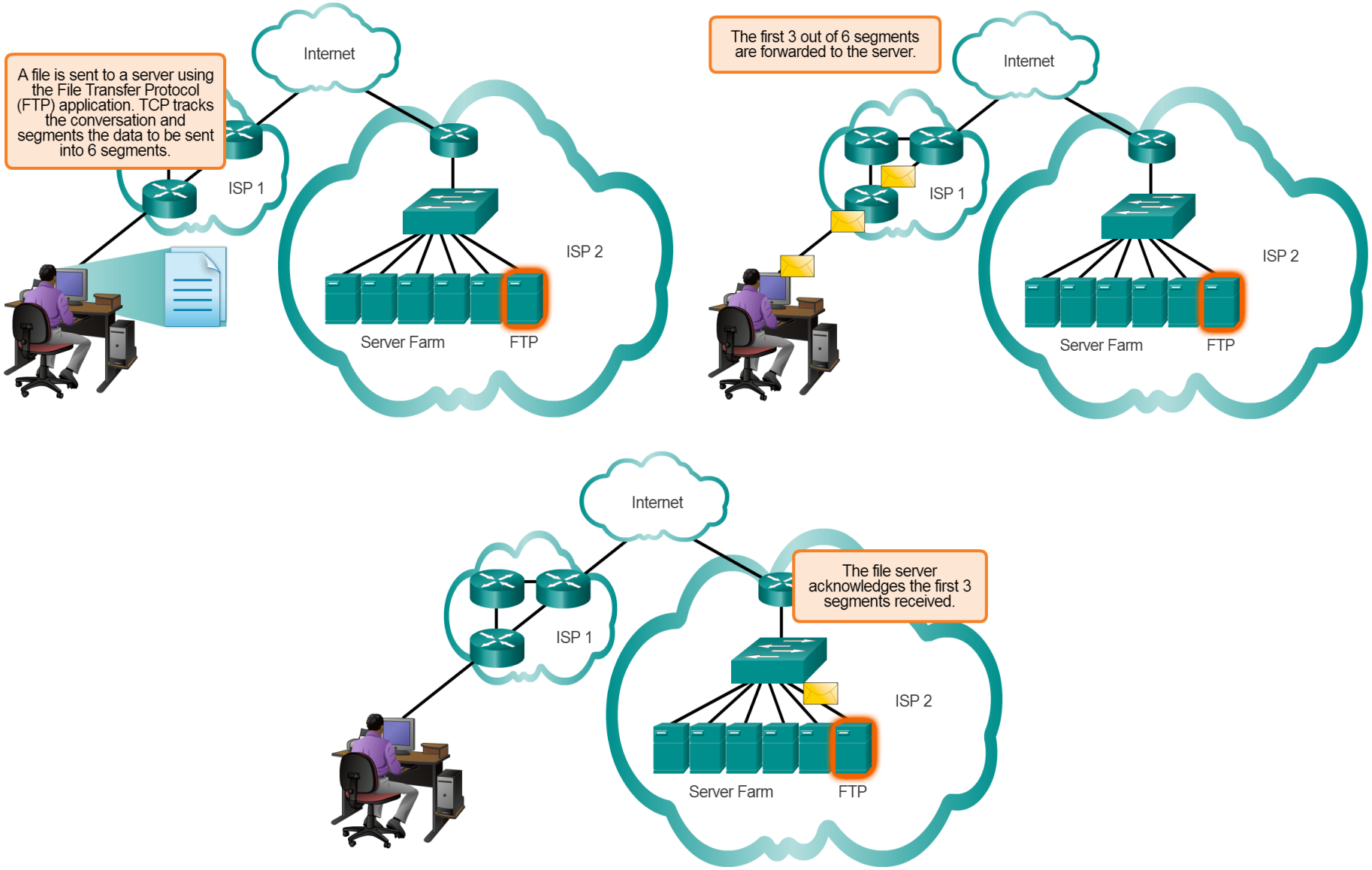
• Riadi tok dát	• Bez riadenia toku dát
• Bajtovo orientovaný ○ dáta sú transparentne prenášané ako kontinuálny tok dát	• Datagramovo orientovaný
• Veľkosť hlavičky: 20B bez prídavných informácií	• Veľkosť hlavičky: 8B

Rozdiely medzi TCP a UDP

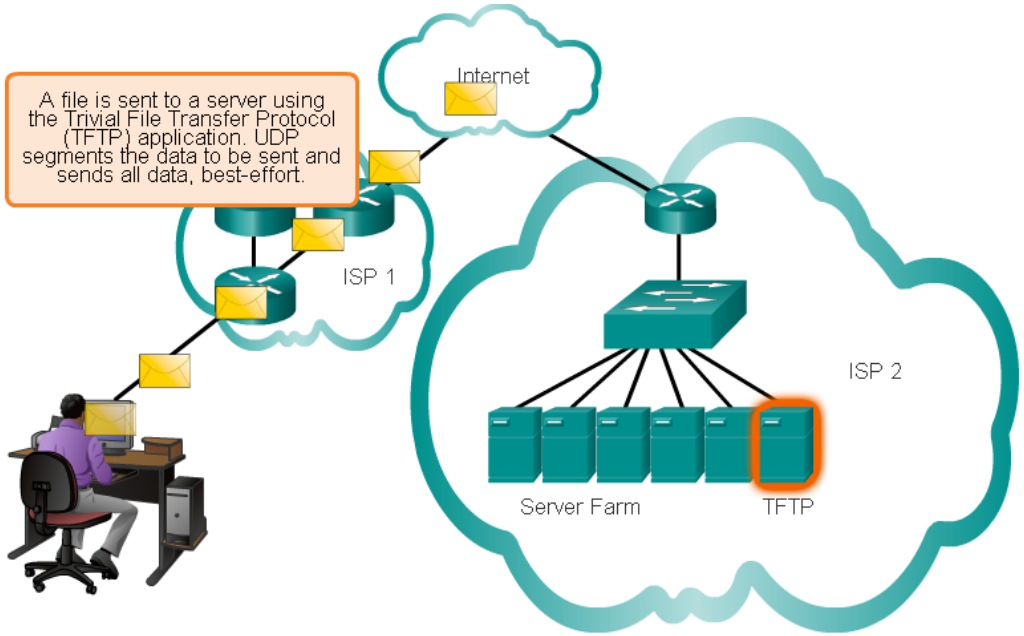
- TCP a UDP majú zámerne rozdielne vlastnosti, aby vyhoveli rôznym požiadavkám komunikujúcich aplikácií na prenos dát
  - Oba protokoly:
    - pomocou tzv. **čísel portov** identifikujú pár komunikujúcich procesov
    - pomocou **chechsum** identifikujú chybu pri prenose segmentu
  - UDP nemá žiadne ďalšie schopnosti
  - TCP navyše:
    - čísluje prenášané segmenty
    - potvrdzuje ich úspešné doručenie
    - zabezpečí opätovný prenos nedoručených alebo poškodených segmentov
    - zabezpečí použitie segmentov v pôvodnom poradí
  - UDP poskytuje tzv. nespojovanú datagramovú službu
  - TCP poskytuje tzv. spojovanú tokovú (stream) službu



TCP

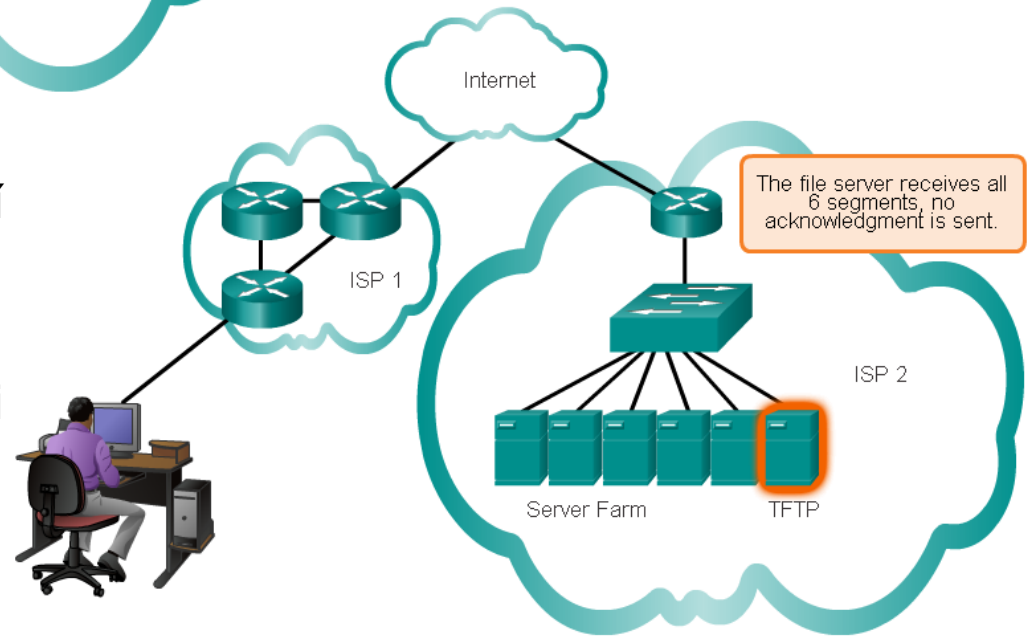

A file is sent to a server using the File Transfer Protocol (FTP) application. TCP tracks the conversation and segments the data to be sent into 6 segments.
Internet
ISP 1
ISP 2
Server Farm
FTP
The first 3 out of 6 segments are forwarded to the server.
Internet
ISP 1
ISP 2
Server Farm
FTP
Internet
ISP 1
The file server acknowledges the first 3 segments received.
ISP 2
Server Farm
FTP

UDP



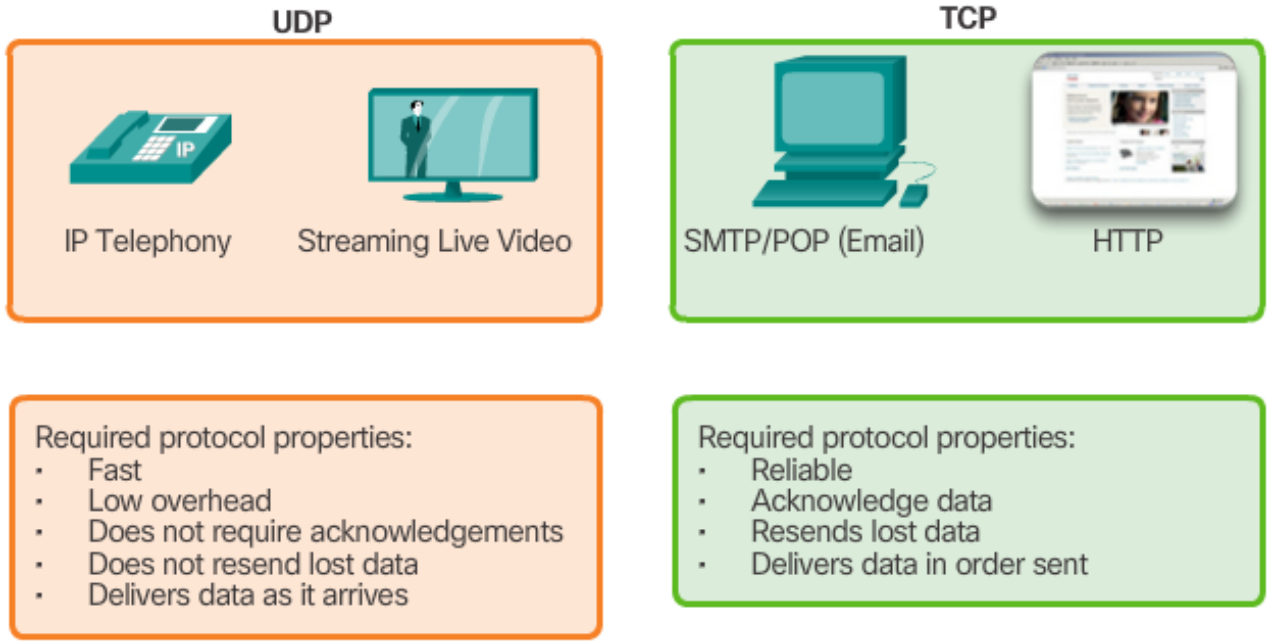
- Niektoré aplikácie nevyžadujú spoľahlivosť komunikácie – prináša to prídavnú réžiu a možné oneskorenie pri prenose
- Pridanie réžie môže pre niektoré aplikácie výrazne znížiť ich použiteľnosť a byť vlastne na škodu

- Ak aplikácia nevyžaduje spoľahlivý prenos, tak UDP je pre ňu vhodnejší transportný protokol
- Poskytne základné funkcie pre doručenie segmentov medzi danými aplikáciami s veľmi malou réžiou a kontrolou dát



Ktorý transportný protokol pre ktorú aplikáciu?

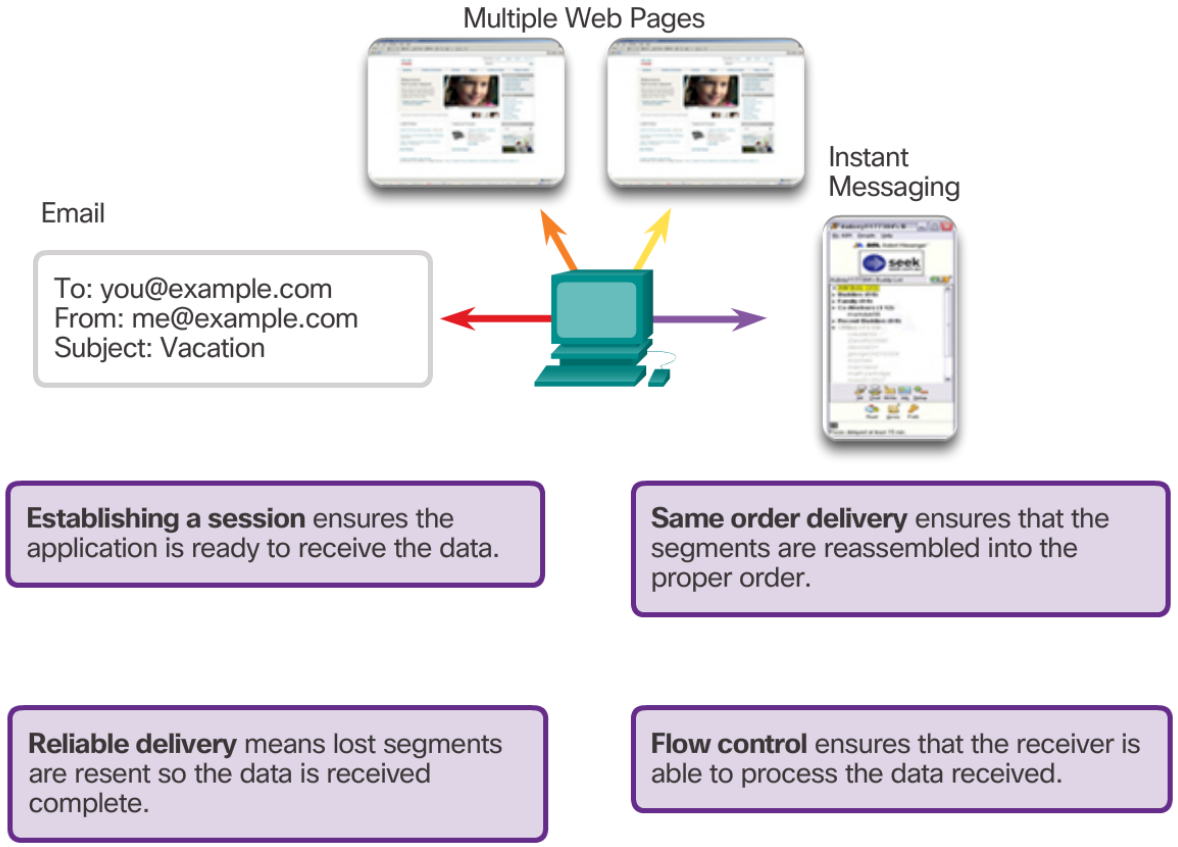
- TCP je vhodný pre aplikácie:
  - v ktorých segmenty musia doraziť v presne stanovenom poradí, aby mohli byť úspešne spracované
  - v ktorých všetky dáta musia doraziť do cieľa, aby boli použiteľné pre danú aplikáciu
  - napr. databázy, web prehliač, emailový klient, ...
- UDP je vhodný pre aplikácie:
  - ktoré vedia tolerovať straty, ale pre ktoré je neprijateľné veľké oneskorenie
  - napr. live audio/video streaming, Voice over IP, ...



Téma 9.1.2:
Polia hlavičiek TCP a UDP

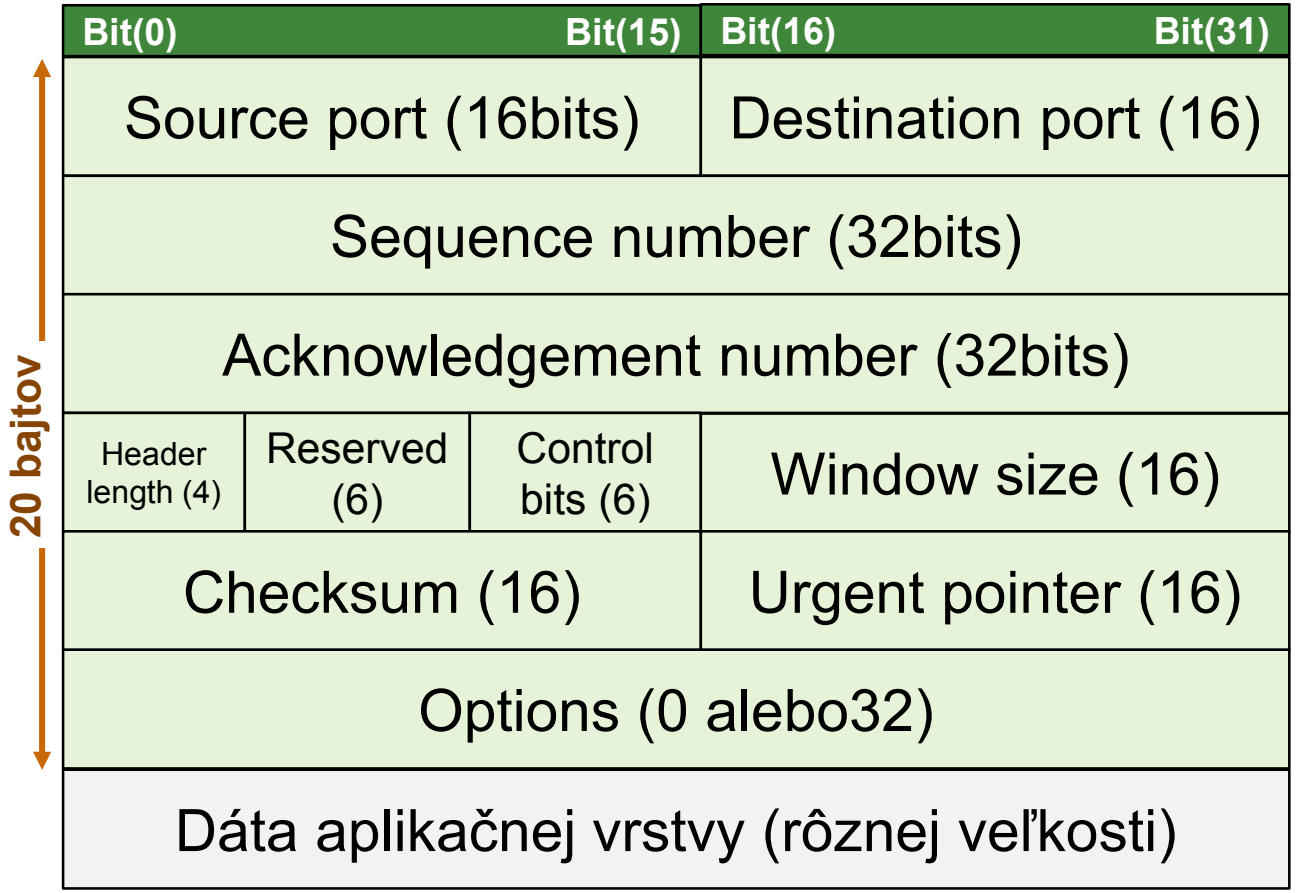
Služby TCP

- TCP poskytuje **spoľahlivú**, **spojovo orientovanú** službu doručovania tokov dát (streams) **s riadením toku** (flow control)
- TCP je podstatne komplexnejší a zložitejší ako UDP
  - 20 B réžia v hlavičke
- TCP je **stavový** protokol (statefull)
  - Udržiava si záznam o stave relácie tým, že si zapamätá, ktorú informáciu už poslalo a ktorú informáciu už potvrdilo (že prišla)



Polia TCP hlavičky

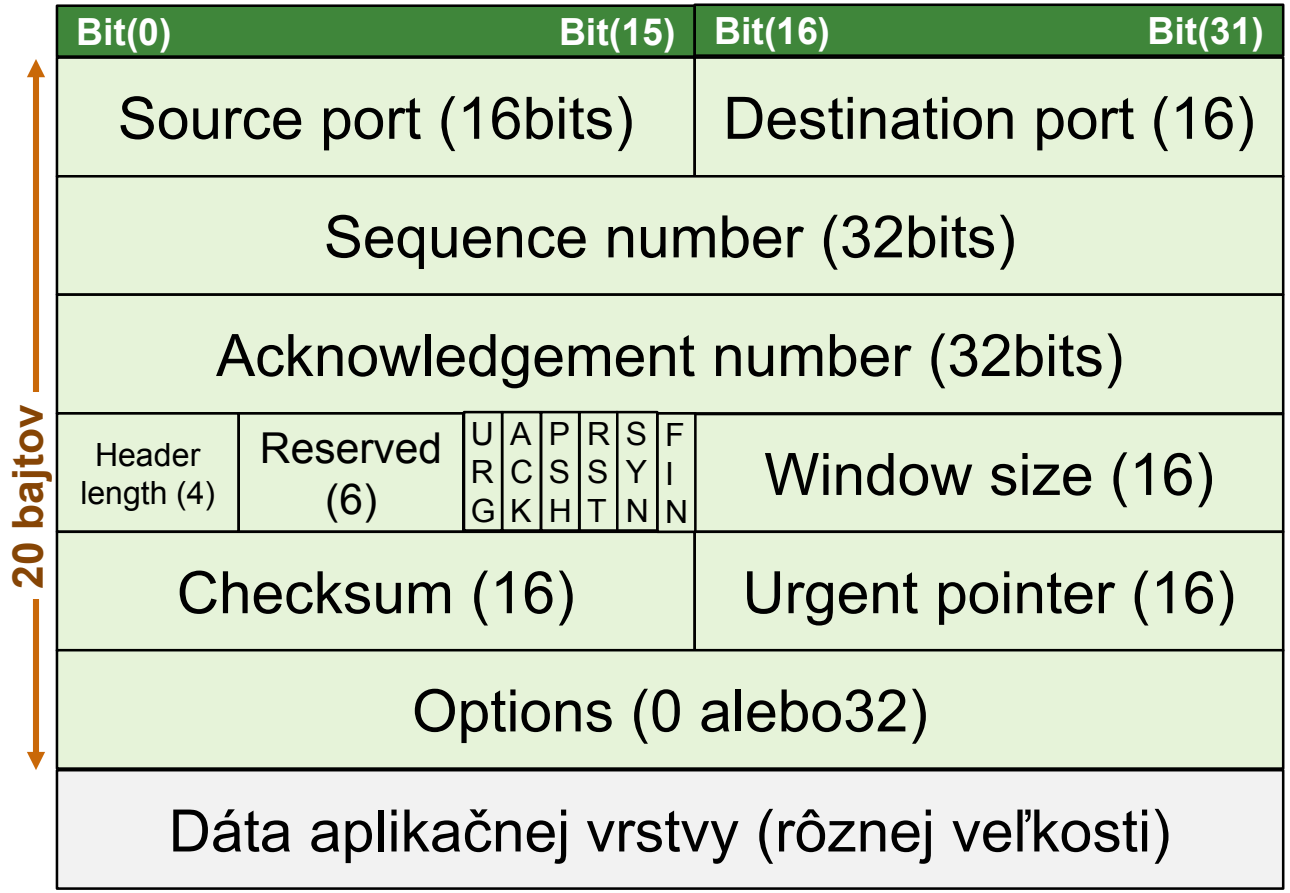
- **Source Port**
  - Zdrojový TCP port
- **Sequence Number**
  - Poradové číslo odoslaného segmentu
- **Acknowledgement Number**
  - Potvrdenie prijatých segmentov (vyjadrené B)
- **Header Length**
  - Veľkosť hlavičky v 4B slovách
- **Reserved**
  - Rezervované pole, nepoužité bity, vždy s hodnotou 0
- **Destination Port**
  - Cieľový TCP port



Polia TCP hlavičky

- **Window size**
  - Oznamovaná veľkosť okna pre príjemcu tohto segmentu (neposielaj mi segmenty rýchlejšie)
- **TCP Checksum**
  - Kontrolná suma celého TCP segmentu
- **Urgent Pointer**
  - Ukazateľ na posledný bajt urgentných dát
- **Options**
  - Doplňuj. voľby, nepovinné
- **Control bits**
  - 6 jednobitových príznakov (Flags)
    - URG, ACK,PSH, RST, SYN, FIN

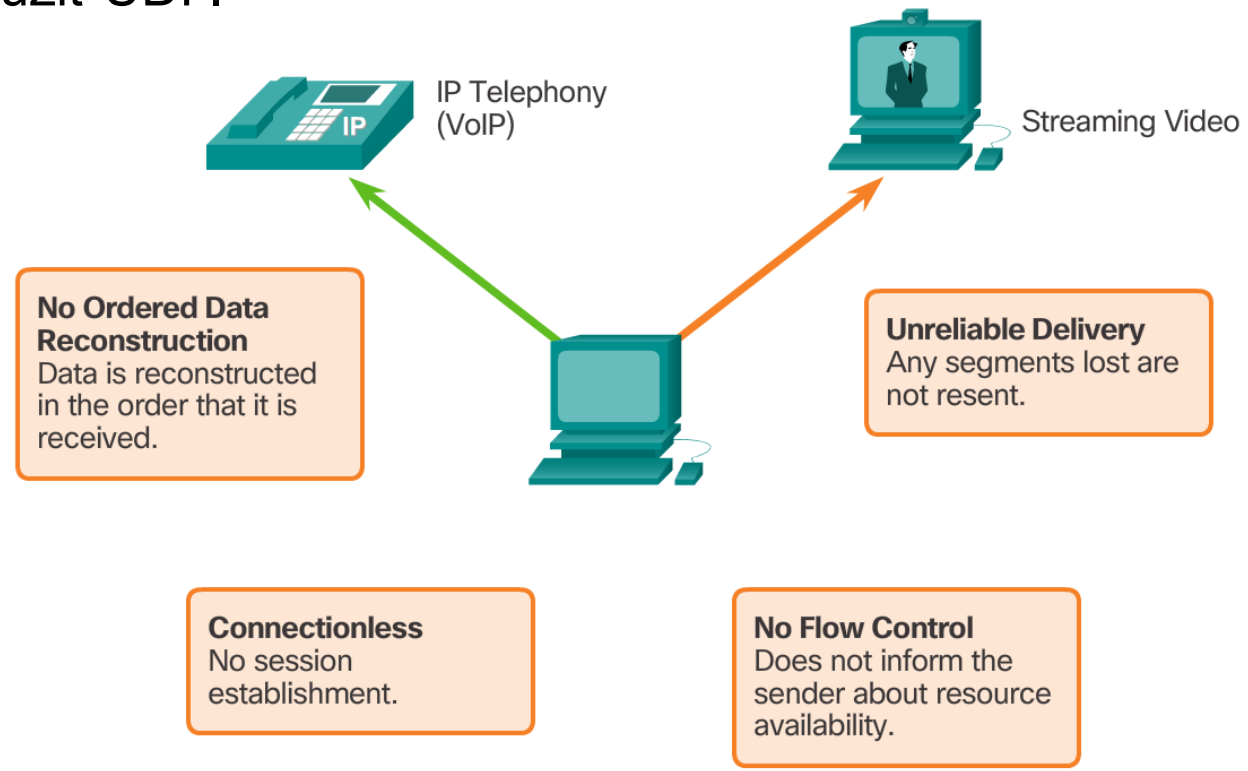
- ACK = acknowledgement
  Segment potvrdzuje prijaté dáta
- SYN = synchronization
  Synchronizačná značka pri vytváraní spojenia
- RST = reset
  Reset spojenia (odmietnutie alebo neočakavané ukončenie)
- PSH = push
  Pošli dáta hneď, nečakaj na naplnenie MSS
- URG = urgent
  Segment obsahuje urgentné dáta
- FIN = finish
  Nemám viac dát, končím.



Služby UDP

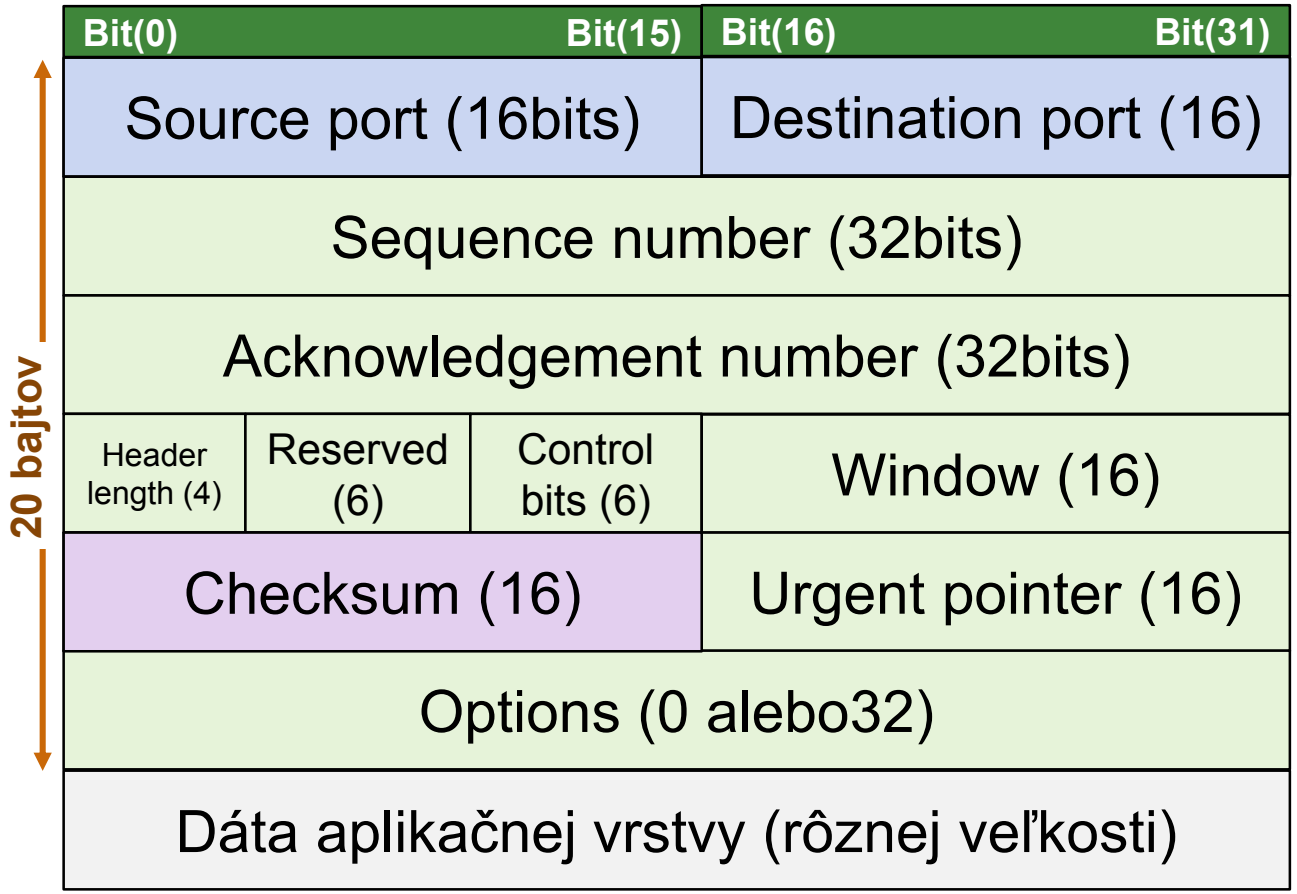
- UDP je bezstavový protokol (stateless) – ani odosielateľ ani príjemca nie sú povinný udržiavať záznam o stave komunikačného spojenia.
  - Spoľahlivosť musí zabezpečiť aplikácia.
- Aplikácie pre prenos videa a hlasu v reálnom čase musia rýchlo doručiť dáta, tolerujú aj nenulové straty, preto je vhodné pre ne využiť UDP.
- Aplikačné dáta + UDP hlavička
  = tzv. UDP datagram.
- Má nízku réžiu 8B.
- UDP posiela datagramy ako best-effort:

  „*Pokúsim sa doručiť, ale za nič neručím.*“



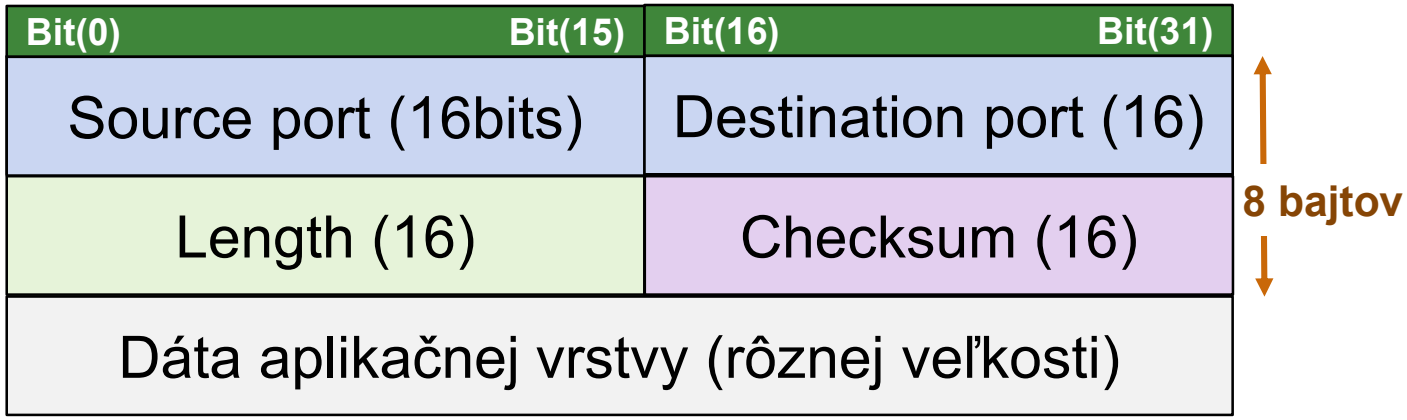
Hlavičky

- Každý segment TCP aj UDP obsahuje samostatnú hlavičku, ktorá nesie informácie potrebné pre správne spracovanie segmentu u príjemcu
- Oba majú 2 adresné polia a kontrolný súčet (checksum)



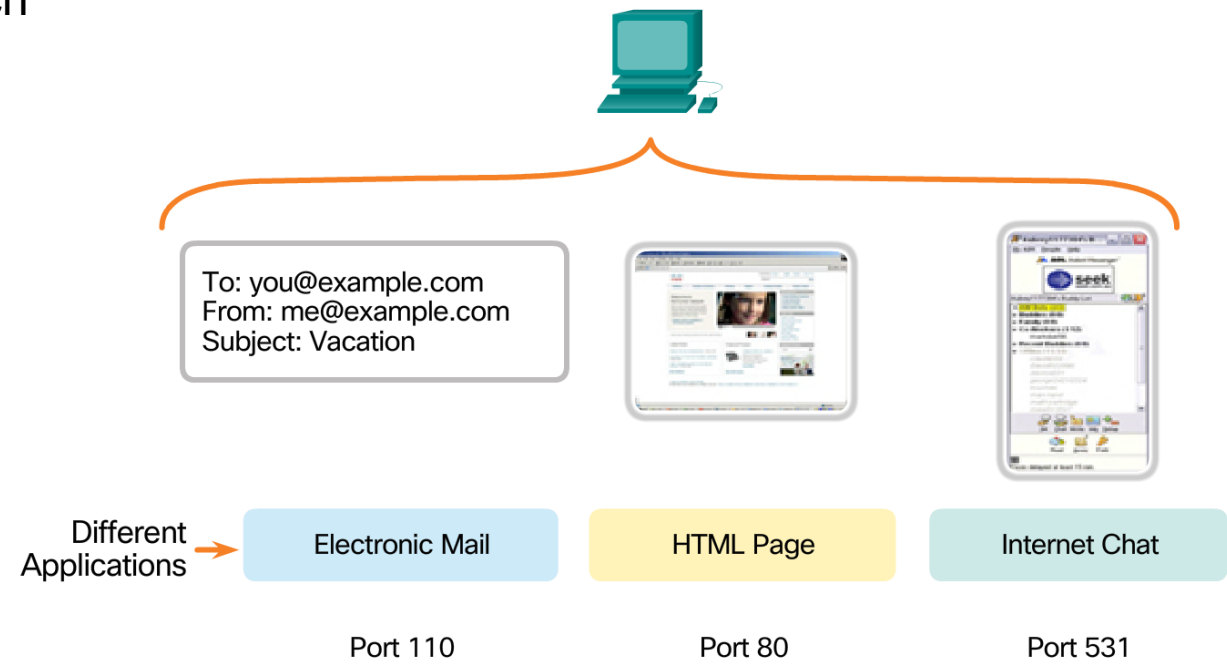
TCP

UDP



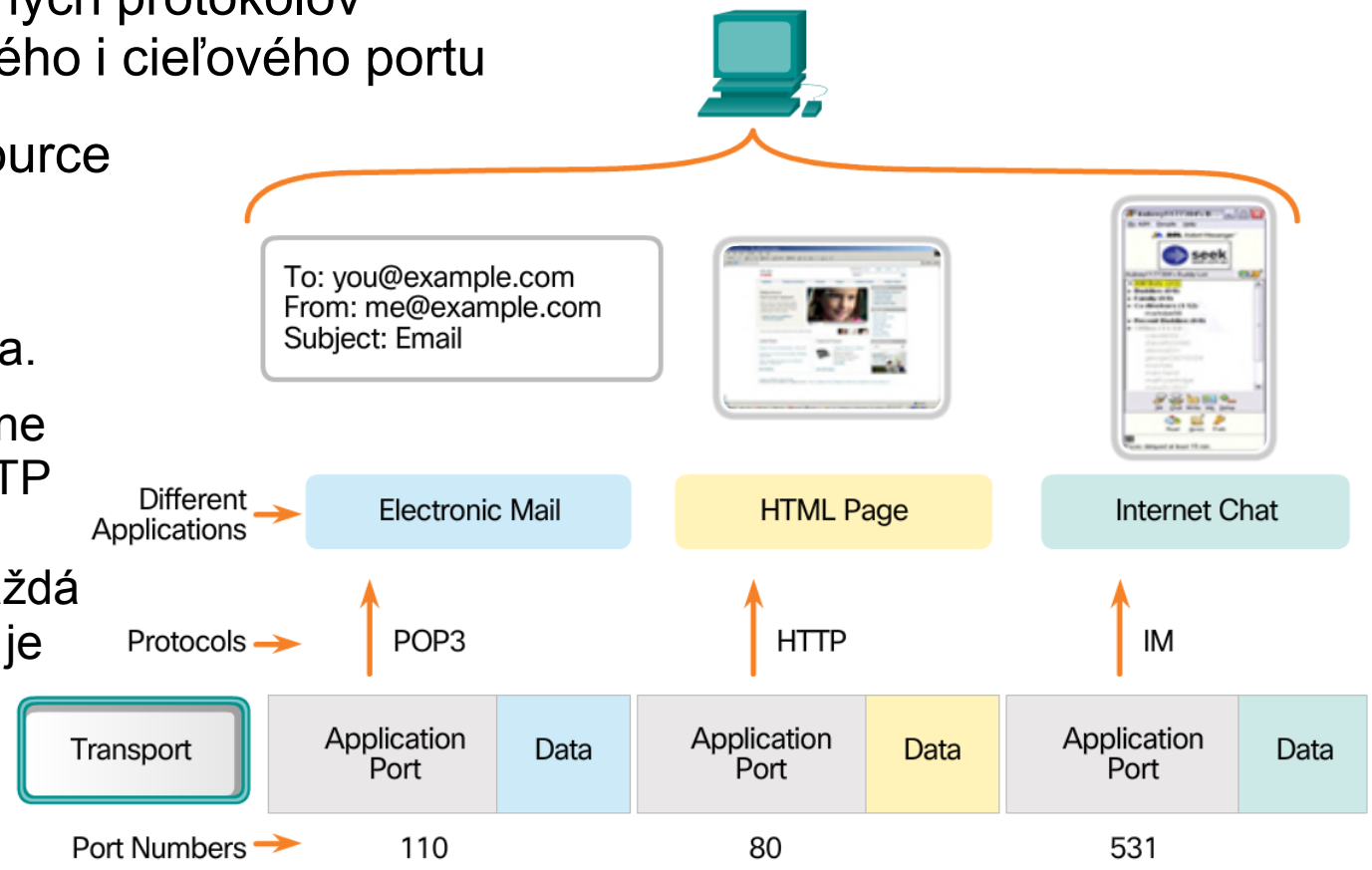
Ako odlíšiť viaceré konverzácie

- Rôzne aplikácie posielajú a prijímajú dáta cez sieť v jednom čase, s rôznymi požiadavkami na prenos.
  - Transportná vrstva ich musí odlíšiť a manažovať.
  - Používa na to jedinečné identifikátory – čísla portov
- Rôzne procesy na tom istom uzle dostanú pre svoje spojenia nad daným transportným protokolom rôzne porty
  - Čísla portov v rôznych transportných protokoloch sú **nezávislé**, t.j. TCP/80 je iný ako UDP/80
- Pojmom **transportný port** sa označuje číslo, ktoré **operačný systém** priradí konkrétnej komunikujúcej **aplikácii** (procesu) pre **konkrétne sieťové spojenie** v danom transportnom protokole



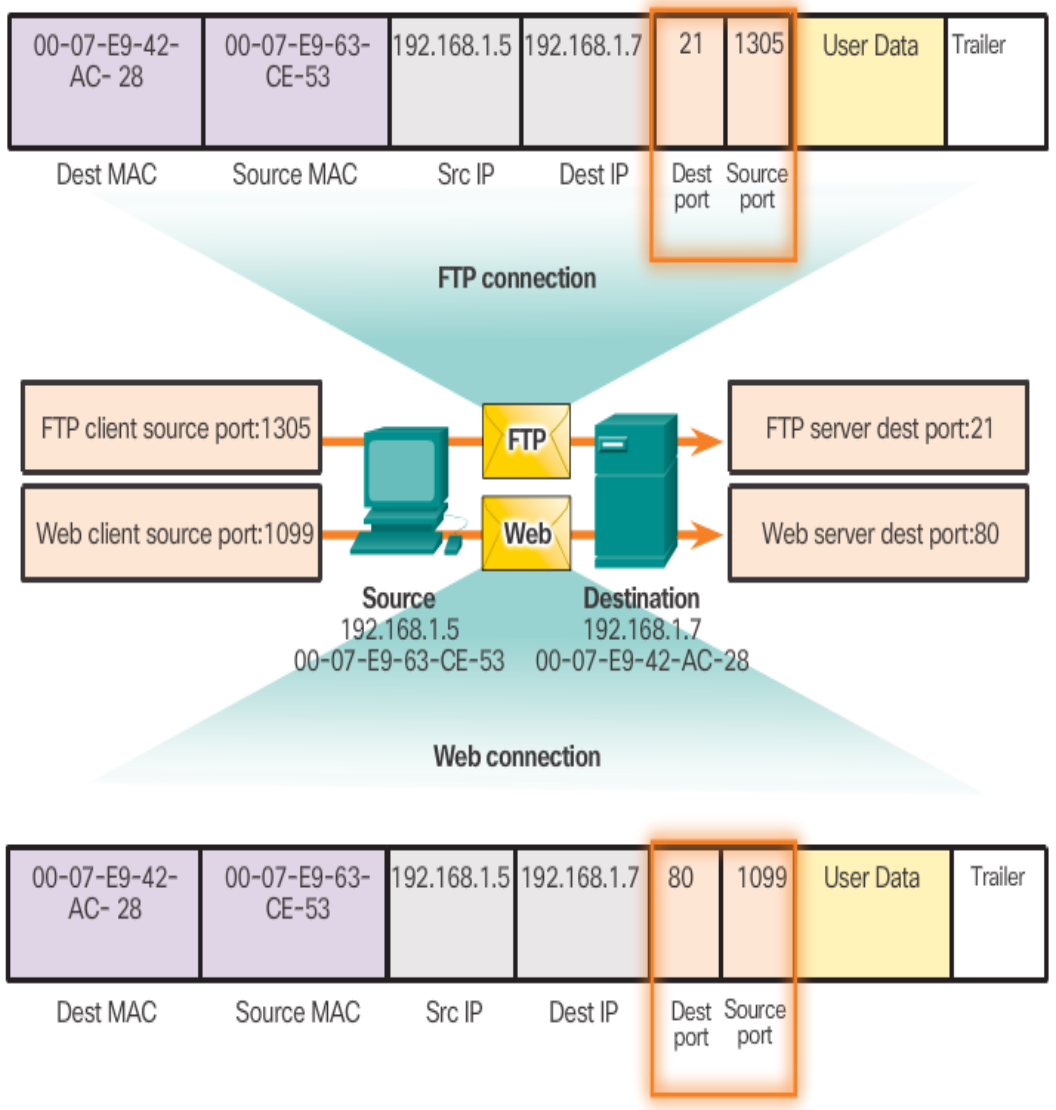
Význam transportného portu

- Hlavičky transportných protokolov nesú čísla zdrojového i cieľového portu
- **Zdrojový port** (Source port)
  - Volí dynamicky odosielajúca stanica.
  - HTTP klient obyčajne posiela viaceré HTTP požiadavky na web server v 1 čase. Každá takáto konverzácia je identifikovaná zdrojovým portom.
- **Cieľový port** (Destination Port)
  - Na identifikáciu aplikácie alebo služby bežiacej na serveri.
  - Server môže ponúkať viac ako 1 službu v jednom čase, napr. web službu na porte 80 a FTP na porte 21.



Socket

- Kombinácia IP adresy uzla, transportného protokolu a portu sa nazýva **socket** a uvádza sa v tvare **IPadresa:port**
  - t.j. zdrojová IP adresa:zdrojový port je jeden socket
    - zdrojový port slúži ako spätná adresa pre odpoveď klientovi
  - cieľová adresa:cieľový port je druhý socket
    - používa sa na identifikáciu servera a služby, ktorú požaduje daný klient.
- Dvojica socketov **jednoznačne** identifikuje pár komunikujúcich procesov
  - 192.168.1.5:1099
    192.168.1.7:80
  - Sockety umožňujú odlíšiť od seba viaceré procesy bežiace na klientovi a viaceré spojenia na danom serveri.
- Je úlohou transportnej vrstvy udržiavať zoznam aktívnych socketov.



Skupiny čísiel transportných portov

- Čísla portov v TCP a UDP sú 2-bajtové a ich rozsah (2^{16}=65536) je rozdelený do **3 skupín**
- Porty z 1. a 2. skupiny prideľuje **IANA**
  (Internet Assigned Numbers Authority)
- Porty z 3. skupiny prideľuje konkrétny operačný systém pre aplikáciu/službu, ktorá o to žiada:
  - Buď dynamicky, ak proces nežiada žiadne špecifické číslo portu (DYNAMIC)
  - Alebo ak proces požiada o nejaké špecifické číslo, pridelí to (PRIVATE)

Skupiny

Port Number Range	Port Group
0 to 1023	Well-known Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

Well-Known Port Numbers

Port	Protocol	Application	Acronym
20	TCP	File Transfer Protocol (data)	FTP
21	TCP	File Transfer Protocol (control)	FTP
22	TCP	Secure Shell	SSH
23	TCP	Telnet	–
25	TCP	Simple Mail Transfer Protocol	SMTP
53	UDP, TCP	Domain Name Service	DNS
67	UDP	Dynamic Host Configuration Protocol (server)	DHCP
68	UDP	Dynamic Host Configuration Protocol (client)	DHCP
69	UDP	Trivial File Transfer Protocol	TFTP
80	TCP	Hypertext Transfer Protocol	HTTP
110	TCP	Post Office Protocol version 3	POP3
143	TCP	Internet Message Access Protocol	IMAP
161	UDP	Simple Network Management Protocol	SNMP
443	TCP	Hypertext Transfer Protocol Secure	HTTPS

Skupiny čísiel transportných portov

1. skupina: 0 - 1023: tzv. **well-known ports**

- Porty v tomto rozsahu sú rezervované pre serverovské služby (HTTP, FTP, POP3, SMTP, ...)
- Na týchto portoch počúvajú servery
- Proces, ktorý chce počúvať na well-known porte, musí pri svojom spustení v operačnom systéme mať zvýšené privilégiá
- Prideľuje sa procesom štandardizácie protokolov organizáciou IETF (viac v RFC6335). Nazývajú sa aj ako tzv. „system ports“. Zoznam udržiava IANA.

2. skupina: 1024 - 49151: tzv. **registered ports**

- Porty v tomto rozsahu sú určené pre používateľské služby a procesy
- Prideľuje IANA na požiadanie pre nejaký špecifický proces alebo aplikáciu
  - IANA má na prideľovanie svoj systém - využíva proces „IETF review“ a proces „IESG approval“, alebo „Expert review“ (viac v RFC6335)
  - Napr. Cisco má registrovaný port 1985 pre HSRP (Hot Standy Routing Protocol) – t.j. musel prejsť vyššie spomínaným procesom
- Na týchto portoch môžu počúvať klienti i servery (zväčša však servery)

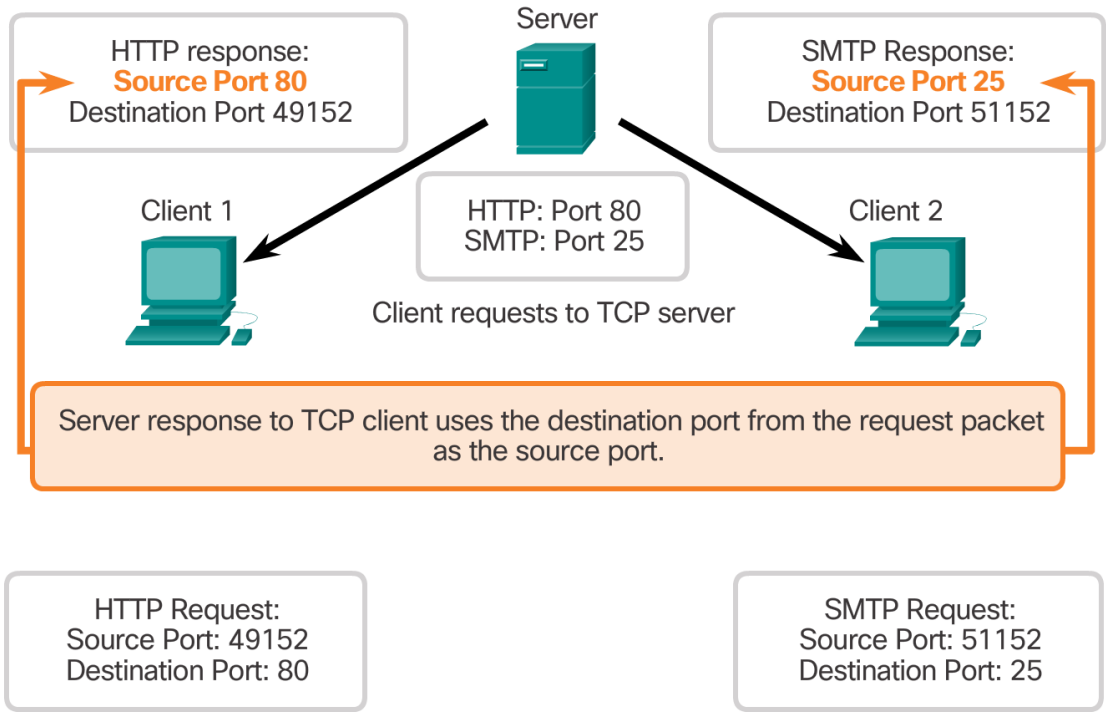
Skupiny čísiel transportných portov

- 3. skupina: 49152 - 65535: tzv. private/dynamic
  - Porty v tomto rozsahu sú určené najmä pre klientské procesy
  - Tieto porty neprideľuje IANA, pridelí ich daný operačný systém (OS)
  - Proces v OS môže získať číslo portu buď:
    - dynamicky, t.j. bez požiadania o konkrétnu hodnotu (tzv. dynamic port)
    - alebo požiada OS o pridelenie špecifického portu (tzv. private port)
  - Označujú sa aj ako dočasné = ephemeral ports, keďže OS ich pridelí zväčša len na nejaký čas, kým je to potrebné pre danú aplikáciu/službu. Potom ich môže prideliť zase úplne inej aplikácii.
  - Na týchto portoch počúvajú najmä klienti, servery len zriedka
- Väčšinou teda platí, že:
  - Servery počúvajú na vopred známych, vždy rovnakých portoch (1. a 2. skupina čísiel)
  - Klienti používajú dynamicky pridelené porty (3. skupina čísiel portov)

Porty v klient-server TCP komunikácii

- Porty identifikujú komunikujúce procesy (zdroj, cieľ)
- Vzájomná komunikácia dvoch konkrétnych procesov je identifikovaná dvojicou portov, ktoré zostávajú počas trvania konverzácie **rovnaké**
  - Mení sa iba poradie portov (zdroj, cieľ) podľa smeru komunikácie klient ↔ server

- Použitie portov v UDP komunikácii je úplne rovnakej filozofie ako pri TCP



Verifikácia otvorených socketov

- Pre zistenie otvorených socketov a spojení na nich je možné použiť v OS Windows i Linux príkaz **`netstat`**
- Nevysvetliteľné TCP spojenia môžu indikovať bezpečnostnú hrozbu.
- Defaultne sa utilita **`netstat`** snaží preložiť IP adresu na doménové meno a číslo portu na „well-known“ aplikáciu.
  - Prepínačom **-n** si možno zobraziť zoznam IP adries a čísiel portov v numerickom.

```
C:\> netstat

Active Connections

Proto   Local Address   Foreign Address           State
TCP     kenpc:3126      192.168.0.2:netbios-ssn   ESTABLISHED
TCP     kenpc:3158      207.138.126.152:http      ESTABLISHED
TCP     kenpc:3159      207.138.126.169:http      ESTABLISHED
TCP     kenpc:3160      207.138.126.169:http      ESTABLISHED
TCP     kenpc:3161      sc.msn.com:http           ESTABLISHED
TCP     kenpc:3166      www.cisco.com:http        ESTABLISHED
```

Parametre príkazu netstat

Príkazový riadok

```
NETSTAT [-a] [-b] [-e] [-f] [-n] [-o] [-p proto] [-r] [-s] [-x] [-t] [interval]

  -a            Displays all connections and listening ports.
  -b            Displays the executable involved in creating each connection or
                listening port. In some cases well-known executables host
                multiple independent components, and in these cases the
                sequence of components involved in creating the connection
                or listening port is displayed. In this case the executable
                name is in [] at the bottom, on top is the component it called,
                and so forth until TCP/IP was reached. Note that this option
                can be time-consuming and will fail unless you have sufficient
                permissions.
  -e            Displays Ethernet statistics. This may be combined with the -s
                option.
  -f            Displays Fully Qualified Domain Names (FQDN) for foreign
                addresses.
  -n            Displays addresses and port numbers in numerical form.
  -o            Displays the owning process ID associated with each connection.
  -p proto      Shows connections for the protocol specified by proto; proto
                may be any of: TCP, UDP, TCPv6, or UDPv6.  If used with the -s
                option to display per-protocol statistics, proto may be any of:
                IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, or UDPv6.
```

Parametre príkazu netstat

Príkazový riadok

```
C:\Users\janau>netstat -h

Displays protocol statistics and current TCP/IP network connections.

NETSTAT [-a] [-b] [-e] [-f] [-n] [-o] [-p proto] [-r] [-s] [-x] [-t] [interval]
```

Príkazový riadok

```
  -q            Displays all connections, listening ports, and bound
                nonlistening TCP ports. Bound nonlistening ports may or may not
                be associated with an active connection.
  -r            Displays the routing table.
  -s            Displays per-protocol statistics.  By default, statistics are
                shown for IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, and UDPv6;
                the -p option may be used to specify a subset of the default.
  -t            Displays the current connection offload state.
  -x            Displays NetworkDirect connections, listeners, and shared
                endpoints.
  -y            Displays the TCP connection template for all connections.
                Cannot be combined with the other options.
  interval      Redisplays selected statistics, pausing interval seconds
                between each display.  Press CTRL+C to stop redisplaying
                statistics.  If omitted, netstat will print the current
                configuration information once.
```

Parametre príkazu netstat

Príkazový riadok

```
C:\Users\janau>netstat -s

IPv4 Statistics
          . . . . .(skrátený výpis) . . . . .
TCP Statistics for IPv4                                3

  Active Opens                            = 16364
  Passive Opens                           = 1041
  Failed Connection Attempts              = 850
  Reset Connections                       = 1147
  Current Connections                     = 20
  Segments Received                       = 3505199
  Segments Sent                           = 940146
  Segments Retransmitted                  = 51410

TCP Statistics for IPv6

  Active Opens          . . . . .(skrátený výpis) . . . . .  = 582
UDP Statistics for IPv4

  Datagrams Received    = 503223
  No Ports              = 12270
  Receive Errors        = 188
  Datagrams Sent        = 343244
```

Časť 9.2: Proces klient-server TCP a UDP komunikácie

Ešte chceme vedieť:

Ako sa vytvorí TCP spojenie a ako sa ukončí?

Ako sa prenášajú a potvrdzujú prenesené TCP segmenty?

Ako sa správa UDP klient, keď chce komunikovať so serverom?

Kedy použiť UDP a kedy TCP?

Aké sú medzi nimi rozdiely?

Téma 9.2.1:
Proces klient-server TCP komunikácie

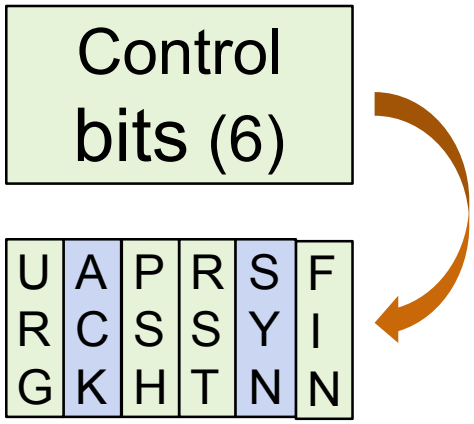
Procesy TCP servera

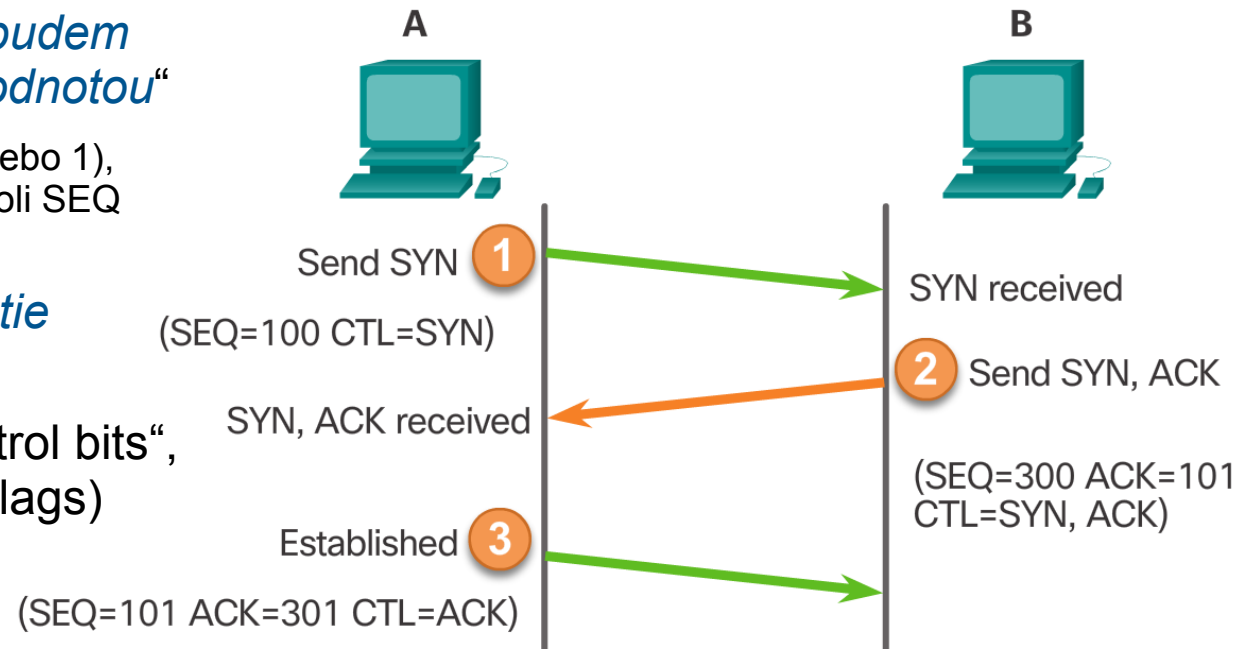
- Každý aplikačný proces bežiaci na serveri používa číslo portu.
- Jeden server nemôže mať dve služby priradené tomu istému číslu portu vrámci jednej služby transportnej vrstvy (TCP, UDP).
- Aktívne aplikácie daného servera priradené špecifickému portu sa považujú za **otvorené** (open).
- Akákoľvek klientská požiadavka adresovaná na otvorený port je akceptovaná a spracovaná aplikáciou servera, ktorá je zviazaná s daným portom.
- Na jednom serveri môže byť viacero otvorených portov, jeden pre každú aktívnu aplikáciu na serveri.

Otvorenie TCP spojenia

- Pred výmenou dát v TCP je nutné zostaviť spojenie
  - Zostavením spojenia sa komunikujúce strany navzájom dohodnú na poradových číslach, počnúc ktorými budú číslovať svoje segmenty
  - Až po tejto sekvencii môže začať prenos užitočných dát
  - Využívajú sa na to 2 príznaky v TCP hlavičke: SYN (synchronization) a ACK (acknowledgement)
    - SYN: „*Svoje segmenty budem číslovať počnúc touto hodnotou*“

      Pozor, toto je len príznak (0 alebo 1), konkrétnu hodnotu uvedie v poli SEQ (sequence number)
    - ACK: „*Potvrdzujem prijatie Tvojho segmentu*“
    - CTL je skratka pre „Control bits“, jedno bitové príznaky (Flags)

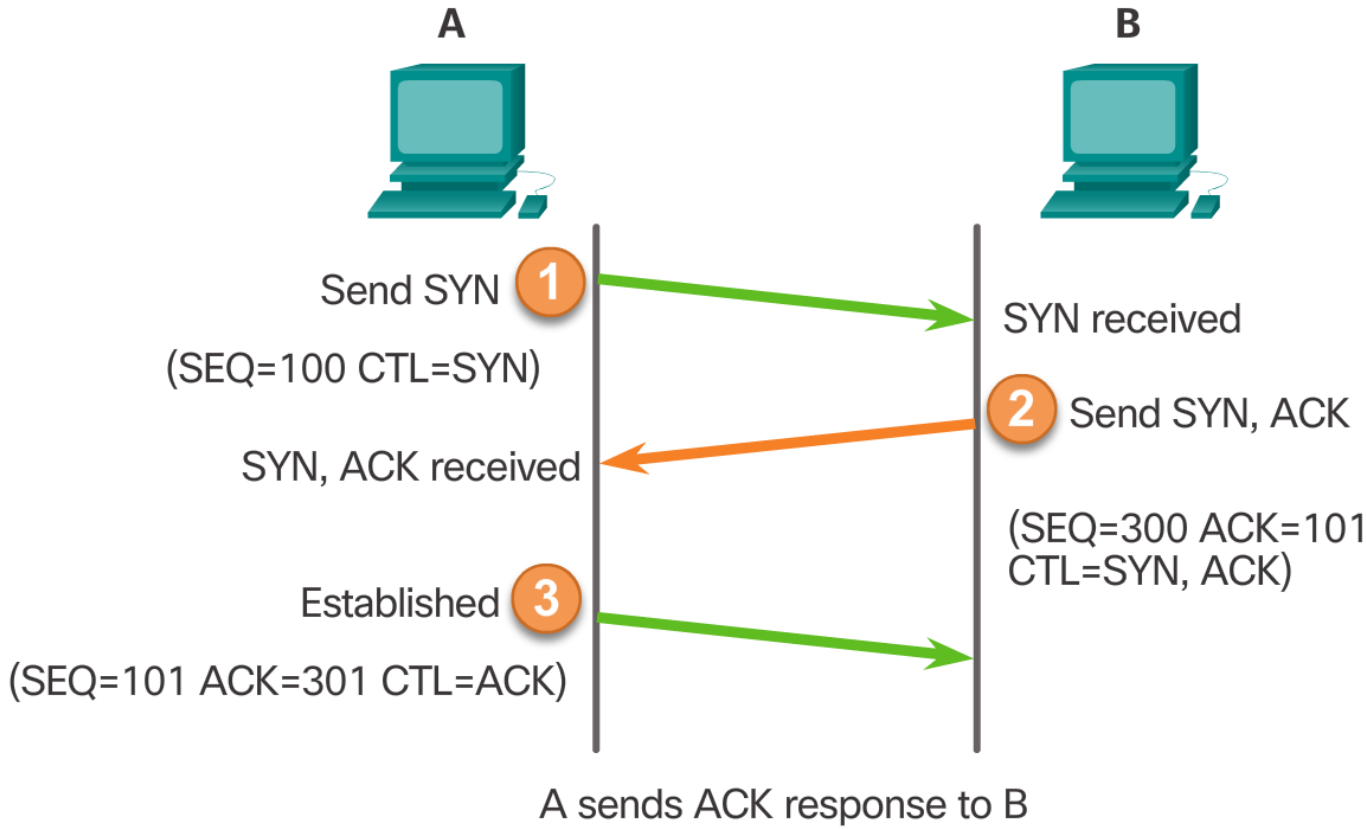




Otvorenie TCP spojenia

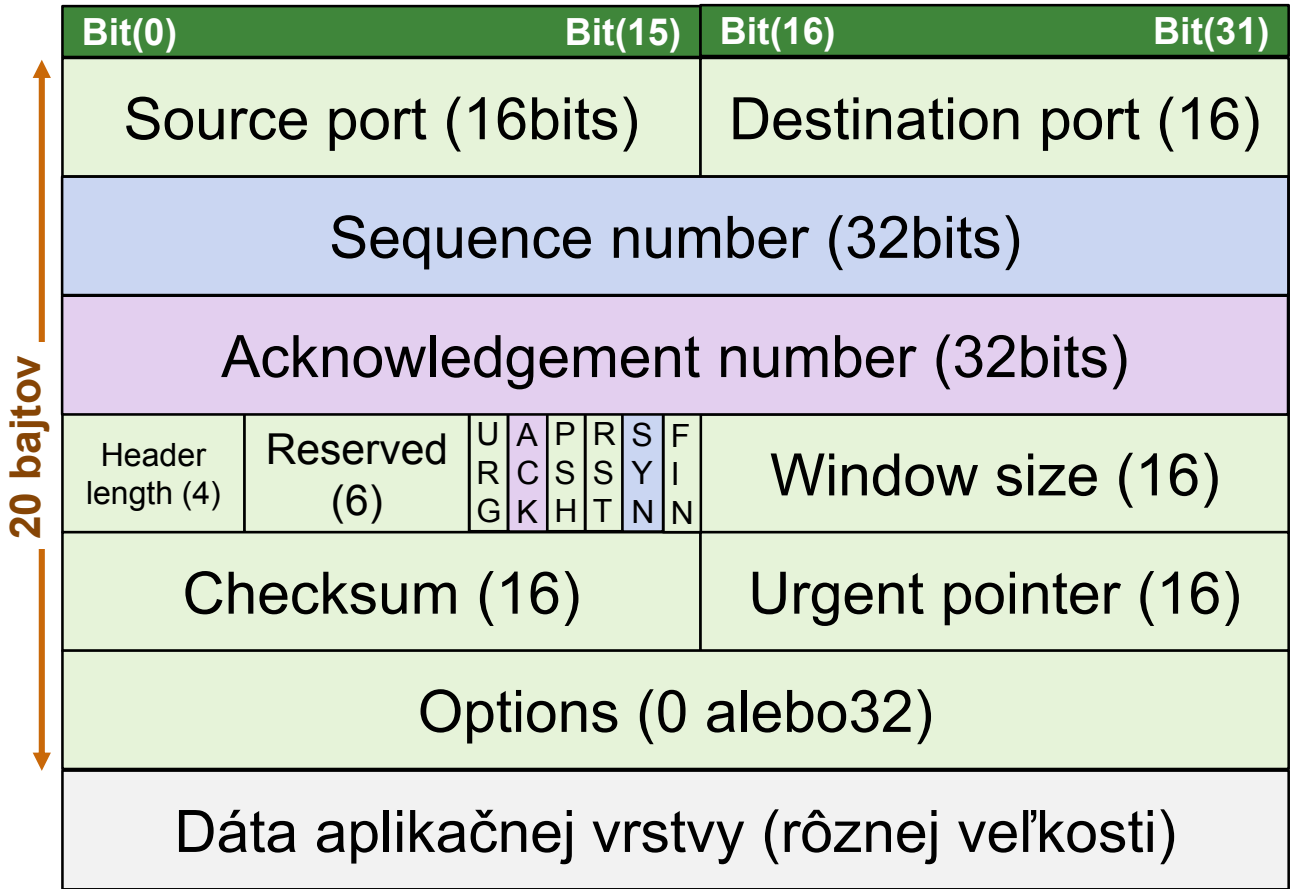
TCP spojenie je vytvorené v 3 krokoch (tzv. **3-way-handshake**):

1. Klient iniciuje vytvorenie spojenia – požiada o client-to-server komunikačnú reláciu (session) so serverom.
2. Server potvrdí túto reláciu a požiada klienta o server-to-client kom. reláciu.
3. Klient potvrdí server-to-client komunikačnú reláciu.

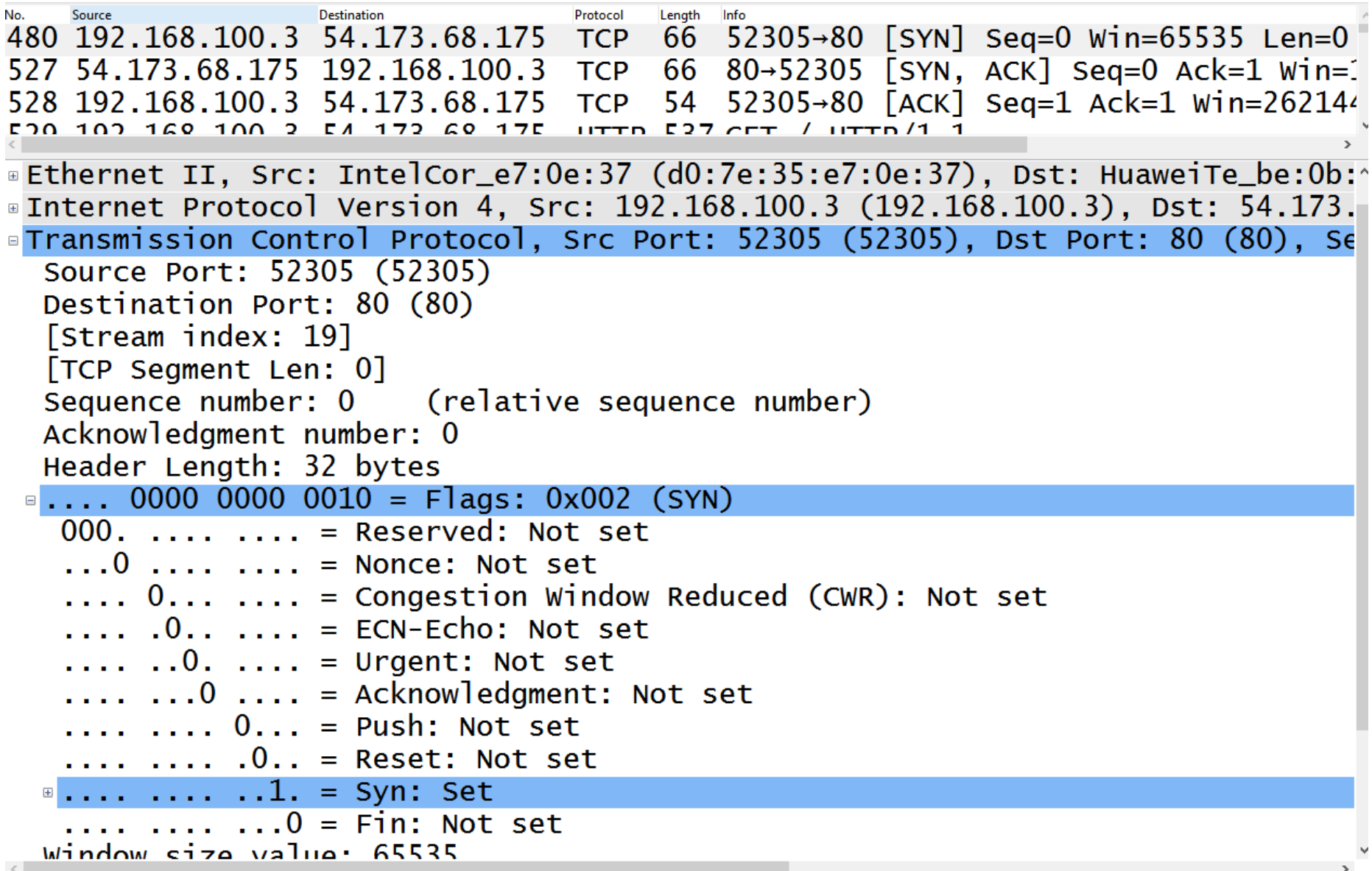


Analýza TCP 3-Way Handshake

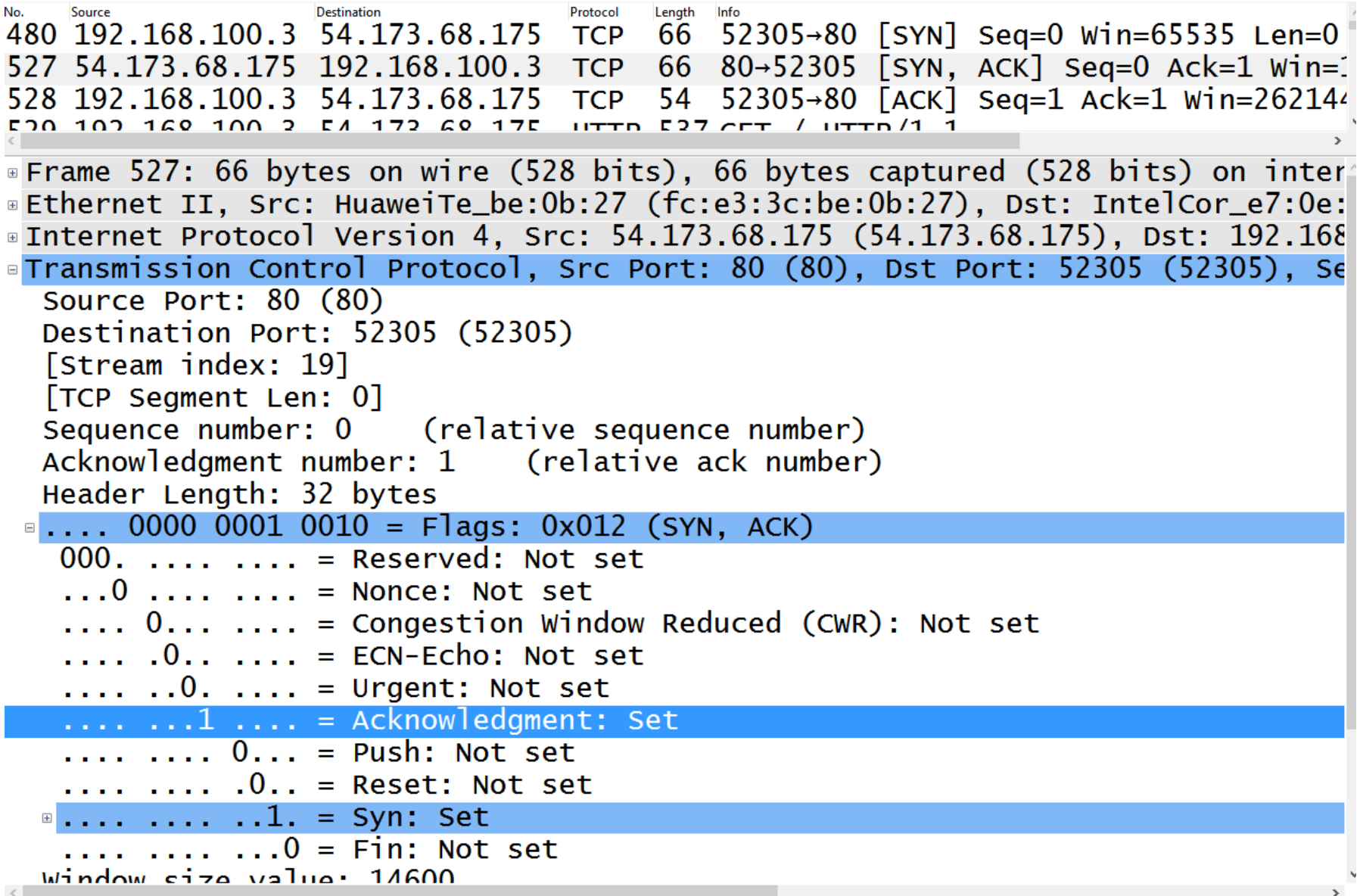
- Potvrdí, že cieľové zariadenie je v sieti prítomné.
- Overí, že cieľové zariadenie má aktívnu službu a prijíma požiadavky na danom cieľovom porte, ktorý chce daný klient použiť.
- Informuje cieľové zariadenie, že zdroj (klient) chce vytvoriť komunikačnú reláciu na danom porte.
- Príznaky:
  - SYN – v tomto segmente ti oznamujem, od akej hodnoty budem číslovať svoje segmenty
  - ACK – v tomto segmente ti posielam aj potvrdenie (čo mi už od teba úspešne prišlo)



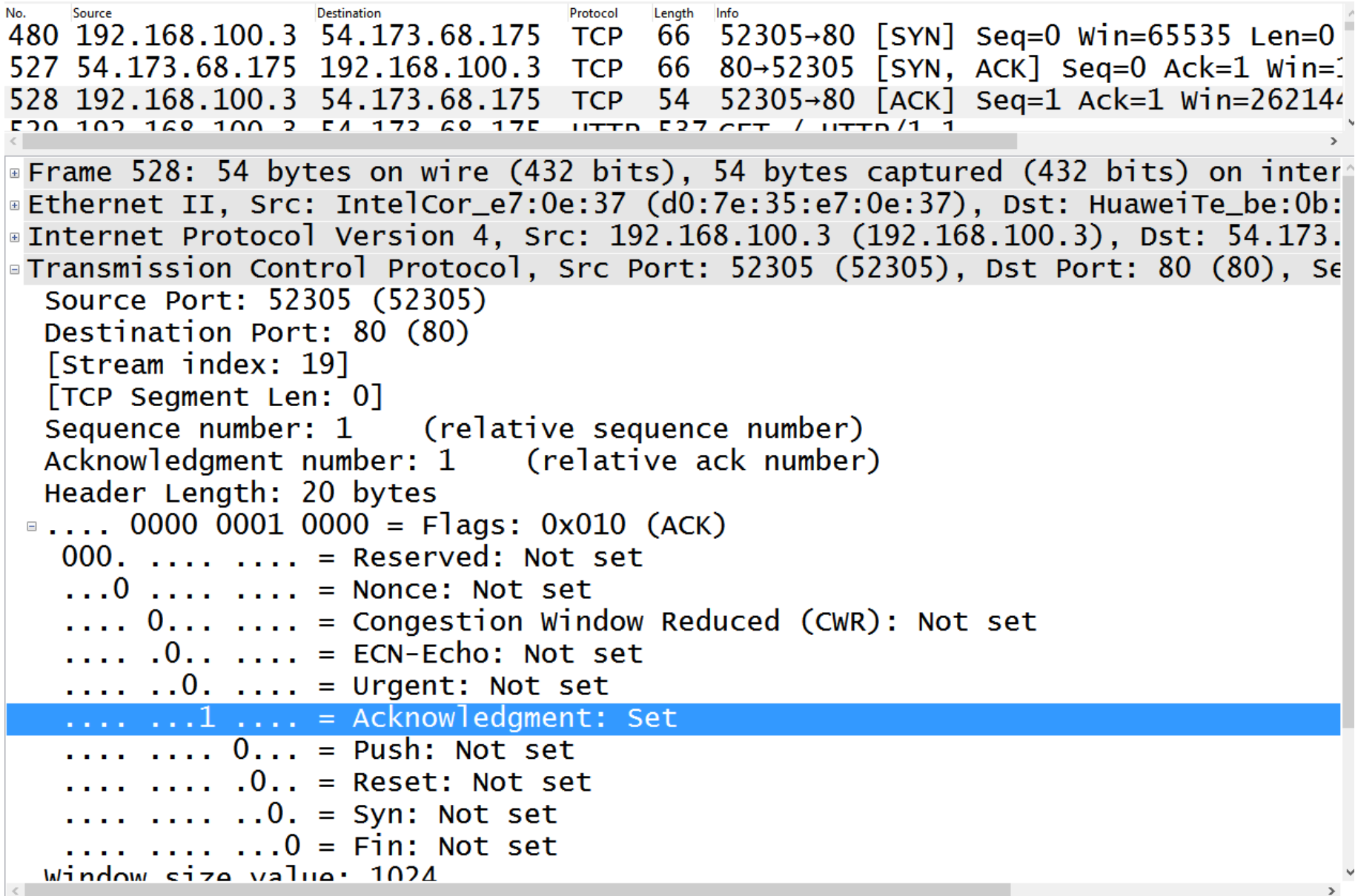
Wireshark: 3-way handshake [SYN]



Wireshark: 3-way handshake [SYN, ACK]



Wireshark: 3-way handshake [ACK]

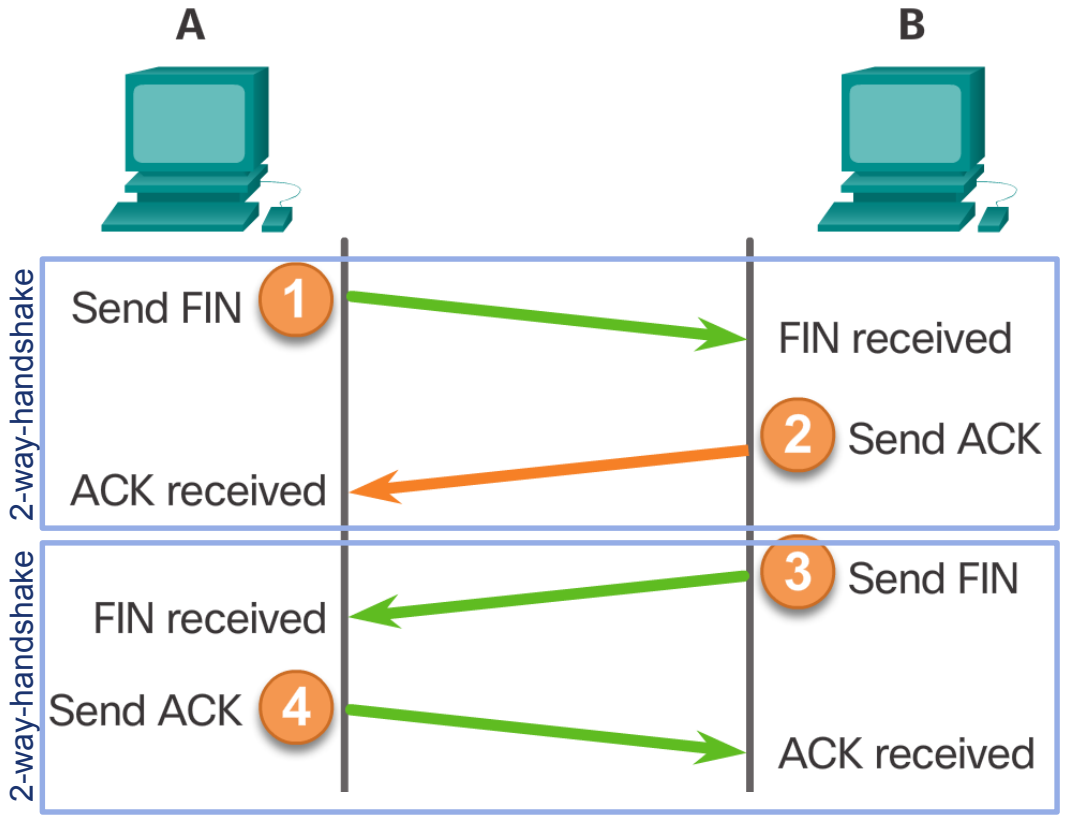


Uzatvorenie TCP spojenia

- Po konci komunikácie je potrebné TCP spojenie uzatvoriť
- Tzv. 2x 2-way-handshake
- Využíva sa príznak FIN (finish)
  - FIN: „*Nemám viac dát na odoslanie, za seba môžem skončit*“

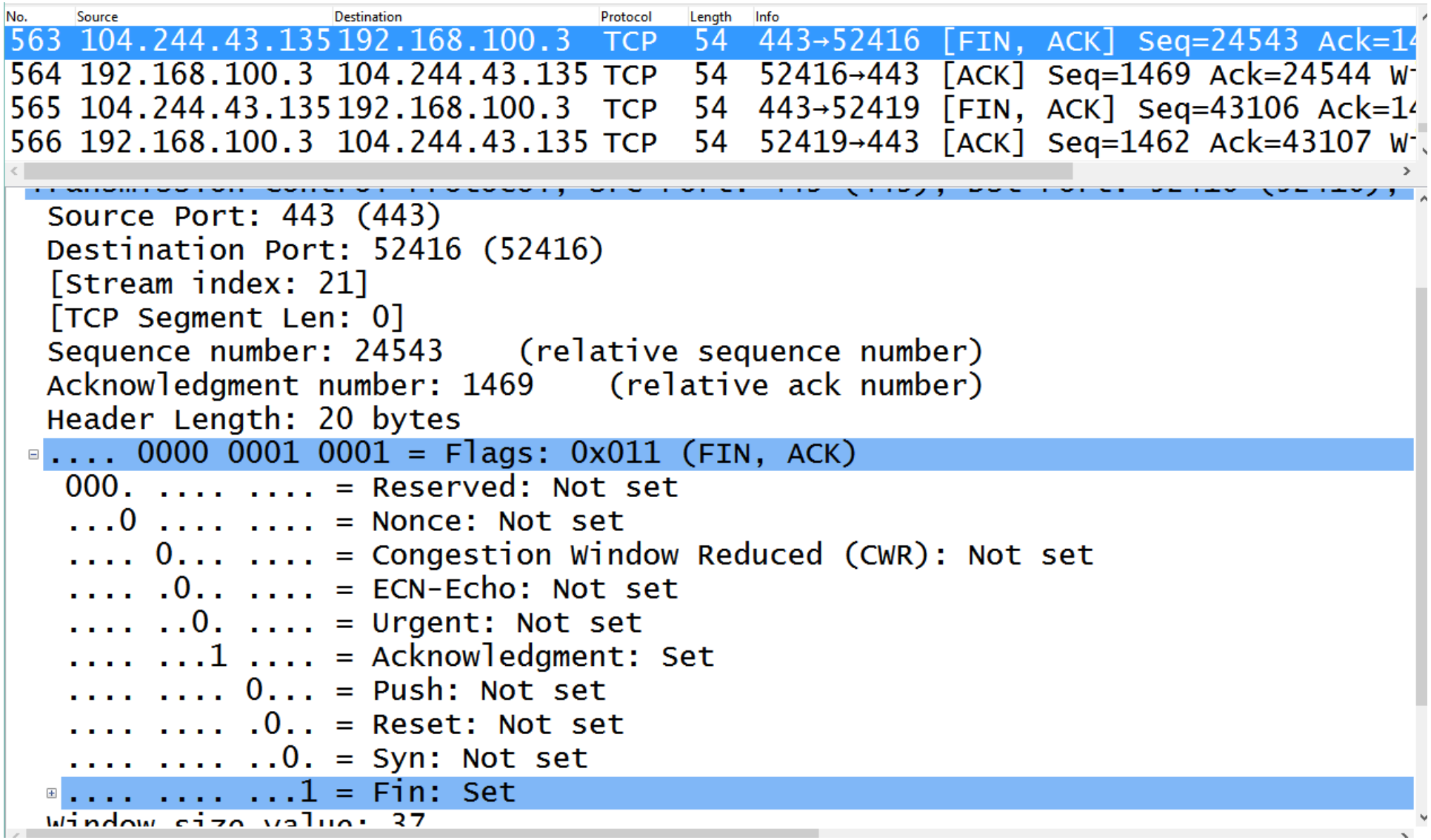
1. Keď klient poslal z daného TCP toku už všetky segmenty, pošle segment s nastaveným príznakom FIN.
2. Server pošle ACK, čím potvrdí prijatie FIN, že klient žiada o ukončenie danej relácie client-to-server.
3. Server pošle klientovi FIN, že súhlasí a ukončuje reláciu server-to-client.
4. Klient odpovie segmentom s ACK, čím potvrdí prijatie FIN od servera.

Po skončení týchto krokov je relácia **zatvorená**.



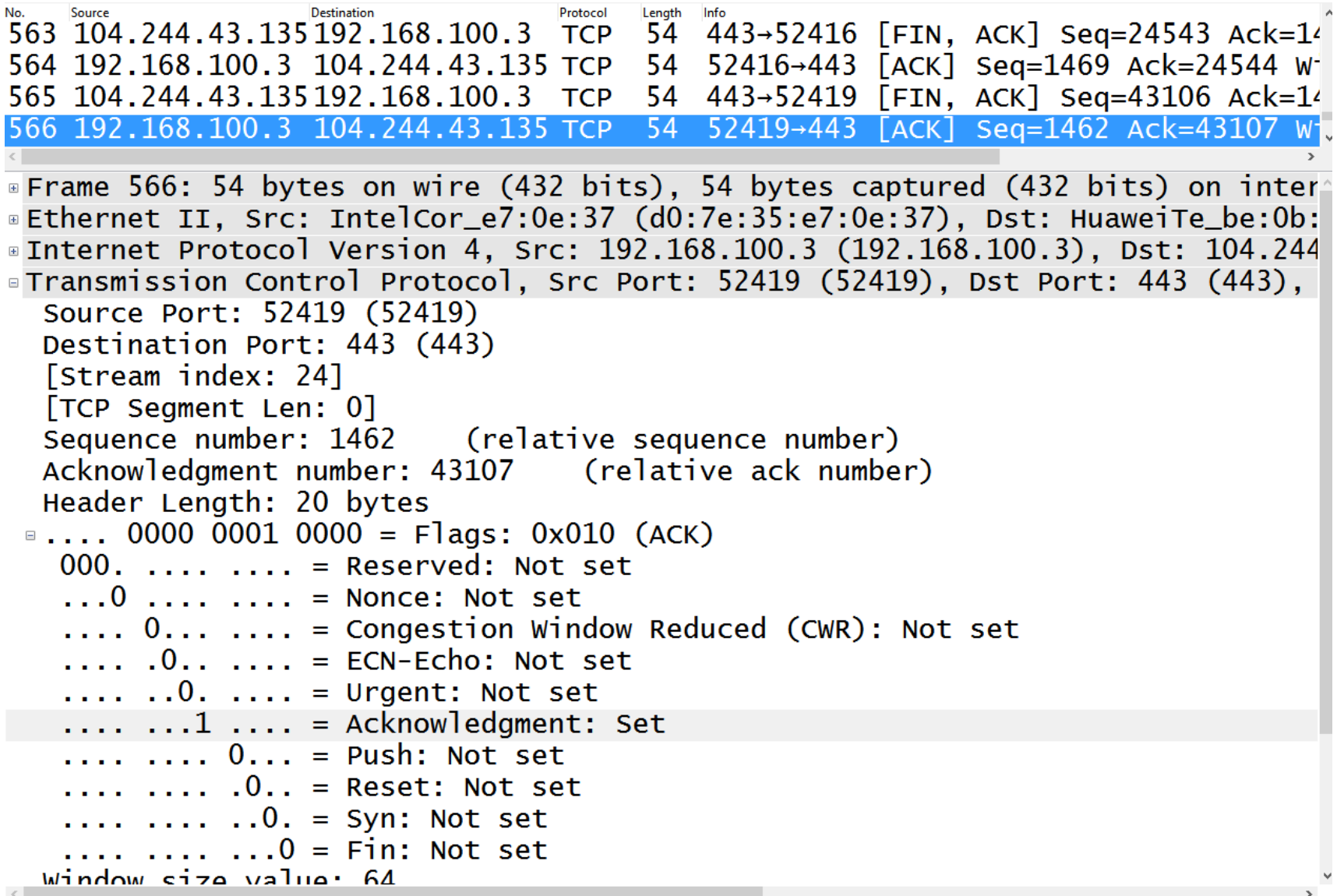
Wireshark: uzatvorenie spojenia [FIN, ACK]

Prvý 2-way-handshake



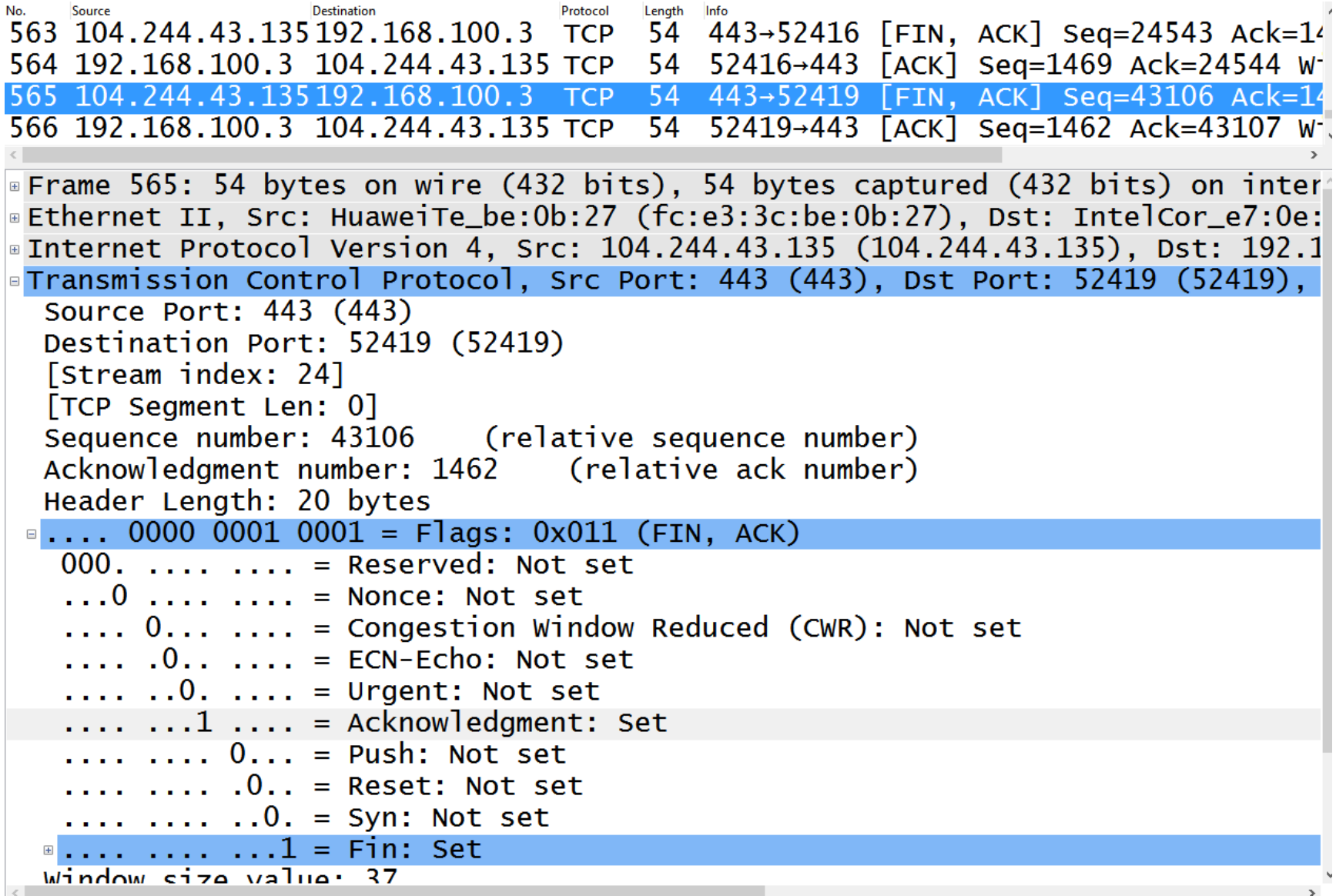
Wireshark: uzatvorenie spojenia [ACK]

Prvý 2-way-handshake



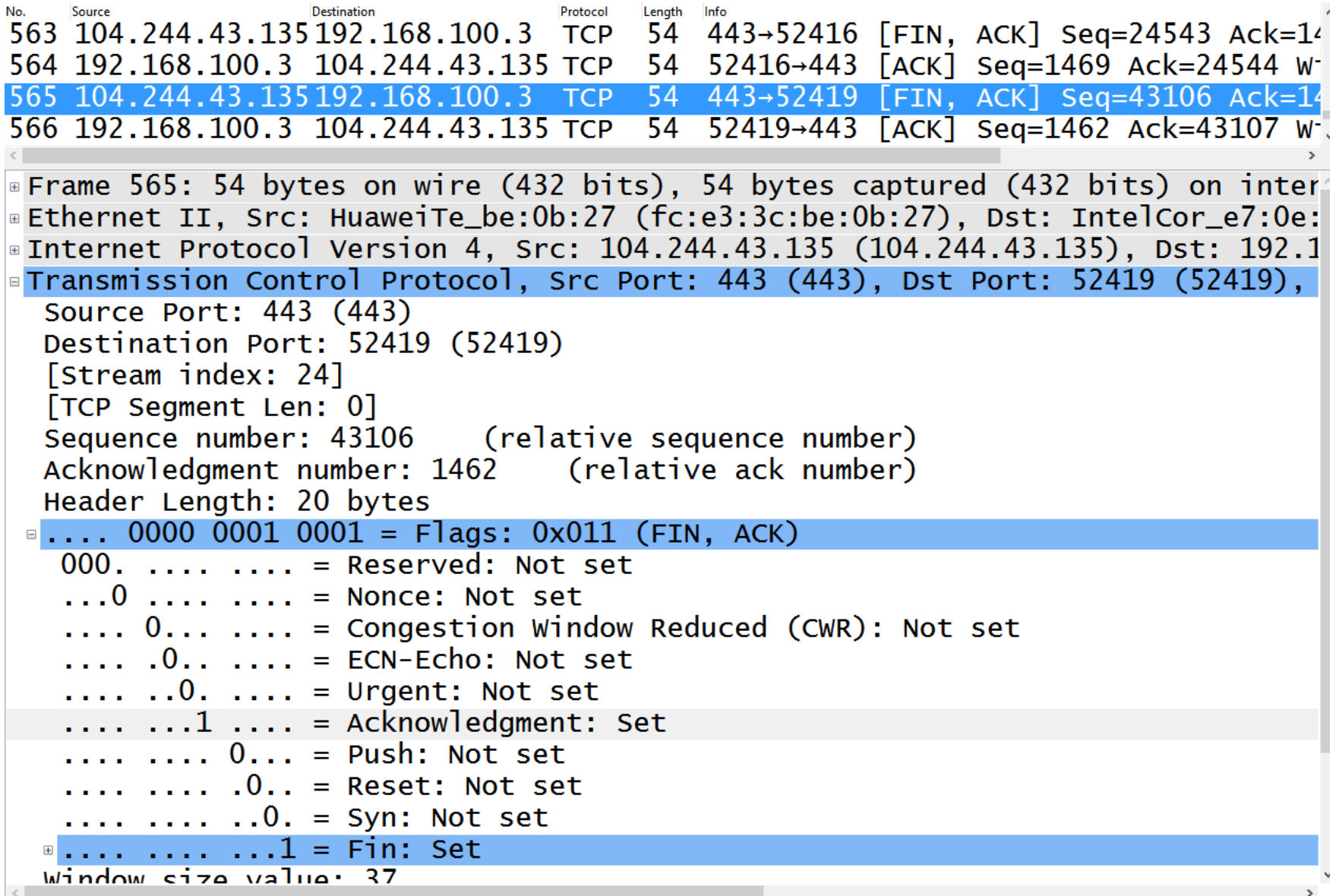
Wireshark: uzatvorenie spojenia [FIN, ACK]

Druhý 2-way-handshake

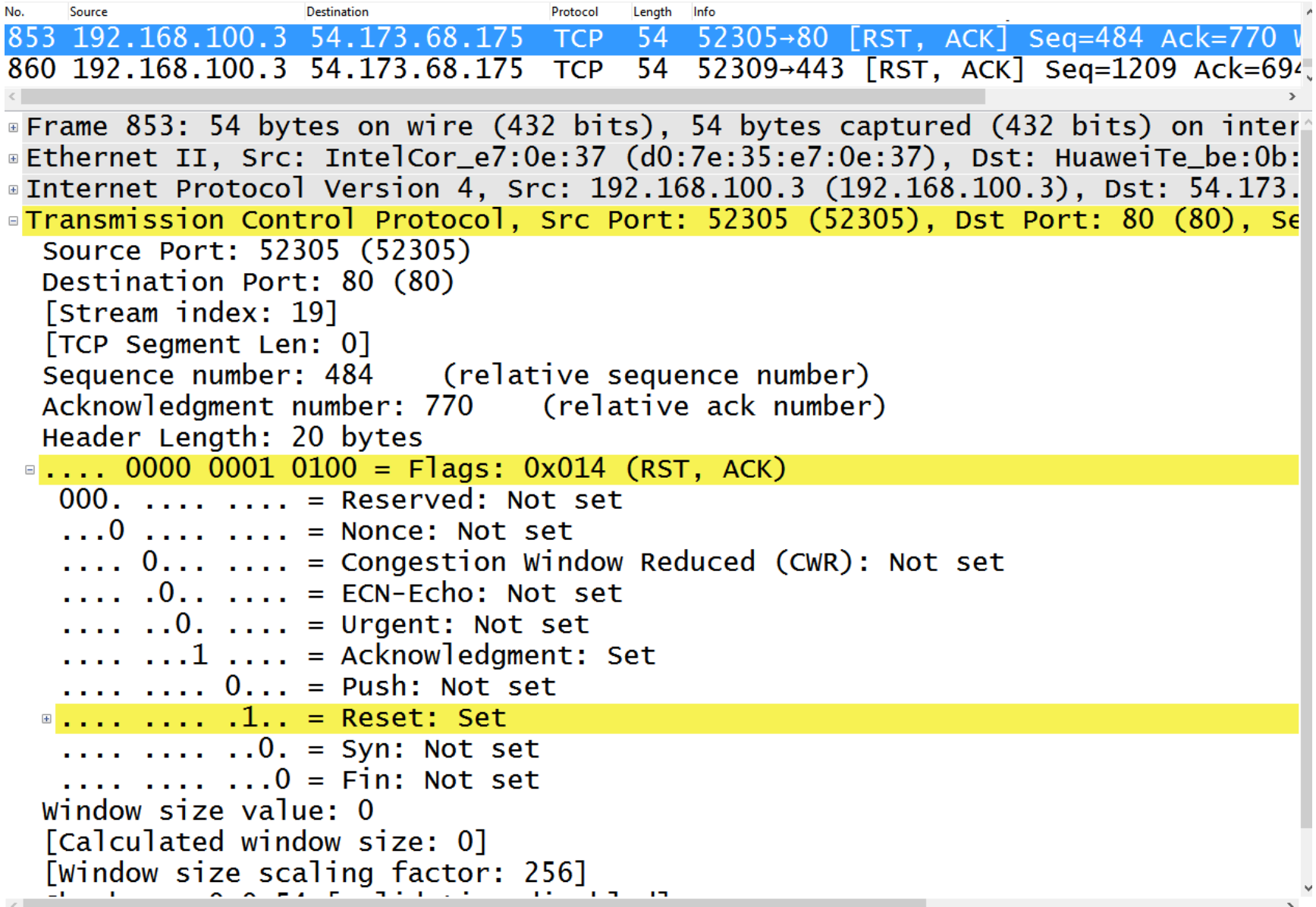


Wireshark: uzatvorenie spojenia [ACK]

Druhý 2-way-handshake

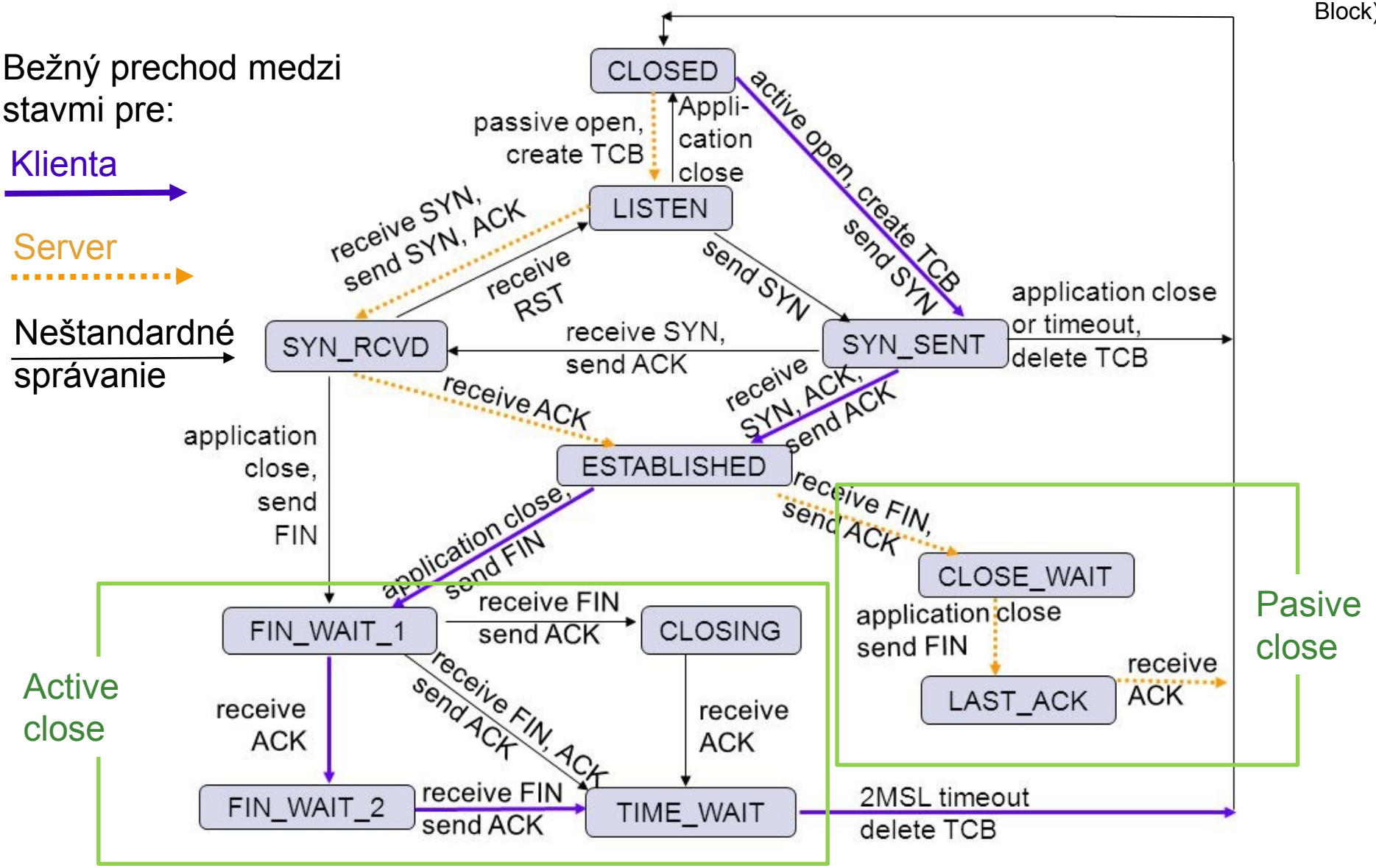


Wireshark: náhle ukončenie spojenia [RST]



TCP stavový diagram prechodov
Len pre skalných (nebude na skúške)

(skratka TCB = Transmission Control Block)



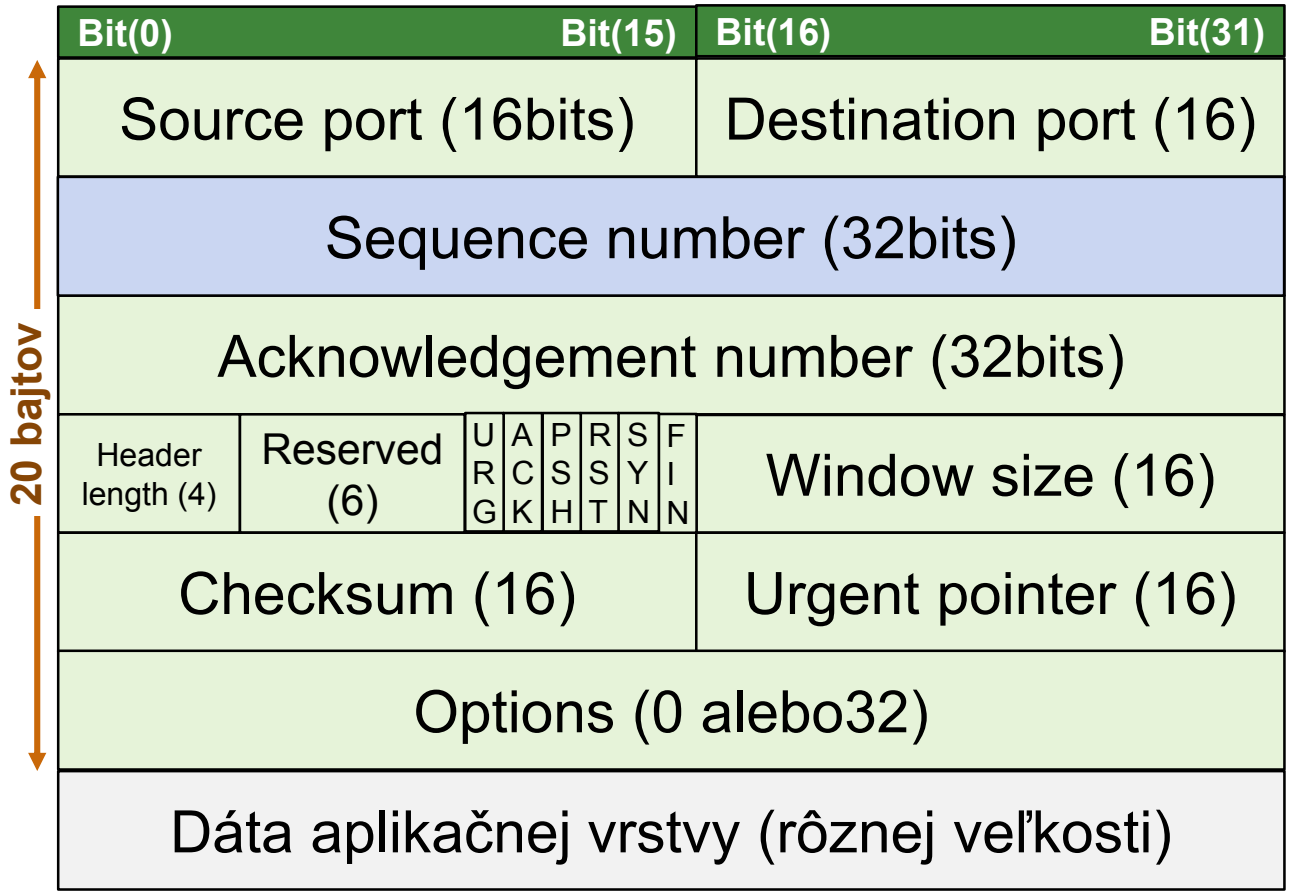
Téma 9.2.2:
Spoľahlivosť a kontrola toku dát v TCP

Spoľahlivost TCP – Usporiadanie prijatých dát

- TCP segmenty používajú **sekvenčné čísla** (sequence numbers - SN):
  - na jednoznačné identifikovanie a potvrdzovanie každého segmentu

„*Strata každého segmentu je odhalená.*“

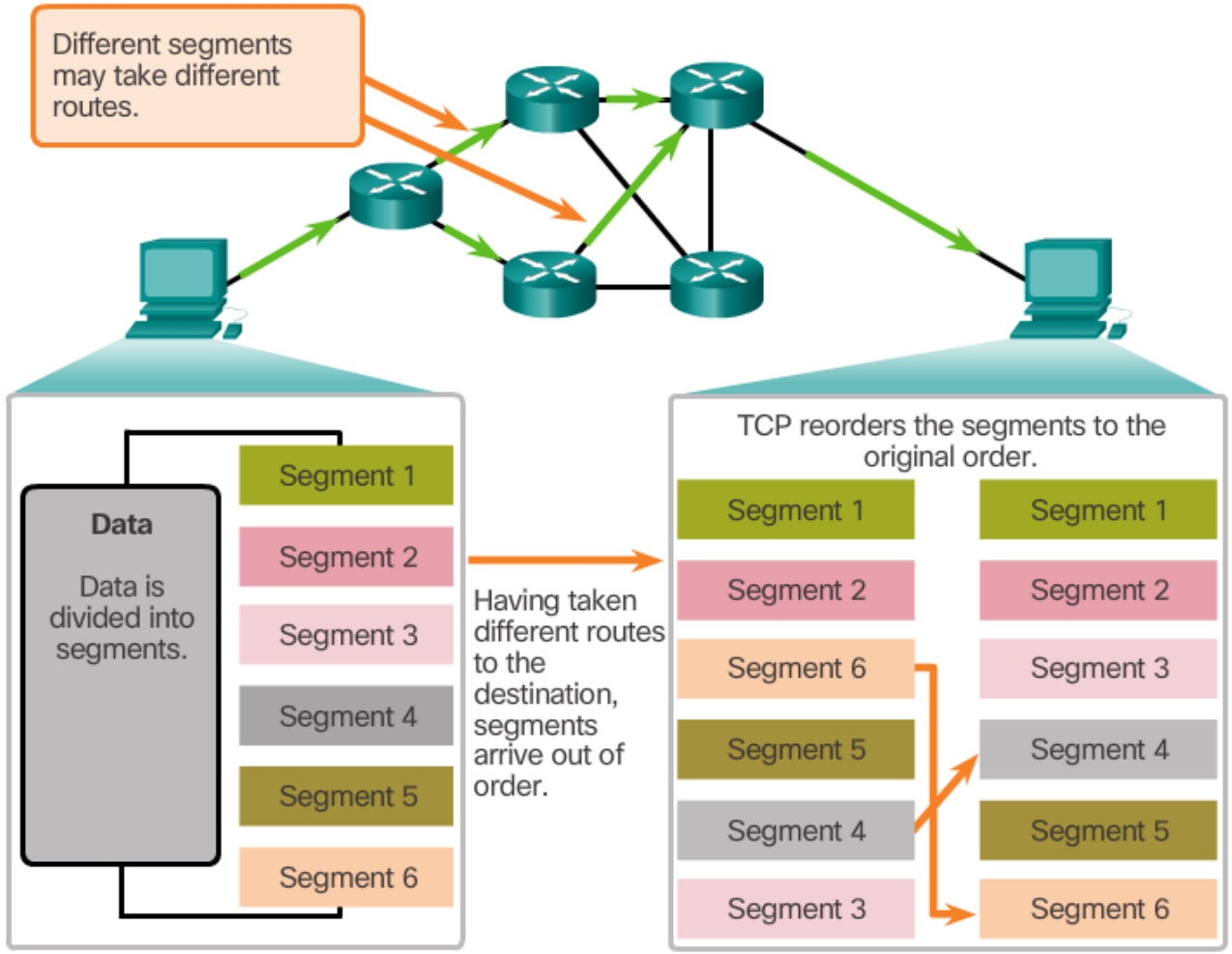
  - aby si udržiavali správne poradie segmentov.
  - aby druhej strane dali vedieť v akom poradí má zrekonš--truovať jednotlivé segmenty do celej správy



- Úvodné/prvé SN sa volí **náhodne** (v minulosti sa brala 0) počas vytvorenia spojenia (**3-way-handshake**), a **inkrementuje** sa vždy **o počet odoslaných bajtov**

Spoľahlivost TCP – Usporiadanie prijatých dát

- TCP proces u príjemcu si ukladá prijaté segmenty do frontu - buffer, segmenty ktoré prídu mimo poradia sú odložené na neskoršie spracovanie.
- Až keď príjemca dostane všetky segmenty (slušne sa ukončí spojenie), začne robiť rekonštrukciu prijatých segmentov do pôvodnej správy.
- Ak sa rekonštrukcia podarí, dáta predá na spracovanie aplikačnej vrstve.



Spoľahlivosť TCP – Potvrdzovanie prijatých dát

- TCP proces u príjemcu **potvrdzuje** každý segment, ktorý prijme od TCP procesu odosielateľa. Táto metóda sa nazýva **Stop and wait**:

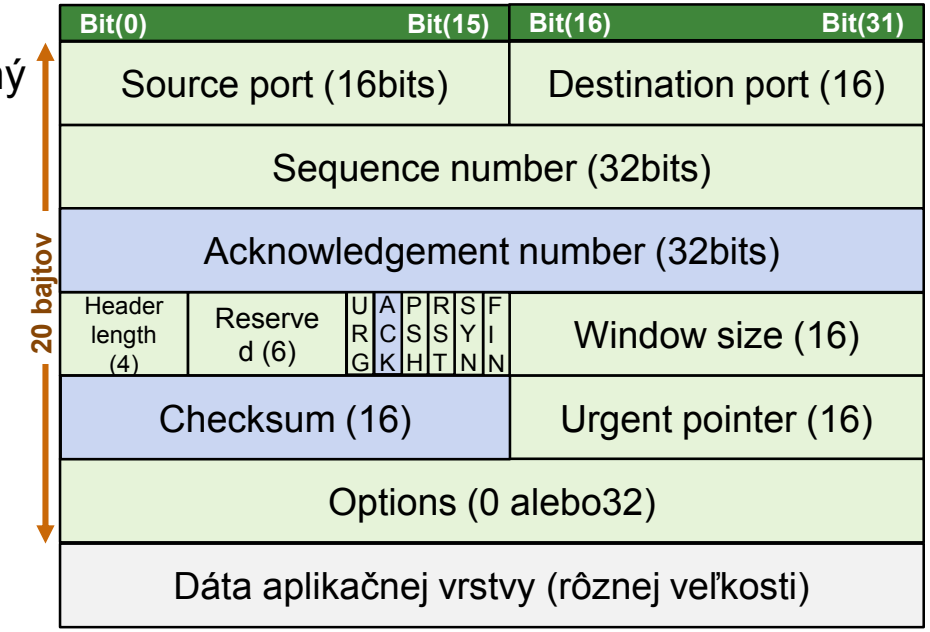
  „*Pošli segment a zastav posielanie ďalších segmentov (Stop), čakaj na potvrdenie - ACK (Wait).*“

  - Používa sa na to pole **Acknowledgement Number** v hlavičke TCP

- Aby sa odosielateľ nedostal do slepej uličky (deadlock), kedy by čakal na ACK do nekonečna:

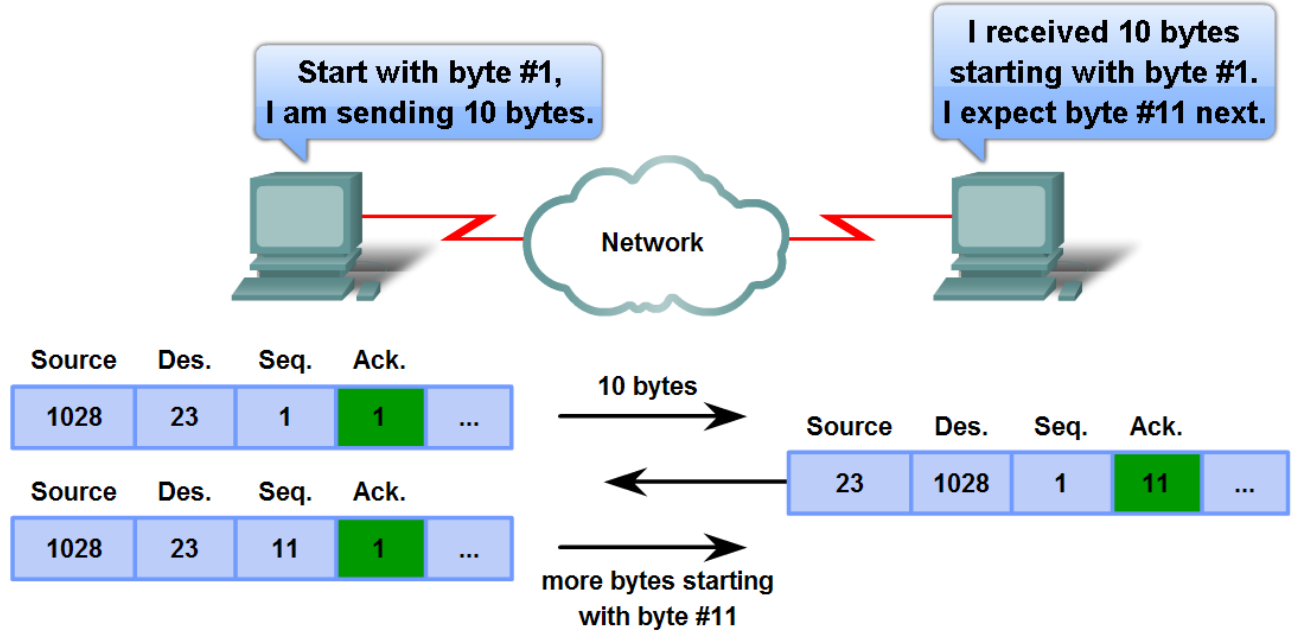
  1. pri strate segmentu
  2. pri strate potvrdenia
  3. keď príjemcovi segment príde, ale poškodený (zistí.. **checksum**), a neposiela späť ACK

- tak sa používa časovač, tzv. **Retransmission Timeout (RTO)**:
  - spúšťa sa pri odoslaní každého segmentu
  - keď vyprší a nepríde ACK, odosielateľ **zopakuje odoslanie** toho istého segmentu
  - ak do tohto času príde ACK, odosielateľ vyšle ďalší segment, pre ktorý znovu spúšťa RTO



Spoľahlivosť TCP – Potvrdzovanie prijatých dát

- Potvrdzovanie je tzv. pozitívne alebo dopredné: ak 1 strana pošle potvrdzovacie číslo ***n***, znamená to, že správne prijala všetky **bajty** až po ***n-1***
  - Potvrdzovacie číslo ***n*** teda znamená:
    „*Pokračuj bajtom n, pretože všetky bajty od počiatku až po n-1 už mám*“
  - Potvrdenie hovorí o prvom bajte, ktorý očakávame (resp. ktorý chýba)
- Potvrdzovanie a prenos dát sa môže diať v jednom TCP segmente súčasne – tzv. **piggybacking** – ACK správy nepošlem samostatne, ale vložím do segmentu, ktorý mám pripravený na odoslanie druhej strane, tzv. nesamostatné potvrdzovanie



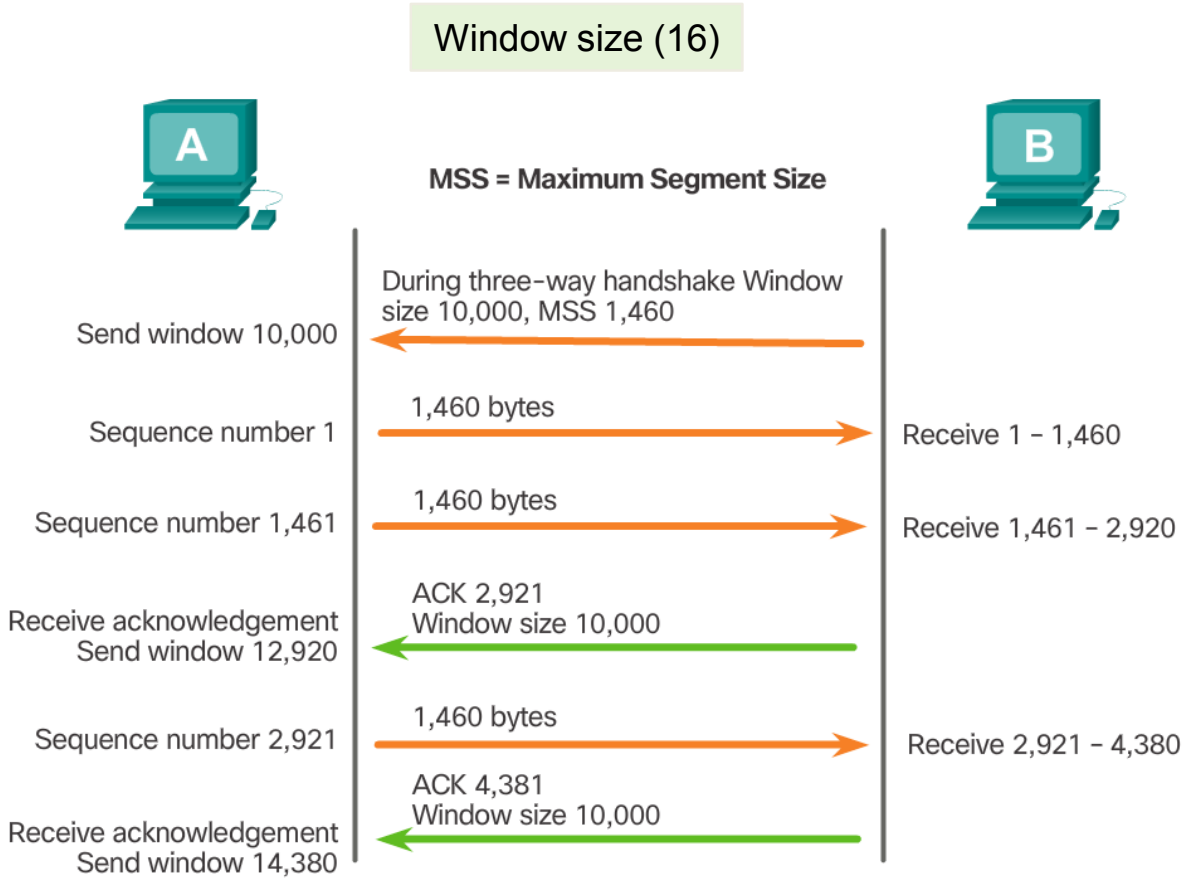
Riadenie toku dát – technika posuvného okna

- Zhodnotenie Stop&Wait:
  - výhoda: je jednoduché implementovať
  - nevýhoda: nevyužíva dostatočne prenosové pásmo, vzniká oneskorenie – za čas, kedy sa čaká na potvrdenie, by sa mohol odoslať ďalší segment
- Obmedzenia Stop&Wait rieši mechanizmus pre riadenie toku tzv. **sliding window** (plávajúce/posuvné okno), ktorého cieľom je, aby sa dáta vysielali čo najrýchlejšie, ale tak, aby oba konce TCP komunikácie stíhali prijímať a spracovávať segmenty spoľahlivo
- Používa na to pole **Window size (WS)** v hlavičke TCP
  - 16 bitové pole (2^{16} možných veľkostí okna)
  - min. WS = 0 B, max. WS = 2^{16} -1 = 65 535 B
  - Je to maximálny objem dát (v bajtoch), ktoré mi môže druhá strana poslať ešte pred tým, ako jej doručím potvrdenie o ich prijatí (t.j. nemusí čakať na potvrdenie)
    - potvrdenia prísť nakoniec musia, ale odosieľateľ na ne nemusí čakať, kým súčet bajtov zo všetkých odvysielaných segmentov nepresiahne veľkosť okna
  - Je to počet bajtov, ktoré som ako príjemca schopný prijať, preto veľkosť môjho okna musím oznámiť druhej strane, aby sa mi vedela prispôsobiť

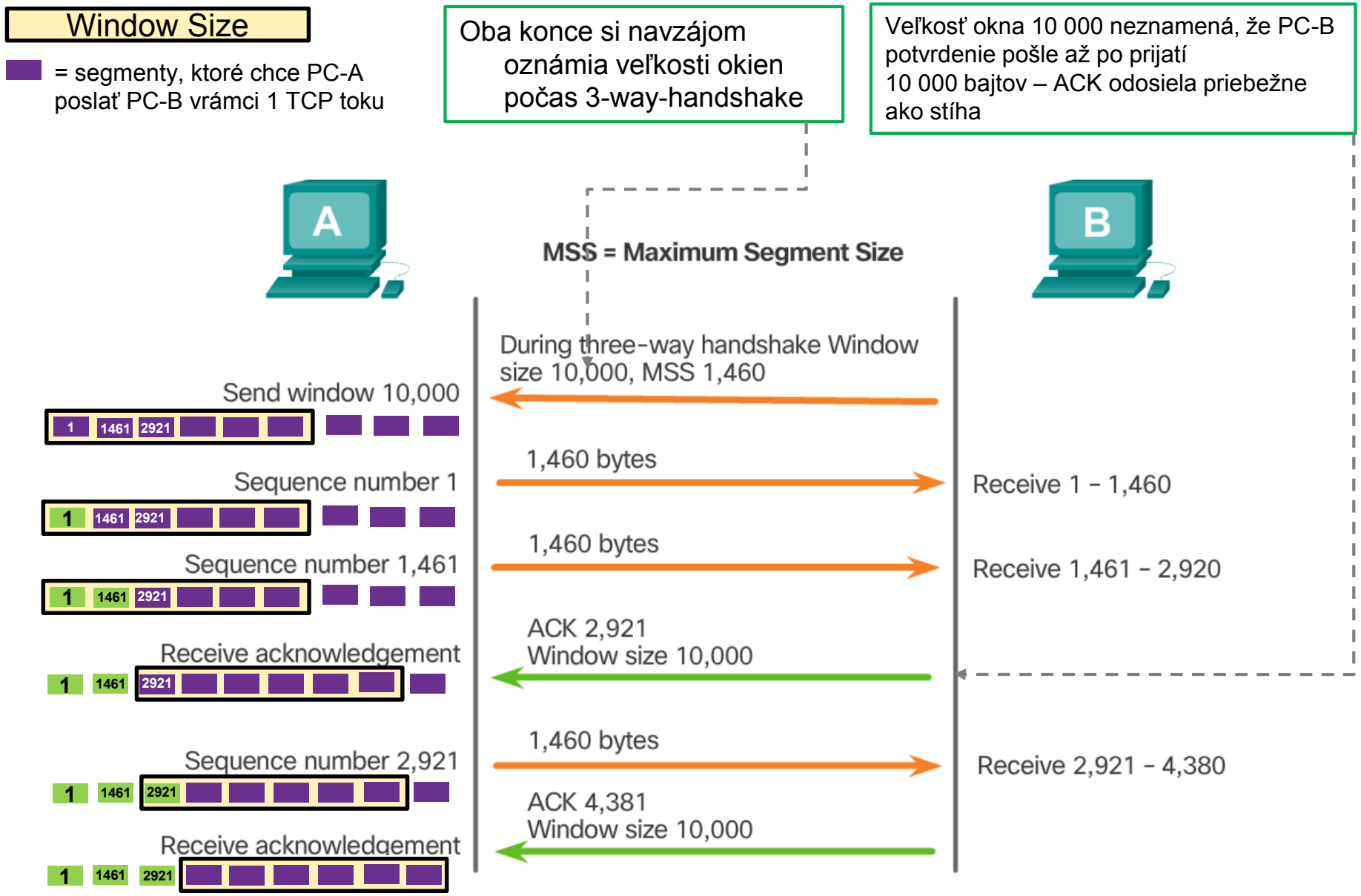
Window size (16)

Riadenie toku dát – technika posuvného okna

- Oba konce si navzájom oznámia veľkosti okien
  počas 3-way-handshake
- Po vytvorení spojenia môžu veľkosť tohto okna meniť - veľkosť okna stanica uvádza v **každom** odoslanom TCP segmente v poli **Window Size**
- Veľkosti okna **N** neznamená, že potvrdenie pošleme až po prijatí
  N bajtov – odosielanie potvrdení nemusí byť (a zväčša nie je)
  veľkosťou okna ovplyvnené

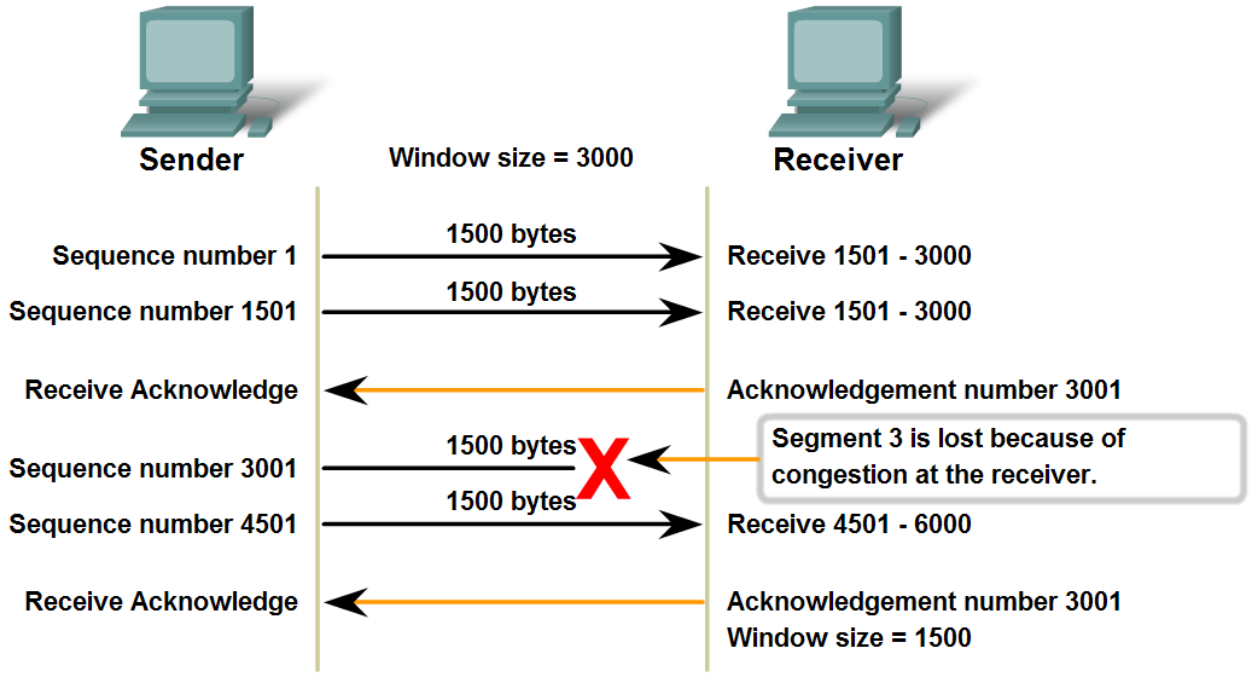


Riadenie toku dát – technika posuvného okna



Riadenie toku dát – technika posuvného okna

- Pri strate segmentu so SN=3001, príjemca potvrdí znova posledný prijatý bajt – aby odosielateľ vedel, čo posledné mu prišlo a čo ďalšie očakáva
  - T.j. nepotvrdzuje segment so SN=4501, aj keď mu tento prišiel
  - Následne odosielateľ bude musieť zopakovať prenos segmentov SN=3001, SN=4501
- Navyše odosielateľ si **zmenší** veľkosť okna WS=3000, reaguje na stratené segmenty (alebo stratené ACK, alebo poškodené segmenty), v tomto príklade na ½, t.j. WS=1500
- Keď sa situácia zlepší a odosielateľ začne dostávať ACK, zase si okno **zväčší**



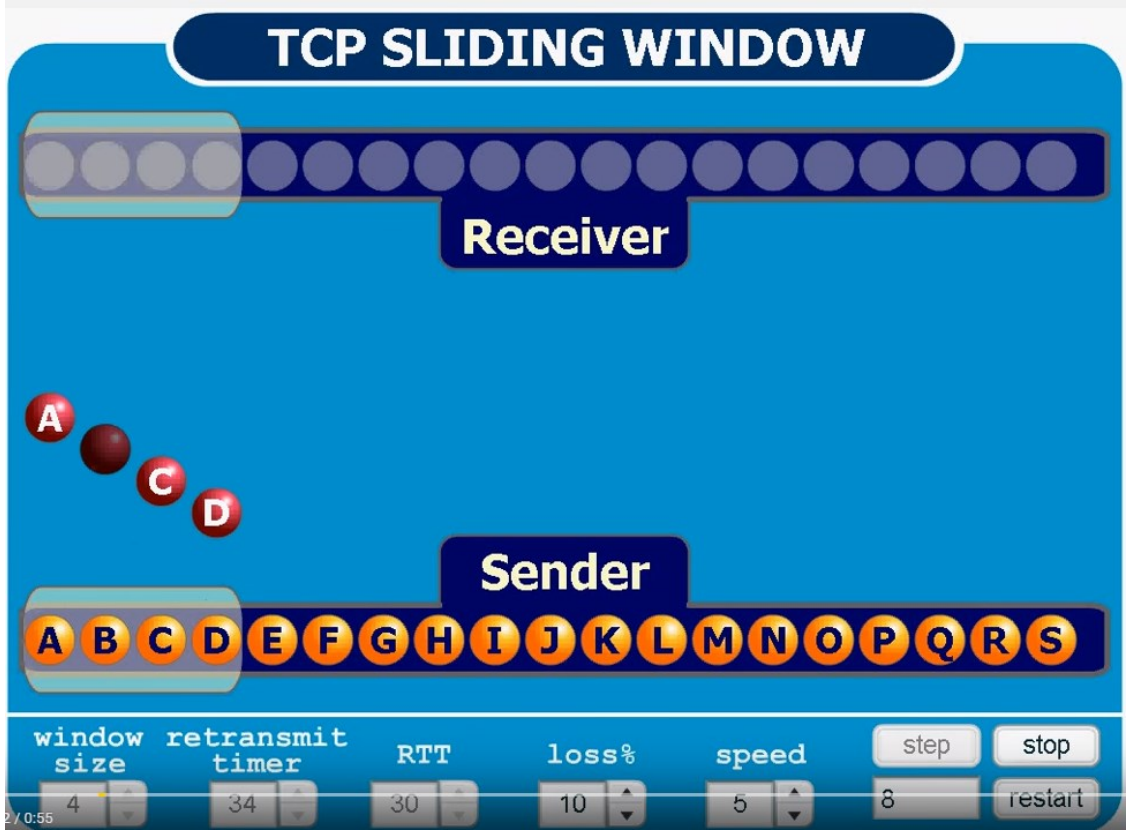
Riadenie toku dát – technika posuvného okna

- Prijatím potvrdenia sa celé okno **posúva** smerom na **neodoslané data**, odtiaľ názov plávajúce/posuvné okno (sliding window)
- Animácia ako pracuje technika plávajúceho okna:
  - WS = 4, RTO = 34, RTT = 30, straty = 10%, rýchlosť animácie=5
    - https://www.youtube.com/watch?v=lk27yiITOvU
  - WS = 4, RTO = 34, RTT = 30, straty = 10%, rýchlosť animácie=5
    - https://www.youtube.com/watch?v=zY2pwPF6pl8
  - RTT = Round Trip Time - čas za ktorý sa správa šíri od zdroja do cieľa + za ktorý príde naspäť potvrdenie.
  - Priestor na vaše experimenty:

    http://www.exa.unicen.edu.ar/catedras/comdat1/material/Filminas3_Practico3.swf

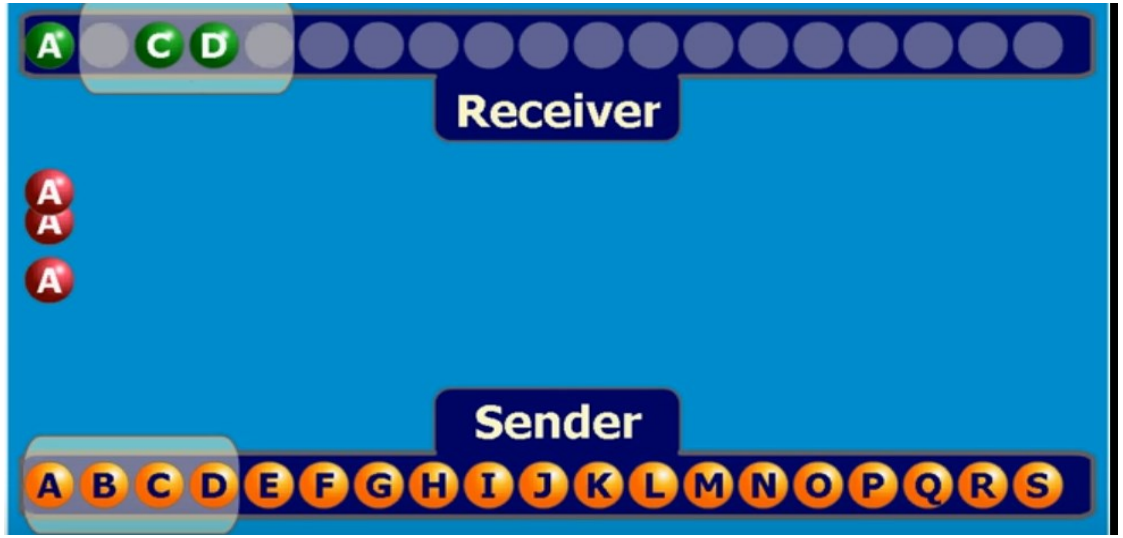
    (Potrebný je nainštalovaný Adobe Flash Player)

    Aký problém vidíte pri nenulových stratách a týchto nastaveniach:
    a. RTT = 40, RTO = 20 ?
    b. RTT = 20, RTO = 40 ?



Riadenie toku dát – technika posuvného okna

- Efektívnejšie potvrdzovanie:
  - V situácií, keď sa príjemcovi nedoručí jeden alebo niekoľko segmentov, ale ostatné áno (zistí podľa čísiel SN):
    - namiesto toho, aby príjemca čakal, že odosielateľovi vyprší RTO a prepošle mu chýbajúci segment aj všetky za ním nasledujúce (vždy sa preposiela celé okno)..
    - ...upozorní odosielateľa na chýbajúci rámec, a znova mu pošle ACK na posledný prijatý bajt dát, a aby na odosielateľa poriadne zakričal:
      „*Toto od teba očakávam, prepošli znova!!!*“
      tak pošle takéto **ACK 3 po sebe** (3x ACK pre A)



Riadenie toku dát – predchádzanie zahlteniu

Congestion Avoidance

- **Zahltenie** v sieti (congestion) zvyčajne vedie k **stratám** paketov
- Nedoručené TCP segmenty sa musia **preposielať znova**, čo môže situáciu ešte viac zhoršiť.
- Odosielateľ môže **detegovať** zahltenie v sieti tým, že **rýchlosť** akou vysiela dáta, neodpovedá rýchlosti akou **mu chodia ACK**, alebo mu vôbec nechodia.
- Vtedy odosielateľ môže **zmenšiť rýchlosť odosielania dát** – zníži si veľkosť okna ešte pred tým, ako mu zníženie oznámi príjemca, ktorý takéto zahltenie zväčša zistí až neskôr, alebo ho vôbec nezaregistruje.
- Keď sa situácia zase zlepší, a odosielateľovi začnú chodiť ACK, znova si zväčší okno.. zväčšuje ho postupne
- Takto sa okno dynamicky „otvára“ a „zatvára“
- O ktorom okne je reč? ....

Riadenie toku dát – predchádzanie zahlteniu

Congestion Avoidance

- V skutočnosti má odosielateľ svoje vlastné okno,
  Sender Window, tiež nazývané ako
  Congestion Window, ktoré si prispôsobuje podľa situácie v sieti (či mu nechýbajú ACK alebo dostáva chybné segmenty)
  - Okno, ktoré mu ohlasuje príjemca v TCP hlavičke v poli **Window Size,** potom môžeme nazvať aj ako **Receiver Window** - to čo príjemca ohlasuje odosielateľovi:
    „*Neposielaj mi dáta rýchlejšie ako je Window size*.“
  - Rýchlosť akou odosielateľ nakoniec skutočne vysiela dáta sa potom berie ako minimum z:

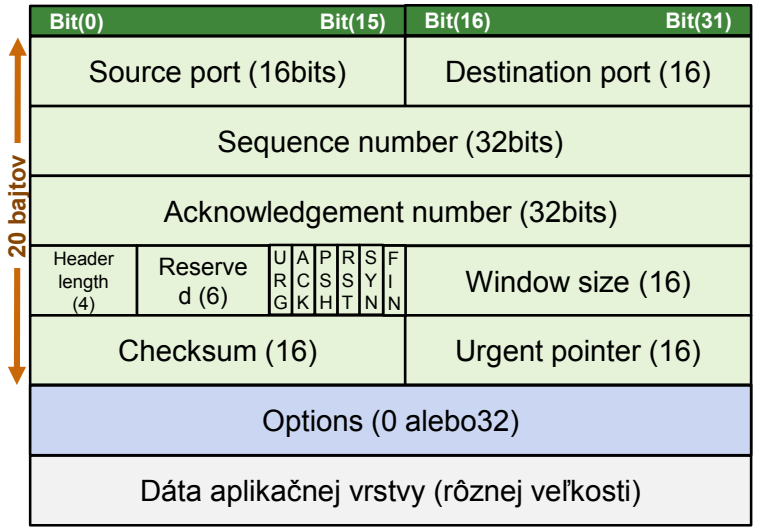
$$\min\{ \text{Sender window}, \text{Receiver window} \}$$

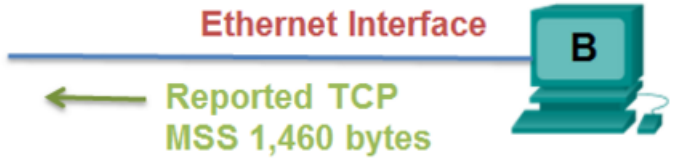
  - T.j. nemôžem vysielať dáta rýchlejšie ako si to praje príjemca

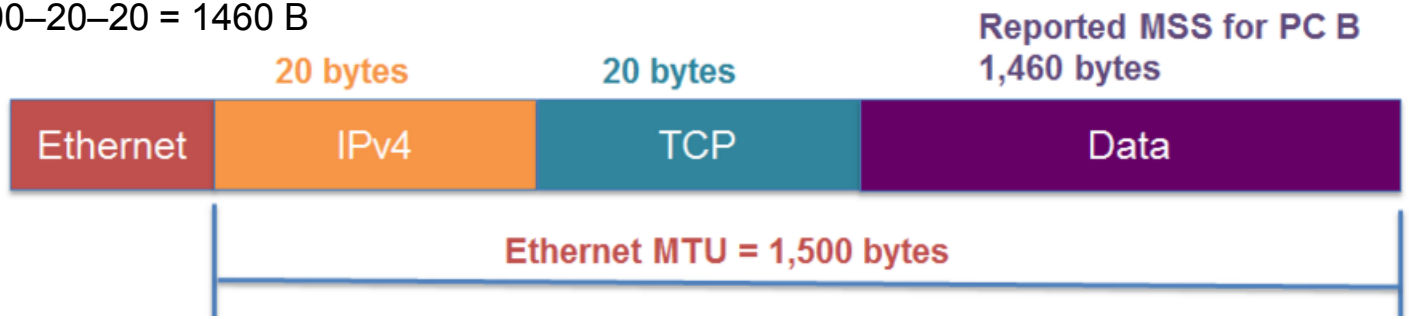
Doplňujúce voľby v TCP - Options

Dohadovanie MSS (maximum segment size) pri nadväzovaní spojenia

- 16 bitová hodnota
- Najväčší počet dát (v bajtoch), ktoré je schopné dané zariadenie prijať v jednom TCP segmente (bez hlavičky)
  - Obe komunikujúce zariadenia si to oznámia počas 3-way-handshake
  - Zväčša je to MTU (Maximum Transmission Unit) výstupného rozhrania, ktorým odíde daný segment zo zariadenia, mínus veľkosť IP a TCP hlavičky
  - Ethernetové rozhranie má MTU = 1500 B (najväčšie množstvo dát ktoré možno zabaliť do Ethernetového rámca).
    - Potom MSS = 1500–20–20 = 1460 B





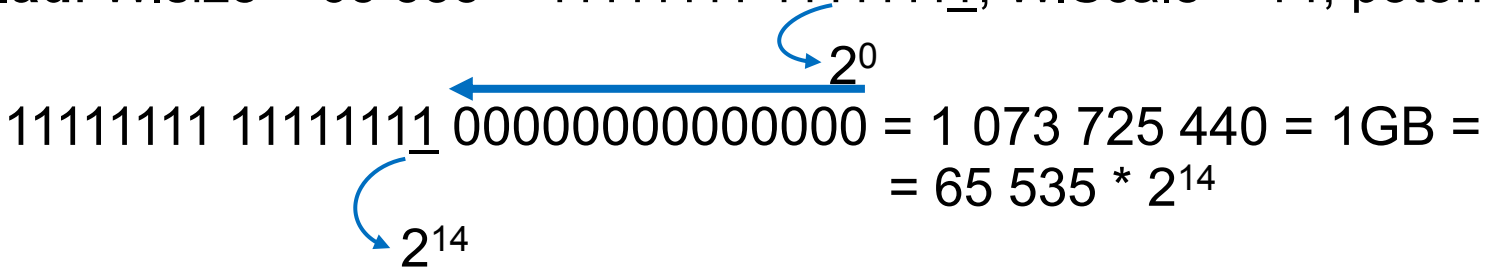


Doplňujúce voľby v TCP - Options

Násobky veľkosti okna (window scale)

- Pre niektoré aplikácie nestačí ani max. veľkosť okna (64kB = 65535 B)
- Napr. na trasách s veľkým oneskorením (satelitné siete), alebo siete s veľmi nízkym oneskorením, ale veľkou šírkou pásma (bandwidth), t.j. siete, pre ktoré súčin delay*bandwith vychádza vysoký (viac ako 10^5)
- Skutočná veľkosť okna = Window Size * $2^{WindowScale}$
  - Násobky (W.size) mocniny (W.Scale) so základom 2
    - $2^{WindowScale}$ je tzv. Scale factor
  - t.j. W.Size sa posunie – spraví sa shift doľava na vyššie mocniny

Príklad: W.size = 65 535 = 11111111 11111111, W.Scale = 14, potom:



(viac info v RFC 1323)

Doplňujúce voľby v TCP - Options

Násobky veľkosti okna (window scale)

- hodnota **Window scale** sa uvedie v hlavičke TCP v poli **Options**
  - nastaví sa počas 3-way handshake v SYN segmente, potom je fixná počas celého spojenia
  - max. hodnota je 14
- hodnota **Window size** sa uvedie v poli **Window size** v hlavičke
- toto uvidíme aj pri snifovaní cez Wireshark na cvičení

(viac info v RFC 1323)

Doplňujúce voľby v TCP - Options

Násobky veľkosti okna - Wireshark sniff: pohľad na Window Scale

```
[Next sequence number: 6806    (relative sequence number)]
Acknowledgment number: 218    (relative ack number)
Header Length: 20 bytes
› Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 131072]
[Window size scaling factor: 256]
Checksum: 0x7179 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
```

Časové značkovanie paketov (timestamps)

- Kvôli výpočtu RTT (round trip time)
- a následnej kalkulácii RTO (retransmission timeout) pre znovuopakovanie vysielania, pri strate segmentu a pod.

Doplňujúce voľby v TCP - Options

SACK – iná technika potvrdzovania segmentov

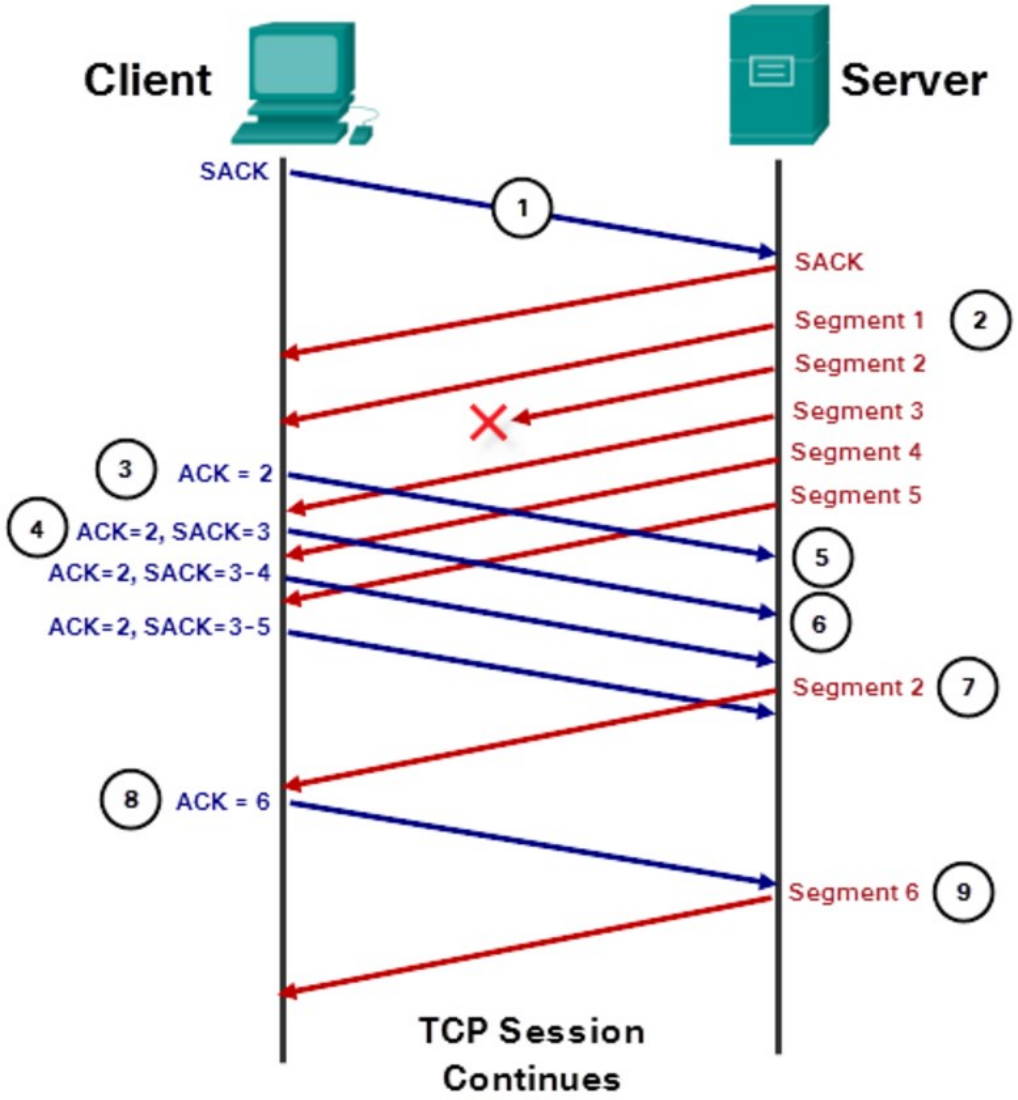
V základnom TCP sa pri strate 1 alebo viac segmentov, ktoré odosielateľ posiela bez potvrdenia v rámci dohodnutého okna, musia preposlať všetky segmenty od posledného potvrdeného bajtu (nielen 1-2 stratené) – toto rieši SACK – voliteľná implementácia TCP :

Selective Acknowledgement (SACK)

- pri strate segmentu, môže odosielateľ zopakovať prenos iba strateného segmentu, nemusí preposielať celé okno
  - Príjemca mu totiž vie oznámiť, ktorý segment mu chýba aj ktoré segmenty mu medzi časom prišli (po danom stratenom)
    - ACK = „*Toto očakávam že pošleš.*“ (hodnota sa uvedie v poli Acknowledgement number)
    - SACK= „*Toto mi už ale medzi časom prišlo, nepreposielaj.*“ (hodnoty sa uvedú v poli Options ako SACK)
- Použitie SACK si zariadenia dohodnú počas 3-way-handshake
  - Ak oba konce podporujú SACK, tak sa použije počas celého prenosu

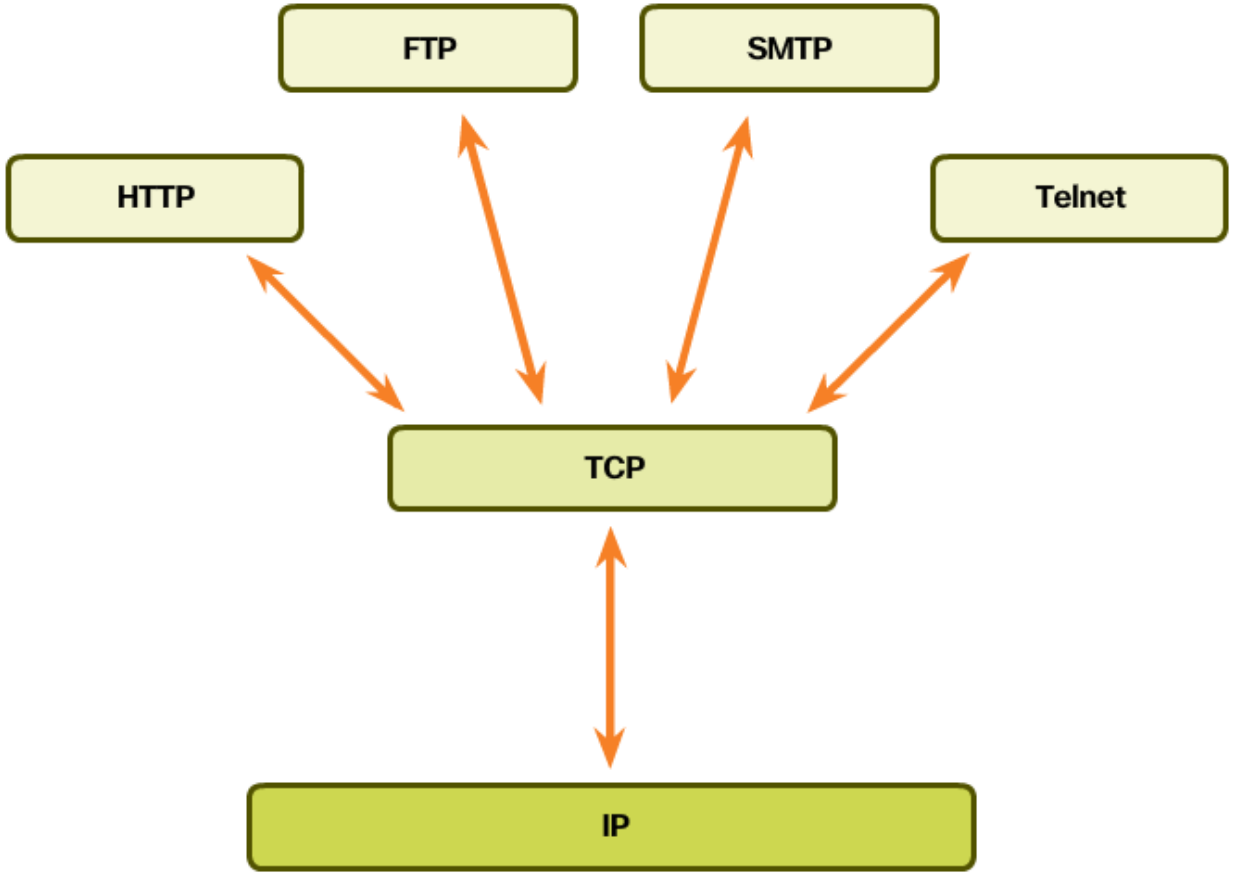
(viac info v RFC 2018)

Doplňujúce voľby v TCP - Options
SACK – iná technika potvrdzovania segmentov



Využitie TCP

- Všade kde potrebujem **spoľahlivú**, **spojovo orientovanú** službu doručovania tokov dát (streams) **s riadením toku** (flow control)



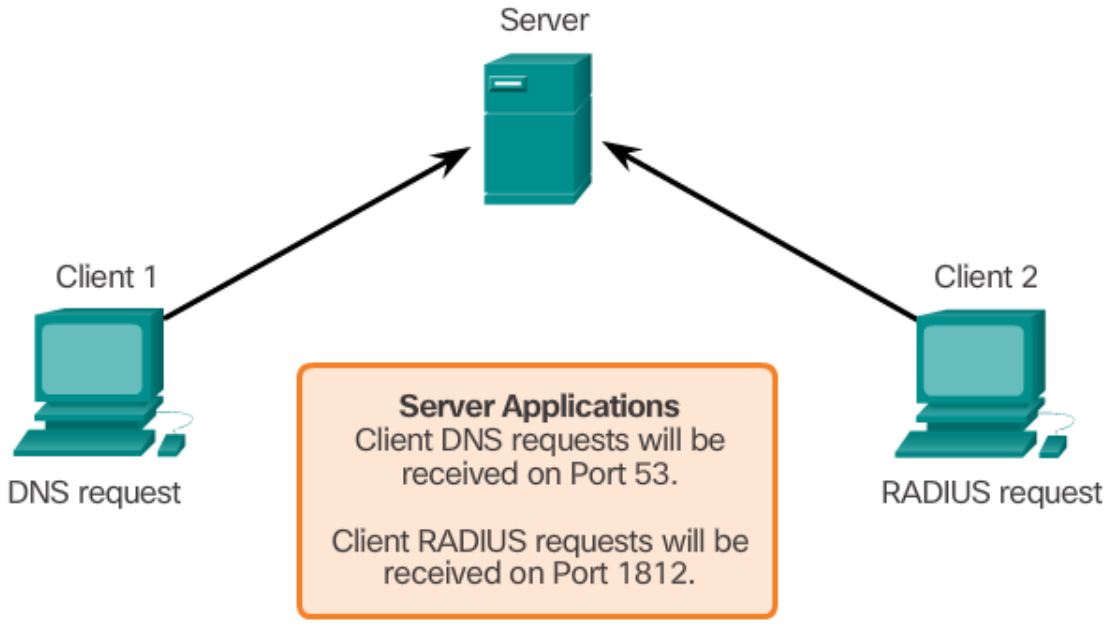
Téma 9.2.3:
UDP komunikácia

Protokol UDP

- UDP poskytuje **nespojovanú nespoľahlivú datagramovú** službu
  - UDP v skutočnosti **segmentáciu ani nerieši** – dáta musí segmentovať odosielajúca aplikácia
  - UDP len priloží k dátam prijatým z aplikačnej vrstvy vlastnú **hlavičku**, ktorá identifikuje zdrojový a cieľový proces, popisuje veľkosť datagramu a obsahuje kontrolný súčet, a segment odošle
  - U príjemcu sa prijatý segment po overení kontrolného súčtu odovzdá cieľovému procesu (podľa cieľového čísla portu)
  - UDP neoveruje, či segmenty dorazili všetky a v pôvodnom poradí
  - UDP nerieši opakovaný prenos chýbajúcich alebo poškodených segmentov
- Najčastejšie služby využívajúce UDP
  - DHCP (porty 67 a 68)
  - DNS (port 53)
  - RADIUS (porty 1812, 1813)
  - Voice over IP (porty 5060 a dynamické porty), video, IPTV

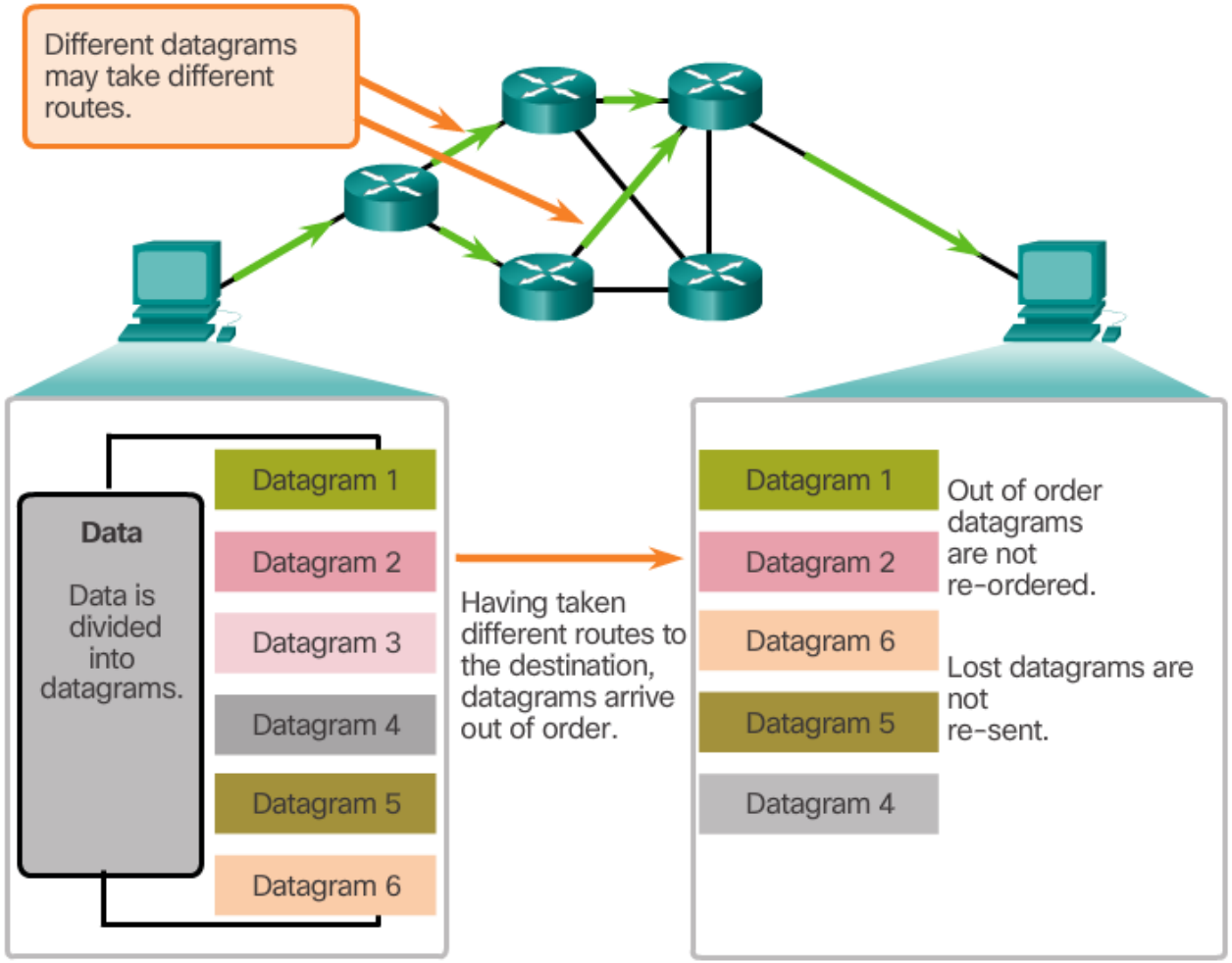
UDP klient-server proces

- Aplikácia na strane servera má rovnako ako v TCP priradené číslo portu zo skupiny „**well-known**“ alebo „**registered**“.
- Klientovi priradí OS číslo portu **dynamicky**
- Keďže sa nevytvára žiadne spojenie ako v TCP, služby a aplikácie na strane servera jednoducho prijímajú požiadavky UDP klientov (ak taká služba na danom servery beží – identifikuje sa číslom portu).



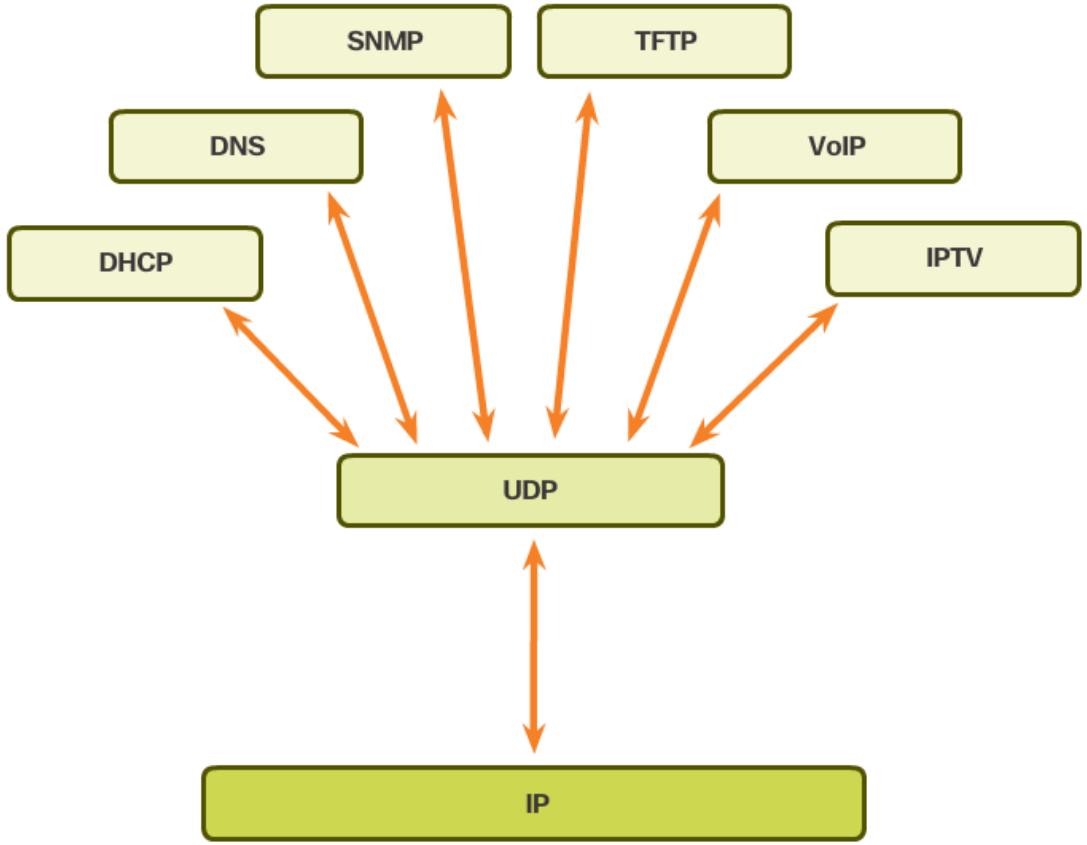
Protokol UDP

- Ak je potrebné, aby aj nad UDP bol zaručený prenos dát v pôvodnom tvare a poradí, musí si potrebné mechanizmy implementovať aplikácia, ktorá UDP používa



Využitie UDP

- UDP sa s výhodou používa v aplikáciách, v ktorých je spojovanosť alebo spoľahlivosť na obtiaž:
  - Video, voice over IP, obvykle real-time aplikácie, ktoré tolerujú drobné straty, avšak sú **citlivé na oneskorenie** segmentov
  - Aplikácie, ktoré používajú **multicast alebo broadcast**
  - Aplikácie, ktorých konverzácie sú spravidla typu „**otázka – odpoveď**“, kde réžia so zostavovaním spojenia a jeho ukončovaním je porovnateľná alebo väčšia s objemom dát v otázke a odpovedi (DHCP, DNS)
  - Aplikácie, kde hrozí, že udržiavanie veľkého počtu spojení spôsobí **veľkú spotrebu systémových prostriedkov** na komunikujúcich uzloch, alebo si aplikácia vie/chce zabezpečiť spoľahlivosť prenosu **sama** (SNMP, TFTP)



Úlohy transportnej vrstvy v TCP/IP

Zhrnutie

- Úlohy transportnej vrstvy v TCP/IP:
  - **Oddelenie konverzácií** (Track individual conversations)
  - **Segmentácia dát** (Segment data)
    a spätná rekonštrukcia pôvodných dát zo segmentov (Reassemble segments)
    - Umožňuje multiplexing konverzácií v sieti
    - Každý segment má svoju hlavičku - riadiace dáta transportnej vrstvy
  - **Identifikácia komunikujúcich aplikácií** (Identify the applications)
  - **Spojovanosť** (Connection-oriented data stream support)
    - Spojovo orientovaný prenos (connection-oriented)
    - Nespojovaný prenos (connectionless)
  - **Usporiadanosť** (Delivery ordering)
    - Rekonštrukcia segmentov v pôvodnom/správnom poradí (same-order delivery)
    - Rekonštrukcia segmentov v takom poradí ako prídu (no order data reconstruction)
  - **Spoľahlivosť** (Reliability)
    - Spoľahlivý prenos (reliable)
    - Nespoľahlivý prenos (unreliable)
  - **Riadenie toku dát** (Flow control) – kvôli predchádzaniu zahlteniu a stratám segmentov
    - S riadením toku dát (flow control)
    - Bez riadenia toku dát (no flow control)

MINISTERSTVO
ŠKOLSTVA, VEDY,
VÝSKUMU A ŠPORTU
SLOVENSKEJ REPUBLIKY

Ďakujem za pozornosť

Ohodnoť našu CNA na google:

- https://goo.gl/maps/BAnFvQKYCBpffcEX7