

TCP és UDP

Médiakommunikációs hálózatok (VIHIM161)
2013. évi fóliái alapján készült

Dr. Lencse Gábor
tudományos főmunkatárs
BME Hálózati Rendszerek és Szolgáltatások Tanszék
lencse@hit.bme.hu

2022. március 8.,
Budapest

Tartalom

- A TCP és az UDP közös jellemzői és képességei
- Transmission Control Protocol
  - TCP szegmens felépítése
  - Kapcsolat felépítése és lebontása
  - Megbízható átvitel megoldása
  - Forgalomszabályozás (flow control)
  - Torlódásvezérlés (congestion control)
- User Datagram Protocol

A TCP és az UDP közös jellemzői és képességei

A TCP és az UDP helye a rétegekben

Alkalmazási	(7. Application)
Megjelenítési	(6. Presentation)
Viszonylati	(5. Session)
Szállítási	(4. Transport)
Hálózati	(3. Network)
Adatkapcsolati	(2. Data Link)
Fizikai	(1. Physical)

ISO OSI

Alkalmazási	(Application)
Szállítási	(Transport)
Hálózati	(Internet)
Hordozó-hálózat	(Link)

TCP/IP

- *hálózati réteg:* datagramok átvitele a „host”-ok között
- *szállítási réteg:* a „host”-okon futó alkalmazások (processzek) közötti logikai kapcsolatok
  - a hálózati réteg szolgáltatásainak igénybevételére alapozva

A szállítási protokollok működési helye

- Logikai kapcsolatok a végpontokban futó alkalmazások között
- A szállítási protokollok csak a végpontokban futnak, a köztes csomópontokban nem
  - Emlékeztető: mely aktív eszközök mely rétegben működnek?
    - Router: 3. rétegben (benne vannak az 1-2. szintű funkciók is)
    - Switch, bridge: 2. réteg (benne vannak az 1. szintű funkciók is)
    - Hub, repeater: 1. réteg
  - A szállítási protokollok ezek egyikében sincsenek jelen!
- Az alkalmazások adategységeit *szállítási protokoll-adategység*ekbe tördeljük, a kapottakból pedig összerakjuk

Szállítási protokollok: TCP és UDP

- TCP – Transmission Control Protocol
- UDP – User Datagram Protocol
- Az TCP és az UDP közös képességei:
  - Portok kezelése
  - Multiplexelési képesség
- Alapvető különbség az TCP és az UDP között:
  - A TCP összeköttetés-alapú (connection-oriented) transzport-szolgáltatást nyújt
  - Az UDP összeköttetés-menteset (connectionless)

A TCP és az UDP közös képességei

- Portok kezelése
  - Az IP-rétegben a csomagok a „host”-nak vannak címezve
  - A „host”-okon belül: több alkalmazás, folyamat
  - Megkülönböztetésük: portok használatával
  - Portok: 16 bites számok (0-tól 65535-ig terjedő értékkészlettel)
  - Adott alkalmazás eléréséhez a megfelelő porthoz kell csatlakozni, például:
    - Web szerver (HTTP): 80-as port
    - FTP szerver: 21-es port
    - Biztonságos távoli bejelentkezés (SSH): 22-es port
  - Egyes alkalmazások TCP-t, mások UDP-t használnak (és vannak, amelyek mindkettőt)
- Multiplexelés/demultiplexelés
  - A port mechanizmus segítségével

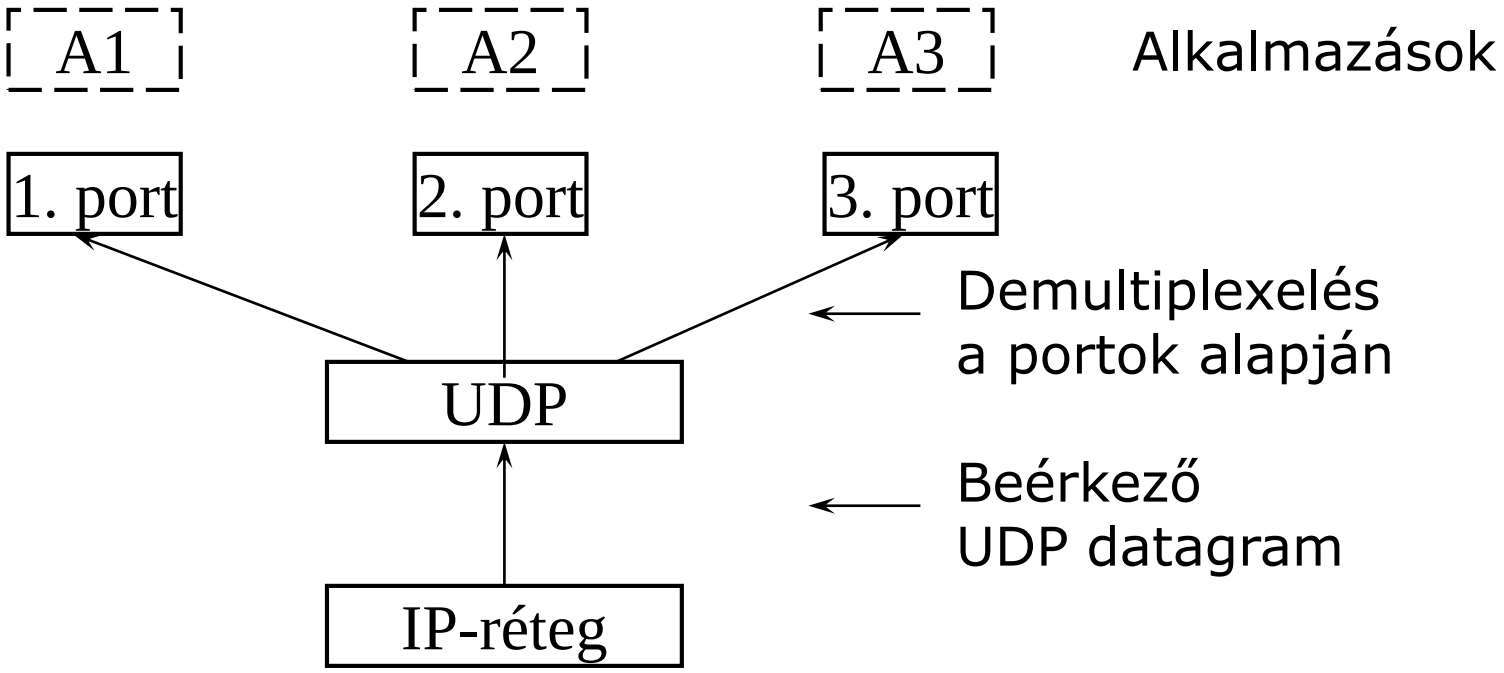
Portszámok szabványos kiosztása

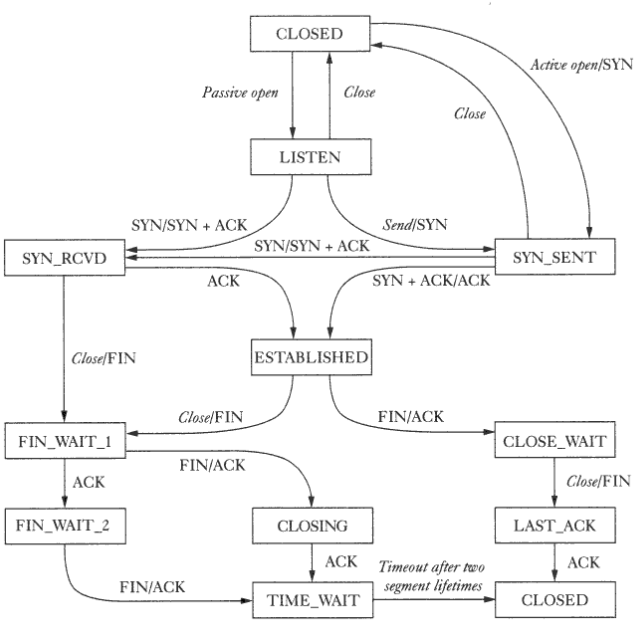
- Az IANA rendelkezik felőle
- Három tartományt definiáltak
  - 0-1023: System Ports (korábban: Well Known Ports)
    - Ezeket a portokat használják az alapvető, általánosan használt hálózati szolgáltatások. Ezekhez a portokhoz csak privilegizált szerver programok kapcsolódhatnak szolgáltatás nyújtása céljából.
  - 1024-49151: User ports (korábban: Registered Ports)
    - Ezeken a portokon egyéb szolgáltatások érhetők el. (Ha valaki kitalál egy szolgáltatást, igényelhet hozzá ilyen portszámot.) Ezeken a portokon történő szolgáltatásnyújtáshoz nem szükséges privilegizált módban futtatni a szerver programot.
  - 49152-65535: Dynamic / Private / Ephemeral Ports
    - Ebből a tartományból az operációs rendszer oszt ki portokat a kommunikációhoz a programoknak.

Bővebben: http://www.iana.org/assignments/port-numbers

Multiplexelés és demultiplexelés

- Példa:





TRANSMISSION CONTROL PROTOCOL

Az TCP fő jellemzői

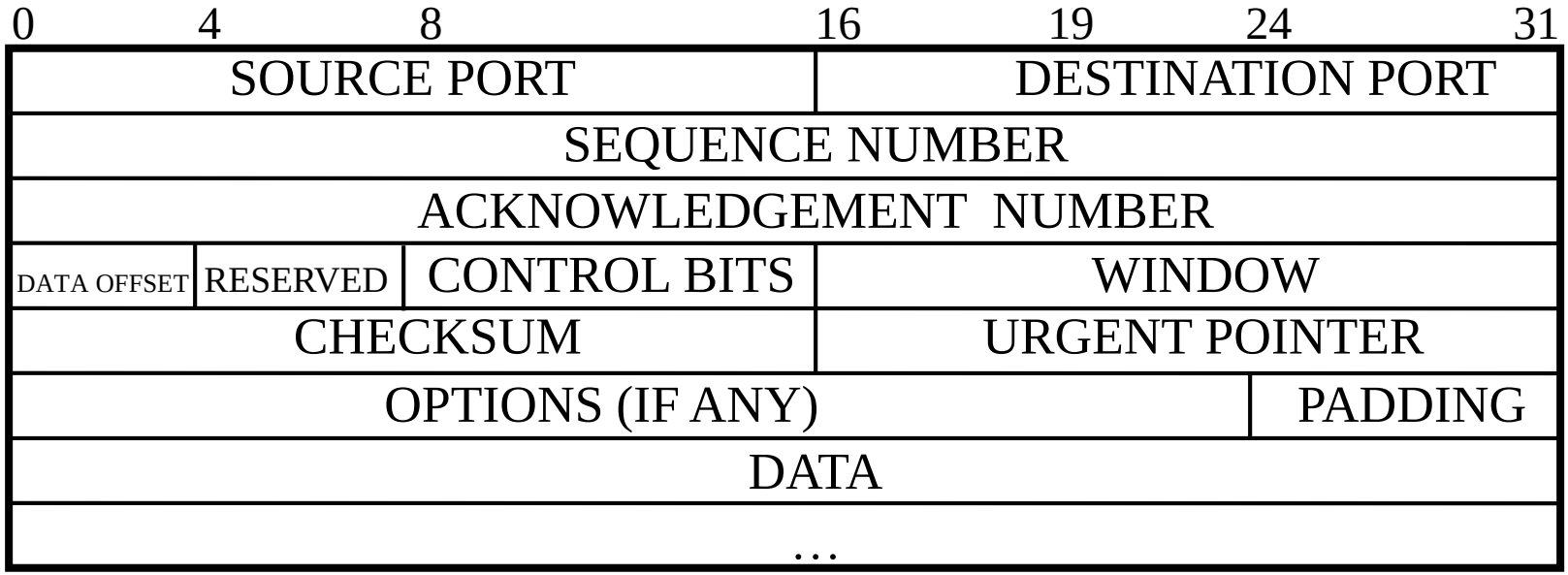
- **Célkitűzés:** megbízható szállítási szolgáltatás nyújtása az IP nem megbízható datagram szolgáltatására építve
- **Jellemzői:**
  - ***(Virtuális) összeköttetések:*** összeköttetés épül fel és marad fenn a kommunikáció tartamára
  - ***Stream-típusú szolgáltatás:*** *oktett-stream*ek sorrendhelyes átvitele
  - ***Strukturálatlan stream:*** nincsenek határolók a streamen belül
  - ***Pufferelt átvitel:*** a streamből a datagram megtöltéséhez szükséges mennyiséget várja össze
  - ***Duplex kapcsolatok:*** két független stream; irányonként egy-egy
  - ***Vezérlő információk küldése:*** az ellenkező irányban folyó streambe ágyazva (piggybacking)

Az TCP – ezekről lesz szó

- A szegmens* felépítése és a mezőinek értelmezése
- Összeköttetés-alapú kommunikáció: kapcsolat felépítése és lebontása
- Megbízható átvitel sorszámozás és pozitív nyugtázás segítségével
- Forgalomszabályozás (flow control) ablakmechanizmus segítségével
- Torlódásvezérlés (congestion control)

** a TCP PDU-jának neve*

A TCP szegmens felépítése



A TCP szegmens mezői – 1

- **Source Port** (16 bit)
  - A portszám azon a gépen, ahonnan a szegmenst küldték.
- **Destination Port** (16 bit)
  - A portszám azon a gépen, ahova a szegmenst küldték.
- **Sequence Number** (32 bit)
  - Az átvitel során az oktetteket sorszámozzuk. A sorszám megmutatja, hogy a szegmens adatmezőjének kezdő oktettje hányas sorszámot kapott.
- **Acknowledgement Number** (32 bit)
  - Annak az oktettnek a sorszámát adja meg, amelyiket várja; addig az összes nyugtázva van.

A TCP szegmens mezői – 2

- **Data Offset** (4 bit)
  - 32 bites egységekben mérve megadja az adatmező kezdetét a TCP szegmens kezdetéhez képest. Tulajdonképpen a fejrész hossza, mint IP-nél az IHL mező.
- **Reserved** (4 bit)
  - Későbbi használatra fenntartva. Az RFC 793 szerint még a 4 bites Data Offset után következő 6 bit nem volt használatban. Ez jobbról 2-vel csökkent RFC 3168 alapján.
- **Control bits** (8 bit)
  - RFC 793 szerint csak 6 darab volt, az RFC 3168 vezette be a CWR és az ECE vezérlőbiteket a korábban fenntartott 6 bites terület végén.
  - A vezérlőbiteknél mindig azok 1 értéke esetén áll fenn az, amit a nevük jelent.

A TCP szegmens mezői – 3

- A vezérlőbitek és értelmezésük
  - **CWR** (Congestion Window Reduced) - RFC 3168
    - A torlódási ablakot szűkítették.
  - **ECE** (ECN-Echo) - RFC 3168
    - Torlódásjelző visszhang.
  - **URG** (urgent)
    - A sürgős adat mutató érvényes.
  - **ACK** (acknowledgment)
    - A nyugta mező érvényes.
  - **PSH** (push)
    - Jelzi az adat késedelem nélküli továbbításának igényét.
  - **RST** (reset)
    - Egy host összeomlása, vagy más okból összezavart összeköttetés helyreállítására szolgál, illetve ha egy porton semmilyen program sem figyel, akkor az adott portra megkísérelt kapcsolatfelépítés esetén mindjárt az első SYN csomagra RST a válasz.

A TCP szegmens mezői – 4

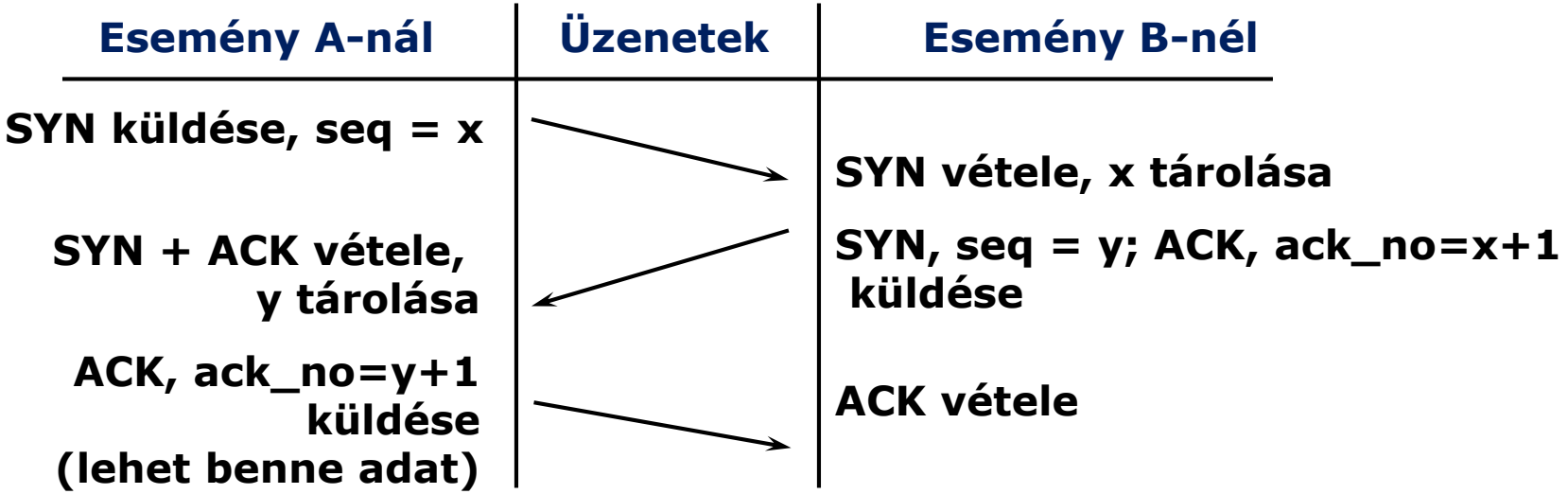
- A vezérlőbitek és értelmezésük (folytatás)
  - **SYN** (synchronize)
    - Összeköttetés létesítésére szolgál.
    - A kapcsolatfelépítés bemutatásánál bővebben foglalkozunk a használatával
  - **FIN** (finish)
    - Összeköttetés bontására szolgál
- **Window** (16 bit)
  - Az ablakméretet adja meg. A forgalomszabályozáshoz szükséges mechanizmus használja, részletesen foglalkozunk vele.

A TCP szegmens mezői – 5

- **Checksum** (16 bit)
  - A kiszámításához egy *pseudo headert* tesznek a TCP szegmens elé, ami többek között a forrás és cél IP-címeket is tartalmazza (lásd az UDP-nél), az adategység végén pedig esetleg egy 0 értékű oktettel kiegészítik, ha anélkül páratlan oktettből állna (mert 16 biten történik az ellenőrző összeg kiszámítása)
  - A számítás ugyanazzal a módszerrel történik, mint az IP-nél.
- **Urgent Pointer** (16 bit)
  - A *sürgős adat* az adatfolyam menetét megszakítva a többi adatot megelőzve feldolgozandó adat. A sürgős adat az adatmező elejétől kezdődik, a mutató a sürgős adat után következő (már nem sürgős) adat kezdetére mutat.
- **Options**, **Padding**
  - Az IP-hez hasonlóan itt is lehetnek opciók és szükség lehet helykitöltésre.

A TCP kapcsolat felépítése

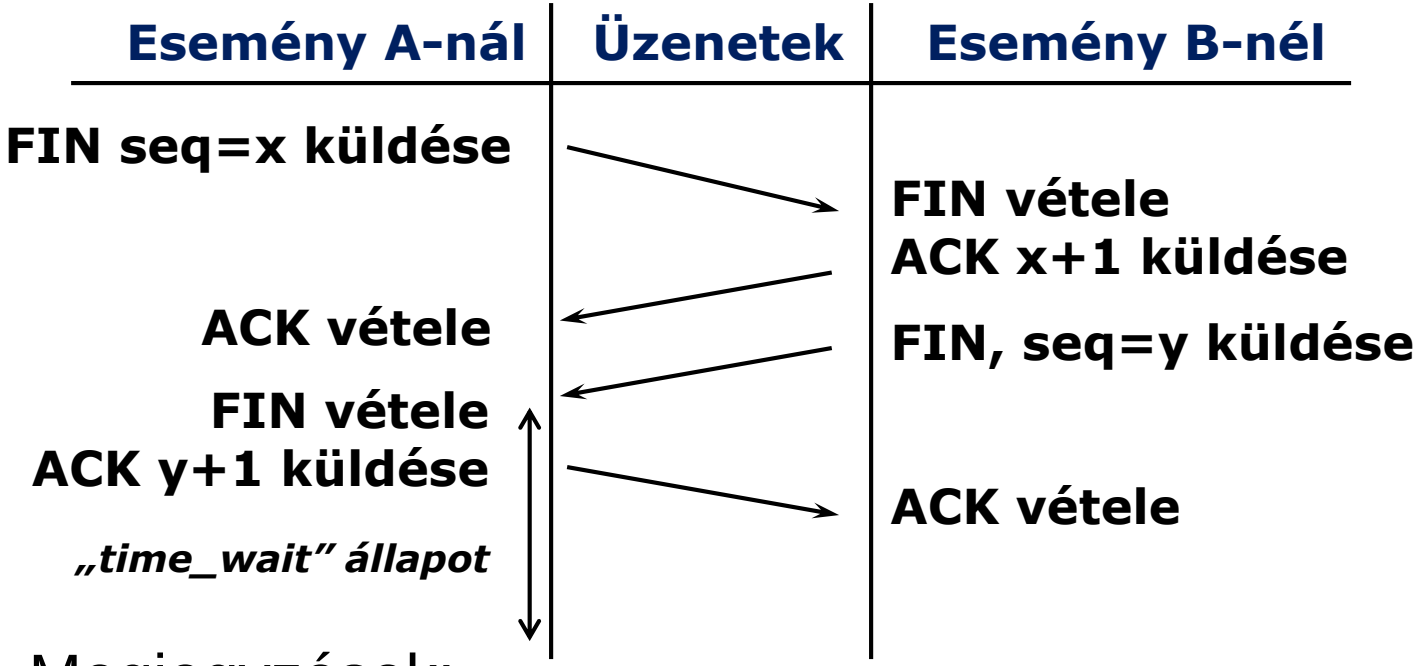
- A *háromutas kézfogás* (3-way handshake) eljárás



- Megjegyzések
  - A két kezdő sorszám nem nulla, hanem véletlenszerűen választott
  - Mindkét oldalon ún. kapcsolatstruktúrában tárolják a kapcsolat paramétereit.
  - A harmadik üzenet után a felek készen állnak a megbízható kommunikációra

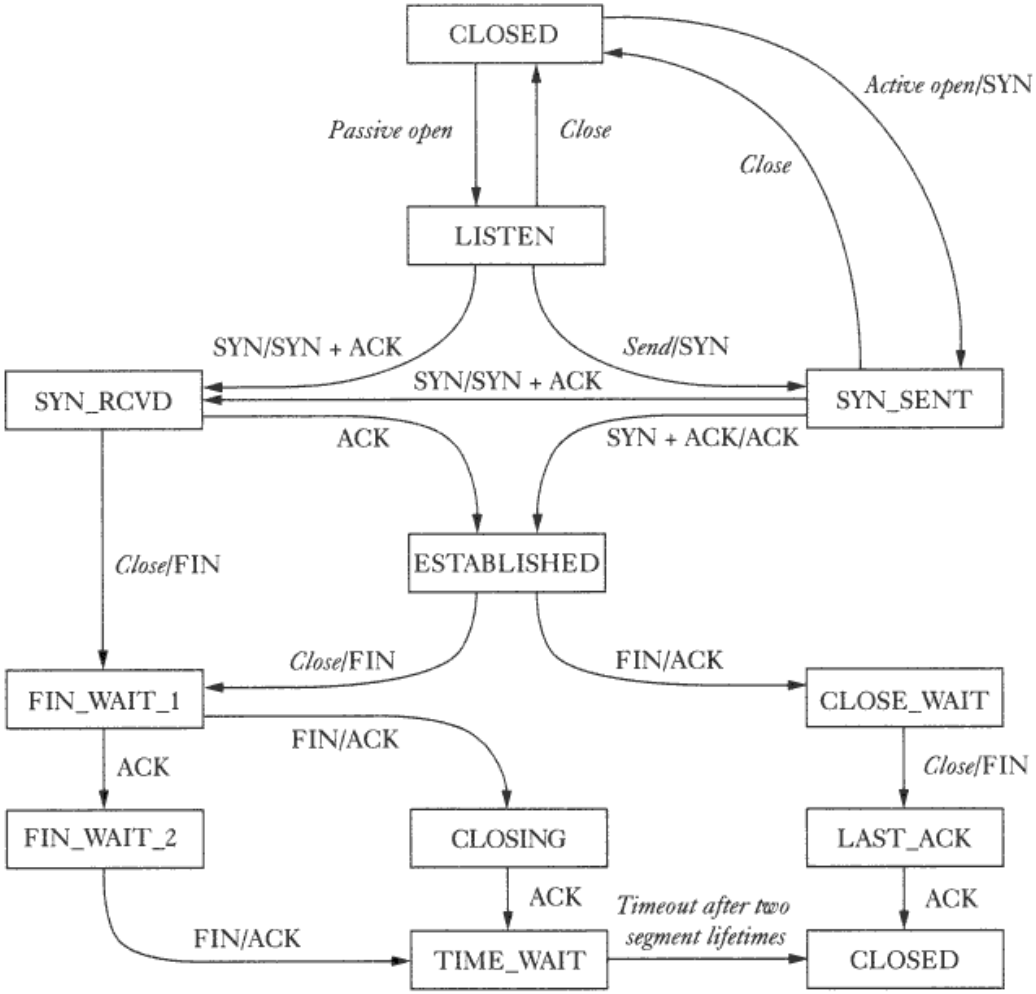
A TCP kapcsolat lebontása

- A *négyutas kézfogás* (4-way handshake) eljárás



- Megjegyzések:
  - A kapcsolat lezárását kezdeményező fél a 3. lépésben (a FIN vételére) lép be a *time_wait* állapotba
  - A 2. és 3. lépés akár össze is vonható, B ACK+FIN-t is küldhetne

A TCP állapotátmenet diagramja



Forrás: http://ssfnet.org/Exchange/tcp/tcpTutorialNotes.html#ST (Nem kell megtanulni!)

A megbízható átvitel eszközei

- Ellenőrző összeg használata
  - IP: csak a fejrészre
  - TCP: az egész szegmensre
    - sőt: pseudo header
      - IPv4-nél 3x4=12 bájt: benne IP-címek és Protocol mező
      - IPv6-nál jóval nagyobb: https://tools.ietf.org/html/rfc8200#section-8.1
- Sorszámozás és nyugtázás
  - Irányonként
  - A bájtokat sorszámozzuk
  - A kezdőértékeket a kapcsolatfelvétel már beállította

A TCP nyugtázása

- Eredetileg kumulatív nyugtázás
  - ha a nyugta értéke n, akkor azt jelenti, hogy:
    - (n-1) sorszámig mindent sikeresen vett
    - n-től várja
  - A kezdeti implementációkban (Tahoe, Reno) csak ez
- Opcióként: SACK: Selective ACK (RFC 2018)
  - SYN szegmensben engedélyezhető („SACK Permitted” opció)
  - Továbbiakban egy opcióban akár több blokk is megadható

Várakozás nyugtázásra

- Várakozás ACK-ra
  - time–out esetén újra el kell küldeni
- Mekkorára válasszuk a time–out-ot?
- Probléma a túl kicsivel és a túl naggyal
- Megoldás: a teljes terjedési időhöz (RTT - round-trip time) igazítani, adaptívvá tenni

Time-out meghatározása

- Eredeti algoritmus
  - Measure SampleRTT for each segment/ACK pair
  - Compute weighted average of RTT
  - EstimatedRTT = a*EstimatedRTT + b*SampleRTT, where a+b = 1
    - a between 0.8 and 0.9; b between 0.1 and 0.2
  - Set timeout based on EstimatedRTT
    - TimeOut = 2 * EstimatedRTT
  - Rosszul kezeli az elvesző nyugtát, és a „szórást”
- Karn/Partridge Algoritmus
  - Do not sample RTT when retransmitting
  - Double timeout after each retransmission
- Jacobson/Karels Algoritmus

  Érdeklődőknek:
  http://ssfnet.org/Exchange/tcp/tcpTutorialNotes.html#AR

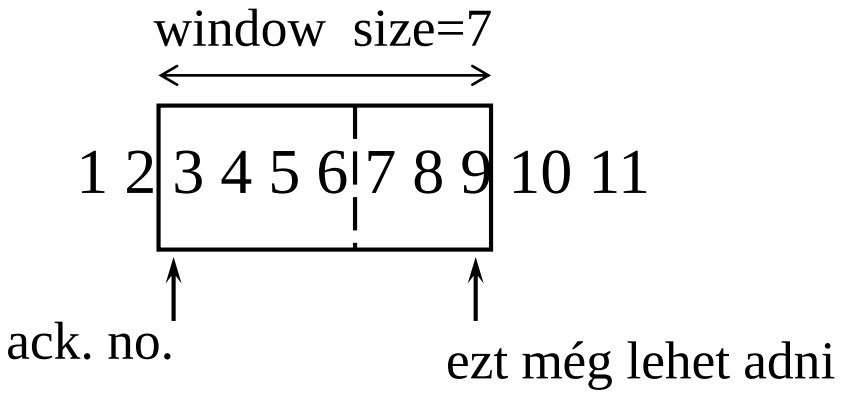
Forgalomszabályozás – 1

- A *forgalomszabályozás* (flow control) célja annak az elkerülése, hogy egy gyorsabb állomás egy lassabbat forgalommal elárasszon úgy, hogy a lassabb nem képes azt feldolgozni.
- A problémára a nyugta megvárása nem jó megoldás, erre nézzünk meg egy számpéldát:
  - Legyen A és B távolsága 200km, az őket összekötő üvegszál törésmutatója n=1,5, a fénysebesség 300000km/s (vagyis üvegben 200km/ms), az átviteli sebesség 100Mbit/s, MTU=1000 bájt.
  - Egy 1000 bájtos csomag 8000 bit, ennek a leadásához 80 us szükséges. Az oda-vissza út ideje csak a terjedési időt számítva is 2ms. Ez azt jelenti, hogy minden 2000us-ból 80us-ot tudunk kihasználni, ami 4%!
  - A nyugtát tehát nem szabad megvárni!

Forgalomszabályozás – 2

- A TCP forgalomszabályozásra a *csúszó ablak* (sliding window) módszert használja.
  - Az ablak értéke egy hitelkeretnek tekinthető, azt fejezi ki, hogy egy állomás ennyi adat elküldését engedélyezi a másik fél számára úgy, hogy a másik fél még nem kapott nyugtát róla.
  - A kapcsolat felépülésekor az állomások közlik a másik féllel a kezdő ablakméretüket is
  - A kommunikáció során az állomások az ablakméretet megváltoztathatják, de ha csökkentik, azt csak úgy tehetik, hogy annak jobb széle (ami a még elküldhető oktett sorszáma) ne mozogjon balra (ha egy oktettről egyszer kijelentettük, hogy elküldhető, azt már nem vonhatjuk vissza).

Forgalomszabályozás – 3



- Az ablak bal széle az utolsó nyugta értékétől kezdődik (ez 3, ami azt jelenti, hogy 2-ig nyugtázva, a 3 következik), az ablak mérete pedig 7.
- Ez a küldő számára azt jelenti, hogy a 3-tól 9-ig terjedő sorszámú, összesen 7 darab oktett az, amit anélkül elküldhet, hogy nyugtát kapna. Például elküldi a 3-6 sorszámú oktetteket.
- Ekkor nem kell nyugtára várnia, hanem küldheti a 7-9 sorszámúakat is.
- Egy 7-es nyugtával azonban legalább 3-as ablakméretet kell kapnia
- Ha viszont változatlan ablakméret mellett kap egy 7-es nyugtát, akkor a 7-től 13-as oktettek küldhetők: az ablak előre csúszott.

Elég-e az ablakméret?

- A TCP fejrész Window mezőjének mérete 16 bit.
- Elég-e ez? Nézzünk meg egy számpéldát:
  - A 16 bites mező 64 kB-os (pontosabb 65535 bájt) ablakméretet tesz lehetővé, ami 512 kbit.
  - Legyen A és B távolsága 2000km, és az átviteli sebesség 1Gbit/s.
  - Az oda-vissza út ideje csak a terjedési időt számítva is 20ms. Ennyi idő alatt 20Mbit adatot tudnánk leadni, de az ablakméret csak 0.5Mbit-et tesz lehetővé!
- Megoldás: Window Scaling Option, RFC 7323
  - Ha mindkét fél jelezte a SYN szegmensben, hogy támogatja
  - Használatkor az opcióban n-et küldik, a Window mező számértéke 2^n-nel szorzandó, n<=14

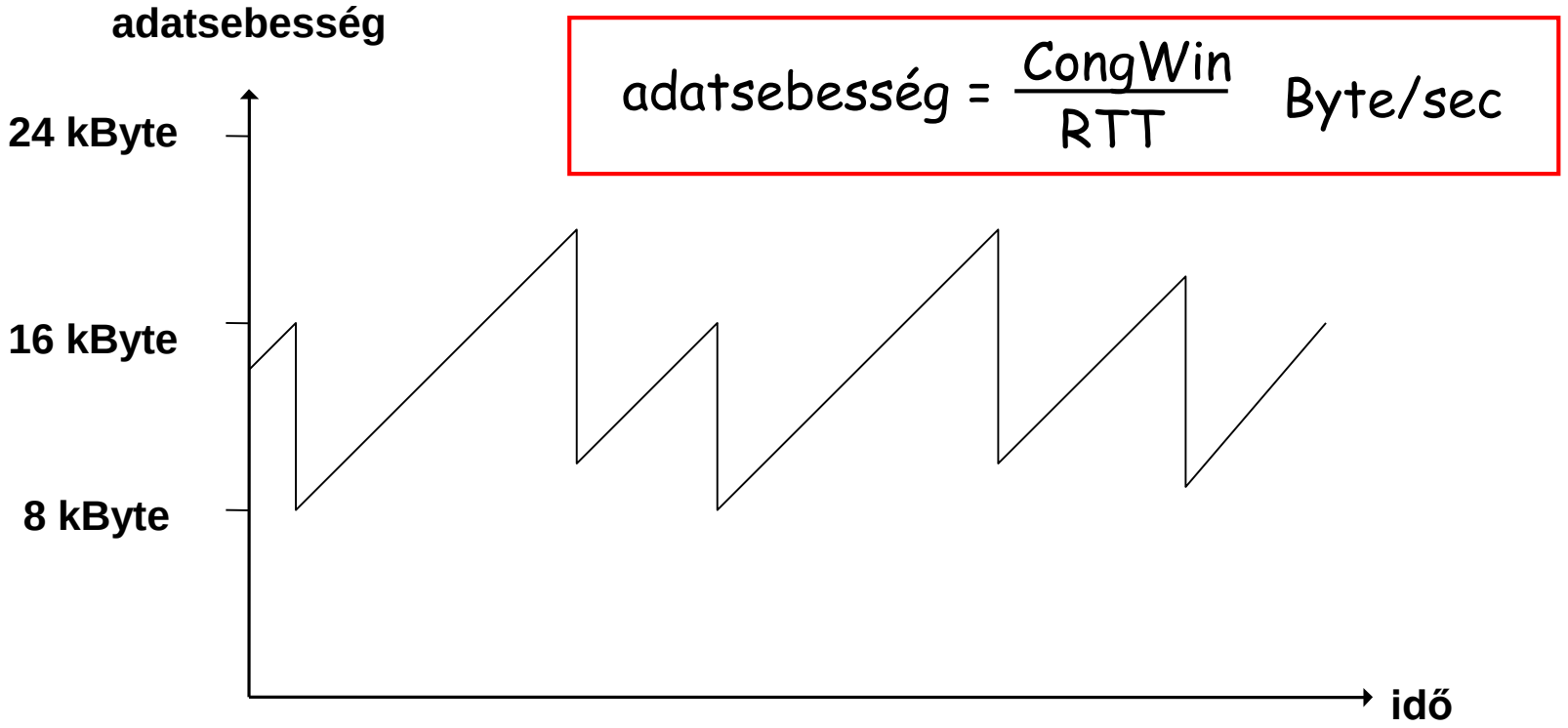
Torlódásvezérlés

- A *torlódásvezéslés* (congestion control) célja annak az elkerülése, hogy valamely közbenső hálózati eszköz (router, link) túlterhelése miatt a hálózat teljesítőképessége radikálisan csökkenjen (congestive collapse).
- Miért is fordulhatna elő ilyen állapot?
  - Ha a hálózat terhelése túl nagy (megközelíti a kapacitását), akkor a csomagvesztés megnő, és a TCP elkezdi újra küldeni a nyugtázatlan szegmenseket.
  - Az újraküldés okozta terhelésnövekedés további csomagvesztéseket okoz, és az önmagát gerjesztő folyamat odáig jut, hogy a hálózat kapacitásának csak a töredékét kitevő átvitelre képes, ráadásul rendkívül rossz minőségi jellemzők mellett.

Torlódásvezérlés a TCP-ben (elvek)

- Módszer: ha úgy érzékeljük, van elég átbocsátóképesség, akkor növeljük az adatsebességet, amíg torlódásra utaló jeleket nem tapasztalunk
- Additive increase – multiplicative decrease (AIMD) módszer
  - Új paraméter: congestion window (rövidítve: CongWin)
  - Kezdetben: CongWin:=MSS (maximum segment size)
  - Növeljük a CongWin-t minden RTT (round-trip time) alatt MSS-sel, amíg vesztést nem érzékelünk
  - Csökkentsük a CongWin-t felére minden vesztéskor
  - Küldhető adatok = min(vevő Window size, CongWin)
- Hogyan érzékeljük a torlódást?
  - timeout letelt, nem jött nyugta
  - többszörös nyugta érkezett ugyanarra a szegmensre (miért?)

Az adatsebesség alakulása AIMD esetén



$$\text{adatsebesség} = \frac{\text{CongWin}}{\text{RTT}} \quad \text{Byte/sec}$$

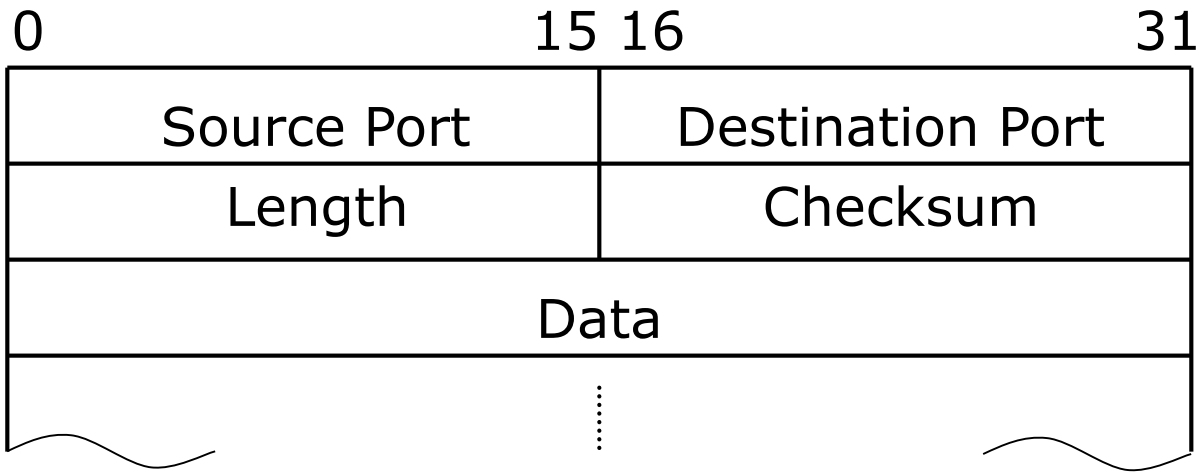
TCP torlódásvezérlés: további részletek

- Az AIMD kiegészítései:
  - „Slow Start” az összeköttetés kezdetén a sebesség gyors (exponenciális) növelése az első vesztésig vagy egy küszöbig (ssthresh), utána AIMD
  - Speciális viselkedés többszörös nyugták (3 duplicate ACK, azaz 4 ACK ugyanarra a szegmensre) esetén (fast retransmit, Fast Recovery), lásd: TCP Tahoe, TCP Reno
- Számos TCP implementáció közül néhány fontosabb:
  - TCP Reno, TCP New Reno (RFC 6582)
  - TCP CUBIC (Linux default volt)
  - Compound TCP (Microsoft)
  - TCP BBR (új, nem csomagvesztés alapú)
- Érdeklődőknek:

  http://en.wikipedia.org/wiki/TCP_congestion_avoidance_algorithm

User Datagram Protocol

A User Datagram felépítése

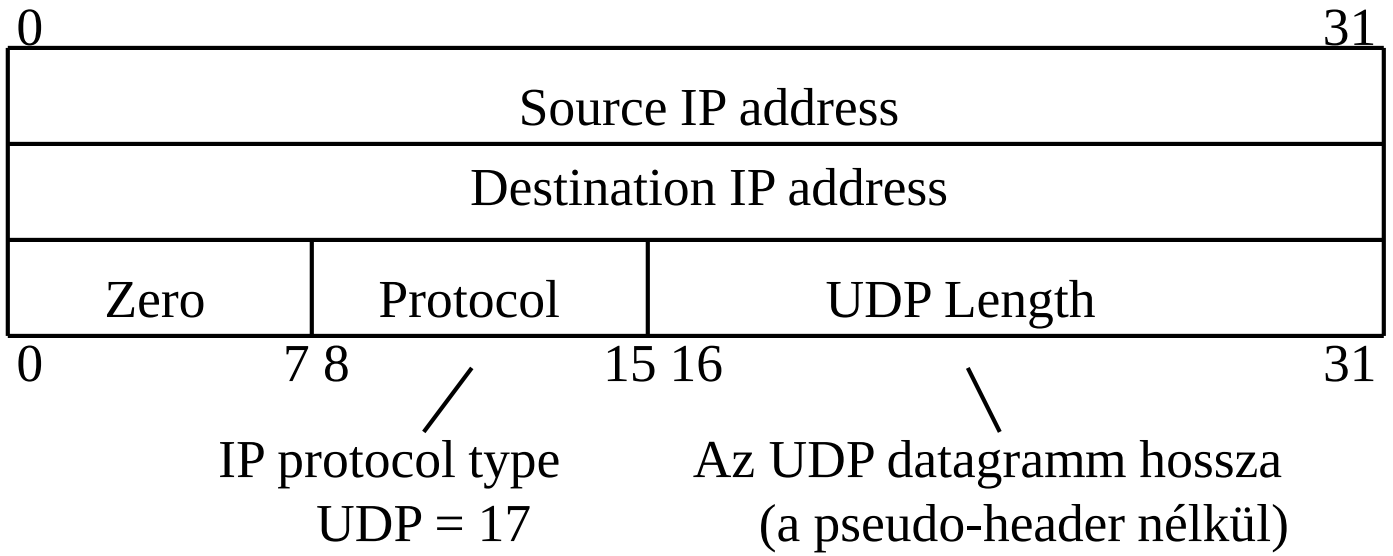


A User Datagram mezői

- **Source Port** (16 bit)
  - A portszám azon a gépen, ahonnan a user datagramot küldték.
- **Destination Port** (16 bit)
  - A portszám azon a gépen, ahova a user datagramot küldték.
- **Length** (16 bit)
  - A user datagram hossza oktettekben. (min. 8)
- **Checksum** (16 bit)
  - A TCP ellenőrző összeghez hasonlóan, ún. pszeudo-fejrésszel együtt számolják, ami tartalmazza az IP címeket is. (Számítási módszere is azonos.)
  - Opcionális: ha nincs, akkor a mező értéke: 0.

UDP checksum számítása

- A checksum számítási módja: 1-es komplemens 16 bites szavakra
- Tartalmazza az IP-címeket is
  - annak ellenőrzésére, hogy a datagramm elérte a helyes címzettet, nemcsak a helyes portot
- „Pseudo-header” használatával:



A TCP és az UDP összehasonlítása

- Mindkettő host layer/transport layer protokoll
- Mindkettő portokat kezel
  - multiplexelés/demultiplexelés
  - ezáltal interface nyújtása az alkalmazói folyamatok felé
- A TCP összeköttetés-alapú, megbízható transzport szolgáltatás
  - sorrendhelyes, hibamentes szállítást nyújt
  - ára: késleltetés (kapcsolat felépítése és bontása)
- Az UDP összeköttetés-mentes, best effort szolgáltatás
  - nem garantál célba juttatást, csak hibajelzést nyújt
  - gyorsan célba juttat

Néhány alkalmazás szállítási protokollja

Alkalmazás	***Alkalmazási rétegbeli protokoll***	***Használt transzport-protokoll***
E-mail	SMTP	TCP
Távoli bejelentkezés	Telnet, SSH	TCP
Web-elérés	HTTP	TCP
File-átvitel	FTP	TCP
Névfeloldás	BIND	UDP, TCP
Routing	RIP	UDP
Hálózatmenedzsment	SNMP	UDP, TCP
VoIP, média-streaming	RTP vagy nem szabványos	UDP

Összefoglalás

- A TCP és az UDP közös jellemzői és képességei
- Transmission Control Protocol
  - TCP szegmens felépítése
  - Kapcsolat felépítése és lebontása
  - Megbízható átvitel megoldása
  - Forgalomszabályozás (flow control)
  - Torlódásvezérlés (congestion control)
- User Datagram Protocol

Kérdések?

KÖSZÖNÖM A FIGYELMET!

Hálózati Rendszerek és Szolgáltatások Tanszék

Dr. Lencse Gábor
tudományos főmunkatárs
BME Hálózati Rendszerek és Szolgáltatások Tanszék
lencse@hit.bme.hu