



HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK

HÁLÓZATOK ALAPJAI ÉS ÜZEMELTETÉSE

TCP

2023. március 17.

Mészáros András

BME Hálózati Rendszerek és Szolgáltatások Tanszék

meszarosa@hit.bme.hu



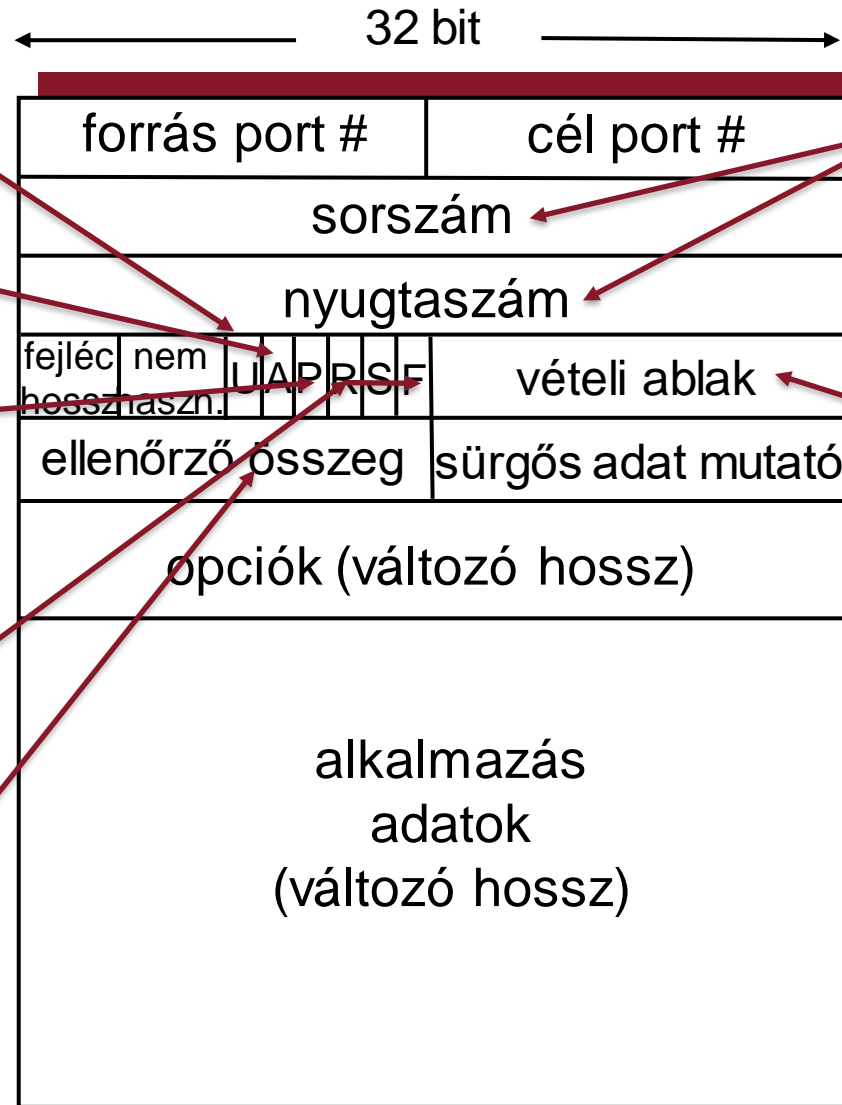
1. A TCP protokoll
2. Szegmensek nyugtázása
3. Forgalomszabályozás
4. Kapcsolatkezelés

A fóliák elkészítéséhez felhasználtuk Jim Kurose és Keith Ross „Számítógép hálózatok működése” című könyvéhez készült fóliákat.

- Több változat, különböző RFC-kben
 - 793, 1122, 1323, 2018, 2581
- **Pont-pont** elrendezés
 - Egy küldő
 - Egy fogadó
- Megbízható, sorrendtartó adatfolyam
 - Nincs korlátozva a tartalom, sima bájtok
- Csővezetékezett (pipelined)
 - Aktuálisan küldhető adatok „ablaka”
 - Torlódáskezelésben és forgalomszabályozásban használjuk
- **Ablakozás** a küldő és a fogadó oldalon
- **Full-duplex** adatátvitel
 - Kétirányú adatáramlás ugyanazon a összeköttetésen
- Maximális szegmensméret
 - MSS (maximum segment size)
- **Összeköttetés** alapú (connection-oriented)
 - Kézfogás: vezérlő üzenetek cseréje, a küldő és a fogadó megfelelő állapotba hozására
 - A kliens (küldő) kezdeményezi
 - Az adatcsere megkezdése előtt
 - Lezárás
- **Forgalomszabályozás**
 - A küldő ne terhelje túl a fogadót
- **Torlódáskezelés**
 - Igazodás a hálózat aktuális állapotához

TCP SZEGMENS-SZERKEZET

- URG: sürgős adat (általában nem használt)
- ACK: nyugtamező érvényes
- PSH: (push data now) azonnal továbbítandó (általában nem használt)
- RST, SYN, FIN: összeköttetést vezérlő flagek (felépítés, lebontás parancsok)
- Internet ellenőrző összeg (mint UDP-nél)

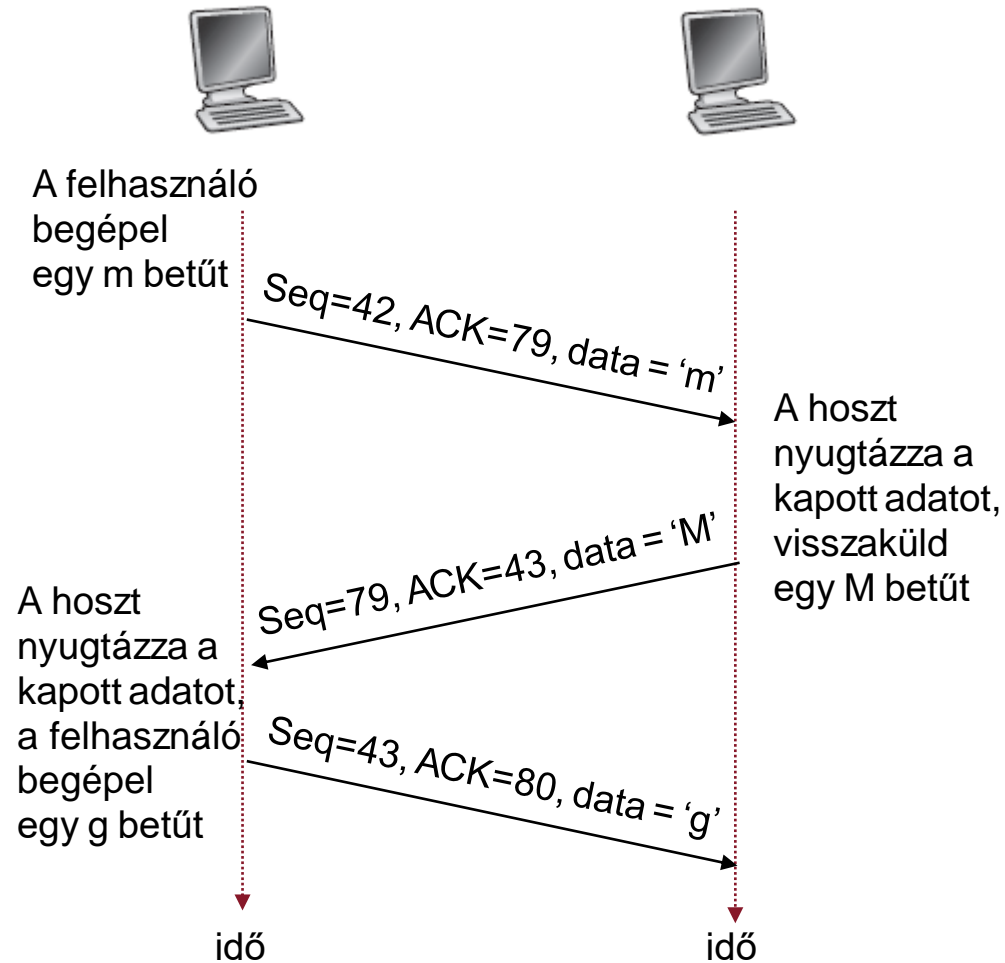


- Alkalmazás adat bájtsorszám (nem a szegmens sorszáma)
- A vevőnél fogadható bájtszám

1. A TCP protokoll
2. Szegmensek nyugtázása
3. Forgalomszabályozás
4. Kapcsolatkezelés

SORSZÁM ÉS NYUGTASZÁM

- Sorszám (SEQ, sequence number)
 - A teljes adat hányadik bájttal kezdődik a szegmens adatrésze
- Nyugta (ACK, acknowledgement)
 - Vételi információ
 - Melyik bájjal kezdődjön a következő küldése
 - Nyugta arról, hogy az addigiak megjöttek
 - Összegző (kumulatív) nyugta: minden addigi megjött
- Mi lesz a nem sorrendben érkező szegmensekkel?
 - Nincs előírva a specifikációban
 - Megvalósításfüggő



- Meddig várjunk a nyugtára?
- Ha túl sokat késik, akkor időtúllépésnek (timeout) veszi a küldő
- A **Timeout** idő értéke
 - Túl kicsi: felesleges időtúllépés
 - Felesleges újraküldések
 - Túl nagy: lassú reagálás egy szegmens elvesztésére
 - Nagyobb legyen, mint az RTT
 - Az RTT nem állandó
- Az RTT **becslése** szükséges
- Mérés alapú minta:
SampleRTT
 - Egy szegmens elküldési és az arra kapott nyugta érkezési idejének különbsége
 - Az ismételten átvitt szegmenseket figyelmen kívül hagyva
- A **SampleRTT** nem állandó
 - Elég változékony
 - Simítani kell
 - Több minta alapján készült, mozgó átlag kell a becsléshez

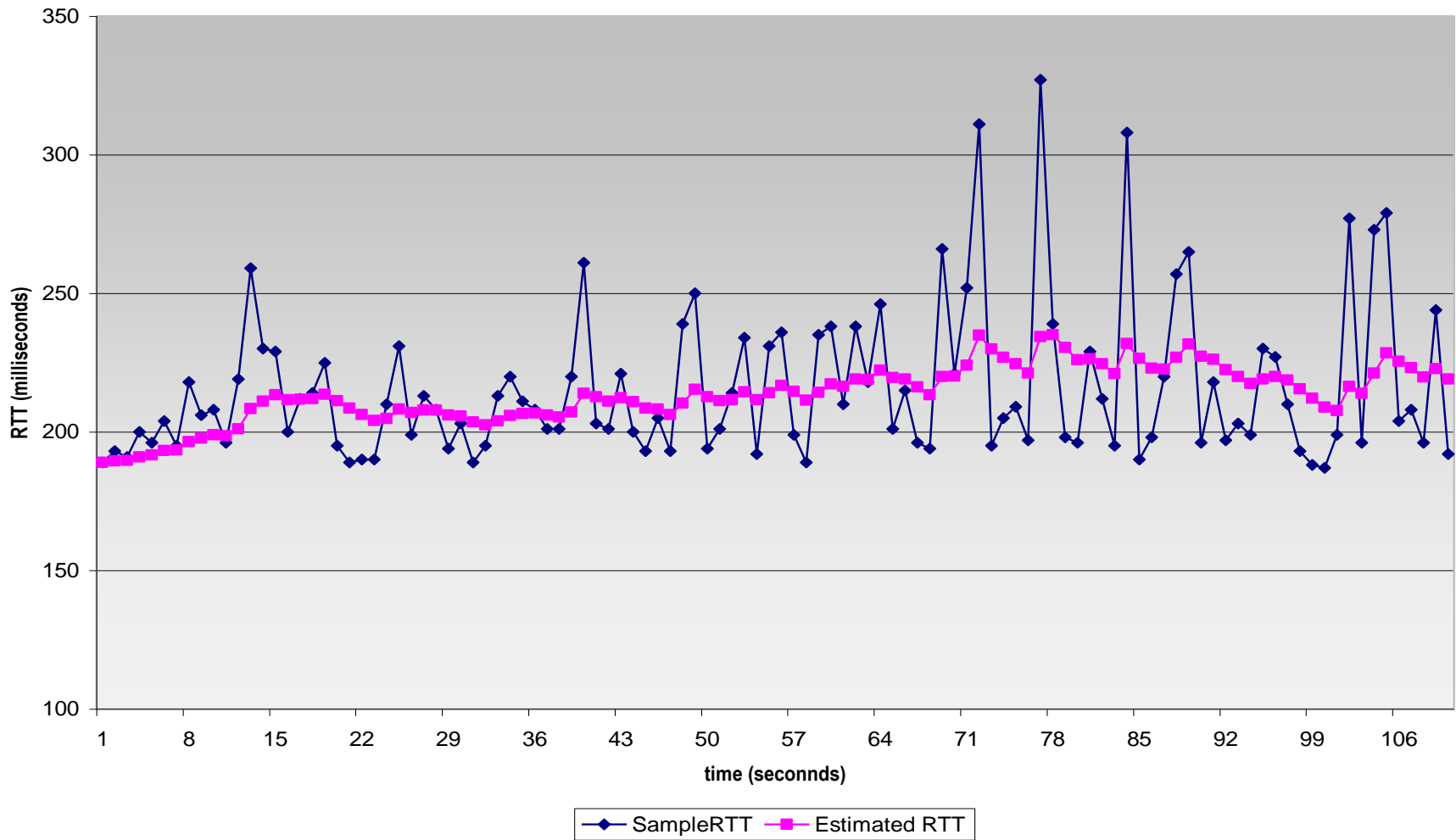
- Mozgóátlag exponenciális súlyozással
 - Exponential weighted moving average
 - A korábbi minták hatása exponenciálisan (hatványozottan) csökken

$$\text{EstimatedRTT} = (1 - \alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$$

- Javasolt érték: $\alpha = 0.125$

PÉLDA AZ RTT BECSLÉSÉRE

RTT: gaia.cs.umass.edu to fantasia.eurecom.fr



- A becsült RTT némi biztonsági ráhagyással megfelel
 - Ha nagyobb a minták dinamikája, akkor nagyobb ráhagyás kell
 - Mozgóátlag exponenciális súlyozással
 - Figyelembe véve, hogy mennyire tér el az aktuálisan mért RTT a becsléstől

$$\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$$

- Javasolt érték $\beta = 0.25$

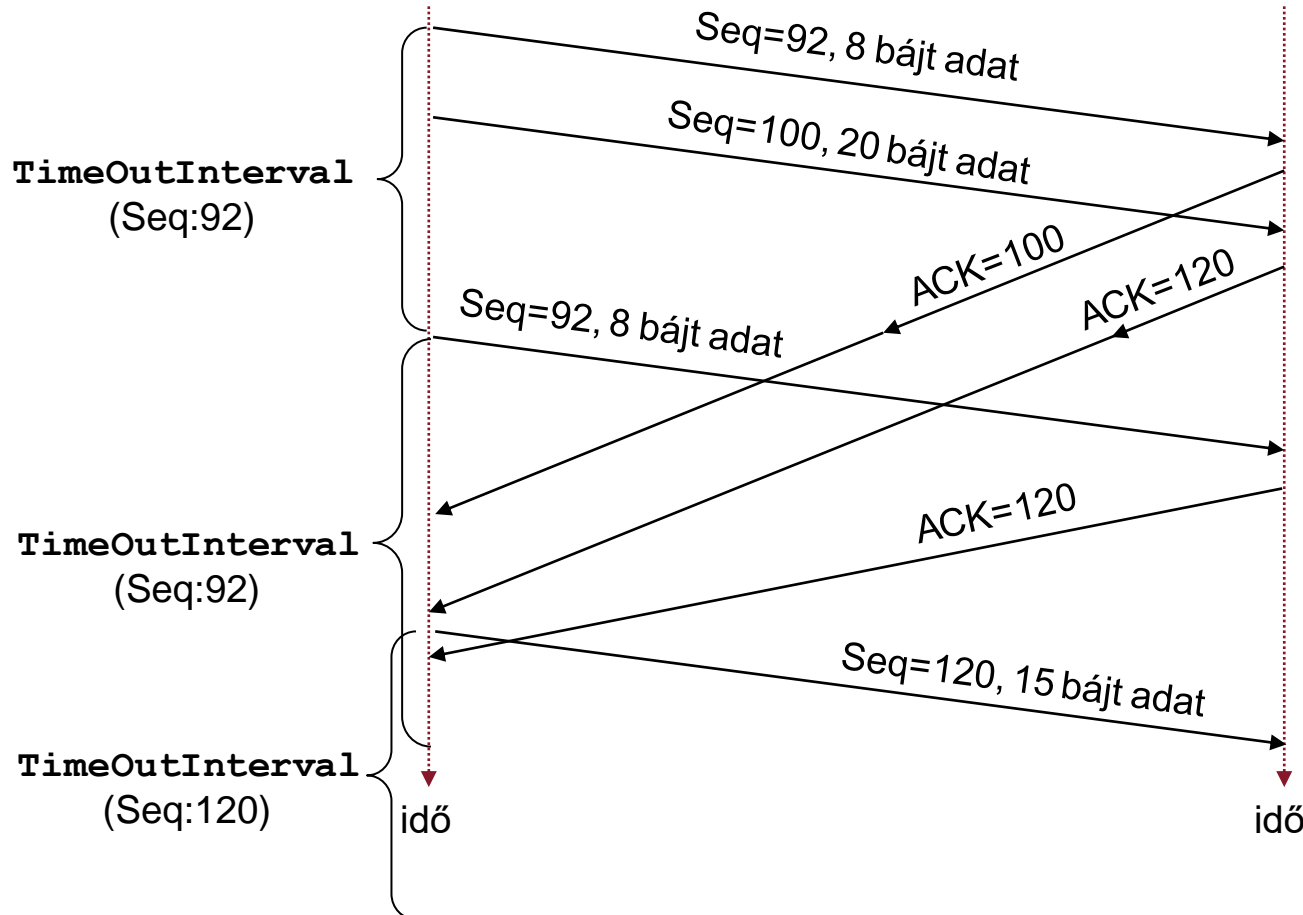
- Timeout értéke

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$$

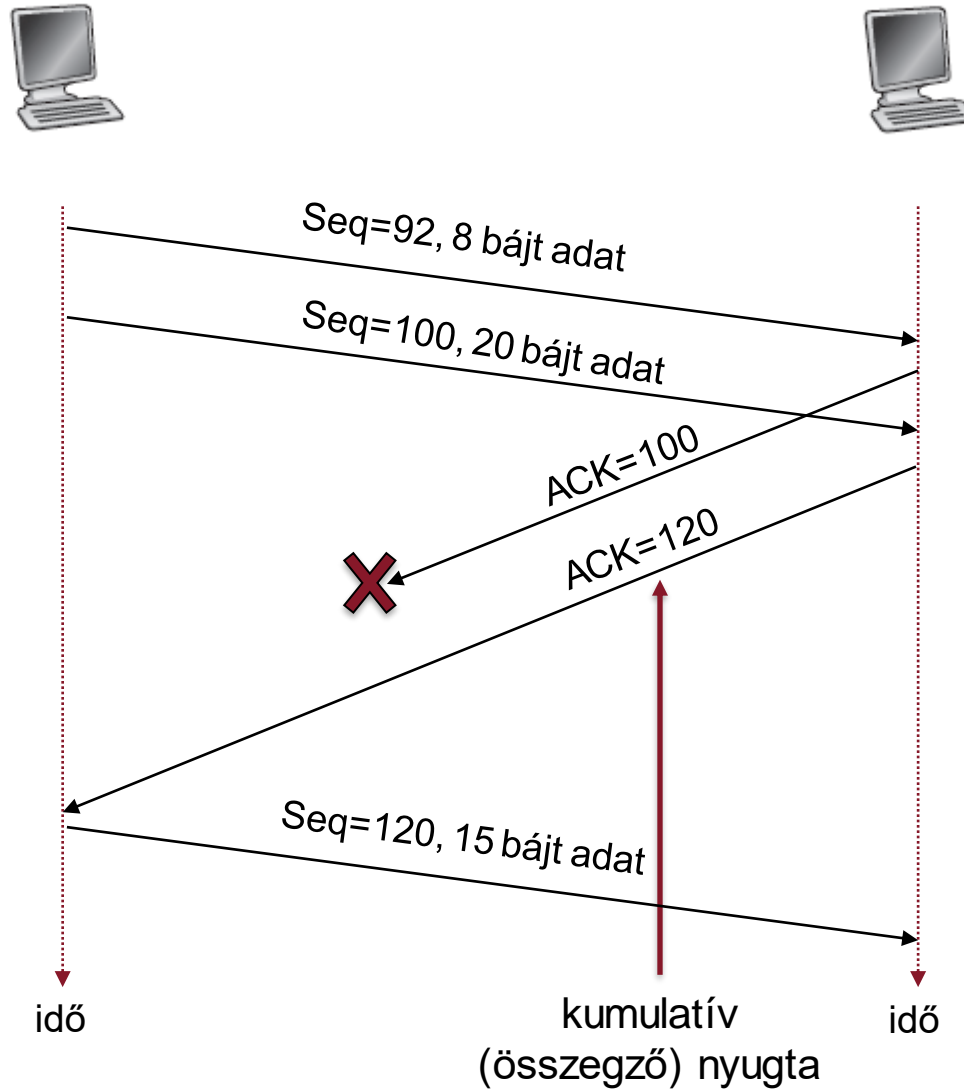
- A TCP a megbízhatatlan IP szolgáltatása felett biztosít megbízható átvitelt
- Szegmensek küldése pipeline módban
 - Nem kell bevárni az előző küldött szegmens nyugtáját
 - A vevő úgysem tudja mikor küldtük el, csak minden megérkezzen (jó esetben sorrendben)
- Összegző nyugta
- Egyetlen időmérési mechanizmus az újraküldéshez
- Újraküldést kiváltó okok
 - Időtűllépés
 - **Kettőzött nyugta (duplicate ack)**
- Vizsgáljunk meg először egy egyszerűsített esetet
 - Nincs kettőzött nyugta
 - Nincs forgalomvezérlés
 - Nincs torlódáskezelés

- Megkapja az alkalmazás adatait
 - Szegmens képzése MSS-nek megfelelően
 - Az első bájt sorszáma kerül a Seq mezőbe
 - Elindítja az időzítőt, ha még nem fut
 - Legjobb lenne a legrégebb, még nem nyugtázott szegmens idejét mérni
 - Az időzítő lejárt: **TimeoutInterval**
- Időtűllépés
 - Az időtűllépést okozó szegmens újraküldése
 - Időzítő újraindítása
- Nyugta érkezik
 - Ha még nem nyugtázott szegmensre vonatkozik
 - A már nyugtázott szegmensek halmazának frissítése
 - Időzítő újraindítása, ha van még nyugtázatlan szegmens





KUMULATÍV NYUGTA CSOMAGVESZTÉS MIATT



• Esemény a vevőnél

- Helyes sorrendben érkező, elvárt (nem túl nagy) sorszámú szegmens érkezik és minden addigi szegmens nyugtázva van

- Helyes sorrendben érkező, elvárt (nem túl nagy) sorszámú szegmens érkezik és van korábbi nem nyugtázott szegmens

- A sorrendből kilógó, túl nagy sorszámú szegmens érkezik

- Lyukat betömő szegmens érkezik

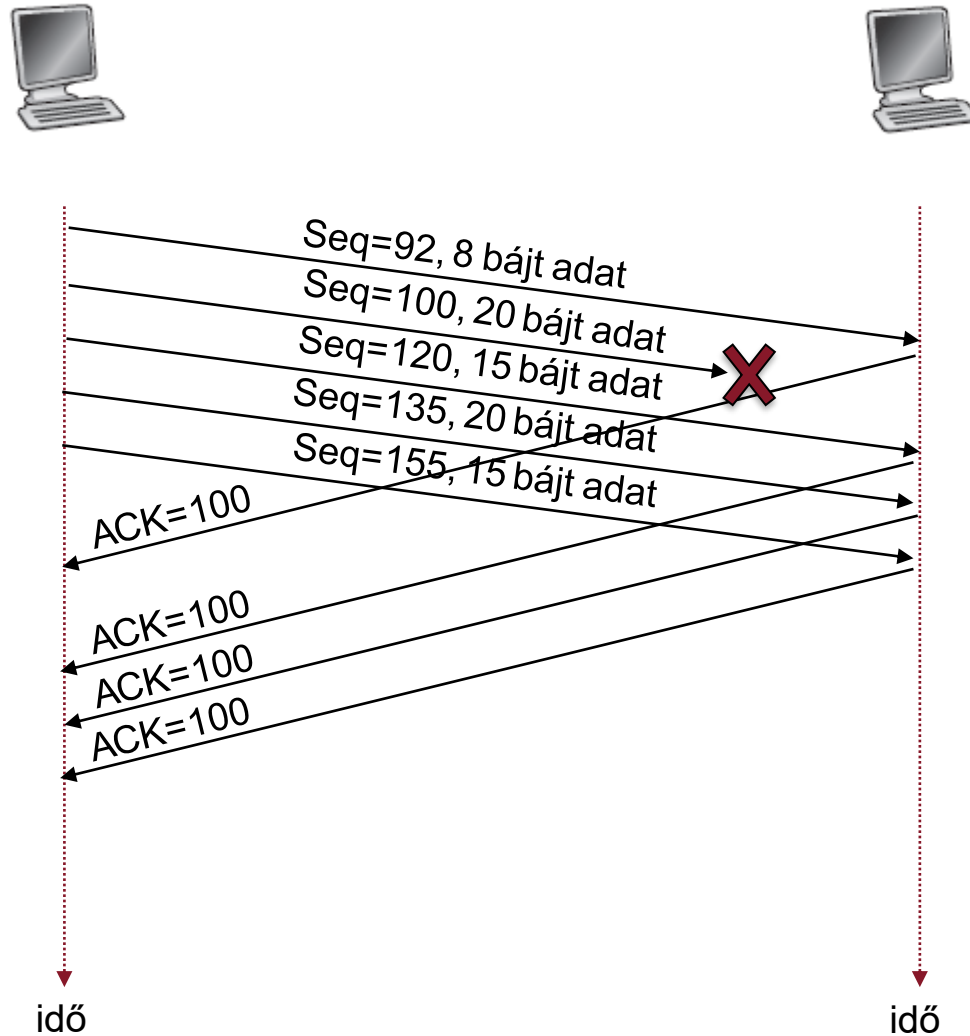
• Akció

- Késleltetett nyugtázás (delayed ACK), legfeljebb 500ms-t vár a következő szegmensre és utána nyugtáz

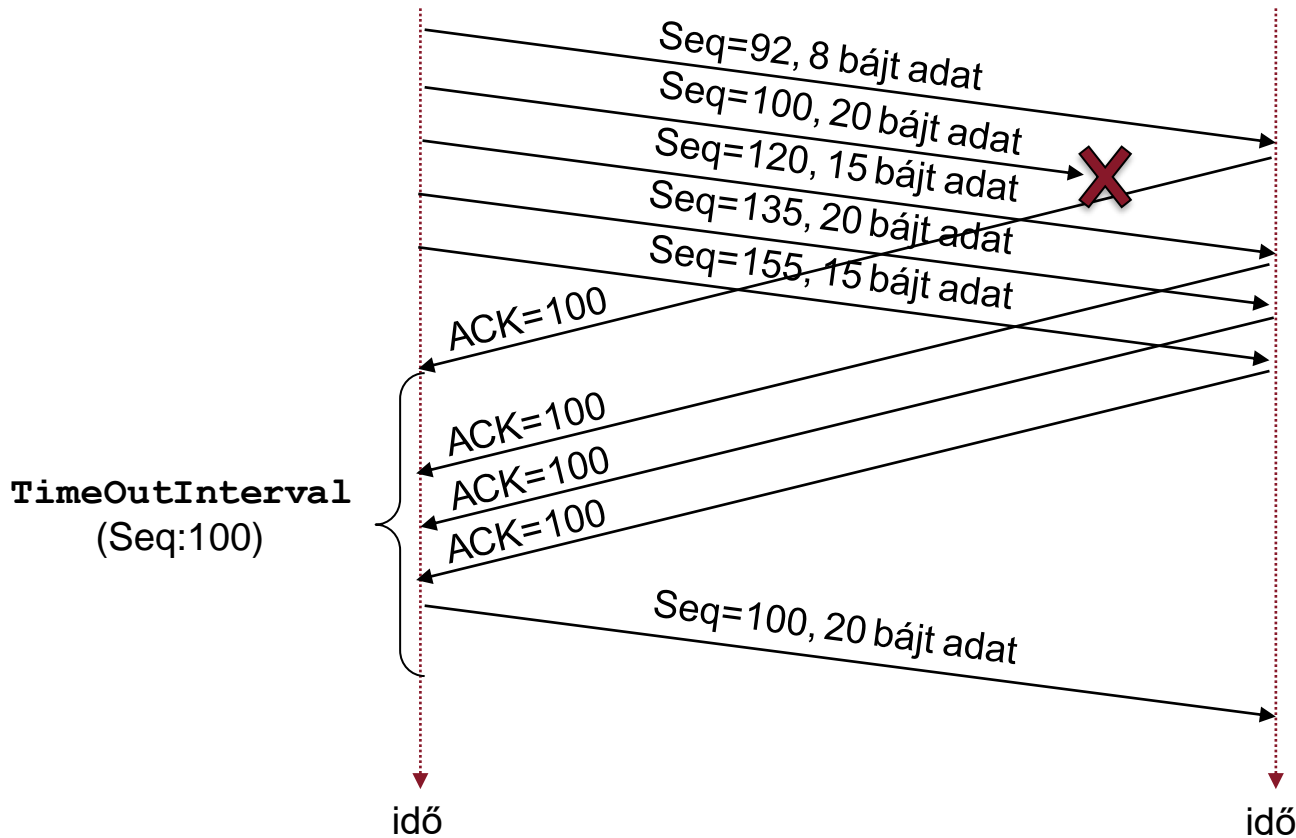
- Azonnal kumulatív nyugtát küld a szegmensekre

- Azonnal kettőzött (ismételt) nyugtát küld a hiányzó szegmenst megadva az ACK-ban

- Azonnal nyugtát küld a legrégebb megmaradt lyuk elejét megadva az ACK-ban

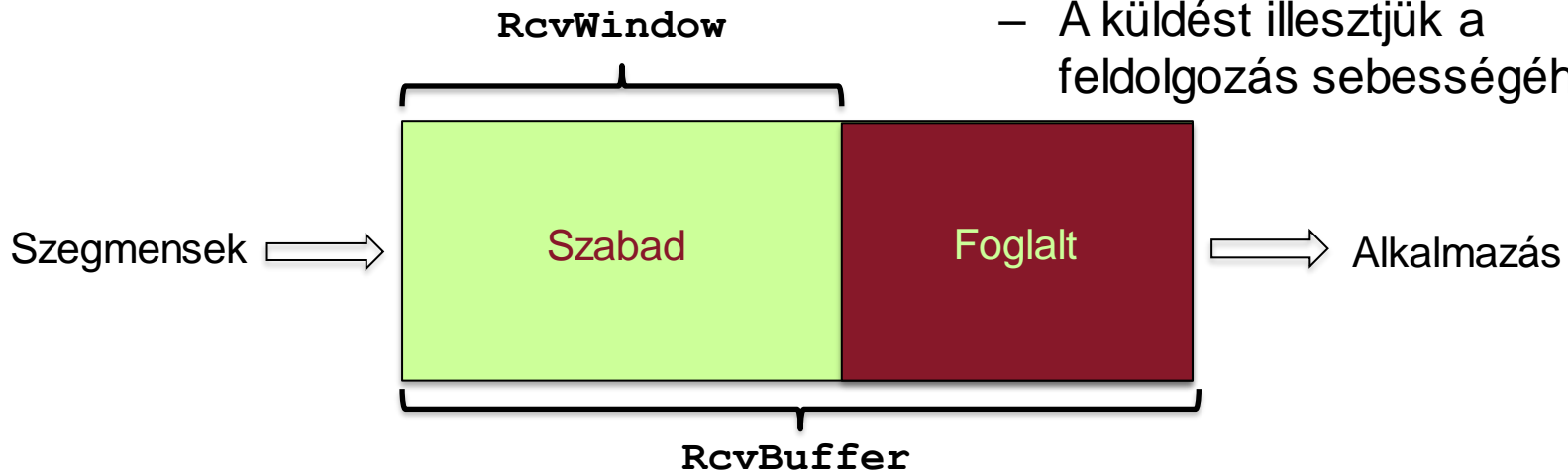


- A Timeout gyakran indokolatlanul nagy
 - Túl sokat várunk mielőtt újraküldenénk
- Csomagvesztés felismerhető duplikált (kettőzött, többszörös) nyugták alapján is
 - A küldő sok szegmenst küld el egymás után
 - Egy szegmens elveszése sok duplikált nyugtát indukál
- A harmadik duplikált nyugta után feltehetjük, hogy csomagvesztés volt
 - Gyors újraküldés (Fast Retransmit)
 - Szegmens újraküldése időtűllépés nélkül
 - (Nem biztos, hogy tényleg volt rá szükség)

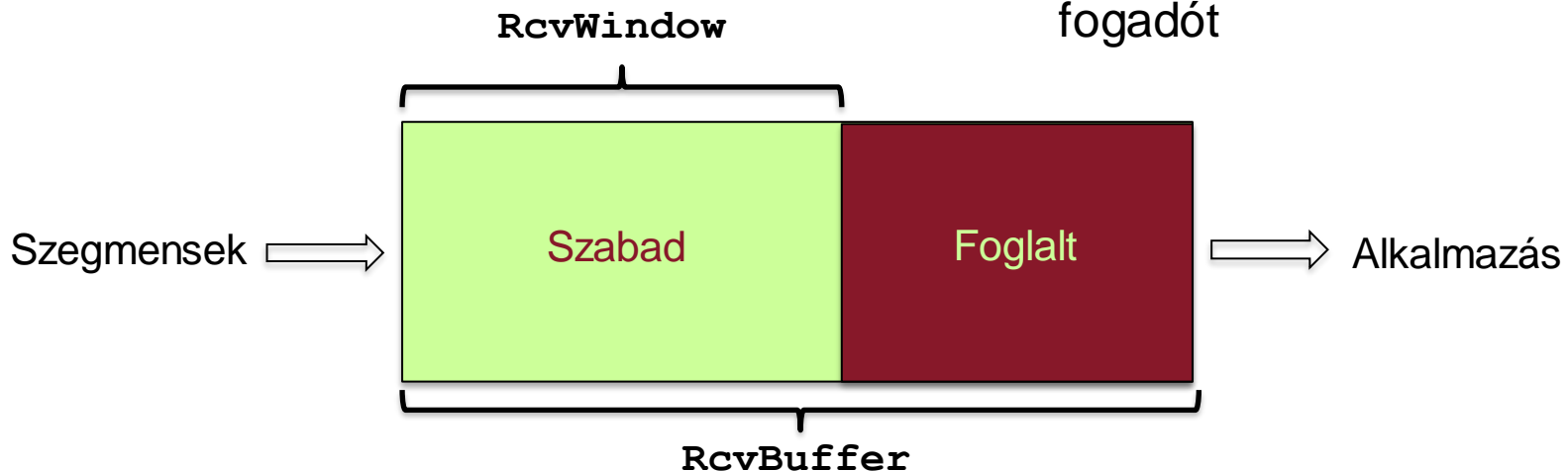


1. A TCP protokoll
2. Szegmensek nyugtázása
3. **Forgalomszabályozás**
4. Kapcsolatkezelés

- A vevő oldalra érkező adatok egy véges méretű pufferbe kerülnek
- Az alkalmazás estenként lassan üríti a puffert
- Forgalomszabályozás (**flow control**)
 - Megakadályozza, hogy a küldő **túltöltse** a puffert
 - Túl gyorsan küld
 - Túl sokat küld
- **Sebességillesztési** szolgáltatás
 - A küldést illesztjük a feldolgozás sebességéhez



- Egyszerűsítés: tekintsünk el a nem sorrendben érkező szegmensektől
- Szabad hely a pufferben
= **RcvWindow**
= **RcvBuffer** -
[**LastByteRcvd** -
LastByteRead]
- A fogadó jelzi a szabad hely mennyiségét a (nyugta)fejléc **RcvWindow** mezőjében
- A küldőnek legfeljebb **RcvWindow** mennyiségű elküldött, de nyugtázatlan adata lehet
 - Így biztosan nem tölti túl a fogadót



1. A TCP protokoll
2. Szegmensek nyugtázása
3. Forgalomszabályozás
4. **Kapcsolatkezelés**

- A TCP kapcsolatorientált

- Kapcsolatfelépítés az adatok küldése előtt
- TCP-változók beállítása
 - Sorszám inicializálás
 - Forgalm szabályozás inicializálása
- Kliens kezdeményez

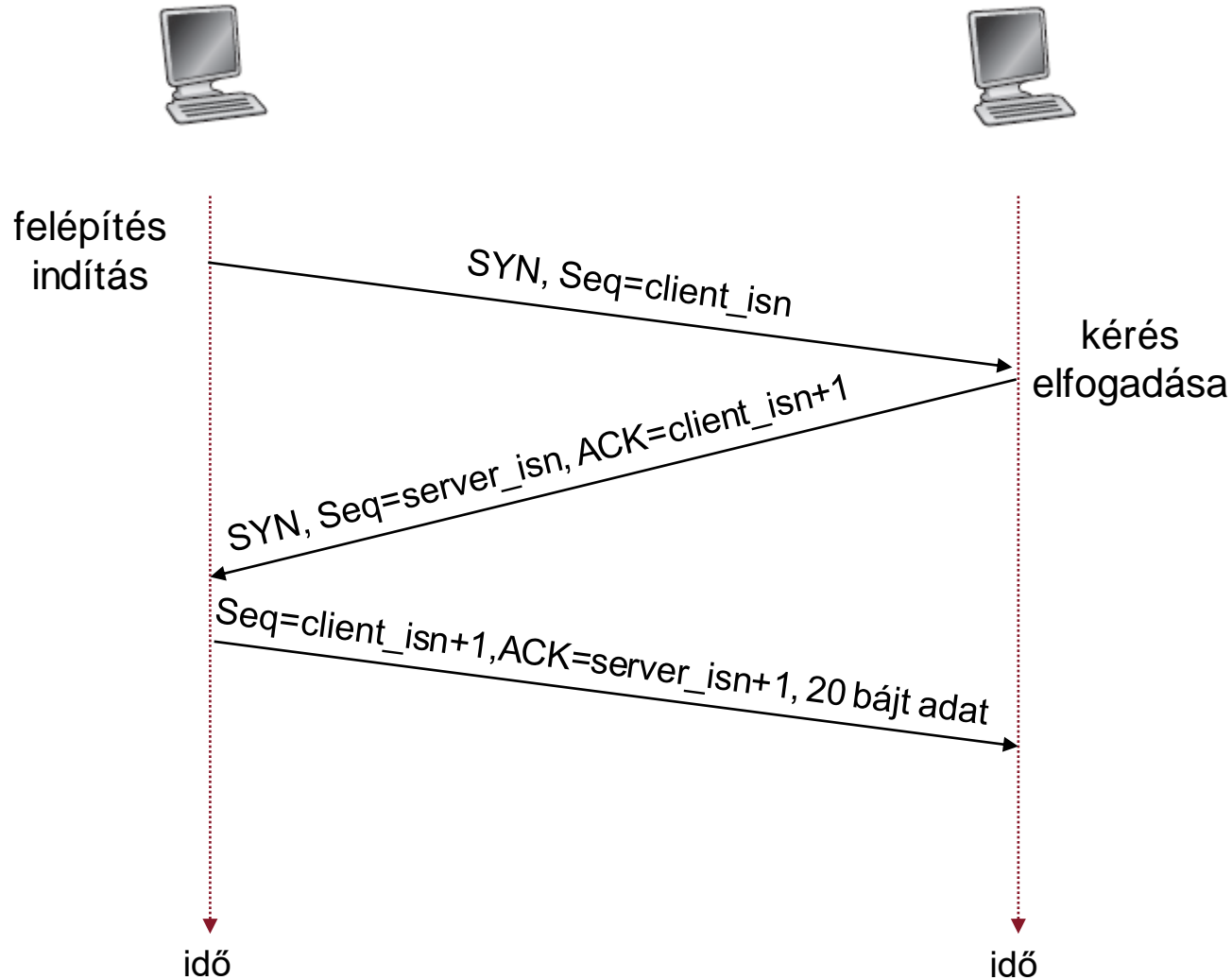
```
clientsock = socket.socket( )
clientsock.connect( )
```

- Szerver fogadja a megkeresést

```
connsock = servsock.accept( )
```

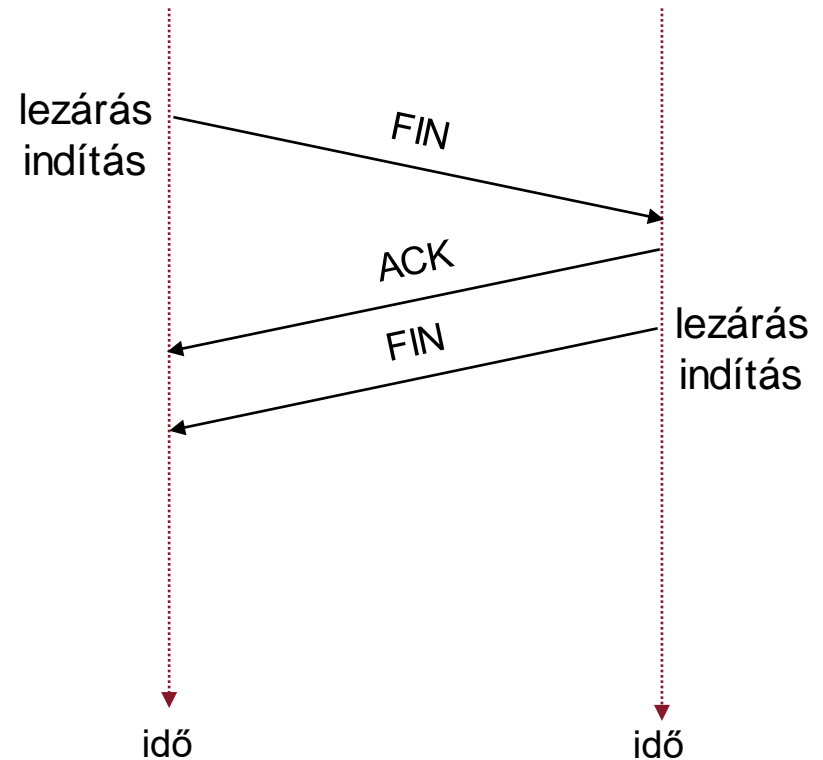
- Kapcsolatfelépítés: háromutas kézfogás (three-way-handshake)

1. A kliens SYN típusú vezérlő szegmenst küld a szervernek
 - Kezdő sorszám küldése
 - Adat nélkül
2. A szerver válaszol egy SYNACK típusú vezérlő szegmennel
 - Inicializáló lépések ezen az oldalon is
3. A kliens válaszol egy ACK szegmennel, ami már adatot is tartalmaz

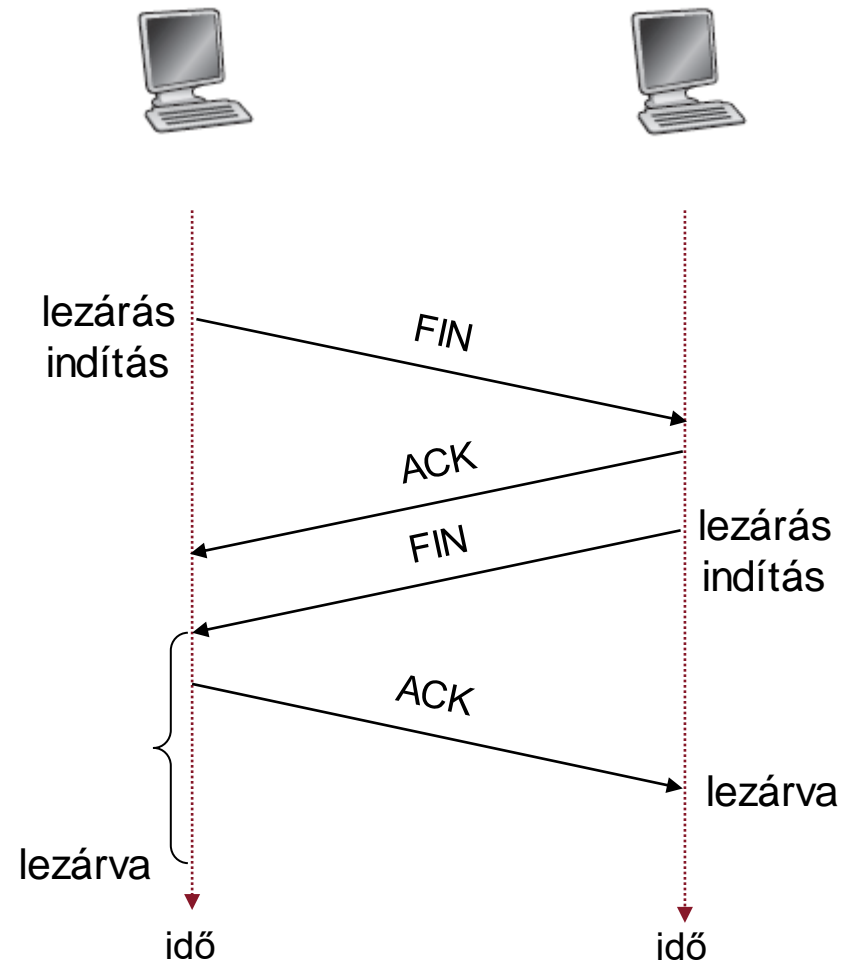


- Kapcsolatlezárás: négyutas kézfogás (four-way-handshake)
 - Tipikusan a kliens kezdeményezi a socketjén keresztül `clientsock.close()`

1. A kliens indítja a lezárást és FIN típusú vezérlő szegmenst küld a szervernek
2. A szerver válaszol egy ACK szegmenssel
3. A szerver FIN típusú vezérlő szegmenst küld a kliensnek és indítja a lezárást



4. A kliens megkapja FIN típusú vezérlő szegmenst és időzített várakozás állapotba lép
5. A kliens válaszol egy ACK szegmensen
6. A szerver megkapja az ACK-ot és lezártnak tekinti a kapcsolatot
7. A kliensnél lejár az időzítő és lezártnak tekinti a kapcsolatot





HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK

