HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK

HÁLÓZATOK ALAPJAI ÉS ÜZEMELTETÉSE

TCP torlódáskezelés, QUIC

2023. március 24.

Mészáros András

BME Hálózati Rendszerek és Szolgáltatások Tanszék

meszarosa@hit.bme.hu

MŰEGYETEM 1782

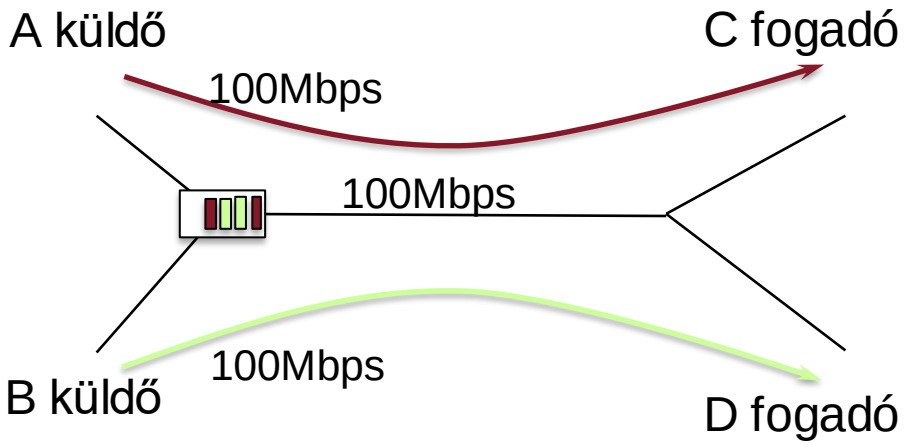
MA MIRŐL LESZ SZÓ?

1. Torlódáskezelés a TCP-ben
2. TCP átbocsátás és igazságosság
3. További TCP funkciók
4. A QUIC protokoll

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

A TÚLTERHELÉS PROBLÉMÁJA

- A probléma okai:
  - Minden küldő a lehető legnagyobb sávszélességgel küldi az adatokat
  - A küldők összes forgalma meghaladja egy link átbocsátó kapacitását
- **Végtelen kimeneti puffer** esetén
  - A csomagok várakozási ideje végtelen nagyra nőhet
  - A lejáró időzítők miatti újraküldések csak súlyosbítanak
- **Véges kimeneti puffer** esetén
  - A várakozási idő korlátos (de nagy lehet)
  - Csomagvesztés lép fel
  - Újraküldés veszteség miatt is
- A csomagok **torlódása (congestion)**
  - Jó lenne elkerülni
  - A küldő oldal tud érte tenni



TCP TORLÓDÁSKEZELÉS

- A TCP-ben kihasználhatók a megbízható szállításra használt mechanizmusok
- A **végpont-végpont** forgalom információi
  - Nem a hálózati eszközök jelzik a torlódást
  - Késleltetési információk a nyugták alapján
  - Csomagvesztési (szegmensvesztési) információk a nyugták hiánya vagy késése alapján
  - Nem tudni, hogy pontosan hol van torlódás (akár több helyen is lehet)
- Gyorsan kell cselekedni, ezért egyszerű mechanizmus kell
- Sebességillesztési szolgáltatás
  - Nem a fogadóhoz, hanem **a hálózat átbocsátóképességéhez illeszkedik**

A TORLÓDÁSKEZELÉS RÉSZLETEI

- A következő nyugtáig átküldhető adatok mennyiségét korlátozó ablak: torlódási ablak, **`CongWin`**
- A küldő betartja az alábbi szabályt:

  `LastByteSent-LastByteAcked ≤ CongWin`
- Átlagos sávszélesség (megközelítőleg)

  `CongWin / RTT`
- Megközelítés:
  - Ha minden rendben – növeljük az ablakot
  - Ha baj (csomagvesztés) van – csökkentjük az ablakot
- A torlódási ablak dinamikusan követi a hálózat állapotát

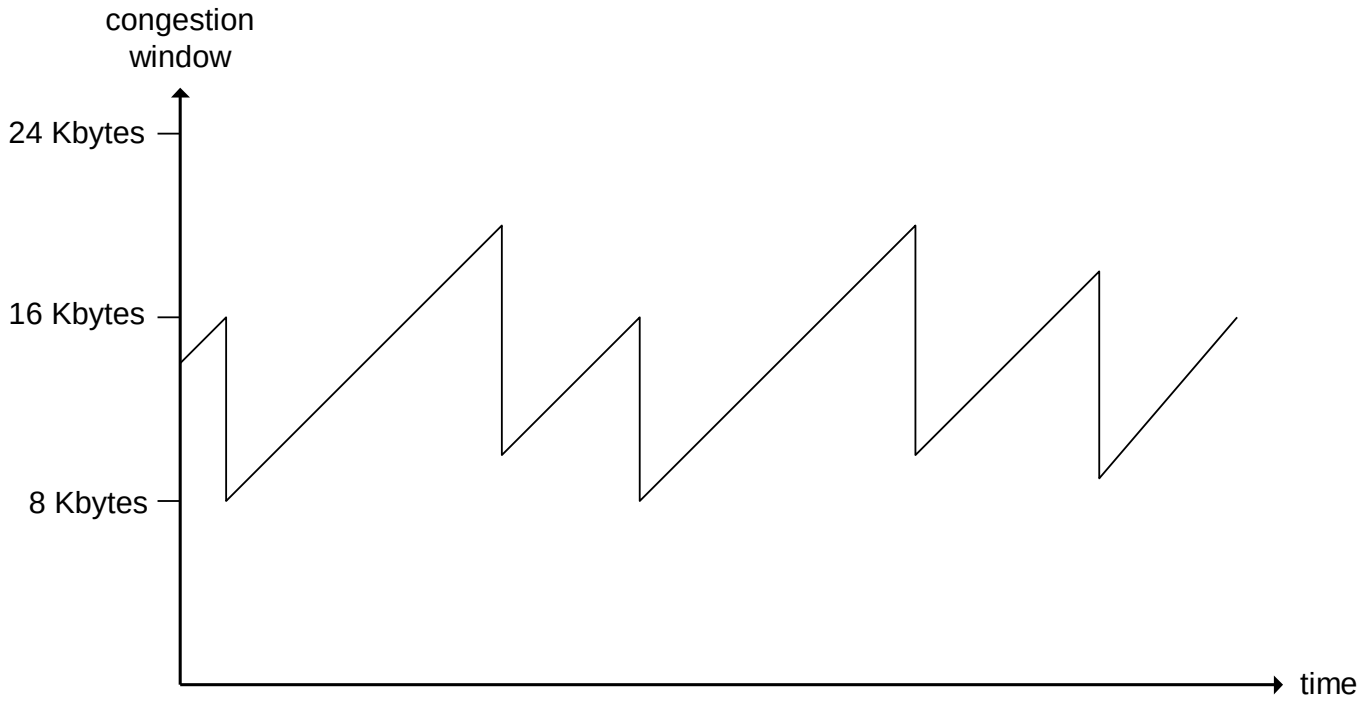
A TORLÓDÁSI ABLAK VÁLTOZTATÁSA

- Minden rendben = nem duplikált nyugta
- Torlódás érzékelése a küldőnél – (feltételezett) csomagvesztés
  - Időtúllépés nyugta várása közben
  - Három duplikált nyugta érkezése
- Az ablakot szabályozó mechanizmusok
  - **Lassú indulás (Slow Start, SS)**
  - **Torlódás-elkerülés (Congestion Avoidance, CA)** – AIMD
  - Időtúllépés esetén nagyon konzervatív

AIMD – ADDITÍV NÖVELÉS, MULTIPLIKATÍV CSÖKKENTÉS

- Additív növelés (increase)
  - Lineáris növelés: egy MSS-sel növeljük az ablakot minden RTT-ben
- Multiplikatív csökkentés (decrease)
  - Exponenciális csökkentés: a felére csökkentjük az ablakot ha elveszett egy csomag
- Fűrészfogszerű viselkedés
- A cél a maximális, még átvihető sávszélességgel küldeni
- Torlódás esetén erőteljes beavatkozás

AIMD ILLUSZTRÁCIÓ



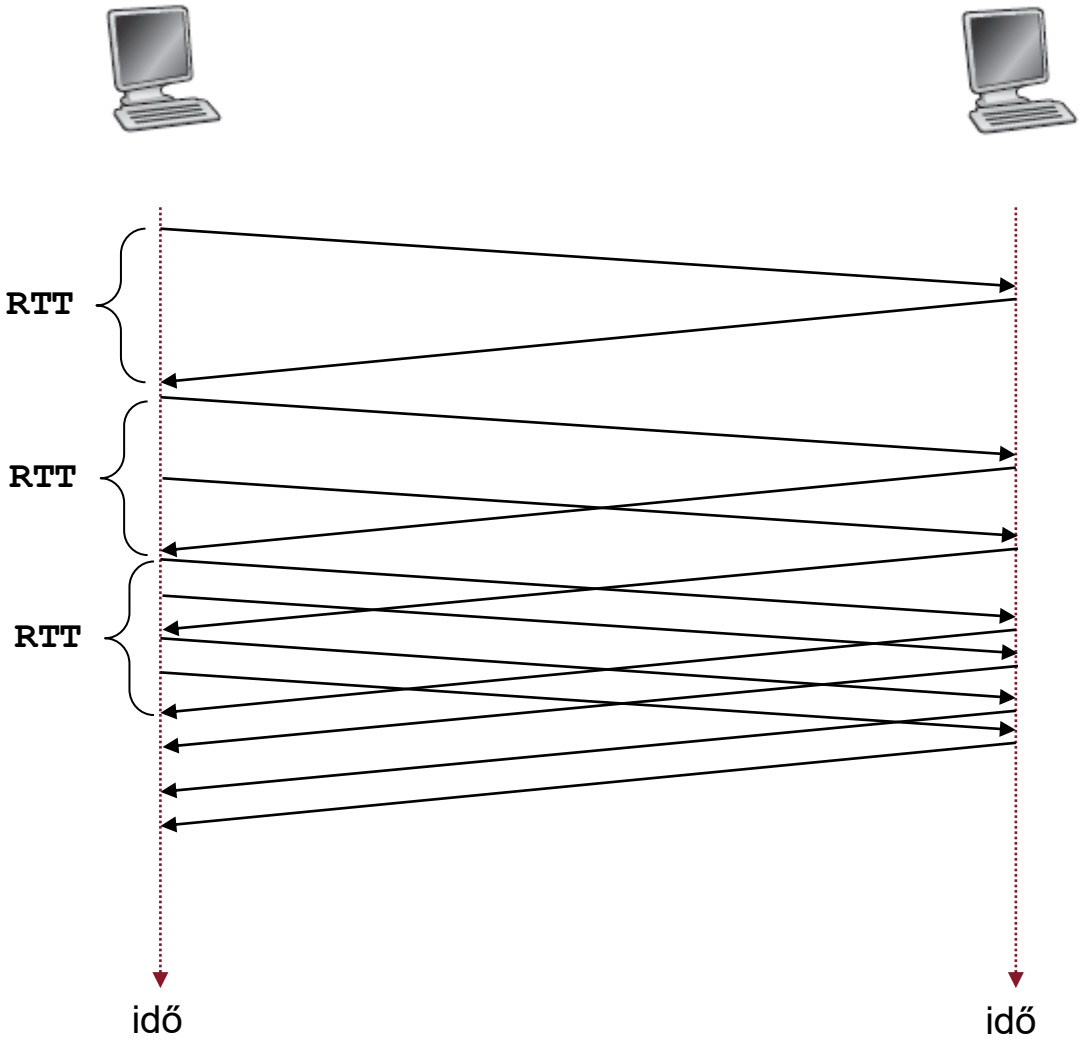
LASSÚ INDULÁS

- A kapcsolat kezdetén

  `CongWin = 1 MSS`

  - A tényleges átbocsátóképességnél jóval kisebb lehet
  - Például
    - MSS: 500 bájt
    - RTT: 20 ms
    - Torlódási ablak: 500 bájt
    - Kezdeti sávszélesség: 200kbps
- Gyorsan fel kellene futtatni
  - Exponenciális növelés
  - Amíg nem lép fel csomagvesztés
- Minden nyugtaérkezéskor egy MSS-sel növeljük az ablakot
- Minden RTT alatt megkétszereződik (megközelítőleg)

LASSÚ INDULÁS – ILLUSZTRÁCIÓ

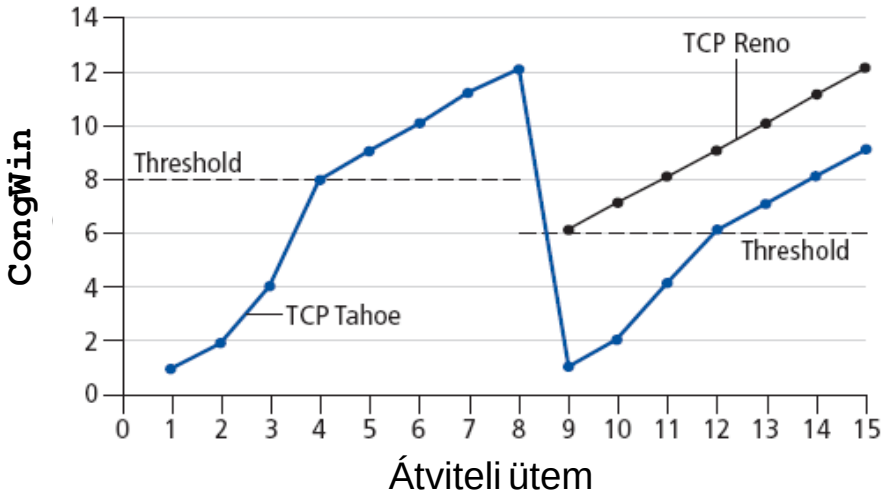


REAGÁLÁS CSOMAGVESZTÉSRE

- Három duplikált nyugta esetén
  - A hálózaton még átküldhetők csomagok
  - Fast Retransmit újraküldés
  - Felezve az ablakot folytathatjuk additív növekedéssel (CA)
- Időtúllépés után
  - Nagyobb a baj, nem jönnek nyugták
  - Kezdjük elölről, lassú indítással (SS)
- TCP változatok
  - Tahoe – nem tesz különbséget, mindig SS-sel folytatja
  - Reno – hatékonyabb a torlódáskezelés
  - További változatok (NewReno, SACK, FACK, Vegas, CUBIC, stb.) – tovább csiszolnak bizonyos részeken, pl. más függvényeket használnak
  - Nem csak a torlódáskezelésben különbözhetnek, extra opciókat is használnak

CSOMAGVESZTÉS UTÁNI SS

- Mikor kellene az exponenciális növekedésből lineáris növekedésre átállni?
  - Az első vesztés helyett egy korábban beállított küszöbnél
  - Amikor elérjük a legutolsó vesztéssel végződő ablak felét
- Megvalósítás
  - **`Threshold`** változó
  - Csomagvesztés után az ablak felére állítjuk be



TORLÓDÁSKEZELÉS ÖSSZEFOGLALVA

- Ha **`CongWin`** kisebb mint **`Threshold`** akkor Slow Start fázis, exponenciális ablaknövelés
- Ha **`CongWin`** nagyobb mint **`Threshold`** akkor Congestion Avoidance fázis, lineáris ablaknövelés
- Miután három duplikált nyugta jön, a **`Threshold`** és a **`CongWin`** változót is **`CongWin/2`** –re állítjuk
- Ha időtúllépés (timeout) lép fel, a **`Threshold`** változót **`CongWin/2`** –re, a **`CongWin`** változót 1 **`MSS`** –re állítjuk

MA MIRŐL LESZ SZÓ?

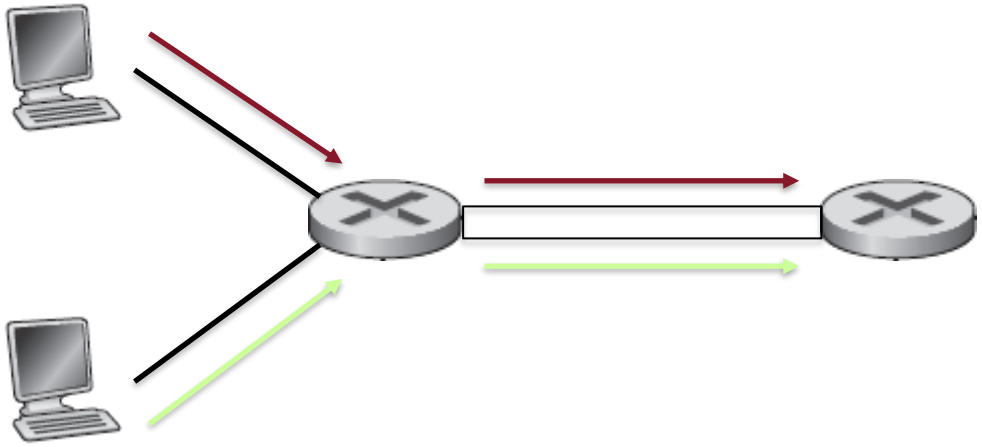
1. Torlódáskezelés a TCP-ben
2. TCP átbocsátás és igazságosság
3. További TCP funkciók
4. A QUIC protokoll

TCP ÁTVITELI SÁVSZÉLESSÉG

- Mekkora átvitel (átbocsátás) adódik ebből a működésből?
  - Figyelmen kívül hagyva az SS állapotot
- Tegyük fel, hogy W ablakméretnél lép fel csomagvesztés
  - W méretnél az átvitel W/RTT
  - A felezés után W/(2*RTT) –től lineárisan nő
  - Az átlag 0.75W/RTT
- Ez nagyon leegyszerűsített modell volt
  - Valójában a hálózat állapotától függ a W és az RTT is
  - A TCP mohón, de szabályozottan fogyasztja a link-kapacitásokat

IGAZSÁGOSSÁG

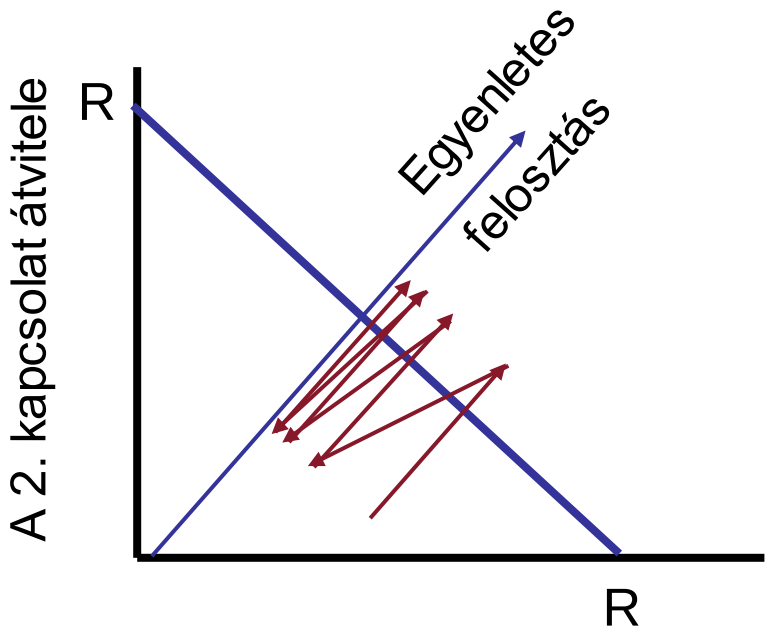
- A kapcsolatok osztoznak a hálózati erőforrásokon
- Az lenne **fair**, ha átlagban „testvériesen” osztoznának (**fairness**)
  - Egyenletesen?
  - Az átviendő adatmennyiség arányában?



TCP FAIRNESS

- Annak, hogy csökkentjük is néha az ablakméretet, a fairness is a célja
- Példa: két kapcsolat forgalma versenyhelyzetben egy R kapacitású linken
  - Additív ablaknövelés sikeres átvitel esetén
  - Multiplikatív ablakcsökkentés torlódás esetén

CA: ablakméret lineáris növelése
Csomagvesztés: ablakfelezés
CA: ablakméret lineáris növelése
Csomagvesztés: ablakfelezés
CA: ablakméret lineáris növelése



Az 1. kapcsolat átvitele

- Sok kapcsolat esetén azért nem ennyire egyszerű

FAIRNESS MÁS PROTOKOLLOKBAN

- UDP
  - A multimédia alkalmazások gyakran nem TCP alapúak
    - Nem nagy baj, ha nem megbízható az átvitel
    - Nem praktikus a torlódáskezelés
  - Konstans sávszélességgel lehet küldeni
  - Veszteség lehet
  - Nem igazán barátságos
- Vannak ötletek torlódáskezelés nélküli megbízható protokollokra is
- TCP torlódáskezelés kihasználása magasabb rétegben
  - Több párhuzamos kapcsolat is nyitható egy applikációban
    - Például web-böngészők
  - Példa: egy R kapacitású linken 9 kapcsolat osztozik
    - Egy újabb applikáció egy kapcsolatot nyit: R/10 átbocsátás jut neki
    - Egy újabb applikáció 9 kapcsolatot nyit: R/2 átbocsátás jut neki

MA MIRŐL LESZ SZÓ?

1. Torlódáskezelés a TCP-ben
2. TCP átbocsátás és igazságosság
3. További TCP funkciók
4. A QUIC protokoll

SZELEKTÍV NYUGTÁZÁS

- Egy elveszett szegmens miatt duplikált nyugták jöhetnek
  - Csak azt adják meg, hogy mi az első hiányzó bájt
  - A sorrenden kívüli, de **megkapott szegmenseket** nem kellene újraküldeni
  - Nagy RTT esetén nem kapjuk meg időben a kumulatív nyugtát
- **SACK** – a fogadó nyugtában küldi vissza a sorrenden kívüli bájtok tartományait
  - Korlátozott mennyiségben
  - A fejléc opcióit felhasználva
  - A kapcsolat felépítésekor egyeztetnek a felek a használatról
- Az újraküldést önmagában is gyorsíthatja

TAIL LOSS PROBE (TLP)

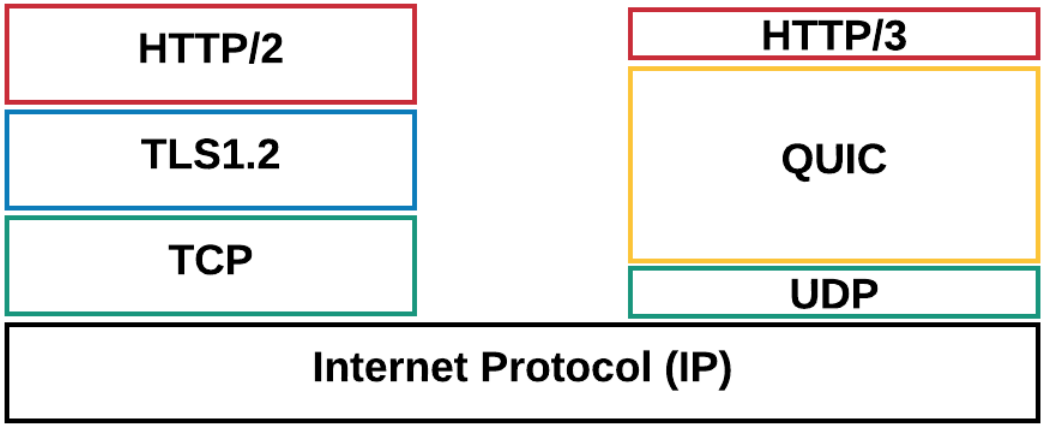
- Sok szegmens küldése esetén
  - Duplikált nyugták alapján látjuk , hogy melyik szegmenst nem kapta meg a fogadó
  - Gyors újraküldés
- Egy szegmens esetén?
  - Többnyire időtúllépés lesz
- Ötlet
  - Próbáljuk ki, hogy nincs-e csomagvesztés
  - Kis mennyiségű adat újraküldése még az időtúllépés előtt
  - Szelektív nyugtázás adataira támaszkodik

MA MIRŐL LESZ SZÓ?

1. Torlódáskezelés a TCP-ben
2. TCP átbocsátás és igazságosság
3. További TCP funkciók
4. A QUIC protokoll

QUIC ALAPOK

- A Google-nél kezdték fejleszteni - RFC 9000
- A TCP változatok előnyeit ötvözi
- **Quick UDP Internet Connections**
  - Megbízható adatátvitel UDP felett?
  - Egy fél réteggel feljebb (szállítási és alkalmazási réteg között)
  - Egyelőre nem a TCP helyett, csak mellett
- A HTTP/2 és a HTTP/3 hatékony kiszolgálására

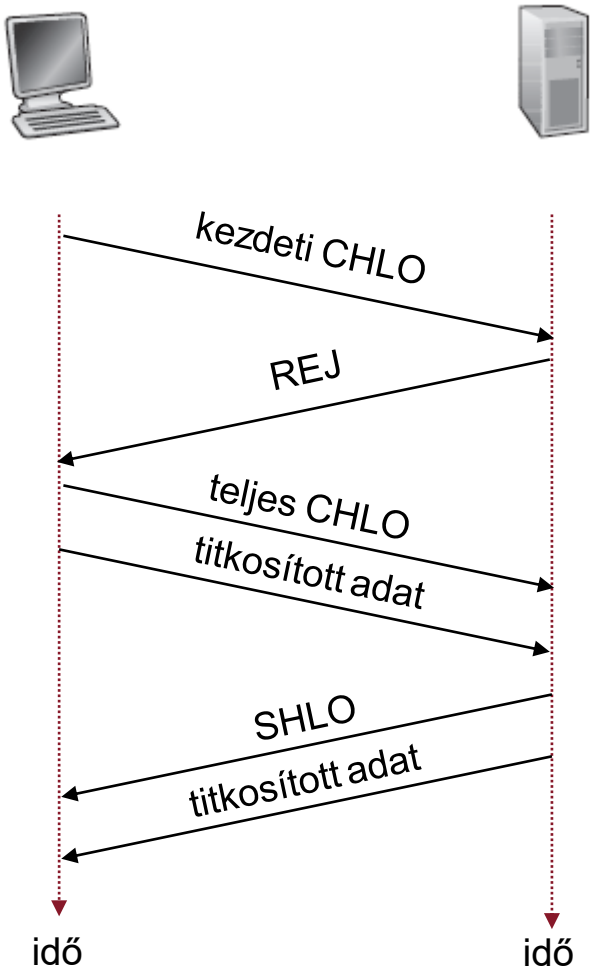


A QUIC FŐ ELEMEI

- **Alkalmazási rétegben** implementálva
- **Gyors** kapcsolat-felépítés (titkosítás is)
- **Hatékony** újraküldés és torlódáskezelés
- Adatfolyamok (stream) **multiplexálása**
  - Egy adat nem tartja fel a többit
- Kapcsolatvégpontok **migrálása**
  - Internet értelemben mozoghatnak a végpontok
  - A kapcsolatnak egyedi azonosítója van
  - Nem a TCP-nél látott négyes információ, tehát akár változhat az IP cím, vagy a port száma
- **Titkosítás** beépítve
  - Az összes adat csomagonként titkosítva megy át (TLS)

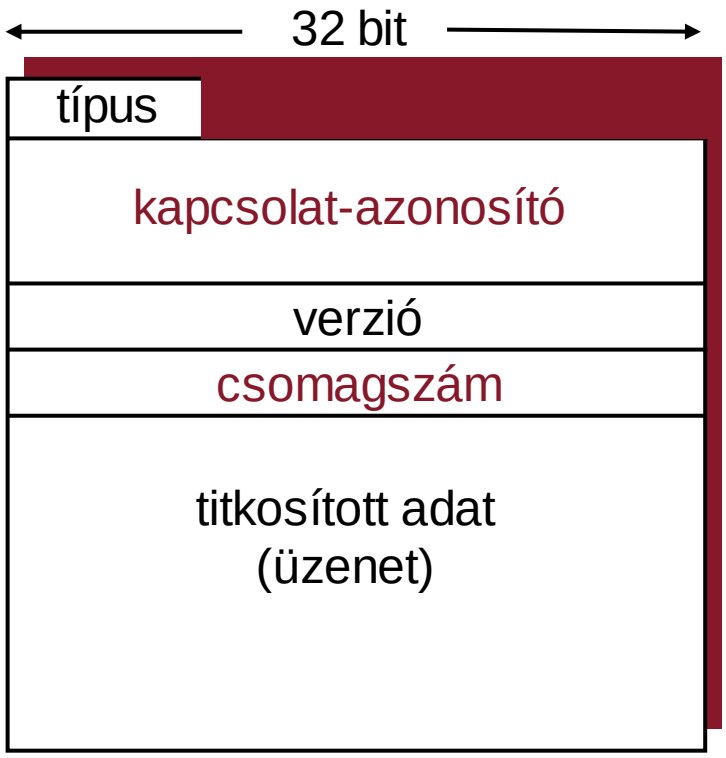
QUIC – KAPCSOLATFELÉPÍTÉS

- Ismeretlen szerver
  - **1-RTT** felépítés
  - CHLO, REJ, SHLO
  - Ebben már a titkosítás kulcskiosztása is benne van
- Ismert szerver
  - **0-RTT** felépítés
  - CHLO/SHLO után egyből küldhető az adat a korábban megismert információk alapján
- Verzió-egyeztetés
  - Kicsit hosszabb lehet
  - Ritkán kell



QUIC - CSOMAGFORMÁTUM

- A csomag az UDP szegmens adata
- Verzióra csak egyeztetéskor van szükség
- **Egyedi csomagszám**
  - Monoton nő, lehet tudni, hogy melyiket küldték előbb
  - Újraküldés esetén már más lesz
- Az adat különböző részekből állhat, pl:
  - Adatfolyamok adatai
  - Nyugta



QUIC – KÜLDÉS ÉS NYUGTÁZÁS

- A küldésnél torlódáskezelést használ
  - TCP CUBIC (Linux és Win10 default)
- A nyugta egy azonosított csomagra vonatkozik
  - SACK
    - A TCP-nél több ACK intervallumot kezel
  - Időbélyeges
    - RTT számítás könnyebb (csomagonként lehet számláló)
- TLP-t is használ
- Probléma: köztes csomópontok UDP-ként érzékelik

  Megoldás fallback to TCP

HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK