Számítógépes Hálózatok 2007

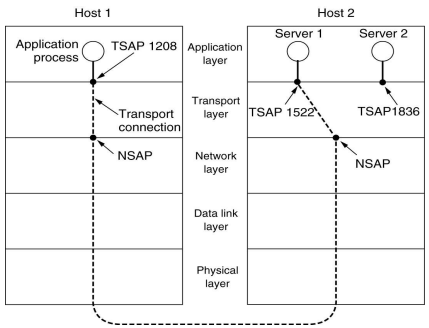
11. Szállítói réteg –
TCP, Tahoe, Reno, AIMD, Fairness, hatékonyság

A szállítói réteg (transport layer) szolgáltatásai

- Kapcsolat nélküli vagy kapcsolat orientált (connectionless/connection oriented)
  - Gondoljunk az ISO/OSI ülés rétegére
- Megbízható vagy nem megbízható (reliable/unreliable)
  - „Best effort” vagy „Quality of Service”
  - Hibafelügyelet
- Torlódás felügyelet (congestion control) vagy torlódás felügyelet nélkül
- Lehetőség több végpontra egy végrendszeren (host)
  - Demultiplexálás
- Több interakciós modell támogatása
  - Byte-áram, üzenetek, „Remote Procedure Call“

Multiplexálás a szállítói rétegben

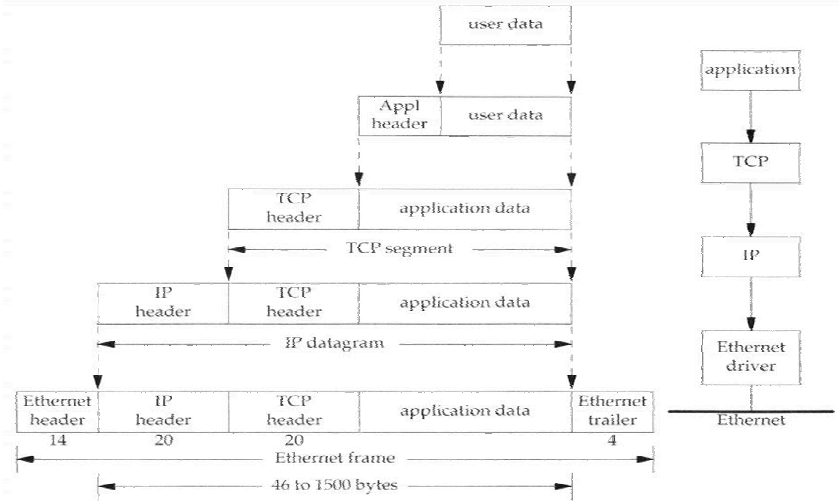
- A hálózati réteg az adatokat kontroll nélkül továbbítja a szállítói rétegnek
- A szállítói rétegnek az adatokat különböző felhasználásokhoz kell hozzárendelni:
  - pl. Web, Mail, FTP, ssh, ...
  - TCP/UDP ezt port-szám alapján teszi
  - pl. port 80 a Web-szerverhez



Szállítói réteg (transport layer)

- TCP (transmission control protocol)
  - Megbízható adatfolyamot hoz létre két végpont között
  - A felhasználói réteg adatáramát csomagokra osztja
  - A másik oldal a csomagok fogadásától nyugtákat küld (Acknowledgment)
- UDP (user datagram protocol)
  - Egyszerű nem megbízható szolgáltatás csomagok küldésére
  - Az inputot egy datagrammá alakítja
  - A felhasználói réteg határozza meg a csomag méretét
- A csomagokat a hálózati réteg által küldi
- Routing nincs: végpont-végpont protokollok

Adatok burkolása



TCP-fejléc (I)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Acknowledgment Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- „Checksum“
  - Fejléchez és adatokhoz
- Fejléchossz (data offset)
  - A változó hosszúságú opció-mező miatt
- Küldő-Port + Cél-Port-Nr.
  - Megenged több TCP-kapcsolatot IP-címenként
- Sorszám
  - Minden adatbyte meg van számozva modulo 2^{32}
  - = a szegmens első byte-jának a száma
- Nyugta szám
  - Az ACK-Flag által aktiviált
  - Az első még nem feldolgozott adatbyte száma
  - = utolsó sorozatszám + utolsó adatmennyiség

TCP-Fejléc (II)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Acknowledgment Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Opció-mező pl. MSS (maximum segement size):
  - A fogadó megadja a kívánt csomagméretet
  - Tekintet nélkül az IP MTU-ra (max. transmission unit)
  - Fragmentálás lehetséges az IP által
- FLAGS (függetlenül felhasználhatók)
  - URG: sürgős (urgent)
  - ACK: nyugta (acknowledgment)
    - Aktiviálja a nyugta számot
  - PSH: Push
    - Gyors adattovábbítás a felhasználói rétegnek
  - RST: Reset
    - A válasz hiba esetén: `connection reset by peer`
  - SYN: Synchronize
    - Kapcsolatfelépítés és a kezdő sorszám megadása
  - FIN: Finished
    - (Egy) adatfolyam befejezése

TCP (I)

TCP (Transmission Control Protocol) egy kapcsolatorientált megbízható szolgáltatás bidirekcionális byte-folyamokhoz

TCP Kapcsolatorientált

- Két résztvevő. Egy egy résztvevő socket által azonosított: socket: IP-cím és port
- TCP-kapcsolat egyértelműen azonosított egy socketpár által
- Nincs broadcast sem multicast
- Kapcsolatfelépítés és lezárás szükséges
- Amíg egy kapcsolat nincs (rendesen) lezárva, addig aktív

TCP (II)

TCP egy kapcsolatorientált megbízható szolgáltatás bidirekcionális byte-folyamokhoz

TCP megbízható

- Minden adatcsomag megérkezését nyugtázza *(acknowledgment)*
- A nem nyugtázott adatcsomagokat újraküldi
- “Checksum” a fejléchez és csomaghoz
- TCP számozza a csomagokat és sorbarendezi a fogadónál
- Törli a duplikált csomagokat

TCP (III)

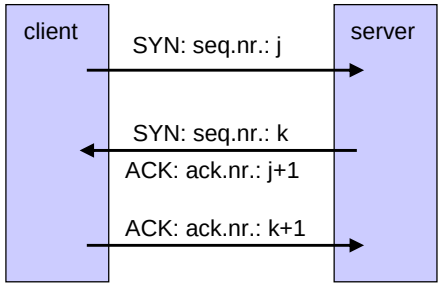
TCP egy kapcsolatorientált megbízható szolgáltatás bidirekcionális byte-folyamokhoz

TCP egy szolgáltatás bidirekcionális byte-folyamokhoz

- Az adatok két egymással ellentétes irányú byte-sorozatként (=8 bit) kerülnek átvitelre
- A tartalom nem interpretálódik
- Az adatcsomagok időbeli viselkedése megváltozhat: átvitel sebessége növekedhet, csökkenhet, más késés, más sorrendben is megérkezhetnek
- Megpróbálja az adatcsomagokat időben egymáshoz közel kiszállítani
- Megpróbálja az átviteli közeget hatékonyan használni
  = kevés csomag

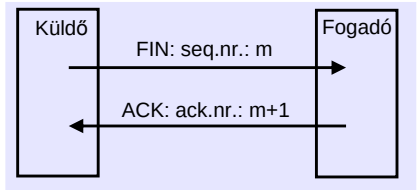
Kapcsolatfelépítés

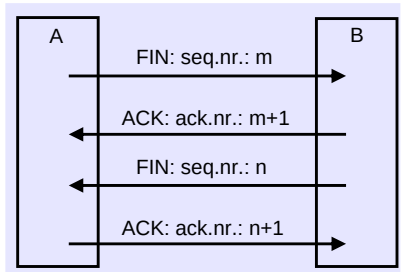
- Rendszerint Client-Server-kapcsolat
  - Ekkor felépítés 3 TCP-csomaggal (=3 szegmens)
  - Az első SYN-szegmensben az MSS (maximum segment size) is átvitelre kerül



Kapcsolat lezárása

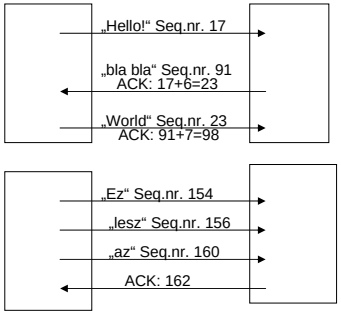
- Félig lezárás (half-close)
  - A küldő jelzi a kapcsolat befejezését egy FIN-szegmensben és vár annak nyugtájára
  - Az ellenkező irányban továbbra is lehet küldeni
- Két félig lezárás lezárja a TCP-kapcsolatot



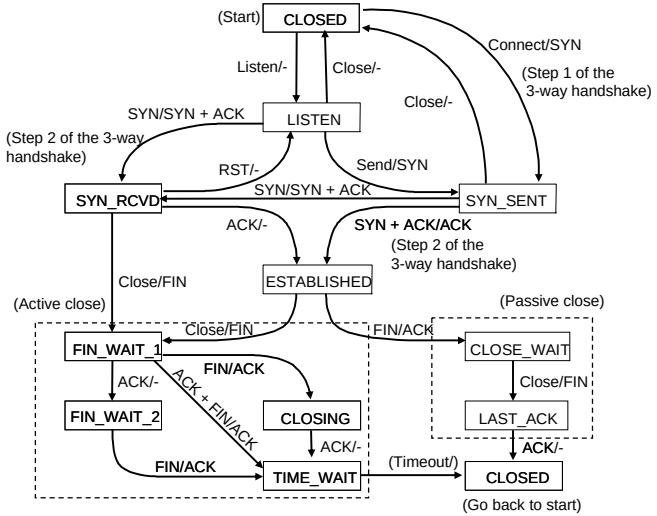


Nyugták (acknowledgement -- ACK)

- Hátizsák technika „piggybacking“
  - A nyugták (ACK) az ellenkező irány adatszegmensein „utaznak“
- Egy nyugta több adatszegmenst is nyugtázhat
  - Ha nincs küldeni való adat, késleteti az ACK-kat



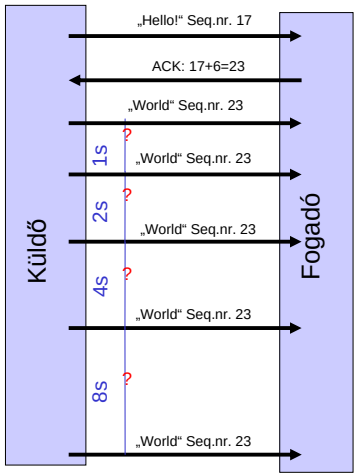
TCP állapot átmeneti diagramm



Exponenciális visszavétel (exponential backoff)

- **Retransmission Timout (RTO)**
  - szabályozza az időközt a küldés és egy duplikátum újraküldése között, ha egy nyugta kimarad
- Mikor nem kerül nyugtázásra egy TCP-csomag?
  - Ha a nyugta lényegesen több időt vesz igénybe, mint az átlagos „round trip time” (RTT)
    - 1. Probléma: RTT mérése
    - 2. Probléma: Csak a nyugta jön túl későn
  - Küldő
    - Vár az RTO-nak megfelő ideig
    - Ha nem érkezett nyugta, újraküldi a csomagot és növeli

      RTO ← 2 RTO (RTO = 64 másodpercig)
- RTO újraszámolása, ha a csomagok nyugtázódnak



A Round Trip Time (RTT) becslése

- A TCP-csomag nem nyugtázottnak számít, ha a nyugta „lényegesen” tovább tart, mint az RTO
- RTT nem számítható on-line (csak visszatekintve)
- RTT erősen ingadozik
- Ezért: Retransmission Timeout Value nagyvonalú becsléssel:
  - RFC 793: (M := utoljára mért RTT)

    $R \leftarrow \alpha R + (1- \alpha) M$, ahol $\alpha = 0,9$

    $RTO \leftarrow \beta R$, ahol $\beta = 2$
  - Jacobson 88: a becslés nem elég robosztus, ezért

    $A \leftarrow A + g (M - A)$, ahol $g = 1/8$

    $D \leftarrow D + h (|M - A| - D)$, ahol $h = 1/4$

    $RTO \leftarrow A + 4D$
- Többszörösen elküldött csomagoknál nem aktualizálunk

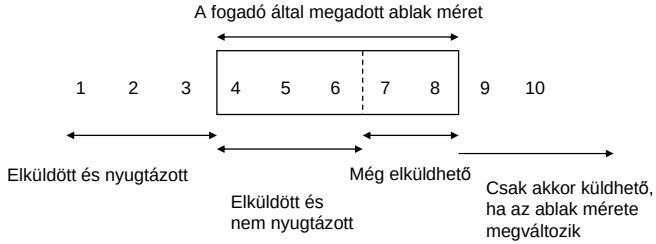
TCP – Nagle algoritmusa

- Hogyan biztosíthatjuk,
  - hogy kis csomagok időben egymáshoz közel kerüljenek kiszállításra
  - és hogy sok adat esetén nagy csomagok előnyben részesüljenek?

- **Nagle algoritmusa:**
  - Kis csomagok nem kerülnek addig küldésre, amig nyugták hiányoznak
    - egy csomag kicsi, ha az adathossz < MSS
  - Ha a korábban küldött csomag nyugtája megérkezik, küldi a következőt
- Tulajdonsága
  - Önmagát ütemező: Gyors kapcsolat = sok kis csomag

Csúszó Ablakok (sliding windows)

- Adatátráta szabályozása ablak segítségével
  - A fogadó meghatározza az ablak méretet (wnd) az ACK-szegmensek TCP-fejlécében
  - Ha a fogadó fogadási puffere tele van, akkor wnd=0 -t küld
  - Máskülönben a fogadó wnd>0 -t küld
- A küldőnek be kell tartani:
  - Az elküldött nem nyugtázott adatcsomagok száma ≤ ablak mérete



Lassú Start (slow start)

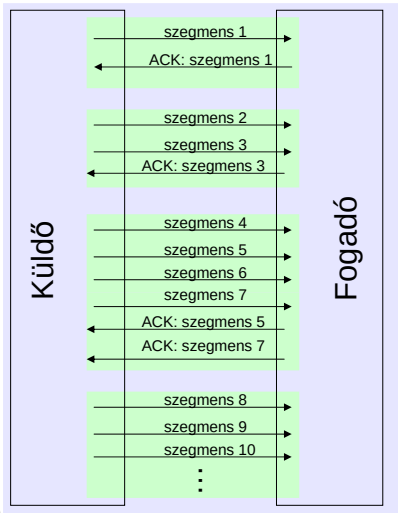
- A küldőnek nem szabad a fogadó által felajánlott ablakméretet azonnal kihasználni
- Második ablak: Congestion-ablak (cwnd: congestion window)
  - A küldő választja
  - Az ablak amiben küld: min {wnd,cwnd}
  - Kezdetben:

    cwnd ← MSS
  - Minden csomagnál a megkapott nyugta után nő

    cwnd ← cwnd + MSS

    (azaz megduplázódik minden RTT után)
  - Addig, amíg egyszer egy nyugta kimarad

Slow start = exponenciális növekedés
(hisztórikus elnevezés: korábban még aggresszívebb sémák)



Torlódás elkerülés (congestion avoidance) TCP Tahoe

- Jacobson 88:
  - Paraméter: cwnd és ssthresh (= slow-start-küszöb, slow start threshold)

1. Kapcsolatfelépítés:

   cwnd ← MSS ssthresh ← 65535
2. Csomagvesztésnél, azaz nyugta ideje > RTO: multiplicatively decreasing

   cwnd ← MSS ssthresh ← $\max\left\{2\ \mathrm{MSS}, \frac{\min\{cwnd, wnd\}}{2}\right\}$
3. Nyugta jön a szegmenshez és cwnd ≤ ssthresh: slow start

   cwnd ← cwnd + MSS
4. Nyugta jön a szegmenshez és cwnd > ssthresh: additively increasing

   cwnd ← cwnd + MSS $\frac{\mathrm{MSS}}{cwnd}$

TCP Tahoe

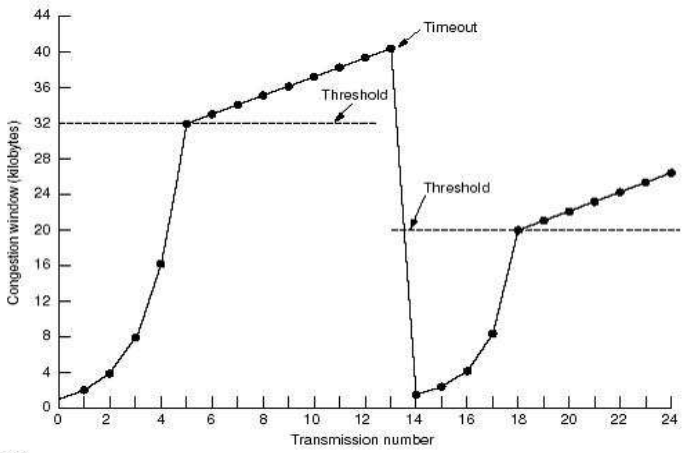


Fig3

pictures from TANENBAUM A. S. *Computer Networks 3rd edition*

Fast Retransmit és Fast Recovery

- TCP Tahoe [Jacobson 1988]:
  - Ha csak egy csomag veszik el, akkor
    - a csomagot újraküldi + a fennmaradó ablakot
    - és egyidejűleg slow start
  - Fast retransmit
    - ha ugyanazon csomaghoz 3 nyugta-duplikátum (azaz 4 azonos nyugta) érkezik (triple duplicate ACK),
    - újraküldi az elveszet csomagot, egyidejűleg slow start
- TCP Reno [Stevens 1994]
  - Fast retransmit után:
    - ssthresh ← max(min(wnd,cwnd)/2, 2 MSS)
    - cwnd ← ssthresh + 3 MMS
  - Fast recovery a fast retransmit után
    - Miden további nyugta után növeli a rátát:
    - cwnd ← cwnd + MMS
  - Congestion avoidance: amikor új adat nyugtája megérkezik (ujraküldés ACK-ja) :
    - cwnd ← ssthresh

Torlódás elkerülési elv: AIMD

- A TCP a „fast recovery” mechanizmussal lényegében a következőképp viselkedik:

  x: csomagok száma per RTT

- Kapcsolatfelépítés:

  x ← 1

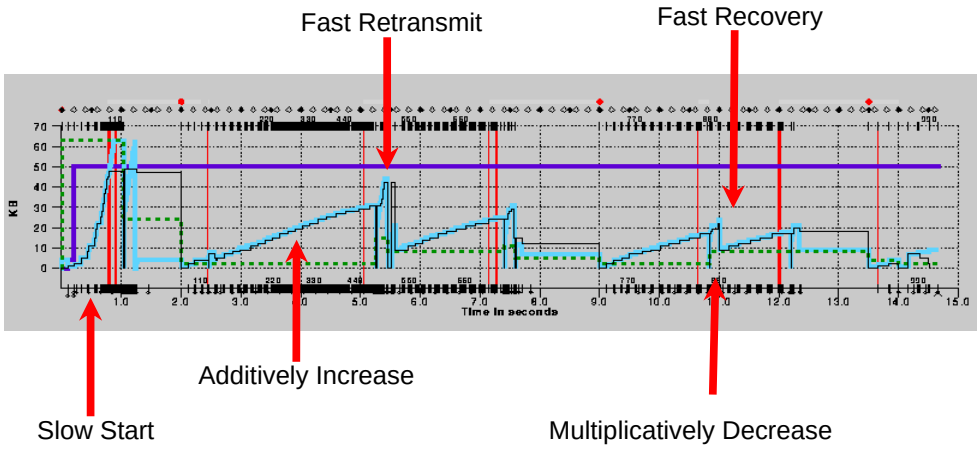
- Csomagvesztésnél, MD: multiplicative decreasing

  x ← x/2

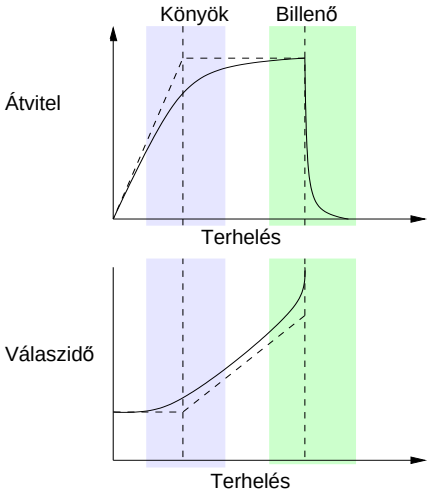
- Nyugtázott szegmenseknél, AD: additive increasing

  x ← x +1

Példa: TCP Reno „in aktion”



Additive Increase Multiplicative Decrease (AIMD): Fairness és Hatékonyság



A hálózati terhelés az átvitellel és a válasziővel kölcsönösen hat egymásra.

- Az átvitel maximális, ha a terhelés a hálózat kapacitását majdnem eléri.
- Ha a terhelés tovább nő, túlcsordulnak a pufferek, csomagok vesznek el, újra kell küldeni, drasztikusan nő a válaszidő. Ezt a toródást congestion-nak nevezzük.
- Ezért a maximális terhelés helyett, ajánlatos a hálózat terhelését a könyök közelében beállítani. Itt a válaszidő csak lassan emelkedik, míg az adatátvitel már a maximum közelében van.
- Egy jó torlódáselkerülési (*congestion avoidance*) stratégia a hálózat terkelését a könyök közlében tartja: hatékonyság. Emellett fontos, hogy minden résztvevőt egyforma rátával szolgáljunk ki: fairness.

AIMD Fairness és Hatékonyság – Egy egyszerű modell

- n résztvevő, forduló-modell
- résztvevő i adatrátája a t-eik fordulóban $x_i(t)$
- Kezedeti adatráták: $x_1(0)$, …, $x_n(0)$
- A visszacsatolás (feedback) forduló t után: y(t) = 0, ha $\sum_{i=1}^{n} x_i(t) \leq K$

  y(t) = 1, ha $\sum_{i=1}^{n} x_i(t) > K$
- Minden résztvevő aktualizálja az adatrátáját a t+1-edik fordulóban:

  $x_i(t+1) = f(x_i(t), y(t))$

  - Increase-stratégia $f_0(x) = f(x,0)$
  - Decrease-stratégia $f_1(x) = f(x,1)$
- Tekintsük a következő lineáris függvényeket :

  $f_0(x) = a_I + b_I x, \qquad f_1(x) = a_D + b_D x$

AIMD Fairness és Hatékonyság– A Modell

- A következő lineáris függvényeket vizsgáljuk:

  $f_0(x) = a_I + b_I x, \qquad f_1(x) = a_D + b_D x$
- Érdekes speciális esetek:
  - MIMD: Multiplicative Increase/Multiplicative Decrease

    $f_0(x) = b_I x, \quad f_1(x) = b_D x, \quad$ ahol $\quad b_I > 1,\ b_D < 1.$
  - AIAD: Additive Increase/Additive Decrease

    $f_0(x) = a_I + x, \quad f_1(x) = a_D + x, \quad$ ahol $\quad a_I > 0,\ a_D < 0.$
  - AIMD: Additive Increase/Multiplicative Decrease

    $f_0(x) = a_I + x, \quad f_1(x) = b_D x, \quad$ ahol $\quad a_I > 0,\ b_D < 1.$

AIMD Fairness és Hatékonyság

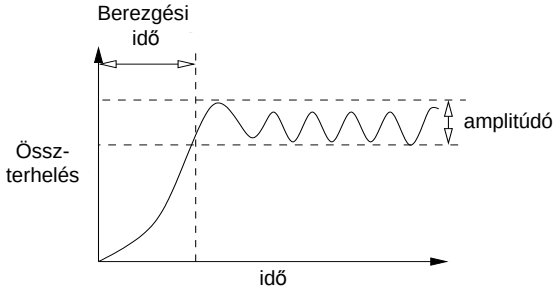
- Hatékonyság
  - Terhelés: $X(t) := \sum_{i=1}^{n} x_i(t)$
  - Mérték: $|X(t) - K|$
- Fairness: x=(x_1, …, x_n) esetén:

  $$F(x) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} (x_i)^2}.$$
  - $1/n \leq F(x) \leq 1$
  - $F(x) = 1 \leftrightarrow$ absolut Fairness
  - skálázástól független
  - Folytonos, differenciálható
  - Ha n közül k fair, a többi 0, akkor F(x) = k/n

Konvergencia

- Konvergencia nem lehetséges
- Legjobb esetben oszcilláció az optimális érték körül
  - Az oszcilláció amplitúdója A

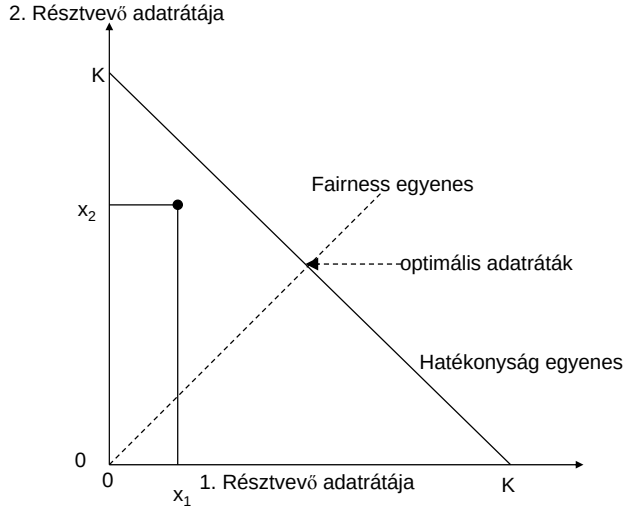
$$A = \inf_{t_0 \geq 0} \sup_{t \geq t_0} |X(t) - K| \ .$$



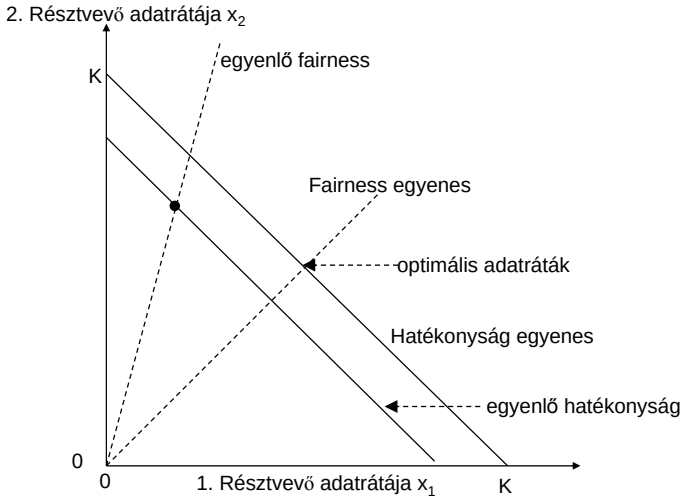
  - Berezgési idő T

$$T = \min\{t_0 \mid \forall t \geq t_0 \ : \ |X(t) - K| \leq A\} \ .$$

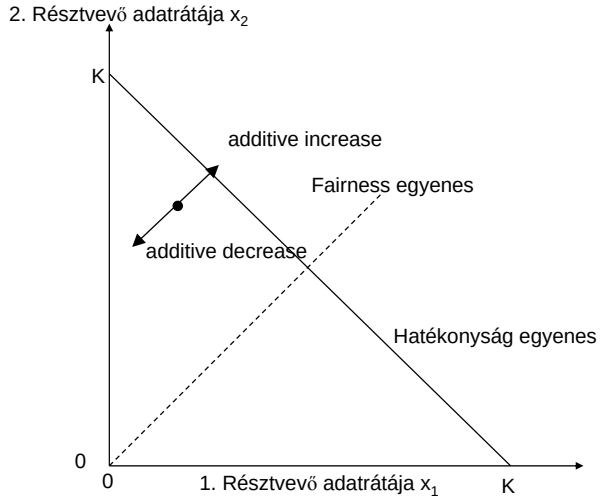
Vektor Ábrázolás (I)



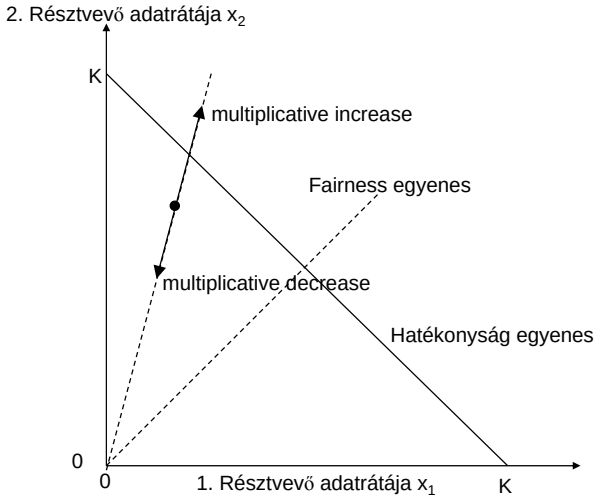
Vektor Ábrázolás (I)



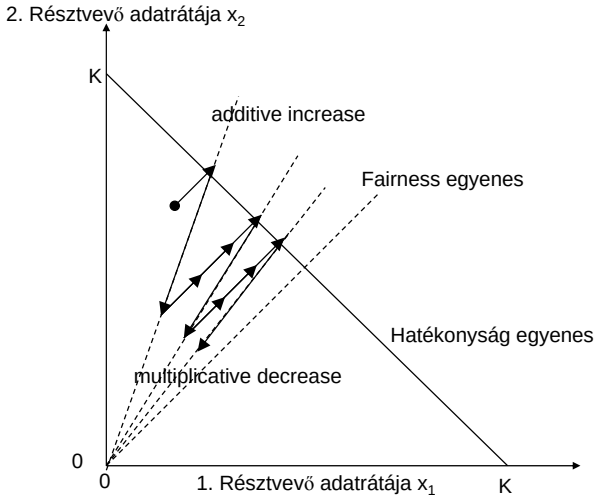
Vektor Ábrázolás (I)



Vektor Ábrázolás (I)



Vektor Ábrázolás (I)



TCP összefoglalás

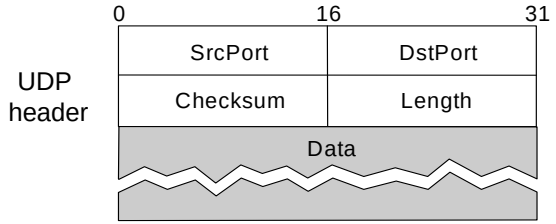
- TCP egy megbízható byte-folyamot hoz létre
  - Hibafelügyelet “Go-Back-N” által
- Congestion control
  - Ablak alapú
  - AIMD, Slow start, *Congestion Threshold*
  - Folyamfelügyelet *Window* által
  - Kapcsolatfelépítés
  - Nagle algoritmusa

TCP fairness & TCP friendliness

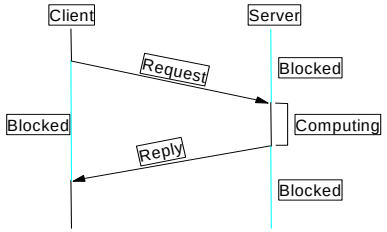
- TCP
  - Dinamikusan reagál a rendelkezésre álló sávszélességre
  - A sávszélesség fair felosztása
    - Idealis esetben: n TCP-kapcsolat mindegyike 1/n részt kap
- TCP más protokollokkal
  - Reakció más szállítói protokollok terhelésétől függ
    - pl. UDP-ben nincs congestion control
  - Más protokollok mindenkor felhasználhatók
  - UDP és más protokoll el tudja nyomni a TCP kapcsolatokat
- Végkövetkeztetés
  - A szállítói protokolloknak TCP-kompatibilisnek kell lenni (TCP friendly)

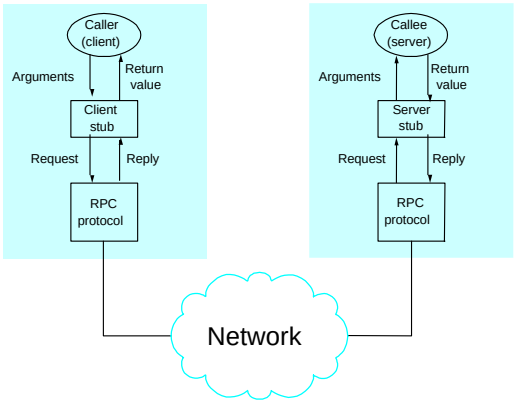
UDP

- User Datagram Protocol (UDP)
  - Egy nem megbízható kapcsolat nélküli szállítói réteg protokoll csomagoknak
- Fő funkció:
  - A hálózati réteg csomagjainak demultiplexálása
- Egyéb funkció (opcionális):
  - Checksum: UDP header + data



Remote procedure call – Struktúra





- Komplexebb interakcióhoz:
  - Egy függvény hívása egy másik számítógépen
- Cél: Transzparens protokoll a hívónak/hívottnak