# Hálózati adattorlódás kivédése mint irányítástechnikai feladat

## Tartalomjegyzék:



## Bevezető

A számítógép hálózatok legfontosabb feladata adatok továbbítása a hálózatba csatolt számítástechnikai eszközök (kommunikációs végpontok - end nodes) között. Ehhez a végpontok között kommunikációs csatornák alakulnak ki, amelyek több hálózati eszközön áthaladnak. Amennyiben nagy számú kommunikációs csatorna ugyanazt a hálózat eszközt (kommunikációs útvonal szakaszt) használja előfordulhat, hogy az útvonal szakaszon torlódás lép fel, a hálózati eszköz nem képes az összes adatot megbízhatóan továbbítani. Ennek elkerülésére a végpontok és a hálózati eszközök olyan kommunikációs módszereket kell alkalmazzanak, amelyek képesek biztosítani azt, hogy az útvonal szakaszok megbízhatóan tudjanak több kommunikációs csatornát is kiszolgálni.

A torlódások kivédésre kidolgozott módszereket (cogestion control) irányítástechnikai megközelítésben tárgyalhatjuk. Ez a megközelítés lehetővé teszi az adattorlódás-kivédési feladat

szisztematikus tárgyalását, általánosítását, lehetőséget ad új kommunikációs módszerek továbbfejlesztésére.

Az alábbiakban néhány irányítástechnikai fogalmat elevenítünk fel, amelyek a torlódás kivédési módszerek tárgyalására alkalmazunk.

*Diszkrét folyamatok*: A számítástechnikai hálózatok elemeinek dinamikus viselkedésének leírására differencia egyenleteket alkalmazhatunk. Ezek a diszkrét modellek a folyamat állapotát (vagy kimenetét) a *k+1*-ik diszkrét időpillanatban (*y*), az előző diszkrét időpillanatokban levő állapotok és a bemenet (*u*) függvényében adják meg:

$$y[k+1] = f(y[k],\ y[k-1],\ ...,\ u[k],\ u[k-1],\ ...) \tag{1}$$

ahol *f* egy ismert függvény a megfelelő érték- és értelmezési tartományokkal.

*Példa diszkrét idejű folyamatra* (kamatozó banki lekötések): A bankszámlán lévő pénzösszeg dinamikus viselkedését az alábbi differencia egyenlet adja meg:

$$y[k+1] = max(0,\ (1+\rho)\, y[k] + u[k]) \tag{2}$$

ahol $y[k]$ a bankszámlán lévő pénzösszeg a *k*-ik diszkrét időpillanatban, $\rho$ a kamat, $u[k]$ a *k*-ik és *k+1*-ik diszkrét időpillanat között a bankszámlára betett (vagy a bankszámláról kivett) pénzösszeg.

*Kapcsoló kimenetű szabályozás (switching control)*: Ennél az irányítási módszernél a beavatkozó jelet különböző módon számítjuk, a folyamat mért kimenetének különböző értéktartományában van.

Tipikus példa a *kétállású szabályozás*. Legyen az irányítási feladat egy diszkrét folyamat kimenetének (*y*) egy elvárt értéken (*r*) tartása. A folyamat bementét az alábbi módon számítja ki a szabályozó:

$$u[k] = \quad U,\ \ ha\ y[k] < r \tag{3}$$

$$u[k] = -U,\ \ ha\ y[k] \geq r$$

ahol *U>0*. Az *y<r* értéktartományában pozitív, az $y[k] \geq r$ értéktartományban negatív a beavatkozó jel.

*Példa kétállású szabályozásra:* Legyen az (1) modellel adott folymat, ahol $\rho = 0.1,\ \ y[0] = 0$. Vizsgáljuk a folyamat irányított viselkedését a (2) beavatkozással. A referencia érték $r = 5$, a szabályozó paraméter $U = 2.$ A folyamat kimenetének változását az alábbi táblázat tartalmazza

| *k* | *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *y* | 0 | 2.2 | 4.42 | 6.862 | 5.54 | 4.09 | 6.499 | 5.139 | 3.653 | 6.01 | 4.61 |
| *u* | 2 | 2 | 2 | -2 | -2 | 2 | -2 | -2 | 2 | -2 | 2 |

Látszik, hogy a kapcsoló kimenetű szabályozással a folyamat kimenete leng a referencia érték körül. A lengés nagysága függ az *U* szabályozó paramétertől.

# A számítógép hálózat, mint irányított folyamat

A csomagkapcsolt számítógép hálózat (packet switching computer network) lehetővé teszi a hálózati kommunikációs csatornák időben megoszott használatát több számítástechnikai eszköz között. Ez a kommunikációs módszer csak a csomag továbbításának idejére sajátítja ki a kommunikációs csatornát két kommunikáló végpont között.

A modellezés során végpontoknak a számítástechnikai eszközökön futó egymással kommunikáló alkalmazásokat tekintjük. Nevezzük az adatot küldő végpontokat *forrás* csomópontoknaknak (*S* - source). Ha egy számítástechnikai eszközön több alkalmazás küld adatot, mindegyiket külön forrásnak tekintjük. Legyenek az adatokat fogadó végpontok az elnyelő végpontok (sinks).

Legyen a legkisebb elküldött adatmennyiség a csomag (packet), mértékegysége pl. byte.

Az adatküldés sebessége (*ráta* - rate) a *T* idő alatt átküldött csomagok száma (*W*)

$$R = W/T \quad (\#/\text{s}) \tag{4}$$

*Hálózati útvonal-szakasz:* Feltételezzük, hogy az adatátvitelre szolgáló hálózati eszközök adattárolási képességekkel rendelkeznek. A hálózati útválasztók, adat kapcsolók nem tudják pillanatszerűen kiszolgálni ugyanazon az útvonal szakason kommunikálni kívánó csatornákat, ezért a csomagokat feldolgozás és továbbküldés előtt tárolják.


1. Ábra: Tároló hálózati útvonalszakasz modellje

A tárolási képességgel rendelkező hálózat útvonalszakasz (link - *L)* dinamikus viselkedésének leírására sorbanállási modellt (queue modell) alkalmazhatunk. Ezt az alábbi differencia egyenlet modellezi:

$$Q[k+1] = max(0, Q[k] + T[k](R[k] - C)) \tag{5}$$

ahol - $Q[k]$ a sor hossza a *k*-ik diszkrét időpillanatban (queue length)

- $T[k]$ - a *k* és *k+1* diszkrét időpillanatban között eltelt idő

- $R[k]$ - az átlagos küldési sebesség a *k* és *k+1* diszkrét időpillanatok között, vagyis $R[k] = W[k]/T[k]$ ahol $W[k]$ a *k* és *k+1* diszkrét időpillanatok között fogadott csomagok száma
- *C>0* - a hálózat útvonalszakasz kiszolgállási sebessége (*#/s*).

Amennyiben T és C konstans, a sor kapacitása (capacity) $T \cdot C\ (\#)$.

Látszik, hogy amennyiben $R[k] \leq C$ a sor hossza zéró körül marad. Ha $R[k] > C$ huzamosabb ideig, a sor hossza növekedni kezd és nagy értékeket vehet fel.

*Drop tail stratégia:* Valós hálózati eszközöknél a sor hossza véges. Legyen a maximális sorhossz $Q_M$. Amennyiben a hálózati útvonal szakaszhoz több csomag érkezik, mint amennyit képes tárolni, a érkező csomagokat eldobjuk, a tárolásukra nincs lehetőség. Ezt drop tail stratégiának hívjuk. A sorbanállási modellünket az alábbi módon bővíthetjük ki:

$$Q[k+1] = max(0, min\ (Q[k] + T[k](R[k] - C),\ Q_M)) \tag{6}$$

*A hálózati útvonal szakaszok csatolása:* Legyen egy számítógép hálózat, amelyben *M* adatforrás ($S_1, S_2, \ldots S_M$) és *N* útvonalszakasz ($L_1, L_2, \ldots L_N$) található.

Hálózat leírásához vezessük be az $A$ útirányító mátrixot (routing matrix), $dim(A) = N \times M$:

$$A_{ij} = 1,\ ha\ L_i - n\ áthaladnak\ az\ S_j - ből\ küldött\ csomagok$$

$$A_{ij} = 0,\ máskülönben$$

A 2. Ábrán látható hálózat útirányító mátrixa:

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

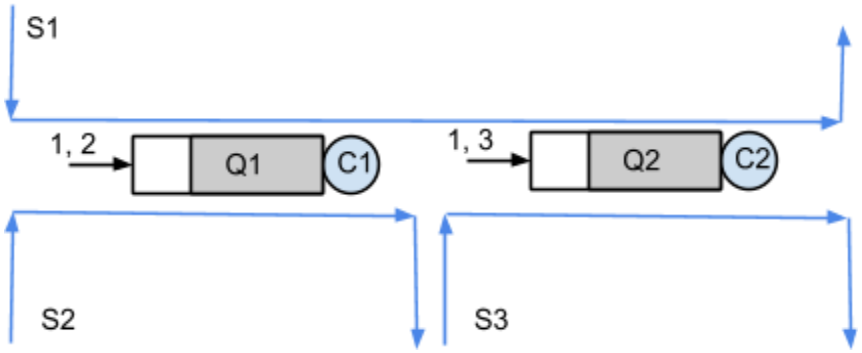
Ha az *j*-ik forrás $R_{jS}$ rátával küld adatokat, akkor az *i*-ik útvonalszakasz bemeneti rátája:

$$R_i = \sum_{j=1}^{M} A_{ij}\ R_{jS} \tag{7}$$

Ha a *j*-ik forrásból induló kommunikációs csatorna egy szakaszán a sor hossza $Q_i$, akkor a teljes úton a kumulált sorhossz:

$$\overline{Q}_j[k] = \sum_{i=1}^{M} A_{ij}\ Q_i[k] \tag{8}$$

$$Q_i[k+1] = max(0, min\ (Q_i[k] + T[k](R_i[k] - C_i),\ Q_{Mi})) \tag{9}$$




2. Ábra: Példa hálózat modellre

*A kommunikációs késleltetés (TT - Transit Time):* Az átlagos időtartam, amíg egy csomag a forrás csomópontból elér az elnyelő csomópontba. Egy kommunikációs útvonalon fellépő kommunikációs késleltetés függ:

- az útvonalon található sorok átlagos kiszolgálási sebességétől ( $\overline{C}$ ),
- a hálózat fizikai rétegében fellépő propagációs idők összegével $T_p$.
- Arányos az útvonalon található sorok kumulált hosszával.

A *j*-ik forrásból induló kommunikációs csatorna késleltetését ( $\delta_j$ ) az alábbi módon modellezhetjük:

$$\delta_j[k] = \overline{Q}_j[k]/\overline{C} + T_p \quad (s) \tag{10}$$

ahol $\overline{C} > 0$ , $T_p$ >0 nem feltétlenül konstans paraméterek.

# Kommunikációs minőségi jellemzők, mint szabályozási követelmények

*1. A sorhossz (queue size):* A torlódás kivédési módszerek legfontosabb célja annak biztosítása, hogy a számítógép hálózatok útvonal szakaszaiban a sorok hossza ne érje el a maximális értéket:

$$Q_i[k] < Q_{Mi}, \ \forall i,k \tag{11}$$

Szintén fontos jellemző a sorok változása. A (9) összefüggésből látszik,hogy amennyiben nem érjük el a maximális vagy zéró sorhosszt:

$$\left|Q_i[k+1] - Q_i[k]\right| = \left|T[k](R_i[k] - C_i)\right| \tag{12}$$

A nagyméretű változásokat úgy lehet elkerülni, ha az $R_i$ küldési sebességek nem, vagy csak lassan változnak.

*2. A kommunikációs késleltetés és változása (communication delay and jitter):* A (10) összefüggés alapján a kommunikációs késleltetés ( $\delta_j$ ) arányos a kommunikációs útvonalon lévő sorhosszak összegével.

Számos alkalmazásnál (multimédia alkalmazások, hang- és mozgókép-átvitel) fontos, hogy a kommunikációs késleltetés változása $\left|\delta_j[k] - \delta_j[k-1]\right|$ (*jitter*) is korlátok között tartsuk.

3. *Elvesztett csomagok száma (packet loss rate):* A drop tail stratégia értelmében amennyiben a sorok elérik a maximális hosszukat, a sorhoz érkező újabb csomagokat a sor eldobja, ezek a csomagok elvesznek.

Legyen egy időegység alatt egy forrásból elküldött csomagok száma *N* és az elvesztett csomagok száma $N_L < N$.

A csomagvesztési minőségi mutatót százalékban adjuk meg:

$$\lambda = \frac{N_L}{N}(100)\ \% \tag{13}$$

4. *Áteresztőképesség (throughput)*: Az *i*-ik útvonal szakasz áteresztő képessége egy *T* időegység alatt szakaszon sikeresen túljutott csomagok száma

$$\tau_i = \frac{N_i - N_{Li}}{T} \tag{14}$$

Az összefüggésben $N_i$ és az $N_{Li}$ az *i*-ik útvonal szakaszhoz megérkezett valamint az *i*-ik útvonalszakaszban elvesztődött csomagok száma.

Ez a minőségi mutató a küldési sebességgel arányos és függ a csomagvesztési mutatótól is.
Ha $R_i = N_i/T$, akkor

$$\tau_i = \left(1 - \frac{\lambda_i}{100}\right) R_i \tag{15}$$

5. *Kihasználtság (link utilization)*: Az *i*-ik útvonal szakasz áteresztő képességének és kiszolgálási sebességének a százalékos aránya:

$$u_i = \frac{\tau_i}{C_i} 100\ \ (\%) \tag{14}$$

Nagy adatforgalom esetében kommunikációs útvonalszakasz hatékonyságát jellemzi.

6. *Fairness*: Annak a mértéke, hogy a különböző kommunikációs csatornák mennyire egyenlően, "igazságosan" jutnak hozzá ugyanannak az útvonalszaksznak az erőforrásaihoz.

*Jain's fairness index:* Legyen az *i*-ik útvonalszakasznál a j-ik forrásból induló kommunikációs csatorna áteresztőképessége $\tau_{ij}$ (*j =1 ...M*). A fairness mértékét az alábbi módon számíthatjuk.

$$J_i = \frac{1}{M} \frac{\left(\sum_{j=1}^{M} \tau_{ij}\right)^2}{\sum_{j=1}^{M} \tau_{ij}^2} \qquad (15)$$

Ha az összes csatornának egyenlő az áteresztő képessége, akkor az index értéke $J_i$ = *1*. Ha csak az egyik csatorna használhatja az útvonalszakaszt (például $\tau_{i1} > 0$ és $\tau_{ij} = 0$, *j = 2...M* ), az index értéke $J_i$ = *1/M*. Minél igazságosabban van elosztva az útvonalszakasz erőforrása a csatornák között, az index értéke annál közelebb van 1-hez.

# A torlódás detektálása és mérése

A torlódás akkor alakulhat ki, hogy ha a kommunikációs csatornán egy vagy több útvonal szakaszban a sorhossz eléri a maximális értéket. Ennek hatására a kommunikációs késleltetés megnő illetve, a drop-tail mechanizmus miatt a csomagok elveszhetnek.

*RTT (Round Trip Time)*: Két kommunikációs csomópont (például a forrás és az elnyelő csomópont) közötti oda-vissza út megtételéhez szükséges idő. Amennyiben a kommunikációs késleltés a csatornán megnő, megnő az RTT értéke is. Ha a csomag elvesztődik, RTT értéke végtelenbe tart.

Az RTT mérése a forrás csomópontnál történik. A csomag küldésekor a forrás elindít egy számlálót (a számláló értéke legyen *count*, a frekvenciája pedig $f_c$). Amint a csomag megérkezik az elnyelőhöz, az visszaküld a forrásnak egy válasz csomagot (*ACK* - Acknowledge). Ha az ACK csomag visszaérkezik a forráshoz, az RTT értékét ki tudjuk számolni. Mivel az RTT érték két mérés (csomagküldés + visszaigazolás) között nagymértékben változhat, aluláteresztő szűrőt alkalmazunk a mérésnél

A torlódás *detektálása*:

- Ha a *count* számláló értéke meghalad egy maximális értéket ($count_M$) a csomagot elveszettnek tekintjük és ezt torlódásnak értelmezzük.

- Ha az RTT értéke nagyobb, mint egy küszöbérték ($RRT_M$), ugyancsak úgy tekintjük, hogy a kommunikációs csatornában torlódás alakult ki.

A torlódás *mérése*: Az RRT érték nagyságsa információ a torlódás mértékéről. Minél nagyobb az RRT, a (10) összefüggés alapján a kumulált sorhossz a kommunikációs csatornán annál nagyobb, annál nagyobb az esély a torlódás kialakulására

Lehetséges pszeudokód torlódás detektálásra, mésésére:

send(package)

count =0

Repeat with period $1/f_c$ {

count++

if (count > $count_M$) {

Congestion = true

Exit }

If (ACK received) {

rtt = count / $f_c$

RTT = $\alpha$ RTT + (1 - $\alpha$) rtt // $\alpha \in (0, 1)$

if (RTT > $RRT_M$) Congestion = true

else Congestion = false

Exit }

} // end repeat

Az ACK válasz csomagot a kommunikációs protokollok, mint például a TCP (Transport Control Protocol) a torlódás detekció mellett a biztonságos adatátvitel biztosítására is alkalmazzák. Ha az ACK nem érkezik meg az előírt időintervallumon belül az adatcsomagot ismételten elküldik.

*ECN (Explicit Congestion Notification):* Az RTT alapú torlódás detektálás implicit módon (csomagvesztés vagy megnövekedett kommunikációs késleltetés) alapján következtet a torlódás megjelenésére. Az explicit torlódás detektálás esetén, a kommunikációs útvonal szakasz értesíti a forrást a torlódásról: ha a sór hossza a kommunikációs eszközben megnő, az útvonal szakaszon áthaladó kommunikációs csomagokba bele írja ezt az információt amely így eljut az elnyelő csomóponthoz. Az elnyelő csomópont az ACK visszaigazoló csomag segítségével az információt visszajuttatja a forráshoz.

# Beavatkozás

A torlódások abból adódnak, hogy a sorok hossza megközelíti és meghaladja a maximális értéket.. A (9) modell alapján látszik, hogy a források küldési sebességét kell megfelelően beállítani a torlódások elkerüléséhez.

*Küldési sebesség alapú beavatkozás* (Rate based) Ha egy új forrás jelenik meg a hálózatban, annak a fix küldési sebességét előírja a hálózat vezérlő. A forrás küldési sebessége felülírható kommunikáció közben is a hálózati vezérlő által. A fix küldési sebességet úgy kell beállítani az összes forrásnál (kiosztani), hogy az összes útvonal szakaszban a torlódás garantáltan ne forduljon elő.

*Hop by hop beavatkozás:* ebben az esetben a kommunikációs csatorna első útvonal szakasza írja elő a forrás küldési sebességét, illetve az *i* -ik útvonal szakasz írja elő az *i-1* -ik útvonal szakasz küldési sebességét. Az útvonal szakasz javasolhatja a küldési sebesség növekedését (credits) illetve csökkentését (backpressure)

*Forrás alapú end-to-end beavatkozás*: A torlódás elkerüléséhez a források a küldési sebességüket kommunikáció közben módosítják. Torlódás detekciót vagy mérést kell alkalmazni a beavatkozó jel meghatározásához.

Ahhoz, hogy a torlódást elkerüljük, nem elégséges, hogy csak néhány forrás csökkentse a küldési sebességét, majdnem az összes forrás megfelelően kell módosítsa a küldési sebességet.

A forrás alapú beavatkozás esetén a küldési sebesség változtatásának legelterjedtebb módja az ablak alapú (*window based*) beavatkozás. Ablaknak nevezzük az egy küldési ciklusban elküldött csomagok számát (*W*). Ezt az értéket módosítjuk minden küldési ciklusban. Az *i*-ik forrás átlagos küldési sebessége $\overline{R}_j = \overline{W}/\overline{T}$ , ahol $\overline{W}$ a küldési ablakok átlagos hossza, $\overline{T}$ a küldési ciklusok átlag periódusa. Amennyiben a következő küldés akkor történik meg, amikor a forrás megkapja az ACK csomagot, az átlagos küldési sebesség

$$\overline{R}_j = \overline{\frac{W}{RTT}} \tag{16}$$

ahol $\overline{RTT}$ válaszidők átlagértéke.

# Torlódás kivédésre alkalmazott end to end szabályozási algoritmusok

A torlódást kivédő szabályozók bemenete a torlódás detektálás, mérés eredménye, a beavatkozó jel a források küldési rátája, például a küldési ablak mérete. Az *end to end* megnevezés arra utal, hogy a kommunikációs útvonal teljes hosszára (a forrástól az elnyelőig) történik a torlódás detekció.

Számos esetben kapcsoló kimenetű szabályozást alkalmazunk. Ez abban az esetben mindenképpen szükséges, ha a torlódást csak detektáljuk, vagyis nincs információnk a torlódás mértékéről, csak a jelenlétéről. Más startégiát alkalmazunk a küldési sebesség módosítására ha torlódást detektálunk és mást, ha nincs torlódás a kommunikációs útvonalon.

A TCP (Transfer Control Protocol) által is alkalmazott AIMD (Additive Increase Multiplicative Decrease) szabályozó algoritmus általános formája (*scalable TCP*):

$$R[k+1] = R[k] + a[k], \quad ha\ nincs\ torlódás \tag{17}$$

$$R[k+1] = m[k]\,R[k], \quad ha\ torlódás$$

A szabályozóban $a[k] \geq 0$ az inkremens torlódás hiányában. A $0 \leq m[k] \leq 1$ értékkel a küldési sebesség csökkenési rátáját állíthatjuk.

A küldési sebesség additív növekedése a sorhosszak növekedéséhez is vezet.

Torlódás jelenlétében a küldési ablakot csökkentjük, aminek a következménye a sorhosszak csökkenése a kommunikációs csatornákon.

*Az AIMD szabályozás hatása a sorhossz változására:* Legyen az (5) modellel megadott kommunikációs útszakasz. Feltételezzükk, hogy $T[k] = T$ konstans, $Q[k] = Q_0$, $R[k] = R_0 = a_0$. $R[k]$-t AIMD szabályozó módosítja.

Amennyiben $R[k] \geq C$

$Q[1] = Q[0] + T(R[0] - C))$

$Q[2] = Q[1] + T(R[1] - C))$

…..

$Q[k] = Q[k-1] + T(R[k-1] - C))$

Összegezve a fenti egyenleteket, kapjuk

$$Q[k] = Q_0 + T(\sum_{i=0}^{k-1} R[i] - k\,C)) \tag{18}$$

Ha nincs torlódás a kommunikációs csatornában az additív növekedési tartományban vagyunk, tehát $\sum_{i=0}^{k-1} R[i] = a_0 + \sum_{i=1}^{k-1} a[i]$ és a sorhossz az alábbi módon számítható:

$$Q[k] = min(0,\ Q_0 + Tk\,(\overline{a} - C)) \tag{19}$$

ahol $\overline{a}$ az $a[i]$ inkremensek számtani középarányosa.

Ha nincs torlódás a kommunikációs csatornában és $m[k] = m$ konstans, akkor $\sum_{i=0}^{k-1} R[i] = R_0 + mR_0 + \ldots + m^{k-1}R_0$ és a sorhossz az alábbi módon számítható:

$$Q[k] = min(0, Q_0 + TkR_0\,\tfrac{m^k-1}{m-1}) \tag{20}$$

A (19) és (20) összefüggések segítségével számítható a sorhossz a *k*-ik diszkrét időpillanatban, és megmutatja az *m*, $\bar{a}$ a szabályozó paraméterek hatását a szabályozás gyorsaságára.

A gyakorlatban alkalmazott AIMD algoritmusok az additív növekedésnek van egy felső korlátja ( $R_M$ )

$$R[k+1] = max(R_M,\ R[k] + a[k])\ , ha\ nincs\ torlódás \qquad (21)$$

$$R[k+1] = m[k]\ R[k],\ ha\ torlódás$$

*Fairness AIMD szabályozással:* Ha egy útvonal szakszon minden kommunikációs csatorna az AIMD a stratégiát követi, és a szabályozók, valamint a torlódás detekció ugyanúgy vannak paraméterezve, a kommunikációs csatornák egyenlő módon jutnak hozzá az útvonal szakasz erőforrásához.

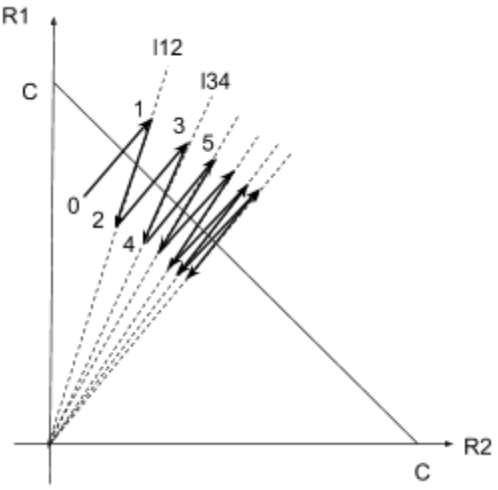
*Példa*: Legyen egy kommunikációs csatorna, amelynek a kiszolgálási sebessége *C* és a maximális sorhossza kicsi ($Q_M$<<). Két AIMD-t alkalmazó kommunikációs csatorna használja az útvonal szakaszt. Az átviteli sebesség $R_1$ és $R_2$.

A kis sorhossz miatt a torlódás detekció az alábbi módon működik:

Ha $R_1 + R_2 \leq C \Rightarrow nincs\ torlódás$

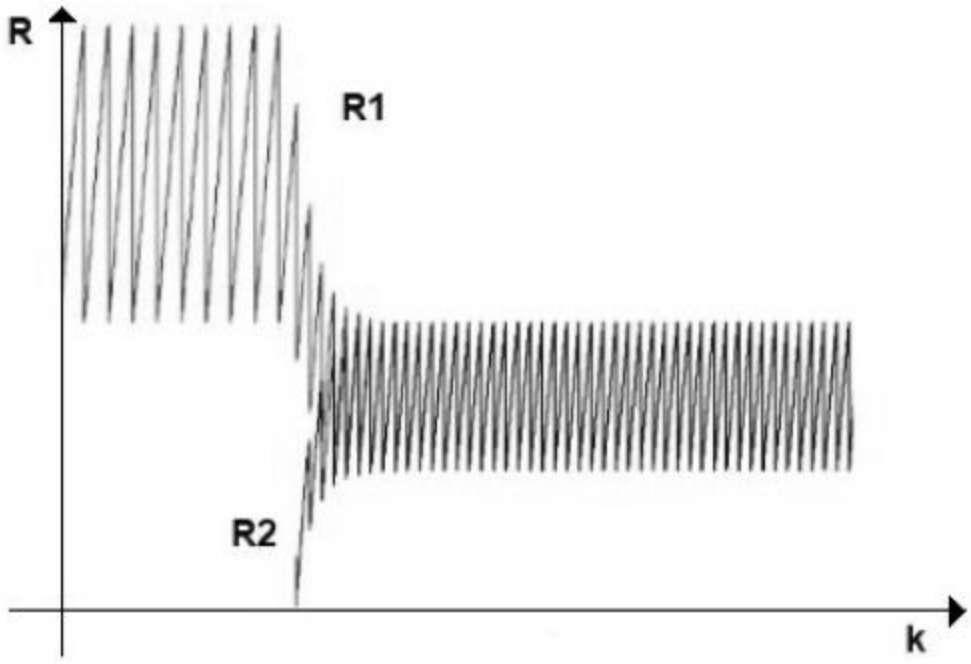
Ha $R_1 + R_2 > C \Rightarrow torlódás$

A 3. Ábrán az AIMD algoritmus viselkedésére láthatunk egy példát. A 0-ik diszkrét időpillanatban, $R_1, R_2 \leq C$ tehát additívan növeljük a rátát. Az 1-es diszkrét időpillanatban $R_1, R_2 > C$, tehát multiplikatívan csökkentjük a rátát. Ezt ismételve az AIMD szabályozás biztosítja azt, hogy az iterációk során az *lij* egyenesek meredeksége $\pi/4$ -hez konvergál, tehát $R_1[k] = R_2[k]$ lesz nagy *k* értékekre.




3. Ábra: AIMD fairness

A 4. Ábra az átviteli sebességek időbeli viselkedését mutatja a két csatornán: abban az esetben, amikor induláskor csak a 1. csatornán van adatátvitel. Az 1. csatorna eléri az útvonalszakasz kapacitását. Amikor a 2. csatorna elindítja az adatátvitelt, az 1 csatorna átvitele csökken, a 2. csatorna átvitele nő és ugyanahhoz az értékhez konvergálnak, közösen kihasználva a kommunikációs útvonal kapacitását. Az is látszik, hogy az adatátviteli sebesség az AIMD szabályozással a diszkrét idő pillanatok között nagyléptékű változásokat mutat.




4. Ábra. Az AIMD algoritmus időtartománybeli dinamikus viselkedése

Az AIMD fairness sérülhet, ha a kommunikációs csatornák külömböző paraméterezésű RRT alapú torlódás detekciót alkalmaznak. Ha az egyik csatorna nagyobb küszöbértékeket alkamaz a tórlódás detekcióra (lásd a $count_M$, $RRT_M$ paramétereket a javasolt RRT alapú torlódásdetekciós algoritmusban) később detektálja csak a torlódást és így több küldési ciklusban növeli a küldési rátáját, mint azok a csatornák, amelyek alacsonyabb küszöb értékeket alkalmaznak.

*Példák AIMD megvalósítására:*

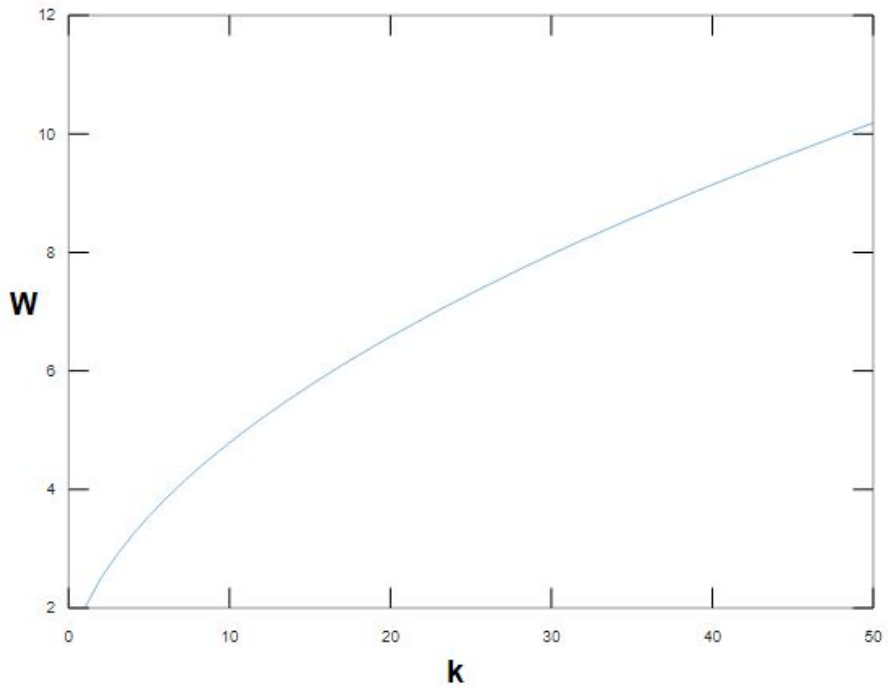
*TCP Reno:* A TCP protokollok a kommunikációs ablakok (*W*) méretét módosítják. A TCP Reno esetén a torlódás detekció csak a csomagvesztést vesszi figyelembe az ACK visszaigazoló csomagok alapján. Amennyiben megadott időn belül nem érkezik visszaigazoló csomag, az elküldött adatcsomagot a torlódás miatt elveszettnek tekintjük.

Az AIMD az alábbi módon van megvalósítva:

$$W[k+1] = W[k] + \frac{1}{W[k]}, \quad ha\ ACK \tag{21}$$

$$W[k+1] = \frac{1}{2} W[k] \quad , \quad másk\text{ülönben}$$

Az alkalmazott additív tag a lineárisnál lassabb növekedést biztosít, lásd az 5. Ábrát.




5. Ábra: A TCP Reno ráta növelése (*W[0]=1* esetén)

*TCP Vegas*: felhasználja a *RTT* méréseket is az additív növekedés fázisban. A *RTT* mérések alapján megbecsüli a torlódás mértékét. Biztosít egy sávot, amikor az ablak mérete nem változik, tehát a küldési ráta nem vagy csak kis mértékben változik. Az algoritmus:

$$If \ ACK \tag{22}$$

$$\text{If } \frac{W[k]}{RTT_{MIN}} - \frac{W[k]}{RTT} < \alpha$$

$$W[k+1] = W[k] + \frac{1}{W[k]}$$

$$\text{Else If } \frac{W[k]}{RTT_{MIN}} - \frac{W[k]}{RTT} > \beta$$

$$W[k+1] = W[k] - \frac{1}{W[k]}$$

Else

$$W[k+1] = W[k]$$

$$Else \quad W[k+1] = \frac{1}{2} W[k]$$

Az algoritmusban $0 < \alpha < \beta$.

Az algoritmus, amikor a válaszcsomag vesztés torlódásra utal, felezi az ablakméretet. Ha nincs csomagvesztés additívan növeli, csökkenti vagy nem módosítja az ablakméretet a mért *RTT* függvényében.

A $\frac{W}{RTT_{MIN}} - \frac{W}{RTT} < \alpha$ feltétel a kommunikációs csatorna becsült kumulált sorhossza és az átlagos küldési sebesség közötti összefüggés. Feltételezzük, hogy *RTT* a (10) összefüggéssel modellezhető, vagyis:

$$RTT = \overline{Q}/\overline{C} + RTT_{MIN} \tag{23}$$

Ennek alapján következik, hogy $\frac{W\,\overline{Q}/\overline{C}}{RTT_{MIN}RTT} < \alpha$. A (16) ráta összefüggés alapján a kapcsolási feltétel ekvivalens az alábbi egyenlőtlenséggel:

$$R < \alpha \frac{RTT_{MIN}\overline{C}}{\overline{Q}} \tag{24}$$

A feltételben a küldési sebesség fordítottan arányos a kumulált sorhosszal.

# AQM (Active Queue Management) algoritmus

A útvonalszakoszon megvalósított sorhossz módosító algoritmusok.

*Buttlefloat szindroma:*

A drop tail stratégia alapján, ha egy útvonal szakaszon a sorhossz eléri a maximális értékét, minden azután érkezett csomag eldobódik.

A torlódás kivédési algoritmusok az eldobott csomagok alapján növelik vagy csökkentik a küldési sebességet.

Ha egy útvonal szakaszt több kommunikációs csatorna használ, az eldobások egyszerre történnek több csatornán amennyiben a drop tail stratégiát alkalmazzuk, tehát a források egyidőben csökkentik a küldési sebességet. Ezért a sor hamar kiürül, így a kommunikációs csatornák forrásai egyidőben növelik a küldési sebességet. Ennek hatására a sorhossz ismét hamar eléri a maximális értéket.

Tehát a drop tail stratégia egyidejű alkalmazása a csomagelvesztés alapú torlódás detektálást alkalmazó AIMD algoritmusokkal a sohosszak oszcillációjához vezet. Ez nagy késleltetés változáshoz (jitter) vezet.

Ennek kivédéséhez azt a stratégiát alkalmazhatjuk, hogy véletlenszerűen egyes csomagokat már azelőtt eldobunk, hogy a sorhossz elérné a maximális értékét. Így egyes csatornák forrásai már

azelőtt elkezdik csökkenteni az átküldési sebességet, hogy a sorhossz elérné a maximális értéket, az *egyidejű* küldési sebesség csökkentés megszűnik.

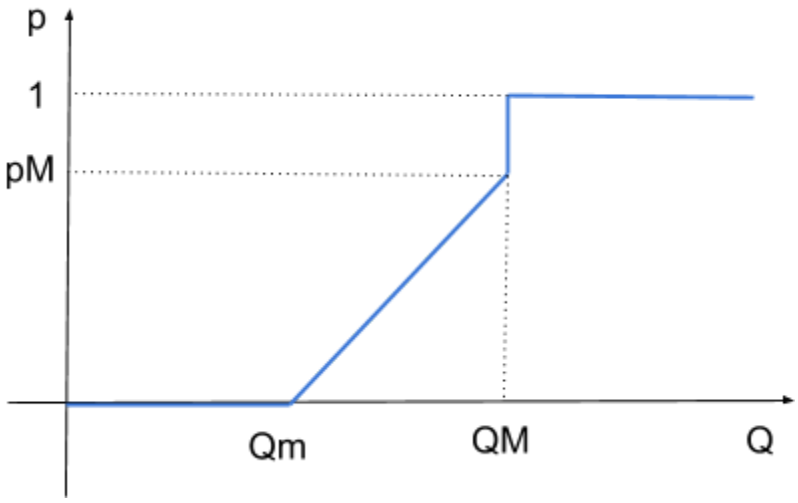
*RED (Random Early Detection) AQM algoritmus:* Az alábbi összefüggés alapján számolja egy adott csomag eldobási valószínűségét (*p*) a sorhossz függvényében:

$$p = 0, \; ha \; Q < Q_m \tag{25}$$

$$p = \frac{p_M}{Q_M - Q_m}(Q - Q_m), \; ha \; Q_m \leq Q < Q_M$$

$$p = 1, \; ha \; Q \geq Q_M$$

ahol $0 < p_M < 1$, $0 < Q_m < Q_M$ útvonalszakasz-függő paraméterek, $Q_M$ a maximális sorhossz.




6. Ábra: A RED algoritmus szemléltetése

Látszik, hogy az AQM algoritmus nem akadályozza meg a csomagvesztést, hanem azt olyan módon végzi el, hogy a sorhosszak az AIMD típusú szabályozással jó dinamikus viszelkedést mutassanak, a buttlefloat szindrómát kivédje.

# Irodalom


1. Ch. Houmkozlis, G. Rovithakis, End-to-End Adaptive Congestion Control in TCP/IP Networks, CRC Press, 2012.
2. J-Y, Boudec, Rate adaptation, Congestion Control and Fairness: A Tutorial, 2019.
3. Steven H. Low, Analytical Methods for Network Congestion Control. M&C, 2017.
4. Mils et al., Study of Proposed Internet Congestion Control Mechanisms, NIST Special Publication, 2010.
5. Andrew S. Tanenbaum, David J. Wetherall, Computer Networks, Pearson, 2012.